FINAL YEAR PROJECT REPORT

APPLICATION OF MACHINE LEARNING ALGORITHMS IN BEARING FAULTS DIAGNOSIS OF INDUCTION MOTOR USING MOTOR CURRENT SIGNATURE ANALYSIS (MCSA),VIBRATIONAL AND ACOUSTIC EMISSION

By

A/C AHSAN ULLAH

Pak/20095003, 95(B) EC

ADVISOR

SQUADRON LEADER M NOMAN RIAZ

CO-ADVISOR

SQUADRON LEADER DR ALI IQBAL

COLLEGE OF AERONAUTICAL ENGINEERING

PAF Academy, Asghar Khan, Risalpur

February 15, 2024

APPLICATION OF MACHINE LEARNING ALGORITHMS IN BEARING FAULTS DIAGNOSIS OF INDUCTION MOTOR USING MOTOR CURRENT SIGNATURE ANALYSIS (MCSA),VIBRATIONAL AND ACOUSTIC EMISSION

By

A/C AHSAN ULLAH

Pak/20095003, 95(B) EC

ADVISOR

SQUADRON LEADER M NOMAN RIAZ

CO-ADVISOR

SQUADRON LEADER DR ALI IQBAL

Report submitted in partial fulfillment of the requirements for the degree of Bachelors of Engineering in Avionics, (BE Avionics)

In

COLLEGE OF AERONAUTICAL ENGINEERING

PAF Academy, Asghar Khan, Risalpur

February 15, 2024

Approval

It is certified that the contents and form of the project entitled "Application of Machine Learning Algorithms In Bearing Faults Diagnosis of Induction Motor Using Motor Current Signature Analysis (MCSA), Vibrational and Acoustic Emission" submitted by Aviation Cadet Ahsan Ullah have been found satisfactory for the requirement of the degree.

Advisor: Sqn Ldr M Noman Riaz

Signature: ________________

Date: ________________

Co-Advisor: Sqn Ldr Dr. Ali Iqbal

Signature: ________________

Date: ________________

Dedication

I wish to take a moment to express my profound gratitude to those who have played pivotal roles in bringing this project to fruition. My heart felt thanks go to my family, whose unwavering support has been my source of strength. I am immensely grateful to my esteemed advisor, for their invaluable guidance and unwavering belief in my potential. I also extend my appreciation to Co-Advisor and Department for their support and guidance. To my dear parents, their selfless sacrifices and boundless encouragement have been the driving force behind my success. This report is dedicated to you and all who have contributed to my journey.

Acknowledgement

I extend my sincerest gratitude to the Almighty Allah, whose boundless blessings and guidance have empowered me to navigate through the challenges of completing this project. My deepest appreciation goes to my parents, whose unwavering love, unwavering support, and constant prayers have been the guiding light in my life. Without their unwavering dedication, this achievement would have remained beyond my grasp. I am profoundly thankful to my advisor,Sqn Ldr M Noman Riaz, for his tireless guidance, invaluable feedback, and steadfast support. His mentorship has played a pivotal role in honing my research skills and fostering my intellectual growth. Additionally, I express heartfelt thanks to my co-advisor,Sqn Ldr Dr. Ali Iqbal and Wg Cdr Hammad for his expert guidance and valuable contributions to my work. I also wish to acknowledge the indispensable assistance provided by Lab Engineer Hammad,Maqsood, Avn Cdt Zaid Ali, and Avn Cdt Aqib, whose support has been invaluable. Gratitude is also due to all my teachers and colleagues whose contributions have enriched my academic and professional journey. Lastly, I am grateful for the unwavering support, encouragement, and motivation from my friends and family members who have stood by me throughout this endeavor.

Abstract

Motor, due to their increased use in many of the application,have become critical in the safety and reliability of engineering system.An induction motor is a type of ac motor in which power is supplied to the rotor by means of electromagnetic induction.A practical machine learning based fault diagnosis method is proposed for induction motors using experimental data.In this Project ,Condition Monitoring based on motor current signature analysis (MCSA), Vibrational and Acoustic emission of bearing faults.For condition monitoring,two identical single phase induction motor is used,one for healthy data acquisition and other one is use for faulty data acquisition.The project has two parts,first one deals with the design of data acquisition setup to acquire baseline (without fault) data under various rpm and loads conditions.In second phase frequently occurring bearing faults (Outer Race,Inner Race, Ball Fault and Compound Fault) are injected and data under same conditions is acquired.The faulty data along with the baseline healthy data is analyzed and extract the time and frequency domain features using MATLAB. Classification algorithms applied for prediction of motor condition(Healthy or faulty).Three classification algorithms, support vector machine (SVM), K-nearest neighbors (KNN), and Ensemble, with 17 different classifiers offered in MATLAB Classification Learner toolbox are used in the study to evaluate the performance and suitability of different classifiers for induction motor fault diagnosis.

Contents

List of Figures

List of Tables

Chapter 1

1 Introduction to the Project

1.1 Project Title

The title of the project is ”Application Of Machine Learning Algorithms In Bearing Faults Diagnosis Of Induction Motor Using Motor Current Signature Analysis (MCSA), Vibrational and Acoustic Emission”.

1.2 Project Overview

This project focus to investigate the use of machine learning methods in the area of induction motor failure diagnosis and the main aiming to improve the accuracy and efficiency of the diagnostic process and fill the Research gape. Our main goal is to utilize machine learning techniques to analyze data acquisition from induction motors Experimental Setup. The development of models identify and categorize various faults by utilizing machine learning techniques. In the proposed study, labeled datasets comprising details regarding various bearing fault types including inner race ,outer race,ball fault and compound bearing fault.There are many condition monitoring techniques which includes thermal monitoring, Acoustic monitoring,chemical monitoring but out of all techniques three are selected for this project (Current signature, Vibrational and Acoustic). These datasets will be used to train machine learning algorithms to discover the patterns and traits unique to each fault type. These algorithms can be trained to recognize an induction motor’s typical working behavior and extracted features.Electric machines, in the form of synchronous and induction generators, produce about 95% of all electric power on Earth (as of early 2020s),[1] and in the form of electric motors consume approximately 60% of all electric power produced.

1.3 Scope of the Project

The scope of applying machine learning algorithms algorithms for diagnosing faults in induction motors, specifically targeting bearing faults and analyzing vibration, Acoustis and motor current signatures. It discuss real-world applications, and consider future directions like incorporating additional sensors or advanced techniques, all within the scope of practical considerations for real-time industrial deployment. This research work aims to unlock the potential of condition monitoring for early, reliable fault detection in induction motors, offering significant benefits for predictive maintenance and industrial efficiency.The project evaluates the performance and suitability of three classification algorithms - Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Ensemble methods - for predicting motor condition (healthy or faulty). The ultimate goal is to assess the effectiveness and reliability of the proposed methodology in practical fault diagnosis scenarios

1.4 Project Milestone

Following milestones have been established for this project:-

- Understanding of project
- Understanding working principles of induction motors
- Establishing experimental bench
- Setup of data acquisition system
- Injecting faults and Acquisition of healthy and faulty data
- Feature Extraction of the data
- Machine Learning algorithms design and prediction
- Development of Graphical User Interface for Result show

1.5 Research Focus and Specific Fault Types

The research primarily focuses on the development and application of machine learning algorithms for fault diagnosis in induction motors, with a specific emphasis on bearing faults(Outer Race, Inner Race, Ball fault and Compound fault). The project aims to investigate the effectiveness of condition monitoring techniques, including Motor Current Signature Analysis (MCSA), vibrational analysis, and acoustic emission analysis, for detecting and diagnosing various types of bearing faults. Specifically, the study targets frequently occurring bearing fault types such as outer race, inner race, ball fault, and compound faults. Through comprehensive experimental data acquisition and analysis, the project seeks to identify distinctive fault signatures and develop robust classification models capable of accurately distinguishing between healthy and faulty motor conditions. Furthermore, the research evaluates the performance and suitability of different classification algorithms, including Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Ensemble methods, to determine the most effective approach for practical fault diagnosis in induction motors.

Chapter 2

2 Literature Review

2.1 Basic Overview

Machine learning algorithms acquired sensor data of induction motors improve failure diagnostics. These algorithms can be trained on labeled datasets that contain details about various fault kinds, allowing them to discover the patterns and traits unique to each fault type. Extracting relevant features from motor signals vibration data identify the presence of faults and train the model on machine learning algorithms.There are three streams of research on fault diagnosis for induction motors [2]:

- 1) signature extraction based approaches: The signature extraction based techniques are finished by using fault signatures in time and/or frequency domain. Current, voltage, vibration, temperature, and acoustic emission can function monitoring signals. Signatures extracted from the recorded tracking alerts are used to discover faults. Motor Current Signature Analysis (MSCA), a well-known spectral analysis approach, is one of the maximum famous techniques for online monitoring induction cars in business environments.

- 2) model-based approaches: The model-based approaches rely on mathematical modeling to predict behaviors of induction motors under fault conditions.Although model-based approaches can provide warnings and estimate incipient faults, its accuracy is largely dependent on explicit motor models, which may not be always available.

- 3) knowledge based approaches; The knowledge-based approaches, do not require a trigger threshold, machine models, motor or load characteristics. Knowledge-based approaches use machine learning techniques for on-line and off-line applications.AI methods have been applied for fault diagnosis in very time-varying and non-linear systems. With continuous advancement of machine learning algorithms, the knowledge-based approach emerges as a promising research direction for induction motor fault diagnosis with great industrial application potential.

2.2 Induction Motor

An induction motor or asynchronous motor is an AC electrical machine that converts electrical energy into mechanical energy. It is a widely used type of motor due to its simplicity, reliability, and cost-effectiveness. An Induction machine is defined as an asynchronous machine is an AC electric motor in which the electric current in the rotor needed to produce torque as shown in Figure.[3]

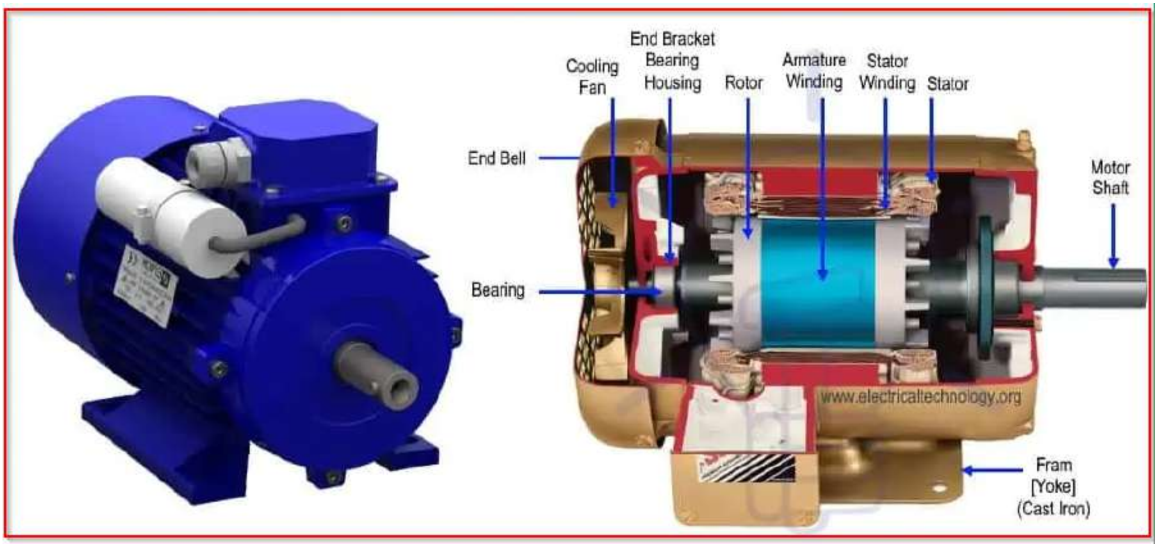


Figure 1: Induction Motor view

The construction of an induction motor consists of two main components:

- Stator
- Rotor

The stator is the stationary part of the motor and contains a series of winding coils that are evenly distributed around the stator's core. These windings are typically made of copper or aluminum and are connected to an alternating current (AC) power supply. The rotor, on the other hand, is the rotating part of the motor. It is separated from the stator by a small air gap. The rotor can be of two types [4]:

- Squirrel Cage

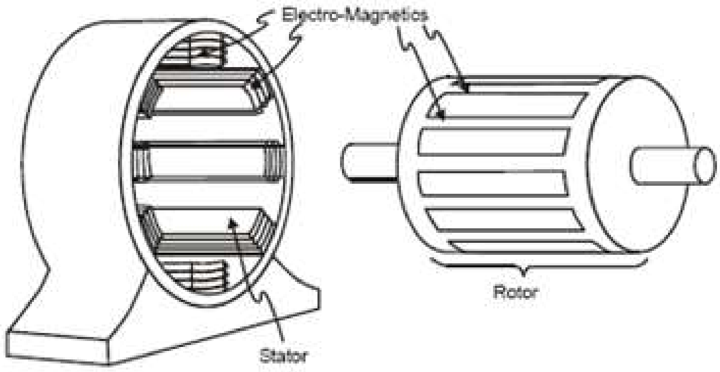


Figure 2: Stator (left) and Rotor (right)

- Wound Rotor

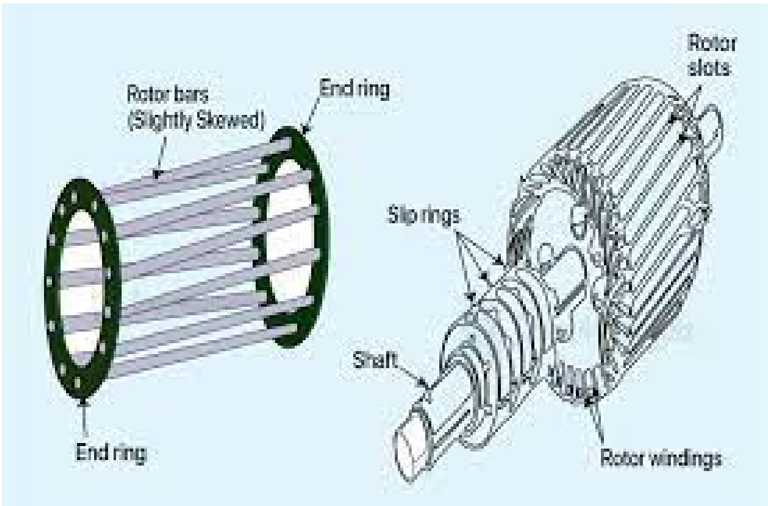


Figure 3: Squirrel cage (left) and wound rotor (right)

2.3 Faults in Induction Motors

A range of faults can occur within induction motor during the course of operation. These faults can lead to a potentially disastrous failures if unnoticed. There are different types of Induction motor faults which is classified into two groups:

- Electrical Faults
- Mechanical Faults

Different faults of induction motors are generally classified as either electrical or mechanical faults. Different types of faults include stator winding faults, rotor bar breakage, misalignment, static and/or dynamic air-gap irregularities and bearing gearbox failures. The most common fault types of these rotating devices have always been related to the Machine shaft or rotor.

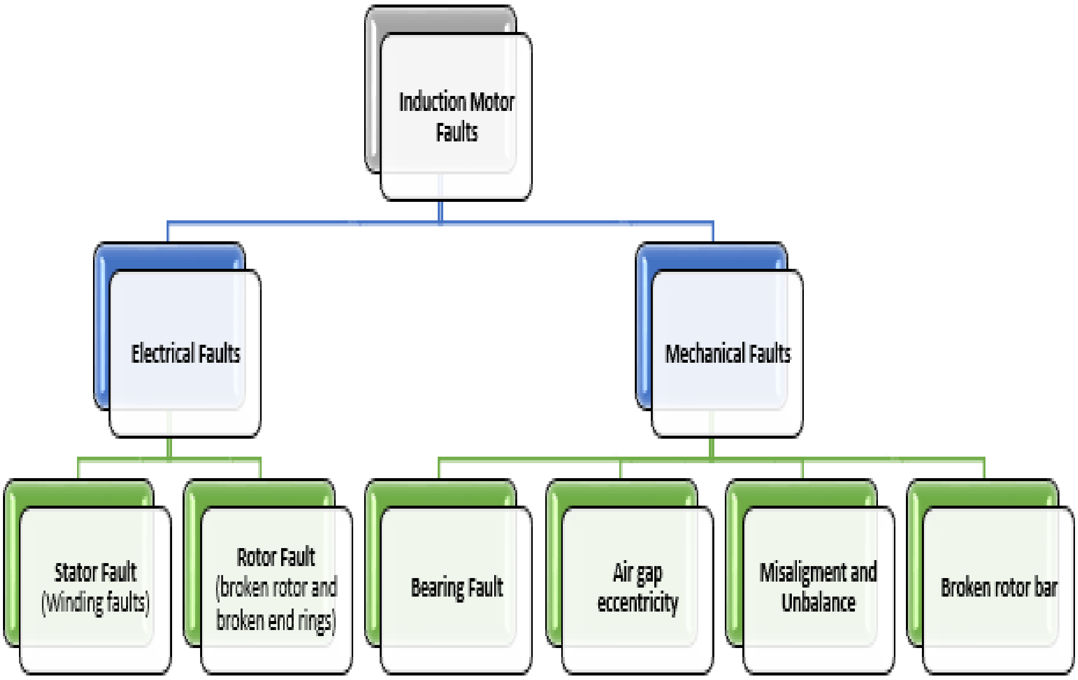


Figure 4: Block diagram classification of Induction Motor Faults

2.4 Mechanical Faults

The mechanical faults occurrence priority is highest in the induction motor. The mechanical faults are classified as bearing fault, Air gap eccentricity fault and broken rotor bar fault.

2.4.1 Air gap eccentricity

Air gap eccentricity is known as a condition that occurs when there is a non-uniform or asymmetric distance between the rotor and stator in the air gap. It is a specific fault that can occur in induction motors. It refers to the deviation of the rotor's center line from the true circular path within the stator's air gap. This fault can cause of vibration and noise. It can be caused by various factors, including manufacturing defects, improper assembly, or mechanical stresses. There are three types of air gap eccentricity:

- Static eccentricity
- Dynamic eccentricity
- Mixed eccentricity

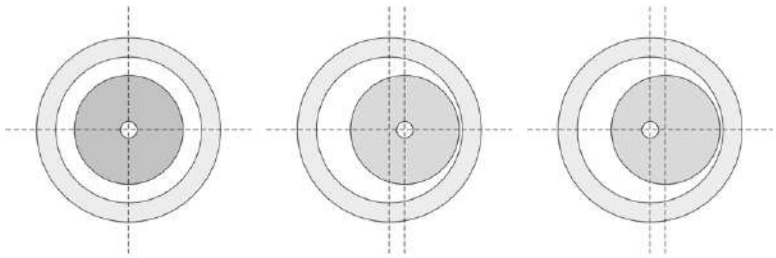


Figure 5: Air gap eccentricity and its three types

2.4.2 Broken rotor bar

During the process in manufacture, non-uniform metallurgical stresses may be built into cage assembly and lead to failure during operation. When thermal stresses imposed upon it during starting of machine. Because of these reasons, rotor bar may be damaged and simultaneously unbalance rotor situation occur.

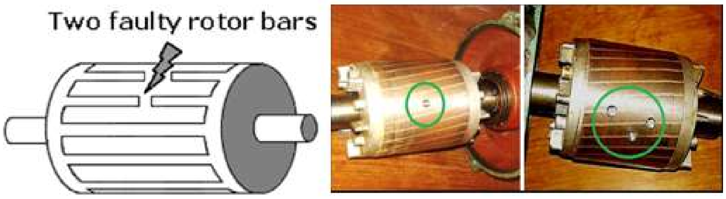


Figure 6: Broken Rotor Bar

2.4.3 Bearing Faults

This fault contains over 40% of all induction machine failures. The majority of electrical machines use ball or rolling element bearings and these are one of the most common causes of failure. These bearing consist of an inner and outer ring with a set of balls or rolling elements placed in raceways rotating inside these rings as shown in Figure. In the Figure Artificial bearing defects are shown which are outer race defect and inner race defect. Since, the rolling elements of a rolling element bearing ride on races. The large race that goes into a bore is called outer race, and the small race that the shaft rides is called inner race. Faults in the inner raceway, outer raceway or rolling elements will produce unique frequency components in the measured machine vibration and other sensor signals. These bearings fault frequencies are functions of the bearing geometry and the running speed. Bearing faults can also cause rotor eccentricity.[14][23]

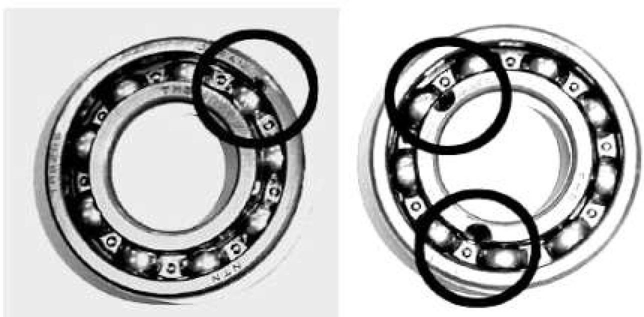


Figure 7: Artificial Bearing Fault (a) Outer Race Fault (Left) (b) Inner Race Fault (Right)

2.5 Electrical Faults

Electrical faults in induction motors refer to faults that occur within the electrical components of the motor, including the stator, rotor, windings, and electrical connections.

These faults can have a significant impact on the motor's performance, efficiency, and reliability. It consists of Stator winding fault, Rotor bar fault, Insulation degradation, Voltage imbalance etc.

2.5.1 Stator Fault

Stator fault occur mainly due to inter turn winding faults caused by insulation breakdown. They are generally known as phase-to-ground or phase-to-phase faults. The stator winding consists of coils of insulated copper wire placed in the stator slots. Stator winding faults are often caused by insulation failure between two adjacent turns in a coil. This is called a turn-to-turn fault or shorted turn as shown in fig 8.[24]

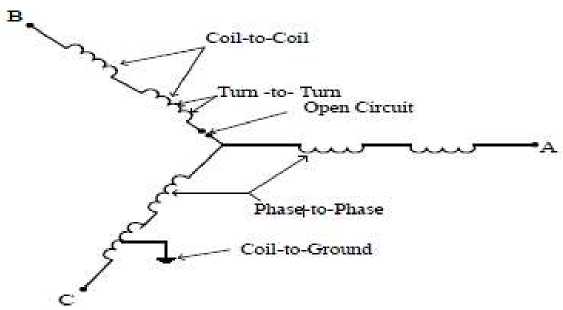


Figure 8: Graphical Representing of Stator Fault

2.5.2 Rotor Fault

Rotor faults occur about almost 10 percentage of total induction motor faults. These faults are caused by rotor winding. The rotor faults are mainly broken rotor bars because of pulsating load and direct on-line starting. It results into fluctuation of speed, torque pulsation, vibration, overheating, arcing in the rotor and damaged rotor laminations.[24]

2.6 Faults by Percentage Occurance

A statistical study conducted jointly by the Electric Power Research Institute (EPRI) and the Institution of Electrical and Electronics Engineers (IEEE) examines the percentage failure components of induction motors. This comprehensive investigation aims to provide the prevalence and distribution of faults in induction motors across various faults condition. By analyzing a large dataset of motor failures, the study aims to identify the most common failure components, quantify their occurrence rates, and understand their impact on motor reliability and performance. Through rigorous statistical analysis and data interpretation, the study seeks to contribute valuable knowledge to the field of motor reliability engineering, informing maintenance strategies, design improvements, and operational practices aimed at reducing downtime and enhancing system reliability. A statistical study of induction motor Faults by Electric Power Research Institute(EPRI) and Institution of Electrical and Electronics Engineers (IEEE) of percentage failure components of induction motor are as shown in Figure.[5]

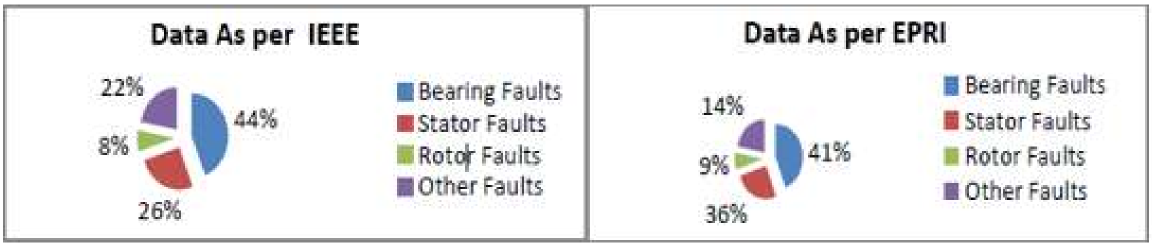


Figure 9: Percentage Component of Induction Motor Failure (Courtesy IEEE and EPRI)

2.7 Importance of Induction Motors in Various Industries

Induction motors are crucial workhorses in a vast array of industries, playing a vital role in powering diverse applications. Their simplicity, robustness, reliability, and energy efficiency make them the most widely used electric motor globally. Here's an overview of their significance in different sectors:

2.7.1 Manufacturing:

- Production lines: Induction motors drive conveyors, pumps, fans, robots, and numerous other machinery crucial for manufacturing processes across diverse sectors (automotive, textile, food processing, etc.).
- Machine tools: Milling, drilling, and cutting machines utilize induction motors for controlled, precise operation in metalworking and fabrication.

- HVAC systems: These motors power compressors, fans, and pumps responsible for heating, ventilation, and air conditioning in factories and industrial buildings.

2.7.2 Energy and Utilities:

- Power generation: Large induction motors are used in generators to convert wind energy, hydroelectric power, and natural gas power into electricity.
- Oil and gas industry: Pumps, compressors, and pipelines within this sector rely on induction motors for efficient operation.
- Water and wastewater treatment plants: These facilities utilize induction motors for pumps, blowers, and other critical equipment.

2.7.3 Agriculture and Mining:

- Irrigation systems: Pumps driven by induction motors provide water for crops and livestock.
- Conveyors and processing equipment: Mines and quarries utilize these motors for transporting and processing raw materials.
- Farm machinery: Grain mills, hay balers, and other agricultural equipment often rely on induction motors.

2.7.4 Consumer Goods and Appliances:

- Washing machines, refrigerators, and air conditioners: These common household appliances often rely on induction motors for their functionality.
- Power tools and lawnmowers: Portable induction motors provide the power for these consumer goods.
- Medical equipment: MRI machines, dialysis equipment, and other medical devices utilize these motors for precise operation.

2.8 Available Dataset

Nowadays, a huge amount of data is collected in industry and science for different purposes; some of it is made public in repositories or on websites. But obtaining the appropriate data in the needed quality and quantity for specialized research often is still challenging, especially, if a wide range of different types of damages or the yet rarely used MCS are the target of interest. This also applies to training data for bearing diagnostics employing ML-algorithms.Some diagnostic data sets for bearing damages are publicly available; the most popular and comprehensive ones are listed below:

- **CWRU:** The Case Western Reserve University (CWRU) Bearing Data Center is a widely recognized repository of bearing vibration data used for research and development in condition monitoring, fault diagnosis, and prognostics of rotating machinery. The ball bearing test data provided by the CWRU Bearing Data Center includes experiments conducted using a 2 hp Reliance Electric motor. Acceleration data was measured at locations near to and remote from the motor bearings. The motor bearings were intentionally seeded with faults using electro-discharge machining (EDM). These faults ranged in diameter from 0.007 inches to 0.040 inches and were introduced separately at the inner raceway, rolling element (ball), and outer raceway of the bearings.Data was collected for normal bearings, single-point drive end and fan end defects. Data was collected at 12,000 samples/second and at 48,000 samples/second for drive end bearing experiments.All fan end bearing data was collected at 12,000 samples/second.Data files are in Matlab format. Each file contains fan and drive end vibration data as well as motor rotational speed.[6]

- **Paderborn Bearing:** Paderborn bearing datasets provided by the Paderborn University Faculty of Mechanical Engineering.The Paderborn Bearing Data Sets, also known as the Paderborn Bearing Data Center, is a collection of datasets hosted by the University of Paderborn, Germany. These datasets are widely used in the field of machine learning, specifically for tasks related to bearing fault detection and diagnosis.Synchronously measured motor currents and vibration signals with high resolution and sampling rate of 26 damaged bearing states and 6 undamaged (healthy) states for reference.Supportive measurement of speed, torque, radial load, and temperature.20 measurements of 4 seconds each for each setting, saved as a MatLab file.Systematic description of the bearing damage by uniform fact sheets and a measuring log, which can be downloaded with the data.[7]

- **Mendeley dataset:** Mendeley Data is a platform where researchers can share and access datasets associated with scientific publications. The "Bearing Vibration Data under Time-varying Rotational Speed Conditions" dataset available on Mendeley Data likely contains vibration signals recorded from bearings under conditions where the rotational speed varies over time.The data contain vibration signals collected from bearings of different health conditions under time-varying rotational speed conditions. There are 36 datasets in total. For each dataset, there are two experimental settings: bearing health condition and varying speed condition. The health conditions of the bearing include (i) healthy, (ii) faulty with an inner race defect, and (iii) faulty with an outer race defect. The operating rotational speed conditions are (i) increasing speed, (ii) decreasing speed, (iii) increasing then decreasing speed, and (iv) decreasing then increasing speed. Therefore, there are 12 different cases for the setting. To ensure the authenticity of the data, 3 trials are collected for each experimental setting which results in 36 datasets in total.[8]

- **Acoustic data:** Data was recorded from four 0,8 kW, 1400 rpm induction motors (the SZJKe 14a), each having the same working parameters, but differing in terms of health state.Motor (SZJKE 14a) Experiments and development of fault detection and diagnostics methods using the same motors.Acoustic signals were measured by three (G.R.A.S. 46 AE) microphones and with a 3D Sound Intensity Micro flown probe, Model USP regular. [9]

- **MAFAULDA:** Machinery Fault Database is composed of 1951 multivariate time-series acquired by sensors on a SpectraQuest's Machinery Fault Simulator (MFS) Alignment-Balance-Vibration (ABVT). The 1951 comprises six different simulated states: normal function, imbalance fault, horizontal and vertical misalignment faults and, inner and outer bearing faults. This section describes the database.Three Industrial IMI Sensors, Model 601A01 accelerometers on the radial, axial and tangencial directions.[10]

Chapter 3

3 Condition Monitoring Techniques

Condition monitoring, also known as Health monitoring, involves continuously evaluating the equipment's health throughout its operational lifespan. Its primary purpose is to detect faults in their early stages, referred to as incipient failure detection, in order to ensure a safe operating environment. By implementing health monitoring systems for induction motors, the electrical condition of the machines can be continuously assessed. This enables the provision of timely warnings for impending failures, allowing for effective scheduling of preventive maintenance and repair activities. Consequently, this approach minimizes downtime and optimizes maintenance schedules, leading to improved operational efficiency.

3.1 Need of Condition Monitoring Techniques

Condition monitoring is defined as the continuous evaluation of the health of the plant and equipment throughout its service life. It is important to be able to detect faults while they are still developing. This is called incipient failure detection [11]. The incipient detection of motor failures also provides a safe operating environment. It is becoming increasingly important to use comprehensive condition monitoring schemes for continuous assessment of the electrical condition of electrical machines. By using the condition monitoring, it is possible to provide adequate warning of imminent failure.It can result in minimum down time and optimum maintenance schedules [12]. Condition monitoring and fault diagnosis scheme allows the machine operator to have the necessary spare parts before the machine is stripped down, thereby reducing outage times. Therefore, effective condition monitoring of electric machines is critical in improving the reliability, safety, and productivity.

3.2 Existing Condition Monitoring Techniques

This research is focused on the condition monitoring and fault diagnosis of electric machines. Fault diagnosis is a determination of a specific fault that has occurred in system. A typical condition monitoring and fault diagnosis process usually consists of four phases as shown in Figure.Condition monitoring has great significance in the business environment due to the following reasons:

- To reducing unplanned downtime and costly repairs, ultimately leading to cost savings in maintenance operations.
- To predict the equipment failure
- To improve equipment and component reliability
- To improve the accuracy in failure prediction

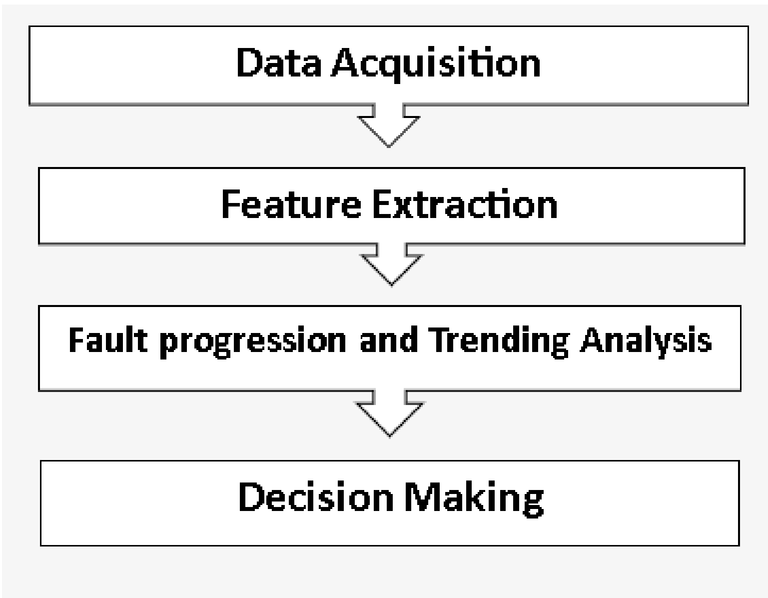


Figure 10: Process for fault diagnosis

Several methods have evolved over time but the most prominent techniques are thermal monitoring, vibrational monitoring, electrical monitoring, noise monitoring etc.

3.2.1 Vibration Monitoring

Vibration analysis is a widely used technique to monitor the mechanical condition of induction motors. It helps detect faults like misalignment, bearing wear, unbalance, and rotor issues. Sensors are strategically placed on the motor to capture vibration signals, which are then processed and analyzed to detect faults such as misalignment, unbalance, bearing wear, mechanical looseness, and rotor faults. By comparing vibration patterns against established baselines or thresholds, deviations can be identified, indicating the presence of abnormalities. Trending the vibration data over time allows for monitoring changes and early fault detection. This information is used to plan appropriate maintenance actions, including corrective and preventive measures, to ensure reliable and efficient motor operation while minimizing downtime and optimizing maintenance schedules.Li et al.[13] carried out vibration monitoring for rolling bearing fault diagnoses. The final diagnoses are made with an artificial NN. The research was conducted with simulated vibration and real measurements. In both cases, the results indicate that a neural network can be an effective tool in the diagnosis of various motor bearing faults through the measurement and interpretation of bearing vibration signatures. Step following in Vibrational monitoring Techniques:

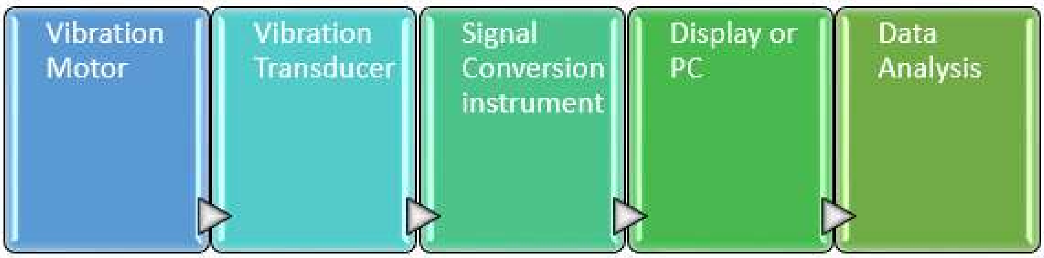


Figure 11: Steps of Vibrational Technique

3.2.2 Temperature Monitoring

Temperatures monitoring as stator windings, bearings, and cooling systems temperature swings can be a sign of problems with insulation deterioration or insufficient cooling. The stator windings, bearings, and cooling systems are just a few of the places on the motor where sensors are strategically positioned to measure temperature. The thermal behavior of the motor is then evaluated using these values, and any variations from typical operating temperatures are found. Increased temperatures may be a sign of problems including deteriorating insulation, failing bearings, insufficient cooling, or overloading. Potential flaws or abnormal circumstances can be detected early on by tracking temperature trends and comparing them to predetermined thresholds or previous data.Electrical machinery can be thermally monitored by estimating parameters or by measuring the motor's overall or local temperature. Excessive heat is produced in the shorted turns by a stator current fault, and this heat extends the fault's severity until it reaches a destructive stage. As a result, some researchers developed thermal model of electric motors.thermal models of electric machines are classified into two categories[15]:

- Finite element Analysis based model
- Lumped parameter thermal models

FEA based models are more accurate, but highly computational intensive. A lumped parameter thermal model is equivalent to thermal network that is composed of thermal resistances, capacitances, and corresponding power losses. The accuracy of model is generally dependent on the number of thermally homogenous bodies used in the model[15][16] .

3.2.3 Acoustic Emission Testing

Acoustic emission testing uses sensors to detect and analyze high-frequency sound waves emitted by the motor. It helps identify issues such as mechanical faults, bearing defects, or arcing. Specialized sensors are used to capture these acoustic emissions, which are then analyzed to assess the health of the motor and detect potential faults. Acoustic emissions can indicate various issues such as mechanical faults, bearing defects, arcing, or structural weaknesses. By analyzing the characteristics of the emitted sound waves, including their intensity, frequency, and duration, faults can be identified and classified. Acoustic emission testing is particularly useful for detecting early signs of deterioration or impending failures that may not be easily observable through other monitoring techniques. It enables proactive maintenance actions to be taken, such as lubrication, repair, or replacement of faulty components, to prevent catastrophic failures and optimize the reliability and performance of the motor.The early fault diagnostic technique based on acoustic signals. The proposed technique was used for the single-phase induction motor. The following states of the motor were analysed:healthy single-phase induction motor, single-phase induction motor with faulty bearing, single-phase induction motor with faulty bearing and shorted coils of auxiliary winding.[17]

3.2.4 Motor Current Signature Analysis (MCSA)

It is a widely used health monitoring technique for induction motors. MCSA involves analyzing the electrical current waveform of the motor to detect abnormalities and diagnose faults. By monitoring the motor's current signature, deviations from normal operating conditions can be identified, indicating the presence of faults such as rotor bar defects, eccentricity, or mechanical problems. MCSA allows for the early detection of developing faults, enabling proactive maintenance actions to be taken before major failures occur. By comparing the current waveform against reference signatures or predefined thresholds, abnormal patterns can be detected, allowing for timely interventions to prevent downtime and optimize maintenance schedules. MCSA is a non-intrusive technique that can be performed while the motor is in operation.Randy R. Schoen et. al.[18] addressed the application of motor current signature analysis for the detection of rolling-element bearing damage in induction machines. The investigates the efficacy of current monitoring for bearing fault detection by correlating the relationship between vibration and current frequencies caused by incipient bearing failures.The bearing failure modes are reviewed and the characteristic bearing frequencies associated with the physical construction of the bearings are defined.M.E.H. Benbouzid and H. Nejjari et. al.[19] stated that preventive maintenance of electric drive systems with induction motors involves monitoring of their operation for detection of abnormal electrical and mechanical conditions that indicate, or may lead to, a failure of the system. Intensive research effort has been for sometime focused on the motor current signature analysis Miletic and Cettolo [20] acknowledged that Motor Current Signature Analysis (MCSA) is one of the widely used diagnostic methods. This method is based on measurement of sidebands in the stator current spectrum. These sidebands are usually located close to the main supply frequency. Frequency converter causes supply frequency to slightly vary in time and, as a result, some additional harmonics in the current spectrum are induced and sidebands are reduced. These harmonics can be easily misinterpreted as the sidebands caused by the rotor faults. In this study, the experimental results of fault diagnosis carried out using standard supply and using frequency converter were compared and presented. All tests were performed on 22 kW induction motor.

