International Journal of Scientific Research and Engineering Development- Volume 7 Issue 3, May-June 2024

Available at www.ijsred.com

RESEARCH ARTICLE

OPEN ACCESS

Fault Detection in Gears Using Machine Learning

¹Deepraj Mulik, ²Tanmay Biradar, ³Ayush Khodke, ⁴Mrs. Ami Barrot, ⁵Dr. P.G. Kulkarni, ⁶Mrs. Rashmi Pakankar

Abstract:

Gears are essential components in various machinery, including automobiles and rotating equipment. When gears fail, it can lead to significant losses. This paper discusses a method for detecting and analysing gear faults to mitigate such risks. To achieve this, a specialized test setup is constructed to examine gear faults experimentally. This setup collects vibration data from both healthy gears and those deliberately made faulty. The collected data is then analysed using the Fast Fourier Transform (FFT) technique, implemented through LabVIEW software. This analysis provides insights into the time and frequency domains of the vibration signals, aiding in identifying fault patterns. Subsequently, an Artificial Neural Network (ANN) is trained using MATLAB with the processed vibration data. This trained ANN forms a system capable of detecting faults in gears based on the patterns learned from the experimental data. In summary, this research presents a comprehensive approach to gear fault detection, utilizing experimental studies, signal processing techniques, and machine learning algorithms.

Keywords —Gear faults, Machinery, Vibration analysis, Fast Fourier Transform (FFT), LabVIEW, MATLAB, Artificial Neural Network (ANN), Fault detection.

1. INTRODUCTION

The gearbox is the primary tool of power transmission. Gears are one of the most significant parts in mechanical transmission systems. It is critical element in a variety of industrial applications, transportation. The gearbox is the primary tool of power transmission. Gears are one of the most significant parts in mechanical transmission systems. It is critical element in a variety of industrial applications, transportation, aerospace, energy, agricultural sectors, wind generation and other fields. Smooth operation and high efficiency of gears are necessary for the normal running of machines. Therefore, gear analysis is an important activity in the field of condition monitoring and fault diagnosis. Early detection of local gear faults in industrial environments is very important to optimize the maintenance schedule and reduce the operating cost

of gearbox damage. Failures of the gearbox may cause injury to human beings and important economic losses. To avoid the consequences of any harmful accidents, several techniques are developed in condition monitoring to detect faults as early as possible. Vibration signal analysis is the most common gear detection technique for gear damage detection.

Through gear vibration analysis, tons of features are acquired, and therefore the next step is optimization and classification. In the present work the authors present a review of a spread of diagnosis techniques for gearbox fault identification with reference to vibration analysis. The vibration techniques were developed for two main reasons. The first purpose is to separate the gearbox related signal from other components and to minimize the noise that may mask the gearbox signal, especially in the early stages of the fault. The second purpose is to spot the status of the gearbox, to differentiate the great and

International Journal of Scientific Research and Engineering Development-– Volume 7 Issue 3, May-June 2024 Available at www.ijsred.com

therefore the faulty gear and to point the defective components. Nowadays the strain for condition monitoring and vibration analysis aren't any longer limited trying to attenuate the results of machine failures, but to utilize existing resources more effectively.

1.2 Fault detection and diagnosis with vibration analysis

Fault detection and diagnosis through vibration analysis involves identifying a machine's condition or faults based on symptoms, akin to medical diagnosis. Vibration is often seen as a potential sign of gearbox issues. Although gearbox vibration is complex, it provides abundant data, akin to a signal for gearbox conditions.

To detect and diagnose impending failures, a thorough understanding of the evidence regarding the failure mode and methods of collecting and quantifying the evidence is necessary. While some component faults can be easily detected physically using methods like microscopy or x-ray, these methods often require component removal and can be impractical for operational machinery. Hence, non-intrusive fault detection methods, especially those based on vibration analysis, are commonly employed for routine monitoring.

Modern gear diagnostic techniques mainly rely on analysing vibration signals collected from the gearbox casing. The primary aim is to detect and classify faults early and monitor their progression to estimate residual machine life and plan maintenance effectively. The key components of geared vibration spectra typically include gear meshing frequency and its harmonics, along with sidebands indicating modulation phenomena. An increase in the number and amplitude of such sidebands can signal a fault condition, and their spacing provides additional information about the fault source.

Identifying the source and detecting faults from vibration signals associated with rotating components such as gears, rotors, bearings, and couplings depend on several factors including rotational speed, background noise level, transducer location, load sharing characteristics, and dynamic interactions with other components.

