International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V73I1P123

Volume 73 Issue 1, 266-287, January 2025

Original Article

Performance Analysis of Machine Learning and Deep Learning Techniques in Diagnosing Imbalance Using Machine Fault Simulator-A Case Study

Vijayalakshmi K[1], Amuthakkannan Rajakannu[2], Ramachandran KP[3], Mohsina Kamarudden[4], Sri Rajkavin AV[5]

[1,4]*Department of Electrical and Communication Engineering, National University of Science and Technology, Oman.*
[2]*Department of Mechanical and Industrial Engineering, National University of Science and Technology, Oman.*
[3]*Deanship of Graduate Studies and Research, National University of Science and Technology, Oman.*
[5]*College of Design and Engineering, Department of Electrical and Computer Engineering, National University of Singapore, Singapore.*

[2]*Corresponding Author : amuthakkannan@nu.edu.om*

Received: 05 October 2024 Revised: 25 November 2024 Accepted: 10 January 2025 Published: 31 January 2025

Abstract *- The growth of Machine and Deep Learning in the manufacturing sector has been tremendous in the past decade, and it is widely used in many fields. Conditional monitoring of machines is a challenging task in manufacturing and process industries. It requires real-time monitoring to reduce downtime of machines, reduce cost and scraps, and improve the productive ML and DL, which have given promising results in the core domains of feature extraction and fault classification in machine fault detection. This paper addresses applying ML and DL techniques to predict the unbalancing of machines accurately and finding the proper techniques for predicting the unbalancing of rotating machines. This research uses an accelerometer, data acquisition card, and lab view software to collect vibration signals due to unbalancing. The output of the vibration data is collected in terms of frequency domain and time domain data. The ML techniques KNN, Support vector machine, Decision Tree, Random Forest, Naïve Bayes, logistic regression, and linear discriminant analysis are applied and predict the accurate prediction of unbalancing of machines. Similarly, DL techniques, MLP, CNN, RNN, and LSTM are used to identify the unbalancing of machines. After predicting the accuracy, precision, recall, and FN score of ML and DL, an extensive comparative analysis is done to identify the proper AI techniques in real-time condition monitoring; this research is executed by collecting data from the Spectra quest machine fault simulator. The result shows that ML techniques DT and RF give better results than other ML techniques. Similarly, MLP provides better results than CNN, RNN, and LSTM.*

Keywords *- Condition monitoring, Machine fault simulator, Machine Learning, Deep Learning, KNN.*

1. Introduction

Rotating machines are widely used across various industries to improve productivity and profit. Ensuring machines' smooth operation and safety in the manufacturing and process industries is a vital assessment factor to avoid downtime. Industrial centrifugal pumps, electric motors, generators, gear reducers, and shafts are essential rotating elements of modern manufacturing industries [1]. Therefore, continuous monitoring is crucial for effective maintenance management and management of production systems [2]. One effective method for condition monitoring is vibration analysis, which offers high accuracy [3]. Vibration analysis has an established theoretical foundation and considered to be a reliable measurement technique [4]. Shaft imbalance, a common issue associated with rotating machinery, significantly impacts operational efficiency and safety. It typically arises from uneven mass distribution along the shaft's axis, which can occur due to manufacturing defects, material inconsistencies, wear, or improper assembly. During rotation, this imbalance generates centrifugal forces that lead to vibrations damaging machinery components if unaddressed. Therefore, identifying and classifying shaft imbalance is essential for ensuring the safe operation of rotating machinery. AI has received significant attention in machine fault detection in industries. Several researchers have introduced a range of practical methodologies for diagnosing faults using AI. However, there are challenges in creating a real-time, precise fault diagnosis system that can effectively predict faults using shaft-imbalanced data in rotating machinery. Although many studies have applied either ML or DL approaches independently, little attention has been paid to comparing the performance of the two methodologies under a consistent and systematic framework. This indicates an important research gap since it is unclear which approach best

suits specific fault classes. Currently, the most commonly used methods for detecting failures in rotating machinery are of three main types: model-based [5], signal-processing-based [6], and data-driven [7]. The model-based approach is complex as it requires creating physical or mathematical models for complex mechanical systems [8]. On the other hand, signal processing methods need a lot of expert knowledge to design the right features and understand the signal properties [9]. Because of this, these two methods are not always easy to implement in real-world situations and lack consistency. In contrast, the data-driven approach can avoid these issues. It analyzes large datasets to find patterns and connections, allowing for effective machine detection or diagnosis [10].

A data-driven approach employs intelligent models to identify fault indicators from machinery's lifecycle data [11]. A study highlights that machine learning techniques are chosen as intelligent models for predictive maintenance [12]. Studies have demonstrated that k-means clustering, support vector machines, and Bayesian networks are applicable in fault diagnosis [13,14]. Deep learning is a popular data-driven method for fault diagnosis that performs fault detection without prior knowledge needed [15]. Nowadays, deep learning-based fault diagnosis research primarily focuses on four types of network models: Convolutional Neural Networks (CNN), Stacked Encoders (SAE), Recurrent Neural Networks (RNN), and Deep Belief Networks (DBN). However, there is a lot of confusion among researchers in the selection of the right ML or DL model for a particular machine condition problem. Some issues can be solved using ML techniques, and for some matters, ML may struggle to give accuracy. Similarly, DL will provide high accuracy in some cases and may not perform well in some classification of faults. So, vast research and analysis are needed to predict the suitable algorithm for a particular fault diagnosis.

This study compares machine learning and deep learning models for intelligent fault diagnosis of rotating machinery, contributing several insights to the field. It investigates how various ML and DL models perform across scaled datasets. This approach enables a comprehensive understanding of the models' scalability and robustness in real-world applications. It compares machine learning models and deep learning models across multiple fault types for intelligent fault diagnosis of rotating machinery and presents several contributions to the domain. It studies the generalization ability of different ML and DL models across scaled datasets. This allows the model's scalability and robustness to be picked up in the real-world use cases. Compared to previous studies that only compare the performance of ML and DL individually, this study contributes to the literature by further investigating and comparing the performance of ML and DL models on a shared dataset, tackling the uncertainty regarding the suitable model that performs best under specific fault conditions. Redefining industrial usage in terms of industrial checks based on the scalability and robustness of these models makes the study realistic and insightful.

1.1. Novelty in the Proposed Research

Since there is a lot of confusion in selecting the correct algorithm for a particular fault diagnosis in rotating machinery in manufacturing or process industries, this paper proposed a comparative study between various Machine Learning and Deep learning algorithms. The existing research works compare either machine learning algorithms or deep learning algorithms. No research compares ML and DL techniques for a fault in the rotating machinery. This work initially compares the accuracy of classification between machine learning techniques. In this regard, seven algorithms have been taken to predict unbalancing using the dataset collected from a machine fault simulator. Four deep learning techniques were used to indicate the faults using the data set, and then comparisons were made among the deep learning techniques. Then, a comparison analysis was conducted between machine learning and deep learning algorithms. Then, a comparison analysis was conducted between machine learning and deep learning algorithms.

Despite existing studies assessing deep learning and machine learning methods independently for rotating machine fault diagnosis. On the other hand, this study compares several ML and DL models on the same dataset generated from a machine fault simulator. This holistic overview fills an important research gap by offering insights notably into their relative strengths and weaknesses as well as suitability for specific industrial settings in relation to the classification of faults. The paper is organized as follows: Section 2 presents a review of relevant literature and a review of ML and DL techniques. Sections 3 and 4 detail the methodology used for fault diagnosis in rotating machinery, describing the machine fault simulator and experimental setup. Section 5 explains data collection and signal pre-processing. The application of ML and DL techniques are discussed in Sections 6 and 7, respectively. Sections 8 and 9 discuss the results of the comparative analysis of the ML and DL techniques, respectively, followed by a thorough evaluation of model performance in Sections 10. Finally, Section 11 concludes the paper by summarizing key findings, discussing limitations, and suggesting future research directions.

2. Literature Review

In the manufacturing sector, production machines are expected to reduce downtime as much as possible to increase productivity and increase the industry's economic growth [16]. Intelligent real-time monitoring systems reduce maintenance costs for machine tools and improve process reliability [17]. The use of AI in detecting faults in rotating machinery allows for the characterization of various health states of machines and improves the efficiency of production machines [18]. In the maintenance management of the production system, machine learning and deep learning play a

crucial role in predicting faults such as unbalancing, misalignment, tool wear, and looseness of machine components [19]. Tyler Ward et al. have comprehensively reviewed machine learning techniques for condition-based maintenance. The author concluded that ML has significant potential to bring more reliable, accurate, efficient, and optimized maintenance processes in condition-based maintenance (CBM) systems. There are many challenges in applying machine learning in CBM related to data quality, data availability, real-time processing, lack of skilled personnel, Industry regulations, security and privacy concerns, model interpretability, and ethical considerations [20]. This emphasizes the need for a well-built model to overcome these challenges and be reliable and straightforward to deploy in the industry.

The types of learning using ML include batch learning, online learning, instance-based learning, model-based learning, supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning, and transfer learning [21]. Mey O et al. have researched the unbalanced detection of a rotating shaft using vibration data and machine learning. The authors received 98.6% prediction accuracy on the evaluation dataset using the ML technique of a fully connected neural network with the input of an FFT-transformer vibration data set [22]. The current research used deep learning techniques that work on machine conditions in a well-organized way to direct researchers to do better research and provide discussion about future directions on condition-based maintenance [23]. The DL integrated with IoT technology and data science connected the physical-based models and data-driven machine condition monitoring [24].

Yahya Mohammed Al Nagger et al. have applied IoT to predict the maintenance of CNC machines. The results show that acceleration signals of time and frequency domain can identify the condition of CNC machines at different places through IoT for predictive maintenance. This method exhibits the possibility of integrating IoT technology and fault diagnosis; however, none of the ML and DL methods has been compared comprehensively, which needs to be addressed in vibration-based fault diagnosis. The researchers have not explored transitioning from traditional methods to machine learning and deep learning in vibration analysis [25]. The deep convolutional network for bearing fault detection under different operating conditions has given more accuracy than other deep learning techniques, which is clearly described by the research works carried out by Zhang, W. et al. [26]. Although the advantage of CNNs over other image classifiers has been proven under certain conditions, there is still no clear guidance on the best DL or ML method for different fault forms or service environments. Due to their nonlinear regression ability, deep learning techniques such as CNN, RNN, Deep encoders, and generative adversarial networks have been applied in rotating machine fault diagnosis [27]. Salim Lahmiri has compared statistical machine learning methods for condition monitoring of electric drive trains. This study uses large data with a Decision tree, Kernel Naïve Bayes, Radial basis function (Gaussian), support vector machine, linear discriminant analysis, K-NN algorithms, and Gaussian Naïve Bayes. As a result, decision trees take only a few seconds to learn and classify new instances from bug data for electric drive fault prediction [28]. Mohammed Javed Ali et al. have done machine learning-based faults in induction motors. This research used measured stator current and vibration signal to predict multi faults using machine learning techniques SVM, fine Gaussian, Fine KNN, weighted KNN, bagged trees, and subspace KNN have performed well in the selected 17 ML classifiers and other 12 ML classifiers do not perform well.

This indicated that not all classifiers would perform well in all kinds of condition monitoring problems. This work calls for systematically comparing various classifiers across different data sets to determine the most appropriate algorithm for a given use case. So, an extensive comparative study is needed among algorithms. Similarly, Pauline Org applied deep learning models CNN, SVM, Naïve Bayes, and Random Forest have been applied in the health monitoring of thermal features. The result of this paper compared the accuracy of the deep learning algorithm and concluded that CNN achieved more accuracy with 96.78%. However, there is a research gap in comparing ML and DL algorithms in predicting faults in machinery. This paper addresses the prediction accuracy in various ML algorithms in unbalancing data sets collected from spectra MFS and its comparison with various DL algorithm classification accuracy [30]. This gap systematically compares the classification accuracy of ML and DL using a data set collected from spectra MFS to fill this gap.

