

MOTOR FAULT PREDICTION USING MACHINE LEARNING

A Project Report Submitted By

SHREE HARSHA N K
SHRINITH
SUHAS O
YASHAWANTH P

4NM19EC162
4NM19EC166
4NM19EC180
4NM19EC201

Under the Guidance of
Dr. Roopa B Hegde
Associate Professor

*in partial fulfillment of the requirements for the award of the Degree
of*

Bachelor of Engineering
in
Electronics and Communication Engineering

from
Visvesvaraya Technological University, Belagavi



Nitte-574110, Karnataka, India

May-2023



NITTE
EDUCATION TRUST

N.M.A.M. INSTITUTE OF TECHNOLOGY

(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)

Nitte – 574 110, Karnataka, India

ISO 9001:2015 Certified, Accredited with 'A' Grade by NAAC

☎: 08258 - 281039 – 281263, Fax: 08258 – 281265

Department of Electronics and Communication Engineering

Certificate

Certified that the project work entitled

"Motor Fault Prediction Using Machine Learning"

is a bonafide work carried out by

Shree Harsha N K (4NM19EC162), Shrinith (4NM19EC166),

Suhas O (4NM19EC180) & Yashawanth P (4NM19EC201)

in partial fulfillment of the requirements for the award of Bachelor of Engineering

Degree in Electronics and Communication Engineering

prescribed by Visvesvaraya Technological University, Belagavi

during the year 2022-2023.

It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library.

The project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the Bachelor of Engineering Degree.

Signature of the Guide

Signature of the HOD

Signature of the Principal

Semester End Viva Voce Examination

Name of the Examiners

Signature With Date

1. _____

2. _____

Abstract

Motors are essential components in industries, as they perform various tasks such as generating power, drilling for oil and gas, extracting minerals from mines, and manufacturing goods. However, motors can experience faults and failures, which can cause production downtime and financial losses. To prevent such losses, it is important to detect faults in motors before they occur. This is where machine learning techniques can be useful. The application of these techniques makes it possible to monitor the motor's condition and usage patterns and predict when a fault is likely to occur. With this information, industries can plan maintenance activities proactively, preventing unplanned downtime and optimizing production efficiency. The use of machine learning in motor fault detection is gaining popularity due to its ability to analyze large amounts of data quickly and accurately. This technique can detect subtle changes in the motor's performance, such as changes in temperature, vibration, and noise, that can indicate potential faults. By monitoring these changes, machine learning algorithms can detect early signs of motor wear and tear, and identify patterns of failure. Hence fault detection based on motor conditions and the application of machine learning techniques have great scope and can be used for optimal fault detection in motors at a proper time for avoiding losses.

Acknowledgement

Our project would not have been successful without the encouragement, guidance and support by various personalities. First and foremost, we would like to express our sincere gratitude towards our project guide **Dr. Roopa B Hegde**, Associate Professor, Department of Electronics and Communication Engineering, N. M. A. M. Institute of Technology, Nitte, for her guidance, encouragement and inspiration throughout the project work.

We extend our gratitude to **Dr. K V S S S S Sairam**, Professor and Head, Department of Electronics and Communication Engineering, N. M. A. M. Institute of Technology, Nitte, for his encouragement and providing necessary facilities in the department.

We wish to acknowledge the support of **Dr. Niranjana N. Chiplunkar**, Principal, N. M. A. M. Institute of Technology, Nitte, for providing motivational academic environment.

We would like to express our sincere thanks to all the teaching and non-teaching staff of the department of Electronics and Communication Engineering, N. M. A. M. Institute of Technology, Nitte, for their patience and help during our project work.

Finally, we would like to thank all our friends and well-wishers who have helped us whenever needed and supported us.

Shree Harsha N K
Shrinith
Suhas O
Yashawanth P

Table of Contents

| | |
|---|------------|
| Abstract | i |
| Acknowledgement | iii |
| Table of Contents | v |
| List of Figures | vii |
| List of Tables | ix |
| 1 Introduction | 1 |
| 1.1 General Introduction | 1 |
| 1.2 Aim | 1 |
| 1.3 Objectives | 2 |
| 1.4 Problem Formulation | 2 |
| 1.5 Proposed Method | 2 |
| 1.6 Methodology | 3 |
| 1.7 Literature Survey | 4 |
| 1.8 Organization of the Report | 5 |
| 2 Block Diagram Description | 7 |
| 2.0.1 Preprocessing | 9 |
| 2.0.2 Feature Selection | 9 |
| 2.0.3 Training on the Machine-Learning model | 10 |
| 2.0.4 The steps involved in Machine Learning | 10 |
| 2.0.5 Model Evaluation Parameters | 11 |
| 2.1 ADXL345 Interfacing | 12 |
| 3 Hardware and Software Description | 15 |
| 3.1 Hardware Description | 15 |
| 3.1.1 ADXL 345 Accelerometer sensor | 15 |
| 3.1.2 RS-555 12V Dc Motor | 16 |
| 3.2 Software Description | 17 |
| 3.2.1 Python | 17 |
| 3.2.2 Google Colaboratory | 18 |
| 4 Methodology and Implementation | 21 |
| 4.1 Machine learning model for online dataset | 21 |

TABLE OF CONTENTS

| | | |
|----------|--|-----------|
| 4.2 | Machine learning model for real-time dataset | 24 |
| 5 | Algorithms | 27 |
| 5.1 | Theoretical Background | 27 |
| 5.1.1 | Support Vector Machine | 27 |
| 5.1.2 | Decision Tree | 27 |
| 5.1.3 | k-Neighbors Classifier | 29 |
| 5.1.4 | Random Forest Classifier | 30 |
| 6 | Results | 31 |
| 6.1 | Model training evaluation parameters | 31 |
| 6.2 | Model validation evaluation parameters | 33 |
| 6.3 | Model testing evaluation parameters | 35 |
| | Bibliography | 39 |

List of Figures

| | | |
|-----|--|----|
| 2.1 | Work flow of fault detection in the motor. | 7 |
| 2.2 | Work Flow: (a) When the number of features is three, (b) When the number of features is eight | 8 |
| 2.3 | ADXL345 Interfacing with Arduino. | 13 |
| 3.1 | ADXL345 accelerometer sensor. | 16 |
| 3.2 | RS-555 12V DC Motor. | 17 |
| 3.3 | Training in Google Colab. | 19 |
| 4.1 | Histogram of features. | 22 |
| 4.2 | Plot of the acquired sensor values. | 24 |

List of Tables

| | | |
|-----|--|----|
| 6.1 | Training parameters for the online dataset for binary classification | 31 |
| 6.2 | Training parameters for the online dataset for multiclass classification . . | 32 |
| 6.3 | Training parameters for the real-time dataset for binary classification . . | 32 |
| 6.4 | Validation parameters for the online dataset for binary classification . . . | 33 |
| 6.5 | Validation parameters for the online dataset for multiclass classification | 34 |
| 6.6 | Validation parameters for the real-time dataset for binary classification . | 34 |
| 6.7 | Testing parameters for the online dataset for binary classification | 35 |
| 6.8 | Testing parameters for the online dataset for multiclass classification . . | 36 |
| 6.9 | Testing parameters for the real-time dataset for binary classification . . . | 37 |

Chapter 1

Introduction

1.1 General Introduction

Motors are used in many industrial processes for various needs. It works in various conditions such as high temperature, electromagnetic signal interference, noisy environment for both signal and physical impairments, the vibration of nearby machines, and environmental stress, which can easily damage the internal parts of the motor, to avoid failure and maintain a motor's life and efficiency, they require maintenance regularly which happens to be one of the most critical tasks that all industries tend to focus. Preventive maintenance, predictive maintenance, reactive maintenance, and condition-based maintenance are the four types of maintenance strategies followed for any electric motor. Maintenance that is done on a regular basis and can help prevent motors from failing is preventive maintenance. Maintenance done to reduce maintenance costs by detecting problems at an early stage and dealing with them is predictive maintenance. Maintenance action taken on the defective motor after the breakdown of the motor is called reactive maintenance. Maintenance strategy that monitors the actual condition of the motor to decide the need for inspection or fixing is condition-based maintenance. Among all the four types of maintenance, it is better to follow preventive maintenance for preventing motors from failing. Hence in the project, early detection of motor fault using the machine learning approach is addressed.

The project aims to collect representative data from both conditioned and faulty motors, analyse the data using signal processing techniques such as frequency-domain analysis, time-domain analysis, and further develop a machine learning model to detect the motor faults based on analysed sensor data obtained, accomplishing condition-based maintenance.

1.2 Aim

To develop a machine-learning-based system that accurately detects motor faults utilizing the sensor data and other relevant features thus minimizing the losses incurred due to conventional methods of detection and diagnosis of motor faults.

1.3 Objectives

- To collect the required dataset.
- To select the appropriate features.
- To classify the motor into normal motor and fault motor using machine learning.

1.4 Problem Formulation

Motor faults can cause significant downtime and costly damage in various sectors, such as manufacturing, transportation, and energy. Traditional methods for detecting motor faults are often manual inspections or rule-based systems that are time-consuming and prone to human error. Therefore, there is a need for a reliable and accurate machine learning-based system that can automatically identify motor defects in real-time using sensor data and other relevant features. Such a system can enhance operational efficiency by minimizing downtime and reducing the cost of maintenance in industrial settings. By utilizing machine learning algorithms, such a system can analyze large amounts of data generated by motors in real-time, enabling the early detection of potential defects before they cause significant damage. It can identify patterns and anomalies that may not be detectable by manual inspections or rule-based systems, thereby providing a more comprehensive assessment of the motor's health. Moreover, the system can learn from the historical data to improve its accuracy and reliability over time, further enhancing its effectiveness. The development of a machine learning-based system for detecting motor faults in real-time has the potential to significantly improve the reliability and efficiency of industrial operations. It can reduce the need for manual inspections, minimize downtime, and prevent costly damages, resulting in substantial cost savings and enhanced operational performance.