Mechanical vibration can arise from various sources including looseness, unbalance, defective bearings, gear inaccuracies, misalignments, critical speeds, drive belt issues, reciprocating forces, oil, and friction whirl, aerodynamic or hydrodynamic forces, resonances, bent rotor shafts, and defective rotor bars.

1.3 Methodology of Experiment:

1. Gear Defect Study: Examine various gear tooth defects.

2. Literature Review: Review fault diagnosis techniques.

3. Vibration Signal Collection: Gather signals from healthy and faulty gears.

4. FFT Analysis: Apply FFT to analyse signals in frequency domain.

5. ANN Implementation: Train ANN using vibration data.

6. Result Comparison: Compare FFT and ANN outcomes.

2. EXPERIMENTAL SETUP:

The setup consists of a motor, gears mounted on two shafts held together by bearings and rotated by motor's shaft itself. Accelerometer has been mounted on the bearings fixture and then further connected to the data acquisition system which is then connected to the laptop where the readings are taken using NI LabVIEW. The VFD is used to control the speed of the motor. Do vibrational analysis of healthy gear using Fast Fourier Transformer (FFT). In lab view first we get vibration graph with respect to time domain then we will convert into frequency domain. Then we will compare the graph of both gear



Fig. No1 Top view of setup

International Journal of Scientific Research and Engineering Development--- Volume 7 Issue 3, May-June 2024 Available at <u>www.ijsred.com</u>



Fig. No2 Experimental Setup



Fig. No 3 VFD Box



Fig. No 4 CAD Model



Fig. No 5 DAQ Box

3. ANSYSANALYSIS OF SPUR GEARMESHING 3.1Introduction:

Spur gears play a vital role in mechanical power transmission systems due to their simplicity and reliability. Analysing the meshing behaviour of spur gears is essential for optimizing system performance and longevity. Ansys software offers robust tools for conducting in-depth analyses of gear meshing phenomena.

3.2 Objective:

The main goal of this analysis is to assess contact patterns, stress distribution, and performance characteristics of spur gears under varying conditions using Ansys. Through advanced simulation techniques, engineers can refine gear designs, prevent potential failures, and improve system efficiency and reliability.

3.3 Methodology:

1. Geometry Creation: Develop 3D models of spur gears in Ansys, ensuring accurate representation of tooth profiles and gear dimensions.

2. Mesh Generation: Create high-quality meshes to accurately capture contact behaviour while optimizing computational efficiency.

3. Material Properties: Assign material properties like Young's modulus and yield strength to simulate mechanical behaviour accurately.

4. Boundary Conditions: Apply realistic operating conditions, including rotational speeds, torque loads, and environmental factors.

5. Contact Analysis: Utilize advanced contact algorithms to analyse gear tooth interaction, contact patterns, and pressure distribution.

International Journal of Scientific Research and Engineering Development-– Volume 7 Issue 3, May-June 2024 Available at <u>www.ijsred.com</u>

6. Stress Analysis: Employ Finite Element Analysis to compute stress distribution, identifying critical areas prone to failure.

7. Performance Evaluation: Assess transmission efficiency, tooth load distribution, and gear deflection to evaluate system performance.

3.4 Results and Discussion:

The Ansys analysis provides insights into spur gear behaviour under different conditions. Contact analysis reveals efficient power transmission, while stress analysis identifies high-stress areas for design optimization. Key results include total deformation, maximum principle elastic strain, and maximum principal stress values.

3.5 Conclusion:

The Ansys analysis showcases the software's capability in evaluating mechanical components. By using advanced simulation techniques, engineers can enhance gear designs, improve reliability, and optimize system efficiency.

4. MACHINE LEARNING CLASSIFICATION MODELS

The use of Isolation Forest and Random Forest for gear fault detection is a promising approach in the field of mechanical fault diagnosis. Both algorithms have been successfully applied in various studies to detect faults in manufacturing equipment and predict equipment failures.

4.1 Isolation Forest:

Isolation Forest is an unsupervised anomaly detection algorithm that operates on the principle that anomalies are few and different, thus easier to isolate. It works by recursively partitioning the data space and computing an anomaly score based on the path length required to isolate a data point. Anomalies tend to have shorter paths as they are easier to isolate. This algorithm is suitable for scenarios where labelled fault data is scarce and is effective in high-dimensional spaces with minimal parameter tuning.



4.2 Random Forest:

Random Forest is a supervised learning algorithm used for classification and regression tasks. It combines predictions from multiple decision trees to make a final decision, introducing diversity among the trees through bootstrap aggregating and feature randomness. This algorithm is ideal for supervised scenarios with sufficient labelled data, providing high accuracy and interpretability through feature importance metrics.