3. Spectra Quest Machinery Fault Simulator – An overview

The Spectra Quest Machinery Fault Simulator (SQMFS) is used to simulate the balanced and imbalanced conditions of the rotating shaft. The Machine Fault Simulator (MFS) is a versatile platform designed to study and simulate various mechanical faults in rotating machinery. The simulator monitors and analyzes fault conditions, providing insights into vibration analysis and fault diagnostics. [31]. Figure 1 shows the photograph of SQMFS with its essential components. The motor-driven shaft is the primary rotating element where faults can be introduced and studied. The motor's precise speed control allows one to observe the system's mechanical behaviour under various operating conditions. The MFS includes multiple adjustable disks and couplings mounted onto the shaft. These components are specifically designed with various tapped holes to accommodate the attachment of screws, weights, and other accessories. This flexibility is vital for simulating fault conditions, such as unbalance, misalignment, or looseness. For instance, introducing an imbalance involves asymmetrically adding weights to the disks, which leads to uneven mass distribution and the

characteristic vibration patterns of unbalanced systems. The shaft is supported by high-precision bearings that ensure smooth rotation and allow the introduction of bearing faults.

The MFS has various sensors, such as accelerometers, proximity probes, and load cells, to monitor vibrations, forces, and rotational speeds to capture the data required for fault analysis. The critical components of the setup include a motor-driven shaft, adjustable disks, a Lenze controller, and a data acquisition system to facilitate accurate data collection and analysis. Before each test, the system is set up to replicate the analyzed fault condition.

As the shaft rotates, the fault produces vibrations recorded by the connected accelerometer. In imbalance scenarios, uneven weight distribution creates centrifugal forces that lead to repetitive vibrations appearing as peaks in frequency spectrum analysis. The sensors deliver real-time data on these vibrations, enabling researchers to identify fault characteristics. The Lenze Controller delivers precise control over motor speeds and torque during the simulation, which is essential for accurately replicating various fault conditions.

4. Experimental Setup and Unbalancing Data Collection in Machine Fault Simulator

An accelerometer is used to predict the vibration signal in the experiments, which is done using a Machine Fault Simulator. An adjustable disk is used with tapped holes to attach weights to unbalance the MFS. A previously weighted screw-type weight is added to unbalance the MFS's shaft. Different weights have been added to have various unbalancing to give more learning to the ML/DL algorithms. The experimental setup is shown in Figure 2.

After adding weights to unbalancing, the motor is operated at various speeds using an accelerometer to get a vibration signal. The accelerometer was mounted on the rotary shaft, and the signal conditioning was done using the module NI 9234. NIcDAQ 9174 is used to collect the data, which the data transferred to the LabVIEW software. LabVIEW software displays the time domain and frequency domain vibration signals. These signals have been used as input to the ML/DL algorithm for classification. Figures 3 (a) and (b) show the Data acquisition and signal conditioning module used in the experimental work. By recording axial and radial vibration data, the accelerometer sheds light on how the shaft behaves dynamically in both balanced and unbalanced situations. Its vertical positioning increases sensitivity, which raises the accuracy of defect detection.

4.1. Data Collection and Signal Preprocessing

The experiment has been conducted in two conditions

- Balanced condition
- Unbalanced condition

Balanced condition is not having any extra weights and carefully inspecting the other faults before the data is analyzed. The time and frequency domain data were collected without adding weights or screws by uniformly distributing the shaft's mass across its entire length.

This reference data served as a critical reference point for all analyses, providing a standard against various imbalances that could be measured. To collect this minimum vibration dataset, an accelerometer is used to capture the frequency and time domain dataset.

This is the initial working of the data collection, which is mandatory to learn the algorithm in AI models. This balanced dataset was collected using three different aspects.

- No added weights or defects
- Uniform mass distribution
- Data collection as the reference dataset

The following work is the collection of vibration datasets for various unbalanced conditions. In this regard, multiple weights have been added to cause faults in the MFS with the shaft's unbalanced condition. This unbalanced condition will serve more vibration, captured using an accelerometer, NI-cDAQ card, and LabVIEW software. The data collection has been done in two critical aspects.

- Addition of weights
- Variable weight magnitude

To create an unbalanced dataset, 18 different weights were taken and placed randomly in either the rotary shaft's left or right disc. The added weight gave distinct vibration datasets, which is very useful as input to the classifier algorithms. Table 1 shows the sample weight used in the unbalanced dataset collection.

The accelerometer captures axial and radial vibration data, providing insights into the shaft's dynamic behavior under balanced and imbalanced conditions. Its vertical placement improves sensitivity, improving fault detection accuracy. During the data collection, the motor was operated at 30 HZ with a shaft frequency of 216 HZ. Once the motor stabilized at 30Hz, data collection was started. In the first data collection phase, data was recorded from a balanced system with no added weights, as shown in Figures 4 and 5.

In the second phase, screws of different weights were added to the rotary shaft disks to simulate various levels of imbalance, and the signals were collected, which are shown in Figures 6 and 7. The comprehensive data set was collected for frequency and time domain vibration data. The acceleration data frequency variation is set from 1 to 100 Hz in this experiment during balanced and imbalanced data collection.

Table 1. Sample weights used for the experiment

Sample No.	Weight (gm)	Sample No.	Weight(gm)	Sample No.	Weight (gm)	Sample No.	Weight (gm)
1	4.34	6	5.86	11	4.91	16	9.29
2	4.93	7	4.38	12	5.48	17	11.03
3	5.56	8	4.3	13	9.25	18	8.74
4	5.55	9	4.49	14	10.2		
5	4.93	10	4.36	15	10.05		

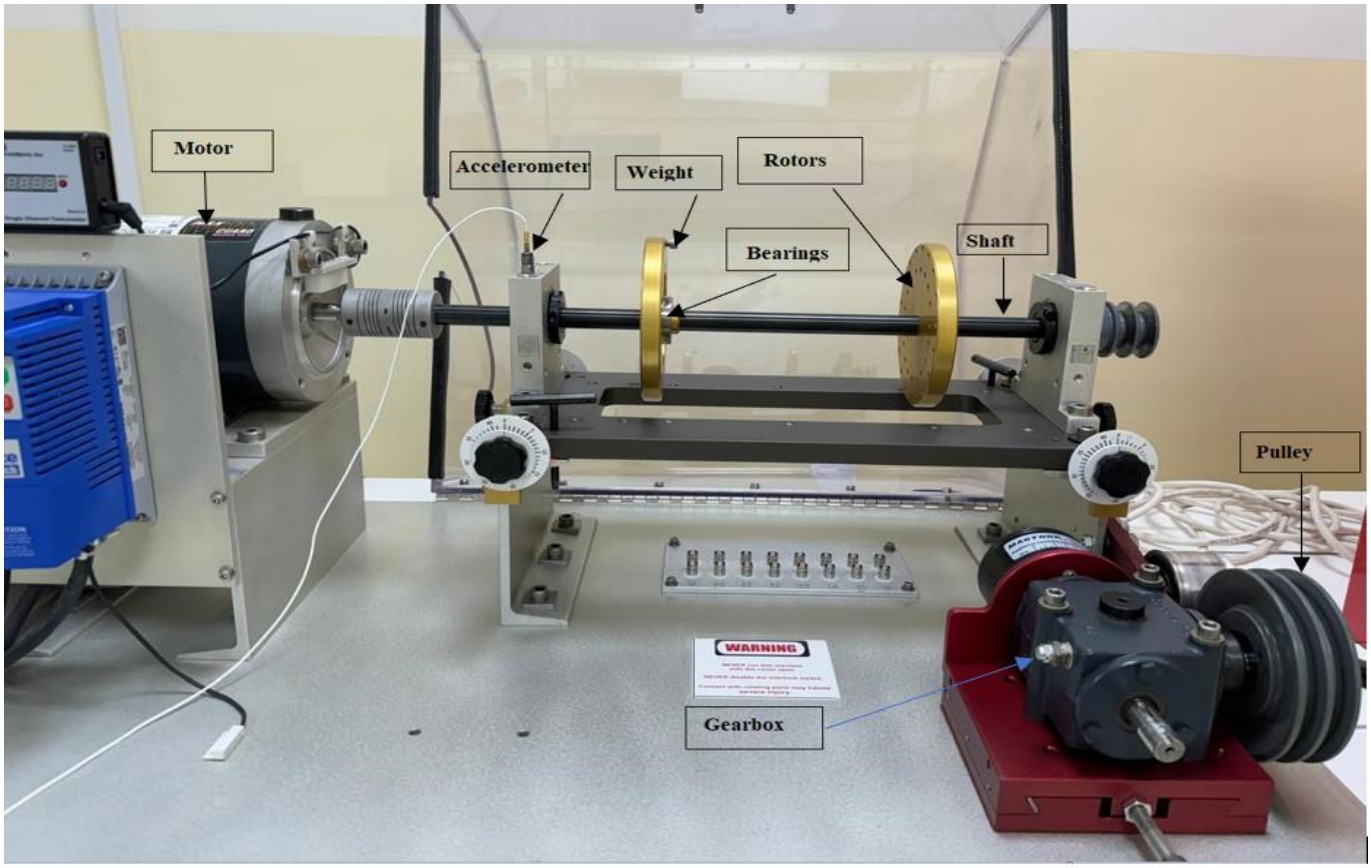


Fig. 1 Key components and structure of MFS

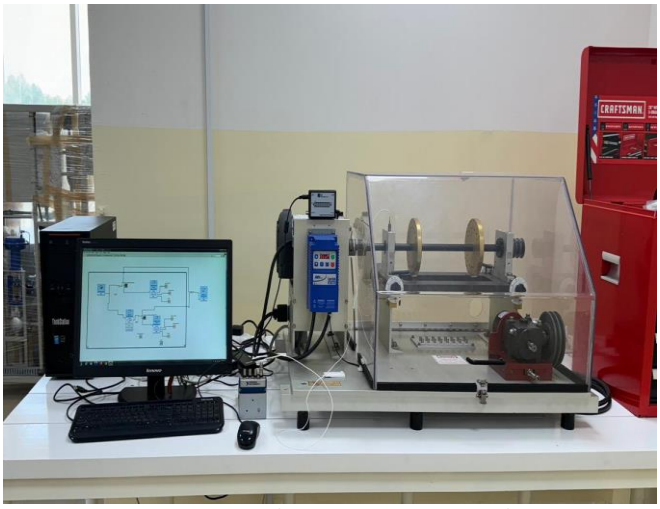
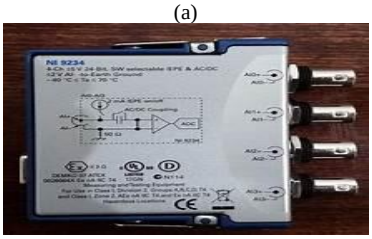


Fig. 2 Integration of MFS with LabVIEW software

(a)



(b)

Fig. 3 (a) Data acquisition components (NI Cdaq9174) (b)NI 9234 – Signal conditioning device