1.5 Proposed Method

The objective of the analysis is to detect faults in a machinery system and classify the type of fault using machine learning models. The machinery fault database is a popular online dataset that contains several types of faults such as bearing faults, misalignment, unbalance, etc. The dataset is preprocessed to extract relevant features that are indicative of faults in the machinery system.

The dataset is then split into training and testing sets, and several machine learning models are trained on the data. The models considered for the analysis could include decision trees, random forests, support vector machines, and knn algorithms. The performance of each model is evaluated using appropriate metrics such as accuracy, precision, recall, F1 score, etc. The model with the best performance is chosen as the final model for detecting faults in the machinery system. For the real-time dataset, an ADXL 345 accelerometer sensor and RS-555 DC motor are used to acquire vibration signals produced by the motor in the axial, radial, and tangential directions. These signals are pre-processed and used as features for the machine learning models. The dataset is again split into training and testing sets, and the same machine learning models used for the previous analysis are trained on the data.

The performance of each model is evaluated using the same metrics used in the previous analysis. The model with the best performance is chosen as the final model for detecting faults in the motor. The chosen model should be able to detect faults in the motor and classify the type of fault accurately.

The analysis involves preprocessing the dataset, training and evaluating several machine learning models, and selecting the best-performing model for detecting faults in the machinery system. The same process is repeated for the real-time dataset acquired using an ADXL accelerometer sensor and RS-555 DC motor to detect faults in the motor.

1.6 Methodology

- An online machinery fault database is accessed through Kaggle [1]. This dataset is used for fault detection through binary classification, dividing the dataset into two classes - normal and faulty, and multiclass classification, dividing the dataset into multiple classes based on the type of fault. A suitable machine learning algorithm is trained on the pre-processed dataset to classify each sample as normal or faulty or classify each sample into one of the fault categories.
- A real-time dataset has been acquired using an ADXL345 accelerometer that captures acceleration data at a sampling frequency of 100 Hz. The data is collected with a delay of 10 milliseconds for a duration of 100 seconds, resulting in a dataset of 10,000 samples. The accelerometer captures data in three axes - x, y, and z, providing a three-dimensional view of the vibration signals. To detect faults in the machinery system, binary classification is performed on

the dataset. The dataset is labeled as normal or faulty, with the fault category not being specified.

1.7 Literature Survey

Saud Altaf et al. [2] proposed the concept of fault diagnosis in an industrial motor using knowledge-level modelling, motor current signature analysis (MCSA), the obtained data from MCSA sensors was processed using a combination of fast Fourier transform (FFT) and Hilbert transform (HT), filtered data from Kalman filter provided the features, from which significant features were selected and applied to a supervised neural network which yielded an accuracy of 96 percent in the detection of faults, the proposed method was successful in significantly reducing the cost invested on expensive diagnostic equipment used in invasive fault diagnosis.

In the paper presented by Chun-Yao Lee et al. [3], the detection of faults in an induction motor is performed using multiresolution analysis (MRA), which is a useful method to analyse the faulty motor, and the data was processed and transformed using Hilbert Huang transform, features were selected based on correlation and fitness values, trained using an artificial neural network (ANN) which yielded an accuracy of 93 percent in the detection of induction motor faults.

The fault detection and classification system discussed in the paper presented by Majid Hussain et al. [4] proposed an Induction motor fault detection and classification system using signal processing techniques such as fast Fourier transform (FFT), continuous wavelet transform (CWT) and short-time Fourier transform (STFT), and various deep learning (DL) methods were adopted to detect and classify the stator winding faults occurring in the three-phase induction motors and the comparative investigation among the DL models confirmed the superiority of the long short-term memory (LSTM) model which achieved an accuracy of 98.87 percent in identifying the faults.

Thomas Amanuel et al. [5] proposed the detection of healthy and faulty conditions of the motor. The methods used are Fast Fourier transform, wavelet transform, and short-time Fourier transform. The proposed wavelet transform method proves that detection accuracy is high when compared to the existing algorithms by including the FFT algorithm. The fault detection is based on the proposed wavelet transform, which detects the rotor broken bar fault easily.

Yasmina Bella et al. [6] proposed fault monitoring of the rolling element bearings. The methods used are Spectral analysis techniques, Quadratic time-frequency distributions, and short-time Fourier transform. For different rotational values, the graphs of different distributions obtained by the time-frequency methods are analyzed.

Prashant D.Bharad et al. [7] proposed the detection of faults in motor bearings. The methods used are independent component analysis and Fast Fourier Transform. The feature extraction is done using Independent Component Analysis (ICA). This method was applied to detect faults in the induction motor shaft and classification accuracy was found to be more than 90 percent using an ANN.

Ana L. Martinez-Herrera et al. [8] proposed the identification and classification of faults in an induction motor. Techniques used for induction motor fault detection are motor current signal analysis, Fast Fourier Transform and an index-based classifier, short-time Fourier transform, HT, and ANN. This methodology reached 100 percent of effectiveness in identifying and discriminating among a healthy motor (HLT), a motor with one broken rotor bar (1BRB), a motor with two broken rotor bars (2BRB), a motor with outer-race bearing damage (BRN), and a motor with an unbalanced mechanical load (UNB).

1.8 Organization of the Report

Chapter 1 explains the introduction.

Chapter 2 explains the block diagram description.

Chapter 3 explains the hardware and software description.

Chapter 4 explains the methodology and implementation.

Chapter 5 explains the algorithms used.

Chapter 6 explains the results obtained.

Chapter 2

Block Diagram Description

The project comprises of the aspects as shown in 2.1

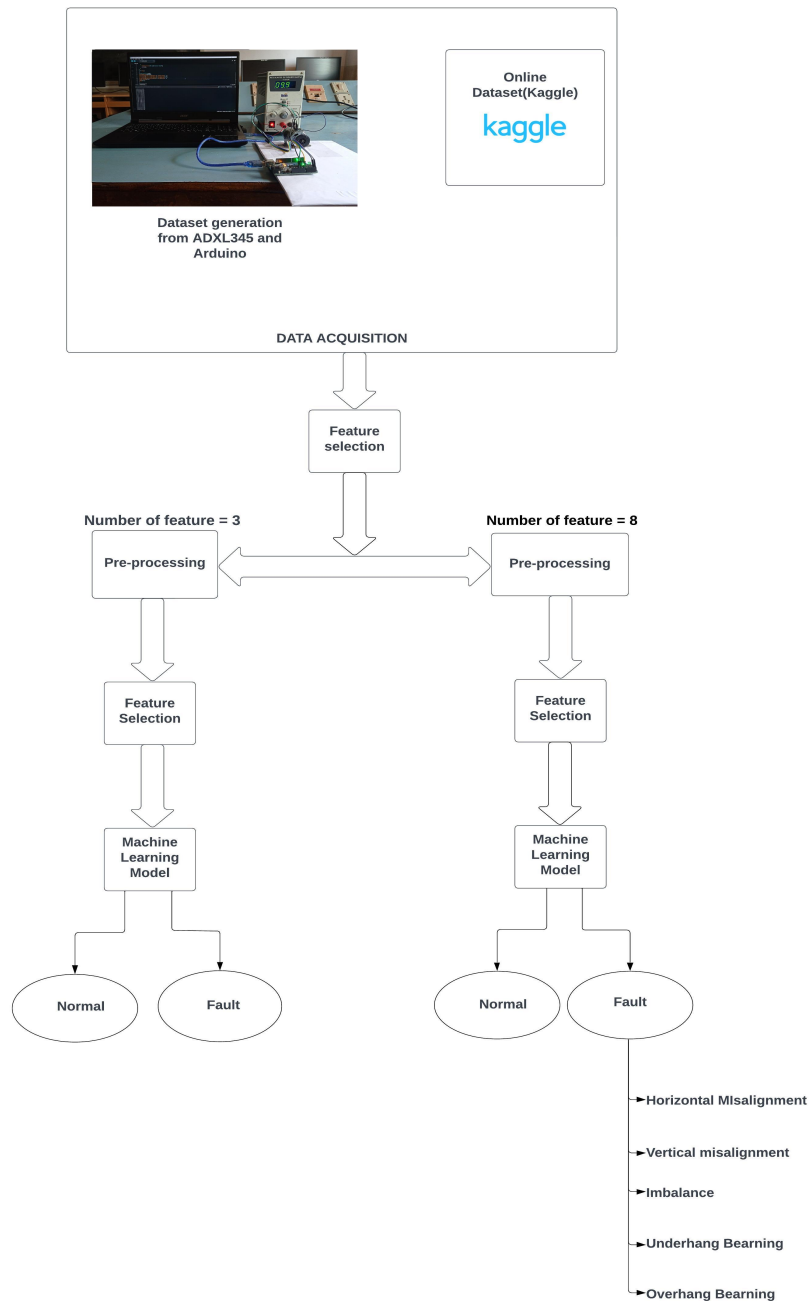


Figure 2.1: Work flow of fault detection in the motor.