Fig. No 7 Random Forrest

4.3 Comparison and Conclusion:

The choice between Isolation Forest and Random Forest for gear fault detection depends on the data availability and the specific requirements of the fault detection task. Isolation Forest is ideal for unsupervised anomaly detection when labelled fault data is not available, while Random Forest excels in supervised classification with labelled data, providing high accuracy and interpretability. Both algorithms have their unique strengths and can be

International Journal of Scientific Research and Engineering Development--- Volume 7 Issue 3, May-June 2024 Available at www.ijsred.com

effectively used for gear fault detection, depending on the specific context and data availability.

5. MACHINE LEARNING MODEL PROGRAMMING AND RESULT

<pre>import pandas as pd df - pd.read_excel(r'C:\Users\WVBXCR\Documents\Wy Project\Tarmay_project\bfull 1000_1 (.xisx') In [3]: df.aise Out[3]: d000 In [4]: import pandas as pd f Create a DataFrame from your data f data -: f cation - dir(arceleration - dir(arceleration) - dir</pre>	<pre>pandas as pd .read_excel(r'C:\Users\NVBSKOR\Documents\My Project\Tanmay_project\bfull 1000_26\0) e a a a a a a a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a b a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c a c c a c c a c c a c c a c c a c c a c c a c c a c c a c c a c c a c c a c c a c c a c c a c c c c c c c c c c c c c</pre>	In [2]:	
<pre>df - pd.read_excel(t'C:\Users\NVBJKOR\Documents\My Project\Tanmay_project\bfull 1000_3 0.x1x*) Tn [3]: df.aize dout[3]: df.aize dout[3]: import pandae as pd f Create a DataFrame from your data f data = [</pre>	<pre>sead_excel(r'C:\Users\WVB3KOR\Documents\Wy Project\Tanmay_project\bfull 1000_2600) pandas as pd se a DafFrame from your data</pre>	import pandas	as pd
<pre>fm [3]: df.size dd.size dd.[3]: d000 fm [d]: import pandes as pd # Create a DataFrame from your data # data = { ''AccelerationTime*': [0.000000, 0.000039, 0.000078, 0.00017, 0.000156], ''AccelerationTime*': [-2.128012, -1.272800, -0.584086, 0.395377]] ''Acceleration*': [-2.128012, -1.272800, -0.584086, 0.395377]] ''Acceleration*': [-2.128012, -1.272800, -0.584086, 0.395377]] ''Acceleration*': [-2.128012, -1.272800, -0.584086, 0.395377]] ''Acceleration = df['Acceleration'].mean[acceleration] thus = df.Bab(aff'Acceleration*'] = mean[acceleration] > threshold] print('Fineshold (eg., 2 standard deviations from the mean) threshold = attrashold (eg., 2 standard deviations from the mean) threshold = attrashold (eg., 2 standard deviations from the mean) threshold for Fault Detections*, threshold] print('Fineshold for Fault Detections*, threshold] print('Fineshold for Fault Detections*, threshold) print(faults) Wean Acceleration.clice* Acceleration: 4.555184217541365 Threshold for Fr Fr Detections 9.11043843308273 Detection*, acceleration*, acceleration 4</pre>	<pre>pandas as pd te a DataFrame from your data - { Acceleration_Time*': [0.000000, 0.000038, 0.000078, 0.00017, 0.000156], Acceleration_Time*': [0.000000, 0.000038, 0.000078, 0.00017, 0.000156], Acceleration i: [c2.128012, -1.951796, -1.272800, -0.584006, 0.359377]) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame(data) pd.bataFrame</pre>	df - pd.read_e: 0.xlsx')	ccel(r'C:\Users\NVB3KOR\Documents\My Project\Tanmay_project\bfull 1000_26\0
<pre>df.size Out[3]: 4000 In [4]: import pandes as pd # Create a DataFrame from your data # data = { 'Acceleration_Time*: [0.000000, 0.000039, 0.000078, 0.000117, 0.000156], 'Acceleration_Time*: [0.000000, 0.000039, 0.000078, 0.000117, 0.000156], 'Acceleration_Time*: [0.000000, 0.000039, 0.000078, 0.000117, 0.000156], 'Acceleration': [-2.128012, -1.91796, -1.272800, -0.584086, 0.395377] ' dccleration control (-2.128012, -1.91796, -1.272800, -0.584086, 0.395377] ' dccleration control (-2.12800, -0.584086, 0.395377] ' dccleration control (-2.0810, -2.12800, -0.584086, 0.395377] print (**mendod for Fault Detection: *, threshold) print (**mendod for Fault Detection: *, threshold print (**mendod for Fault Detection: *, threshold) print (**mendod for Fault Detection: *, threshold</pre>	<pre>pandas as pd pandas as pd te a DataFrame from your data</pre>	In [3]:	