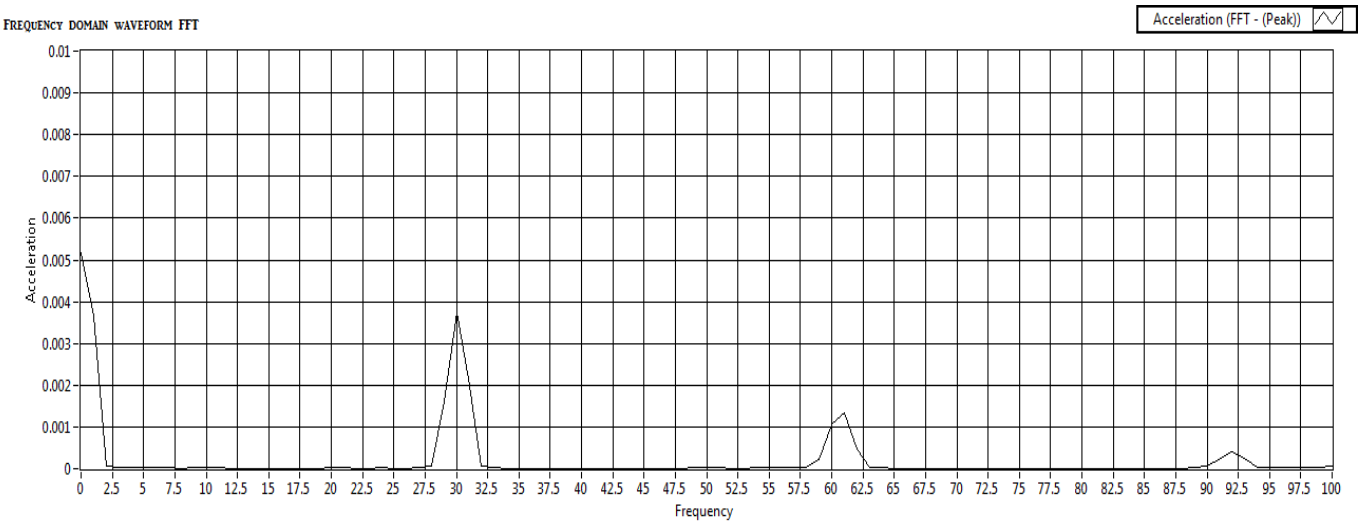


Fig. 4 Frequency domain chart for balanced data

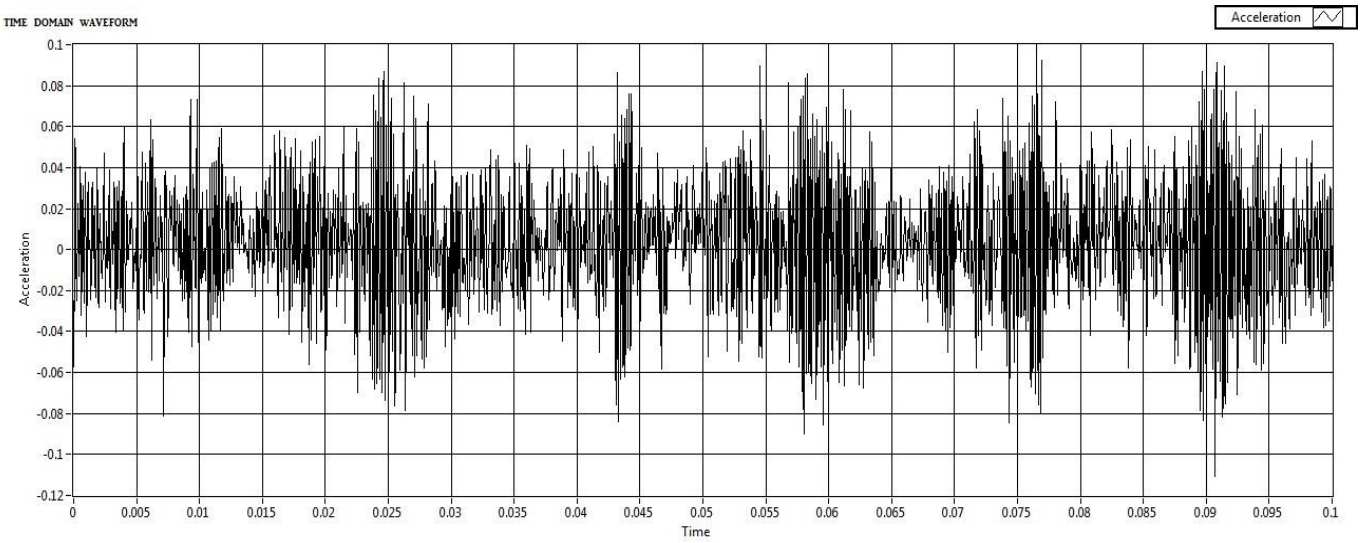


Fig. 5 Time domain chart for balanced data

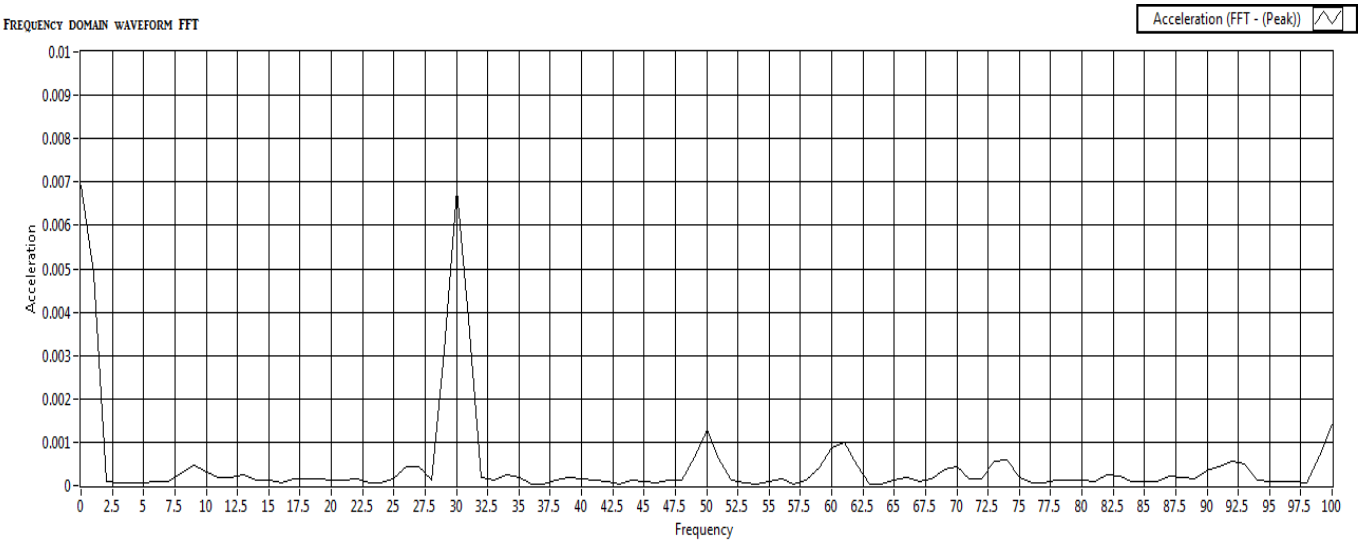


Fig. 6 Frequency domain chart for imbalanced data

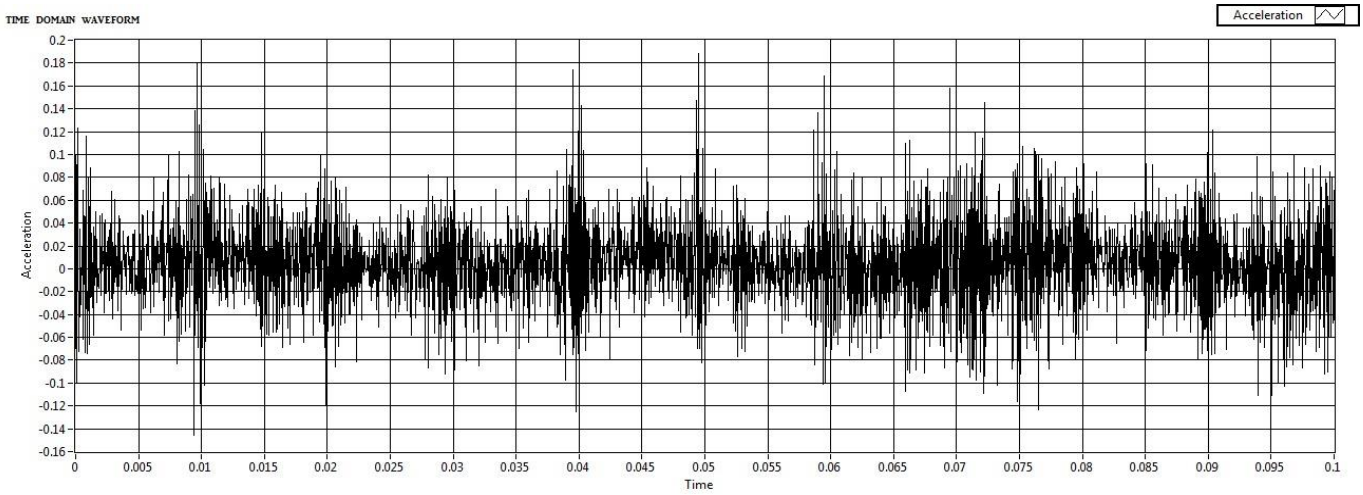


Fig. 7 Time domain chart for imbalanced data

5. Data Preprocessing

The collected data underwent several preprocessing steps to prepare it for input into the ML and DL models, ensuring that the models received data in a format conducive to effective learning, as shown in Table 2. Many research works include data pre-processing techniques such as Principal Component Analysis (PCA), discrete wavelet transform, wavelet packet decomposition, etc, for feature extraction before feature classification. In this work, it is not mandatory for several reasons, such as raw data's inherent characteristics being sufficient for ML and DL to learn and detect anomalies. Also, the modern and improved AI models will handle raw data without extensive pre-processing techniques. This may reduce the time needed to execute the real-time monitoring of fault diagnosis. Some of the strategies implemented before the application of AI algorithms. They are given as follows.

- Sampling strategy to balance classes to oversharing of balanced minority dataset
  A sampling strategy was executed to balance the dataset, as the proportions of the dataset were not balanced. This meant the minority class had to be oversampled for the deep learning models to receive a balanced dataset. This technique is crucial because it prevents the models from biasing towards the majority class and results in a model that can give a better generalization.

- Shuffling of the dataset to randomize the order of the data points
  The dataset is shuffled to randomize the order of the data points. This step prevents the models from learning any order-based biases, which could lead to overfitting and poor generalization of new data.

- Standardization using a standard scaler
  A standard scaler is a data pre-processing technique that standardizes the data by removing the mean and scaling it to unit variance to standardize the features. This step is significant for deep learning models as it allows all features to contribute evenly to learning; otherwise, functions with broad numerical ranges might build up such bias over the accuracy of the prediction.

- Reshaping of data into 2D and 3D arrays
  The data were reshaped to the input of different models of deep learning. For instance:

- MLP: data is converted to X, Y, and Z arrays, where the first two dimensions represent samples and features, respectively.
  CNN + LSTM: Data were reshaped into 3D arrays (the dimensions correspond to the number of samples, timesteps, and features). This reshaping is important for CNNs and LSTMs to learn spatial and temporal functions.

- Data augmentation and duplication to have a large set of data for training
  The data was replicated in two and four experiments to enlarge the dataset and make it more robust. This duplication was done to provide more examples and learn well because it can be generalized well in test data or unseen data. However, redundancy that can trigger overfitting, especially in models like LSTM, was relatively avoided in the augmentation process.

6. Machine Learning for Condition Monitoring

These models are selected based on their ability to work well on classification problems of any size and their tendency not to be too sensitive to data noise.

For example, Decision Trees (DT) and Random Forests (RF) are commonly recognized for their interpretability and capability to model complex nonlinear relationships essential in vibration data analysis. Finally, SVMs were added because they provide clear margins between classes, even for high-dimensional datasets.

Table 2. Final dataset after balancing the classes or types

	F1	F2	F3	F4	F5	…	F98	F99	F100	F101	Type
0	0.00474	0.00338	0.00004	0.00001	0.00002	…	0.00004	0.00000	0.00001	0.00005	Balanced
1	0.00732	0.00519	0.00018	0.00014	0.00014	…	0.00014	0.00017	0.00069	0.00138	Balanced
2	0.00524	0.00374	0.00008	0.00005	0.00003	…	0.00001	0.00001	0.00003	0.00005	Balanced
3	0.00714	0.00507	0.00005	0.00012	0.00012	…	0.00017	0.00008	0.00067	0.00154	Balanced
4	0.00517	0.00366	0.00006	0.00004	0.00003	…	0.00003	0.00002	0.00002	0.00004	Balanced
…	…	…	…	…	…	…	…	…	…	…	…
195	0.00509	0.00361	0.00004	0.00003	0.00002	…	0.00004	0.00001	0.00003	0.00005	Balanced
196	0.00718	0.00502	0.00007	0.00004	0.00008	…	0.00007	0.00014	0.00052	0.00130	Balanced
197	0.00738	0.00516	0.00017	0.00016	0.00009	…	0.00017	0.00017	0.00070	0.00152	Balanced
198	0.00809	0.00574	0.00018	0.00023	0.00028	…	0.00032	0.00022	0.00101	0.00192	Imbalanced
199	0.00711	0.00512	0.00021	0.00028	0.00025	…	0.00030	0.00015	0.00069	0.00142	Imbalanced

6.1. K-Nearest Neighbour

K-Nearest is a supervised ML technique that identifies k nearest to the given data point. The steps involved in k-nearest neighbours are as follows:

1. Calculating the distance between query and training points.
2. Selection of the k-nearest neighbours to the query point
3. Predicting class based on majority class (or) the mean value of neighbours

There are several distance measures as follows:
Euclidean distance

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$$

Manhattan distance

$$d(x,y) = \left(\sum_{i=1}^{n}|x_i - y_i|\right)$$

Minkowski distance

$$d(x,y) = \sum_{i=1}^{n}|x_i - y_i|^{1/p}$$

Hamming distance

$$d(x,y) = D_H = \sum_{i=1}^{k}|x_i - y_i| \quad \begin{matrix} x=y & D=0 \\ x\neq y & D\neq 1 \end{matrix}$$

6.2. Support Vector Machine

SVM is a supervised ML technique that can classify data using an optimal line. In SVM, the smallest distance between points and the decision boundary must be as large as possible. Three important points to have in SVM are

1. A point in a space
2. Defining decision boundary
3. Distance measurement

The two-dimensional linearly separable data can be separated by a line. The function of the line is $y = ax + b$. Here, it is considered as x with x_1 and y with x_2, and the equation is written as

$$ax_1 - x_2 + b = 0$$

If $x = (x_1, x_2)$ and $w = (a, -1)$, Then it can be written as,

$$w \cdot x + b = 0$$

This equation is derived from two-dimensional vectors. This Equation is the hyperplane equation. If hyperplane is defined, it can be used to make predictions. The hypothesis function h is defined as

$$h(x_i) = \begin{cases} +1 & \text{if } w \cdot x + b \geq 0 \\ -1 & \text{if } w \cdot x + b < 0 \end{cases}$$

The point above or on the hyperplane is classified as class +1, and the point below the hyperplane will be classified as class-1. So, the SVM aims to find a hyperplane that can accurately distinguish the data. Similarly, there are many such hyperplanes during the execution of a data set, and we need to find the best one to identify the optimal hyperplane.