Jason R. Stack et. al. [21] introduced the notion of categorizing bearing faults as either single-point defects or generalized roughness. This is important because it divides these faults according to the type of fault signatures they produce rather than the physical location of the fault. The benefit of this categorization is twofold. First, it ensures that the faults categorized as generalized roughness are not overlooked. The majority of bearing condition monitoring schemes in the literature focus on detection of single-point defects. While this is an important class of faults, a comprehensive and robust scheme must be able to detect both generalized roughness and single-point defect bearing faults. Second, grouping faults according to the type of fault signature they produce provides a clearer understanding of how these faults should be detected.

3.3 ISO standard of Condition Monitoring of Machines

ISO (the International Organization for Standardization) is a worldwide federation of national ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee.The document provides guidelines for condition monitoring and diagnostics of machines using parameters such as vibration, temperature, flow rates, contamination, power, and speed typically associated with performance, condition, and quality criteria. The evaluation of machine function and condition may be based on performance, condition or product quality.[22]

- ISO 22096:2007(en) is the ISO standard for condition monitoring and diagnostics of machines using acoustic emission.Acoustic emission is a technique that monitors a component for defects by causing tiny earthquakes in the material. This technique allows large structures and machines to be monitored while in operation with minimal disruption.
- ISO 20958:2013 is an international standard that provides guidance for online condition monitoring and diagnostics of machines using electrical signature analysis. It was introduced on August 15, 2013 and is applicable to three-phase induction motors
- ISO 11342:1998, Mechanical vibration — Methods and criteria for the mechanical balancing of flexible rotors
- ISO 13381-1, Condition monitoring and diagnostics of machines — Prognostics — Part 1: General guidelines
- ISO 20816 (all parts), Mechanical vibration — Measurement and evaluation of machine vibration
- ISO 13373-1, Condition monitoring and diagnostics of machines — Vibration condition monitoring — Part 1: General procedures
- ISO 13379-1 was prepared by Technical Committee ISO/TC 108, Mechanical vibration, shock and condition monitoring, Subcommittee SC 5, Condition monitoring and diagnostics of machines.

The part of ISO 13379 contains general procedures that can be used to determine the condition of a machine relative to a set of baseline parameters. Changes from the baseline values and comparison to alarm criteria are used to indicate anomalous behaviour and to generate alarms: this is usually designated as condition monitoring. Additionally, procedures that identify the cause(s) of the anomalous behaviour are given in order to assist in the determination of the proper corrective action: this is usually designated as diagnostics.

Chapter 4

4 Methodology

The main aim of this project to get an efficient and accurate fault diagnosis system for single phase Induction motor bearing Faults using machine learning algorithms . By this approach we can timely maintenance of our Induction Motor, reduce downtime and enhance the overall reliability of Induction motor.Experiments were conducted on two identical induction motors under healthy,single- and multi-fault conditions. Stator currents and vibration signals and Acoustic signals of the motors were measured simultaneously in each testing.In this paper, 4-pole, 0.5 HP, 208-230V, 1450 rpm rated squirrel-cage induction motor purchased for experiments. Two identical motors named as “Healthy Motor ” and “Faulty Motor ”, which are treated as sister units, are used.Healthy Motor is mainly tested for Healthy conditions, and Faulty Motor for different faults condition. The healthy, single- and compound-fault conditions are applied to the step-by-step methodology that will be adopted for the project is shown below:

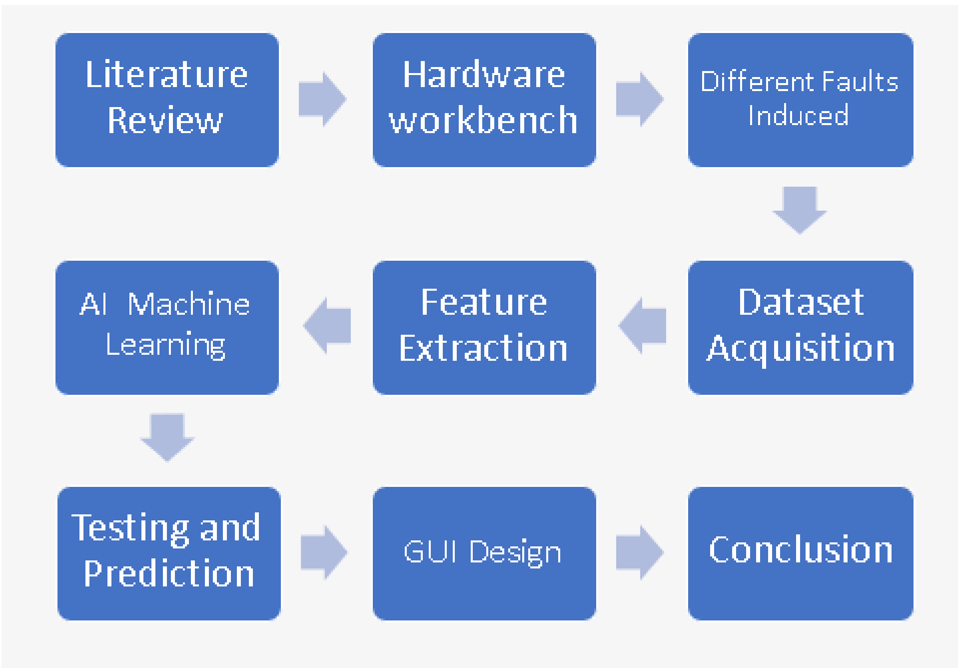


Figure 12: Methodology

4.1 Induction Motor Workbench

The process of acquiring data from induction motors involves setting up appropriate experimental configurations and deploying a robust data acquisition system capable of capturing relevant signals. Induction motor setup typically includes mounting sensors, such as accelerometers for vibration measurement , Current sensors for current signature analysis and Acoustic sensor for Sound data at strategic locations on the motor housing. These sensors are carefully positioned to ensure optimal signal acquisition and coverage of critical motor components, such as bearings.In the experimental test bench (in Figure), an induction motor is connected with the voltage Regulator to a single-phase power supply.Through Voltage Regulator control the rpm of the motor.The vibration is measured by a tri-axial accelerometer (adxl-335), for Current measurement (AC-712) Sensor and for Acoustic Data (KY-037) is used.The accelerometer is mounted on the top of the motor near the face end, vibration at the axial (x-axis), vertical (y- axis) and horizontal (z-axis) directions is measured. The sampling frequency for

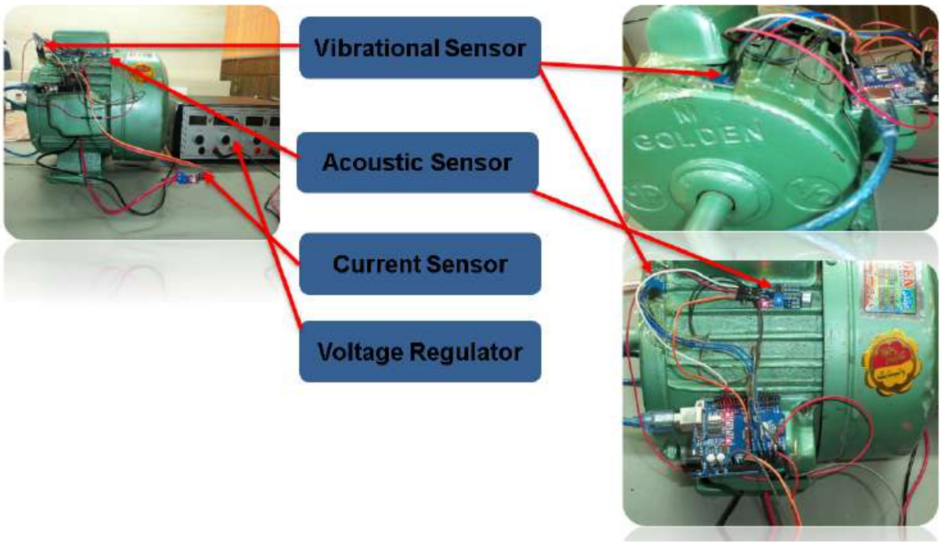


Figure 13: workbench

vibration measurements is 1 kHz. In each test, single phase stator currents (I) , vibration at x-, y-, and z-axis and Sound data during the start-up and steady-state conditions are recorded simultaneously for two minutes. A single- or compound-fault creates unbalance inside the motor, which will be reflected in stator currents ,Acoustic and vibration signals.Induction Motor are tested on different rpm (1450, 1200 and 950) and different Load (100W , 200W , 300W) conditions.

4.2 Different Sensors Used for Different Condition monitoring

A sensor is a device or component that detects and measures physical phenomena or environmental conditions and converts them into electrical signals or other readable forms. Sensors can be a wide range of applications, like industrial systems, automotive systems, consumer electronics, medical devices etc. They enable monitoring, control, and feedback mechanisms by capturing information about various parameters such as temperature, pressure, light intensity, motion etc. There are various sensors that you can use for fault diagnosis in induction motors. Here are some commonly used sensors for condition monitoring :

- Current Sensor
- Acoustic Sensor
- Vibrational Sensor

4.3 Data Acquisition

Dataset covers a wide range of fault and healthy condition including bearing faults,Acoustic faults and Current faults.We have the three sensor for data collection of these Faults. Normalize and Auto-scale the data set for comparison and analysis of fault diagnosis.It enables the extraction of relevant information from the raw signals, facilitating fault diagnosis, condition monitoring, and performance analysis.All data comprises of different load conditions(no load, 100W , 200W and 300W) and Different rpm conditions (1450 rpm, 1200 rpm and 950 rpm).Collecting all dataset and by using Matlab convert the dataset files into Ensemble data (memtable) to ready for the feature Extraction.All the dataset divide into two parts 1) Training data 2) Testing data .Overall data divide into 70 percentage data for Training and 30 percentage data for Testing.

Sensor Used for Dataset				
Vibrational Sensor			Acoustic Sensor	Current Sensor
X-axis	Y-axis	Z-axis	Sound Signal (RMS)	Current Signal (RMS)

Figure 14: Three different dataset

The overall data consists of four load condition and three RPM condition.Four Load condition involve :

- No Load
- 100 Watt
- 200 Watt
- 300 Watt

Three RPM condition involve:

- 1450 rpm
- 1200 rpm
- 950 rpm

The overall table of the data table is include all dataset such as :

Loads	RPMs	Healthy	OUTER RACE	INNER RACE	BALL FAULT	COMPOUND FAULT
NO LOAD	1450	40 Files	40 Files	40 Files	40 Files	40 Files
	1200	40 Files	40 Files	40 Files	40 Files	40 Files
	950	40 Files	40 Files	40 Files	40 Files	40 Files
100W LOAD	1450	40 Files	40 Files	40 Files	40 Files	40 Files
	1200	40 Files	40 Files	40 Files	40 Files	40 Files
	950	40 Files	40 Files	40 Files	40 Files	40 Files
200W LOAD	1450	40 Files	40 Files	40 Files	40 Files	40 Files
	1200	40 Files	40 Files	40 Files	40 Files	40 Files
	950	40 Files	40 Files	40 Files	40 Files	40 Files
300W LOAD	1450	40 Files	40 Files	40 Files	40 Files	40 Files
	1200	40 Files	40 Files	40 Files	40 Files	40 Files
	950	40 Files	40 Files	40 Files	40 Files	40 Files

Figure 15: Overall Dataset Table

4.4 Feature Extraction

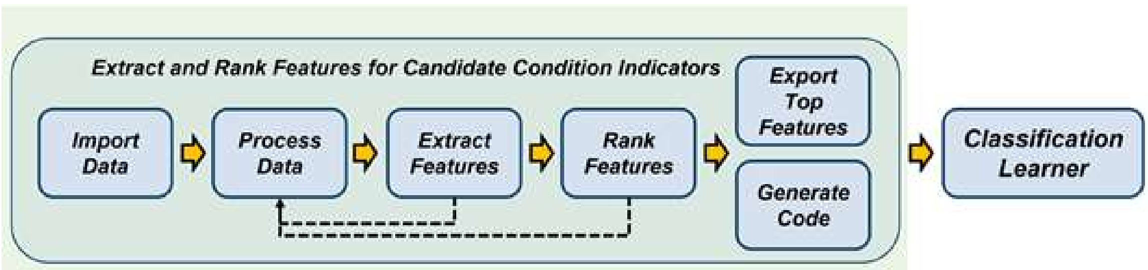


Figure 16: Diagnostic Feature Designer workflow

For Feature Extraction of Time Domain features and frequency Domain features used Diagnostic feature designer tool of the Matlab. Machine learning relies on features extracted from measurement signals [31] The most effective features can ultimately become your condition indicators for fault diagnosis and prognostics.The app operates on ensemble data. Ensemble data contains data measurements from multiple members, such as from multiple similar machines or from a single machine whose data is segmented by time intervals such as days or years. The data can also include condition variables, which describe the fault condition or operating condition of the ensemble member. Often condition variables have defined values known as labels. It divide into three parts:-

Time Domain	**Frequency Domain**	**Time-Frequency Domain**
RMS	Peak amplitude	Spectrogram
Standard deviation	Peak frequency	Short-Time Fourier Transform (STFT)
Shape factor	Number of peaks	Wavelet Transform
Kurtosis	Damping factor	Mel-Frequency Cepstral Coefficients (MFCCs)
Skewness	Band Energy	Wigner-Ville Distribution (WVD)

Table 1: Table of Extracted Features

4.4.1 Time Domain Features:

	1 FaultCode	2 Data_sigstats/ClearanceFactor	3 Data_sigstats/CrestFactor	4 Data_sigstats/ImpulseFactor	5 Data_sigstats/Kurtosis	6 Data_sigstats/Mean	7 Data_sigstats/PeakValue	8 Data_sigstats/RMS	9 Data_sigstats/ShapeFactor	10 Data_sigstats/Skewness	11 Data_sigstats/Std
1	1	4.5213	3.2406	4.0057	6.6732	0.0166	0.0680	0.0210	1.2361	1.5272	0.0129
2	0	1.1055	1.1007	1.1039	2.1263	0.1761	0.1943	0.1766	1.0029	-0.4114	0.0134
3	1	2.0508	1.7561	1.9313	2.0633	0.0516	0.0996	0.0567	1.0998	0.1790	0.0236
4	1	4.5399	3.2553	4.0237	6.6887	0.0166	0.0683	0.0210	1.2360	1.5391	0.0129
5	3	3.3297	2.2238	2.7513	1.9162	0.0184	0.0557	0.0250	1.2372	-0.0679	0.0170
6	3	3.5653	3.2623	3.4429	19.0860	0.0232	0.0833	0.0255	1.0554	-2.6569	0.0106
7	3	3.5432	3.2419	3.4211	19.1561	0.0232	0.0827	0.0255	1.0553	-2.6671	0.0106
8	2	1.4555	1.4360	1.4469	2.1192	0.0587	0.0850	0.0592	1.0090	0.2926	0.0079
9	4	2.4402	1.8884	2.1765	1.7855	0.0209	0.3350	0.1774	1.1526	-0.0184	0.1762
10	2	1.4102	1.3511	1.3891	1.5312	0.0846	0.1175	0.0870	1.0281	-0.1838	0.0202
11	1	2.0729	1.5892	1.8387	1.5271	0.0318	0.0585	0.0368	1.1570	-0.1871	0.0185
12	3	3.3430	2.2238	2.7543	1.9087	0.0184	0.0556	0.0250	1.2385	-0.0652	0.0170
13	0	1.1406	1.1343	1.1385	1.6232	0.1808	0.2059	0.1815	1.0037	-0.2133	0.0156
14	3	3.3692	2.2437	2.7781	1.9156	0.0184	0.0562	0.0250	1.2382	-0.0659	0.0170
15	1	4.4385	3.1818	3.9329	6.6660	0.0165	0.0667	0.0209	1.2361	1.5317	0.0129
16	2	1.5616	1.4393	1.5159	1.5640	0.0519	0.0787	0.0547	1.0533	-0.0944	0.0172
17	0	1.1076	1.1028	1.1059	2.1302	0.1761	0.1947	0.1766	1.0029	-0.4097	0.0134
18	4	2.4402	1.8884	2.1765	1.7855	0.0209	0.3350	0.1774	1.1526	-0.0184	0.1762
19	2	1.4520	1.4325	1.4455	2.1075	0.0586	0.0848	0.0592	1.0090	0.2895	0.0079
20	4	2.4402	1.8884	2.1765	1.7855	0.0209	0.3350	0.1774	1.1526	-0.0184	0.1762
21	2	1.5724	1.4488	1.5262	1.5659	0.0519	0.0792	0.0547	1.0534	-0.0949	0.0172
22	0	1.1096	1.1047	1.1079	2.1189	0.1760	0.1950	0.1765	1.0029	-0.4064	0.0134
23	0	1.1085	1.1037	1.1069	2.1404	0.1760	0.1949	0.1766	1.0029	-0.4104	0.0134
24	2	1.5711	1.4479	1.5250	1.5643	0.0519	0.0792	0.0547	1.0533	-0.0950	0.0172
25	1	2.0844	1.6000	1.8502	1.5278	0.0319	0.0589	0.0368	1.1563	-0.1896	0.0185

Diagnostic Feature Designer app provides a user-friendly interface for extracting time domain features from signals, which can be particularly useful for diagnostic purposes in various applications such as condition monitoring, fault detection, and predictive maintenance. Time domain features are derived directly from the signal's amplitude values over time, without any transformation to the frequency domain.

- **Mean (Average):** The mean is a measure of the central tendency of the signal and represents the average value of all the data points in the signal.
- **Standard Deviation:** The standard deviation quantifies the dispersion or spread of the signal around its mean. It provides a measure of the variability or fluctuation in the signal.
- **Root Mean Square (RMS):** The RMS value is calculated as the square root of the mean of the squared values of the signal. It represents the effective amplitude of the signal and is often used to quantify signal power.
- **Skewness:** Skewness measures the asymmetry of the signal's distribution around its mean. A positive skewness indicates that the tail of the distribution extends more to the right, while a negative skewness indicates a longer tail to the left.

$$x_{skew} = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^3}{\left[\frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2\right]^{3/2}}$$

-

- **Kurtosis:** Kurtosis measures the ”peakedness” or ”tailedness” of the signal’s distribution. A higher kurtosis value indicates a sharper peak and heavier tails, while a lower kurtosis value indicates a flatter distribution.

$$x_{kurt} = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^4}{\left[\frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2\right]^{2}}$$

- **Peak Amplitude:** The peak amplitude is the maximum absolute value of the signal, regardless of its polarity. It represents the maximum excursion of the signal from its mean.
- **Crest Factor:** The crest factor is the ratio of the peak amplitude of the signal to its RMS value. It provides information about the signal’s peak-to-average power ratio and can indicate the presence of transient peaks or spikes.

$$x_{crest} = \frac{x_p}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}x_i^2}}$$

- **Entropy:** Entropy measures the randomness or unpredictability of the signal. Higher entropy values indicate greater randomness, while lower entropy values indicate more predictability or regularity in the signal.

- **Shape factor:**RMS divided by the mean of the absolute value. Shape factor is dependent on the signal shape while being independent of the signal dimensions.

$$x_{SF} = \frac{x_{rms}}{\frac{1}{N}\sum_{i=1}^{N}|x_i|}$$

4.4.2 Frequency Domain Features:

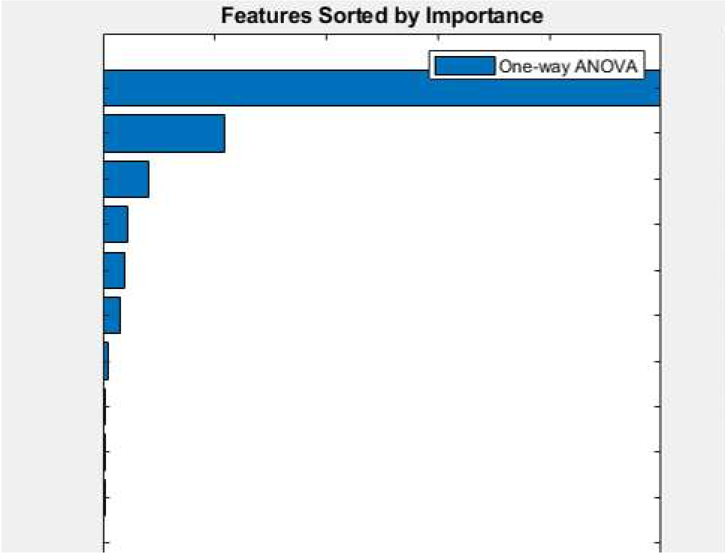
	FaultCode	Data_ps_spec/PeakAmp1	Data_ps_spec/PeakFreq1	Data_ps_spec/BandPower
1	1	1.8832e-04	0.4595	4.8419e-04
2	0	1.8722e-04	0.2268	8.9714e-05
3	1	0.0095	0.4248	0.0192
4	1	1.6467e-04	0.4741	5.5303e-04
5	3	5.5857e-04	0.2457	1.9418e-04
6	3	1.3987e-04	0.4305	0.0022
7	3	1.1965e-04	0.4627	2.2339e-04
8	2	6.5746e-05	0.2707	0.0326
9	4	NaN	NaN	0.0159
10	2	0.3649	0.4550	0.0013
11	1	3.5465e-04	0.2882	0.0011
12	3	3.9895e-04	0.2817	0.0022
13	0	0.0288	0.4206	2.9735e-04
14	3	3.1996e-04	0.2302	0.0014
15	1	1.8416e-04	0.3652	5.2709e-04
16	2	3.2433e-04	0.1459	0.0046
17	0	1.8213e-04	0.2320	8.9220e-05
18	4	NaN	NaN	0.0159
19	2	8.9890e-05	0.2288	3.1362e-05
20	4	NaN	NaN	0.0159
21	2	3.2656e-04	0.1661	0.0049
22	0	3.1642e-04	0.3544	9.0216e-05
23	0	2.1289e-04	0.3889	1.1724e-04
24	2	3.8851e-04	0.1539	0.0531

Frequency domain features are characteristics of a signal or system that can be extracted from its frequency-domain representation. They are often used in signal processing, machine learning, and other fields to analyze and classify signals based on their frequency content.In engineering and statistics, frequency domain is a term used to describe the analysis of mathematical functions or signals with respect to frequency, rather than time. The frequency domain representation of a signal allows you to observe several characteristics of the signal that are either not easy to see, or not visible at all when you look at the signal in the time domain.In the frequency domain, the total average power is the sum of the power of all the frequency components of the signal.The power spectrum is a frequency-domain plot of power per unit Hz vs. frequency. It indicates the relative magnitudes of the frequency components that combine to make up the signal.Three frequency domain feature selected:

- **Peak Amplitude**: In the frequency domain, peak amplitude is a measure of how much a wave or vibration deviates from its central value. Amplitude is plotted on the y-axis, and frequency is plotted on the x-axis.n a sinusoidal waveform, peak amplitude is the maximum positive or negative deviation from the waveform's zero reference level. In the frequency domain, a signal is described by a complex function of frequency. The modulus of the number is the amplitude of that component, and the argument is the relative phase of the wave.he frequency of a sinusoid determines the ”pitch” of the tone, while the amplitude determines the ”loudness”.Peak amplitude = 1.414 x rms amplitude. How does this work? If we took 4 equally-spaced samples of a sine wave with an amplitude of 1, we would get 0, 1, 0, -1. Squaring each we get 0,1,0,1.

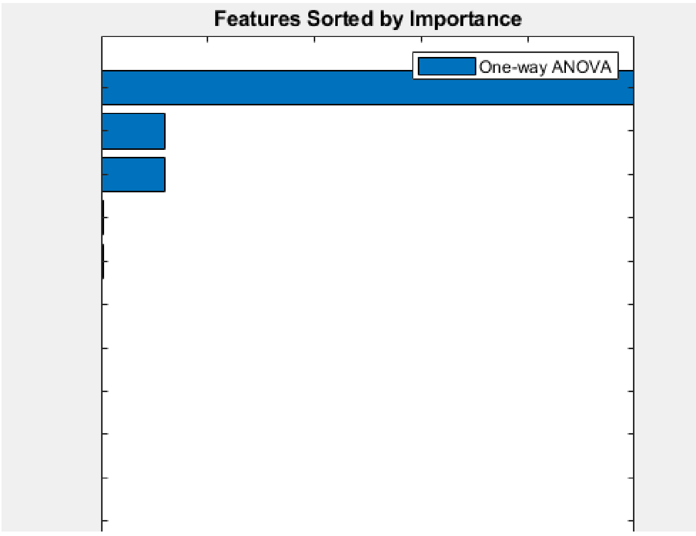
- **PeakFrequency**: Peak frequency is the frequency of waves represented by a peak in the wave spectrum. It can also refer to the frequency of the peak of greatest amplitude within a call.The frequency (period/wavelength) of waves represented by a peak (maximum energy) in the wave spectrum; sometimes known as the dominant frequency.The formula for peak frequency is fmax = k × T, where k is a numerical constant equal to 5.8789232×10^1 Hz/K.
- **Band Power**: Band power is a single number that summarizes the contribution of a given frequency band to the overall power of a signal. It's calculated by using a modified periodogram to determine the average power in a frequency range.The band power of a signal with length is computed as the area beneath the graph of the power spectral density of versus the frequencies.In MATLAB, bandpower is calculated using the formula: p = bandpower(pxx , f , freqrange ,”psd”)

Now “Rank Features,” the app uses one-way ANOVA to calculate ranking scores for all the features. The results of the ANOVA test are displayed on the Screen whereas the bars also shows the normalized scores for different features. For training a machine learning model, we will choose features that have a high ANOVA score and leave out the ones with a much smaller score as these won't contribute to training a model. When we are extracting features to train a model,Everyone find his self trying out different sets of features to see which set works best for classifying fault types.Now we're ready to export the extracted features to the Classification Learner to train a machine learning model.



Feature	One-way ANOVA
Data_sigstats/Std	3.5993e+03
Data_sigstats/RMS	781.0662
Data_sigstats/PeakValue	296.5319
Data_sigstats/Skewness	162.2992
Data_sigstats/Kurtosis	140.2436
Data_sigstats/CrestFactor	113.8369
Data_sigstats/ImpulseFactor	38.1176
Data_sigstats/Mean	18.4225
Data_sigstats/ShapeFactor	16.5455
Data_sigstats/ClearanceFactor	16.0448
Data_sigstats/SINAD	8.4630
Data_sigstats/SNR	8.3355
Data_sigstats/THD	0

Figure 17: Feature one-way Anova



Feature	One-way ANOVA
Data_sigstats/Std	4.4831e+04
Data_sigstats/PeakValue	5.3373e+03
Data_sigstats/RMS	5.2791e+03
Data_sigstats/Mean	158.2272
Data_sigstats/ShapeFactor	132.4781
Data_sigstats/Skewness	52.4824
Data_sigstats/ClearanceFactor	47.8319
Data_sigstats/ImpulseFactor	47.6713
Data_sigstats/Kurtosis	47.6057
Data_sigstats/CrestFactor	47.0868
Data_sigstats/SNR	18.5966
Data_sigstats/SINAD	18.5658
Data_sigstats/THD	0

Figure 18: Other Data Feature one-way Anova

4.5 Algorithm Selection and Modelling

Machine learning (ML) is a discipline of artificial intelligence (AI) that provides machines with the ability to automatically learn from data and past experiences while identifying patterns to make predictions with minimal human intervention. Machine learning derives insightful information from large volumes of data by leveraging algorithms to identify patterns and learn in an iterative process. ML algorithms use computation methods to learn directly from data instead of relying on any predetermined equation that may serve as a model.

4.5.1 How does machine learning work?

Machine learning algorithms are molded on a training dataset to create a model. As new input data is introduced to the trained ML algorithm, it uses the developed model to make a prediction.

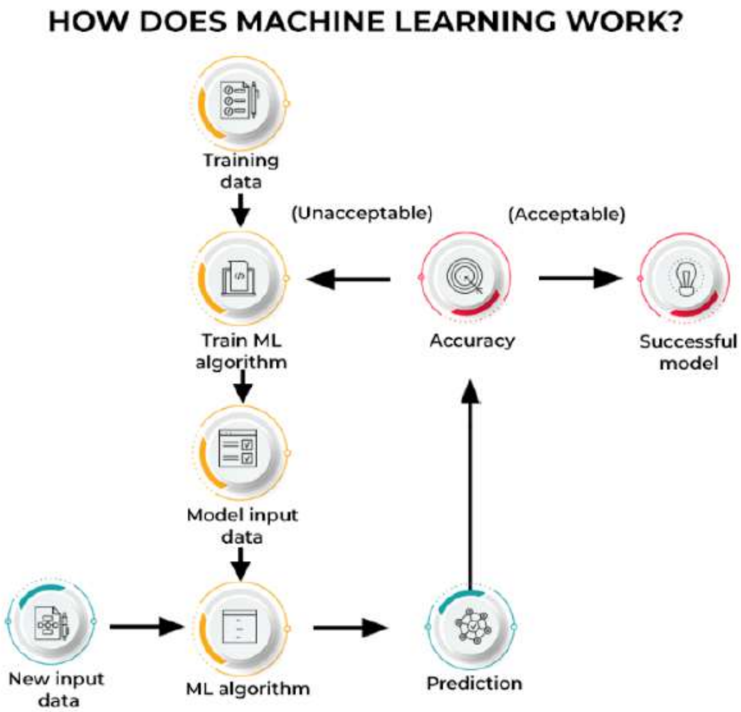


Figure 19: Machine Learning

4.5.2 Types of Machine Learning

Machine learning algorithms can be trained in many ways, with each method having its pros and cons. Based on these methods and ways of learning, machine learning is broadly categorized into four main types: During past two decades machine learning

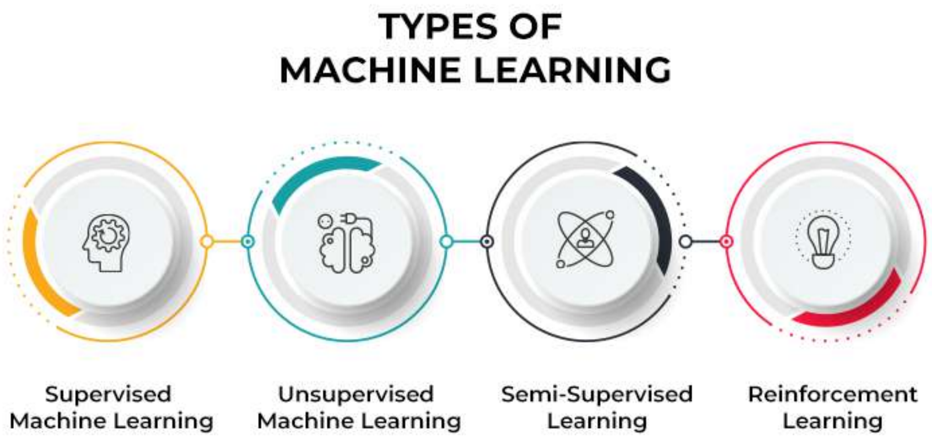


Figure 20: Types of Machine Learning

methods for fault diagnosis of induction motors are the artificial neural network (ANN) or hybrid ANN combined with other techniques [25] One of the most popular hybrid ANN methods is combining ANN with Fuzzy logic, which can provide accurate fault detection with heuristic interpretation [26]Several other machine learning approaches are employed last few years.The immunological principles are applied for induction motor fault detection in [27]. A pattern recognition approach associated with Kalman interpolator/extrapolator is proposed in [28]. An integrated class-imbalanced learning scheme for diagnosing bearing defects is reported in [29]. A sparse deep learning method proposed in [30] can overcome overfitting risk of deep networks.Among machine learning based fault diagnosis methods,stator current is the most widely used signal, either alone or combined with other signals.

4.5.3 Selected Machine Learning Algorithms

Use the machine learning algorithms suitable for fault diagnosis in induction motors. Selection of algorithms for fault diagnosis in induction motors have several factors including the complexity of the problem, the availability and quality of data, computational resources, and the desired performance metrics. There are different Algorithms which include decision trees, random forests, support vector machines, or neural networks. We use and interchange some build in Machine Algorithms available on matlab. a practical machine learning based approach for induction motor fault diagnosis is proposed using experimental data in this project.The Classification Learner app trains models to classify data. Using this app, you can explore supervised machine learning using various classifiers. You can explore your data, select features, specify validation schemes, train models, and assess results. You can perform automated training to search for the best classification model type, including decision trees, discriminant analysis, support vector machines, logistic regression, nearest neighbors, naive Bayes, kernel approximation, ensemble, and neural network classification.A selection of model types appears in the Models pane. When the models finish training, the best percentage Accuracy (Validation) score is outlined in a box.Four Classification Learner Model Selected .

- **Fine Gaussian SVM**
- **Fine KNN**
- **Ensemble(Bagged Tree)**
- **Medium Neural Network**

4.5.4 Fine Gaussian SVM

Support Vector Machine (SVM) and Gaussian SVM (additionally called Gaussian kernel SVM) are each variations of the SVM algorithm, with the important thing difference lying within the preference of kernel feature used to model the selection boundary between training.

- **SVM** In a standard SVM with a linear kernel, the decision boundary is a hyperplane that separates the feature space into two classes.The linear kernel computes the dot product between input feature vectors, resulting in a linear decision boundary.Linear SVM is effective when the data is linearly separable, meaning the classes can be separated by a straight line or hyperplane in the input space.Linear SVM is computationally efficient and easy to interpret, but it may not perform well on data with complex nonlinear relationships.SVM is a commonly used machine learning method for data classification and regression based on statistical learnings and structural risk minimization [32]

- **Fine Gaussian SVM** Gaussian SVM (or Radial Basis Function, RBF, SVM) is a variant of SVM that uses a Gaussian (or radial basis function) kernel to model the decision boundary.The Gaussian kernel computes the similarity between input feature vectors based on their Euclidean distance in the feature space. It assigns higher weights to nearby points and lower weights to distant points.Gaussian SVM is capable of capturing complex nonlinear relationships between features and class labels, making it suitable for data that is not linearly separable.However, Gaussian SVM may be more computationally expensive and prone to overfitting, especially with a large number of training samples or when the kernel bandwidth parameter is not properly tuned.A kernel function converts a nonlinearly separable object into linearly separable by mapping them in a higher dimensional feature space [33]. The common types of kernel functions include linear kernel, polynomial kernel, Gaussian radial basis function (RBF) kernel [34][37].

Specification of the Selected Model is :-

Model	**Fine Gaussian SVM**
KernelParameters	0.7900
Binary Loss	hinge loss= max(0,1-Y*f(x))
LearnerRate	0.41 0.40 0.37 0.39 0.42 0.39 0.41 0.38 0.40 0.38
Score Transform	None
Bias	0.489

Table 2: Specification of Fine Gaussian SVM

4.5.5 Fine KNN

Fine KNN” likely refers to a variant or modification of the k-nearest neighbors (KNN) algorithm that involves fine-tuning certain parameters or hyperparameters to optimize its performance. The term ”fine” in this context typically suggests a process of fine-tuning or optimizing the algorithm for better results.Unlike standard KNN where k can be larger, Fine KNN uses k = 1, meaning it relies on the single nearest neighbor for classification.Choosing centroid value is an iterative process. To generate an initial set of random clusters, the emanated classifier is used.Then it continue to adjust the centroid value until it becomes stable. The stable centroids are used to classify input data by transforming an anonymous dataset into a known one. [35]

Model	**Fine KNN**
NumNeighbors	1
Distance	’euclidean’
DistanceWeight	Equal
BreakTies	Smallest
Bucket Size	50
Type	Classification

Table 3: Specification of Fine KNN

4.5.6 Ensemble(Bagged Tree)

Ensemble learning is a machine learning technique where multiple models are combined to improve the overall performance of the system. Bagged Trees, short for Bootstrap Aggregating Trees or Bootstrap aggregating of decision trees, is a specific type of ensemble method that involves training multiple decision trees on different subsets of the training data and aggregating their predictions.Bagging (Bootstrap Aggregation) is a specific type of ensemble method that uses decision trees as base learners.Bagged Trees represent a valuable ensemble method for bearing fault diagnosis in induction motors. Their ability to improve accuracy, reduce variance, and handle complex data makes them a strong contender in your research toolbox.This flexibility may lead to over fitting,which is overcome in Bagged Trees where each classifier is trained in different partitions and combined through a majority voting. A weaker correlation of error of single classifiers leads to a better prediction accuracy. Therefore, diverse single classifiers are preferred for ensemble [36]

Model	Ensemble
TrainedWeight	1
Combined Weight	30
NumNodes	15
Training each tree	Independently
Each subset	bootstrap sample

Table 4: Specification of Ensemble

4.6 Training and Testing the Models

Train the selected machine learning models using the training data. During training, the models learn the patterns and relationships in the data that distinguish between normal and faulty operating conditions.we have five parameters available for training the model.now I will explain each model training and testing one by one .

4.6.1 SVM Model Training and Testing

Train the SVM model on the dataset and Specify the SVM kernel (e.g., linear, polynomial, Gaussian) and any hyperparameters that need to be tuned (e.g., regularization parameter C, kernel parameters).Evaluate the trained SVM model using the testing dataset. Make predictions on the testing data and compare the predicted labels with the ground truth to compute evaluation metrics such as accuracy and confusion matrix.

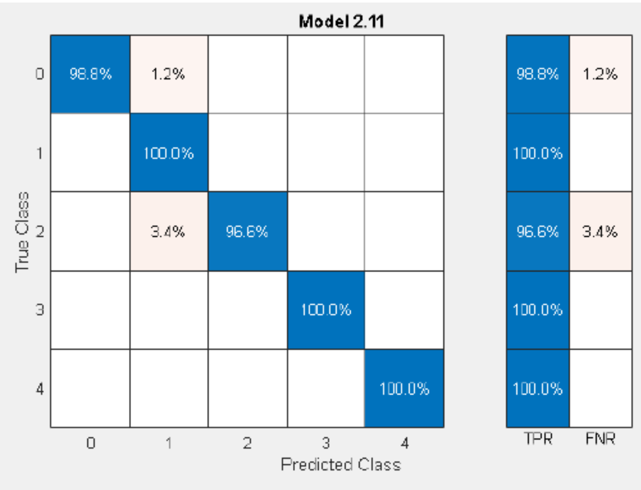


Figure 21: SVM Vibrational Confusion matrix

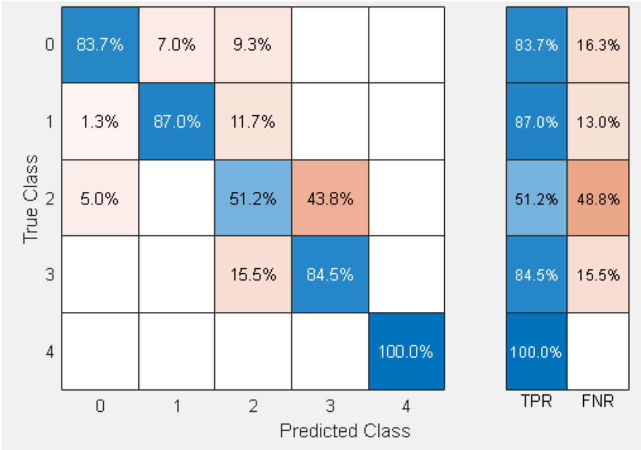


Figure 22: SVM Current Confusion matrix

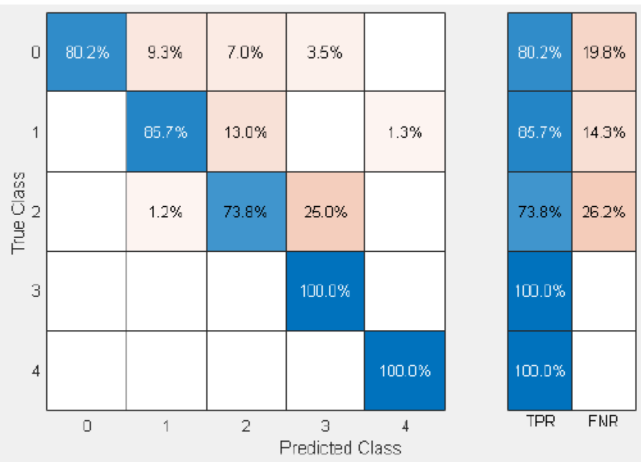


Figure 23: SVM Acoustic Confusion matrix

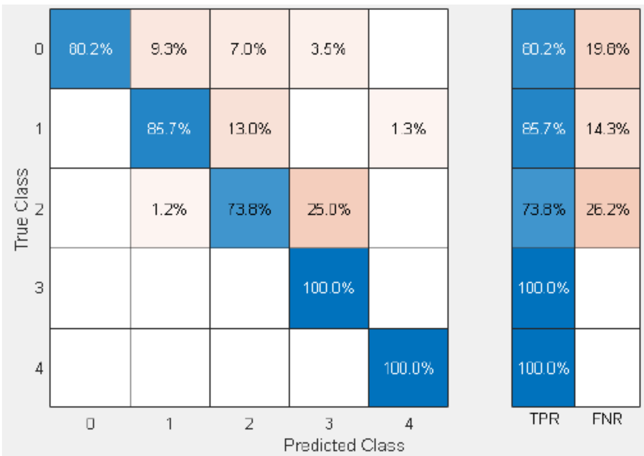


Figure 24: SVM Acoustic Confusion matrix

4.6.2 KNN Model Training and Testing

Train the KNN model using the training dataset. Specify the number of neighbors (k) and any other hyperparameters that need to be tuned.Evaluate the trained KNN model using the testing dataset. Make predictions on the testing data and compare the predicted labels with the ground truth to compute evaluation metrics such as accuracy, precision,and confusion matrix.