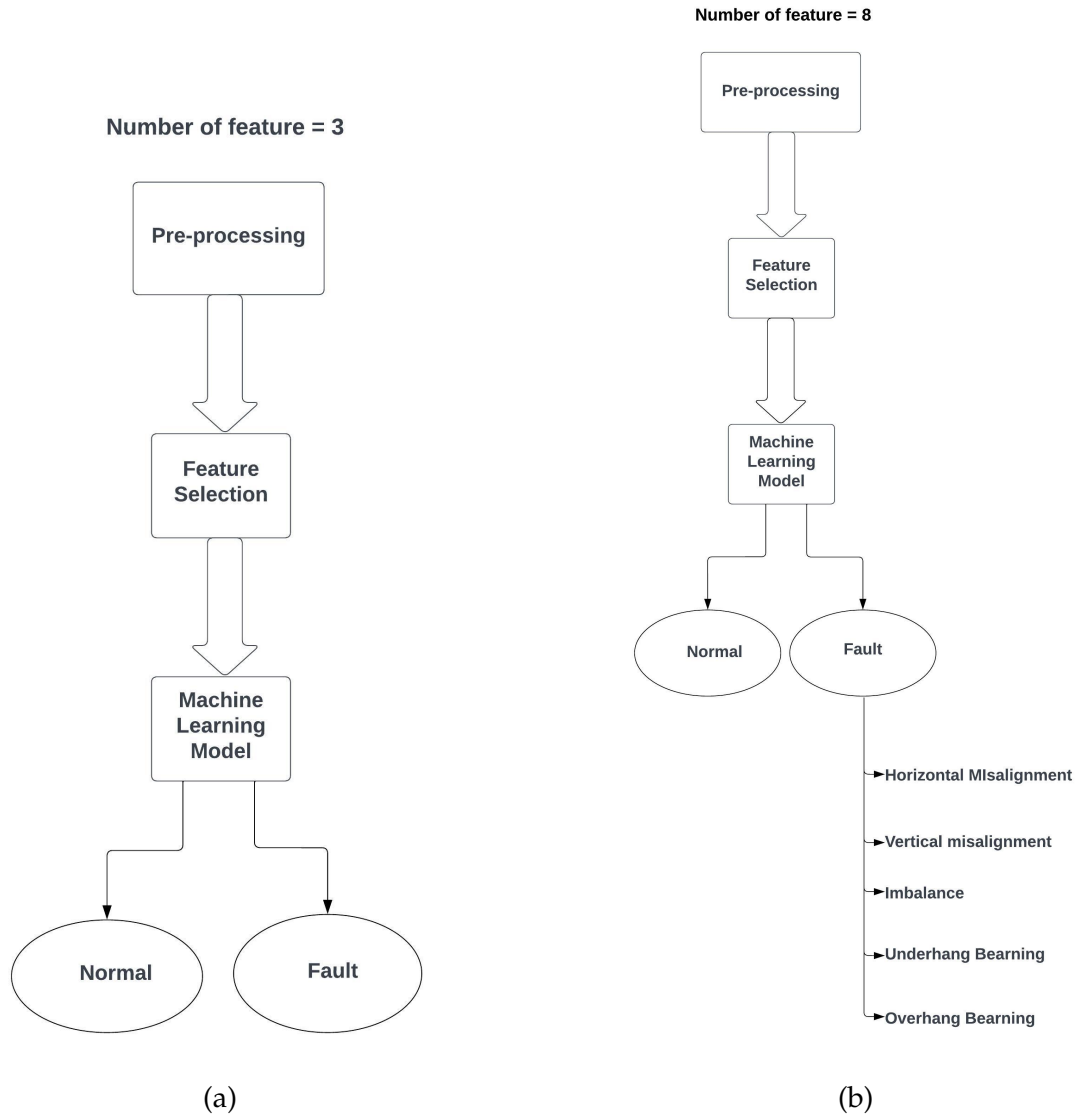


Figure 2.2: Work Flow: (a) When the number of features is three, (b) When the number of features is eight

The data acquired is initially verified for the number of features it comprises, and based on that number, the decision to follow 2.2a or 2.2b of the block diagram is made. The input given to the machine learning model as represented in 2.1 will be preprocessed sensor data obtained through the preprocessing block. Preprocessing block is used to remove noise, and unwanted data, or eliminate variations that arise during the acquisition of the data without evading essential information. Feature selection methods are used to achieve efficient data reduction by taking only relevant features expected by the machine learning model. The obtained features are trained with machine-learning models to predict the presence of a fault.

2.0.1 Preprocessing

Preprocessing refers to the set of procedures or methods used to prepare raw, unprocessed data for further analysis or modeling. This step is essential for ensuring that the data is clean, consistent, and formatted correctly before being used for analysis or modeling.

The process of preprocessing includes several key steps, such as data cleaning, transformation, and formatting. Data cleaning involves removing any inconsistencies, errors, or missing values in the dataset. Transformation involves converting the data into a suitable format for analysis or modeling, such as normalizing or scaling the data. Formatting involves structuring the data in a way that is easily understood and used by algorithms. The goal of preprocessing is to improve the quality and reliability of the data by removing any errors, inconsistencies, or outliers that could affect the accuracy of the analysis or modeling. It also aims to extract meaningful insights or patterns from the data that can be used to inform decision-making or generate predictions. Data preprocessing is a critical step in the data analysis or modeling process. Without proper preprocessing, the quality and accuracy of the results can be compromised, leading to incorrect or misleading conclusions. By taking the time to properly preprocess the data, analysts, and data scientists can ensure that the data is ready for analysis or modeling and that the insights and patterns extracted from it are accurate and reliable.

2.0.2 Feature Selection

Feature selection is the process of selecting a subset of the most informative and relevant features from a larger set of available features in a dataset. The primary objective of feature selection is to improve the performance and interpretability of machine learning models by reducing the complexity and noise in the data. By selecting only the most informative features, feature selection can help to improve the accuracy and efficiency of a model, while also making it easier to interpret the results.

The feature selection process involves evaluating each feature's relevance and importance to the model's performance, and then selecting only the features that are most valuable. This is typically done using statistical or machine learning techniques, such as correlation analysis, mutual information, or decision trees. The benefits of feature selection in machine learning are numerous. First, it can help to reduce the dimensionality of the dataset, making it easier to work with and analyze. By reducing the number of features, it can also improve the accuracy and

efficiency of the model, making it faster and more effective. Additionally, by removing redundant or unimportant features, feature selection can help to eliminate noise and improve the model's interpretability, making it easier to understand and analyze.

Overall, feature selection is a crucial step in the machine learning process, as it can help to improve the accuracy, efficiency, and interpretability of models. By selecting only the most informative and relevant features, feature selection can help to optimize the performance of machine learning models, making them more effective and easier to understand.

2.0.3 Training on the Machine-Learning model

The act of building a mathematical or statistical model that can learn from data in order to produce predictions, classifications, or other sorts of analysis is known as training a machine learning model. A labelled dataset with input data also referred to as features and corresponding output data is used to train the model. The objective of the training procedure is to modify the model's parameters in order to enable accurate output prediction for new, unseen data, and to select the model with the highest performance, the training procedure is usually repeated several times using various combinations of hyperparameters, model architectures, and datasets. It is important to properly validate the trained model to ensure its accuracy, reliability, and generalization capability.

2.0.4 The steps involved in Machine Learning

1. **Data Splitting:** The first step is to split the data into training and testing datasets using the train test split function. The data is normalized using the using standard scalar, and the target variable y is passed along with the test size and random state.
2. **Grid Search:** GridSearchCV() function is used to perform a hyperparameter tuning. Here, we have set the range of the hyperparameters to be tried in a dictionary format called parameter grid.
3. **Model Fitting:** After finding the best hyperparameters, the model is fitted on the training dataset using an appropriate function with the best hyperparameters obtained from the GridSearchCV() function.

4. Prediction: The trained model is used to make predictions on the testing dataset using the `predict()` function.
5. Evaluation: Finally, the performance of the model is evaluated by printing the classification report and confusion matrix and accuracy score of both the training and testing datasets.
6. Model Saving: The trained model is saved using the `pickle.dump()` function.

2.0.5 Model Evaluation Parameters

Model evaluation parameters are essential tools used in assessing the quality and effectiveness of machine learning models. These parameters help to determine whether a model has been properly trained and is capable of producing reliable and accurate predictions on new data. The significance of model evaluation parameters lies in their ability to help data scientists and machine learning engineers identify and address issues with their models. By using these parameters, one can determine how well a model performs in terms of accuracy, precision, recall, F1 score, AUC-ROC, and other performance metrics. For instance, accuracy measures the percentage of correctly classified instances in a dataset, while precision measures the proportion of correctly classified positive instances among all positive predictions. Recall measures the proportion of correctly classified positive instances among all true positives in a dataset. F1 score combines precision and recall to provide an overall assessment of a model's performance. AUC-ROC measures the model's ability to distinguish between positive and negative classes by plotting the true positive rate against the false positive rate, by evaluating a model based on these metrics, one can determine its strengths and weaknesses and make necessary improvements to enhance its performance. They can also compare the performance of different models and select the best one for a given task. In summary, model evaluation parameters are crucial in ensuring the quality and effectiveness of machine learning models. They enable data scientists to evaluate models accurately, identify issues, and make improvements to enhance their performance.

The evaluation parameters used in the project to determine the strengths and effectiveness are discussed below.

Accuracy: The probability of correctly predicting the results of a test.

$$Accuracy = \frac{TruePositives + TrueNegatives}{TotalValues} \quad (2.1)$$

Precision: Precision is the measure of identifying the results correctly from the total true cases.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \quad (2.2)$$

Recall: Recall is the measure of correctly predicting the true results in a test scenario.

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \quad (2.3)$$

F1 Score: It is a metric that computes how many times a model made a correct prediction across the entire dataset.

$$F1Score = \frac{2 * (Precision * Recall)}{Precision + Recall} \quad (2.4)$$

where

- True positives (TP) are the cases where the model predicted a positive outcome, and the actual outcome was also positive.
- True negatives (TN) are the cases where the model predicted a negative outcome, and the actual outcome was also negative.
- False positives (FP) are the cases where the model predicted a positive outcome, but the actual outcome was negative.
- False negatives (FN) are the cases where the model predicted a negative outcome, but the actual outcome was positive.

2.1 ADXL345 Interfacing

The ADXL345 is a comprehensive 3-axis acceleration measurement device having a measurement range of ± 2 g, ± 4 g, ± 8 g, or ± 16 g that can be selected. It monitors both dynamic accelerations caused by motion or shock and static acceleration caused by gravity, allowing it to be utilised as a tilt sensor. 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 resistance against forces due to applied acceleration. Differential capacitors, which include separate fixed plates and plates attached to the moving mass, are used to measure the bending of the structure. Acceleration causes the differential capacitor to become unbalanced and the proof mass to deflect, which produces a sensor output whose amplitude is proportional to acceleration. The magnitude and polarity of the acceleration are determined using phase-sensitive demodulation.