6.3. Naïve Bayes

The Naïve Bayes classifier is a supervised learning algorithm in ML techniques that applies the Bayes theorem with the 'naïve' assumption of conditional assumptions between features.

Consider the given class variable y and dependent feature vector x_1 through x_n.

$$P(y\,|x_i \ldots x_n) = \frac{P(y)P(x_i \ldots x_n|y)}{P(x_1 \ldots x_n)}$$

Using the naïve independence assumptions that

$$P(x_i|y, x_1 \ldots, x_{i-1}, x_{i+1}, \ldots x) = P(x_i\,|y)$$

For i, this relationship is given as

$$P(y|x_1, \cdots x_n) = \frac{P(y)\pi_{i=1}^{n}P(x_i|y)}{P(x_1, \cdots x_n)}$$

Since P($x_1, \cdots x_n$) is constant for the given input, the classification rule is as follows.

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg \max_y P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

Maximum A Posteriori (MAP) estimation can be used in the next step to estimate $P(y)$ and $P(x_i \mid y)$; The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i \mid y)$

6.4. Logistic Regression (LR)

LR is a supervised ML algorithm used for feature classification. It predicts the probability that an instance belongs to a required class or not. LR is used for binary classification that takes input values 0 and 1.

The logistic function input greater than 0.5 belongs to class one, and others belong to 0. It is mentioned as regression because it is the extension of linear regression. The sigmoid function serves the purpose of mapping real-valued numbers to the range 0 and 1 to represent probabilities.

Sigmoid function $f(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$

If P(x) is an unbounded linear function of x, bound it between 0 and 1. Then

$$\log \frac{p(x)}{1-p(x)} = \alpha_0 + \alpha \cdot x$$

After solving for p(x) :

$$p(x) = \frac{e^{\alpha_0+\alpha}}{e^{\alpha_0+\alpha}+1}$$

A certain threshold, for example, 0.5, must be chosen to make the logistic regression a linear classifier. Now, the misclassification rate can be minimized if it is predicted as y = 1 when p ≥ 0.5 and y = 0 when p < 0.5.

Here, 1 and 0 are the classes. Since Logistic regression is used to predict probabilities, it can be fit using the likelihood function. Therefore, for each training data point x, the predicted class is y. The probability of y is either p if $y = 1$ or 1-p if $y = 0$. Now, the likelihood can be written as:

$$L(\alpha_0, \alpha) = \prod_{l=1}^{n} p(x_i)^{y_i}(1-p(x_i)^{1-y_i}$$

The multiplication can be transformed into a sum by taking the log:

$$l(\alpha_0, \alpha) = \sum_{i=0}^{n} y_i \log\, p(x_i) + (1-y_i) \log\, 1-p(x_i)$$

$$= \sum_{i=0}^{n} \log\, 1-p(x_i) + \sum_{i=0}^{n} y_i \log \frac{p(x_i)}{1-p(x_i)}$$

Further, after putting the value of p(x) :

$$l(\alpha_0, \alpha) = \sum_{i=0}^{n} -\log\, 1 + e^{\alpha_0+\alpha} + \sum_{i=0}^{n} y_i(\alpha_0 + \alpha \cdot x_i)$$

The next step is to take a maximum of the above likelihood function because, in the case of logistic regression, gradient ascent is implemented (opposite of gradient descent). In the estimation of the maximum likelihood function, MLE can be found by differentiating the above equation with respect to different parameters and setting it to zero. For example, the derivative with respect to one of the components of parameter alpha, i.e. a_j, is given by:

$$\frac{\partial l}{\partial \alpha_j} = \sum_{i=0}^{n} \big(y_i - p(x_i; \alpha_0, \alpha)\big) x_{ij}$$

6.5. Linear Discriminant Analysis (LDA)

LDA is a statistical method that finds the linear feature combination that separates two or more classes. The result of the class combination may be used as a linear classifier. LDA will differentiate between the classes by data to solve multi-class classification problems. The representation of the linear discriminant function for two classes is mathematically with the following.

$$\delta(x) = x * \Big(\sigma^2 * (\mu_0 - \mu_1) - 2 * \sigma^2 * (\mu_0^2 - \mu_1^2) + \ln\big(P(w_0)/P(w_1)\big)\Big)$$

Where:

$\delta(x)$ - linear discriminant function.

x - the input data point.

μ_0 and μ_1 - means of the two classes.

σ^2 - The common within-class variance.

P(ω_0) and P(ω_1) - the prior probabilities of the two classes.

6.6. Decision Tree

Decision trees are a non-parametric supervised learning method used for classification and regression for multi-class classification on a data set. This powerful tool is used in machine learning, data mining, and statistics. It is a flowchart-like structure that is used to make decisions as predictions. It consists of nodes, branches, and leaf nodes to represent decisions, the outcome of the decision, and final predictions. The decision tree works based on selecting the best attribute, splitting the dataset, and repeating the process.

Metrics for Splitting

- Gini Impurity: Measures the likelihood of an incorrect classification of a new instance if it was randomly classified according to the distribution of classes in the dataset.

$$\text{Gini} = 1 - \sum_{i=1}^{n} (p_i)^2$$

Where pi is the probability of an instance being classified into a particular class.

Entropy: Measures the amount of uncertainty in the data set or impurity in the dataset.

$$\text{Entropy} = -\sum_{i=1}^{n} p_i \log_2 (p_i),$$

Where pi is the probability of an instance being classified into a particular class.

Information Gain: Measures the reduction in entropy or Gini impurity after a dataset is split on an attribute.

$$\text{Information Gain} = \text{Entropy}_{\text{ parent}} \sum_{i=1}^{n} \left(\frac{|D_i|}{|D|} * \text{Entropy}(D_i) \right)$$

Where Di is the subset of D after splitting by an attribute.

6.7. Random Forest

RF, developed by Leo Breiman and Adele Cutter, combines the output of multiple decision trees to reach a single result. This is a powerful ML technique for classifying and finally predicting the set objectives.

It tackles both classification and regression problems effectively. This algorithm can handle complex data and mitigate overfitting, making it a powerful tool for predictive tasks in machine learning. RF can handle the data set containing continuous variables and categorical variables.

7. Application of Deep Learning in Condition Monitoring

Deep learning is a branch of machine learning that uses neural networks with multiple layers to automatically learn and recognize complex patterns in large amounts of data [32]. This project used deep learning to analyze frequency-domain data from a rotary shaft for condition monitoring.

Deep learning models allowed for automatically detecting imbalances and other faults in the machinery without manual feature engineering. Deep learning can handle complex and nonlinear data, allowing models to distinguish between balanced and imbalanced states. This shows how these techniques can be used to monitor conditions using frequency-based signals.

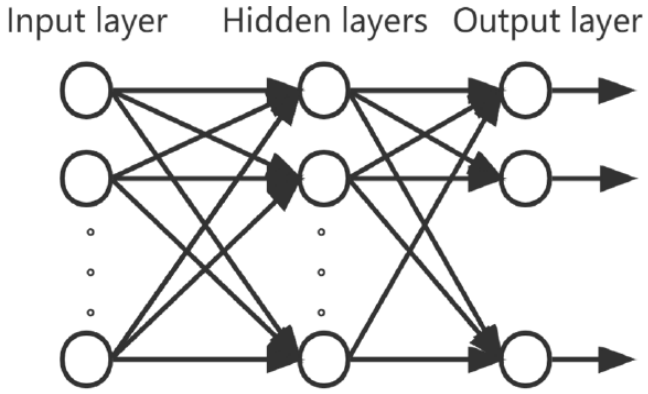


Fig. 8 Multilayer Perceptron [33]

7.1. Rationale for Algorithm Selection

The multilayer perceptron, convolution neural network, recurrent neural network, and long short-term memory models were selected based on their unique strengths in rotating machinery fault diagnosis. MLP is known for its robustness in structured data scenarios; CNN excels in spatial feature extraction, making it suitable for vibration analysis. RNN and LSTM are particularly effective for temporal sequences, essential for understanding dynamic operational states.

7.1.1. Multilayer Perceptron

An MLP is a type of feedforward artificial neural network made of at least three layers of nodes: an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, in one layer, is linked to all nodes in the next layer with an associated weight. During training, these weights are updated using backpropagation, a supervised learning method that reduces the difference between the predicted and actual outputs [33].

As shown in Figure 8, a connection between two nodes is assigned a weight value representing their relationship. A hierarchical connection also has a weight property, and the node function can perform summation and activation operations. The summation function is

$$S_j = \sum_{i=1}^{n} w_{i,j} I_i + \beta_j \quad (2)$$

Where n is the amount of input data, I_i is the input data, β_j Is the deviation and $w_{i,j}$ is the connection weight.

The output is obtained in the hidden layer using the activation function as

$$f_j(x) = \frac{1}{1+e^{-s_j}} \quad (3)$$

The output of the output layer cell in the MLP can be obtained by combining Equations (2) and (3)

$$y_i = f_j\left(\sum_{i=1}^{n} w_{i,j} I_i + \beta_j\right) \quad (4)$$

MLP is commonly applied in tasks such as pattern recognition, classification, and regression [34].

In this study, the MLP model is structured as follows:

- Input Layer: Accepts input with dimension X
- First Hidden Layer: 64 neurons with ReLU activation
- Second Hidden Layer: 32 neurons with ReLU activation
- Output Layer: 1 neuron with sigmoid activation

MLP is a feedforward neural network with two hidden layers. It can be described mathematically as:

- Layer 1: h1=ReLU(W1x+b1)
- Layer 2: h2=ReLU(W2h1+b2)
- Output: y=σ(W3h2+b3)

where

ReLU Activation: The hidden layers use the Rectified Linear Unit f(x)=max(0,x). ReLU helps mitigate the vanishing gradient problem and allows for faster training.

7.1.2. Convolutional Neural Network

CNN is a deep neural model that handles grid-like data like images. It uses convolutional layers to detect and filter the input data, identifying local patterns like edges and textures. These patterns are then combined and classified by fully connected layers. CNN has transformed computer vision tasks, achieving great success in image classification, object detection, and signal processing [35].

As shown in Figure 9, the convolution operation is at the heart of CNN, from which it derives its name.
The discrete convolution operation can be expressed as:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \quad (4)$$

Where
S (i, j) is the output of the convolution operation at position (i, j).
I is the input matrix.
K is the kernel (filter) matrix.
(i, j) are the coordinates of the output matrix.
(m, n) are the coordinates of the input matrix.