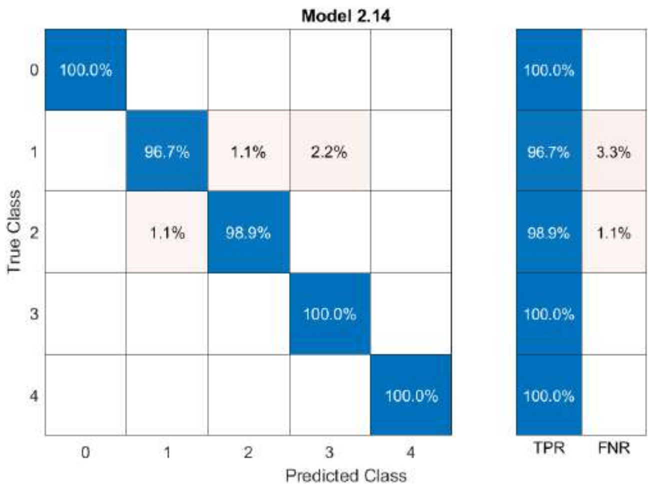


Figure 25: KNN Vibrational Confusion matrix

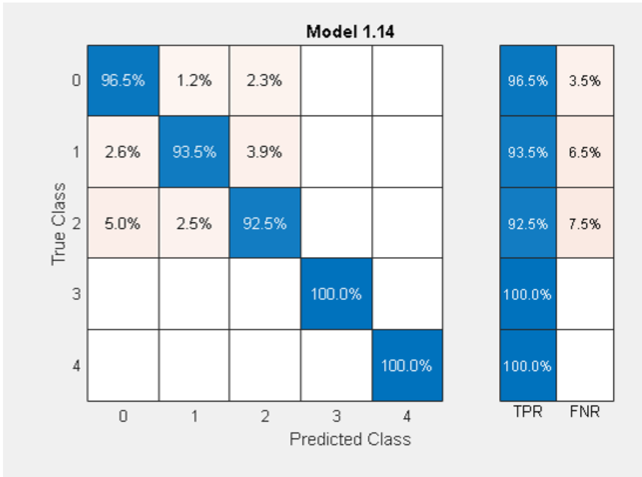


Figure 26: KNN Current Confusion matrix

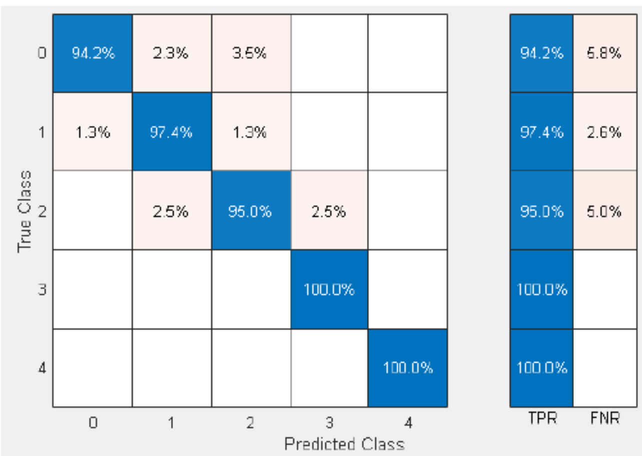


Figure 27: KNN Acoustic Confusion matrix

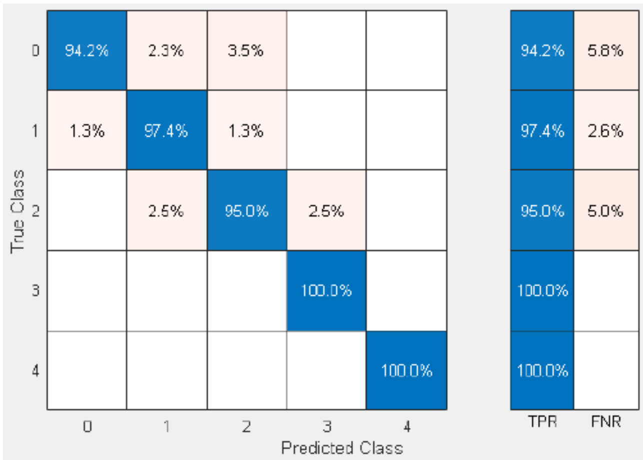


Figure 28: KNN Acoustic Confusion matrix

4.6.3 Ensemble Model Training and Testing

Train the ensemble model using the training dataset. For bagging, train multiple base learners on different subsets of the training data and aggregate their predictions. For boosting, train base learners sequentially, with each subsequent learner focusing on the errors made by the previous ones. For stacking, train multiple base learners and combine their predictions using a meta-learner.valuate the trained ensemble model using the testing dataset. Make predictions on the testing data and compare the predicted labels.

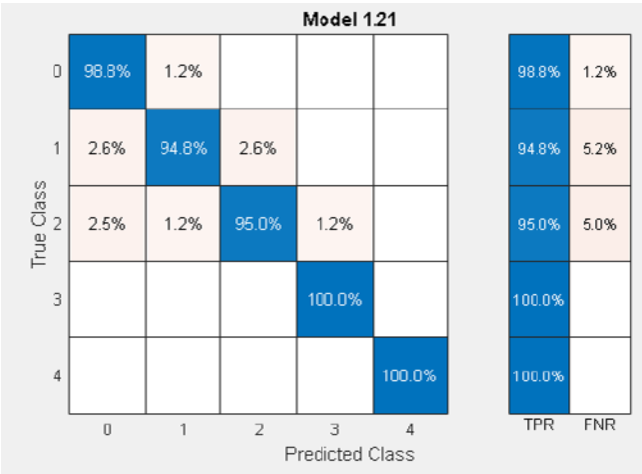


Figure 29: Ensemble Vibrational Confusion matrix

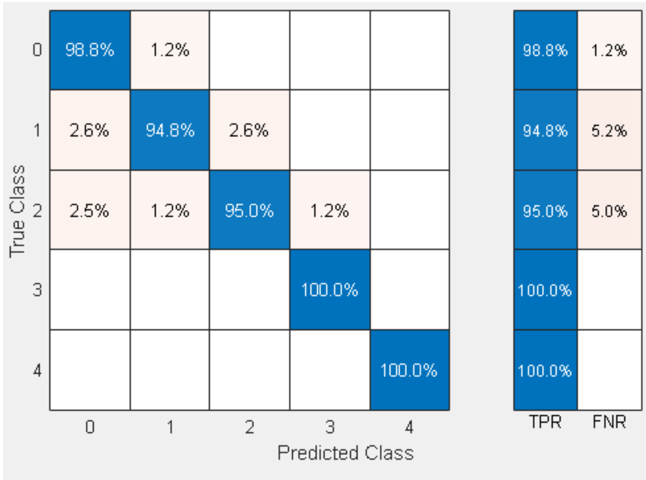


Figure 30: Ensemble Current Confusion matrix

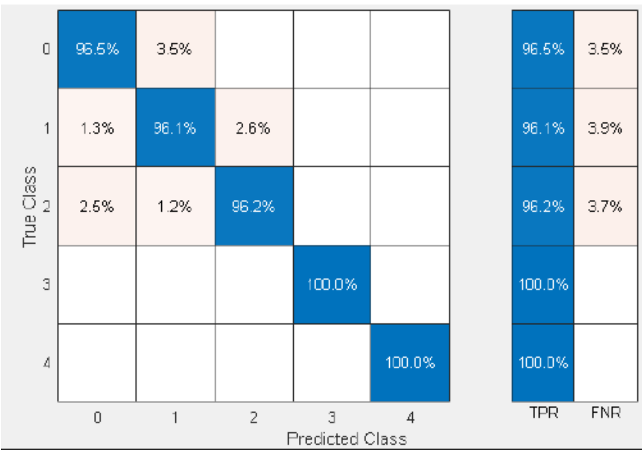


Figure 31: Ensemble Acoustic Confusion matrix

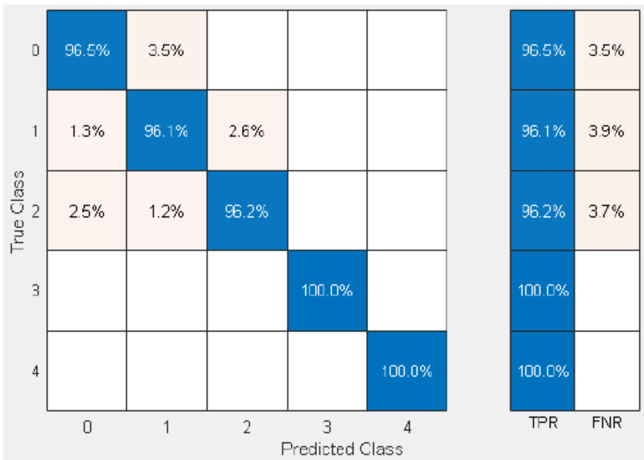


Figure 32: Ensemble Acoustic Confusion matrix

4.6.4 Neural Network Model Training and Testing

Train the neural network model using the training dataset. This involves feeding the training data through the network, computing the loss function (e.g., cross-entropy loss for classification tasks), and updating the model parameters using optimization algorithms.Make predictions on the testing data and compare the predicted labels with the ground truth to compute evaluation metrics such as accuracy.Then tested the performance of your trained neural network model on the unseen test set.

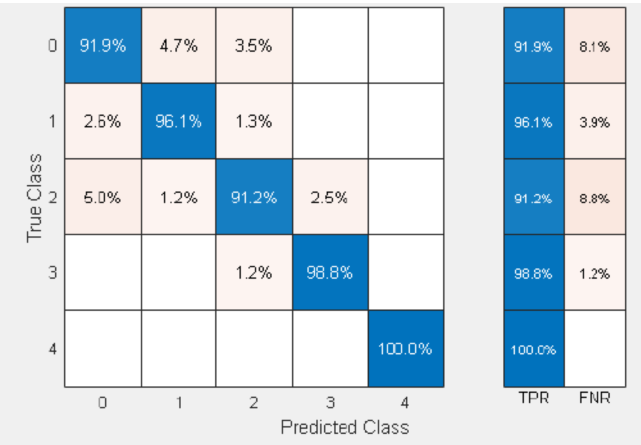


Figure 33: Neural Network Vibrational Confusion matrix

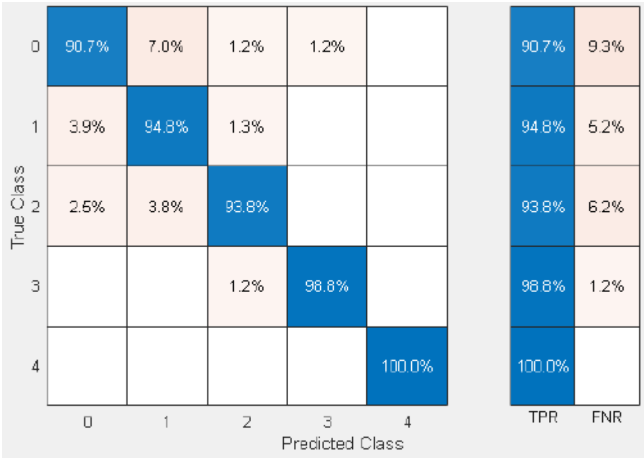


Figure 34: Neural Network Current Confusion matrix

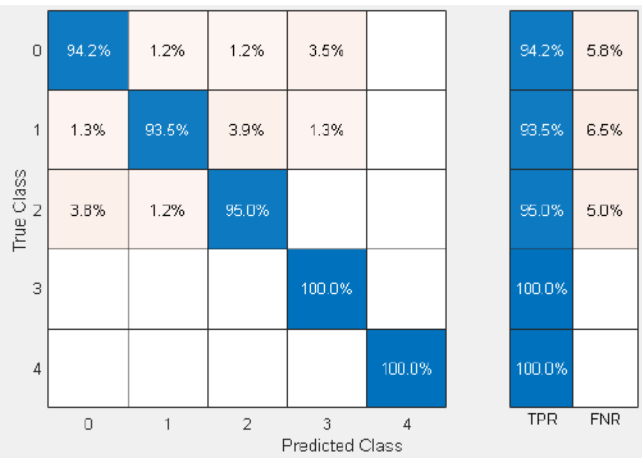


Figure 35: Neural Network Acoustic Confusion matrix

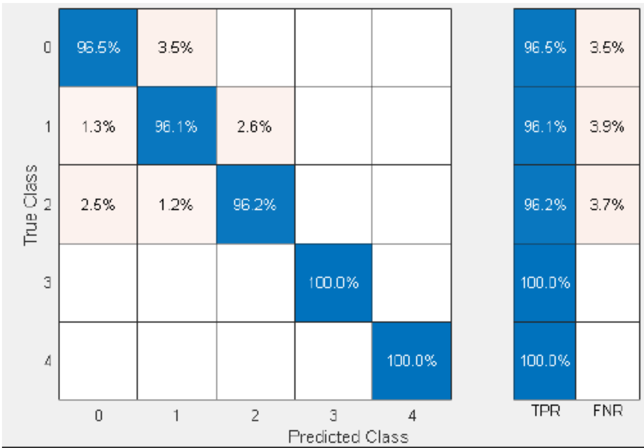


Figure 36: Neural Network Acoustic Confusion matrix

4.6.5 ROC and AUC of the model

Receiver Operating Characteristic (ROC) curve and Area Under the Curve (AUC) are important metrics used to evaluate the performance of classification models, including those used for fault diagnosis of induction motors using machine learning techniques.

The ROC curve is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. It is created by plotting the true positive rate (sensitivity) against the false positive rate (1-specificity) at various threshold settings. A higher true positive rate and a lower false positive rate indicate better performance of the classifier.

The AUC represents the area under the ROC curve and provides a single scalar value that summarizes the performance of the classifier across all possible threshold settings. The AUC value ranges from 0 to 1, where a model with an AUC of 1 indicates perfect discrimination (i.e., the model achieves a true positive rate of 1 and a false positive rate of 0 across all threshold settings), while a model with an AUC of 0.5 suggests random classification (no better than chance).The ROC curve is a graphical representation of the confusion matrix. It summarizes the overall performance of a classifier over all possible thresholds, and the area under the curve (AUC) gives an insight about how confidently the classification is done.

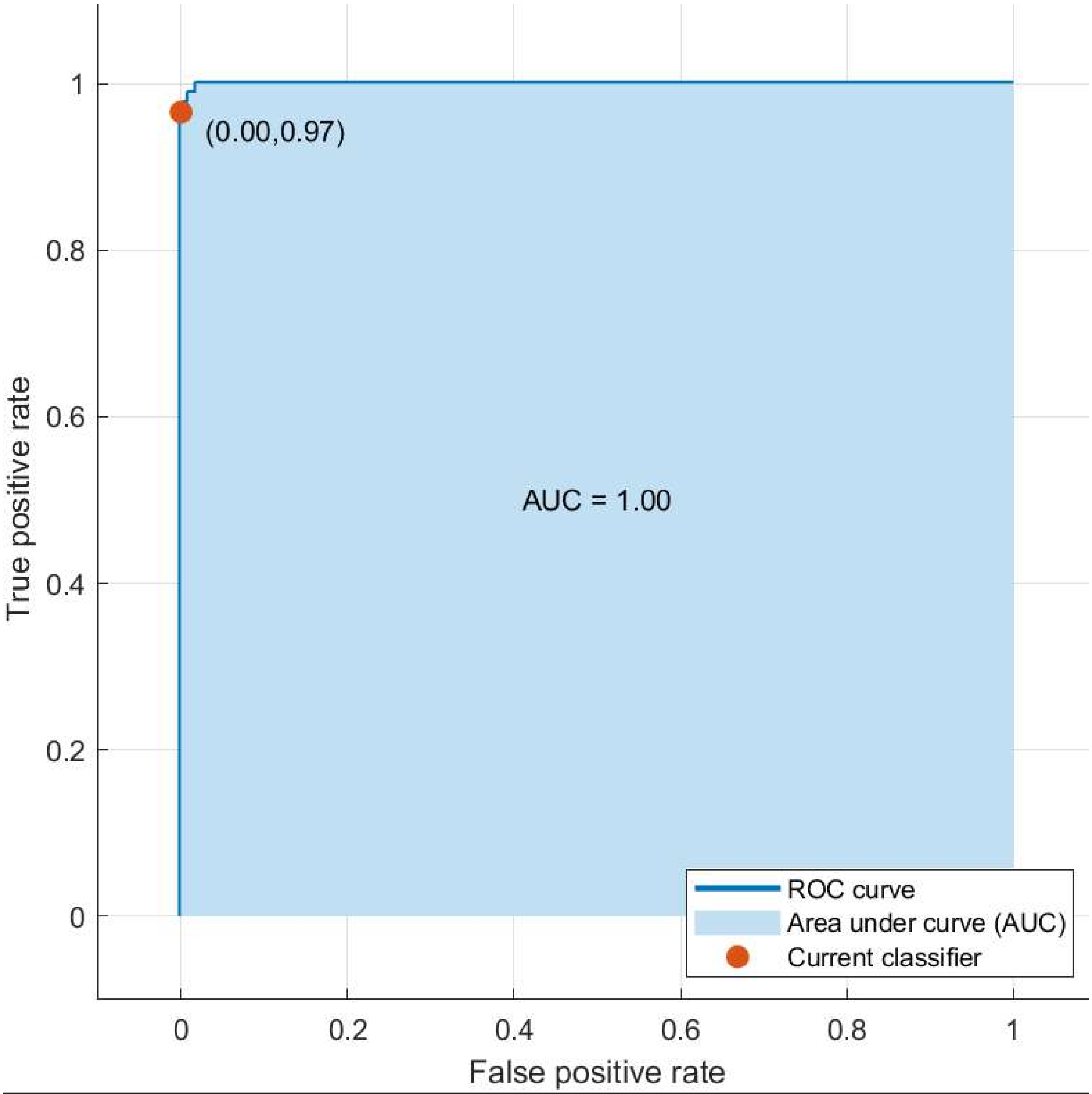


Figure 37: ROC and AUC of the data

Chapter 5

5 Experimental Setup

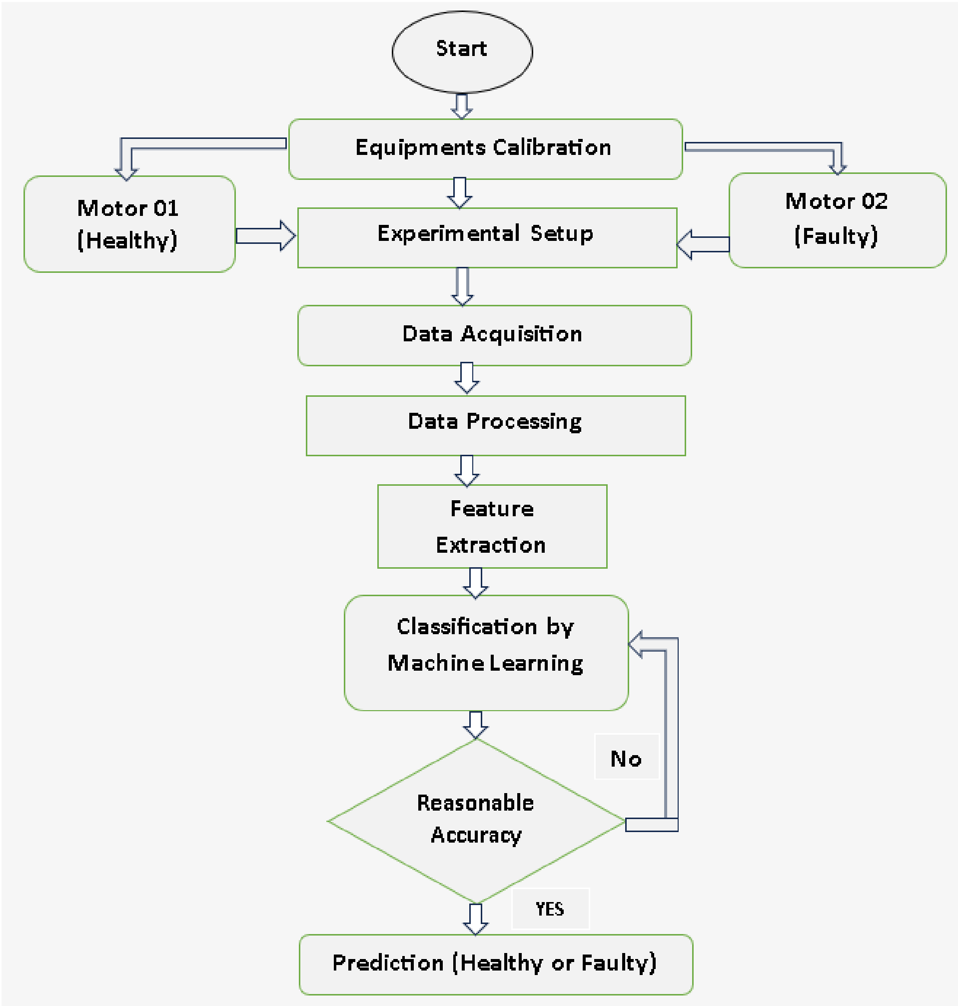


Figure 38: Flow chart for purposed methodology

In order to diagnose the fault of induction motor with high accuracy and result.Experimental test bench was set up as shown in Figure.It consists of

- **Induction Motor**
- **Arduino Uno**
- **Acoustic Sensor**
- **Current Sensor**

- **Vibrational Sensor**
- **Voltage Regulator**
- **Tachometer**
- **Loads**
- **Power Supply**

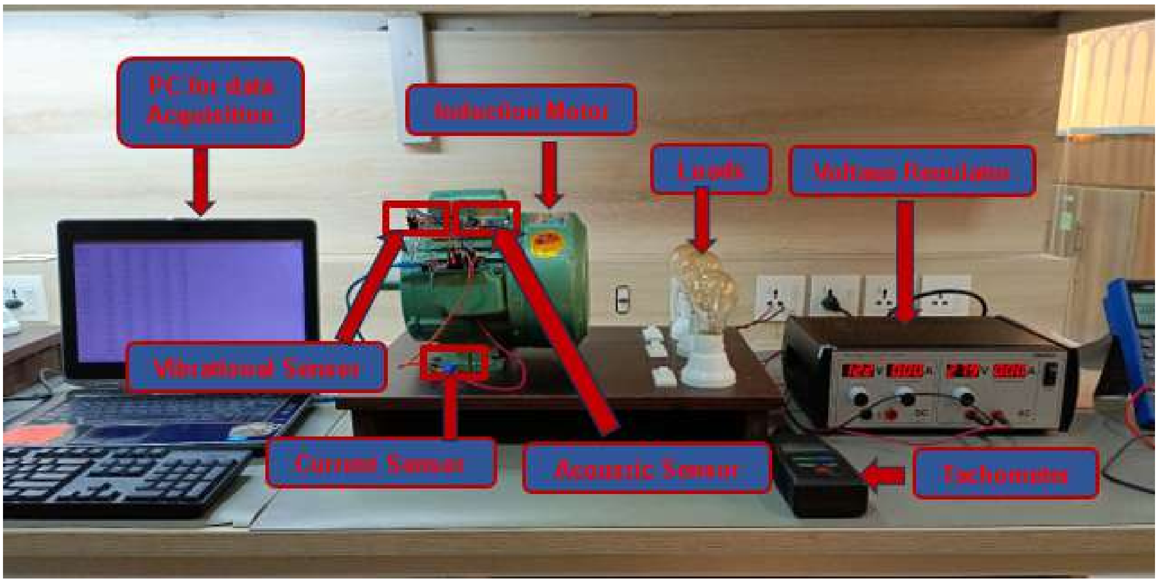


Figure 39: Experimental Test Bench

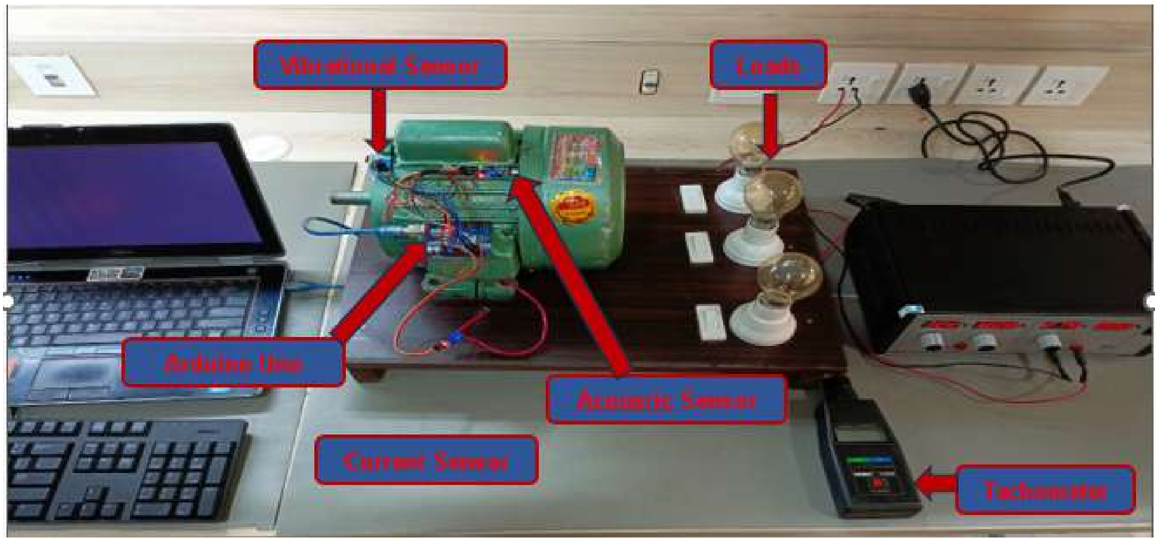


Figure 40: Experimental Test Bench Top view

5.1 Induction Motor

At Start a Induction motor was bought as a beginner to this project.The cost of this induction motor is PKR 9500/-.Two Squirrel-cage induction motors purchased for experiments.One for Healthy Data Acquisition and Second for Faulty Data Acquisition on different load and rpm conditions. It is the single phase Induction motor with Specification were as follow:

Parameters	Data
Voltage	208-230 V
Frequency	50 Hz
Power	0.37 W
HP	0.5
Pole	4
RPM	1450

Table 5: Specification of Induction Motor

Figure 41: Induction Motor for Experiment

5.2 Arduino Uno

The Arduino Uno software and hardware is a popular microcontroller board based on the ATmega328P microcontroller. It's part of the Arduino family of development boards, which are designed for building and prototyping electronic projects. The Arduino Uno is particularly well-suited for beginners due to its simplicity and ease of use. The Arduino Uno can be used for fault diagnosis in induction motors by monitoring various parameters and analyzing the data.The Arduino Uno has the following features:

- 14 digital input/output pins, including six that can be used as PWM outputs
- Six analog inputs
- A 16 MHz ceramic resonator
- A USB connection
- A reset button

- An ICSP header

The Arduino Uno can be used in a variety of electronic projects, including interfacing with other Arduino boards, Arduino shields, and Raspberry Pi boards. It can also control relays, LEDs, servos, and motors as an output. Here's how you can utilize the Arduino Uno in this project:

5.2.1 Sensor Integration:

Connect sensors to the Arduino Uno to measure relevant parameters of the induction motor. For fault diagnosis, you may consider using sensors such as current sensors, voltage sensors, temperature sensors, vibration sensors, or Hall Effect sensors. These sensors will help you gather data about the motor's performance.

5.2.2 Data Acquisition:

Use the analog or digital input pins of the Arduino Uno to read the sensor data. You can interface the sensors directly with the Arduino Uno or use additional circuitry, depending on the sensor requirements.

5.2.3 Data Processing and Analysis:

All the sensor data acquired and then get the rms values of all the induction motor sensors.Develop a program using the Arduino IDE to process and analyze the sensor data. You can implement algorithms or techniques to detect faults or anomalies in the motor's behavior. For example, you could monitor the current waveform for irregularities, analyze temperature trends, or detect excessive vibrations.

Figure 42: Arduino Uno

5.3 Acoustic Sensors

The KY-037 can detect loud sounds, such as slamming doors. It has analog and digital outputs, so the sensor can be ready by any microcontroller.Acoustic sensors are used to detect and measure sound waves or acoustic signals in the environment. They have many applications, including environmental monitoring, industrial condition monitoring, security and surveillance, healthcare, automotive and robotics.

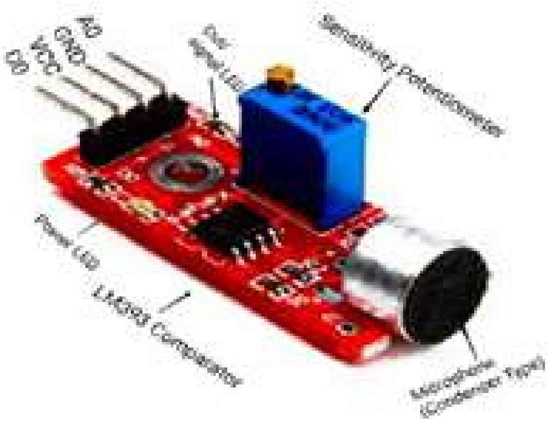


Figure 43: KY-037

Here are some specifications for the KY-037 High Sensitive Sound Microphone Sensor:

Parameters	Specification
Main chip	LM393
Microphone type	Electret condenser microphone
Working voltage	DC 4-6V
Output	Single channel signal
Induction distance	Maximum of 0.5m
Electret condenser microphone	CMA-6542PF
resistors	6
potentiometer	3296W
Microphone sensitivity	-42 ±3 db

Table 6: Specification of Acoustic Sensor

5.4 Current Sensors

Current sensors such as Hall Effect current sensors can be used to measure the current flowing through the motor windings. They provide data on the motor's electrical performance and can help detect issues like overloading, phase imbalances, or abnormal current patterns.The current sensor used in the project for current flowing through the motor is The ACS712 is a linear current sensor that uses its conductor to calculate and measure the amount of current applied. It has a low-noise analog signal path, an 80 kHz bandwidth, and a 5 µs output rise time in response to step input current.

Figure 44: AC-712

Here are some specifications for the ACS712 current sensor module:

Parameters	Specification
SDimensions	31 x 13 x 14 mm (LxWxH)
Weight	3 gm
Working voltage	DC 4-6V
Output sensitivity	66 to 185 mV/A
Measure current range	-20A 20A
Sensitivity	100mV/A
Internal conductor resistance	1.2 m
Minimum isolation voltage	2.1 kV RMS from pins 1-4 to pins 5-8
Operation	5.0 V, single-supply
Measurement range	-30 to +30 A
Scale factor	66 mV per A

Table 7: Specification of Current Sensor

5.5 Vibration Sensors

Vibration sensors, such as accelerometer or vibration modules, are used to detect abnormal vibrations in the motor. Unusual levels of vibration can be an indication of misalignment, rotor imbalance, bearing wear, or mechanical faults. By monitoring vibrations, we can identify faults and schedule maintenance or repairs.we use adxl-335 sensor for the vibrational data acquition.

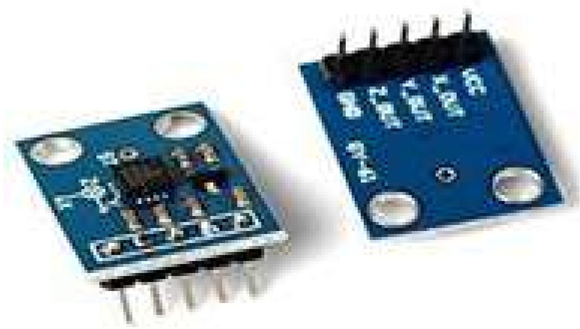


Figure 45: adxl-335

Here are some specifications for the ADXL-335 accelerometer module:

Parameters	**Specification**
Supply voltage	2.8–3.6 V
Current consumption	320 uA
Sensitivity	300 mV/g
Bandwidth	03 Hz to 05 kHz
Dynamic range	±3 g
Operating temperature	-40°C to +85°C
Package type	Surface mount plastic package (LFCSP)
Pin configuration	5 pin, 1.27 mm pitch

Table 8: Sensor Specifications

5.6 Tachometer

A tachometer is a device that measures the rotation speed of a shaft or disk, such as in a motor or other machine. It's designed to measure the revolutions per minute (RPM) of a moving object.Tachometers are typically used in motors and other machines, and are widely found in the automotive and aviation industries. They can be available as a handheld or fixed-mount models, depending on if they're to be used as permanent monitoring or spot-checking tools.

Figure 46: Tachometer

5.7 Different Loads

A bulb is an ohmic load, and adding lamps can add extra heat to the filament of the lamp.An induction motor's power factor changes with load. At full load, the power factor is usually around 0.85 or 0.90, while at no-load it can be as low as about 0.20.An induction motor use incandescent light bulbs as test loads in this project.

Figure 47: Workbench with 100W

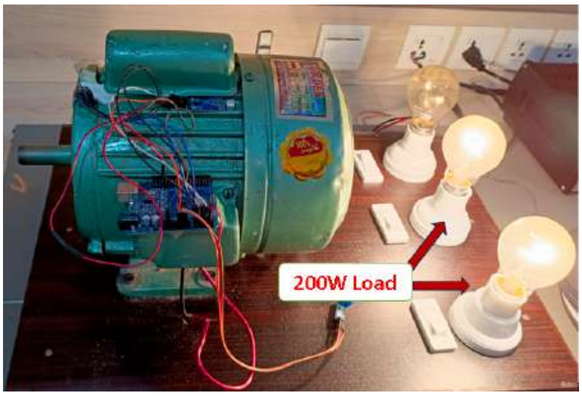


Figure 48: Workbench with 200W

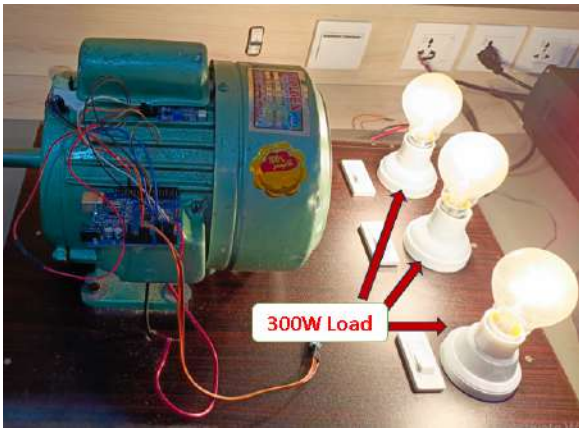


Figure 49: workbench with 300W

5.8 Software

Matlab used for Data Acquisition, for Decision and comparison of the healthy and faulty Induction Motor data and Design the GUI for Monitoring of the Induction Motor.In Matlab use Diagnostic feature Designer for feature Extraction and Classification Learner for Algorithm modelling.MATLAB is a high-level programming language and environment widely used in various fields, including engineering, science, and mathematics. With MATLAB, users can perform tasks such as data manipulation, algorithm development, simulation, modeling, and the implementation of complex mathematical operations. MATLAB is commonly used in the health monitoring of induction motors due to its powerful computational capabilities and extensive toolboxes for signal processing and data analysis.

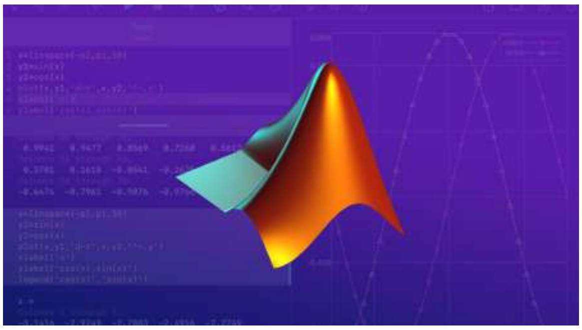


Figure 50: Matlab

5.9 Different Faults Induced

For collection of data,our main focus is on the bearing faults including inner race fault,outer race fault and compound fault .Collect the faults and healthy the dataset using vibrational sensor,Acoustic and Current.Different Faults is induced in the bearing before start the motor.We have two Motor 1)Used for Healthy dataset 2) Used for Faulty dataset.The main focus of data collection is the bearing faults of the motor.Using Healthy and Faulty bearing(Outer Race ,Inner Race ,Ball Fault and Compound Fault) in the project.Fault induced in the bearing through drill and rough the surfaces according to the faults.Specification of the bearing is :-

Geometry parameter	Size
Outer diameter (D_O)	47 mm
Inner diameter (D_I)	20 mm
Pitch diameter (D_P)	33.5 mm
Ball diameter (D_B)	8 mm
Number of balls (N_B)	8
Contact angle (θ)	0°

Figure 51: Bearing Specification

Figure 52: Healthy Bearing

Figure 53: Compound Fault

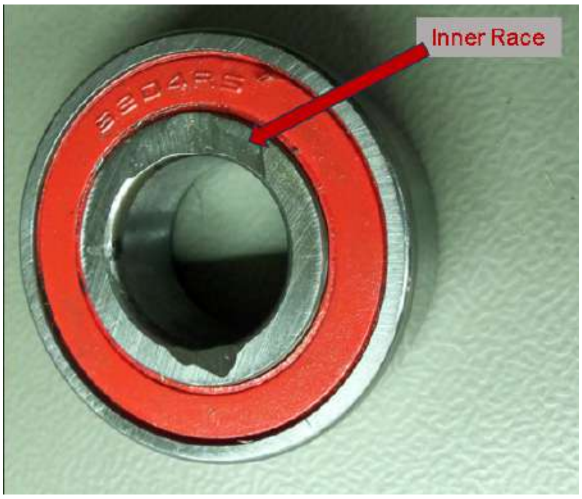


Figure 54: Inner Race Fault

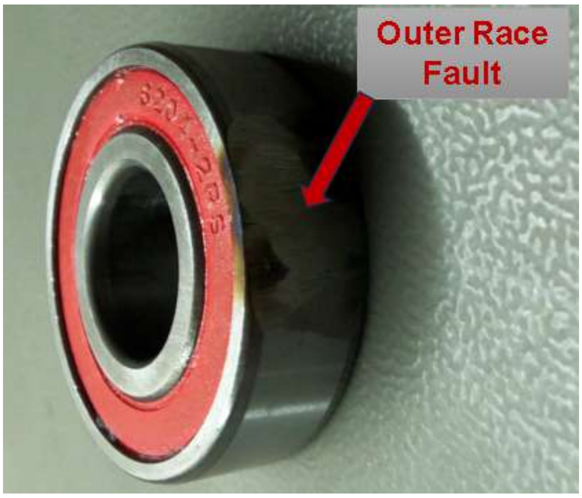


Figure 55: Outer Race Fault

Figure 56: Ball Fault

Chapter 6

6 Vibrational Setup

6.1 Studies Relevant to Bearing Faults

Numerous studies have focused on the application of machine learning techniques specifically for the diagnosis of bearing faults in induction motors. These studies utilize various types of data, including vibration signals, current signatures, and acoustic emissions, to detect and classify different types of bearing faults such as inner race, outer race, and Ball faults.