The circuit connection is as follows:

Connect A4 pin (SDA) of Arduino -> SDA pin of ADXL345

Connect A5 pin (SCL) of Arduino -> SCL pin of ADXL345

Connect GND of Arduino -> GND pin of ADXL345

Connect 5V of Arduino -> Vcc of ADXL345

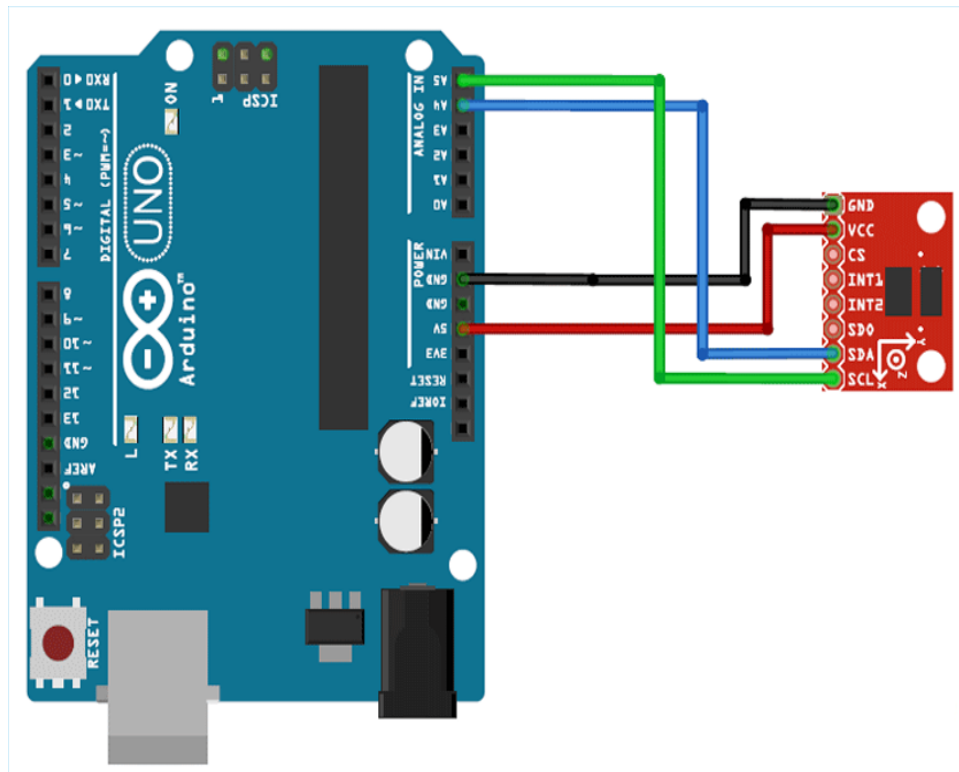


Figure 2.3: ADXL345 Interfacing with Arduino.

Chapter 3

Hardware and Software Description

3.1 Hardware Description

3.1.1 ADXL 345 Accelerometer sensor

The ADXL345 is a small, thin, low-power MEMS (Micro-Electromechanical Systems) sensor that is widely used in a variety of applications. Its digital I2C/SPI interface makes it easy to communicate with microcontrollers or other devices. The accelerometer sensor can measure acceleration in three perpendicular axes (x, y, and z) with a range of up to 16g, making it suitable for many applications that require high accuracy. One of the most significant advantages of the ADXL345 accelerometer sensor is its versatility. It can be used for tilt sensing, motion detection, vibration monitoring, and impact detection. This makes it a desirable choice for many different industries, including automotive, aerospace, robotics, and consumer electronics. Another advantage of the ADXL345 is its excellent resolution, which enables it to detect even small changes in acceleration. Additionally, the sensor has low power consumption, making it suitable for battery-powered applications. The ADXL345 also has integrated motion detection, which allows it to detect changes in motion without the need for additional sensors.

The ADXL345 accelerometer sensor is adaptable to different applications due to its various configuration options. It has adjustable bandwidth, data rate, and power settings, allowing it to be optimized for different use cases. The sensor also has built-in self-test and calibration features, which enable it to maintain its accuracy over time. Overall, the ADXL345 accelerometer sensor is a popular choice for many applications due to its high accuracy, low power consumption, and versatility. Its small size and digital interface make it easy to integrate into different systems.

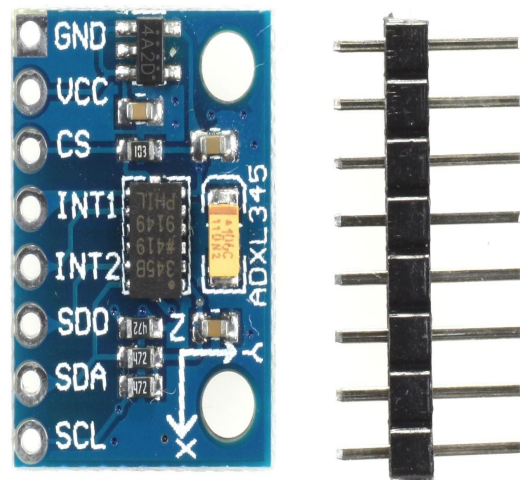


Figure 3.1: ADXL345 accelerometer sensor.

3.1.2 RS-555 12V Dc Motor

The RS-555 motor is a type of permanent magnet-brushed DC motor that is designed for high-performance applications. It has a compact size, with a diameter of 36mm and a length of 66mm, which makes it suitable for use in smaller devices and tight spaces. Its power rating of 30 W and 12V DC-rated voltage allow it to provide strong and reliable performance in a variety of applications. One of the key advantages of the RS-555 motor is its high speed. It has a no-load speed ranging between 5500 to 6500 revolutions per minute (RPM), which makes it suitable for applications that require fast and precise movements. Additionally, the motor has a long service life and minimal noise, making it a preferred choice for many industries.

The RS-555 motor is used in various applications, such as robotics, automation equipment, electric vehicles, and many more. Due to its efficiency and reliability, it is widely used in industries that require high-performance motors. For instance, in robotics, it is used to power the movement of robotic arms, grippers, and other components. In electric vehicles, it is used to drive the wheels and provide the necessary torque for movement. Another significant advantage of the RS-555 motor is its ability to tolerate significant mechanical stress and high temperatures. Its design enables it to operate in harsh environments, making it suitable for industrial applications that require robust motors. Additionally, the motor is easy to install

and maintain, which makes it a preferred choice for many manufacturers.

Overall, the RS-555 12V DC motor is a high-performance motor that is widely used in various industries due to its speed, efficiency, and reliability. Its small size, long service life, and ability to withstand mechanical stress and high temperatures make it a popular choice for many different applications.



Figure 3.2: RS-555 12V DC Motor.

Other Hardware requirements:

- Breadboards
- DC Supply
- Jumper Wires

3.2 Software Description

3.2.1 Python

Python is a high-level programming language that offers several benefits for developers. One of its significant advantages is its flexibility, which allows it to be used for a wide range of applications, including web development, data analysis, machine learning (ML), and artificial intelligence (AI). Its versatility also makes it

a popular choice for both beginners and experienced developers. Another advantage of Python is its platform freedom. Python can be run on various operating systems, including Windows, Linux, and macOS. This makes it an accessible language for developers working on different platforms. Additionally, Python has a vast community of developers who contribute to its development and support.

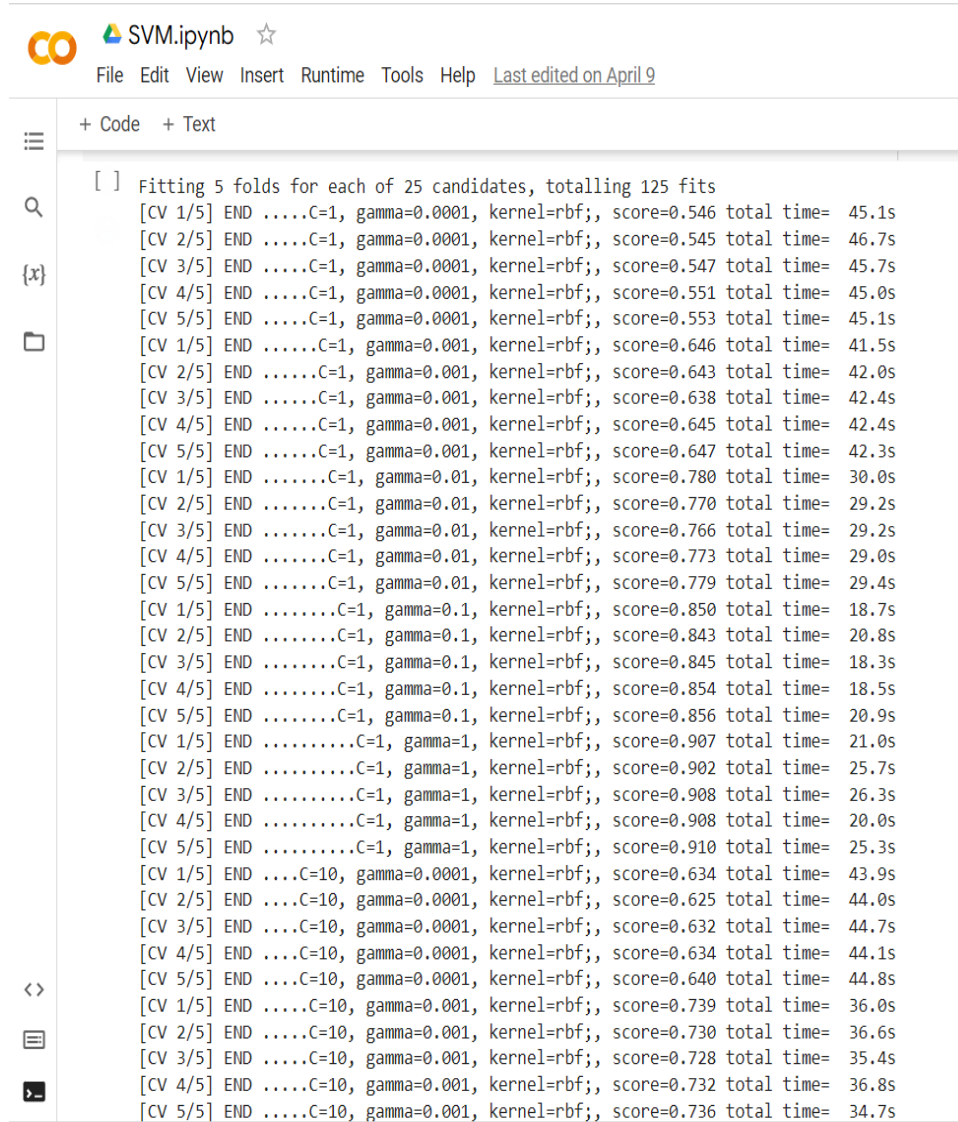
Python provides access to a broad range of excellent libraries and frameworks for AI and ML, making it a popular choice for these applications. Its libraries, such as NumPy, Pandas, and Scikit-learn, allow for data manipulation, data analysis, and scientific computing. Additionally, frameworks like TensorFlow and PyTorch provide efficient and scalable solutions for deep learning and neural network development. It is also known for its simplicity and consistency, which makes it easy to learn and use. Its syntax is clear and straightforward, making it accessible for developers of all levels. The language's consistency and readability also make it easy for developers to collaborate on projects, as the code is easy to understand and maintain. It supports the tasks of data preprocessing, model creation, training, and testing, making it a popular choice for data analysis and ML applications. Its ease of use and extensive library support make it a powerful tool for developing complex machine-learning models quickly and efficiently.