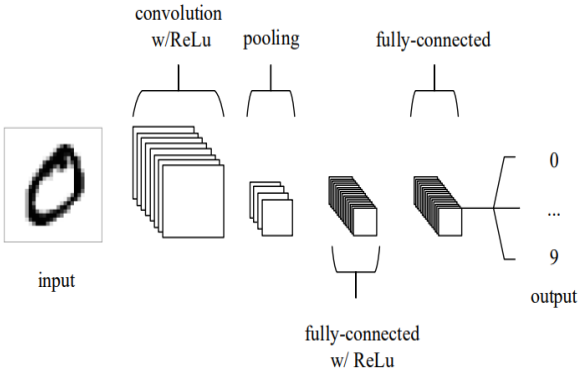


Fig. 9 Simple CNN architecture [36]

Equation 4 represents the convolution operation, where the kernel (K) is applied to the input (I) to produce the output (S). For each position (i, j) in the output matrix, the value is computed by summing the products of the overlapping elements of the input matrix and the kernel [37]. The architecture of a CNN typically comprises several specialized layers:

Convolutional layers apply filters to the input data, detecting local patterns. Activation layers introduce non-linearity. A common activation function is the Rectified Linear Unit (ReLU):

The mathematical equation of CNN pooling, classification, and weight update rule is given in equations 5.6.7 and 8

$$f(x) = max(0, x)(2) \quad (5)$$

Pooling layers reduce spatial dimensions. Max pooling, a common operation, can be expressed as:

$$y_{ij} = max((p,q) \in R_{ij})x_{pq} \quad (6)$$

R_i,j represents a local neighborhood around position (i,j)[38]. Fully connected layers perform final classification based on extracted features:

$$y = \sigma(\sum_i w_i P_i + b) \quad (7)$$

Here,σ is an activation function, w_i are weights, x_i Re inputs, and b is a biased term. The learning process in CNNs is facilitated by backpropagation, adjusting network parameters to minimize error. The weight update rule can be expressed as:

$$w_{ij} = w_{ij} - \eta \frac{\partial E}{\partial w_{ij}} \quad (8)$$

Where η is the learning rate, and E is the error function [35]. CNN is a major machine learning and AI breakthrough, especially in computer vision. Its design, based on convolution and nonlinear activations, makes it effective in processing grid-like data.

In this research work,
The CNN architecture is designed as follows:
Conv1D Layer: 64 filters, kernel size of 3, ReLU activation.
MaxPooling1D Layer: Pool size of 2.
Flatten Layer: Converts 2D feature maps to a 1D feature vector.
Dense Layer: 64 neurons with ReLU activation.
Output Layer: 1 neuron with sigmoid activation.
The CNN applies convolution operations followed by pooling:
Conv Layer: $a = ReLU(W * x + b)$, where * denotes convolution.

Pooling: $p = \max pool(a)$ reduces spatial dimensions and provides translation invariance.
Flatten: Converts the pooled features into a 1D vector f=flatten(p).
Dense layer: $d = ReLU(W1f + b1)$
Output: $y = \sigma(W2d + b2)$

- Convolutional Layer: Applies 64 filters of size 3 to extract local features from the input sequence.
- Pooling Strategy: Max pooling with a size of 2 reduces the spatial dimensions and helps achieve translation invariance.
- Filter Sizes: The kernel size of 3 captures local patterns in the input data, which can be essential in sequence analysis.

7.1.3. Recurrent Neural Network

Recurrent Neural Networks (RNNs) are a type of neural network designed to process sequences by keeping a memory of previous inputs. They are useful for tasks like Language translation, Speech recognition, Time series forecasting, Text generation, Sentiment analysis, etc.

In Figure 10, the values of θi, θh, and θo represent the parameters associated with the inputs, previous hidden layer states, and outputs Equations 9 and 10 define how an RNN evolves over time

$$Ot = f(ht;\theta) \quad (9)$$

$$ht = g(ht\text{-}1, xt;\theta) \quad (10)$$

Where Ot is the output of the RNN at time t, xt is the input to the RNN at time t, and *h*t is the state of the hidden layer(s) at time t. The image in Figure 12 shows a simple graphical model to illustrate the relation between these three variables in an RNN's computation graph.

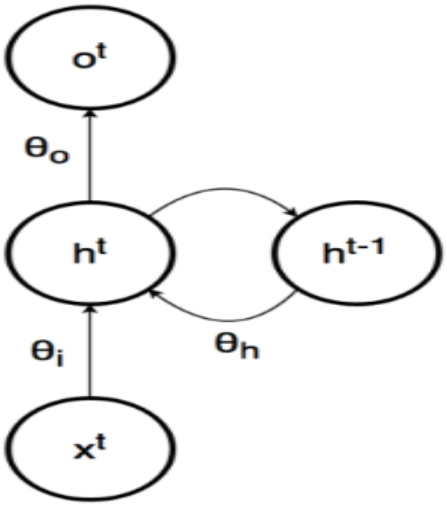


Fig. 10 RNN graphical model

Equation 10 shows how RNN remembers its past by allowing previous computations h^(t-1) to influence the present computations h^t. The aim of training the RNN is to make the sequence o^(t+τ) to match the sequence y_t, where τ represents the time lag (that y=0) between RNN's output o^(τ+1) and the first target output y_t. [39].

In the research work, The RNN model consists of:

- SimpleRNN Layer: 64 units with tanh activation.
- Output Layer: 1 neuron with sigmoid activation.

The SimpleRNN can be described as:

$$h_t = \tan h\,(W_{hh}h_{t-1} + W_{xh}x_t + b_h$$

$$y = \sigma(W_{h_y}h_t + b_y)$$

Where:

h_t is the hidden state at time t, and x_t is the input at time t.

- RNN Type: Simple RNN. This architecture is known for its limitations in capturing long-term dependencies due to vanishing gradient problems.
- Hidden Units: 64 recurrent units in the RNN layer.
- Activation: tanh is used, which is standard for RNNs because it maintains better stability across long sequences compared to ReLU.

7.1.4. Long Short-Term Memory

Long-Short-Term Memory (LSTM) units are a type of RNN designed to handle long-term dependencies in sequential data.

They use memory cells to store information over time and solve vanishing gradient problems in standard RNNs, making them better at learning long-term patterns [40]. LSTM has three gates: (i)Input Gate, (ii) Forget Gate and (iii) Output Gate

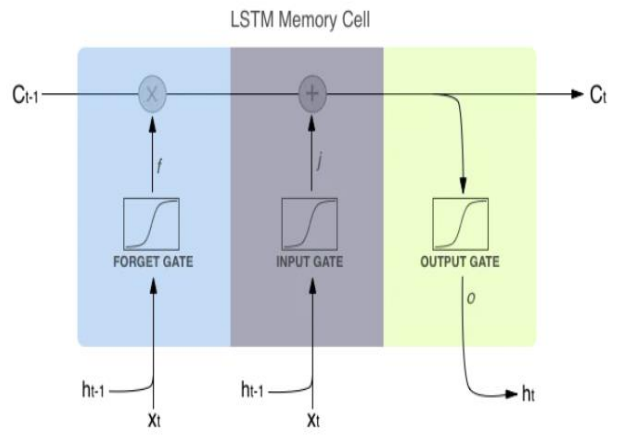
The gate comprises a sigmoid σNeural network layer. Equations 11, 12, and 13 are for the gates.

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i) \quad (11)$$

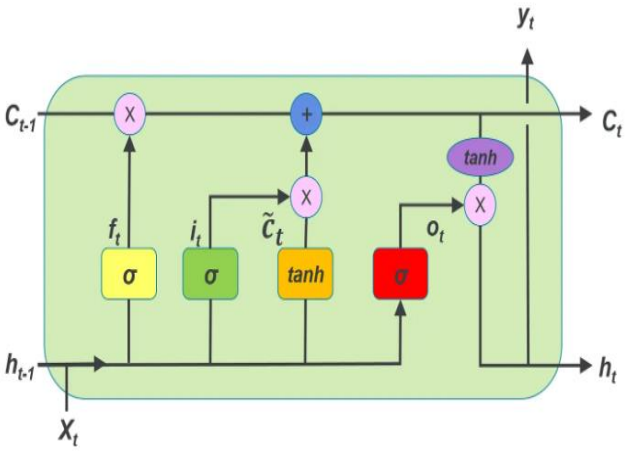
$$f_t = \sigma\big(w_f[h_{t-1}, x_t] + b_f\big) \quad (12)$$

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o) \quad (13)$$

$i_t \rightarrow$ represents the input gate.
$f_t \rightarrow$ represents the forget gate.
$o_t \rightarrow$ represents the output gate.
$\sigma \rightarrow$ represents a sigmoid function.
$w_x \rightarrow$ weight for the respective gate(x) neurons.
$h_{t-1} \rightarrow$output of the previous lstm block(at timestamp $t-1$).
$x_t \rightarrow$ input at current timestamp.
$b_x \rightarrow$ biases for the respective gates (x).



(a)



(b)

Fig. 11 (a) LSTM Memory cell (b) Block of LSTM at any timestamp {t} [41]

Equation 11 tells what new information will be stored in the cell state. Equation 12 is for the forget gate, which throws the information away from the cell state. Equation 13 is the output gate, which activates the final output of the LSTM block at timestamp 't', as shown in Figure 11. Equations 14, 15, and 16 represent the cell state, candidate cell state, and final output, respectively.

$$\tilde{c_t} = \tanh\left(w_c[h_{t-1}, x_t] + b_c\right) \quad (14)$$

$$c_t = f_t * c_{t-1} + i_t * \tilde{c_t} \quad (15)$$

$$h_t = o_t * \tanh\left(c^t\right) \quad (16)$$

$c_t \rightarrow$ cell state(memory) at timestamp (t).
$\tilde{c_t} \rightarrow$ represents a candidate for cell state at timestamp (t).
Note* others are the same as above.

LSTMs are useful for tasks with sequential data, like time series forecasting, natural language processing, and speech recognition. Their ability to retain information over time makes them perfect for tasks where the order and timing of events matter [42].

In the study, The LSTM architecture is structured as follows:

- LSTM Layer: 64 units with tanh activation.
- Output Layer: 1 neuron with sigmoid activation.
- LSTMs are designed to address the vanishing gradient problem using memory cells and gating mechanisms.
- 64 LSTM units are stored and updated in the hidden state based on the input and past hidden state.
- tanh is used, which is the standard activation for LSTM units. It's responsible for squashing values into the range [−1,1] [-1, 1] [−1,1] inside the LSTM cell and helps in handling long-term dependencies better.

8. Performance of ML Models

The performance of each model was evaluated using the following metrics:

- Accuracy: Indicates the overall correctness of the model's predictions.
- Precision: Reflects the model's ability to identify true positives among predicted positives.
- Recall: Measures the model's ability to detect all actual positives.
- F1 score: Provides a balance between precision and recall, especially important in imbalanced datasets.

As per the objective of this research work, ML algorithms of KNN, SVM, DT, RF, NB, LDA and LR were applied to predict the unbalancing in the shaft using the dataset collected from MFS.

As an outcome of the ML models, table 3 shows the various models' accuracy, precision, Recall, and FI scores. Figures 12 and 13 show the ROC curve of the ML algorithms and confusion matrix.