<pre>out[3]: 4000 In [4]: import pads as pd # drate = [# drate = not possible =</pre>	<pre>pandas as pd is a DataFrame from your data - { - { - { - { - { - { - { - {</pre>	df.size	
<pre>4000 In [4]: Import pandas as pd f Creats a DataFrame from your dats d data = {</pre>	<pre>pandas as pd pandas as pd te a DataFrame from your data - { 'Acceleration_Time'': [0.000000, 0.00039, 0.000078, 0.00017, 0.000156), Acceleration [:c.1.128012, -1.1551786, -1.272800, -0.584086, 0.395377] pd.DataFrame(data) hate mean and standard deviation of acceleration coleration - df('Acceleration'].sedu() be a threshold (e.g., 2 standard deviations from the mean) bid - 2 * std_acceleration tf fulls - df(abs(df['Acceleration'] - mean_acceleration) > threshold] Wean Acceleration:*, mean_acceleration) 'Mean Acceleration:*, mean_acceleration) 'Mean Acceleration:*, mean_acceleration) 'Mean Acceleration: 4.555184217541365 hd for Fault Detection: 9, 110436643508273 df Faults coeleration_Time* Acceleration 0.00132815.09507 0.00132815.09507 0.001328 - 10.59507 0.001328 - 11.59527 ws x 2 columns] pandas as pd matplotlib.pyplot as plt pandas as pd matplotlib.pyplot as plt pandas as pd</pre>	Out[3]:	
<pre>In [4]: Import pandas as pd # Creats a DataFrame from your data # data = { # 'Acceleration Time': [0.000000, 0.000039, 0.000078, 0.000117, 0.000156], # 'Acceleration : [-1:20012, 1.1931796, -1:2728000, -0.584086, 0.395377]) # 'Calculate mean and standard deviation of acceleration mean_acceleration = df('Acceleration'].mean() # define a threshold (e.g., 2 standard deviations from the mean) threshold - 2 * sta_acceleration'] - mean_acceleration) # Define a threshold (e.g., 2 standard deviations from the mean) threshold - 2 * sta_acceleration', mean[acceleration) > threshold] print('Mam Acceleration', mean_acceleration) print('WataAcceleration', mean_acceleration) print('WataAcceleration', mean_acceleration) print('WataAcceleration', mean_acceleration) print('WataAcceleration of Acceleration', std_acceleration) print('WataAcceleration' 4.5552184217541365 Threshold for Fault Detection: 4.5552184217541365 Threshold for Fault Detection: 4.5552184217541365 Threshold for Fault Detection: 9.11043843508273 Detected Paults: Acceleration Time' Acceleration # Ou01237 - 10.20643 # 0.001285 - 10.20643 # 0.002808 - 11.337856 1939 0.078008 - 11.337856 1939 0.078008 - 11.358427 [133 rows x 2 columns] In [5]: import matpleTilh.pyplot as plt</pre>	<pre>pandas as pd pandas as pd c a DubaFrame from your data - { /*Acceleration_fime'': [0.000000, 0.00039, 0.000078, 0.000176, 0.000156), /Acceleration fime'': [-2.128012, -1.531786, -1.272800, -0.584086, 0.395377]) pd.DataFrame(data) late mean and standard deviation of acceleration coeleration - df('Acceleration'].std() set a threshold (e.g., 2 standard deviations from the mean) lat - 2 * std_acceleration :: faults - df(abs(df['Acceleration'] - mean_acceleration) > threshold] Wean Acceleration: *, mean_acceleration) 'Threshold for faceleration: *, std_acceleration) 'Threshold for faceleration: *, std_acceleration) 'Detected Faults Detection: *, threshold) Detected Faults: ' in faults coeleration_fime* Acceleration 0.0012815.09507 0.001367 -10.094204 0.001363 -12.647133 0.001365 -10.20464 0.077891 14.707655 0.078006 -11.55837 0.078006 -11.55837 0.078066 -11.55837 we x 2 columns] pandas as pd matplotlib.pyplot as plt pandas as pd matplotlib.pyplot as plt</pre>	4000	
<pre>import pandas as pd f Creats a DataFrame from your data f data = {</pre>	<pre>pandas as pd pandas as pd c: a DubaFrame from your data - { /*Acceleration_fime'': [0.000000, 0.00039, 0.000078, 0.000176, 0.000156], Acceleration = file('.cceleration').cceleration cceleration = df'('Acceleration').scel() be a threshold (e.g., 2 standard deviations from the mean) bid = 2 * std_acceleration tf faults - df [abs(df['Acceleration'] - mean_acceleration] > threshold] Wean Acceleration = df Acceleration Mean Acceleration: #, mean_acceleration) Mean Acceleration: #, std_acceleration) Mean Acceleration: #, std_acceleration) Threshold for Acceleration: #, std_acceleration) Mean Acceleration: #, St52184217541365 bid for Fault Detection: #, 5152184217541365 bid for Fault Detection: #, 5152184217541365 bid for Fault Detection: #, 5552184217541365 bid for Fault Detection: #, 555287 0.0036711.094204 0.00136710.20464 0.007895 - 10.20464 0.00789613.377656 0.07800613.377656 0.07800713.57587 0.078066 - 11.558477 we x 2 columns] pandas as pd matplotlib.pyplot as plt pandas as pd</pre>	In [4]:	