Overall, Python's versatility, platform freedom, excellent libraries and frameworks for AI and ML, simplicity, and consistency make it a popular choice for developers. Its ease of use and support for data analysis and ML tasks also makes it an essential tool in various industries, including finance, healthcare, and technology.

3.2.2 Google Colaboratory

Google Colaboratory (Google Colab) is a cloud-based tool offered by Google that allows users to write, run, and share Python code in a group setting. One of its key advantages is that it provides free access to powerful hardware resources such as GPUs and TPUs. This allows users to take advantage of high-performance computing capabilities without having to purchase expensive hardware or set up their own computing infrastructure. Another benefit of Google Colab is that it comes with a rich set of pre-installed libraries, including popular ones like TensorFlow, PyTorch, and scikit-learn. These libraries are essential for working on machine learning projects, and having them pre-installed means users can start working on their projects immediately without the need to install additional software.

Google Colab is an appealing option for training machine learning algorithms



```
[ ] Fitting 5 folds for each of 25 candidates, totalling 125 fits
[CV 1/5] END .....C=1, gamma=0.0001, kernel=rbf; score=0.546 total time= 45.1s
[CV 2/5] END .....C=1, gamma=0.0001, kernel=rbf; score=0.545 total time= 46.7s
[CV 3/5] END .....C=1, gamma=0.0001, kernel=rbf; score=0.547 total time= 45.7s
[CV 4/5] END .....C=1, gamma=0.0001, kernel=rbf; score=0.551 total time= 45.0s
[CV 5/5] END .....C=1, gamma=0.0001, kernel=rbf; score=0.553 total time= 45.1s
[CV 1/5] END .....C=1, gamma=0.001, kernel=rbf; score=0.646 total time= 41.5s
[CV 2/5] END .....C=1, gamma=0.001, kernel=rbf; score=0.643 total time= 42.0s
[CV 3/5] END .....C=1, gamma=0.001, kernel=rbf; score=0.638 total time= 42.4s
[CV 4/5] END .....C=1, gamma=0.001, kernel=rbf; score=0.645 total time= 42.4s
[CV 5/5] END .....C=1, gamma=0.001, kernel=rbf; score=0.647 total time= 42.3s
[CV 1/5] END .....C=1, gamma=0.01, kernel=rbf; score=0.780 total time= 30.0s
[CV 2/5] END .....C=1, gamma=0.01, kernel=rbf; score=0.770 total time= 29.2s
[CV 3/5] END .....C=1, gamma=0.01, kernel=rbf; score=0.766 total time= 29.2s
[CV 4/5] END .....C=1, gamma=0.01, kernel=rbf; score=0.773 total time= 29.0s
[CV 5/5] END .....C=1, gamma=0.01, kernel=rbf; score=0.779 total time= 29.4s
[CV 1/5] END .....C=1, gamma=0.1, kernel=rbf; score=0.850 total time= 18.7s
[CV 2/5] END .....C=1, gamma=0.1, kernel=rbf; score=0.843 total time= 20.8s
[CV 3/5] END .....C=1, gamma=0.1, kernel=rbf; score=0.845 total time= 18.3s
[CV 4/5] END .....C=1, gamma=0.1, kernel=rbf; score=0.854 total time= 18.5s
[CV 5/5] END .....C=1, gamma=0.1, kernel=rbf; score=0.856 total time= 20.9s
[CV 1/5] END .....C=1, gamma=1, kernel=rbf; score=0.907 total time= 21.0s
[CV 2/5] END .....C=1, gamma=1, kernel=rbf; score=0.902 total time= 25.7s
[CV 3/5] END .....C=1, gamma=1, kernel=rbf; score=0.908 total time= 26.3s
[CV 4/5] END .....C=1, gamma=1, kernel=rbf; score=0.908 total time= 20.0s
[CV 5/5] END .....C=1, gamma=1, kernel=rbf; score=0.910 total time= 25.3s
[CV 1/5] END .....C=10, gamma=0.0001, kernel=rbf; score=0.634 total time= 43.9s
[CV 2/5] END .....C=10, gamma=0.0001, kernel=rbf; score=0.625 total time= 44.0s
[CV 3/5] END .....C=10, gamma=0.0001, kernel=rbf; score=0.632 total time= 44.7s
[CV 4/5] END .....C=10, gamma=0.0001, kernel=rbf; score=0.634 total time= 44.1s
[CV 5/5] END .....C=10, gamma=0.0001, kernel=rbf; score=0.640 total time= 44.8s
[CV 1/5] END .....C=10, gamma=0.001, kernel=rbf; score=0.739 total time= 36.0s
[CV 2/5] END .....C=10, gamma=0.001, kernel=rbf; score=0.730 total time= 36.6s
[CV 3/5] END .....C=10, gamma=0.001, kernel=rbf; score=0.728 total time= 35.4s
[CV 4/5] END .....C=10, gamma=0.001, kernel=rbf; score=0.732 total time= 36.8s
[CV 5/5] END .....C=10, gamma=0.001, kernel=rbf; score=0.736 total time= 34.7s
```

Figure 3.3: Training in Google Colab.

because it allows users to run their code on powerful hardware, making the training process much faster. Additionally, users can collaborate with others in real time, making it an excellent tool for group projects or for seeking help from other developers. Furthermore, Google Colab allows users to save their notebooks in Google Drive, making it easy to access and share their work with others. It also provides a variety of features that make it simple to visualize and analyze data, including charting libraries and the ability to upload and work with large datasets. Overall, Google Colab is a useful tool for data scientists, researchers, and developers who need to write and run Python code. Its free access to powerful hardware resources, pre-installed libraries, collaboration features, and ability to save and share work make it a popular choice for machine learning projects.

Chapter 4

Methodology and Implementation

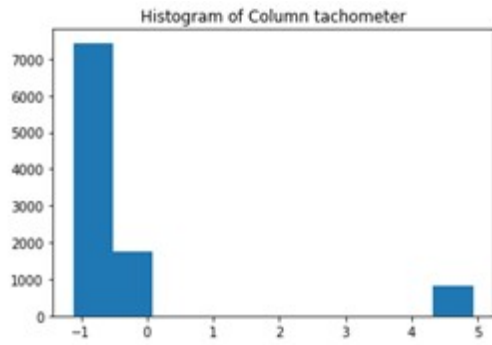
4.1 Machine learning model for online dataset

A literature review was conducted as the first step in the research, and it included analysis and descriptions of various methods for feature extraction, data pre-processing, and machine learning models.

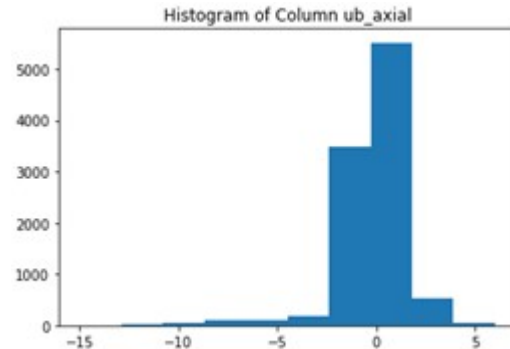
An online motor fault dataset was downloaded from the machinery fault database [1]. This database is composed of multivariate time series acquired by sensors on Spectra Quest's Machinery Fault Simulator (MFS) Alignment-Balance-Vibration (ABVT). There are six different simulated states namely normal function, imbalance fault, horizontal and vertical misalignment faults, and, inner and outer bearing faults. The motor used in the testbench is $\frac{1}{4}$ horsepower with a frequency range varying between 700-3600 rpm. The whole system weighed about 22 kg. The data was acquired using accelerometers on the radial, axial, and tangential directions, a tachometer was used to record the speed of the motor in rpm, and a microphone was used to record the sound produced in the frequency range of 20 – 20kHz. There are 49 sequences without any fault, each with a fixed rotation speed within the range from 737 rpm to 3686 rpm with steps of approximately 60 rpm. Each sequence was generated at a 50kHz sampling rate for a duration of 5 seconds, totaling 250000 samples for different frequencies and different load values. The database acquired is composed of several CSV files, each one with eight columns, and one column for each sensor.

- Column 1 → Tachometer signal that allows estimating rotation frequency.
- Column 2 to 4 → Underhang bearing accelerometer (axial, radial, tangential directions).
- Column 5 to 7 → Overhang bearing accelerometer (axial, radial, tangential directions).
- Column 8 → Microphone signal.