Table 3. ML algorithms ranked on performance

Model	Accuracy	Precision	Recall	F1-Score
K-Nearest Neighbors	0.997727	0.997733	0.997727	0.997715
Support Vector Machine	0.995455	0.995477	0.995455	0.995406
Random Forest	1.000000	1.000000	1.000000	1.000000
Decision Tree	1.000000	1.000000	1.000000	1.000000
Naïve Bayes	0.990909	0.991000	0.990909	0.990710
Linear Discriminant Analysis	0.986364	0.986567	0.986364	0.985903
Logistic Regression	0.990909	0.991000	0.990909	0.990710

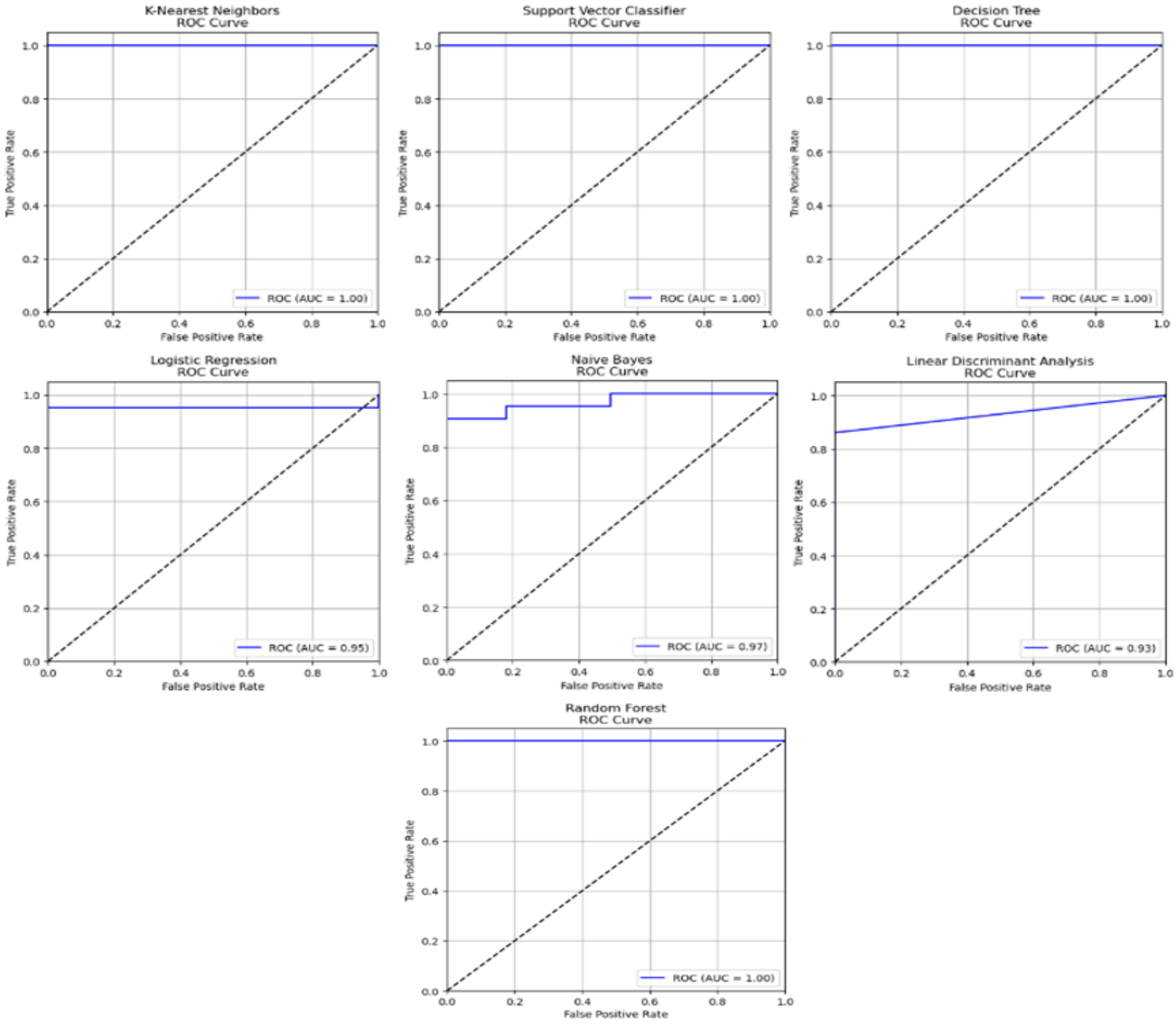


Fig. 12 ROC curve of ML Techniques

8.1. Perfect Performers

An interesting aspect of this research results is that five out of seven machine learning models achieved perfect scores, either 100% or near 100% accuracy across all metrics. This group encompasses a diverse range of algorithms:

- Linear models: Logistic Regression
- Tree-based models: Decision Tree, Random Forest
- Probabilistic model: Linear Discriminant Analysis

The common perfect performance of these models is notable and permits additional discussion. The features chosen help identify various conditions, as the conditions monitoring problem might have clear data patterns. Simple models like Logistic Regression perform to classify perfectly as the dataset is prepared well and balanced. However, this perfect performance raises doubts about the problem's complexity or the data's variety. In real-world monitoring, perfect classification is unlikely due to noise, sensor errors, and the complexity of mechanical systems [44]. Therefore, these results might indicate that the dataset represents a somewhat idealized scenario, possibly derived from a controlled experimental setup.

8.2. Near-Perfect Performers

K-Nearest Neighbors and Support Vector Classifier both achieved an accuracy of 0.99, with matching precision, recall, and F1-score. These models' slightly lower but still excellent performance might indicate minor difficulties in classifying a small subset of the data points. This could be due to some overlapping or closely spaced data points in the feature space. The high performance of these models, which use different approaches (instance-based for KNN and margin-based for SVC), suggests that the decision boundary for this problem is well-defined but may have some regions of uncertainty.

8.3. Good Performer

Naive Bayes achieved an accuracy of 0.990, with similar precision, recall, and F1-score. While still performing well, the slightly lower scores of Naive Bayes might be attributed to its feature independence assumption, which may not be true for this dataset. The good performance of Naive Bayes, despite its simplifying assumptions, indicates that the features in the dataset may be largely independent, which is an interesting characteristic of condition monitoring data.

8.4. Consistency Across Metrics

The accuracy, precision, recall, and F1-score are identical or very close for each model. This consistency suggests a balanced dataset with similar performance across different classes, indicating that the models are not biased toward any particular class. This balance is noteworthy in condition monitoring, where the class imbalance is often challenging (e.g., far fewer instances of failure than in a normal operation). It might not be representative of all real-world scenarios [43].

8.5. Robustness of Tree-based Methods

The flawless operation of tree-based techniques such as Random Forests and Decision Trees demonstrates how effectively these algorithms can identify intricate patterns in the condition monitoring work. These techniques seem useful in this situation because they are adept at managing nonlinear correlations and interactions between characteristics [43].

8.6. Effectiveness of Ensemble Methods

The Random Forest, an ensemble method, achieved perfect scores, showing how combining multiple models can improve prediction accuracy and reduce overfitting.

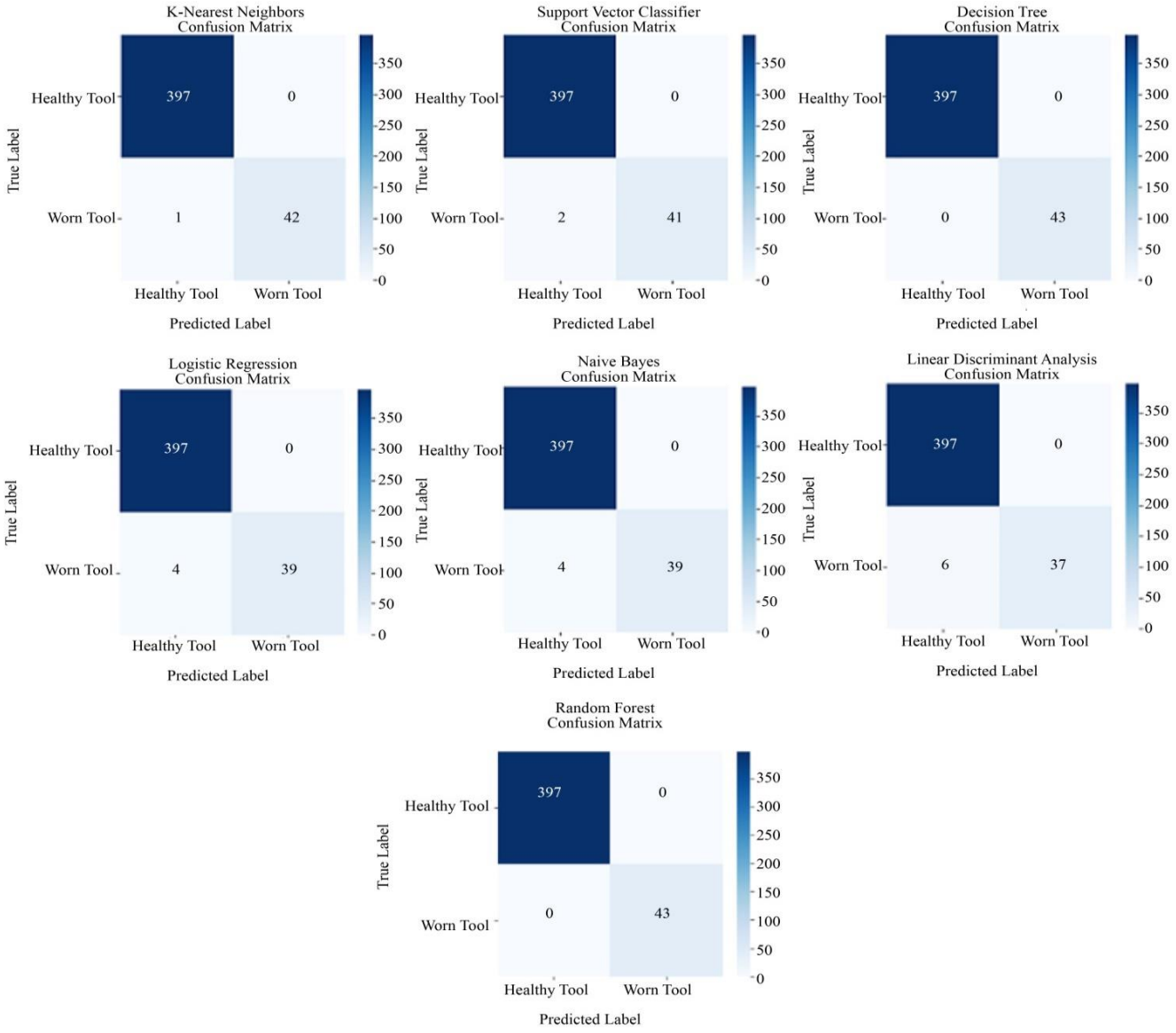


Fig. 13 Confusion matrix of ML techniques

This supports research suggesting ensemble methods often outperform single models in condition monitoring tasks by capturing complex patterns and reducing variance. This result highlights the value of using machine learning in condition monitoring. However, the near-perfect performance across many models also suggests that further investigation might be needed to ensure the dataset and problem formulation adequately represent the complexities of real-world condition monitoring scenarios.

9. Performance of Deep Learning Models

Table 4 presents the performance of deep learning models. This evaluation allows us to assess the effectiveness of various deep learning architectures in addressing condition monitoring tasks and compare their performance with traditional machine learning models. The performance analysis of DL models is shown in Figure 14.

9.1. Multi-Layer Perceptron (MLP)

The MLP achieved perfect scores across all metrics, with 100% accuracy, precision, recall, and F1-score. This suggests that a relatively simple neural network architecture is sufficient to capture the underlying patterns in the data for this condition-monitoring task. The MLP's success indicates that the problem might be linearly separable in a higher-dimensional space, which the MLP can effectively model.

9.2. Convolutional Neural Network (CNN)

CNN achieved 90.63% accuracy and an F1 Score of 0.89933. While it did not match the MLP's perfect score, CNN's performance is still respectable.

This suggests that there might be some spatial or temporal patterns in the data that CNN can capture. The slightly lower performance compared to the MLP may indicate that the added complexity of convolutional layers was not essential for this specific task.

9.3. Recurrent Neural Network (RNN)

The RNN performed moderately with 83.13% accuracy and an F1-score of 0.80292. While its precision remained high at 1.000, the recall dropped to 0.67073, indicating that the RNN struggled with identifying all relevant instances. This imbalance suggests that while the RNN identified some patterns effectively, it missed others, possibly due to the nature of the dataset, which may not heavily rely on sequential dependencies.

9.4. Long Short-Term Memory (LSTM)

The LSTM network exhibited the lowest performance among the deep learning models, with 57.50% accuracy and an F1-score of 0.59036. The precision (0.58333) and recall (0.59756) are relatively close, indicating consistent but poor performance across classes. The poor performance of the LSTM, designed to capture long-term dependencies, suggests that such temporal relationships might not be critical for this condition-monitoring task.

Table 4. DL algorithms ranked on performance

Model	Accuracy	Precision	Recall	F1-Score
MLP_2	1.000	1.000	1.000	1.000
CNN_2	0.906	1.000	0.817	0.899
RNN_2	0.831	1.000	0.670	0.802
LSTM_2	0.575	0.583	0.597	0.590

9.5. Performance Disparity

The stark difference in performance between the MLP and the other deep learning models (CNN, RNN, LSTM) is noteworthy. This disparity suggests that the complexity introduced by convolutional and recurrent architectures was unnecessary or even detrimental to this specific task. It is possible that these more complex models overfitted to the training data or struggled to extract relevant features from the given dataset.