- For instance, Zhang et al. (2019) developed a fault diagnosis method based on deep learning for detecting bearing faults in induction motors using vibration signals. Their convolutional neural network (CNN) model achieved high accuracy in distinguishing between healthy and faulty bearings, demonstrating the effectiveness of deep learning approaches in bearing fault diagnosis.
- Li et al. (2020) proposed a fault diagnosis framework that combined vibration signal analysis with machine learning algorithms, including support vector machines (SVM) and decision trees, to classify bearing faults in induction motors. Their study highlighted the importance of feature selection and fusion techniques in improving fault classification accuracy.
- In another study, Wang et al. (2018) investigated the use of acoustic emission signals for bearing fault diagnosis in induction motors. They developed a novel feature extraction method based on wavelet packet decomposition and applied machine learning algorithms such as k-nearest neighbors (KNN) and random forests for fault classification.

6.2 Analysis of Vibrational Signal Trace

Signal features provide general signal-based statistical metrics that can be applied to any kind of signal, including a time-synchronized average (TSA) vibration signal. Changes in these features can indicate changes in the health status of a system. Diagnostic Feature Designer provides a set of feature options .The statistical features include basic mean, standard deviation, and root mean square (RMS) metrics. In addition, the feature set includes shape factor and the higher order kurtosis and skewness statistics. All these statistics can be expected to change as a deteriorating fault signature intrudes upon the nominal signal. In this project ,after import the dataset and see the Signal trace of all three axis:

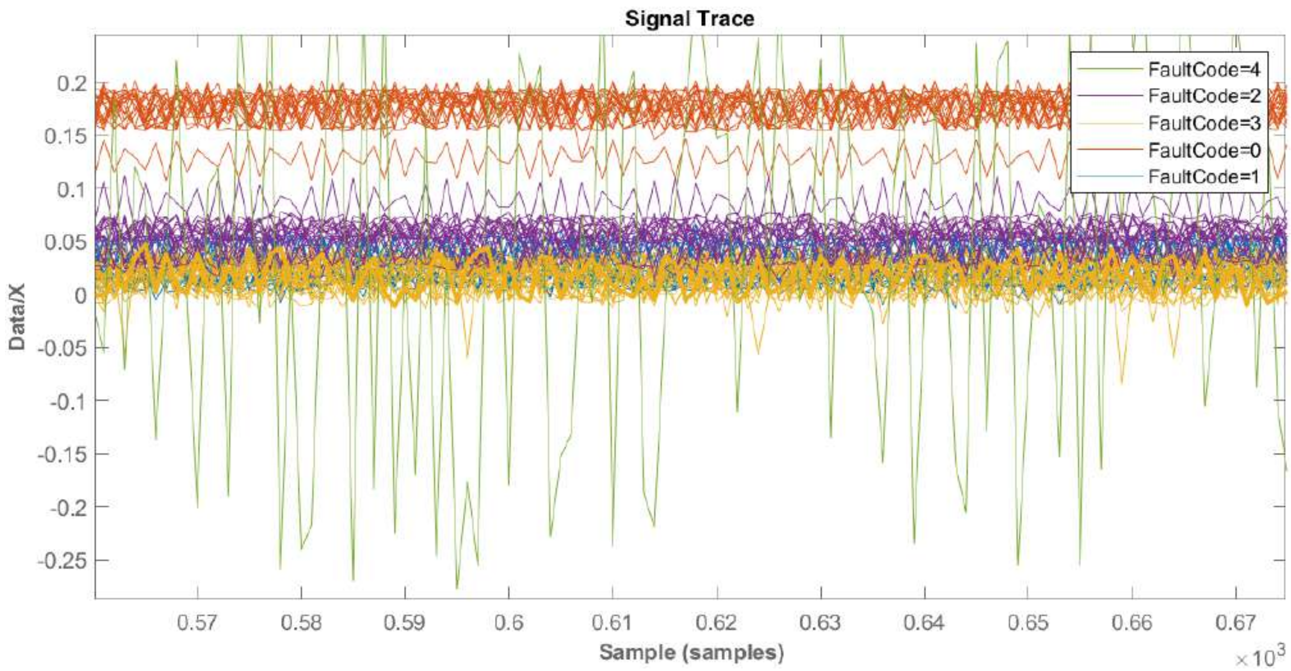


Figure 57: Signal of X

Power Spectrum view of the data with healthy and faulty indication.

Figure 58: Signal of Y

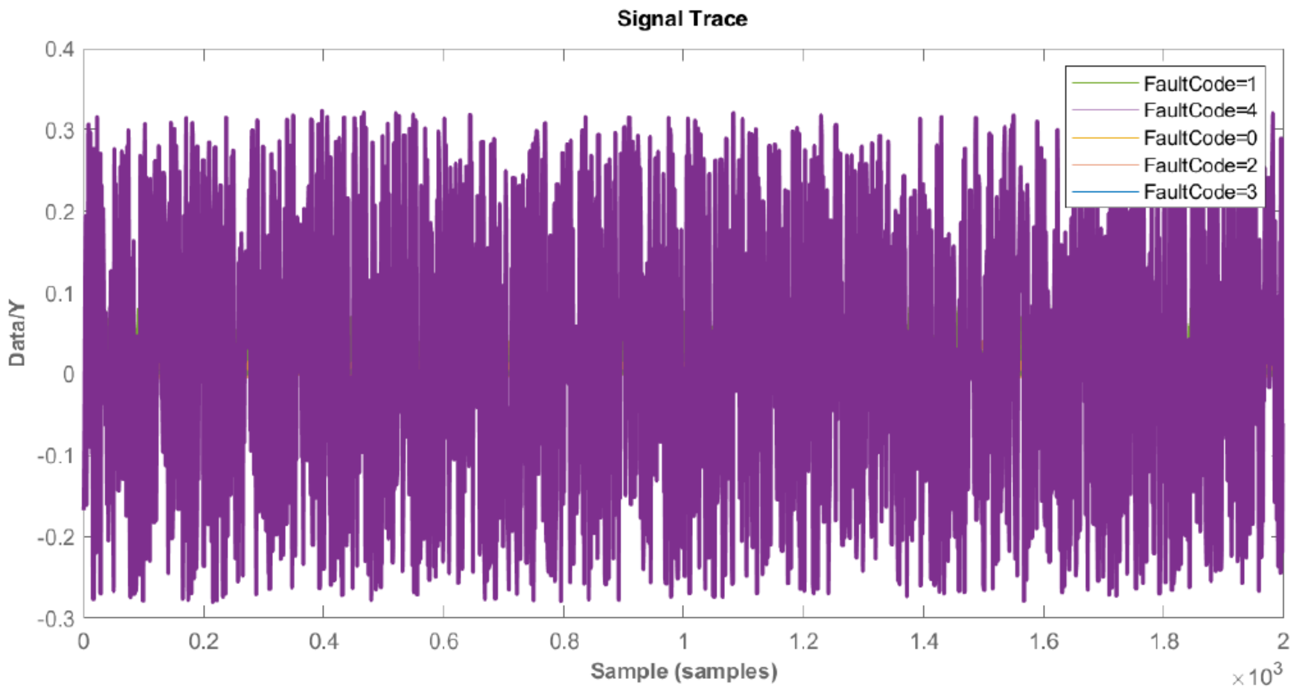


Figure 59: Signal of Z

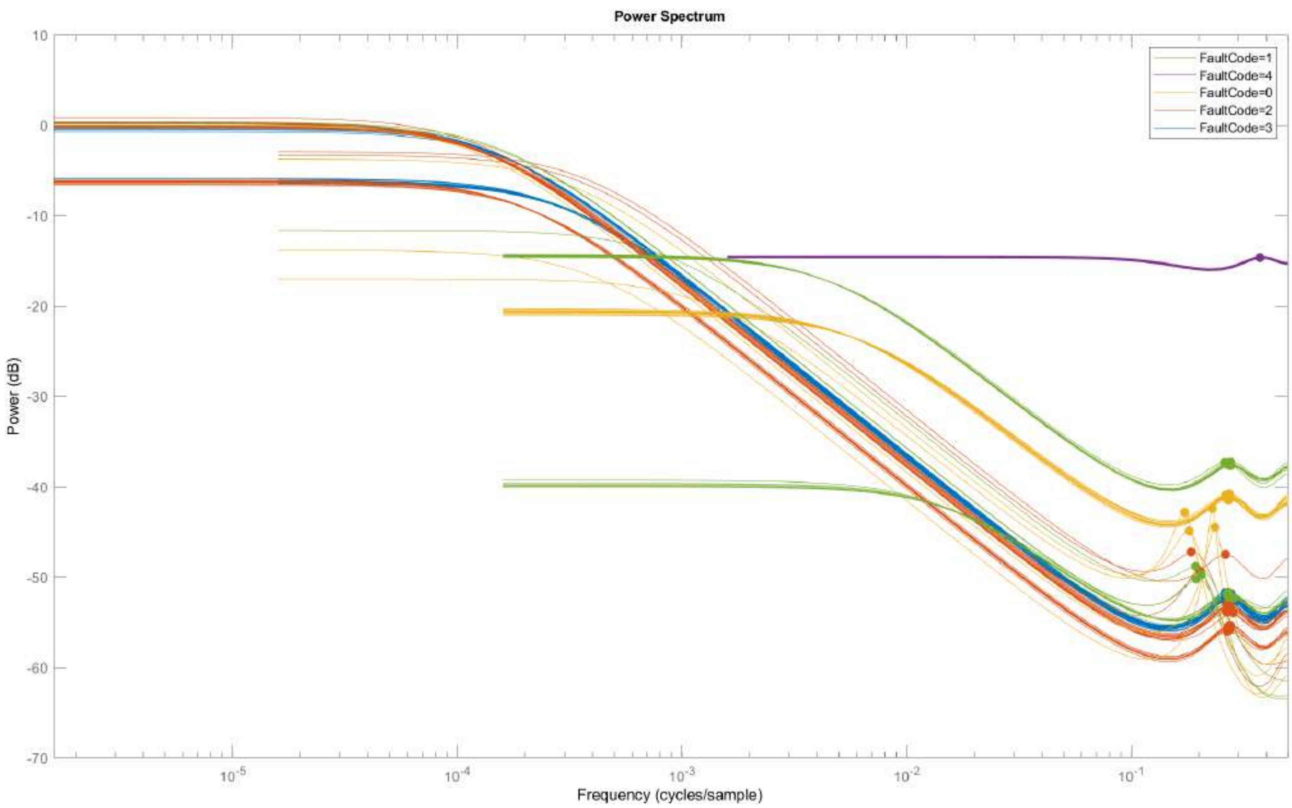


Figure 60: Power Spectrum of X

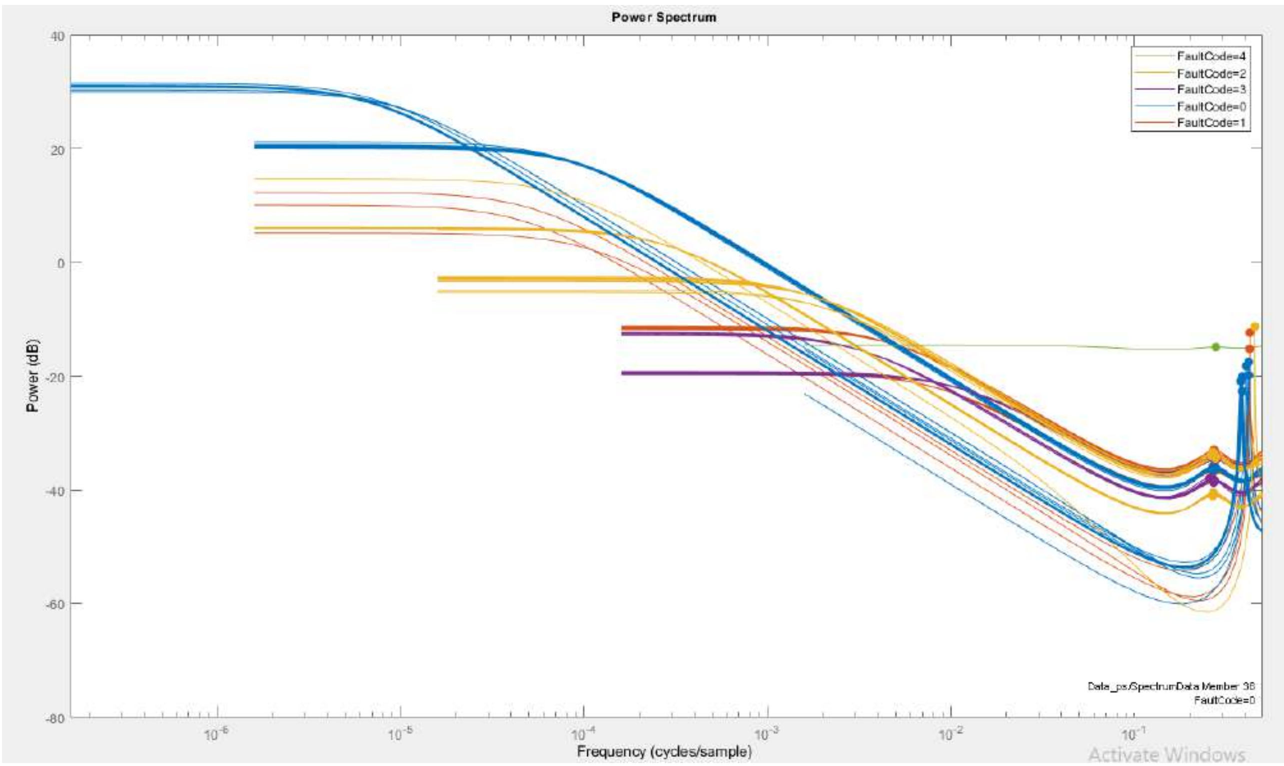


Figure 61: Power Spectrum of Y

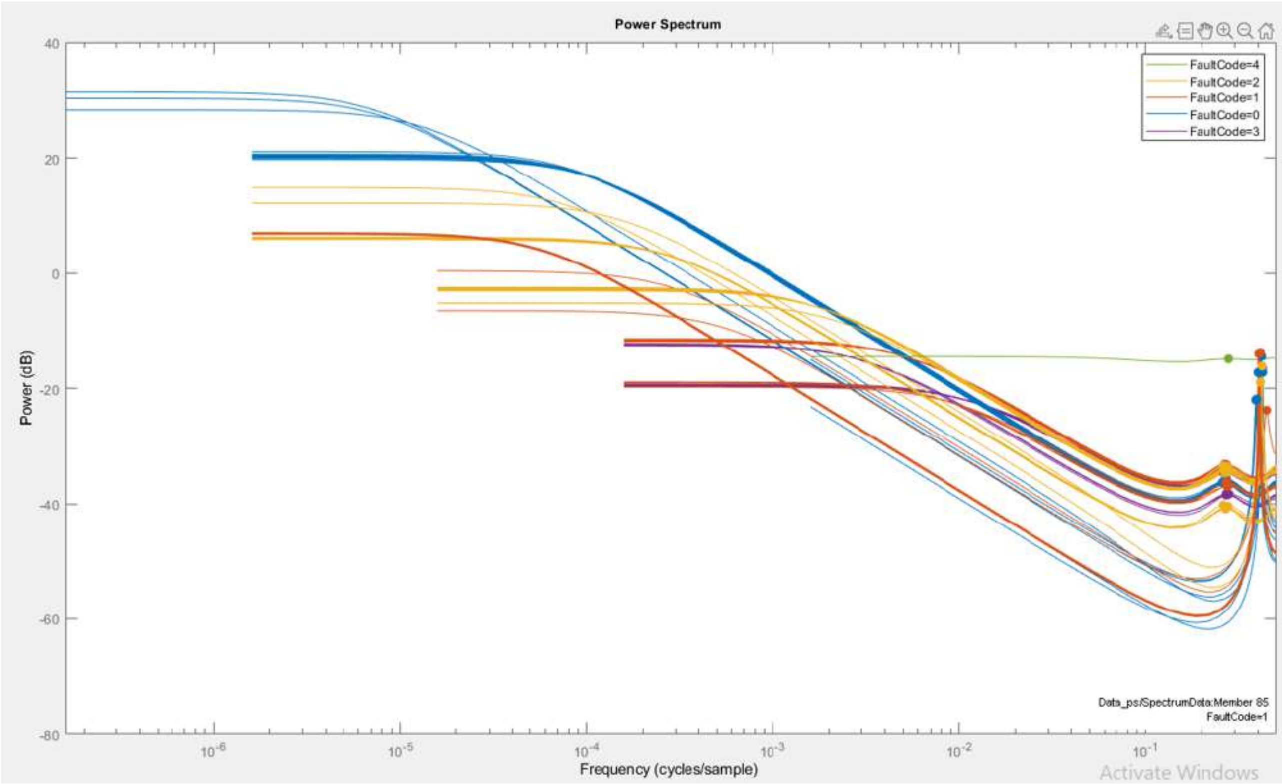


Figure 62: Power Spectrum of Z

6.3 Analysis of data Features

The Diagnostic Feature Designer tool of MATLAB offers a user-friendly interface for extracting both time domain and frequency domain features from signals, which are essential for various diagnostic applications such as condition monitoring, fault detection, and predictive maintenance. Time domain features, derived directly from the signal's amplitude values over time, provide valuable insights into the signal's behavior and characteristics. For instance, the mean (average) represents the central tendency of the signal, while the standard deviation quantifies its spread or variability. The root mean square (RMS) value reflects the effective amplitude and quantifies signal power. Skewness and kurtosis measure the asymmetry and peakedness of the signal's distribution, respectively, providing information about its shape and characteristics. Additionally, features such as peak amplitude, crest factor, entropy, and shape factor offer insights into signal properties such as maximum excursion, peak-to-average power ratio, randomness, and signal shape, respectively. Each of these features contributes to a comprehensive understanding of the signal's behavior, aiding in the diagnosis and analysis of various system conditions and faults.Get the Time domain features and also frequency domain feature extracted from the matlab tool.

	FaultCode	Data_sigstats/ClearanceFactor	Data_sigstats/CrestFactor	Data_sigstats/ImpulseFactor	Data_sigstats/Kurtosis	Data_sigstats/Mean	Data_sigstats/PeakValue	Data_sigstats/RMS	Data_sigstats/ShapeFactor	Data_sigstats/Skewness	Data_sigstats/Std	Data_sigstats_1/ClearanceFactor
1	3	3.3420	2.2315	2.7808	1.9160	0.0184	0.0559	0.0250	1.2372	-0.0698	0.0170	3.3420
2	0	1.1077	1.1028	1.1060	2.1050	0.1761	0.1947	0.1766	1.0029	-0.4032	0.0134	1.1077
3	0	1.1438	1.1368	1.1415	1.8031	0.1830	0.2089	0.1837	1.0041	-0.1880	0.0166	1.1438
4	1	4.5061	3.2341	3.9959	6.6687	0.0166	0.0678	0.0210	1.2356	1.5332	0.0129	4.5061
5	0	1.1066	1.1018	1.1050	2.1348	0.1760	0.1945	0.1765	1.0029	-0.4116	0.0134	1.1066
6	2	1.4399	1.4206	1.4334	2.1128	0.0586	0.0840	0.0592	1.0090	0.2833	0.0079	1.4399
7	1	4.5149	3.2293	3.9941	6.6703	0.0165	0.0677	0.0210	1.2368	1.5264	0.0129	4.5149
8	4	2.4402	1.8884	2.1765	1.7855	0.0209	0.3350	0.1774	1.1526	-0.0184	0.1762	2.4402
9	1	2.0843	1.5999	1.8502	1.5270	0.0319	0.0589	0.0368	1.1564	-0.1856	0.0185	2.0843
10	2	1.5822	1.4398	1.5184	1.5850	0.0519	0.0767	0.0547	1.0534	-0.0981	0.0172	1.5822
11	1	2.0838	1.5981	1.8486	1.5305	0.0318	0.0589	0.0368	1.1568	-0.1878	0.0185	2.0838
12	0	1.1091	1.1043	1.1075	2.1288	0.1760	0.1950	0.1765	1.0029	-0.4057	0.0134	1.1091
13	0	1.1074	1.1025	1.1057	2.1155	0.1760	0.1946	0.1765	1.0029	-0.4055	0.0134	1.1074
14	0	1.1985	1.1848	1.1925	1.9245	0.1239	0.1478	0.1247	1.0068	-0.1471	0.0143	1.1985
15	3	3.4340	2.2819	2.8269	1.9242	0.0184	0.0571	0.0250	1.2388	-0.0699	0.0170	3.4340
16	2	1.4329	1.4138	1.4265	2.1007	0.0586	0.0836	0.0591	1.0090	0.2871	0.0079	1.4329
17	2	1.7662	1.5881	1.6968	1.7086	0.0441	0.0748	0.0471	1.0685	-0.1325	0.0166	1.7662
18	3	3.5673	3.2644	3.4447	19.1278	0.0232	0.0832	0.0255	1.0552	-2.6591	0.0106	3.5673
19	1	4.5134	3.2291	3.9939	6.6972	0.0166	0.0677	0.0210	1.2368	1.5362	0.0129	4.5134
20	3	3.4174	2.2744	2.8172	1.9233	0.0184	0.0569	0.0250	1.2387	-0.0650	0.0170	3.4174
21	3	3.3537	2.2371	2.7687	1.9216	0.0184	0.0560	0.0250	1.2376	-0.0669	0.0170	3.3537
22	3	3.4256	2.2806	2.8233	1.9273	0.0184	0.0571	0.0250	1.2379	-0.0715	0.0170	3.4256
23	3	3.5699	3.2652	3.4461	18.9988	0.0232	0.0833	0.0255	1.0554	-2.6501	0.0106	3.5699
24	3	3.3279	2.2188	2.7455	1.9216	0.0184	0.0555	0.0250	1.2374	-0.0692	0.0170	3.3279
25	4	2.4402	1.8884	2.1765	1.7855	0.0209	0.3350	0.1774	1.1526	-0.0184	0.1762	2.4402
26	0	1.1087	1.1039	1.1071	2.1395	0.1761	0.1949	0.1766	1.0029	-0.4109	0.0134	1.1087
27	2	1.4418	1.4225	1.4353	2.1186	0.0586	0.0842	0.0592	1.0090	0.2907	0.0079	1.4418
28	0	1.1088	1.1040	1.1072	2.1284	0.1760	0.1949	0.1766	1.0029	-0.4104	0.0134	1.1088
29	3	3.3646	2.2447	2.7779	1.9234	0.0184	0.0562	0.0250	1.2375	-0.0704	0.0170	3.3646
30	1	2.0577	1.5782	1.8253	1.5322	0.0318	0.0581	0.0368	1.1568	-0.1889	0.0185	2.0577
31	4	2.4402	1.8884	2.1765	1.7855	0.0209	0.3350	0.1774	1.1526	-0.0184	0.1762	2.4402
32	2	1.5689	1.4458	1.5229	1.5840	0.0519	0.0791	0.0547	1.0533	-0.0950	0.0172	1.5689
33	0	1.1116	1.1068	1.1100	2.1357	0.1760	0.1954	0.1765	1.0029	-0.4096	0.0134	1.1116
34	0	1.1123	1.1075	1.1107	2.1281	0.1760	0.1955	0.1766	1.0029	-0.4055	0.0134	1.1123
35	0	1.1064	1.1016	1.1048	2.1511	0.1760	0.1945	0.1765	1.0029	-0.4120	0.0134	1.1064
36	4	2.4402	1.8884	2.1765	1.7855	0.0209	0.3350	0.1774	1.1526	-0.0184	0.1762	2.4402
37	0	1.1113	1.1064	1.1096	2.1164	0.1760	0.1953	0.1765	1.0029	-0.4061	0.0134	1.1113
38	3	3.3643	2.2410	2.7734	1.9182	0.0184	0.0561	0.0250	1.2375	-0.0713	0.0170	3.3643
39	3	3.5508	3.2491	3.4288	19.1494	0.0232	0.0829	0.0255	1.0553	-2.6674	0.0106	3.5508
40	1	2.0725	1.5904	1.8393	1.5271	0.0319	0.0586	0.0368	1.1565	-0.1883	0.0185	2.0725

Figure 63: Time domain Features

	FaultCode	Data_ps_spec/PeakAmp1	Data_ps_spec/PeakFreq1	Data_ps_spec/BandPower
1	1	1.8832e-04	0.4595	4.8419e-04
2	0	1.8722e-04	0.2268	8.9714e-05
3	1	0.0095	0.4248	0.0192
4	1	1.6467e-04	0.4741	5.5303e-04
5	3	5.5857e-04	0.2457	1.9418e-04
6	3	1.3987e-04	0.4305	0.0022
7	3	1.1965e-04	0.4627	2.2339e-04
8	2	6.5746e-05	0.2707	0.0326
9	4	NaN	NaN	0.0159
10	2	0.3649	0.4550	0.0013
11	1	3.5465e-04	0.2882	0.0011
12	3	3.9895e-04	0.2817	0.0022
13	0	0.0288	0.4206	2.9735e-04
14	3	3.1996e-04	0.2302	0.0014
15	1	1.8416e-04	0.3652	5.2709e-04
16	2	3.2433e-04	0.1459	0.0046
17	0	1.8213e-04	0.2320	8.9220e-05
18	4	NaN	NaN	0.0159
19	2	8.9890e-05	0.2288	3.1362e-05
20	4	NaN	NaN	0.0159
21	2	3.2656e-04	0.1661	0.0049
22	0	3.1642e-04	0.3544	9.0216e-05
23	0	2.1289e-04	0.3889	1.1724e-04
24	2	3.8851e-04	0.1539	0.0531

Figure 64: Frequency Domain Features

All the computed features are now listed, Fault diagnosis of bearings is usually based on vibration signals, and a set of features are extracted in order to classify the faults. we also have the histograms. On these plots, different fault types are highlighted with different colors. Ideally, we want to have a plot that looks separately. All different colored distributions are apart from each other. If our histogram plots looks separately then, we could easily discriminate between different types of faults. But instead they look similar to each other, where there's a lot of overlapping between different fault types. Due to overlapping and a large number of features, it is really hard for us to tell the most useful features just by looking at plots. However, this app lets us rank these features to determine the ones that will help us effectively separate different types of faults. Histograms of the selected dataset:-

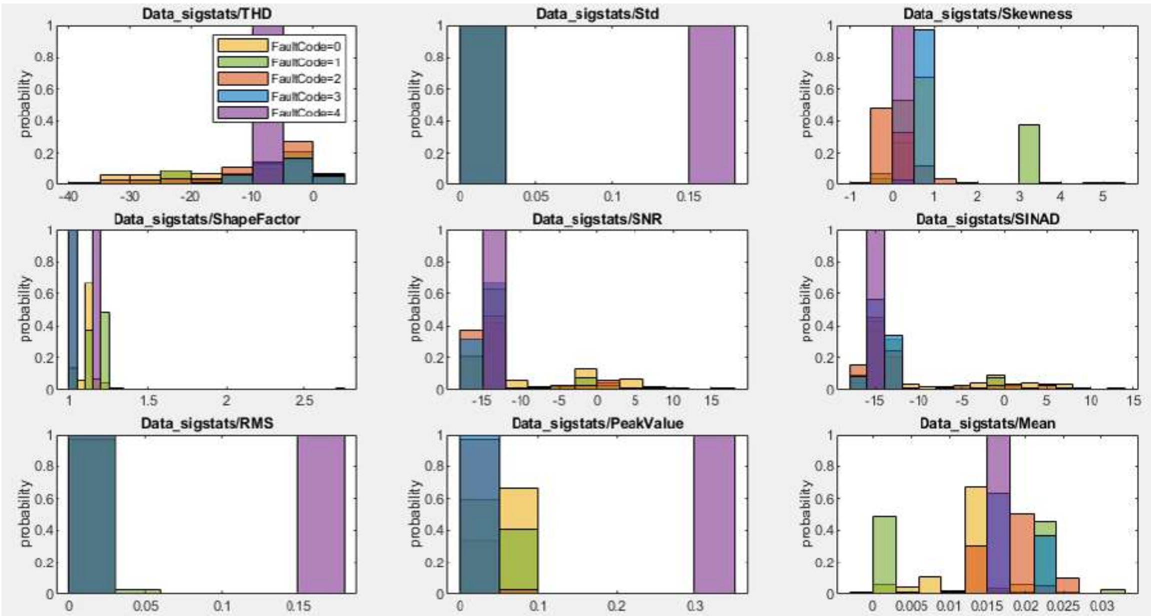


Figure 65: Histograms of Vibrational

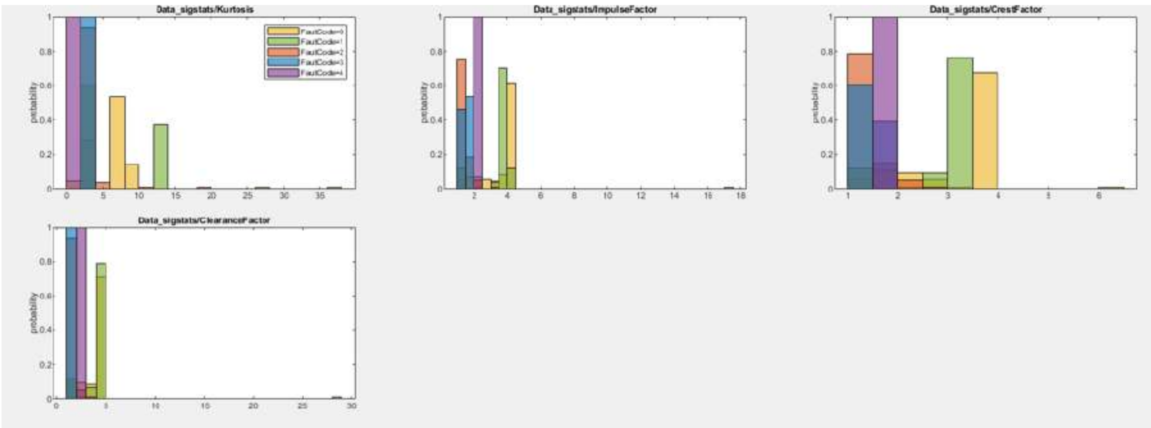


Figure 66: Histograms of Vibrational

6.4 Results

In this project for fault diagnosis in induction motors using machine learning, MATLAB's Classification Learner app is employed to explore and compare the performance of four selected models: Fine Gaussian SVM, Fine KNN, Ensemble (Bagged Tree), and Medium Neural Network. These models represent a range of machine learning algorithms suitable for classification tasks, including support vector machines, k-nearest neighbors, ensemble methods, and neural networks. By using the app's capabilities, such as automated training, feature selection, and validation schemes, the project aims to identify the most effective model for accurately classifying fault conditions in induction motors. Through careful evaluation of performance metrics and model interpretability, the project seeks to provide a practical and reliable solution for diagnosing faults in real-world industrial settings, considering factors such as data complexity, computational resources, and desired performance criteria.In this project utilizes vibrational sensor data collected specifically from bearings to enhance diagnostic accuracy.Confusion matrix of vibrational Train data:

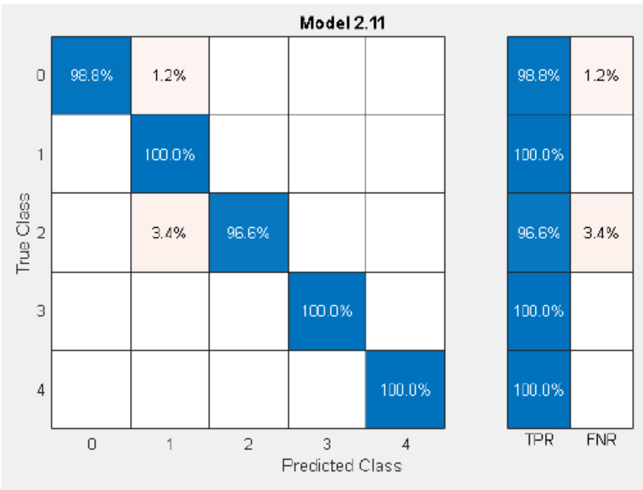


Figure 67: SVM Vibrational Confusion matrix

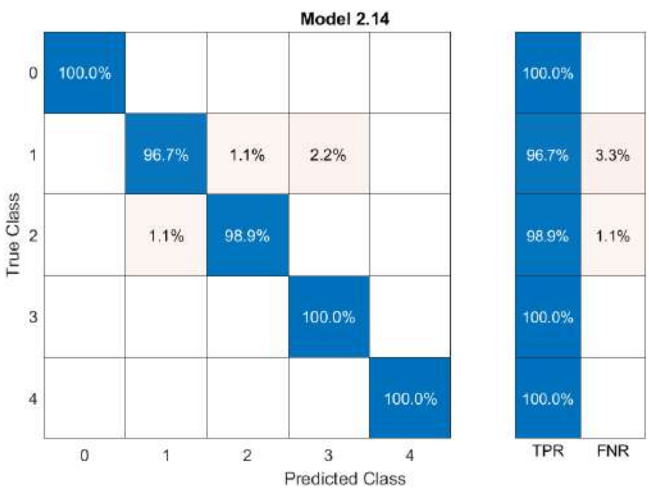


Figure 68: KNN Vibrational Confusion matrix

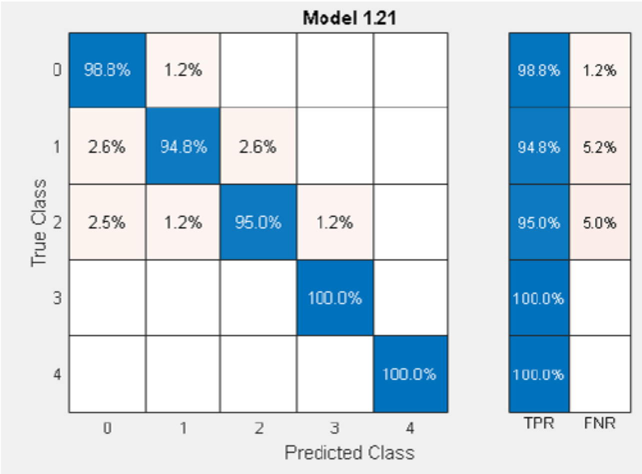


Figure 69: Ensemble Vibrational Confusion matrix

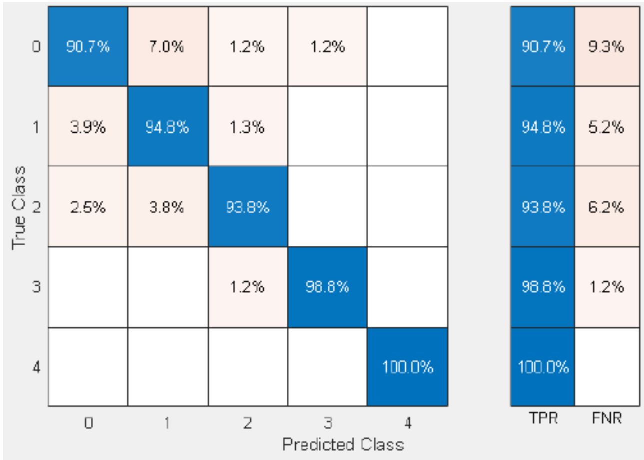


Figure 70: Neural Vibrational Confusion matrix

The area under the ROC curve (AUC) serves as a valuable metric for evaluating the performance of a classifier in distinguishing between different classes of motion or vibration patterns. The ROC curve provides a graphical representation of the classifier's performance across various threshold settings, where the true positive rate (sensitivity) is plotted against the false positive rate (1-specificity). A perfect classifier would have an ROC curve that hugs the upper left corner of the plot, indicating a true positive rate of 1 and a false positive rate of 0 across all threshold settings, resulting in an AUC value of 1. This scenario would suggest that the classifier achieves optimal discrimination between different motion or vibration patterns captured by the accelerometer sensor data. Conversely, a classifier with an AUC value close to 0.5 suggests random classification, indicating no better performance than chance. By analyzing the AUC value in conjunction with the ROC curve, practitioners can gain insights into the confidence and reliability of the classifier's classification decisions based on accelerometer sensor data, thereby facilitating informed decision-making in various applications such as activity recognition, structural health monitoring, or fault diagnosis.

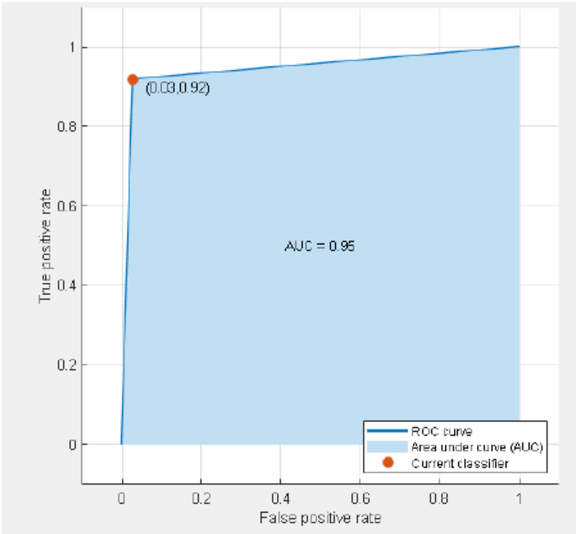


Figure 71: ROC Curve of X

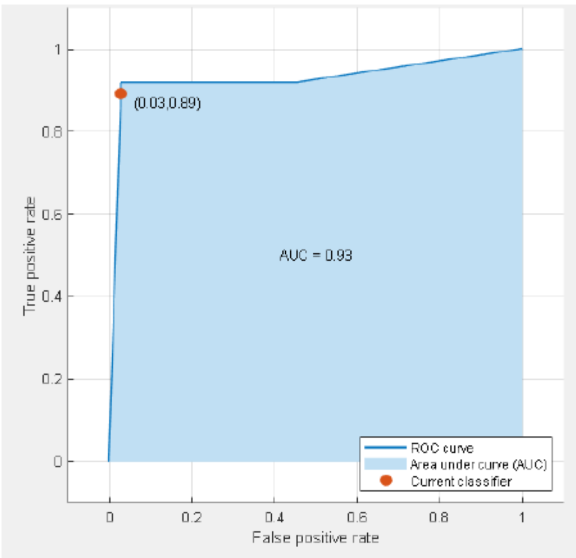


Figure 72: ROC Curve of Y

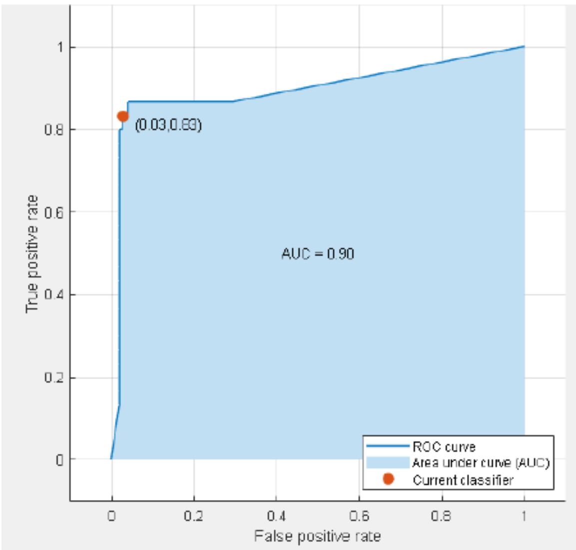


Figure 73: ROC Curve of Z

Chapter 7

7 Acoustic Emission Setup

7.1 Studies Relevant to Bearing Faults

Acoustic Emission technique related to bearing faults in induction motors are instrumental in understanding the characteristic acoustic signals emitted by bearings undergoing various fault conditions. These studies involve monitoring the high-frequency sound waves generated by the friction, impact, and structural changes within the bearing components. By analyzing AE signals, researchers can identify distinctive patterns associated with specific types of faults, such as inner race, outer race, or Ball fault.