<pre>f Create = DataFrame from your data data = {</pre>	<pre>e a DataFrame from your data = {</pre>	import pandas a	as pd
<pre>'/acceleration_time*t:[0.000000, 0.000039, 0.000078, 0.00017, 0.000156], '/acceleration':[-128012, -1.91796, -1.272800, -0.584086, 0.395377] 'df = pd.DataFrame(data) t Calculate mean and standard deviation of acceleration mean_acceleration = df['Acceleration'].mean() atd_acceleration = df['Acceleration'].mean() atd_acceleration = df['Acceleration'] = mean_acceleration) > threshold] threshold = 2 * std_acceleration; 'Detect faults faults = df[abs(df['Acceleration'] = mean_acceleration) > threshold] print ('Wean Acceleration:", mean_acceleration) print ('Wean Acceleration: ", mean_acceleration) print ('Wean Acceleration: ", threshold) print ('Wean Acceleration: ", threshold) print ('Weatalion of Acceleration: ", threshold) print ('Thetected Faults: ") print (Thetected Fa</pre>	<pre>Acceleration_Time*': [0.000000, 0.00038, 0.00078, 0.000174, 0.000155, Macceleration[:[-2.12012, -1.951796, -1.272800, -0.584086, 0.395377] pd.DataFrame(data) late seen and standard deviation of acceleration beleration - df['Acceleration'].sed() beleration - df['Acceleration'].sed() set attrached(e.g., 2 standard deviations from the mean) ld - 2 * std_acceleration tf faults - df labs(df['Acceleration'] - mean_acceleration) > threshold] Wean Acceleration.**, std_acceleration) Threshold for Fault Detection:*, threshold) Wean Acceleration Threshold for Fault Detection:*, threshold) Detected Faults: 10 for Fault Detection: 4.555184217541365 ld for Fault Detection: 9.10436643508273 df Faults Ucceleration_Time* Acceleration 0.00132615.09557 0.00132616.09557 0.001357 - 10.202643 0.001365 - 10.202643 0.007930 14.707556 0.07800611.555827 we x 2 columns] pandas as pd matplotlib.pyplot as plt pandas as pd</pre>	<pre># Create a Data # data = {</pre>	iFrame from your data
<pre># df = pd.DataFrame(data) # Calculate mean and standard deviation of acceleration mean_acceleration = df('Acceleration'].mean() std_acceleration = df('Acceleration'].mean() # Define a threshold (e.g., 2 standard deviations from the mean) threshold = 2 * std_acceleration # Detect faults faults = df(akceleration'] = mean_acceleration) > threshold] print('Mean Acceleration', mean_acceleration) print('Mean Acceleration', mean_acceleration', threshold) print('Mean Acceleration', std_acceleration', threshold) print('Mean Acceleration', std_acceleration', threshold) print('Mean Acceleration', std_acceleration', threshold) print('Mean Acceleration', std_acceleration', threshold', std_acceleration', std_acceleration', threshold', std_acceleration', std_acce</pre>	<pre>pd.DataFrame(data) late mean and standard deviation of acceleration a (1/Acceleration'].sen() seleration = d('Acceleration'].std() be a threshold (e.g., 2 standard deviations from the mean) loid = 2 * stackceleration tf fulis = df(abc(df'Acceleration'] = mean_acceleration) > threshold] Wean Acceleration; mean_acceleration) Treshold for Fault Detection:*, std_acceleration) Threshold for Acceleration:*, std_acceleration) Detected Faults:*) coeleration: -0.744272654005699 d Deviation of Acceleration: 4.555184217541365 loid for Fault Detection: 9.10436843508273 dd Faults: toceleration_Time* Acceleration</pre>	# 'Accelera # 'Accelera	tion_Time*': [0.000000, 0.000039, 0.000078, 0.000117, 0.000156], ation': [-2.128012, -1.951796, -1.272800, -0.584086, 0.395377]