10. Discussion and comparative Analysis of ML and DL

The study revealed a notable performance parity between several Machine Learning (ML) models and Deep Learning (DL) approaches in condition monitoring tasks. Specifically, traditional ML models such as Decision Trees and Random Forests achieved perfect scores across all metrics, matching the Multi-Layer Perceptron (MLP) performance from the deep learning category. This observation aligns with recent research by Zhang et al. [43], who found that well-tuned traditional ML models can often match or exceed the performance of more complex DL architectures in many condition monitoring scenarios. The performance of simple models over complex models contributes to various reasons. Firstly, the nature of the input data plays a crucial role. When features in condition monitoring tasks are well-structured and relatively linearly separable, simpler models can effectively capture the underlying patterns. This aligns with these research findings, suggesting that the complexity of the data did not necessitate the advanced feature extraction capabilities of deep learning architectures.

Dataset size is another critical factor influencing model performance. Lei et al. Cited show that neural AIs performed better than simpler machine learning models only when trained on large amounts of data (which is true for classical methods), with CNN and RNN being the most popular for image recognition. The findings imply that the existing data may not have been sufficient to exploit the capacity of more advanced models to the full extent. This highlights that data availability must also be considered when choosing which modeling approach to use for a condition-monitoring task [44]. The complexity of the problem at hand also influences model selection and performance. Zhao et al. highlighted that advanced DL architectures may not be necessary for relatively straightforward condition monitoring tasks. The findings support this notion, indicating that the condition-monitoring task in the study could be effectively addressed with simpler models.

Lastly, the risk of overfitting in complex models cannot be overlooked. Zhang et al. observed that more complex DL models (CNN, RNN, LSTM) might overfit the training data, leading to poor generalization and lower performance on test sets. This phenomenon could explain the study's lower performance of complex models, emphasizing the importance of model selection that balances complexity with generalization ability. The comparative analysis results reveal several important insights into applying ML and DL techniques in condition monitoring. One of the most striking findings is the effectiveness of simple models. The perfect performance achieved by several ML models and the MLP challenges the notion that more complex models are always better.

The study also revealed potential limitations of complex DL models in certain condition monitoring tasks. The lower performance of CNN, RNN, and LSTM models suggests that these architectures may not always be suitable, depending on the nature of the data and the specific monitoring requirements. This finding resonates with the work of Zhang et al., who found that the effectiveness of deep learning models in bearing fault diagnostics varies significantly based on factors such as noise levels and working loads. Their research emphasizes the need to carefully consider data characteristics and operational conditions when selecting modeling approaches. The critical nature of condition monitoring in industrial settings brings the issue of model interpretability to the forefront.

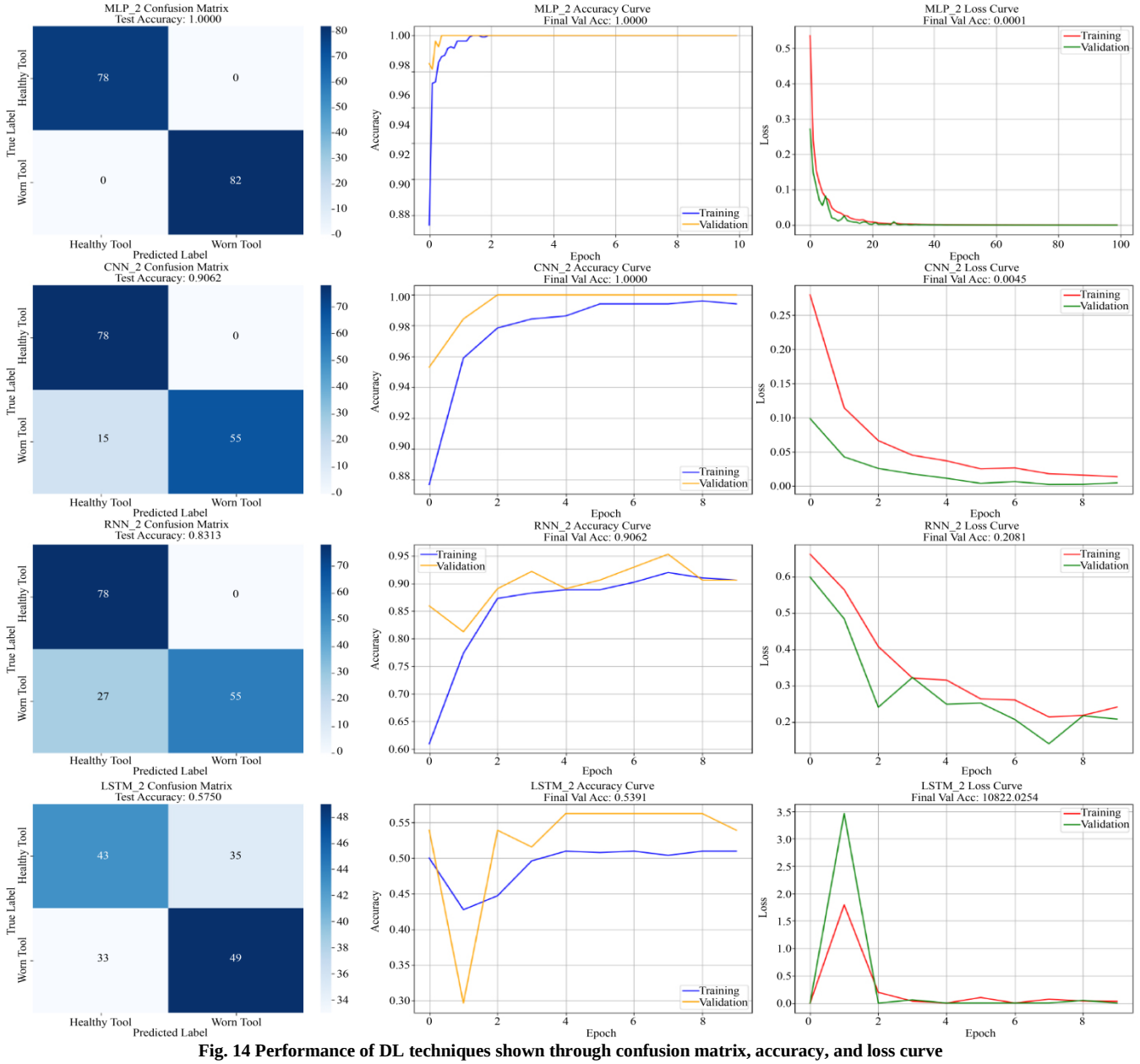


Fig. 14 Performance of DL techniques shown through confusion matrix, accuracy, and loss curve

The interpretability offered by many ML models, particularly decision trees, could be a significant advantage in real-world applications. Rudin successfully argued that there are real trade-offs between push button accuracy, which interpretable models give up, and that understanding and trusting a model's throughput is as important as performance or lack thereof. The results illustrate that the advantage of DL over ML is only demonstrable if the prediction accuracy matters more than the model explainability, especially in industrial settings [45].

The study's varying performance across models highlights the importance of understanding the characteristics of the available data. Data volume, feature complexity, and temporal dependencies influence model selection and performance considerably. This observation aligns with the review by Zhao et al., who noted that the effectiveness of different machine learning techniques in mechanical systems and signal processing highly depends on the nature of the available data and the specific requirements of the monitoring task.

Lastly, the study's high performance of simpler models suggests that efficient, real-time condition monitoring systems can be developed without computationally intensive DL models. This finding has important practical implications, particularly for resource-constrained environments. Wang et al. emphasized the importance of computational efficiency in smart manufacturing applications [46]. The comprehensive analysis of ML and DL approaches in condition monitoring has yielded valuable insights into various models' relative strengths and weaknesses. The key finding is that simple ML models and basic neural networks (MLP) often match or surpass the performance of more complex DL architectures, which aligns with the observations of Zhang et al. [43].

Their study on bearing fault diagnostics demonstrated that well-tuned traditional ML algorithms can achieve comparable or superior performance to deep learning models in many condition monitoring scenarios. The study shows that a model's effectiveness highly depends on the nature of the condition-monitoring task and the characteristics of the available data. This conclusion is supported by the work of Lei et al., who emphasized the importance of considering both data characteristics and task requirements when selecting modeling approaches for machine fault diagnosis [44]. The proposed methodologies reported in this study offer the potential for practical implementation in industrial engineering. If they are high-performing, ML and DL models address domain knowledge and can improve fault detection, substantially reducing unplanned downtime and maintenance. The study's outputs demonstrate how using these concepts can improve maintenance schedules, equipment life, and uptime. These results can be useful for researchers and industry professionals seeking innovative condition monitoring and maintenance techniques.

10.1. Practical Implications

The results of this study could transform the implementation of effective condition monitoring and fault diagnostics within an industrial environment. High accuracy is achieved despite using computationally efficient, interpretable, and low-requirement models such as Decision Trees or Random Forests. This shows the potential to develop simple, accurate, rigorous condition monitoring systems. This is especially useful for industries that work in an environment with limited resources, where real-time diagnostics are necessary, and deep learning of complex models cannot be performed. All these models have shown close-to-perfect results. They thus can be instantiated for monitoring purposes where fast and accurate fault detections are very important to prevent costly downtime or equipment failures.

Moreover, their simplicity and transparency make them easier to deploy in an existing system and interpret by engineers and operators, thus increasing trust and usability in mission-critical industrial environments. The strengths of the ML and DL approaches can be combined to improve accuracy and adaptability while maintaining practical efficiency in addressing real-time prediction concerns. Industries can use the insights from this research to design scalable, resilience-based condition monitoring systems customized to their operating specifications while ensuring cost-effectiveness.

10.2. Implications for Maintenance Practices

The study's findings highlight the potential of ML and DL models to transform industrial maintenance strategies by enabling predictive and proactive maintenance. Decision Trees, Random Forests, and MLP, higher performing models, can detect faults like imbalance with high accuracy and reduced downtime and maintenance costs. These models allow for precise scheduling of maintenance activities and optimizing equipment utilization. Future advancements could focus on hybrid models to address domain-specific challenges, further refining their practical applications.

10.3. Limitations

Although this study offers useful knowledge and perspective into the effectiveness of ML and DL approaches in recognizing imbalance using MFS, several limitations are observed.

10.3.1. Controlled Dataset

Using MFS, data is collected in a controlled experimental environment. Although this allows for a clear dataset, it potentially misses modelling the variances of the real industrial world.

10.3.2. Dataset Size and Diversity

It is possible that the favourable performance of many of the models is due to th fairly small and well-controlled nature of the dataset. However, larger and more diverse datasets are required to assess the model performance thoroughly.

10.3.3. Simple Representation of Features

Moreover, the features on which the model is trained may encode the problem in a way that enables easy gains for some algorithms to have excellent accuracy. This becomes the restriction on generalizing the results.

10.3.4. Model Overfitting Risks

Simple models such as decision trees, random forests, and logistic regression had perfect scores, which raises the question of whether they overfit the dataset. This implies that their performance will drop if applied to complex or unseen datasets.

10.3.5. Deep Learning Performance

Certain DL models' comparatively inferior performance (such as CNNs, RNNs, and LSTMs) may be due to the dataset's small size and relative lack of spatial or temporal dependencies. The observed results suggest that such a dataset may not be well-suited to fully exploit these architectures' potential.

10.3.6. Narrow Range of Real-World Validation

The results have not been validated against real-world datasets or operational settings, which may involve additional complexities, including varied mechanical loads, environmental factors, and mixed fault scenarios.

10.4. Future work

The accuracy of the models needs to be validated using real-world datasets in future work to assess the robustness of the methods across a wider range of industrial environments relative to noise and differing loads. Mixed ML and DL models result in improved performance by combining the interpretability of ML models with the feature extraction capabilities of DL techniques.

The need to improve dataset diversity, use transfer learning for data-scarce scenarios, and explore temporal dependencies to encode sequential data is identified. Also, creating lightweight models for real-time deployment and further investigating other types of faults would increase the models' usability. Finally, it provides further progress in explainable artificial intelligence techniques to make complex deep learning models more interpretable and applicable for mission-critical applications.