- ”Acoustic Emission-Based Bearing Fault Diagnosis: A Review” ,In This paper provides a comprehensive review of acoustic emission-based techniques for bearing fault diagnosis, including methodologies, signal processing techniques, and case studies. It discusses the application of acoustic emission analysis in detecting various types of bearing faults, including defects in the inner race, outer race, and rolling elements, with a focus on its application in induction motors.
- ”Acoustic emission analysis for bearing fault diagnosis of electrical machines: A review”,This review paper explores the use of acoustic emission analysis for bearing fault diagnosis in electrical machines, including induction motors. It discusses the principles of acoustic emission, signal processing techniques, and the application of AE analysis for detecting different types of bearing faults. The paper also highlights the advantages and limitations of AE-based techniques in fault diagnosis.

7.2 Analysis of Acoustic Signal Trace

Acoustic data, capturing sound wave emissions from the system, offers complementary insights alongside vibration signals. By integrating acoustic emission techniques, which analyze high-frequency sound waves generated by system components, such as bearings, additional fault signatures can be detected.

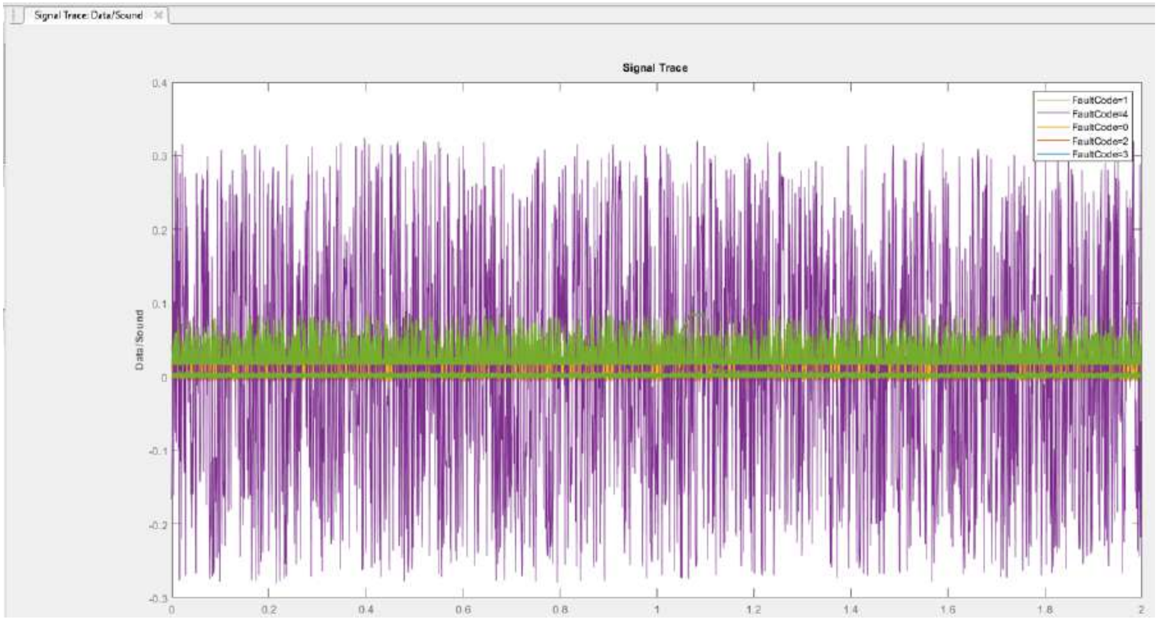


Figure 74: Sound Signal

Power Spectrum of the Acoustic data and sampling frequency is 1000Hz.

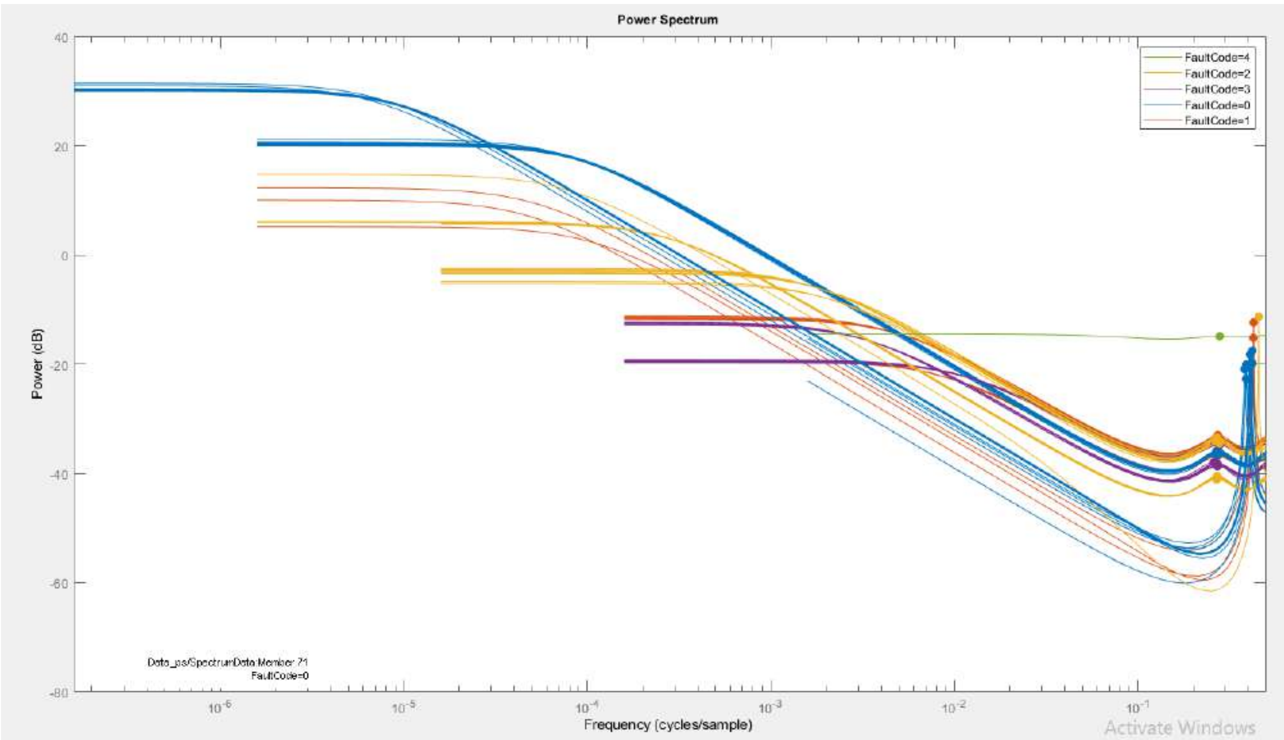


Figure 75: Power Spectrum of Sound signal

7.3 Analysis of data Features

Diagnostic Feature Designer provides a diverse set of feature options, including basic statistical metrics like mean, standard deviation, and root mean square (RMS), as well as shape factor, kurtosis, and skewness statistics.Extract the Features and then model on algorithm.

Data_sigstats_1/CrestFactor	Data_sigstats_1/ImpulseFactor	Data_sigstats_1/Kurtosis	Data_sigstats_1/Mean	Data_sigstats_1/PeakValue	Data_sigstats_1/RMS	Data_sigstats_1/ShapeFactor	Data_sigstats_1/Skewness	Data_sigstats_1/Std
2.2315	2.7608	1.9160	0.0184	0.0559	0.0250	1.2372	-0.0698	0.0170
1.1028	1.1060	2.1050	0.1761	0.1947	0.1766	1.0029	-0.4032	0.0134
1.1368	1.1415	1.6031	0.1830	0.2089	0.1837	1.0041	-0.1880	0.0166
3.2341	3.9959	6.6687	0.0166	0.0678	0.0210	1.2356	1.5332	0.0129
1.1018	1.1050	2.1348	0.1760	0.1945	0.1765	1.0029	-0.4116	0.0134
1.4206	1.4334	2.1128	0.0586	0.0840	0.0592	1.0090	0.2833	0.0079
3.2293	3.9941	6.6703	0.0165	0.0677	0.0210	1.2368	1.5264	0.0129
1.8884	2.1765	1.7855	0.0209	0.3350	0.1774	1.1526	-0.0184	0.1762
1.5999	1.8502	1.5270	0.0319	0.0589	0.0368	1.1564	-0.1856	0.0185
1.4396	1.5164	1.5650	0.0519	0.0787	0.0547	1.0534	-0.0961	0.0172
1.5981	1.8486	1.5305	0.0318	0.0589	0.0368	1.1568	-0.1878	0.0185
1.1043	1.1075	2.1288	0.1760	0.1950	0.1765	1.0029	-0.4057	0.0134
1.1025	1.1057	2.1155	0.1760	0.1946	0.1765	1.0029	-0.4055	0.0134
1.1846	1.1925	1.9245	0.1239	0.1478	0.1247	1.0066	-0.1471	0.0143
2.2819	2.8269	1.9242	0.0184	0.0571	0.0250	1.2388	-0.0699	0.0170
1.4138	1.4265	2.1007	0.0586	0.0836	0.0591	1.0090	0.2871	0.0079
1.5881	1.6968	1.7086	0.0441	0.0748	0.0471	1.0685	-0.1325	0.0166
3.2644	3.4447	19.1278	0.0232	0.0832	0.0255	1.0552	-2.6591	0.0106
3.2291	3.9939	6.6972	0.0166	0.0677	0.0210	1.2368	1.5362	0.0129
2.2744	2.8172	1.9233	0.0184	0.0569	0.0250	1.2387	-0.0650	0.0170
2.2371	2.7687	1.9216	0.0184	0.0560	0.0250	1.2376	-0.0669	0.0170
2.2806	2.8233	1.9273	0.0184	0.0571	0.0250	1.2379	-0.0715	0.0170
3.2652	3.4461	18.9988	0.0232	0.0833	0.0255	1.0554	-2.6501	0.0106
2.2188	2.7455	1.9216	0.0184	0.0555	0.0250	1.2374	-0.0692	0.0170
1.8884	2.1765	1.7855	0.0209	0.3350	0.1774	1.1526	-0.0184	0.1762
1.1039	1.1071	2.1395	0.1761	0.1949	0.1766	1.0029	-0.4109	0.0134
1.4225	1.4353	2.1186	0.0586	0.0842	0.0592	1.0090	0.2907	0.0079
1.1040	1.1072	2.1284	0.1760	0.1949	0.1766	1.0029	-0.4104	0.0134
2.2447	2.7779	1.9234	0.0184	0.0562	0.0250	1.2375	-0.0704	0.0170
1.5782	1.8253	1.5322	0.0318	0.0581	0.0368	1.1566	-0.1889	0.0185
1.8884	2.1765	1.7855	0.0209	0.3350	0.1774	1.1526	-0.0184	0.1762
1.4458	1.5229	1.5640	0.0519	0.0791	0.0547	1.0533	-0.0950	0.0172
1.1068	1.1100	2.1357	0.1760	0.1954	0.1765	1.0029	-0.4096	0.0134
1.1075	1.1107	2.1281	0.1760	0.1955	0.1766	1.0029	-0.4055	0.0134
1.1016	1.1048	2.1511	0.1760	0.1945	0.1765	1.0029	-0.4120	0.0134
1.8884	2.1765	1.7855	0.0209	0.3350	0.1774	1.1526	-0.0184	0.1762
1.1064	1.1096	2.1164	0.1760	0.1953	0.1765	1.0029	-0.4061	0.0134
2.2410	2.7734	1.9182	0.0184	0.0561	0.0250	1.2375	-0.0713	0.0170
3.2491	3.4288	19.1494	0.0232	0.0829	0.0255	1.0553	-2.6674	0.0106
1.5904	1.8393	1.5271	0.0319	0.0586	0.0368	1.1565	-0.1883	0.0185
1.8884	2.1765	1.7855	0.0209	0.3350	0.1774	1.1526	-0.0184	0.1762

Figure 76: Time domain Acoustic Features

	Data_ps_spec/PeakAmp1	Data_ps_spec/PeakFreq1	Data_ps_spec/BandPower
1	2.3264e-04	0.2674	2.1923e-04
2	2.3615e-04	0.2660	0.0163
3	0.0103	0.4217	0.0016
4	2.1247e-04	0.2720	2.1898e-04
5	3.7259e-04	0.2771	3.1243e-04
6	1.5972e-04	0.2581	3.1957e-04
7	1.3475e-04	0.2707	3.1857e-04
8	8.2919e-05	0.2735	0.0017
9	0.0322	0.2785	0.0157
10	0.0738	0.4549	0.0040
11	4.4380e-04	0.2640	6.6850e-04
12	3.7548e-04	0.2673	3.1195e-04
13	0.0176	0.4204	0.0160
14	3.4606e-04	0.2816	3.1270e-04
15	2.1789e-04	0.2764	2.1808e-04
16	4.5484e-04	0.2690	0.0015
17	2.5576e-04	0.2666	0.0158
18	0.0322	0.2785	0.0157
19	8.9800e-05	0.2697	0.0018
20	0.0322	0.2785	0.0157
21	4.1324e-04	0.2712	0.0015
22	2.3874e-04	0.2746	0.0153
23	2.5053e-04	0.2727	0.0154
24	4.5603e-04	0.2663	0.0015
25	4.5513e-04	0.2698	6.6812e-04
26	3.9054e-04	0.2617	0.0015
27	8.3077e-05	0.2714	0.0017
28	4.9475e-04	0.2734	6.7029e-04
29	0.0322	0.2785	0.0157
30	1.4730e-04	0.2665	3.1784e-04
31	0.0322	0.2785	0.0157
32	2.2920e-04	0.2691	0.0161
33	8.1851e-05	0.2680	0.0017
34	0.0322	0.2785	0.0157
35	7.7419e-05	0.2676	0.0018
36	0.0081	0.3812	0.0140
37	4.2941e-04	0.2780	6.7092e-04
38	3.8867e-04	0.2756	0.0015
39	1.5148e-04	0.2704	3.1799e-04
40	0.0322	0.2785	0.0157

Figure 77: Frequency Domain Acoustic Features

Frequency domain fft result of Acoustic data of healthy and faulty conditions:

- Outer Race Fault signal of fft is :

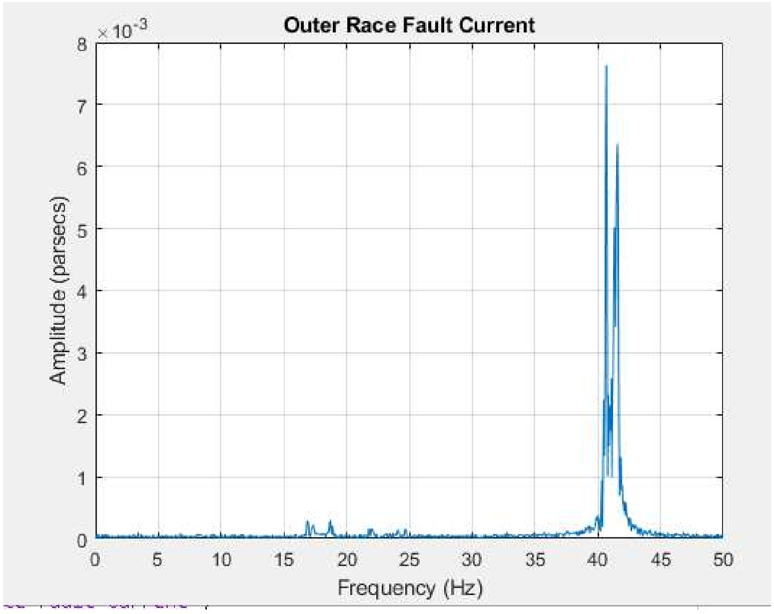


Figure 78: FFT of Outer Race condition of Current

- Inner Race Fault signal of fft is :

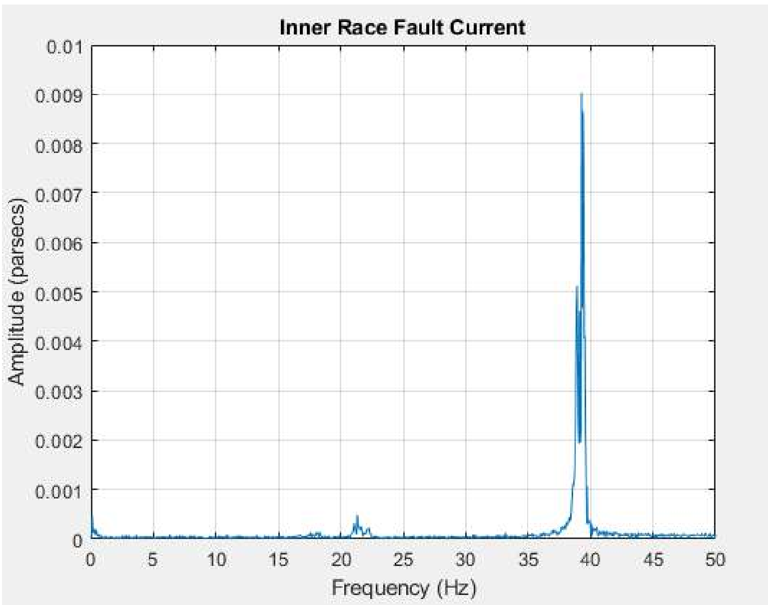


Figure 79: Inner Race Fault signal of Current

- Ball Fault signal of fft is :

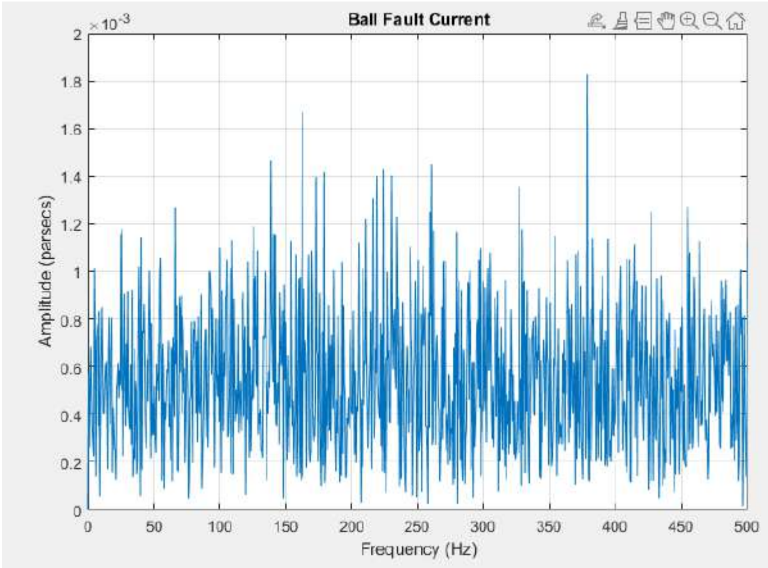


Figure 80: Ball Fault signal of fft

Histograms of Acoustic features:-

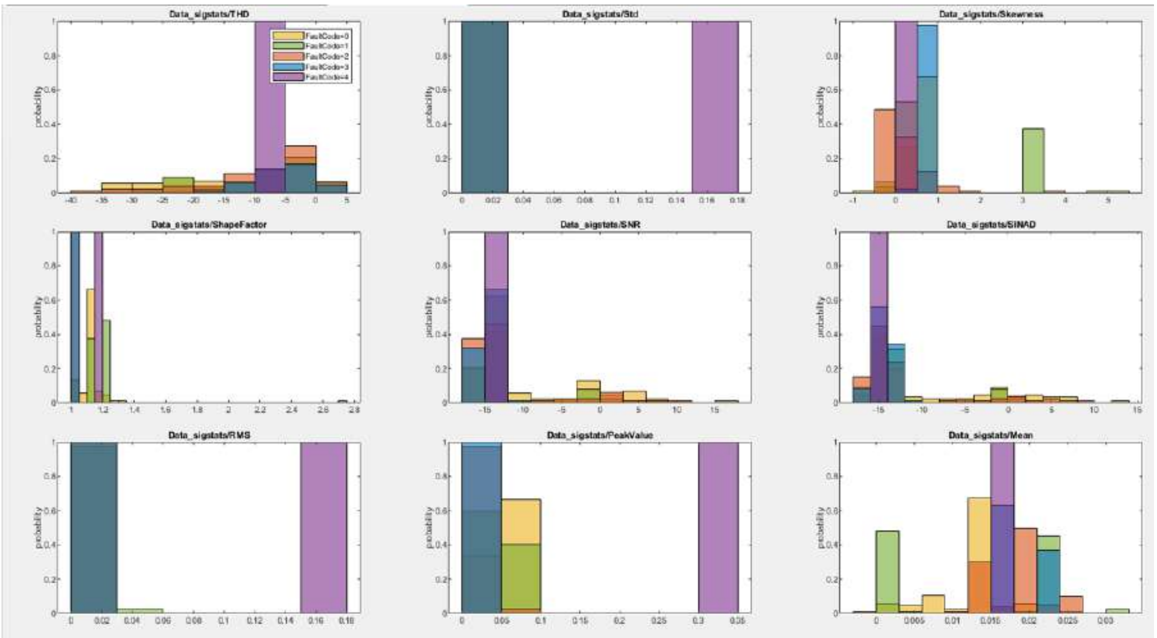


Figure 81: Histograms of Sound

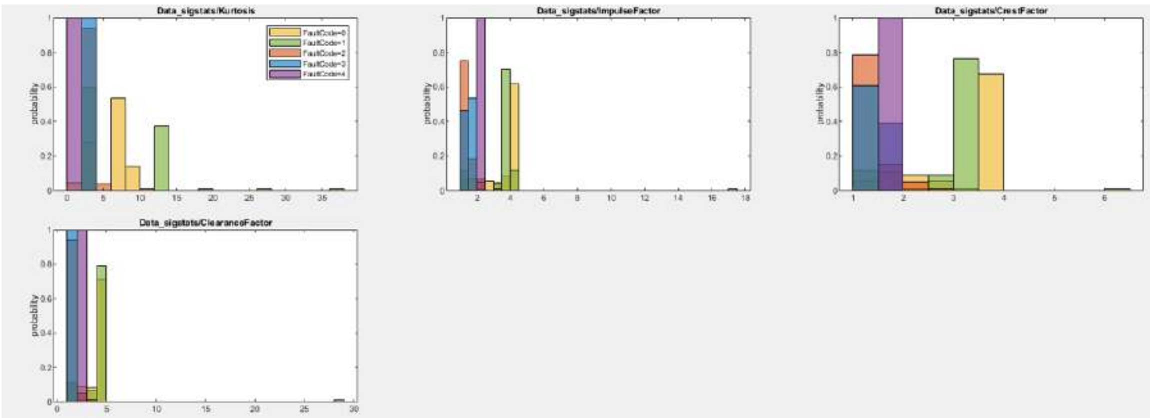


Figure 82: Histograms of Sound

7.4 Results

By integrating Acoustic Emission data alongside vibrational data, the machine learning models developed within MATLAB's Classification Learner app gain a more comprehensive understanding of the motor's health status.selected models: Fine Gaussian SVM, Fine KNN, Ensemble (Bagged Tree), and Medium Neural Network. These models represent a range of machine learning algorithms suitable for classification tasks, including support vector machines, k-nearest neighbors, ensemble methods, and neural networks.

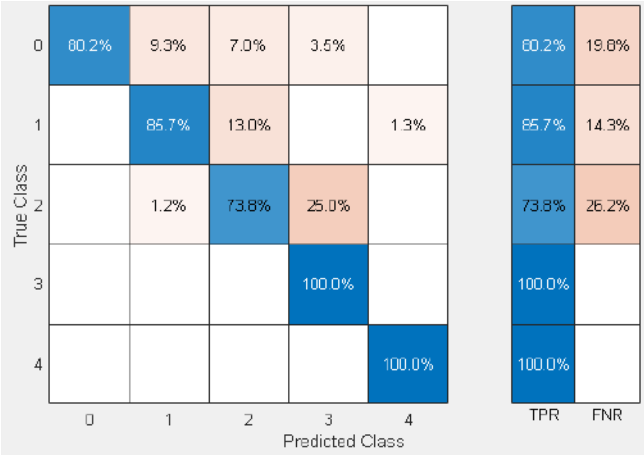


Figure 83: SVM Acoustic Confusion matrix

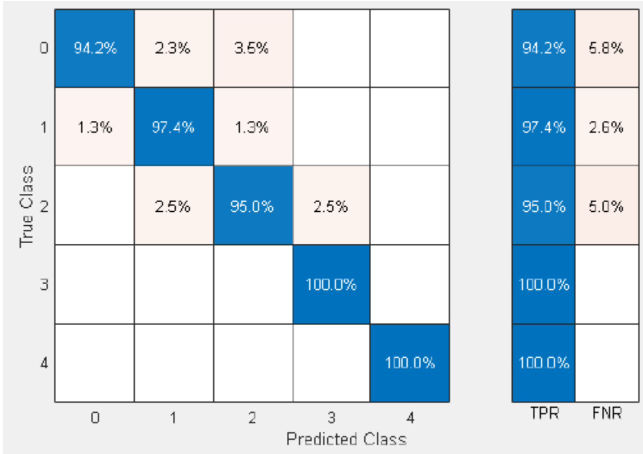


Figure 84: KNN Acoustic Confusion matrix

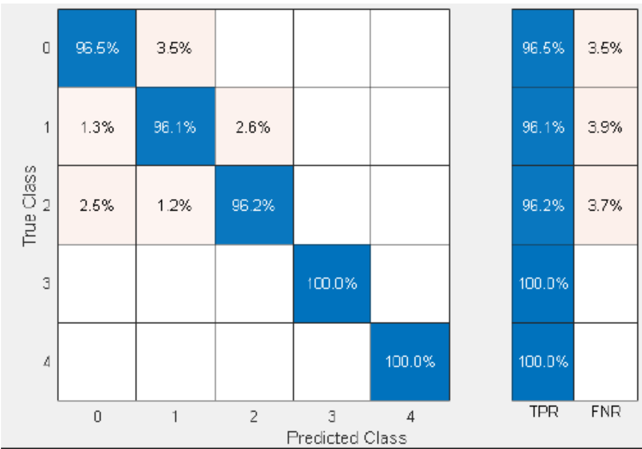


Figure 85: Ensemble Acoustic Confusion matrix

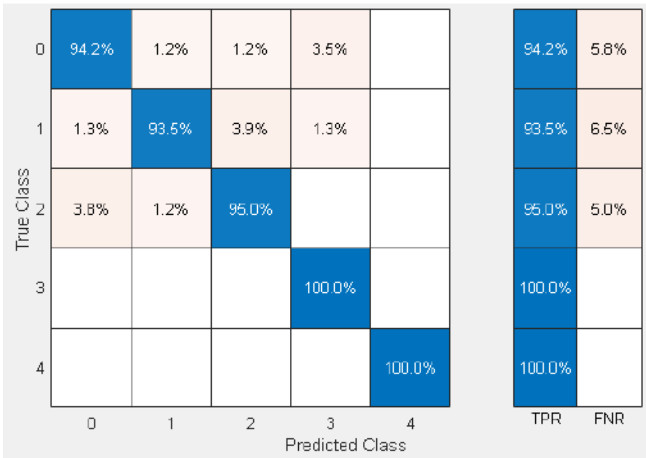


Figure 86: Neural Acoustic Confusion matrix

Chapter 8

8 Motor Current Signature Analysis Setup

8.1 Studies Relevant to Bearing Faults

Numerous studies have investigated bearing faults in various contexts, including those related to induction motors. Here are some relevant studies along with their references:

- ”Bearing Fault Diagnosis Using Time-Frequency Analysis Techniques: A Comprehensive Review” A comprehensive overview of time-frequency analysis techniques for bearing fault diagnosis, covering methodologies, signal processing algorithms, and case studies.Hilbert-Huang transform, and empirical mode decomposition in detecting and diagnosing bearing faults in induction motors.

- ”Fault diagnosis of induction motor bearings using vibration analysis based on ensemble empirical mode decomposition and random forests” Fault diagnosis approach for induction motor bearings using vibration analysis based on ensemble empirical mode decomposition and random forests. It investigates the effectiveness of the proposed method in identifying different types of bearing faults, including inner race, outer race

- ”Bearing Fault Diagnosis of Induction Motors Using Motor Current Signature Analysis and Convolutional Neural Networks” Fault diagnosis approach for induction motor bearings utilizing MCSA combined with convolutional neural networks (CNNs).Use of CNNs for automated fault classification, achieving high accuracy in fault detection.

8.2 Analysis of MCSA Signal Trace

Once the data is loaded, you can visualize the MCSA signal trace by selecting it from the list of available signals in the app.Navigate through the signal trace to explore its characteristics.

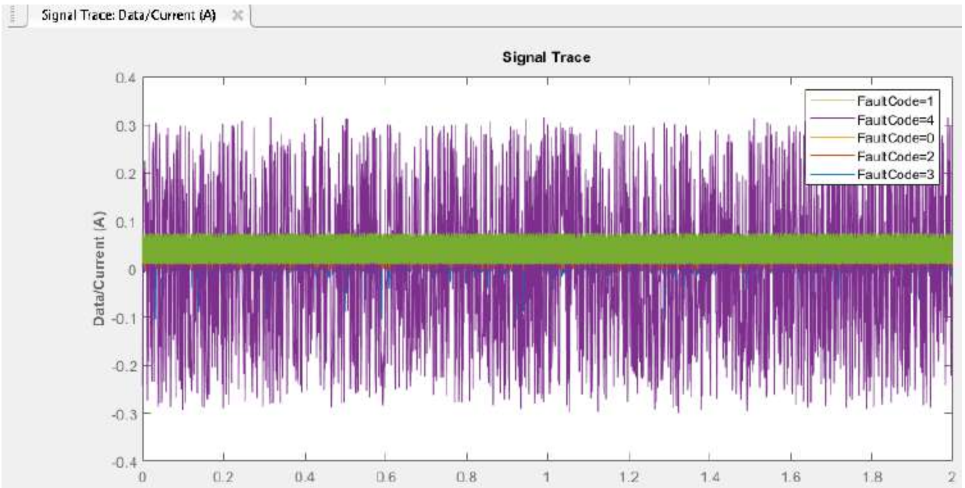


Figure 87: Current Signal

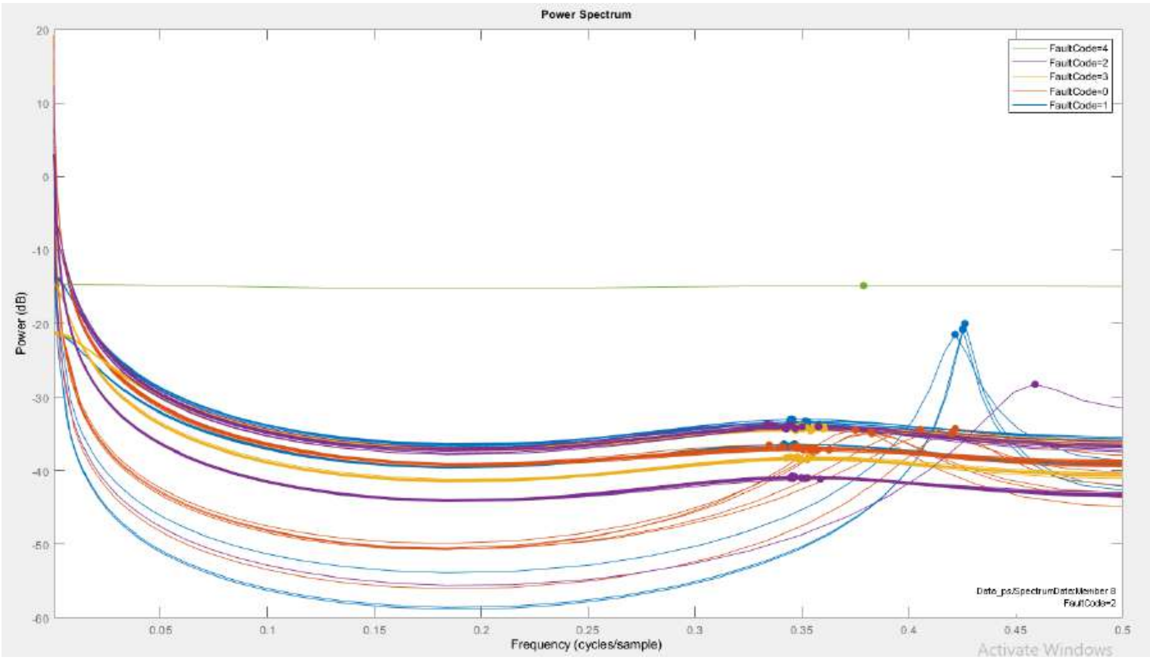
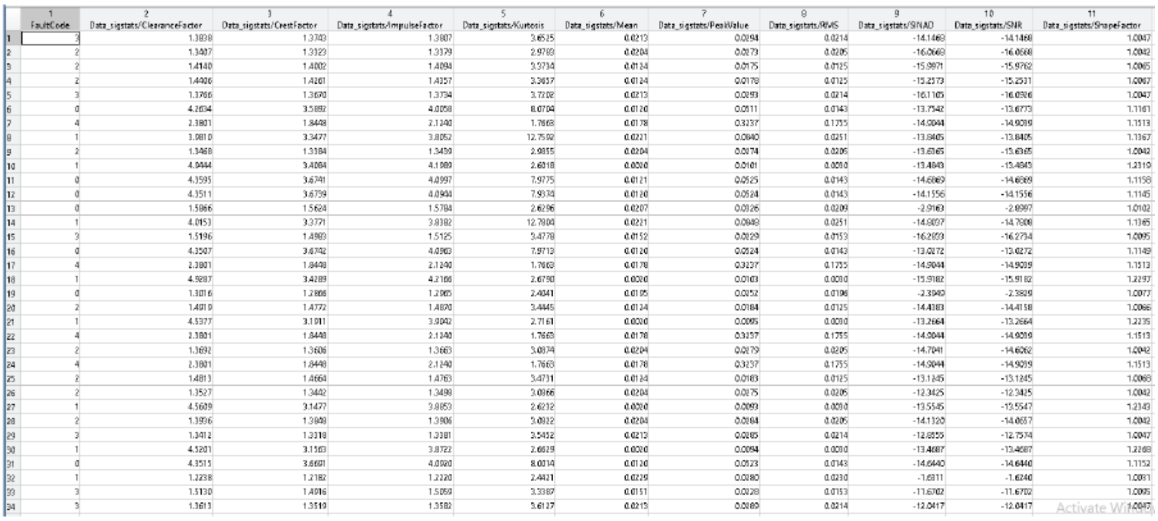


Figure 88: Power spectrum of current

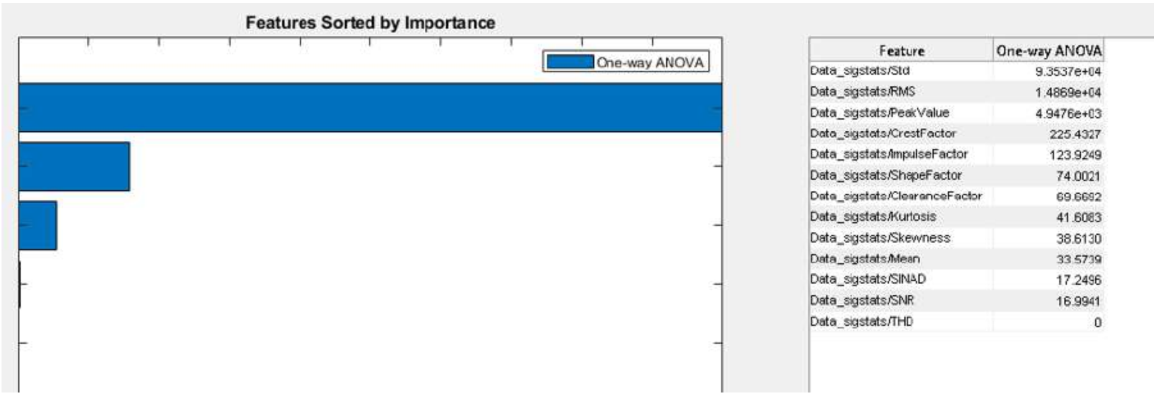
8.3 Analysis of Data features

Use the feature extraction tools provided in the Diagnostic Feature Designer app to extract relevant features from the MCSA signal trace. These features may include time-domain features such as mean, standard deviation, root mean square (RMS), skewness, and kurtosis, as well as frequency-domain features obtained through Fourier analysis or wavelet transform.