<pre># Cloulate mean and standard deviation of acceleration mean_acceleration - df ['Acceleration'].mean() st_d_acceleration - df ('Acceleration'].it() # Define a threshold (e.g., 2 standard deviations from the mean) threshold - 2 + st_acceleration'] - mean_acceleration) > threshold] print ('acceleration', mean_acceleration) > threshold] print ('acceleration', mean_acceleration) print ('acceleration', mean_acceleration) print ('acceleration', mean_acceleration) print ('acceleration', mean_acceleration) print ('acceleration', mean_acceleration) print ('acceleration', mean_acceleration', std_acceleration) print ('acceleration', mean_acceleration', std_acceleration) print ('acceleration', mean_acceleration', std_acceleration', threshold) print ('acceleration', std_acceleration', std_a</pre>	<pre>plate mean and standard deviation of acceleration coeleration - df('Acceleration').scd() se a threshold(e.g., 2 standard deviations from the mean) ld - 2 * std_acceleration rt faults - df(labs(df('Acceleration') - mean_acceleration) > threshold) "Mean Acceleration.", mean_acceleration) "Standard Deviation of Acceleration() "Standard Deviation of Acceleration() "Standard Deviation of Acceleration() Detected Faults:") aults) cceleration: -0.7442729584085699 df dfults: cceleration.rime* Acceleration 0.001326 - 11.04956043508273 df Faults: cceleration_rime* Acceleration 0.00132510.09507 0.001357 -10.09507 0.001357 -10.020643 0.00755 - 10.220643 0.007030 14.707556 0.07800611.558427 we x 2 columns] pandas as pd matplotlib.pyplot as plt pandas as pd matplotlib.pyplot as plt</pre>	#) # df = pd.Datal	'rame(data)
<pre># Define a threshold (e.g., 2 standard deviations from the mean) threshold - 2 * std_acceleration # Detect faults faults - dflabs(df['Acceleration'] - mean_acceleration) > threshold] print('Threshold for Fault Detections', std_acceleration) print('Threshold for Fault Detections', std_acceleration) print('Detected Faults:') print('Detected Faults:') Detected Faults:</pre>	<pre>me a threshold (e.g., 2 standard deviations from the mean) ld = 2 * std_acceleration : ffaults = df[abs(df['Acceleration'] - mean_acceleration) > threshold] /Mean Acceleration (frequencies) /Standard Deviation of Acceleration: *threshold for Fault Detection:*, threshold) /Detected Faults:*) /Aults) // Coeleration: =0.7442729584085699 rd Deviation of Acceleration: 4.552184217541865 ld for Fault Detection: : 10436648508273 df Faults: // Coeleration: Time* Acceleration // 0.00182611.049507 // 0.00182612.647134 // 0.00182612.647134 // 0.00182613.977656 // 0.07800611.958427 // 0.07806611.958427 // ws x 2 columns] // marks as pd matplotlib.pyplot as plt // pandas as pd // marks a</pre>	<pre># Calculate mean_accelerat: std_acceleration</pre>	n and standard deviation of acceleration Lon - df('Acceleration'].mean() n - df('Acceleration'].std()
<pre># Datact faults faults - df[abs(df['Acceleration'] - mean_acceleration) > threshold] print('Wata Acceleration:', mean_acceleration) print('Vatandard Deviation of Acceleration:', threshold) print('Vatandard Deviation of Acceleration:', threshold) print('Vatandard Deviation', threshold, thresh</pre>	<pre>rt faults - df(ab(df'Acceleration') - mean_acceleration) > threshold] Wean Acceleration:*, mean_acceleration) Threshold for Fault Detection:*, std_acceleration) Threshold for fault Detection:*, threshold) Detected Faults:*) Detected Faults:*) Detected Faults:* Detection: 001226 - 15.09507 0.001326 - 15.09507 0.001326 - 14.094204 0.001484 9.70030 0.001355 - 10.20464 0.001495 - 10.20464 0.007930 8.436500 0.078008 - 11.955827 ws x 2 columns] Detected paults: pandas as pd matplotlib.pyplot as plt pandas as pd</pre>	<pre># Define a thre threshold = 2</pre>	shold (e.g., 2 standard deviations from the mean) * std_acceleration
<pre>print("%man Acceleration", man Acceleration) print("international Deviation of Acceleration:", stad, acceleration) print("breached Faults Detections", threshold) print("faults) Mean Acceleration: -0.744272958408509 Standard Deviation of Acceleration Standard Deviation of Acceleration Acceleration context Acceleration Acceleration Acceleration context Acceleration Acceler</pre>	<pre>"Bean Acceleration: * mean_acceleration) "Standard eviation of Acceleration:*, std_acceleration) "Detected Fults:*) faults: faults: faults: cceleration: -0.7442729584085699 df Deviation of Acceleration: 4.5552184217541365 ld for Fult Detection: 9.110438643508273 df Faults:</pre>	<pre># Detect fault: faults = df[ab;</pre>	; s(df['Acceleration'] - mean_acceleration) > threshold]