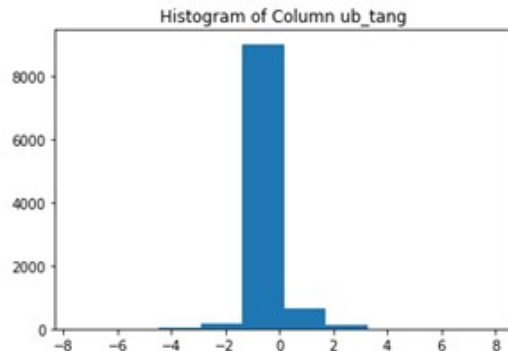
The dataset was initially tested for Gaussian normal distribution by extracting the histogram of all the features. If the data is not normally distributed, it may be necessary to transform the data before preprocessing. This can include techniques such as logarithmic or exponential transformation to make the data more normally



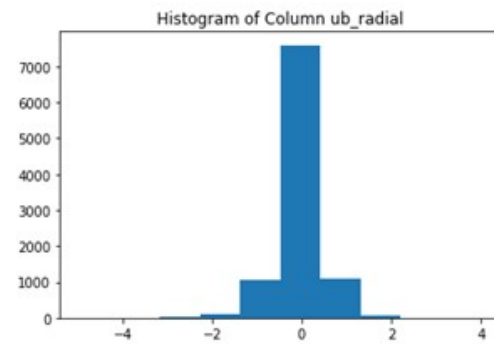
(a) Tachometer data



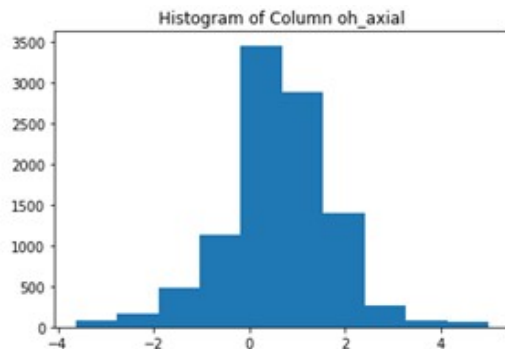
(b) Axial underhang bearing data.



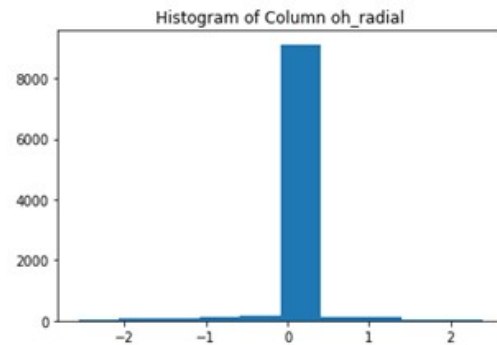
(c) Tangential underhang bearing data.



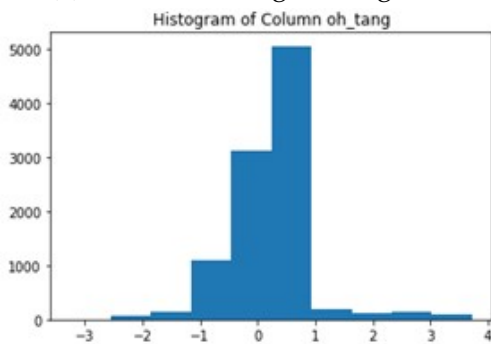
(d) Radial underhang bearing data.



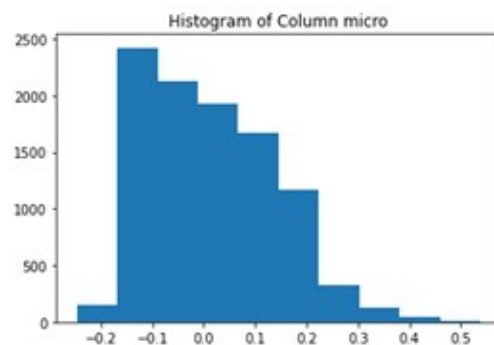
(e) Axial overhang bearing data..



(f) Radial overhang bearing data.



(g) Tangential overhang bearing data.



(h) Microphone data.

Figure 4.1: Histogram of features.

distributed and if the data is normally distributed, it is easier to identify outliers and remove them from the dataset, since they can have a significant impact on the

results of data analysis. It was noted that all the features, barring the tachometer signal showed Gaussian normal distribution characteristics as shown in Fig. 4.1a, hence it was not incorporated in the further processes.

Preprocessing of the dataset includes normalizing the data and filling in null values. A null value is a data point that is undefined or not available. A null value can occur for various reasons, such as data collection errors, or data corruption. The median method is a commonly used technique to handle null values in a dataset. The median is the middle value in a dataset, where half the values are higher than the median and half the values are lower than the median.

Data normalization is the organization of data to appear similar across all records and fields. Standard scalar is used which standardizes a feature by subtracting the mean and then scaling to unit variance using equation 4.1. Unit variance is dividing all the values by standard deviation.

$$Z = \frac{x - \mu}{\sigma} \quad (4.1)$$

$$\mu = \frac{1}{N} \sum_{i=1}^N (x_i) \quad (4.2)$$

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2} \quad (4.3)$$

Where x is the sample value

N is the number of samples

Z is the standard score

μ is the mean

σ is the standard deviation

After standardizing oversampling is done to handle the imbalance dataset. If the model is trained with an imbalanced dataset, the performance of the machine learning model may be biased toward the majority class, leading to poor performance over the minority class. Oversampling techniques are used to address this issue by increasing the number of samples in the minority class. The Borderline-SMOTE technique is used which generates synthetic samples of the minority class by interpolating between existing samples using the equation 4.4. This acquired

data is then trained on different machine-learning models and are evaluated on the model evaluation parameters discussed in 2.0.5.

$$x_{new} = x_i + \lambda * (x_{zi} - x_i) \quad (4.4)$$

Where λ is a random number in the range $[0,1]$. This interpolation will create a sample on the line between x_i and x_{zi} .

4.2 Machine learning model for real-time dataset

A real-time dataset for a normal condition motor and a faulty motor is acquired using an ADXL345 accelerometer, Arduino UNO, and RS 555 12V DC Motor. The ADXL345 is a popular 3-axis digital accelerometer sensor, capable of measuring acceleration in three perpendicular axes (x, y, and z) with a range of up to $\pm 16g$, which is a low-power sensor. The sensor attached to the is connected to Arduino UNO which has an ATmega328P, 8-bit AVR microcontroller that has been programmed to acquire the data with a delay of 10ms for 100 seconds, that is the data stored in a CSV file containing ten thousand rows of data for both the faulty and normal conditions of the motor. The features acquired are axial, tangential, and radial values of the vibration signals produced by the motor.

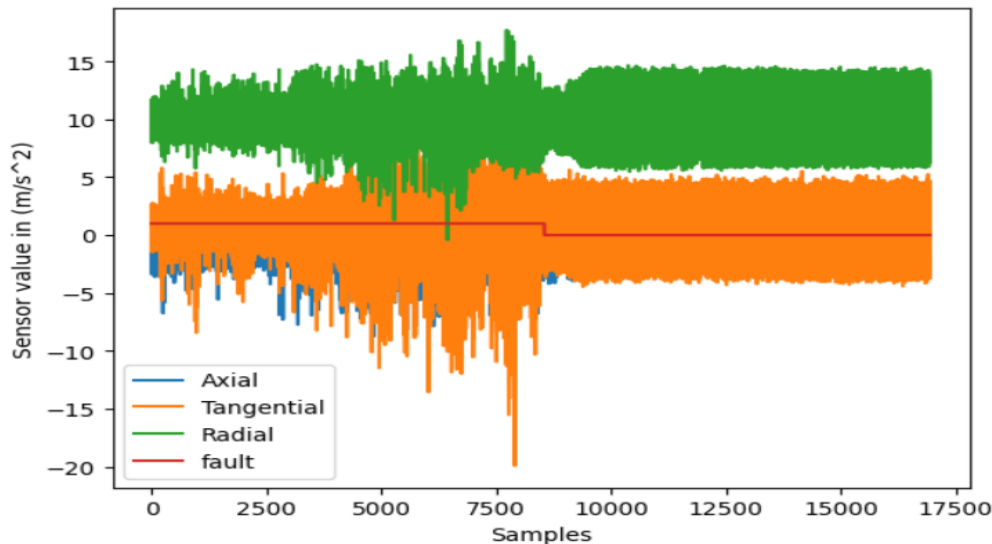


Figure 4.2: Plot of the acquired sensor values.

The graphical representation of the acquired data using the motor and ADXL345 accelerometer sensor is shown in 4.2, which provides a clear representation of the variation in data points for both normal and faulty conditions of the motor. The acquired dataset comprises three features which are accelerometer sensor values in the radial, axial, and tangential directions. This acquired data is manually cleaned

by removing all the nulls produced and acceleration due to gravity is subtracted from the radial direction data to get the data values only from the motor thus minimizing the error in the data values due to gravity. This processed dataset is trained on different machine learning models and is evaluated on the model evaluation parameters discussed in 2.0.5.

Chapter 5

Algorithms

Different machine learning algorithms like Support Vector Machine (SVM), Decision Tree (DT), k-neighbors Classifier, and Random Forest Classifier (RF), are trained on the dataset, and these trained models are tested for unseen data to evaluate their performance.

5.1 Theoretical Background

5.1.1 Support Vector Machine

SVMs are a set of supervised learning methods used for classification, regression, and outliers detection. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that it can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. As the project is dealing with non-linearly separated data, Non-linear SVM is used. If the algorithm is used with default parameters, then accuracy and other results are not as expected so according to the nature of the dataset the parameters are decided. In the project kernel parameter Radial Basis Function (RBF) is used as the dataset in concern is non-linear. The RBF kernel function for two points X_1 and X_2 computes the similarity or how close they are to each other [9].

This kernel can be mathematically represented as given in equation 5.1

$$K(X_1, X_2) = \exp (||X_1 - X_2||^2 / (2\sigma^2)) \quad (5.1)$$

5.1.2 Decision Tree

Decision Trees are a powerful machine-learning method used for solving both classification and regression problems. The primary objective of the DT algorithm is to build a model that can predict the value of a target variable by learning simple decision rules inferred from the data features. DTs are constructed by recursively splitting the dataset into smaller subsets, using the features that are most informative for the prediction task. The DT algorithm creates a tree-structured classifier where the internal nodes represent the features of the dataset, branches represent

the decision rules, and each leaf node represents the outcome. The algorithm iteratively selects the best feature that splits the dataset into subsets that have the highest homogeneity with respect to the target variable. The homogeneity is usually measured using metrics such as Gini impurity or entropy.