Frequency domain fft result:-



	1 FaultCode	2 Data_sigstats/ClearanceFactor	3 Data_sigstats/CrestFactor	4 Data_sigstats/ImpulseFactor	5 Data_sigstats/Kurtosis	6 Data_sigstats/Mean	7 Data_sigstats/PeakValue	8 Data_sigstats/RMS	9 Data_sigstats/SINAD	10 Data_sigstats/SNR	11 Data_sigstats/ShapeFactor
1	3	1.3838	1.3743	1.3807	3.6525	0.0213	0.0294	0.0214	-14.1468	-14.1468	1.0047
2	2	1.3407	1.3323	1.3379	2.9783	0.0204	0.0273	0.0205	-16.0668	-16.0668	1.0042
3	2	1.4140	1.4002	1.4094	3.3734	0.0124	0.0175	0.0125	-15.9971	-15.9762	1.0065
4	2	1.4406	1.4261	1.4357	3.3657	0.0124	0.0178	0.0125	-15.2573	-15.2531	1.0067
5	3	1.3766	1.3670	1.3734	3.7202	0.0213	0.0293	0.0214	-16.1105	-16.0926	1.0047
6	0	4.2634	3.5892	4.0058	8.0704	0.0120	0.0511	0.0143	-13.7542	-13.6773	1.1161
7	4	2.3801	1.8448	2.1240	1.7663	0.0178	0.3237	0.1755	-14.9044	-14.9039	1.1513
8	1	3.9810	3.3477	3.8052	12.7592	0.0221	0.0840	0.0251	-13.8405	-13.8405	1.1367
9	2	1.3468	1.3184	1.3439	2.9855	0.0204	0.0274	0.0206	-13.6365	-13.6365	1.0042
10	1	4.9444	3.4084	4.1989	2.6018	0.0020	0.0101	0.0030	-13.4843	-13.4843	1.2319
11	0	4.3595	3.6741	4.0997	7.9775	0.0121	0.0525	0.0143	-14.6869	-14.6869	1.1158
12	0	4.3511	3.6739	4.0944	7.9374	0.0120	0.0524	0.0143	-14.1556	-14.1556	1.1145
13	0	1.5866	1.5624	1.5784	2.6296	0.0207	0.0326	0.0209	-2.9163	-2.8997	1.0102
14	1	4.0153	3.3771	3.8382	12.7804	0.0221	0.0848	0.0251	-14.8037	-14.7808	1.1365
15	3	1.5196	1.4983	1.5125	3.4778	0.0152	0.0229	0.0153	-16.2833	-16.2734	1.0095
16	0	4.3507	3.6742	4.0963	7.9713	0.0120	0.0524	0.0143	-13.0272	-13.0272	1.1149
17	4	2.3801	1.8448	2.1240	1.7663	0.0178	0.3237	0.1755	-14.9044	-14.9039	1.1513
18	1	4.9287	3.4289	4.2166	2.6790	0.0020	0.0103	0.0030	-15.9182	-15.9182	1.2297
19	0	1.3016	1.2886	1.2965	2.4041	0.0195	0.0252	0.0196	-2.3940	-2.3829	1.0077
20	2	1.4919	1.4772	1.4870	3.4445	0.0124	0.0184	0.0125	-14.4383	-14.4158	1.0066
21	1	4.5377	3.1911	3.9042	2.7161	0.0020	0.0095	0.0030	-13.2664	-13.2664	1.2235
22	4	2.3801	1.8448	2.1240	1.7663	0.0178	0.3237	0.1755	-14.9044	-14.9039	1.1513
23	2	1.3692	1.3606	1.3663	3.0874	0.0204	0.0279	0.0205	-14.7041	-14.6062	1.0042
24	4	2.3801	1.8448	2.1240	1.7663	0.0178	0.3237	0.1755	-14.9044	-14.9039	1.1513
25	2	1.4813	1.4664	1.4763	3.4731	0.0124	0.0183	0.0125	-13.1245	-13.1245	1.0068
26	2	1.3527	1.3442	1.3498	3.0866	0.0204	0.0275	0.0206	-12.3425	-12.3425	1.0042
27	1	4.5609	3.1477	3.8853	2.6232	0.0020	0.0093	0.0030	-13.5545	-13.5547	1.2343
28	2	1.3936	1.3848	1.3906	3.0822	0.0204	0.0284	0.0206	-14.1320	-14.0657	1.0042
29	3	1.3412	1.3318	1.3381	3.5452	0.0213	0.0285	0.0214	-12.8555	-12.7574	1.0047
30	1	4.5201	3.1563	3.8722	2.6629	0.0020	0.0094	0.0030	-13.4687	-13.4687	1.2268
31	0	4.3515	3.6691	4.0920	8.0014	0.0120	0.0523	0.0143	-14.6440	-14.6440	1.1152
32	1	1.2238	1.2182	1.2220	2.4421	0.0229	0.0280	0.0230	-1.6811	-1.6240	1.0031
33	3	1.5130	1.4916	1.5059	3.3387	0.0151	0.0228	0.0153	-11.6702	-11.6702	1.0095
34	3	1.3613	1.3519	1.3582	3.6127	0.0213	0.0280	0.0214	-12.0417	-12.0417	1.0047



Feature	One-way ANOVA
Data_sigstats/Std	9.3537e+04
Data_sigstats/RMS	1.4869e+04
Data_sigstats/PeakValue	4.9476e+03
Data_sigstats/CrestFactor	225.4327
Data_sigstats/ImpulseFactor	123.9249
Data_sigstats/ShapeFactor	74.0021
Data_sigstats/ClearanceFactor	69.6692
Data_sigstats/Kurtosis	41.6083
Data_sigstats/Skewness	38.6130
Data_sigstats/Mean	33.5739
Data_sigstats/SINAD	17.2496
Data_sigstats/SNR	16.9941
Data_sigstats/THD	0

- Outer Race Fault signal of fft is :

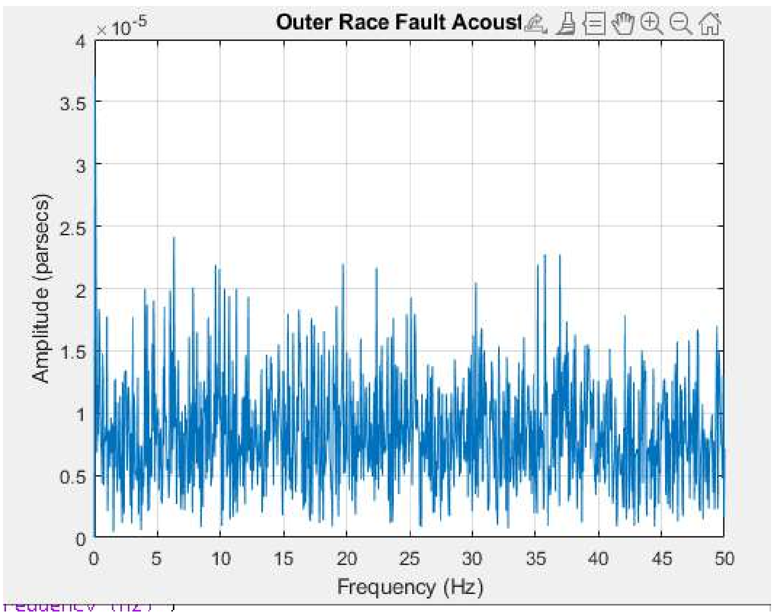


Figure 89: FFT of Outer Race condition of Current

- Inner Race Fault signal of fft is :

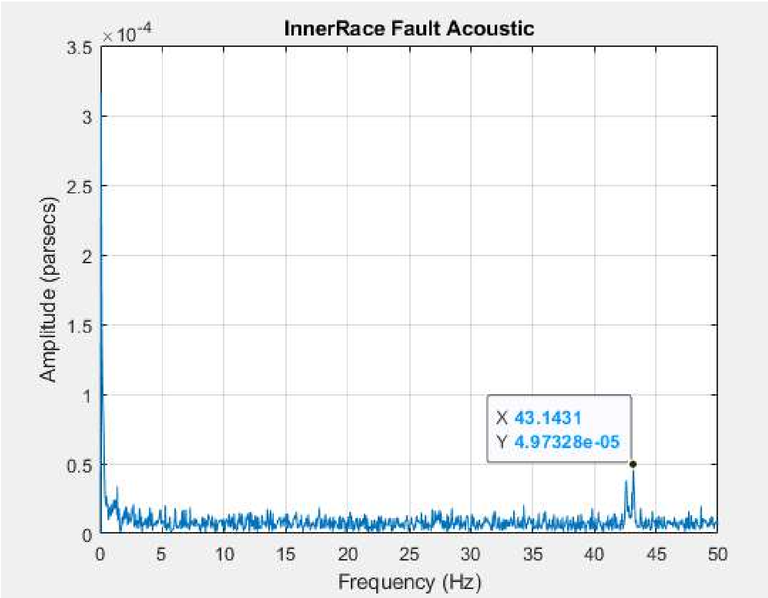


Figure 90: Inner Race Fault condition of Current

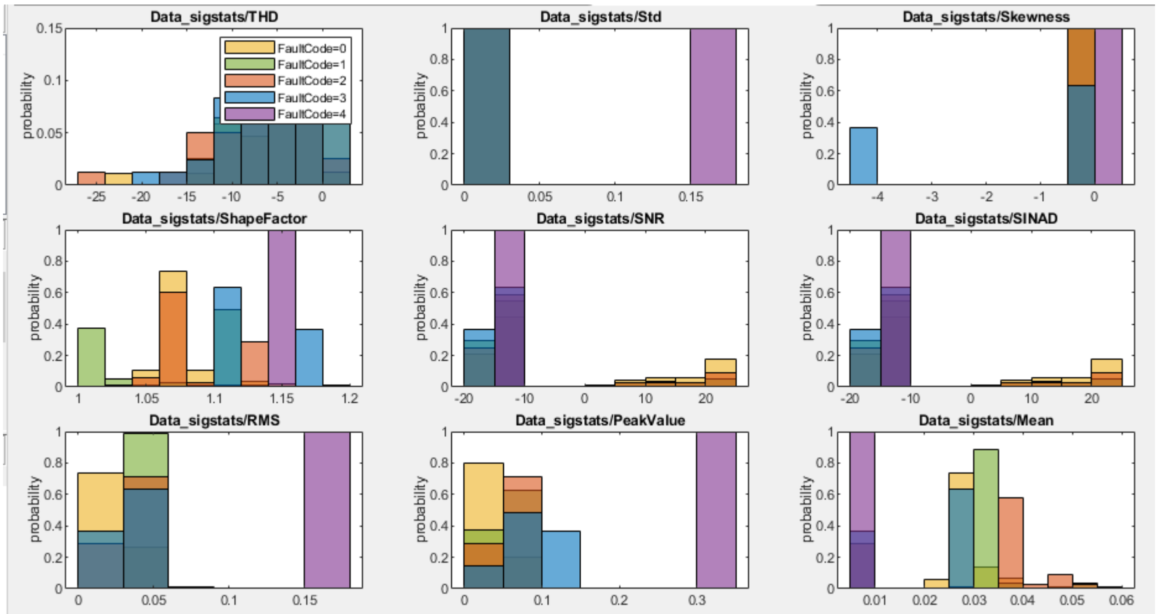


Figure 91: Histograms of Current

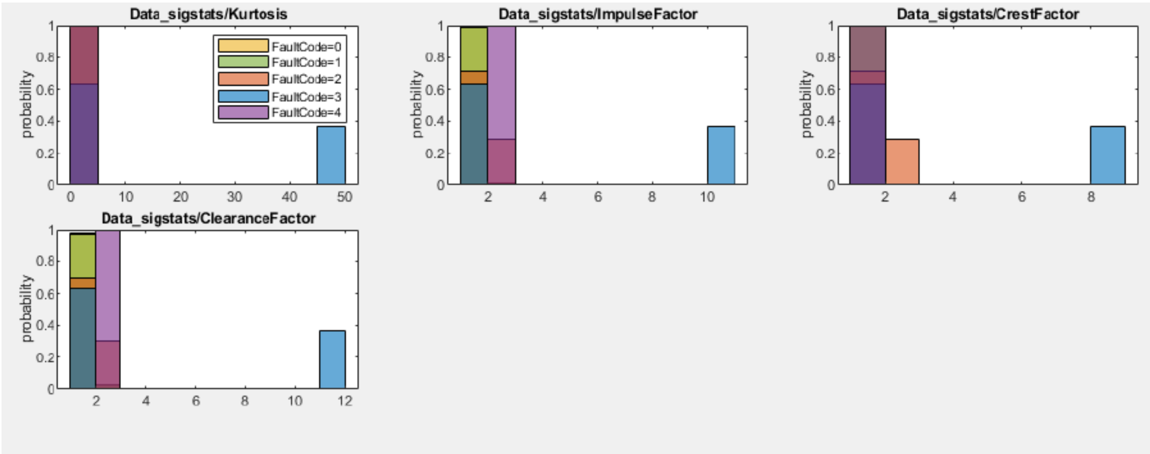


Figure 92: Histograms of Current

8.4 Results

By MCSA data the machine learning models developed within MATLAB's Classification Learner app gain a comprehensive fault indicator of the motor's health status.selected models: Fine Gaussian SVM, Fine KNN, Ensemble (Bagged Tree), and Medium Neural Network. These models represent a range of machine learning algorithms suitable for classification tasks, including support vector machines, k-nearest neighbors, ensemble methods, and neural networks.

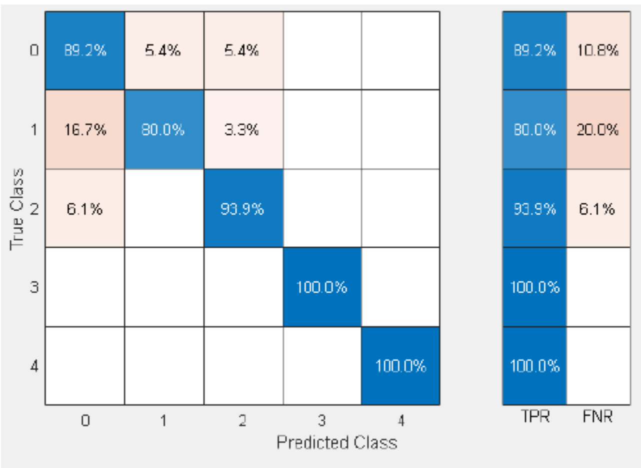


Figure 93: SVM MCSA Confusion matrix

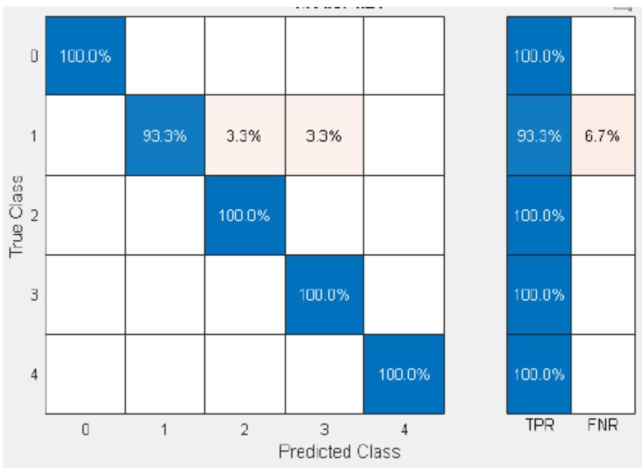


Figure 94: KNN MCSA Confusion matrix

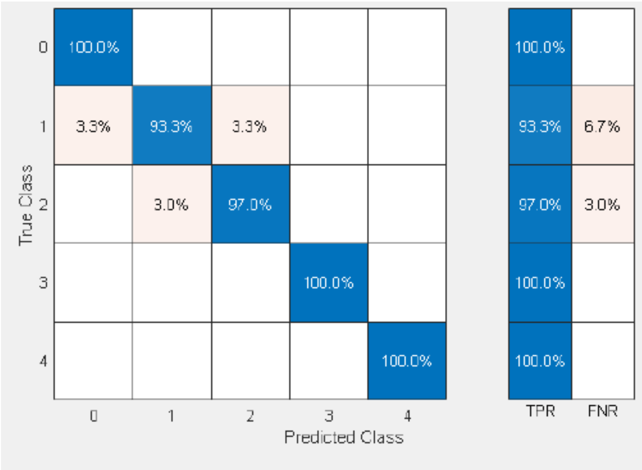


Figure 95: Ensemble MCSA Confusion matrix

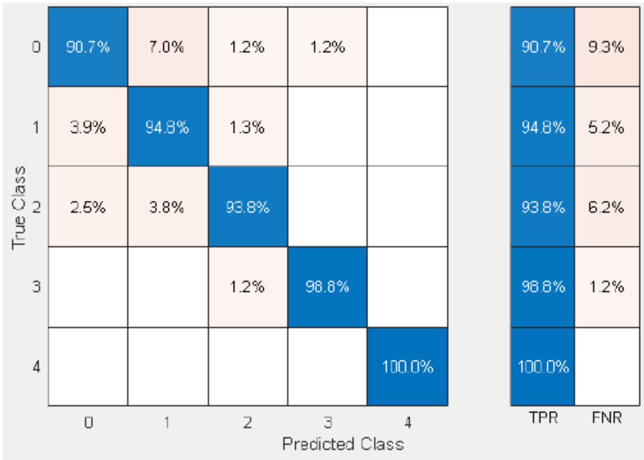


Figure 96: Neural MCSA Confusion matrix

Chapter 9

9 Comparative Analysis

Comparing the bearing data acquisition methods through accelerometer, acoustic, and current data (MCSA technique) involves to indicate or predict the faults in induction motors. Each method has its Pros and cons, which influence their performance for machine learning model-based fault identification.

9.1 vibrational Technique

Pros:Accelerometer data provides information about vibration patterns, which can indicate mechanical faults such as bearing faults.Cons:Accelerometer data may be sensitive to external vibrations and environmental noise, which can affect signal accuracy. Additionally, certain fault types may not manifest significant vibration signatures.

9.2 Acoustic Technique

Pros:Acoustic data captures sound emissions generated by bearing faults, offering complementary information to vibration-based methods. It can detect faults such as surface defects or lubrication issues.Cons: Acoustic data may be influenced by ambient noise in the environment, impacting signal clarity. Additionally, it may be less sensitive to certain fault types compared to vibration or current data.

9.3 Current Technique

Pros:MCSA provides direct insights into the electrical current flowing through the motor, detecting faults such as rotor bar defects, stator winding faults, and bearing defects. It offers high sensitivity and specificity to motor-related faults.Cons:MCSA requires specialized equipment and may not detect certain mechanical faults that do not significantly affect electrical current.

MCSA data is frequently seen to be more accurate in identifying defects in induction motors when it comes to machine learning model-based fault identification. This is due to the reason that MCSA measures the motor's electrical properties directly, which are directly linked with both its fault status and operational state. For operations involving fault diagnosis, MCSA is a dependable option due to its high sensitivity and specificity to motor-related faults.

9.4 Accuracy

Accuracy				
Technique	SVM(fine gaussian)	Ensemble	Fine KNN	Neural Network
X	83%	88.3%	92%	79.3%
Y	89.1%	91.2%	94.3%	93.6%
Z	94.2%	99.3%	89.6%	96.7%
Acoustic	91.8%	97.4%	97.8%	94.5%
Current	99.3%	99.6%	98.6%	98.7%

Chapter 10

10 Development of Graphical User Interface(GUI) for Condition Monitoring

At the end, design the GUI (Graphical User Interface) for to show the output result of fault diagnosis of Induction Motor. GUI allow users to customize the display and adjust parameters, while alerts and notifications provide timely information on critical faults of the Motor.

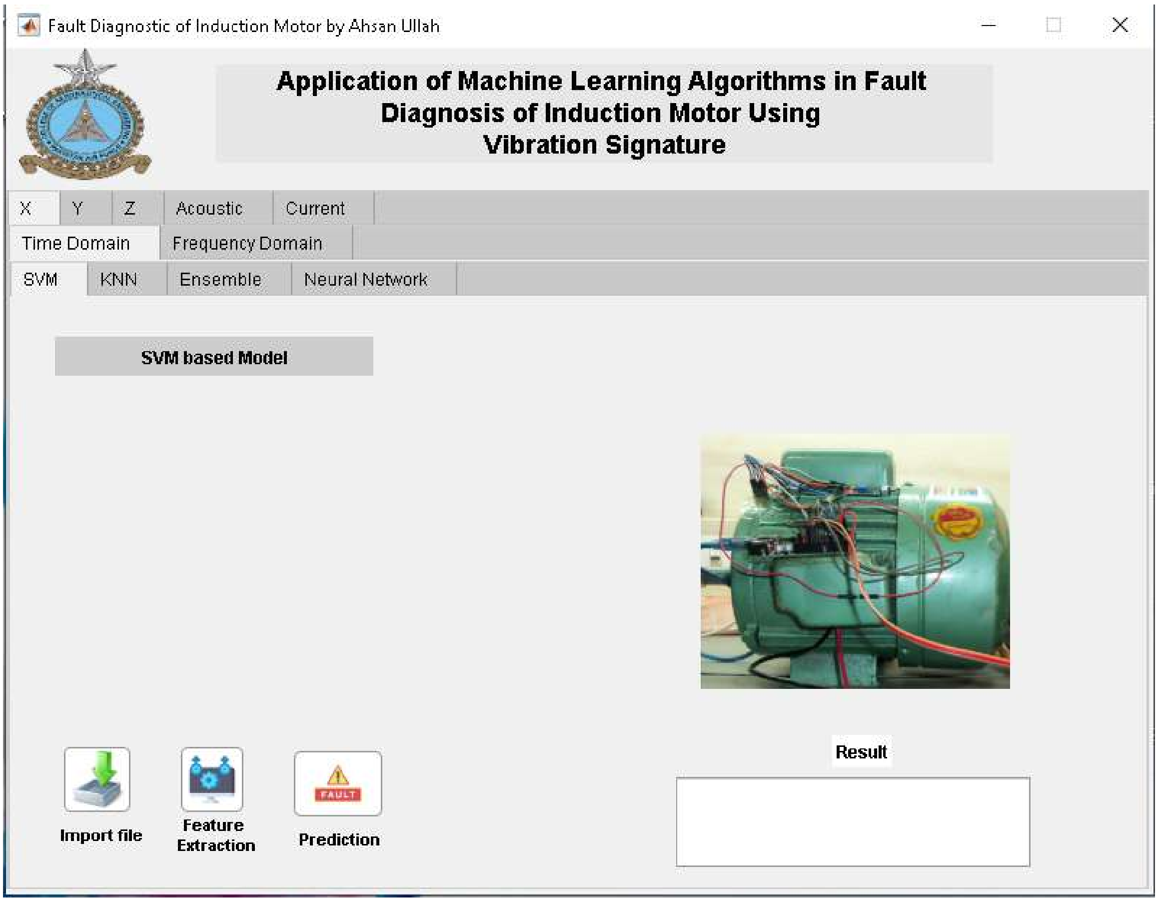


Figure 97: Graphical User Interface(GUI)

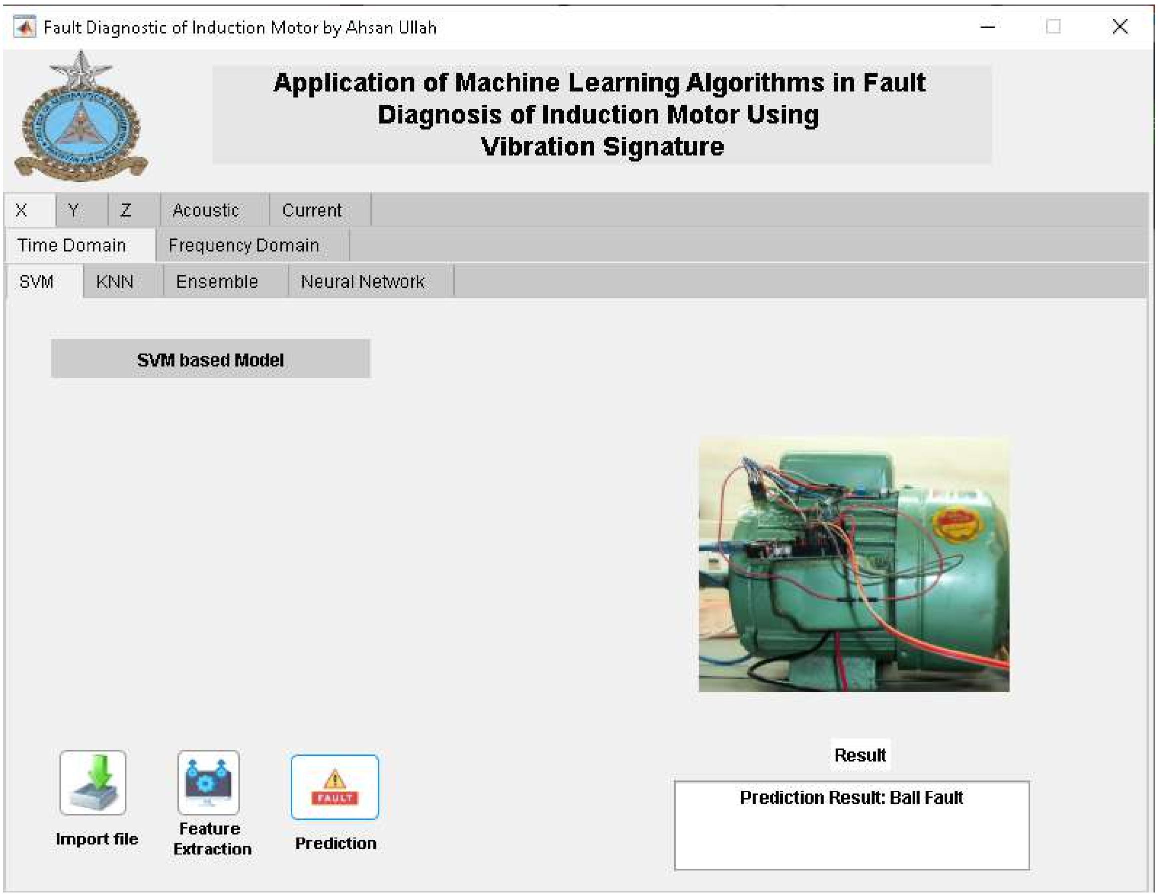


Figure 98: Graphical User Interface(GUI) with Result

Chapter 9

11 Conclusion and Future Work

11.1 Conclusion

In conclusion, the study proposes a practical machine learning-based fault diagnosis method for induction motors using experimental data, focusing on condition monitoring through Motor Current Signature Analysis (MCSA), vibrational, and acoustic emission analysis of bearing faults. By utilizing two identical single-phase induction motors—one for healthy data acquisition and the other for faulty data acquisition—the project effectively captures baseline and faulty data under various operating conditions. Through the analysis of time and frequency domain features extracted from MATLAB, the study evaluates the performance of three classification algorithms—Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Ensemble—using MATLAB Classification Learner toolbox. The results demonstrate the suitability of machine learning techniques in accurately predicting motor conditions (healthy or faulty) By using advanced fault diagnosis methodologies, this approach contributes to proactive maintenance strategies, minimizing downtime, and optimizing motor performance in critical applications.

11.2 Future Work

To further enhance the effectiveness of fault diagnosis of induction motors using bearing, acoustic, and current signature data, the following recommendation for the future:

- Proposed techniques may be applied for fault diagnosis of large size motors.
- It is the diagnostic approach toward fault prediction ,in future work on Prognostic Technique.
- Many other Faults can be investigate like Stator winding,broken rotor bar , electric faults etc
- Integration of advanced signal processing techniques, such as deep learning models

- Multi-Sensor data fusion technique Approach
- Applied RSM (Response Surface Methology)

Appendices

A Program Code

A.1 Code for Scenario 01 :Data Acquisition Using Arduino uno to store data from sensors

```

const int xPin = A0;      // Connect the X-axis output to Arduino A0
const int yPin = A1;      // Connect the Y-axis output to Arduino A1
const int zPin = A2;      // Connect the Z-axis output to Arduino A2
const int soundSensorPin = A3; // Connect the sound sensor's analog output to A3
const int currentSensorPin = A4; // Connect the ACS712 current sensor's analog output to
↪ A4

const int sampleCount = 100; // Number of samples for RMS calculation

int xValues[sampleCount];
int yValues[sampleCount];
int zValues[sampleCount];
int soundValues[sampleCount];
int currentValues[sampleCount];

int currentIndex = 0;

void setup() {
  Serial.begin(9600);
  pinMode(soundSensorPin, INPUT); // Set the sound sensor pin as an input
}

bool printedHeadings = false;

void loop() {
  int xValue = analogRead(xPin);
  int yValue = analogRead(yPin);
  int zValue = analogRead(zPin);
  int soundValue = analogRead(soundSensorPin);
  int currentRawValue = analogRead(currentSensorPin);

  // Store the readings in arrays
  xValues[currentIndex] = xValue;
```

```
  yValues[currentIndex] = yValue;
  zValues[currentIndex] = zValue;
  soundValues[currentIndex] = soundValue;
  currentValues[currentIndex] = currentRawValue;

  // Move to the next index (circular buffer)
  currentIndex = (currentIndex + 1) % sampleCount;

  if (!printedHeadings) {
    // Print the headings only once
    Serial.println("\tX\t\tY\t\tZ\t\tSound\t\tCurrent (A)");
    printedHeadings = true;
  }

  if (currentIndex == 0) {
    // Calculate RMS values for each axis
    float rmsX = calculateRMS(xValues, sampleCount);
    float rmsY = calculateRMS(yValues, sampleCount);
    float rmsZ = calculateRMS(zValues, sampleCount);

    // Calculate RMS value for the sound sensor
    float rmsSound = calculateRMS(soundValues, sampleCount);

    // Calculate RMS value for the current sensor
    float rmsCurrent = calculateRMS(currentValues, sampleCount);

    // Print the RMS values below their respective headings
   Serial.print('\t'); // Tab separator
    Serial.print(rmsX-2.4, 6);
    Serial.print('\t'); // Tab separator
    Serial.print(rmsY-1.50, 6);
    Serial.print('\t'); // Tab separator
    Serial.print(rmsZ-1.32, 6);
    Serial.print('\t'); // Tab separator
    Serial.print(rmsSound-0.20, 6);
    Serial.print('\t'); // Tab separator
    Serial.println(rmsCurrent-2.48, 6);
  }

  // Delay as needed
```

```
  //delay(10); // Adjust the sampling interval as needed
}

// Function to calculate the RMS value of an array of samples
float calculateRMS(int values[], int count) {
  float sumOfSquares = 0.0;

  for (int i = 0; i < count; i++) {
    float voltage = (values[i] / 1023.0) * 5.0; // Convert to voltage
    sumOfSquares += voltage * voltage;
  }

  float rms = sqrt(sumOfSquares / count);
  return rms;
}
```

A.2 Code for Scenario 02: Making memtable and labelling the data

```

clc
close all
clear all

% Initialize arrays to store data and fault codes
dataCells = cell(600, 1);
faultCode = [zeros(120, 1); ones(120, 1);2*ones(120,1);3*ones(120,1);4*ones(120,1)];

% Process healthy data
for i = 1:120
    filename = sprintf('modified_healthy_%d.xlsx', i);
   a_i = readtable(filename);
   %timeVector = seconds(a_i.Time);
   % aa_i = removevars(a_i, 'Time');
    %aaa_i = table2timetable(a_i, 'RowTimes');
    a_i.Properties.VariableNames = {'Y' };

    dataCells{i} = a_i;
end

% Process outer_race faulty data
for j = 1:120
    filename = sprintf('modified_outer_race_fault_%d.xlsx', j);
    f_j = readtable(filename);
  % timeVector = seconds(f_j.Time);
   % ff_j = removevars(f_j, 'Time');
   % fff_j = table2timetable(f_j, 'RowTimes', timeVector);
    f_j.Properties.VariableNames = {'Y'};

    dataCells{120 + j} = f_j;
end

% Process inner_race data
for k = 1:120
    filename = sprintf('modified_inner_race_fault_%d.xlsx', k);
    b_k = readtable(filename);
```

```
   % timeVector = seconds(b_k.Time);
   % bb_k = removevars(b_k, 'Time');
    %bbb_k = table2timetable(b_k, 'RowTimes', timeVector);
    b_k.Properties.VariableNames = {'Y'};

    dataCells{240 + k} = b_k;
end

% Process ball_fault data
for l = 1:120
    filename = sprintf('modified_ball_fault_%d.xlsx', l);
    c_l = readtable(filename);
    %timeVector = seconds(c_l.Time);
    %cc_l= removevars(c_l, 'Time');
    %ccc_l = table2timetable(c_l, 'RowTimes', timeVector);
    c_l.Properties.VariableNames = {'Y'};

    dataCells{360 + l} = c_l;
end
% Process compound_fault data
for m = 1:120
    filename = sprintf('compound_fault_%d.xlsx', l);
    d_m = readtable(filename);
    %timeVector = seconds(c_l.Time);
    %cc_l= removevars(c_l, 'Time');
    %ddd_m = table2timetable(d_m, 'RowTimes', timeVector);
    d_m.Properties.VariableNames = {'Y'};

    dataCells{480 + m} = d_m;
end
% Create memtable by concatenating data and fault code
memtable = table(dataCells, faultCode, 'VariableNames', {'Data', 'FaultCode'});

% Set the random seed for reproducibility
rng(30);

% Shuffle indices for random split
indices = randperm(size(memtable, 1));

% Calculate the number of samples for training (70%) and testing (30%)
```

```
training_samples = round(0.7 * size(memtable, 1));
testing_samples = size(memtable, 1) - training_samples;

% Split the data and fault codes
training_data = memtable(indices(1:training_samples), :);
testing_data = memtable(indices(training_samples+1:end), :);

% Verify the separation of faults and healthy data in training and testing sets
disp('Training Set Distribution:');
disp(tabulate(training_data.FaultCode));

disp('Testing Set Distribution:');
disp(tabulate(testing_data.FaultCode));

```

A.3 Code for Scenario 03:Time Domain Feature Extraction of the Data

---

```
function [featureTable,outputTable] = diagnosticFeatures(testing_data)
%DIAGNOSTICFEATURES recreates results in Diagnostic Feature Designer.
%
% Input:
%  inputData: A table or a cell array of tables/matrices containing the
%  data as those imported into the app.
%
% Output:
%  featureTable: A table containing all features and condition variables.
%  outputTable: A table containing the computation results.
%
% This function computes spectra:
%  Data_ps/SpectrumData
%
% This function computes features:
%  Data_ps_spec/PeakAmp1
%  Data_ps_spec/PeakFreq1
%  Data_ps_spec/BandPower
%
% Organization of the function:
% 1. Compute signals/spectra/features
% 2. Extract computed features into a table
%
% Modify the function to add or remove data processing, feature generation
% or ranking operations.

% Auto-generated by MATLAB on 07-Feb-2024 02:16:23

% Create output ensemble.
outputEnsemble =
↪  workspaceEnsemble(testing_data,'DataVariables',"Data",'ConditionVariables',"FaultCode");

% Reset the ensemble to read from the beginning of the ensemble.
reset(outputEnsemble);

% Append new signal or feature names to DataVariables.
```

```
outputEnsemble.DataVariables = unique([outputEnsemble.DataVariables;"Data_ps";"Data_ps_spec"],'stable');

% Set SelectedVariables to select variables to read from the ensemble.
outputEnsemble.SelectedVariables = "Data";

% Loop through all ensemble members to read and write data.
while hasdata(outputEnsemble)
    % Read one member.
    member = read(outputEnsemble);

    % Get all input variables.
    Data = readMemberData(member,"Data");
    iv = (0:1:(height(Data)-1)*1)';
    Data.Sample = iv;

    % Initialize a table to store results.
    memberResult = table;

    %% PowerSpectrum
    try
        % Get units to use in computed spectrum.
        tuReal = "samples";
        tuTime = "seconds";

        % Compute effective sampling rate.
        tNumeric = time2num(Data.Sample,tuReal);
        [Fs,irregular] = effectivefs(tNumeric);
        Ts = 1/Fs;

        % Resample non-uniform signals.
        x = Data.Y;
        if irregular
            x = resample(x,tNumeric,Fs,'linear');
        end

        % Compute the autoregressive model.
        data = iddata(x,[],Ts,'TimeUnit',tuTime,'OutputName','SpectrumData');
        arOpt = arOptions('Approach','fb','Window','now','EstimateCovariance',false);
        model = ar(data,4,arOpt);
```

```
        % Compute the power spectrum.
        [ps,w] = spectrum(model);
        ps = reshape(ps, numel(ps), 1);
        factor = 1/2/pi
        w = factor*w;

        % Remove frequencies above Nyquist frequency.
        I = w<=(Fs/2+1e4*eps);
        w = w(I);
        ps = ps(I);

        % Configure the computed spectrum.
        ps = table(w, ps, 'VariableNames', ["Frequency", "SpectrumData"]);
        ps.Properties.VariableUnits = ["cycles/sample", ""];
        ps = addprop(ps, {'SampleFrequency'}, {'table'});
        ps.Properties.CustomProperties.SampleFrequency = Fs;
        Data_ps = ps;
    catch
        Data_ps = table(NaN, NaN, 'VariableNames', ["Frequency", "SpectrumData"]);
    end

    % Append computed results to the member table.
    memberResult = [memberResult, ...
        table({Data_ps},'VariableNames',"Data_ps")]; %#ok<AGROW>

    %% SpectrumFeatures
    try
        % Compute spectral features.
        % Get frequency unit conversion factor.
        factor = 2*pi;
        ps = Data_ps.SpectrumData;
        w = Data_ps.Frequency;
        w = factor*w;
        mask_1 = (w>=factor*1.59154943091895e-06) & (w<=factor*0.5);
        ps = ps(mask_1);
        w = w(mask_1);

        % Compute spectral peaks.
        [peakAmp,peakFreq] = findpeaks(ps,w/factor,'MinPeakHeight',-Inf, ...
```

```
    ↪ 'MinPeakProminence',0,'MinPeakDistance',0.001,'SortStr','descend','NPeaks',2);
        peakAmp = [peakAmp(:); NaN(2-numel(peakAmp),1)];
        peakFreq = [peakFreq(:); NaN(2-numel(peakFreq),1)];

        % Extract individual feature values.
        PeakAmp1 = peakAmp(1);
        PeakFreq1 = peakFreq(1);
        BandPower = trapz(w/factor,ps);

        % Concatenate signal features.
        featureValues = [PeakAmp1,PeakFreq1,BandPower];

        % Package computed features into a table.
        featureNames = ["PeakAmp1","PeakFreq1","BandPower"];
        Data_ps_spec = array2table(featureValues,'VariableNames',featureNames);
    catch
        % Package computed features into a table.
        featureValues = NaN(1,3);
        featureNames = ["PeakAmp1","PeakFreq1","BandPower"];
        Data_ps_spec = array2table(featureValues,'VariableNames',featureNames);
    end

    % Append computed results to the member table.
    memberResult = [memberResult, ...
        table({Data_ps_spec},'VariableNames',"Data_ps_spec")]; %#ok<AGROW>

    %% Write all the results for the current member to the ensemble.
    writeToLastMemberRead(outputEnsemble,memberResult)
end

% Gather all features into a table.
featureTable = readFeatureTable(outputEnsemble);

% Set SelectedVariables to select variables to read from the ensemble.
outputEnsemble.SelectedVariables =
↪ unique([outputEnsemble.DataVariables;outputEnsemble.ConditionVariables;outputEnsemble.Independer

% Gather results into a table.
outputTable = readall(outputEnsemble);
```

```
end
```