Mean Acceleration: -0.744725954005509 Standard Deviation of Acceleration: 4.95552184217541365 Threshold for Fault Detection: 9.110436843508273 Detected Faults: Acceleration_Time* Acceleration 34 0.001287 -14.094204 35 0.001367 -14.094204 48 0.001875 -10.220643 48 0.001875 -10.220643 49 0.001875 -10.220643 49 0.077891 14.707655 1995 0.0778008 -13.377656 1997 0.078008 -13.377656 1998 0.078066 -11.956427 1133 rows x 2 columns] In [5]:	coleration: -0.7427255408569 d d Poviation of Acceleration: 4.555184217541365 dd for Fault Detection: 9.110436843508273 dd Faults:	print("Mean Aco print("Standary print("Thresho print("Detected print(faults)	<pre>seleration:", mean_acceleration) 1 Deviation of Acceleration:", std_acceleration) id for Fault Detection:", threshold) i Faults:")</pre>
Acceleration_Time* Acceleration 34 0.001228 -15.095907 35 0.001367 -14.094204 36 0.001487 -10.094204 48 0.001875 -10.220643 1994 0.0071891 14.707655 1995 0.077939 8.436500 1997 0.078008 -13.377656 1998 0.078606 -11.956427 1938 0.078606 -11.956427 1133 rows a 2 columns] In [5]: import matplotlib.pyplot as plt	<pre>kcceleration_Time* Acceleration</pre>	Mean Accelerati Standard Deviat Threshold for I Detected Faults	on: -0.7442729584085699 ion of Acceleration: 4.5552184217541365 'ault Detection: 9.110436843508273 ::
35 0.001367 -14.094204 38 0.001484 9.370030 47 0.001836 -12.647134 48 0.001875 -10.220643 1994 0.077930 8.436500 1995 0.077930 8.436500 1997 0.078047 -17.755387 1998 0.078047 -11.958427 [133 rows x 2 columns] In In [5]: Import pandas as pd import pandat as pd In	0.001367 -14.094204 0.001367 -12.647134 0.001875 -10.220643 0.077931 14.707655 0.0778047 -17.755387 0.078066 -11.958427 www.s.2 columns] pandas as pd matplotlib.pyplot as plt pandas as pd	Accelerat 34	ion_Time* Acceleration 0 001328 -15 099507
38 0.001484 9.370030 47 0.001836 -12.647134 48 0.001875 -10.220643 1994 0.077950 14.707155 1995 0.077950 8.426500 1997 0.078047 -13.377555 1998 0.078047 -11.958427 (133 rovs x 2 columns) In [5]: import pandas as pd import matplotlib.pyplot as plt	0.001484 9.370030 0.0018510.220643 0.00187510.220643 0.077891 4.30566 0.07980 6.315566 0.07804717.755387 0.07806611.958427 www.x.2 columns] pandas as pd matplotlib.pyplot as pit pandas as pd	35	0.001367 -14.094204
46 0.001875 -10.220643 1994 0.077891 14.707655 1995 0.077890 8.436500 1997 0.078008 -13.377656 1998 0.078006 -11.95587 1999 0.078066 -11.956427 [133 rows x 2 columns] In [5]: Import matplotlib.pyplot as plt	0.001875 -10.220643 0.077891 14.707655 0.077808 8.436500 0.078008 -13.377656 0.078068 -11.55887 0.078066 -11.558427 ws x 2 columns] pandas as pd matplotlib.pyplot as plt pandas as pd	38	0.001484 9.370030
1994 0.077991 14.707655 1995 0.077930 8.436500 1997 0.078008 -13.377656 1998 0.078047 -17.755387 1999 0.078086 -11.958427 (133 rows x 2 columns) In [5]: import pandas as pd import matplotlib.pyplot as plt	0.077891 14.707655 0.077930 8.436500 0.078080 -13.77565 0.078047 -17.755387 0.078066 -11.958427 www.x.2 columns] pandas as pd matploilib.pyplot as plt pandas as pd	48	0.001875 -10.220643
1995 0.077930 8.436500 1997 0.078008 -13.377656 1998 0.078086 -11.958427 [133 rovs x 2 columns] [15]: In [5]: import matplotlib.pyplot as plt	0.077930 8.436500 0.07804713.77656 0.07804717.755387 0.07804611.958427 www.s.2 columns] pandas as pd matplotlib.pyplot as plt pandas as pd	1994	0.077891 14.707655
1997 0.078008 -13.377656 1998 0.078047 -17.75587 1999 0.078086 -11.958427 [133 rows x 2 columns] [11] [11] In [5]: import pandas as pd	0.078008 -13.377656 0.078047 -17.75587 0.078066 -11.958427 was x 2 columns] pandas as pd matplotlib.pyplot as plt pandas as pd	1995	0.077930 8.436500
<pre>1999 0.078086 -11.958427 (133 rows x 2 columns) In [5]: import pandas as pd import matplotlib.pyplot as plt</pre>	0.070086 -11.958427 ws x 2 columns] pandas as pd matplotlib.pyplot as plt pandas as pd	1997	0.078008 -13.377656 0.078047 -17.755387
<pre>[133 rows x 2 columns] In [5]: import pandas as pd import matplotlib.pyplot as plt</pre>	ws x 2 columns] pandas as pd matplotlib.pyplot as plt pandas as pd	1999	0.078086 -11.958427
In [5]: import pandas as pd import matplotlib.pyplot as plt	pandas as pd mstplotlib.pyplot as plt pandas as pd	[133 rows x 2 c	olumns]
import pandas as pd import matplotlib.pyplot as plt	pandas as pd matplotlib.pyplot as plt pandas as pd	In [5]:	
	pandas as pd	import pandas a import matplot	is pd ib.pyplot as plt
import pandas as pd		import pandas	as pd