The DT algorithm consists of two types of nodes: decision nodes and leaf nodes. The decision nodes are used to make any decision and have multiple branches, whereas the leaf nodes are the output of those decisions and do not contain any further branches. The decisions or tests are performed based on the features of the given dataset, where each branch represents a possible outcome of the test.

One of the key advantages of DTs is that they are easy to interpret and visualize. The resulting tree can be easily understood and used for making decisions. Moreover, Decision Trees can handle both categorical and numerical data, and they can deal with missing values.

DTs are widely used in many areas, such as finance, medicine, and engineering. They have been successfully applied to problems such as fraud detection, diagnosis of diseases, and prediction of customer churn. However, Decision Trees are prone to overfitting, which can result in poor generalization performance. To overcome this problem, ensemble methods such as Random Forest and Gradient Boosting are often used to improve the performance of DTs [10].

5.1.2.1 Measures for Attribute Selection

If the dataset has N properties, determining which ones to put at the root or at different levels of the tree as internal nodes is a difficult process. Choosing any node at random to be the root will not fix the problem. If we take a random technique, we could get unsatisfactory outcomes with low precision. For solving this attribute selection problem, certain criteria are suggested.

- Entropy
- Information gain
- Gini index

Entropy: Entropy is a measure of the unpredictability of the data being processed. The higher the entropy, the more difficult it is to make any conclusions from the data. A coin flip is an example of an action that produces random information, mathematically for one attribute it is calculated by the equation 5.2.

$$E(s) = \sum_{i=1}^c -p_i \log_2 p_i \quad (5.2)$$

Where S is the current state, and p_i is the Probability of an event i of state S or the Percentage of class i in a node of state S .

Information Gain: Information gain (IG) is a statistical property that measures how well a given attribute separates the training examples according to their target classification mathematically it is calculated by the equation 5.3.

$$InformationGain = Entropy(before) - \sum_{j=1}^k Entropy(j, after) \quad (5.3)$$

Where *before* is the dataset before the split, K is the number of subsets generated by the split, and $(j, after)$ is subset j after the split.

Gini Index: The Gini index is a cost function that is used to evaluate dataset splits. It is calculated by subtracting the sum of each class's squared probabilities from one. It prefers larger partitions that are simple to implement, whereas information gain prefers smaller partitions that have different values, mathematically it is calculated by the equation 5.4.

$$Gini = 1 - \sum_{i=1}^c (p_i)^2 \quad (5.4)$$

Where p_i is the Probability of an event.

5.1.3 k-Neighbors Classifier

Neighbors-based classification is a type of instance-based learning or non-generalizing learning, it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point. k-Neighbors Classifier implements learning based on the k nearest neighbors of each query point, where k is an integer value specified by the user [11]. The k-NN classifier uses the Euclidean distance formula to calculate the distance between two points in a multi-dimensional space given by 5.5.

$$distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (5.5)$$

where x_1 and y_1 are the feature values of the first point in the multi-dimensional space and x_2 and y_2 are the feature values of the second point in the multi-dimensional space

5.1.4 Random Forest Classifier

Random Forest, also known as Random Decision Forest, is a powerful machine learning algorithm used for solving both classification and regression problems. The algorithm works by creating a set of decision trees from a randomly selected subset of the training set. Each decision tree is trained on a different subset of the data and the outputs of all the trees are combined to make a final prediction.

The RF algorithm is a type of ensemble learning method where multiple decision trees are combined to improve the overall performance of the classifier. It is basically a set of DT from a randomly selected subset of the training set, and then it collects the votes from different decision trees to decide the final prediction. The random selection of the training set helps to reduce overfitting and improve the generalization performance of the model.

One of the key advantages of the RF Classifier is its ability to handle large datasets with high-dimensional feature spaces. The algorithm is less prone to overfitting compared to single decision trees, which makes it more robust and reliable. Additionally, the RF Classifier can handle both categorical and numerical features and can deal with missing values in the data, which makes it a versatile algorithm for many applications.

The RF algorithm is widely used in many areas such as finance, healthcare, and marketing. It has been successfully applied to problems such as credit risk assessment, disease diagnosis, and customer segmentation. Moreover, the algorithm is easy to use and requires minimal data preprocessing, which makes it a popular choice for many machine learning applications.

In summary, RF is a powerful machine learning algorithm that uses a set of decision trees to make predictions. It is a versatile algorithm that can handle both categorical and numerical features, deal with missing values, and is less prone to overfitting compared to single decision trees. RF is widely used in many applications and is a popular choice for many machine learning tasks [12].

Chapter 6

Results

6.1 Model training evaluation parameters

The training evaluation parameters calculated for the online dataset for binary classification are shown in table 6.1.

Table 6.1: Training parameters for the online dataset for binary classification

| Sl. no | Algorithms used | Training accuracy | Precision | Recall | f1-score | Confusion matrix |
|--------|---------------------------|-------------------|--------------|--------------|--------------|---------------------------------|
| 1. | Decision Tree | 93% | 0.92 0.94 | 0.94 0.92 | 0.93 0.93 | [15144 897] [1244 14723] |
| 2. | K-neighbors Classifier | 93% | 0.90 0.96 | 0.96 0.90 | 0.93 0.93 | [15383 646] [1670 14309] |
| 3. | Random Forest | 99% | 0.98 0.99 | 0.99 0.98 | 0.98 0.98 | [15939 83] [202 15780] |
| 4. | Support vector classifier | 94.06% | 0.92 0.96 | 0.96 0.91 | 0.94 0.93 | [15402 577] [1321 14704] |

From table 6.1, it can be seen that almost all models perform well when tested against the training data on which the model was trained priorly. It is to be noted that the RF classifier is yielding the best results which could be due to the fact that RF models are one of the best machine learning models and easily grasp the underlying trend present in the data.

The training evaluation parameters calculated for the online dataset for multi-class classification are shown in table 6.2.

Table 6.2: Training parameters for the online dataset for multiclass classification

| Sl.no | Algorithms used | Training accuracy | Precision | Recall | f1-score | Confusion matrix |
|-------|---------------------------|-------------------|--|--|--|--|
| 1. | K-neighbors Classifier | 93% | 0.92 0.87 0.89 0.99 1.00 0.98 | 0.96 0.77 0.82 1.00 0.97 0.95 | 0.94 0.82 0.85 0.99 0.99 0.96 | [15333 320 266 0 0 0] [714 2510 24 0 0 0] [516 25 2603 2 0 0] [0 0 0 3218 0 0] [32 1 0 0 3162 50] [55 35 34 46 0 3013] |
| 2. | Random Forest | 98.87% | 0.98 0.97 0.98 0.99 1.0 0.99 | 0.99 0.98 0.97 0.99 0.99 0.98 | 0.98 0.97 0.97 0.99 0.99 0.98 | [15929 60 22 0 0 1] [144 3073 11 0 0 0] [62 3 3145 2 0 0] [0 0 0 3204 0 1] [2 1 0 0 3177 9] [35 5 0 3 0 3115] |
| 3. | Support vector classifier | 96% | 0.96 0.92 0.92 1.00 1.00 1.00 | 0.97 0.88 0.91 1.00 1.00 1.00 | 0.96 0.90 0.92 1.00 1.00 1.00 | [15629 226 230 0 0 0] [389 2779 6 0 0 0] [293 6 2883 0 0 0] [0 0 0 3186 0 0] [0 0 0 0 3210 8] [0 0 0 0 0 3167] |

From 6.2, it can be observed that all models perform well when tested against the training data on which the model was trained previously.

The training evaluation parameters calculated for the acquired real-time dataset for binary classification are shown in table 6.3.

Table 6.3: Training parameters for the real-time dataset for binary classification

| Sl. no | Algorithms used | Training accuracy | Precision | Recall | f1-score | Confusion matrix |
|--------|---------------------------|-------------------|--------------|--------------|--------------|------------------------------|
| 1. | Decision Tree | 91% | 0.90 0.93 | 0.93 0.89 | 0.91 0.91 | [6365 484] [735 6100] |
| 2. | K-neighbors Classifier | 93% | 0.90 0.96 | 0.96 0.89 | 0.93 0.92 | [6564 285] [722 6113] |
| 3. | Random Forest | 95.00% | 0.94 0.97 | 0.97 0.94 | 0.95 0.95 | [6622 227] [406 6429] |
| 4. | Support vector classifier | 92% | 0.90 0.93 | 0.94 0.90 | 0.92 0.92 | [6286 431] [660 6186] |

From table 6.3, it can be observed that almost all models perform well when tested against the acquired real-time training data on which the model was trained previously.

6.2 Model validation evaluation parameters

The model validation evaluation parameters calculated for the online dataset for binary classification are shown in table 6.4.

Table 6.4: Validation parameters for the online dataset for binary classification

| Sl. no | Algorithms used | Validation accuracy | Precision | Recall | f1-score | Confusion matrix |
|--------|---------------------------|---------------------|--------------|--------------|--------------|------------------------------|
| 1. | Decision Tree | 90% | 0.89 0.91 | 0.92 0.89 | 0.90 0.90 | [3629 335] [445 3593] |
| 2. | K-neighbors Classifier | 90% | 0.86 0.93 | 0.94 0.85 | 0.90 0.89 | [3726 250] [587 3439] |
| 3. | Random Forest | 94% | 0.93 0.95 | 0.95 0.93 | 0.94 0.94 | [3780 199] [275 3748] |
| 4. | Support vector classifier | 93% | 0.91 0.95 | 0.95 0.90 | 0.93 0.92 | [3825 197] [392 3588] |

Table 6.4. shows that the models yield good results when they are tested against the validation set which is sampled randomly from the training dataset. Among them once again Random forest classifier performs really well.

The model validation evaluation parameters calculated for the online dataset for multiclass classification are shown in table 6.5.