A.4 Code for Scenario 04: Frequency Domain Feature Extraction

```
function [featureTable,ranking,outputTable] = diagnosticFeatures(inputData)
%DIAGNOSTICFEATURES recreates results in Diagnostic Feature Designer.
%
% Input:
%  inputData: A table or a cell array of tables/matrices containing the
%  data as those imported into the app.
%
% Output:
%  featureTable: A table containing all features and condition variables.
%  ranking: A table containing ranking scores for selected features.
%  outputTable: A table containing the computation results.
%
% This function computes spectra:
%  Data_ps/SpectrumData
%
% This function computes features:
%  Data_ps_spec/PeakAmp1
%  Data_ps_spec/PeakFreq1
%  Data_ps_spec/BandPower
%
% This function ranks computed feautres using algorithms:
%  One-way ANOVA
%
% Organization of the function:
% 1. Compute signals/spectra/features
% 2. Extract computed features into a table
% 3. Rank features
%
% Modify the function to add or remove data processing, feature generation
% or ranking operations.

% Auto-generated by MATLAB on 07-Feb-2024 00:40:51

% Create output ensemble.
outputEnsemble =
↪  workspaceEnsemble(inputData,'DataVariables',"Data",'ConditionVariables',"FaultCode");
```

```
% Reset the ensemble to read from the beginning of the ensemble.
reset(outputEnsemble);

% Append new signal or feature names to DataVariables.
outputEnsemble.DataVariables =
↪  unique([outputEnsemble.DataVariables;"Data_ps";"Data_ps_spec"],'stable');

% Set SelectedVariables to select variables to read from the ensemble.
outputEnsemble.SelectedVariables = "Data";

% Loop through all ensemble members to read and write data.
while hasdata(outputEnsemble)
    % Read one member.
    member = read(outputEnsemble);

    % Get all input variables.
    Data = readMemberData(member,"Data");
    iv = (0:1:(height(Data)-1)*1)';
    Data.Sample = iv;

    % Initialize a table to store results.
    memberResult = table;

    %% PowerSpectrum
    try
        % Get units to use in computed spectrum.
        tuReal = "samples";
        tuTime = "seconds";

        % Compute effective sampling rate.
        tNumeric = time2num(Data.Sample,tuReal);
        [Fs,irregular] = effectivefs(tNumeric);
        Ts = 1/Fs;

        % Resample non-uniform signals.
        x = Data.X;
        if irregular
            x = resample(x,tNumeric,Fs,'linear');
        end
```

```

        % Compute the state-space model.
        data = iddata(x,[],Ts,'TimeUnit',tuTime,'OutputName','SpectrumData');
        ssOpt =
         ↪  ssestOptions('N4Horizon','auto','N4Weight','CVA','EstimateCovariance',false);
        ssOpt.Utility.Interactive = false;
        ssOpt.SearchOptions.MaxIterations = 20;
        model = ssest(data,4,ssOpt,'Ts',Ts);

        % Compute the power spectrum.
        [ps,w] = spectrum(model);
        ps = reshape(ps, numel(ps), 1);
        factor = 1/2/pi
        w = factor*w;

        % Remove frequencies above Nyquist frequency.
        I = w<=(Fs/2+1e4*eps);
        w = w(I);
        ps = ps(I);

        % Configure the computed spectrum.
        ps = table(w, ps, 'VariableNames', ["Frequency", "SpectrumData"]);
        ps.Properties.VariableUnits = ["cycles/sample", ""];
        ps = addprop(ps, {'SampleFrequency'}, {'table'});
        ps.Properties.CustomProperties.SampleFrequency = Fs;
        Data_ps = ps;
    catch
        Data_ps = table(NaN, NaN, 'VariableNames', ["Frequency", "SpectrumData"]);
    end

    % Append computed results to the member table.
    memberResult = [memberResult, ...
        table({Data_ps},'VariableNames',"Data_ps")]; %#ok<AGROW>

    %% SpectrumFeatures
    try
        % Compute spectral features.
        % Get frequency unit conversion factor.
        factor = 2*pi;
        ps = Data_ps.SpectrumData;
```

```
        w = Data_ps.Frequency;
        w = factor*w;
        mask_1 = (w>=factor*1.59154943091895e-11) & (w<=factor*0.5);
        ps = ps(mask_1);
        w = w(mask_1);

        % Compute spectral peaks.
        [peakAmp,peakFreq] = findpeaks(ps,w/factor,'MinPeakHeight',-Inf, ...
            ↪ 'MinPeakProminence',0,'MinPeakDistance',0.001,'SortStr','descend','NPeaks',1);
        peakAmp = [peakAmp(:); NaN(1-numel(peakAmp),1)];
        peakFreq = [peakFreq(:); NaN(1-numel(peakFreq),1)];

        % Extract individual feature values.
        PeakAmp1 = peakAmp(1);
        PeakFreq1 = peakFreq(1);
        BandPower = trapz(w/factor,ps);

        % Concatenate signal features.
        featureValues = [PeakAmp1,PeakFreq1,BandPower];

        % Package computed features into a table.
        featureNames = ["PeakAmp1","PeakFreq1","BandPower"];
        Data_ps_spec = array2table(featureValues,'VariableNames',featureNames);
    catch
        % Package computed features into a table.
        featureValues = NaN(1,3);
        featureNames = ["PeakAmp1","PeakFreq1","BandPower"];
        Data_ps_spec = array2table(featureValues,'VariableNames',featureNames);
    end

    % Append computed results to the member table.
    memberResult = [memberResult, ...
        table({Data_ps_spec},'VariableNames',"Data_ps_spec")]; %#ok<AGROW>

    %% Write all the results for the current member to the ensemble.
    writeToLastMemberRead(outputEnsemble,memberResult)
end

% Gather all features into a table.
```

```
featureTable = readFeatureTable(outputEnsemble);

% Feature ranking for FeatureTable1
selectedFeatureNames =
↪ ["Data_ps_spec/PeakAmp1","Data_ps_spec/PeakFreq1","Data_ps_spec/BandPower"];

% Get selected features and labels for classification ranking
selectedFeatureValues = featureTable{:,selectedFeatureNames};
label = featureTable{:,"FaultCode"};

% Convert label to numeric values
if iscategorical(label)
    label = string(label);
end
group = grp2idx(label);

% Initialize an empty matrix to store ranking scores
score = zeros(numel(selectedFeatureNames),0);

% Initialize a string array to store ranking method names
methodList = strings(0);

%% One-way ANOVA
% Normalize features using minmax.
fNorm =
↪ (selectedFeatureValues-min(selectedFeatureValues,[],1))./(max(selectedFeatureValues,[],1)-min(se

% Compute ranking score using One-Way ANOVA.
numFeatures = size(fNorm,2);
z = zeros(numFeatures,1);
for k = 1:numFeatures
    [~,tbl] = anova1(fNorm(:,k),group,'off');
    % Get the F-statistic from the output of one-way ANOVA.
    stats = tbl{2,5};
    if ~isempty(stats)
        z(k) = stats;
    end
end

% Append new score and method name.
```

```
score = [score,z];
methodList = [methodList,"One-way ANOVA"];


%% Create ranking result table
featureColumn = table(selectedFeatureNames(:),'VariableNames',{'Features'});
ranking = [featureColumn array2table(score,'VariableNames',methodList)];
ranking = sortrows(ranking,'One-way ANOVA','descend');

% Set SelectedVariables to select variables to read from the ensemble.
outputEnsemble.SelectedVariables =
↪ unique([outputEnsemble.DataVariables;outputEnsemble.ConditionVariables;outputEnsemble.Independer

% Gather results into a table.
outputTable = readall(outputEnsemble);
end
```

A.5 Code for Scenario 05: Graphical User Interface Implementation

```
\newpage


classdef app2 < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        FaultDiagnosticofInductionMotorbyAhsanUllahUIFigure  matlab.ui.Figure
        TabGroup3                      matlab.ui.container.TabGroup
        XTab                           matlab.ui.container.Tab
        TabGroup                       matlab.ui.container.TabGroup
        TimeDomainTab                  matlab.ui.container.Tab
        TabGroup2                      matlab.ui.container.TabGroup
        SVMTab                         matlab.ui.container.Tab
        SVMbasedModelLabel             matlab.ui.control.Label
        ResultTextArea                 matlab.ui.control.TextArea
        ResultTextAreaLabel            matlab.ui.control.Label
        Image                          matlab.ui.control.Image
        PredictionLabel                matlab.ui.control.Label
        Button_3                       matlab.ui.control.Button
        FeatureExtractionLabel         matlab.ui.control.Label
        Button_2                       matlab.ui.control.Button
        ImportfileLabel                matlab.ui.control.Label
        Button                         matlab.ui.control.Button
        KNNTab                         matlab.ui.container.Tab
        KNNbasedModelLabel             matlab.ui.control.Label
        ResultTextArea_21              matlab.ui.control.TextArea
        ResultTextArea_21Label         matlab.ui.control.Label
        Image_21                       matlab.ui.control.Image
        PredictionLabel_21             matlab.ui.control.Label
        Button_63                      matlab.ui.control.Button
        FeatureExtractionLabel_21      matlab.ui.control.Label
        Button_62                      matlab.ui.control.Button
        ImportfileLabel_21             matlab.ui.control.Label
        Button_61                      matlab.ui.control.Button
        EnsembleTab                    matlab.ui.container.Tab
        EnsemblebasedModelLabel        matlab.ui.control.Label
```

```
        ResultTextArea_7              matlab.ui.control.TextArea
        ResultTextArea_7Label         matlab.ui.control.Label
        Image_7                       matlab.ui.control.Image
        PredictionLabel_7             matlab.ui.control.Label
        Button_21                     matlab.ui.control.Button
        FeatureExtractionLabel_7      matlab.ui.control.Label
        Button_20                     matlab.ui.control.Button
        ImportfileLabel_7             matlab.ui.control.Label
        Button_19                     matlab.ui.control.Button
        NeuralNetworkTab              matlab.ui.container.Tab
        NeuralNetworkbasedModelLabel  matlab.ui.control.Label
        ResultTextArea_8              matlab.ui.control.TextArea
        ResultTextArea_8Label         matlab.ui.control.Label
        Image_8                       matlab.ui.control.Image
        PredictionLabel_8             matlab.ui.control.Label
        Button_24                     matlab.ui.control.Button
        FeatureExtractionLabel_8      matlab.ui.control.Label
        Button_23                     matlab.ui.control.Button
        ImportfileLabel_8             matlab.ui.control.Label
        Button_22                     matlab.ui.control.Button
        FrequencyDomainTab            matlab.ui.container.Tab
        YTab                          matlab.ui.container.Tab
        TabGroup_2                    matlab.ui.container.TabGroup
        TimeDomainTab_2               matlab.ui.container.Tab
        TabGroup2_2                   matlab.ui.container.TabGroup
        SVMTab_2                      matlab.ui.container.Tab
        ResultTextArea_22             matlab.ui.control.TextArea
        ResultTextArea_22Label        matlab.ui.control.Label
        Image_22                      matlab.ui.control.Image
        PredictionLabel_22            matlab.ui.control.Label
        Button_66                     matlab.ui.control.Button
        FeatureExtractionLabel_22     matlab.ui.control.Label
        Button_65                     matlab.ui.control.Button
        ImportfileLabel_22            matlab.ui.control.Label
        Button_64                     matlab.ui.control.Button
        KNNTab_2                      matlab.ui.container.Tab
        ResultTextArea_9              matlab.ui.control.TextArea
        ResultTextArea_9Label         matlab.ui.control.Label
        Image_9                       matlab.ui.control.Image
        PredictionLabel_9             matlab.ui.control.Label
```

```
        Button_27                    matlab.ui.control.Button
        FeatureExtractionLabel_9     matlab.ui.control.Label
        Button_26                    matlab.ui.control.Button
        ImportfileLabel_9            matlab.ui.control.Label
        Button_25                    matlab.ui.control.Button
        EnsembleTab_2                matlab.ui.container.Tab
        ResultTextArea_10            matlab.ui.control.TextArea
        ResultTextArea_10Label       matlab.ui.control.Label
        Image_10                     matlab.ui.control.Image
        PredictionLabel_10           matlab.ui.control.Label
        Button_30                    matlab.ui.control.Button
        FeatureExtractionLabel_10    matlab.ui.control.Label
        Button_29                    matlab.ui.control.Button
        ImportfileLabel_10           matlab.ui.control.Label
        Button_28                    matlab.ui.control.Button
        NeuralNetworkTab_2           matlab.ui.container.Tab
        ResultTextArea_11            matlab.ui.control.TextArea
        ResultTextArea_11Label       matlab.ui.control.Label
        Image_11                     matlab.ui.control.Image
        PredictionLabel_11           matlab.ui.control.Label
        Button_33                    matlab.ui.control.Button
        FeatureExtractionLabel_11    matlab.ui.control.Label
        Button_32                    matlab.ui.control.Button
        ImportfileLabel_11           matlab.ui.control.Label
        Button_31                    matlab.ui.control.Button
        FrequencyDomainTab_2         matlab.ui.container.Tab
        ZTab                         matlab.ui.container.Tab
        TabGroup_3                   matlab.ui.container.TabGroup
        TimeDomainTab_3              matlab.ui.container.Tab
        TabGroup2_3                  matlab.ui.container.TabGroup
        SVMTab_3                     matlab.ui.container.Tab
        ResultTextArea_23            matlab.ui.control.TextArea
        ResultTextArea_23Label       matlab.ui.control.Label
        Image_23                     matlab.ui.control.Image
        PredictionLabel_23           matlab.ui.control.Label
        Button_69                    matlab.ui.control.Button
        FeatureExtractionLabel_23    matlab.ui.control.Label
        Button_68                    matlab.ui.control.Button
        ImportfileLabel_23           matlab.ui.control.Label
        Button_67                    matlab.ui.control.Button
```

```
KNNTab_3                        matlab.ui.container.Tab
ResultTextArea_12               matlab.ui.control.TextArea
ResultTextArea_12Label          matlab.ui.control.Label
Image_12                        matlab.ui.control.Image
PredictionLabel_12              matlab.ui.control.Label
Button_36                       matlab.ui.control.Button
FeatureExtractionLabel_12       matlab.ui.control.Label
Button_35                       matlab.ui.control.Button
ImportfileLabel_12              matlab.ui.control.Label
Button_34                       matlab.ui.control.Button
EnsembleTab_3                   matlab.ui.container.Tab
ResultTextArea_13               matlab.ui.control.TextArea
ResultTextArea_13Label          matlab.ui.control.Label
Image_13                        matlab.ui.control.Image
PredictionLabel_13              matlab.ui.control.Label
Button_39                       matlab.ui.control.Button
FeatureExtractionLabel_13       matlab.ui.control.Label
Button_38                       matlab.ui.control.Button
ImportfileLabel_13              matlab.ui.control.Label
Button_37                       matlab.ui.control.Button
NeuralNetworkTab_3              matlab.ui.container.Tab
ResultTextArea_14               matlab.ui.control.TextArea
ResultTextArea_14Label          matlab.ui.control.Label
Image_14                        matlab.ui.control.Image
PredictionLabel_14              matlab.ui.control.Label
Button_42                       matlab.ui.control.Button
FeatureExtractionLabel_14       matlab.ui.control.Label
Button_41                       matlab.ui.control.Button
ImportfileLabel_14              matlab.ui.control.Label
Button_40                       matlab.ui.control.Button
FrequencyDomainTab_3            matlab.ui.container.Tab
AcousticTab                     matlab.ui.container.Tab
TabGroup_4                      matlab.ui.container.TabGroup
TimeDomainTab_4                 matlab.ui.container.Tab
TabGroup2_4                     matlab.ui.container.TabGroup
SVMTab_4                        matlab.ui.container.Tab
ResultTextArea_24               matlab.ui.control.TextArea
ResultTextArea_24Label          matlab.ui.control.Label
Image_24                        matlab.ui.control.Image
PredictionLabel_24              matlab.ui.control.Label
```

```
        Button_72                     matlab.ui.control.Button
        FeatureExtractionLabel_24     matlab.ui.control.Label
        Button_71                     matlab.ui.control.Button
        ImportfileLabel_24            matlab.ui.control.Label
        Button_70                     matlab.ui.control.Button
        KNNTab_4                      matlab.ui.container.Tab
        ResultTextArea_15             matlab.ui.control.TextArea
        ResultTextArea_15Label        matlab.ui.control.Label
        Image_15                      matlab.ui.control.Image
        PredictionLabel_15            matlab.ui.control.Label
        Button_45                     matlab.ui.control.Button
        FeatureExtractionLabel_15     matlab.ui.control.Label
        Button_44                     matlab.ui.control.Button
        ImportfileLabel_15            matlab.ui.control.Label
        Button_43                     matlab.ui.control.Button
        EnsembleTab_4                 matlab.ui.container.Tab
        ResultTextArea_16             matlab.ui.control.TextArea
        ResultTextArea_16Label        matlab.ui.control.Label
        Image_16                      matlab.ui.control.Image
        PredictionLabel_16            matlab.ui.control.Label
        Button_48                     matlab.ui.control.Button
        FeatureExtractionLabel_16     matlab.ui.control.Label
        Button_47                     matlab.ui.control.Button
        ImportfileLabel_16            matlab.ui.control.Label
        Button_46                     matlab.ui.control.Button
        NeuralNetworkTab_4            matlab.ui.container.Tab
        ResultTextArea_17             matlab.ui.control.TextArea
        ResultTextArea_17Label        matlab.ui.control.Label
        Image_17                      matlab.ui.control.Image
        PredictionLabel_17            matlab.ui.control.Label
        Button_51                     matlab.ui.control.Button
        FeatureExtractionLabel_17     matlab.ui.control.Label
        Button_50                     matlab.ui.control.Button
        ImportfileLabel_17            matlab.ui.control.Label
        Button_49                     matlab.ui.control.Button
        FrequencyDomainTab_4          matlab.ui.container.Tab
        CurrentTab                    matlab.ui.container.Tab
        TabGroup_5                    matlab.ui.container.TabGroup
        TimeDomainTab_5               matlab.ui.container.Tab
        TabGroup2_5                   matlab.ui.container.TabGroup
        SVMTab_5                      matlab.ui.container.Tab
        ResultTextArea_25             matlab.ui.control.TextArea
        ResultTextArea_25Label        matlab.ui.control.Label
        Image_25                      matlab.ui.control.Image
        PredictionLabel_25            matlab.ui.control.Label
        Button_75                     matlab.ui.control.Button
```

Bibliography

[1] Ritonja, Jožef (2021-04-21). ”Robust and Adaptive Control for Synchronous Generator’s Operation Improvement”. Automation and Control. IntechOpen. doi:10.5772/intechopen.92558

[2] X. Liang, and K. Edomwandekhoe, “Condition Monitoring Techniques for Induction Motors”, Proceedings of 2017 IEEE Industry Applications Society (IAS) Annual Meeting, pp. 1-10, 2017.

[3] https://www.electricaltechnology.org/2020/05/single-phase-induction-motor.html

[4] P. C. Sen, “Principles of electric machines and power electronics,” John Wiley and Sons, 1989.

[5] PK. S. V. Khadim Moin Siddiqui, ”Health Monitoring and Fault Diagnosis in Induction Motor- A Review,” vol. 3, no. 1, January 2014, 1, January 2014

[6] (https://engineering.case.edu/bearingdatacenter/download-data-file)

[7] https://researchdata.ntu.edu.sg/dataset.xhtml?persistentId=doi:10.21979/N9/X6M827

[8] https://data.mendeley.com/datasets/v43hmbwxpm/1

[9] Stief, Anna, James R. Ottewill, Michal Orkisz, and Jerzy Baranowski. ”Two stage data fusion of acoustic, electric and vibration signals for diagnosing faults in induction motors.” Elektronika ir Elektrotechnika 23, no. 6 (2017): 19-24.

[10] https://www02.smt.ufrj.br/ offshore/mfs/page 01.html

[11] Peter Vas, “Parameter estimation, condition monitoring, and diagnosis of electrical machines”, Clarendon Press Oxford., 1993.

[12] P. J. Tavner and J. Penman, “Condition monitoring of electrical machines”. Hertfordshire, England: Research Studies Press Ltd, ISBN: 0863800610, 1987.

[13] Li, B., Chow, M. Y., Tipsuwan and Y., Hung, J. C., “Neural-Network-Based Motor Rolling Bearing Fault Diagnosis”, IEEE Transactions on Industrial Electronics, Vol. 47, No. 5, October, pp. 1060-1069, 2000

[14] S.Nandi and H.A. Toliyat, “Condition Monitoring And Fault Diagnosis of Electrical Machines- A Review”, in proc, 34th Annual Meeting of the IEEE Industry Applications,pp.197-204,1999

[15] P. H. Mellor, D. Roberts, and D. R. Turner, “Lumped parameter thermal model for electrical machines of TEFC design,” IEEE Pro. Electric Power Application, Vol. 138, pp. 205-218, 1991.

[16] O. I. Okoro, “Steady and transient states thermal analysis of a 7.5-kW squirrel-cage induction machine at rated-load operation,” IEEE Transactions on Energy Conversion, Vol. 20, No. 4, pp. 730-736, December, 2005.

[17] Othman, Mohd Sufian, Mohd Zaki Nuawi, and Ramizi Mohamed. ”Vibration and acoustic emission signal monitoring for detection of induction motor bearing fault.” Int. J. Eng. Res. Technol 4 (2015): 924-929.

[18] Randy R. Schoen, Thomas G. Habetler, Farrukh Kamran and Robert G. Bartheld, “Motor bearing damage detection using stator current monitoring”, IEEE Transactions on Industry Applications, Vol. 31, No 6, pp. 1274-1279, 1995.

[19] M. E. H. Benbouzid , H. Nejjari, R. Beguenane, and M. Vieira, “Induction motor asymmetrical faults detection using advanced signal processing techniques,” IEEE Transactions on Energy Conversion, Vol. 14, No. 2, pp.147-152, June 1999.

[20] Milrtic A. and Cettolo M., “ Frequency converter influence on induction motor rotor faults detection using motor current signature analysis-Experimental research, Symposium on Diagnostic for electric machines, Power Electronics and Derives, Atlanta, GA, USA, 24-26 march, pp. 124-128, Aug.2003.

[21] Jason R. Stack, Thomas G. Habetler, Ronald G. Harley, “Fault classification and

fault signature production for rolling element bearings", IEEE Transactions on Industry Applications, Vol. 40, No. 3, pp.735-739, 2004.

[22] https://www.iso.org/obp/ui/en/iso:std:iso:22096:ed-1:v1:en

[23] Eschmann P, Hasbargen L and Weigand K, "Ball and Roller Bearings: Their Theory, Design and Application", London: K G Heyden, 1958

[24] Khadim Moin Siddiqui and V.K.Giri. "Broken Rotor Bar Fault Detection in Induction Motors using Wavelet Transform", Int. Conf Proc, IEEE, Computing, Electronics and Electrical Technologies [ICCEET], pp. 1-6, Chennai, India, March, 2012

[25] F. Filippetti, G. Franceschini, and C. Tassoni, "Neural networks aided on-line diagnostics of induction motor rotor faults", IEEE Trans.Industry Applications, Vol.31, No.4, pp. 892 – 899, 1995.

[26] S. Altug, M. Chen, and H. J. Trussell, "Fuzzy inference systems implemented on neural architectures for motor fault detection and diagnosis", IEEE Trans. Industrial Electronics, Vol. 46, No. 6, pp. 1069 - 1079, 1999

[27] P. J. C. Branco, J. A. Dente, and R. V. Mendes, "Using immunology principles for fault detection", IEEE Trans. Industrial Electronics, Vol.50, No. 2, pp. 362 - 373, 2003.

[28] O. Ondel, E. Boutleux, E. Blanco, and G. Clerc, "Coupling pattern recognition with state estimation using Kalman filter for fault diagnosis", IEEE Trans. Industrial Electronics, Vol. 59, No. 11, pp. 4293 - 4300,2012.

[29] R. Razavi-Far, M. Farajzadeh-Zanjani, and M. Saif, "An integrated class-imbalanced learning scheme for diagnosing bearing defects in induction motors", IEEE Trans. Industrial Informatics, Vol. 13, No. 6, pp. 2758 - 2769, 2017.

[30] C. Sun, M. Ma, Z. Zhao, and X. Chen, "Sparse deep stacking network for fault diagnosis of motor", IEEE Trans. Industrial Informatics, Vol. 14, No. 7, pp. 3261 - 3270, 2018.

[31] J. Seshadrinath, B. Singh, and B. K. Panigrahi, “Investigation of vibration signatures for multiple fault diagnosis in variable frequency drives using complex wavelets”, IEEE Trans. Power Electronics, Vol. 29, No. 2, pp. 936 – 945, 2014.

[32] S. S., K. U. Rao, R. Umesh, and H. K. S., “Condition monitoring of Induction Motor using statistical processing,” 2016 IEEE Region 10 Conference (TENCON), 2016.

[33] J. Seshadrinath, B. Singh, and B. K. Panigrahi, “Investigation of vibration signatures for multiple fault diagnosis in variable frequency drives using complex wavelets”, IEEE Trans. Power Electronics, Vol. 29, No. 2, pp. 936 – 945, 2014.

[34] K.-R. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf, “An introduction to kernel-based learning algorithms,” IEEE Trans. Neural Networks, vol. 12, no. 2, pp. 181–201, 2001.

[35] H. Jung, S.-W. Kang, M. Song, S. Im, J. Kim, and C.-S. Hwang,“Towards Real-Time Processing of Monitoring Continuous k-Nearest Neighbor Queries,” Frontiers of High Performance Computing and Networking – ISPA 2006 Workshops Lecture Notes in Computer Science,pp. 11–20, 2006.

[36] L. Breiman, “Bagging predictors,” Machine Learning, vol. 24, no. 2, pp. 123–140, 1996.

[37] Xiaodong Liang, Yi He, Massimo Mitolo, and Weixing Li, ”Support Vector Machine Based Dynamic Load Model Using Synchrophasor Data”, Proceedings of IEEE 54th Industrial and Commercial Power Systems Conference, pp. 1-11, May 2018

[38] D. Gupta, “Activation functions — fundamentals of deep learning,” 12 2022.

[39] P. Marimuthu, “How activation functions work in deep learning - kdnuggets,” 6 2022.

[40] M. He and D. He, ”Deep Learning Based Approach for Bearing Fault Diagnosis,” in IEEE Transactions on Industry Applications, vol. 53, no. 3, pp. 3057-3065, May 2017

[41] M. Benninger, M. Liebschner, and C. Kreischer, ”Fault Detection of Induction Motors with Combined Modeling- and Machine-Learning-Based Framework,” in Energies, vol. 16, no. 8, pp. 3429, Apr. 2023

[42] V. Barai, S. M. Ramteke, V. Dhanalkotwar, Y. Nagmote, S. Shende, and D. Deshmukh, ”Bearing fault diagnosis using signal processing and machine learning techniques: A review,” in IOP Conference Series: Materials Science and Engineering, vol. 1259, no. 1, p. 012034, Oct.2022 arXiv:1612.07600, 2016.

B Datasheets

ACS712 Current Sensor

Basic Overview

The ACS712 Current Sensors offered on the internet are designed to be easily used with micro controllers like the Arduino.

These sensors are based on the Allegro ACS712ELC chip.

These current sensors are offered with full scale values of 5A, 20A and 30A.

The basic functional operation of each of these devices is identical. The only difference is with the scale factor at the output as detailed below.

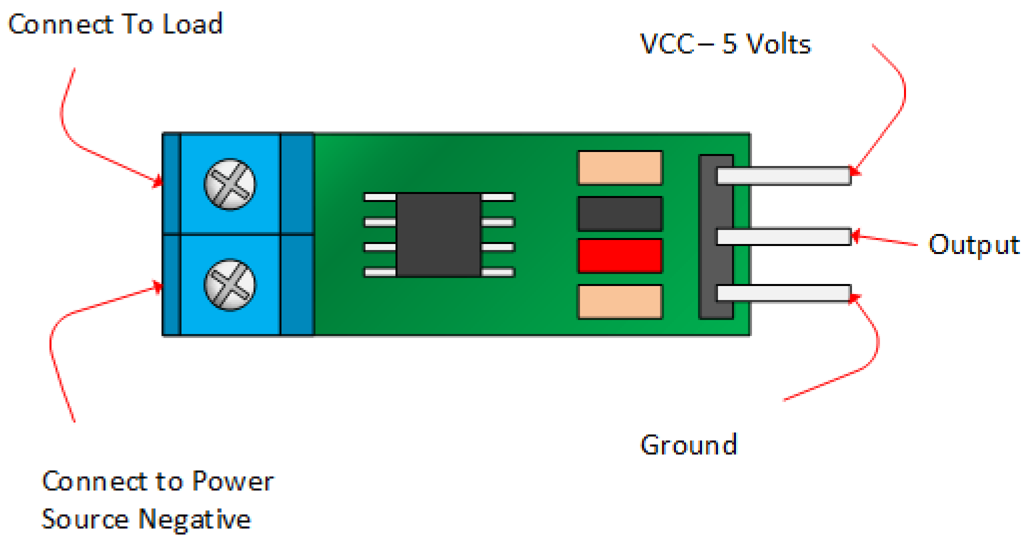
Sensor Specifications

	5A Module	**20A Module**	**30A Module**
Supply Voltage (VCC)	5Vdc Nominal	5Vdc Nominal	5Vdc Nominal
Measurement Range	-5 to +5 Amps	-20 to +20 Amps	-30 to +30 Amps
Voltage at 0A	VCC/2 (nominally 2.5Vdc)	VCC/2 (nominally 2.5Vdc)	VCC/2 (nominally 2.5VDC)
Scale Factor	185 mV per Amp	100 mV per Amp	66 mV per Amp
Chip	ACS712ELC-05A	ACS712ELC-10A	ACS712ELC-30A

ACS712 Module Pin Outs and Connections

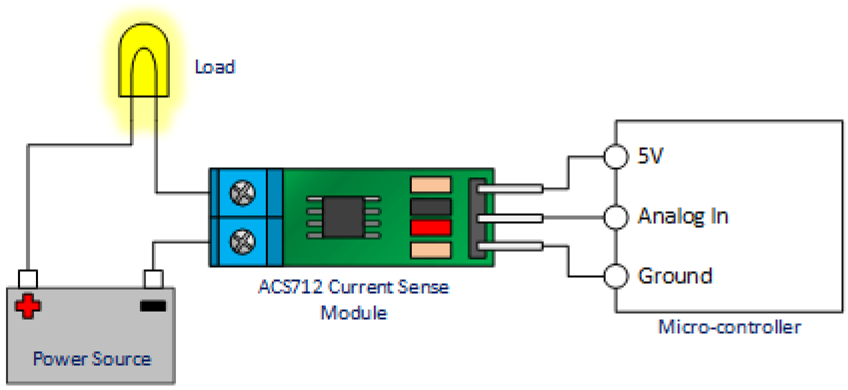
The picture below identifies the pin outs for the ACS172 Modules.

Pay attention to the polarity at the load end of the device.   If you are connected as illustrated below, the output will raise. If you connect it opposite of this picture, the output will decrease from the 2.5 volt offset.



Basic Hook Up and Functional Description

As mentioned before, these modules are primarily designed for use with micro-controllers like the Arduino. In those applications, the connections would be as picture below:



If the light bulb shown in the picture above were disconnected, the output of the ACS712 module would be 2.500 volts.

Once connected, the output would be scaled to the current drawn through the bulb. If this were a 5 Amp module and the light bulb pulled 1 Amp, the output of the module would be 2.685 volts.

Now imagine the battery polarity reversed. Using the same 5A module, the output would be 2.315 volts.

IMPORTANT NOTE – This device is a Hall Effect transducer. It should not be used near significant magnetic fields.

ANALOG DEVICES

Small, Low Power, 3-Axis ±3 *g* Accelerometer

ADXL335

FEATURES

3-axis sensing
Small, low profile package
4 mm × 4 mm × 1.45 mm LFCSP
Low power : 350 μA (typical)
Single-supply operation: 1.8 V to 3.6 V
10,000 *g* shock survival
Excellent temperature stability
BW adjustment with a single capacitor per axis
RoHS/WEEE lead-free compliant

APPLICATIONS

Cost sensitive, low power, motion- and tilt-sensing applications
Mobile devices
Gaming systems
Disk drive protection
Image stabilization
Sports and health devices

GENERAL DESCRIPTION

The ADXL335 is a small, thin, low power, complete 3-axis accelerometer with signal conditioned voltage outputs. The product measures acceleration with a minimum full-scale range of ±3 *g*. It can measure the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion, shock, or vibration.

The user selects the bandwidth of the accelerometer using the C_X, C_Y, and C_Z capacitors at the X_{OUT}, Y_{OUT}, and Z_{OUT} pins. Bandwidths can be selected to suit the application, with a range of 0.5 Hz to 1600 Hz for the X and Y axes, and a range of 0.5 Hz to 550 Hz for the Z axis.

The ADXL335 is available in a small, low profile, 4 mm × 4 mm × 1.45 mm, 16-lead, plastic lead frame chip scale package (LFCSP_LQ).

FUNCTIONAL BLOCK DIAGRAM

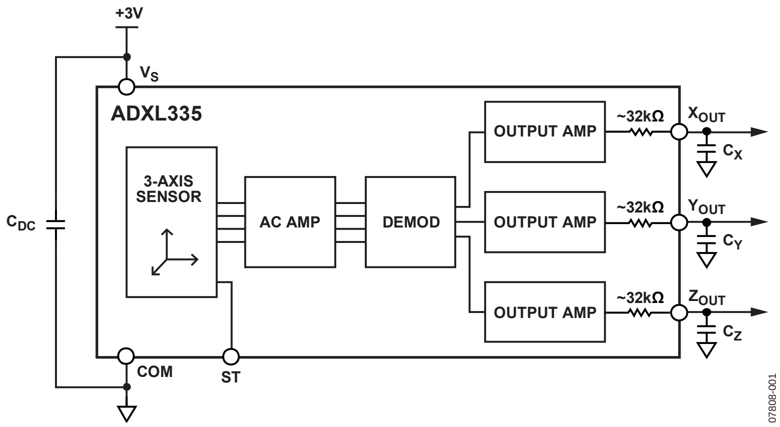


Figure 1.

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700 **www.analog.com**
Fax: 781.461.3113

TABLE OF CONTENTS

REVISION HISTORY

1/10—Rev. A to Rev. B

Changes to Figure 21 ... 9

7/09—Rev. 0 to Rev. A

Changes to Figure 22 ... 9
Changes to Outline Dimensions ... 14

1/09—Revision 0: Initial Version

SPECIFICATIONS

T_A = 25°C, V_S = 3 V, C_X = C_Y = C_Z = 0.1 µF, acceleration = 0 *g*, unless otherwise noted. All minimum and maximum specifications are guaranteed. Typical specifications are not guaranteed.

Table 1.

Parameter	Conditions	Min	Typ	Max	Unit
SENSOR INPUT	Each axis				
Measurement Range		±3	±3.6		*g*
Nonlinearity	% of full scale		±0.3		%
Package Alignment Error			±1		Degrees
Interaxis Alignment Error			±0.1		Degrees
Cross-Axis Sensitivity[1]			±1		%
SENSITIVITY (RATIOMETRIC)[2]	Each axis				
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	V_S = 3 V	270	300	330	mV/*g*
Sensitivity Change Due to Temperature[3]	V_S = 3 V		±0.01		%/°C
ZERO *g* BIAS LEVEL (RATIOMETRIC)					
0 *g* Voltage at X_{OUT}, Y_{OUT}	V_S = 3 V	1.35	1.5	1.65	V
0 *g* Voltage at Z_{OUT}	V_S = 3 V	1.2	1.5	1.8	V
0 *g* Offset vs. Temperature			±1		m*g*/°C
NOISE PERFORMANCE					
Noise Density X_{OUT}, Y_{OUT}			150		µ*g*/√Hz rms
Noise Density Z_{OUT}			300		µ*g*/√Hz rms
FREQUENCY RESPONSE[4]					
Bandwidth X_{OUT}, Y_{OUT}[5]	No external filter		1600		Hz
Bandwidth Z_{OUT}[5]	No external filter		550		Hz
R_{FILT} Tolerance			32 ± 15%		kΩ
Sensor Resonant Frequency			5.5		kHz
SELF-TEST[6]					
Logic Input Low			+0.6		V
Logic Input High			+2.4		V
ST Actuation Current			+60		µA
Output Change at X_{OUT}	Self-Test 0 to Self-Test 1	−150	−325	−600	mV
Output Change at Y_{OUT}	Self-Test 0 to Self-Test 1	+150	+325	+600	mV
Output Change at Z_{OUT}	Self-Test 0 to Self-Test 1	+150	+550	+1000	mV
OUTPUT AMPLIFIER					
Output Swing Low	No load		0.1		V
Output Swing High	No load		2.8		V
POWER SUPPLY					
Operating Voltage Range		1.8		3.6	V
Supply Current	V_S = 3 V		350		µA
Turn-On Time[7]	No external filter		1		ms
TEMPERATURE					
Operating Temperature Range		−40		+85	°C

[1] Defined as coupling between any two axes.
[2] Sensitivity is essentially ratiometric to V_S.
[3] Defined as the output change from ambient-to-maximum temperature or ambient-to-minimum temperature.
[4] Actual frequency response controlled by user-supplied external filter capacitors (C_X, C_Y, C_Z).
[5] Bandwidth with external capacitors = 1/(2 × π × 32 kΩ × C). For C_X, C_Y = 0.003 µF, bandwidth = 1.6 kHz. For C_Z = 0.01 µF, bandwidth = 500 Hz. For C_X, C_Y, C_Z = 10 µF, bandwidth = 0.5 Hz.
[6] Self-test response changes cubically with V_S.
[7] Turn-on time is dependent on C_X, C_Y, C_Z and is approximately 160 × C_X or C_Y or C_Z + 1 ms, where C_X, C_Y, C_Z are in microfarads (µF).

ABSOLUTE MAXIMUM RATINGS

Table 2.

Parameter	Rating
Acceleration (Any Axis, Unpowered)	10,000 *g*
Acceleration (Any Axis, Powered)	10,000 *g*
V_S	−0.3 V to +3.6 V
All Other Pins	(COM − 0.3 V) to (V_S + 0.3 V)
Output Short-Circuit Duration (Any Pin to Common)	Indefinite
Temperature Range (Powered)	−55°C to +125°C
Temperature Range (Storage)	−65°C to +150°C

Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; functional operation of the device at these or any other conditions above those indicated in the operational section of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ESD CAUTION

ESD (electrostatic discharge) sensitive device. Charged devices and circuit boards can discharge without detection. Although this product features patented or proprietary protection circuitry, damage may occur on devices subjected to high energy ESD. Therefore, proper ESD precautions should be taken to avoid performance degradation or loss of functionality.

PIN CONFIGURATION AND FUNCTION DESCRIPTIONS

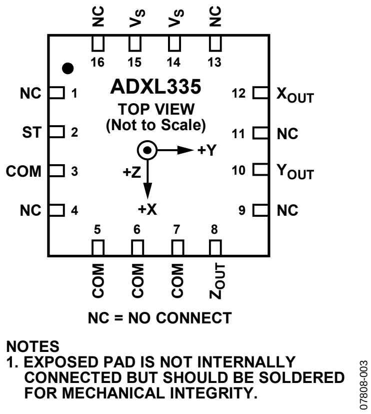


Figure 2. Pin Configuration

Table 3. Pin Function Descriptions

Pin No.	Mnemonic	Description
1	NC	No Connect.[1]
2	ST	Self-Test.
3	COM	Common.
4	NC	No Connect.[1]
5	COM	Common.
6	COM	Common.
7	COM	Common.
8	Z_{OUT}	Z Channel Output.
9	NC	No Connect.[1]
10	Y_{OUT}	Y Channel Output.
11	NC	No Connect. [1]
12	X_{OUT}	X Channel Output.
13	NC	No Connect. [1]
14	V_S	Supply Voltage (1.8 V to 3.6 V).
15	V_S	Supply Voltage (1.8 V to 3.6 V).
16	NC	No Connect. [1]
EP	Exposed Pad	Not internally connected. Solder for mechanical integrity.

[1] NC pins are not internally connected and can be tied to COM pins, unless otherwise noted.

TYPICAL PERFORMANCE CHARACTERISTICS

N > 1000 for all typical performance plots, unless otherwise noted.