11. Conclusion and Future Recommendations

This research findings regarding simpler models' interpretability and computational efficiency underscore their attractiveness for many real-world condition monitoring applications. This aligns with the arguments presented by Rudin [45] and Wang et al. [46] who emphasized the importance of model interpretability and computational efficiency in industrial settings. Their work suggests that the practical benefits of simpler, more interpretable models often outweigh the potential performance gains of complex black-box models in many real-world scenarios. Looking to the future, several promising research directions emerge from the study:

- Investigating Model Performance: Assessing the performance of these models on a wider range of condition monitoring tasks and data types to understand their generalizability and limitations.

- Exploring Hybrid Approaches: Combining the strengths of ML and DL models to leverage the interpretability and efficiency of traditional ML techniques with the powerful feature extraction capabilities of DL approaches.

- Enhancing Interpretability: Developing techniques to make complex DL models more interpretable in the context of condition monitoring. Adadi and Berrada's research on explainable AI provides a solid foundation for this line of inquiry [47].

- Studying Data Pre-processing Techniques: Investigating the impact of different data preprocessing techniques on model performance to develop more robust and efficient condition monitoring systems.

- Exploring Transfer Learning and Domain Adaptation: Investigating the potential of transfer learning and domain adaptation in condition monitoring applications, as demonstrated by Shao et al. in their work on rolling bearing fault diagnosis, to develop effective models with limited task-specific data [48]. Ultimately, both ML and DL approaches have their advantages and disadvantages with respect to condition monitoring and deciding which approach to use should be driven by specific task sets, data availability, and end-user environment constraints. As the field continues to evolve, integrating the strengths of both approaches may lead to more robust and effective condition monitoring systems, ultimately enhancing the reliability and efficiency of industrial operations.

Author Contributions

Conceptualization, K.V., and R.A .; Literature collection, K.V., R.A., M.K and S.RK.A.V; Formal analysis, R.A, K.V. and M.K.; Funding acquisition, R.A., K.V. and KPR; Investigation, R.A., K.V and M.K; Methodology, R.A., KPR, K.V., M.K., and S.RK.A.V; Validation, R.A. and K.V.; Writing—original draft, R.A.; Review and editing, R.A., M.K., KPR, K.V. and S.RK.A.V. All authors have read and agreed to the published version of the manuscript

Funding

This work was performed as part of the research work titled" Deep learning based real-time drill condition monitoring using acoustic emission sensor", funded by MOHERI, OMAN, grant number BFP/RGP/EI/22/433

Institutional Review Board Statement

This study was conducted according to the National University of Science and Technology, Oman guidelines.

Data Availability Statement

The original contributions presented in the study are included in the article/supplementary material; further inquiries can be directed to the corresponding author/s.

References

[1] Issam Attoui, and Amar Omeiri, "Fault Diagnosis of an Induction Generator in a Wind Energy Conversion System Using Signal Processing Techniques," *Electric Power Components and Systems*, vol. 43, no. 20, pp. 2262-2275, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[2] Long Wen et al., "A New Convolutional Neural Network-Based Data-Driven Fault Diagnosis Method," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 7, pp. 5990-5998, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[3] M. Claeys et al., "Modal Interactions Due to Friction in the Nonlinear Vibration Response of the "Harmony" Test Structure: Experiments and Simulations," *Journal of Sound and Vibration*, vol. 376, pp. 131-148, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[4] Jaouher Ben Ali et al., "Online Automatic Diagnosis of Wind Turbine Bearings Progressive Degradations under Real Experimental Conditions Based on Unsupervised Machine Learning," *Applied Acoustics*, vol. 132, pp. 167-181, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[5] S. Tyagi, and S.K. Panigrahi, "Transient Analysis of Ball Bearing Fault Simulation Using Finite Element Method," *Journal of The Institution of Engineers (India): Series C*, vol. 95, pp. 309-318, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[6] P.K. Kankar, Satish C. Sharma, and S.P. Harsha, "Fault Diagnosis of Ball Bearings Using Continuous Wavelet Transform," *Applied Soft Computing*, vol. 11, no. 2, pp. 2300-2312, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[7] Liangwei Zhang et al., "End-to-End Unsupervised Fault Detection Using a Flow-Based Model," *Reliability Engineering & System Safety*, vol. 215, pp. 1-14, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[8] Long Di, and Zongli Lin, "Control of a Flexible Rotor Active Magnetic Bearing Test Rig: A Characteristic Model Based All-Coefficient Adaptive Control Approach," *Control Theory and Technology*, vol. 12, pp. 1-12, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[9] Qiang Zhou et al., "A Hybrid Fault Diagnosis Method for Mechanical Components Based on Ontology and Signal Analysis," *Journal of Intelligent Manufacturing*, vol. 30, pp. 1693-1715, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[10] Mengshi Li et al., "A Data-Driven Residual-Based Method for Fault Diagnosis and Isolation in Wind Turbines," *IEEE Transactions on Sustainable Energy*, vol. 10, no. 2, pp. 895-904, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[11] Weiting Zhang, Dong Yang, and Hongchao Wang, "Data-Driven Methods for Predictive Maintenance of Industrial Equipment: A Survey," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2213-2227, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[12] Rong-Heng Lin et al., "Review on Hydrogen Fuel Cell Condition Monitoring and Prediction Methods," *International Journal of Hydrogen Energy*, vol. 44, no. 11, pp. 5488-5498, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[13] Soukaina Mjahed et al., "Improved PSO Based K-Means Clustering Applied to Fault Signals Diagnosis," *2018 International Conference on Control, Automation and Diagnosis (ICCAD)*, Marrakech, Morocco, pp. 1-6, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[14] Junjie Wang et al., "Fault Diagnosis Method of Photovoltaic Array Based on Support Vector Machine," *Energy Sources, Part A: Recovery, Utilization, and Environmental Effects*, vol. 45, no. 2, pp. 5380-5395, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[15] Mariela Cerrada et al., "A Review on Data-Driven Fault Severity Assessment in Rolling Bearings," *Mechanical Systems and Signal Processing*, vol. 99, pp. 169-196, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[16] Onur Surucu, Stephen Andrew Gadsden, and John Yawney, "Condition Monitoring using Machine Learning: A Review of Theory, Applications, and Recent Advances," *Expert Systems with Applications, Expert Systems with Applications*, vol. 221, pp. 1-21, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[17] Anna Lena Demmerling, and Dirk Soffker, *Condition Monitoring: A Review on Real-Time Measurement Techniques for Tool Condition Monitoring and Analysis of Metalworking Fluids*, University Library of RWTH Aachen, pp. 191-208, 2016. [Google Scholar] [Publisher Link]

[18] Josef Koutsoupakis, Dimitrios Giagopoulos, and Iraklis Chatziparasidis, "AI-Based Condition Monitoring on Mechanical Systems Using Multibody Dynamics Models," *Engineering Applications of Artificial Intelligence*, vol. 123, no. 3, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[19] Purushottam Gangsar, Aditya Raj Bajpei, and Rajkumar Porwal, "A Review on Deep Learning-Based Condition Monitoring and Fault Diagnosis of Rotating Machinery," *Noise & Vibration Worldwide*, vol. 53, no. 11, pp. 550-578, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[20] Tyler Ward et al., "A Comprehensive Review of Machine Learning Techniques for Condition-Based Maintenance," *International Journal of Prognostics and Health Management*, vol. 15, no. 2, pp. 1-20, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[21] Hosameldin Ahmed, and Asoke K. Nandi, "Vibration-Based Condition Monitoring Using Machine Learning," *Condition Monitoring with Vibration Signals: Compressive Sampling and Learning Algorithms for Rotating Machines*, pp. 115-130, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[22] Oliver Mey et al., "Machine Learning-Based Unbalance Detection of a Rotating Shaft Using Vibration Data," *25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Vienna, Austria, pp. 1610-1617, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[23] Rui Zhao et al., "Deep Learning and Its Applications to Machine Health Monitoring," *Mechanical Systems and Signal Processing*, vol. 115, pp. 213-237, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[24] Yahya Mohammed Al-Naggar et al., "Condition Monitoring Based on IoT for Predictive Maintenance of CNC Machines," *Procedia CIRP*, vol. 102, pp. 314-318, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[25] Onur Avci et al., "A Review of Vibration-Based Damage Detection in Civil Structures: From Traditional Methods to Machine Learning and Deep Learning Applications," *Mechanical Systems and Signal Processing*, vol. 147, pp. 1-45, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[26] Wei Zhang et al., "A Deep Convolutional Neural Network with New Training Methods for Bearing Fault Diagnosis under Noisy Environment and Different Working Load," *Mechanical Systems and Signal Processing*, vol. 100, pp. 439-453, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[27] Funa Zhou et al., "Deep Learning Fault Diagnosis Method Based on Global Optimization GAN for Unbalanced Data," *Knowledge-Based Systems*, vol. 187, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[28] Salim Lahmiri, "A Comparative Study of Statistical Machine Learning Methods for Condition Monitoring of Electric Drive Trains in Supply Chains," *Supply Chain Analytics*, vol. 2, pp. 1-7, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[29] Mohammad Zawad Ali et al., "Machine Learning-Based Fault Diagnosis for Single- and Multi-Faults in Induction Motors Using Measured Stator Currents and Vibration Signals," *IEEE Transactions on Industry Applications*, vol. 55, no. 3, pp. 2378-2391, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[30] Pauline Ong et al., "A Deep Learning Approach for Health Monitoring in Rotating Machineries Using Vibrations and Thermal Features," *Decision Analytics Journal*, vol. 10, pp. 1-9, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[31] Machinery Fault Simulator, SpectraQuest, Inc. [Online]. Available: https://spectraquest.com/machinery-fault-simulator/details/mfs/

[32] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep Learning," *Nature*, vol. 521, pp. 436-444, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[33] Chaochun Zhong et al., "Improved MLP Energy Meter Fault Diagnosis Method Based on DBN," *Electronics*, vol. 12, no. 4, pp. 1-12, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[34] Kurt Hornik, Maxwell Stinchcombe, and Halbert White, "Multilayer Feed Forward Networks are Universal Approximators," *Neural Networks*, vol. 2, no. 5, pp. 359-366, 1989. [CrossRef] [Google Scholar] [Publisher Link]

[35] Kaiming He et al., "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, pp.770-778, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[36] Keiron O'Shea, and Ryan Nash, "An Introduction to Convolutional Neural Networks," *arXiv preprint*, pp. 1-11, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[37] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*, MIT Press, pp. 351-354, 2016. [Google Scholar] [Publisher Link]

[38] Kaiming He et al., "Delving Deep into Rectifiers: Surpassing Human-Level Performance on Image Net Classification," *Proceedings of the IEEE International Conference on Computer Vision*, Santiago, Chile, pp. 1026-1034, 2015. [Google Scholar] [Publisher Link]

[39] John McGonagle, Christopher Williams, and Jimin Khim, Recurrent Neural Network, Brilliant, 2020. [Online]. Available: https://brilliant.org/wiki/recurrent-neural-network/

[40] Sepp Hochreiter, and Jurgen Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997. [CrossRef] [Google Scholar] [Publisher Link]

[41] Divyanshu Thakur, LSTM and its Equations, Medium, 2018. [Online]. Available: https://medium.com/@divyanshu132/lstm-and-its-equations-5ee9246d04af

[42] Alex Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*, Springer Berlin Heidelberg, pp. 5-13, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[43] Shen Zhang et al., "Deep Learning Algorithms for Bearing Fault Diagnostics-A Comprehensive Review," *IEEE Access*, vol. 8, pp. 29857-29881, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[44] Yaguo Lei et al., "Applications of Machine Learning to Machine Fault Diagnosis: A Review and Roadmap," *Mechanical Systems and Signal Processing*, vol. 138, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[45] Cynthia Rudin, "Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead," *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206-215, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[46] Jinjiang Wang et al., "Deep Learning for Smart Manufacturing: Methods and Applications," *Journal of Manufacturing Systems*, vol. 48, no. 3, pp. 144-156, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[47] Amina Adadi, and Mohammed Berrada, "Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)," *IEEE Access*, vol. 6, pp. 52138-52160, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[48] Haidong Shao et al., "Rolling Bearing Fault Feature Learning Using Improved Convolutional Deep Belief Network with Compressed Sensing," *Mechanical Systems and Signal Processing*, vol. 100, pp. 743-765, 2018. [CrossRef] [Google Scholar] [Publisher Link]