Table 6.5: Validation parameters for the online dataset for multiclass classification

| Sl.no | Algorithms used | Validation accuracy | Precision | Recall | f1-score | Confusion matrix |
|-------|---------------------------|---------------------|--|--|--|--|
| 1. | K-neighbors Classifier | 90% | 0.88 0.75 0.81 0.98 1.00 0.99 | 0.94 0.63 0.74 1.00 0.97 0.93 | 0.91 0.68 0.77 0.99 0.99 0.96 | [3832 138 112 0 0 0] [271 473 9 0 0 0] [198 11 600 0 0 1] [1 0 0 782 0 0] [7 1 0 3 735 10] [22 7 18 10 0 761] |
| 2. | Random Forest | 94% | 0.93 0.84 0.91 0.99 1.00 0.98 | 0.95 0.81 0.85 1.00 0.99 0.97 | 0.94 0.83 0.88 0.99 1.00 0.98 | [3804 113 66 0 0 6] [138 629 3 0 0 3] [111 5 671 2 0 0] [0 0 0 796 0 0] [0 0 0 0 804 8] [17 0 0 6 0 820] |
| 3. | Support vector classifier | 92% | 0.92 0.78 0.84 0.99 0.99 1.00 | 0.93 0.77 0.82 1.00 1.00 0.96 | 0.92 0.78 0.83 1.00 1.00 0.98 | [3682 156 115 0 0 1] [174 608 12 0 0 0] [135 7 646 0 0 0] [0 0 0 795 0 0] [0 0 0 0 850 1] [17 4 0 6 5 788] |

From table 6.5, it can be observed that the models yield good results when they classify multiple faults when they are tested against the validation set which is sampled randomly from the training dataset.

The model validation evaluation parameters calculated for the acquired real-time dataset for binary classification are shown in table 6.6.

Table 6.6: Validation parameters for the real-time dataset for binary classification

| Sl. no | Algorithms used | Validation accuracy | Precision | Recall | f1-score | Confusion matrix |
|--------|---------------------------|---------------------|--------------|--------------|--------------|------------------------------|
| 1. | Decision Tree | 85.00% | 0.83 0.88 | 0.88 0.82 | 0.86 0.85 | [1503 201] [304 1414] |
| 2. | K-neighbors Classifier | 89.00% | 0.86 0.93 | 0.93 0.85 | 0.89 0.89 | [1587 117] [261 1457] |
| 3. | Random Forest | 90.00% | 0.89 0.91 | 0.92 0.89 | 0.90 0.90 | [1561 143] [195 1523] |
| 4. | Support vector classifier | 92% | 0.89 0.93 | 0.93 0.89 | 0.91 0.91 | [1574 110] [196 1511] |

From table 6.6, it can be observed that almost all models perform well when tested against the acquired real-time validation dataset sampled randomly from the real-time training dataset.

6.3 Model testing evaluation parameters

The model testing evaluation parameters calculated for the online dataset for binary classification are shown in table 6.7.

Table 6.7: Testing parameters for the online dataset for binary classification

| Sl. no | Algorithms used | Testing accuracy | Precision | Recall | f1-score | Confusion matrix |
|--------|---------------------------|------------------|--------------|--------------|--------------|-----------------------------|
| 1. | Decision Tree | 58% | 1.00 0.55 | 0.16 1.00 | 0.27 0.71 | [8 43] [0 52] |
| 2. | K-neighbors Classifier | 78% | 0.43 0.99 | 0.96 0.75 | 0.59 0.85 | [49 2] [65 190] |
| 3. | Random Forest | 83% | 0.00 0.83 | 0.00 1.00 | 0.00 0.91 | [0 51] [0 255] |
| 4. | Support vector classifier | 92% | 0.67 1.00 | 1.00 0.90 | 0.80 0.95 | [51 0] [25 230] |

From table 6.7, it can be noted that the SVM algorithm yields good accuracy compared to the other models being tested, from the confusion matrix it can be seen that only 25 data values were misclassified this might be due to the fact that handles non-linear data very well due to its kernel parameter, that converts input data into a higher-dimensional feature space, also SVMs are generally more robust to outliers in the data and inherently have a regularization effect due to margin maximization. Regularization helps to prevent overfitting by penalizing complex models with small margins.

It is also clear from the parameters that the other models perform poorly compared to SVM this may be due to the overfitting of the models or might be the fact that those models have failed to understand the underlying patterns in the data even after tuning their hyperparameters empirically.

The model testing evaluation parameters calculated on the online dataset for multiclass classification are shown in table 6.8.

Table 6.8: Testing parameters for the online dataset for multiclass classification

| Sl.no | Algorithms used | Testing accuracy | Precision | Recall | f1-score | Confusion matrix |
|-------|---------------------------|------------------|--|--|--|---|
| 1. | K-neighbors Classifier | 83% | 0.51 0.64 1.00 1.00 1.00 1.00 | 0.96 0.18 0.96 1.00 1.00 0.86 | 0.66 0.28 0.98 1.00 1.00 0.93 | [49 2 0 0 0 0] [42 9 0 0 0 0] [2 0 49 0 0 0] [0 0 0 51 0 0] [0 0 0 0 51 0] [4 0 0 6 0 44] |
| 2. | Random Forest | 91% | 0.71 0.96 1.00 1.00 0.98 0.89 | 0.98 0.47 1.00 1.00 1.00 1.00 | 0.83 0.63 1.00 1.00 0.99 0.94 | [50 1 0 0 0 0] [20 24 0 0 1 6] [0 0 51 0 0 0] [0 0 0 51 0 0] [0 0 0 0 51 0] [0 0 0 0 0 51] |
| 3. | Support vector classifier | 75% | 0.55 1.00 1.00 1.00 0.78 0.57 | 0.80 0.75 0.24 1.00 1.00 0.73 | 0.65 0.85 0.38 1.00 0.88 0.64 | [41 0 0 0 0 10] [13 38 0 0 0 0] [21 0 12 0 0 18] [0 0 0 51 0 0] [0 0 0 0 51 0] [0 0 0 0 14 37] |

From table 6.8, it can be noted that the RF algorithm yields good accuracy, this might be due to the fact that RF is an ensemble method that combines multiple decision trees to form a model. Each decision tree in the RF is trained on a subset of the data with bootstrapping and a random subset of features is considered at each split. It calculates feature importance, which aids in determining the most important features for prediction. This is especially beneficial in non-linear datasets where feature interactions and non-linear connections may be important.

The kNN algorithm performs fairly well in the multiclass classification of the faults while other models yield poor results which may be due to overfitting of the models or may be due to randomness during training of the model.

The model testing evaluation parameters calculated on the acquired real-time dataset for binary classification are shown in table 6.9.

Table 6.9: Testing parameters for the real-time dataset for binary classification

| Sl.no | Algorithms used | Testing accuracy | Precision | Recall | F1_score | Confusion matrix |
|-------|---------------------------|------------------|--------------|--------------|--------------|----------------------|
| 1 | Decision Tree | 79.00% | 0.86 0.74 | 0.69 0.88 | 0.77 0.81 | [[18 8] [3 23]] |
| 2 | K-neighbors Classifier | 77.00% | 0.89 0.71 | 0.62 0.92 | 0.73 0.80 | [[16 10] [2 24]] |
| 3 | Random Forest | 50.00% | 0.50 0.50 | 0.04 0.96 | 0.07 0.66 | [[1 25] [1 25]] |
| 4 | Support vector classifier | 81.00% | 0.90 0.75 | 0.69 0.92 | 0.78 0.83 | [[18 8] [2 24]] |

From table 6.9, it can be noted that the SVM provides the best results during classification due to the fact that handles non-linear data very well due to its kernel parameter, which converts input data into a higher-dimensional feature space. During experiments, it was also found that ANN yielded a testing accuracy of 80 percent. Considering the fact that both models yield almost similar results and ANNs generally demand high computational cost, time, and resources SVM might be a better option in particular cases while considering the cost parameters and size of the dataset. It can be observed that some models don't perform well which might be due to the feature constraints present in the acquired real-time dataset.

Bibliography

- [1] F. M. L. Ribeiro, "Machinery fault database," http://www02.smt.ufrj.br/~offshore/mfs/page_01.html, accessed on 07 February 2023.
- [2] S. Altaf, M. W. Soomro, and M. S. Mehmood, "Fault diagnosis and detection in industrial motor network environment using knowledge-level modelling technique," *Modelling and Simulation in Engineering*, vol. 2017, 2017.
- [3] C.-Y. Lee, M.-S. Wen, G.-L. Zhuo, and T.-A. Le, "Application of ANN in induction-motor fault-detection system established with MRA and CFFS." *Mathematics*, vol. 10, 2022.
- [4] M. Hussain, D. K. Soother, I. H. Kalwar, T. D. Memon, Z. A. Memon, K. Nisar, and B. S. Chowdhry, "Stator winding fault detection and classification in three-phase induction motor." *Intelligent Automation & Soft Computing*, vol. 29, 2021.
- [5] T. Amanuel, A. Ghirmay, H. Ghebremeskel, R. Ghebrehiwet, and W. Bahlibi, "Comparative analysis of signal processing techniques for fault detection in three phase induction motor," *Journal of Electronics*, vol. 3, 2021.
- [6] Y. Bella, A. Oulmane, and M. Mostefai, "Industrial bearing fault detection using time-frequency analysis," *Engineering, Technology & Applied Science Research*, vol. 8, 2018.
- [7] P. D. Bharad and S. Subbaraman, "Induction motor bearing fault detection based on ICA and ANN," *International Journal of Computer Applications*, vol. 975, 2022.
- [8] A. L. Martinez-Herrera, E. R. Ferrucho-Alvarez, L. M. Ledesma-Carrillo, R. I. Mata-Chavez, M. Lopez-Ramirez, and E. Cabal-Yepez, "Multiple fault detection in induction motors through homogeneity and kurtosis computation," *Energies*, vol. 15, 2022.
- [9] "Support vector machine." <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>, accessed on 15 March 2023.
- [10] "Decision tree," <http://scikitlearn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>, accessed on 15 March 2023.
- [11] "Kneighbors classifier," <http://scikitlearn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>, accessed on 15 March 2023.

- [12] "Random forest," <http://scikitlearn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html>, accessed on 15 March 2023.