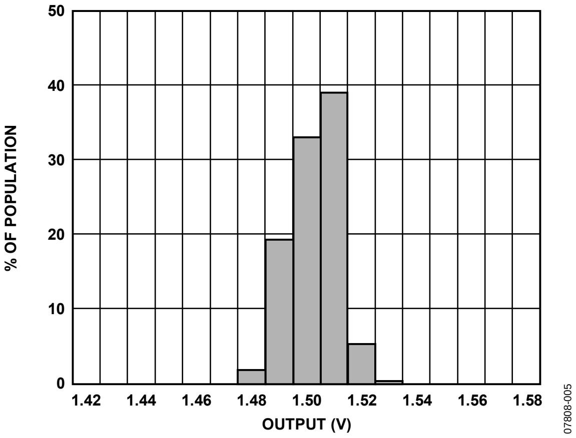


Figure 3. X-Axis Zero g Bias at 25°C, V_S = 3 V

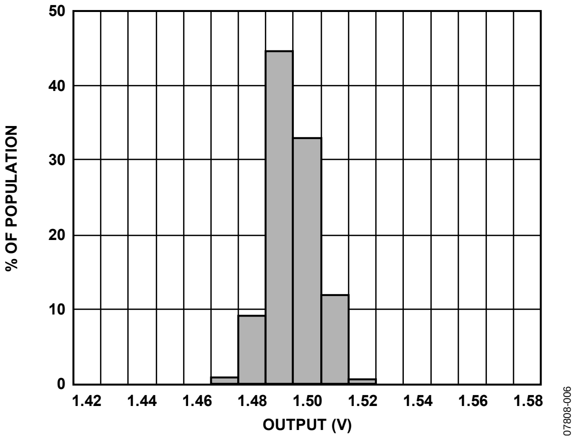


Figure 4. Y-Axis Zero g Bias at 25°C, V_S = 3 V

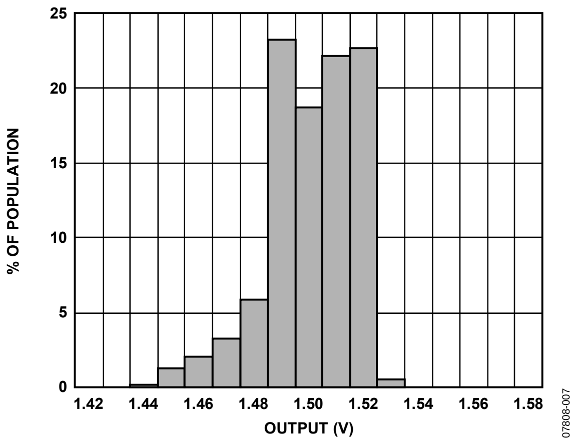


Figure 5. Z-Axis Zero g Bias at 25°C, V_S = 3 V

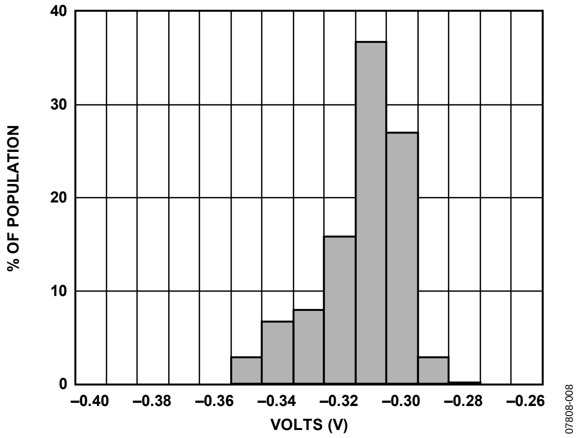


Figure 6. X-Axis Self-Test Response at 25°C, V_S = 3 V

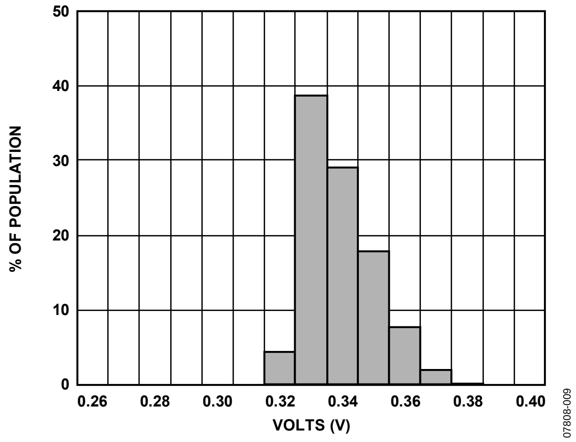


Figure 7. Y-Axis Self-Test Response at 25°C, V_S = 3 V

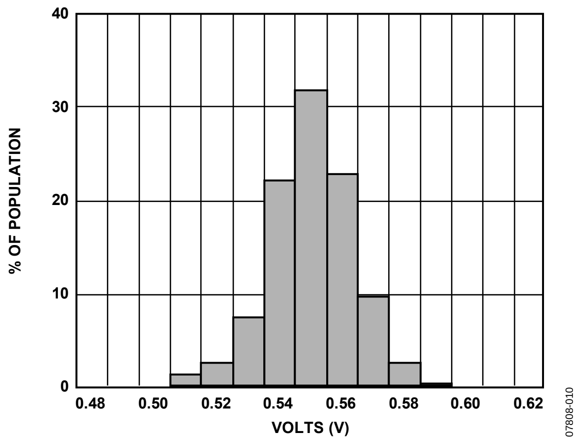


Figure 8. Z-Axis Self-Test Response at 25°C, V_S = 3 V

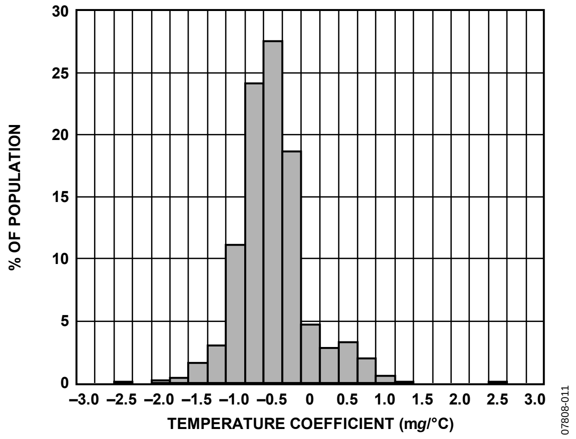


Figure 9. X-Axis Zero g Bias Temperature Coefficient, V_S = 3 V

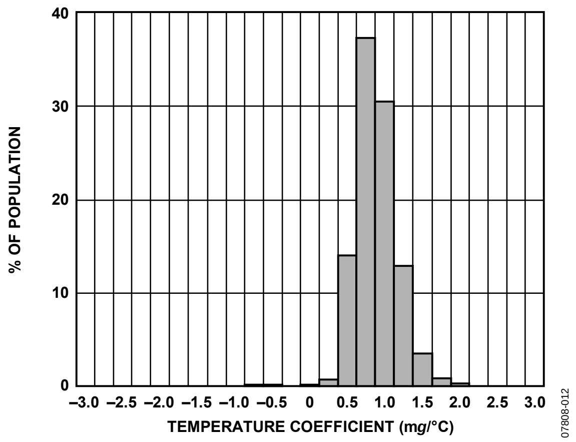


Figure 10. Y-Axis Zero g Bias Temperature Coefficient, V_S = 3 V

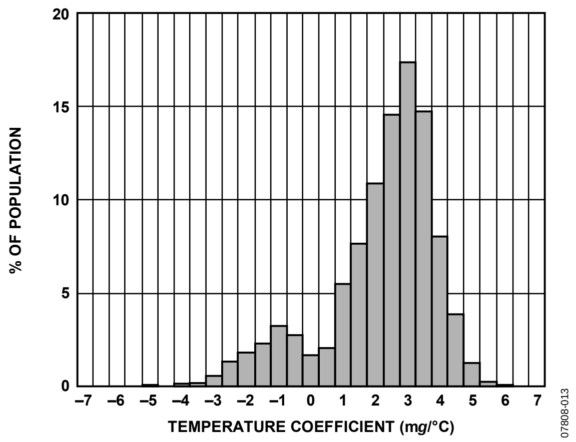


Figure 11. Z-Axis Zero g Bias Temperature Coefficient, V_S = 3 V

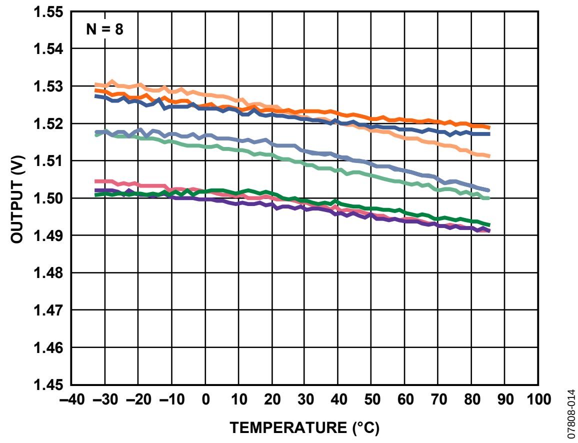


Figure 12. X-Axis Zero g Bias vs. Temperature—
Eight Parts Soldered to PCB

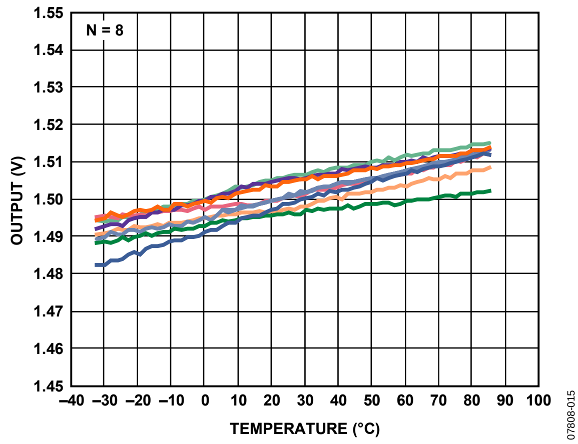


Figure 13. Y-Axis Zero g Bias vs. Temperature—
Eight Parts Soldered to PCB

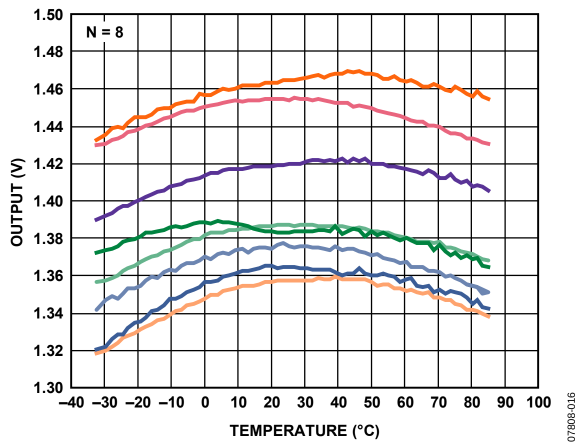


Figure 14. Z-Axis Zero g Bias vs. Temperature—
Eight Parts Soldered to PCB

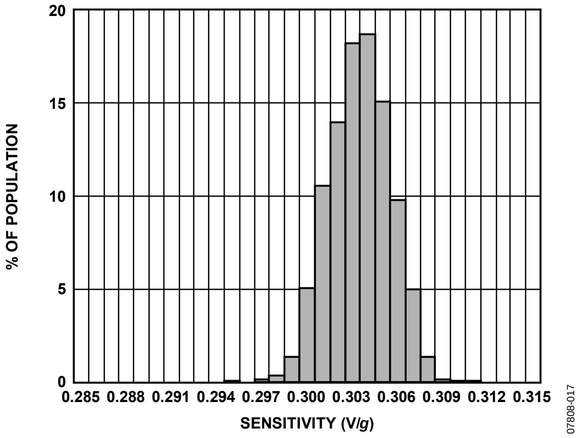


Figure 15. X-Axis Sensitivity at 25°C, V_S = 3 V

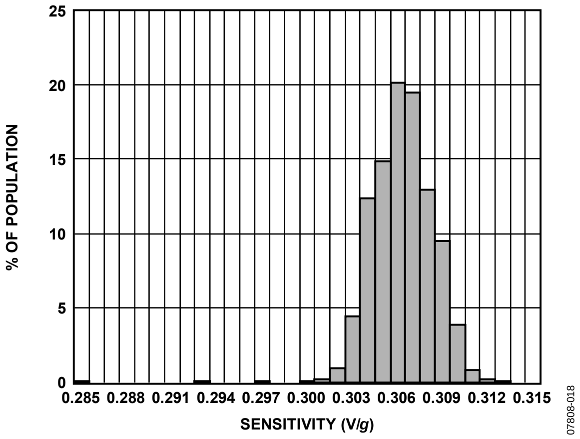


Figure 16. Y-Axis Sensitivity at 25°C, V_S = 3 V

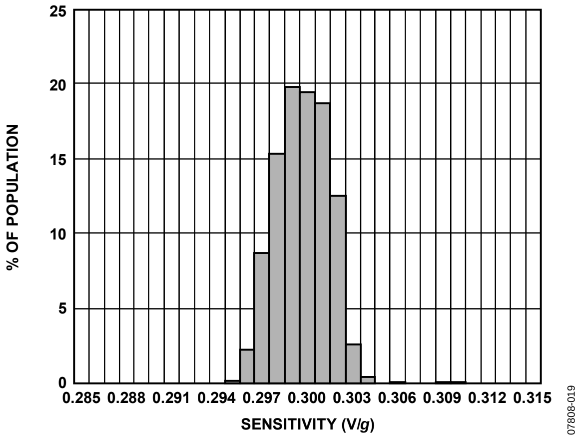


Figure 17. Z-Axis Sensitivity at 25°C, V_S = 3 V

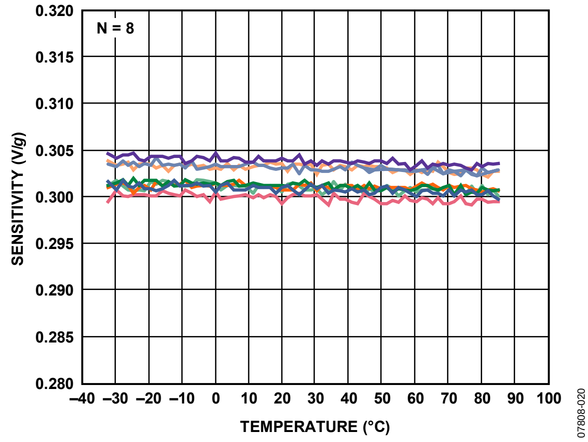


Figure 18. X-Axis Sensitivity vs. Temperature—Eight Parts Soldered to PCB, V_S = 3 V

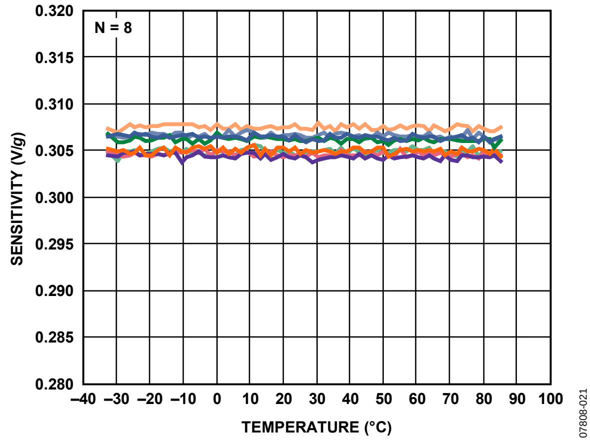


Figure 19. Y-Axis Sensitivity vs. Temperature—Eight Parts Soldered to PCB, V_S = 3 V

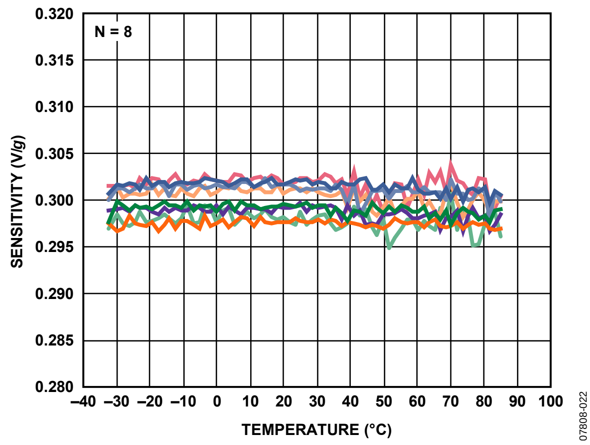


Figure 20. Z-Axis Sensitivity vs. Temperature—Eight Parts Soldered to PCB, V_S = 3 V

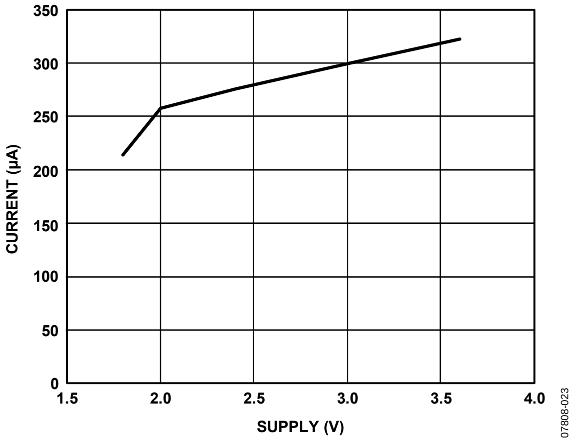


Figure 21. Typical Current Consumption vs. Supply Voltage

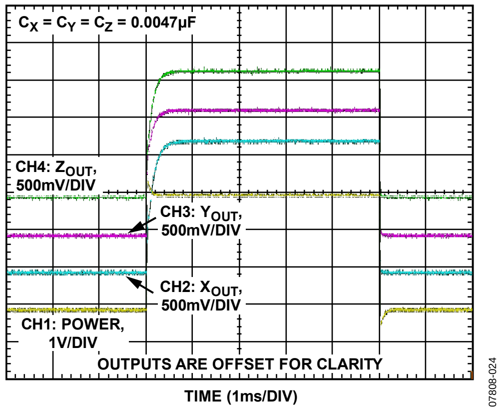


Figure 22. Typical Turn-On Time, $V_S = 3$ V

THEORY OF OPERATION

The ADXL335 is a complete 3-axis acceleration measurement system. The ADXL335 has a measurement range of ±3 *g* minimum. It contains a polysilicon surface-micromachined sensor and signal conditioning circuitry to implement an open-loop acceleration measurement architecture. The output signals are analog voltages that are proportional to acceleration. The accelerometer can measure the static acceleration of gravity in tilt-sensing applications as well as dynamic acceleration resulting from motion, shock, or vibration.

The sensor is a polysilicon surface-micromachined structure built on top of a silicon wafer. Polysilicon springs suspend the structure over the surface of the wafer and provide a resistance against acceleration forces. Deflection of the structure is measured using a differential capacitor that consists of independent fixed plates and plates attached to the moving mass. The fixed plates are driven by 180° out-of-phase square waves. Acceleration deflects the moving mass and unbalances the differential capacitor resulting in a sensor output whose amplitude is proportional to acceleration. Phase-sensitive demodulation techniques are then used to determine the magnitude and direction of the acceleration.

The demodulator output is amplified and brought off-chip through a 32 kΩ resistor. The user then sets the signal bandwidth of the device by adding a capacitor. This filtering improves measurement resolution and helps prevent aliasing.

MECHANICAL SENSOR

The ADXL335 uses a single structure for sensing the X, Y, and Z axes. As a result, the three axes' sense directions are highly orthogonal and have little cross-axis sensitivity. Mechanical misalignment of the sensor die to the package is the chief source of cross-axis sensitivity. Mechanical misalignment can, of course, be calibrated out at the system level.

PERFORMANCE

Rather than using additional temperature compensation circuitry, innovative design techniques ensure that high performance is built in to the ADXL335. As a result, there is no quantization error or nonmonotonic behavior, and temperature hysteresis is very low (typically less than 3 m*g* over the −25°C to +70°C temperature range).

APPLICATIONS INFORMATION

POWER SUPPLY DECOUPLING

For most applications, a single 0.1 μF capacitor, C_{DC}, placed close to the ADXL335 supply pins adequately decouples the accelerometer from noise on the power supply. However, in applications where noise is present at the 50 kHz internal clock frequency (or any harmonic thereof), additional care in power supply bypassing is required because this noise can cause errors in acceleration measurement.

If additional decoupling is needed, a 100 Ω (or smaller) resistor or ferrite bead can be inserted in the supply line. Additionally, a larger bulk bypass capacitor (1 μF or greater) can be added in parallel to C_{DC}. Ensure that the connection from the ADXL335 ground to the power supply ground is low impedance because noise transmitted through ground has a similar effect to noise transmitted through V_S.

SETTING THE BANDWIDTH USING C_X, C_Y, AND C_Z

The ADXL335 has provisions for band limiting the X_{OUT}, Y_{OUT}, and Z_{OUT} pins. Capacitors must be added at these pins to implement low-pass filtering for antialiasing and noise reduction. The equation for the 3 dB bandwidth is

$$F_{-3\,dB} = 1/(2\pi(32\ k\Omega) \times C_{(X,\,Y,\,Z)})$$

or more simply

$$F_{-3\,dB} = 5\ \mu F/C_{(X,\,Y,\,Z)}$$

The tolerance of the internal resistor (R_{FILT}) typically varies as much as ±15% of its nominal value (32 kΩ), and the bandwidth varies accordingly. A minimum capacitance of 0.0047 μF for C_X, C_Y, and C_Z is recommended in all cases.

Table 4. Filter Capacitor Selection, C_X, C_Y, and C_Z

Bandwidth (Hz)	Capacitor (μF)
1	4.7
10	0.47
50	0.10
100	0.05
200	0.027
500	0.01

SELF-TEST

The ST pin controls the self-test feature. When this pin is set to V_S, an electrostatic force is exerted on the accelerometer beam. The resulting movement of the beam allows the user to test if the accelerometer is functional. The typical change in output is −1.08 *g* (corresponding to −325 mV) in the X-axis, +1.08 *g* (or +325 mV) on the Y-axis, and +1.83 *g* (or +550 mV) on the Z-axis. This ST pin can be left open-circuit or connected to common (COM) in normal use.

Never expose the ST pin to voltages greater than V_S + 0.3 V. If this cannot be guaranteed due to the system design (for instance, if there are multiple supply voltages), then a low V_F clamping diode between ST and V_S is recommended.

DESIGN TRADE-OFFS FOR SELECTING FILTER CHARACTERISTICS: THE NOISE/BW TRADE-OFF

The selected accelerometer bandwidth ultimately determines the measurement resolution (smallest detectable acceleration). Filtering can be used to lower the noise floor to improve the resolution of the accelerometer. Resolution is dependent on the analog filter bandwidth at X_{OUT}, Y_{OUT}, and Z_{OUT}.

The output of the ADXL335 has a typical bandwidth of greater than 500 Hz. The user must filter the signal at this point to limit aliasing errors. The analog bandwidth must be no more than half the analog-to-digital sampling frequency to minimize aliasing. The analog bandwidth can be further decreased to reduce noise and improve resolution.

The ADXL335 noise has the characteristics of white Gaussian noise, which contributes equally at all frequencies and is described in terms of μg/√Hz (the noise is proportional to the square root of the accelerometer bandwidth). The user should limit bandwidth to the lowest frequency needed by the application to maximize the resolution and dynamic range of the accelerometer.

With the single-pole, roll-off characteristic, the typical noise of the ADXL335 is determined by

$$rms\ Noise = Noise\ Density \times (\sqrt{BW \times 1.6})$$

It is often useful to know the peak value of the noise. Peak-to-peak noise can only be estimated by statistical methods. Table 5 is useful for estimating the probabilities of exceeding various peak values, given the rms value.

Table 5. Estimation of Peak-to-Peak Noise

Peak-to-Peak Value	% of Time That Noise Exceeds Nominal Peak-to-Peak Value
2 × rms	32
4 × rms	4.6
6 × rms	0.27
8 × rms	0.006

USE WITH OPERATING VOLTAGES OTHER THAN 3 V

The ADXL335 is tested and specified at $V_S = 3$ V; however, it can be powered with V_S as low as 1.8 V or as high as 3.6 V. Note that some performance parameters change as the supply voltage is varied.

The ADXL335 output is ratiometric, therefore, the output sensitivity (or scale factor) varies proportionally to the supply voltage. At $V_S = 3.6$ V, the output sensitivity is typically 360 mV/*g*. At $V_S = 2$ V, the output sensitivity is typically 195 mV/*g*.

The zero *g* bias output is also ratiometric, thus the zero *g* output is nominally equal to $V_S/2$ at all supply voltages.

The output noise is not ratiometric but is absolute in volts; therefore, the noise density decreases as the supply voltage increases. This is because the scale factor (mV/*g*) increases while the noise voltage remains constant. At $V_S = 3.6$ V, the X-axis and Y-axis noise density is typically 120 μ*g*/√Hz, whereas at $V_S = 2$ V, the X-axis and Y-axis noise density is typically 270 μ*g*/√Hz.

Self-test response in *g* is roughly proportional to the square of the supply voltage. However, when ratiometricity of sensitivity is factored in with supply voltage, the self-test response in volts is roughly proportional to the cube of the supply voltage. For example, at $V_S = 3.6$ V, the self-test response for the ADXL335 is approximately −560 mV for the X-axis, +560 mV for the Y-axis, and +950 mV for the Z-axis.

At $V_S = 2$ V, the self-test response is approximately −96 mV for the X-axis, +96 mV for the Y-axis, and −163 mV for the Z-axis.

The supply current decreases as the supply voltage decreases. Typical current consumption at $V_S = 3.6$ V is 375 μA, and typical current consumption at $V_S = 2$ V is 200 μA.

AXES OF ACCELERATION SENSITIVITY

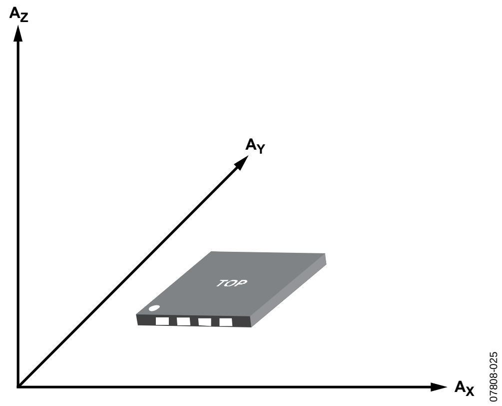


Figure 23. Axes of Acceleration Sensitivity; Corresponding Output Voltage Increases When Accelerated Along the Sensitive Axis.

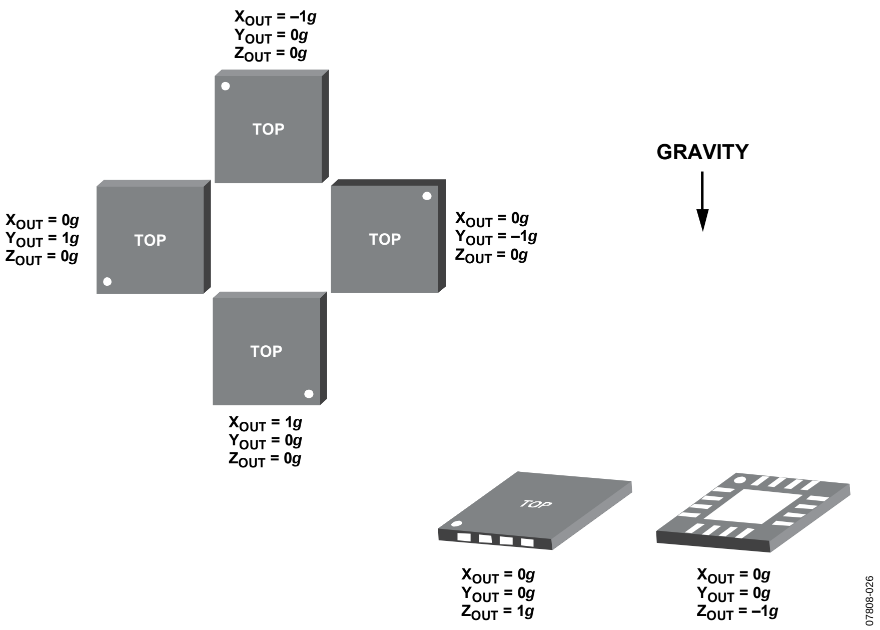


Figure 24. Output Response vs. Orientation to Gravity

LAYOUT AND DESIGN RECOMMENDATIONS

The recommended soldering profile is shown in Figure 25 followed by a description of the profile features in Table 6. The recommended PCB layout or solder land drawing is shown in Figure 26.

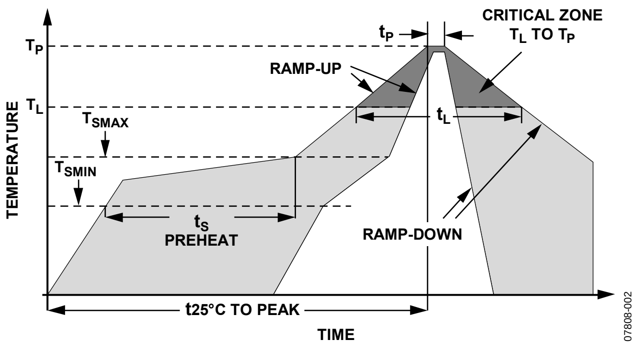


Figure 25. Recommended Soldering Profile

Table 6. Recommended Soldering Profile

Profile Feature	Sn63/Pb37	Pb-Free
Average Ramp Rate (T_L to T_P)	3°C/sec max	3°C/sec max
Preheat		
Minimum Temperature (T_{SMIN})	100°C	150°C
Maximum Temperature (T_{SMAX})	150°C	200°C
Time (T_{SMIN} to T_{SMAX})(t_S)	60 sec to 120 sec	60 sec to 180 sec
T_{SMAX} to T_L		
Ramp-Up Rate	3°C/sec max	3°C/sec max
Time Maintained Above Liquidous (T_L)		
Liquidous Temperature (T_L)	183°C	217°C
Time (t_L)	60 sec to 150 sec	60 sec to 150 sec
Peak Temperature (T_P)	240°C + 0°C/−5°C	260°C + 0°C/−5°C
Time Within 5°C of Actual Peak Temperature (t_P)	10 sec to 30 sec	20 sec to 40 sec
Ramp-Down Rate	6°C/sec max	6°C/sec max
Time 25°C to Peak Temperature	6 minutes max	8 minutes max

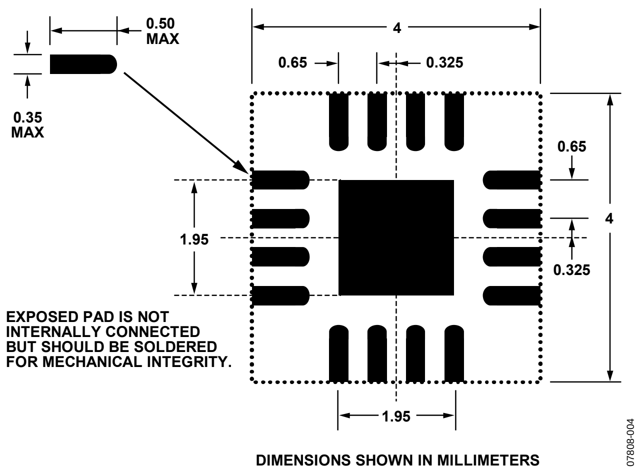


Figure 26. Recommended PCB Layout

OUTLINE DIMENSIONS

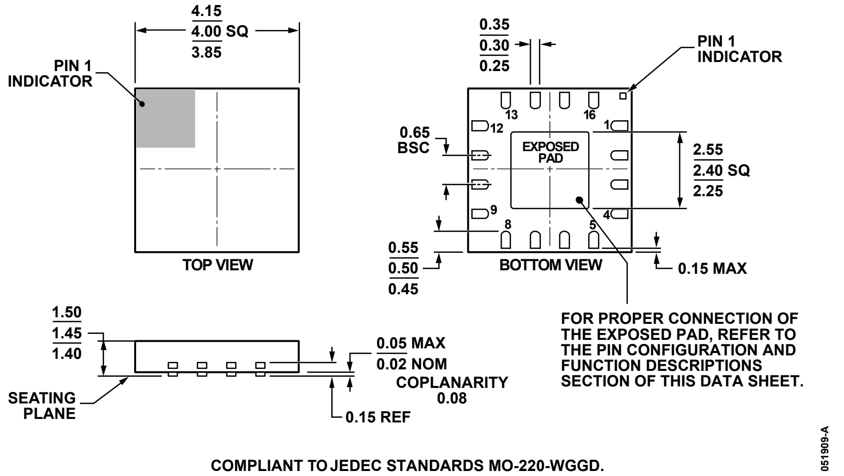


Figure 27. 16-Lead Lead Frame Chip Scale Package [LFCSP_LQ]
4 mm × 4 mm Body, 1.45 mm Thick Quad
(CP-16-14)
Dimensions shown in millimeters

ORDERING GUIDE

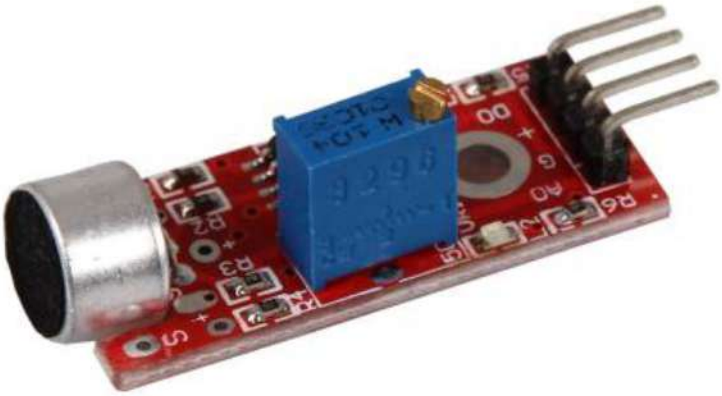
Model[1]	Measurement Range	Specified Voltage	Temperature Range	Package Description	Package Option
ADXL335BCPZ	±3 *g*	3 V	−40°C to +85°C	16-Lead LFCSP_LQ	CP-16-14
ADXL335BCPZ–RL	±3 *g*	3 V	−40°C to +85°C	16-Lead LFCSP_LQ	CP-16-14
ADXL335BCPZ–RL7	±3 *g*	3 V	−40°C to +85°C	16-Lead LFCSP_LQ	CP-16-14
EVAL-ADXL335Z				Evaluation Board	

[1] Z = RoHS Compliant Part.

KY-037 Microphone sensor module (high sensitivity)

Contents

Picture



Technical data / Short description

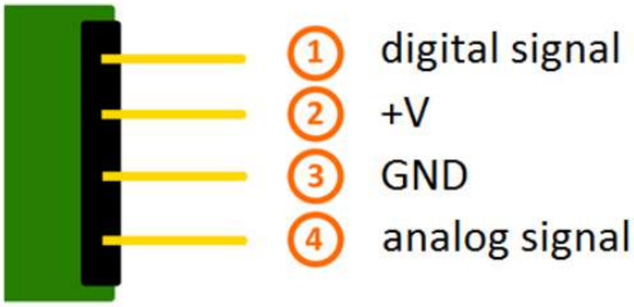
Digital Out: You can use a potentiometer to configure an extreme value for the sonic. If the value exceeds the extreme value, it will send a signal via digital out.

Analog Out: Direct microphone signal as voltage value

LED1: Shows that the sensor is supplied with voltage

LED2: Shows that a magnetic field was detected

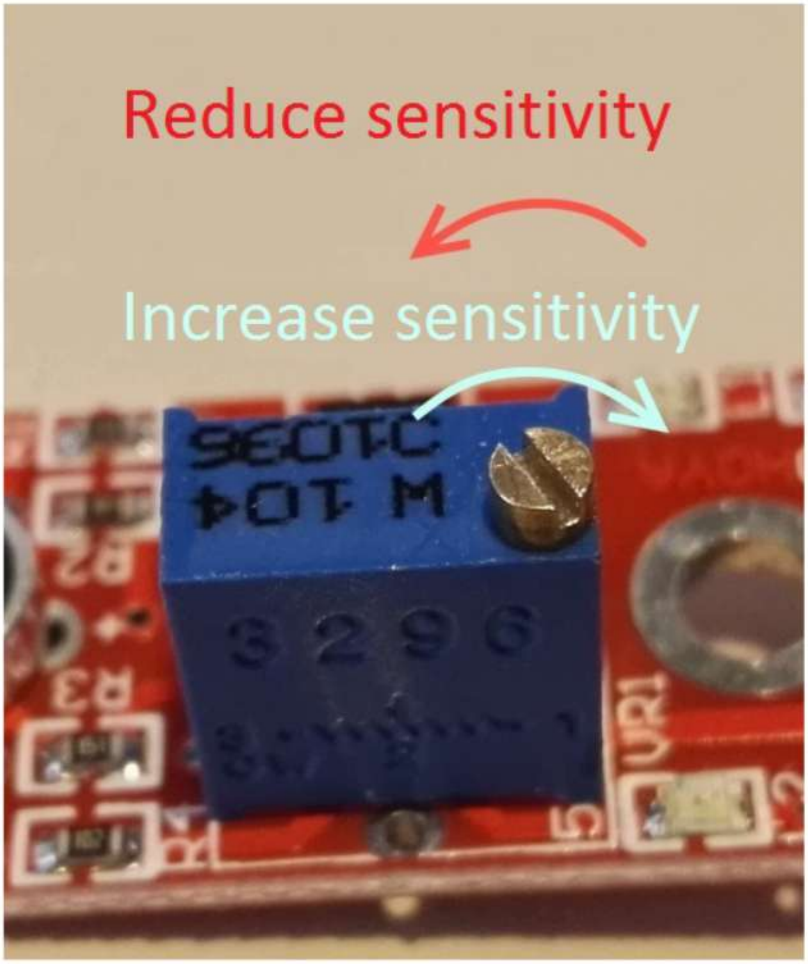
Pinout



Functionality of the sensor

The sensor has 3 main components on its circuit board. First, the sensor unit at the front of the module which measures the area physically and sends an analog signal to the second unit, the amplifier. The amplifier amplifies the signal, according to the resistant value of the potentiometer, and sends the signal to the analog output of the module.
The third component is a comparator which switches the digital out and the LED if the signal falls under a specific value.
You can control the sensitivity by adjusting the potentiometer.

Please notice: The signal will be inverted; that means that if you measure a high value, it is shown as a low voltage value at the analog output.



This sensor doesn't show absolute values (like exact temperature in °C or magneticfield strenght in mT). It is a relative measurement: you define an extreme value to a given normal environment situation and a signal will be send if the measurement exceeds the extreme value.

It is perfect for temperature control (KY-028), proximity switch (KY-024, KY-025, KY-036), detecting alarms (KY-037, KY-038) or rotary encoder (KY-026).

Code example Arduino

The program reads the current voltage value which will be measured at the output pin and shows it via serial interface.

Additional to that the status of the digital pin will be shown at the terminal which means if the extreme value was exceeded or not.

```
// Declaration and initialization of the input pin
int Analog_Eingang = A0; // X-axis-signal
int Digital_Eingang = 3; // Button

void setup ()
{
```



```
  pinMode (Analog_Eingang, INPUT);
  pinMode (Digital_Eingang, INPUT);

  Serial.begin (9600); // Serial output with 9600 bps
}

// The program reads the current value of the input pins
// and outputs it via serial out
void loop ()
{
  float Analog;
  int Digital;

  // Current value will be read and converted to voltage
  Analog = analogRead (Analog_Eingang) * (5.0 / 1023.0);
  Digital = digitalRead (Digital_Eingang);

  //... and outputted here
  Serial.print ("Analog voltage value: "); Serial.print (Analog, 4);  Serial.print ("V, ");
  Serial.print ("Extreme value: ");

  if(Digital==1)
  {
      Serial.println (" reached");
  }
  else
  {
      Serial.println (" not reached yet");
  }
  Serial.println ("----------------------------------------------------------------");
  delay (200);
}
```

Connections Arduino:

digital signal	=	[Pin 3]
+V	=	[Pin 5V]
GND	=	[Pin GND]
analog signal	=	[Pin 0]

Example program download

ARD_Analog-Sensor

Code example Raspberry Pi

!! Attention !! Analog Sensor !! Attention !!

Unlike the Arduino, the Raspberry Pi doesn't provide an ADC (Analog Digital Converter) on its Chip. This limits the Raspbery Pi if you want to use a non digital Sensor.

To evade this, use our *Sensorkit X40* with the *KY-053* module, which provides a 16 Bit ADC, which can be used with the Raspberry Pi, to upgrade it with 4 additional analog input pins. This module is connected via I2C to the Raspberry Pi.
It measures the analog data and converts it into a digital signal which is suitable for the Raspberry Pi.

So we recommend to use the KY-053 ADC if you want to use analog sensors along with the Raspberry Pi.

For more information please look at the infosite: KY-053 Analog Digital Converter

!! Attention !! Analog Sensor !! Attention !!