Calculate mean and standard deviation of acc mean_acceleration = df['Acceleration'].mean() std_acceleration = df['Acceleration'].std() leration

Define a threshold (e.g., 2 standard deviations from the mean, threshold = 2 * std_acceleration

Detect faults
faults = df[abs(df['Acceleration'] - mean_acceleration) > threshold]

Convert DataFrame columns to numpy arrays
time = np.array(df['Acceleration_Time*'])
acceleration = np.array(df['Acceleration'])

import matplotlib.pyplot as plt import numpy as np

Plot the data
plt.figure(figure(10, 6))
plt.plot(time, acceleration, marker='0', linestyle='-', color='b', label='Acceleration')
plt.scatter(faults['Acceleration_Time''], faults['Acceleration'], color='r', label='Faul
t Detected')

Plot mean line and threshold lines
plt.axhline(mean_acceleration, color='g', linestyle='=-', label='Mean Acceleration')
plt.axhline(mean_acceleration + threshold, color='orange', linestyle='-=', label='Thresh
old')
plt.axhline(mean_acceleration - threshold, color='orange', linestyle='=-')

plt.xlabel('Time')
plt.ylabel('Acceleration')
plt.title('Acceleration vs Time with Fault Detection') plt.legend()
plt.grid(True)
plt.show()

Plot the data
plt.figure(i0, 6))
plt.figure(figsize(10, 6))
plt.figure(figsize(10, 6)) //
plt.scatter(faults['Acceleration_Time*'], faults['Acceleration'], color='r', label='Faul
t Detected')

/ Plot sean line and threshold lines
plt schline (mean_acceleration, color-'g', linestyle-'--', label='Mean Acceleration')
plt schline (mean_acceleration + threshold, color-'orange', linestyle-'--', label-'Threshold')
plt.schline(mean_acceleration - threshold, color-'orange', linestyle-'--')

plt.xlabel('Time')
plt.ylabel('Acceleration')
plt.ylabel('Acceleration')
plt.legend(beleration vs Time with Fault Detection')
plt.legend()
plt.grid((True)
plt.show()



In [6]:

import pandas as pd import matplotlib.pyplot as plt from sklearn.ensemble import IsolationForest

Reshape the data
X = df['Acceleration'].values.reshape(-1, 1)

Create the Isolation Forest model
model = IsolationForest(contamination=0.1) # Adjust the co

Fit the model
model.fit(X)
Convert DataFrame columns to numpy arrays # Predict outliers
outliers = model.predict(X)

Convert DataFrame columns to numpy arrays
time = np.array(df['Acceleration_Time*'])
acceleration = np.array(df['Acceleration'])

Plot the data
plt.figure=(10, 6))
plt.plot(time, acceleration, marker='o', linestyle='-', color='b', label='Accel /
plt.scatter(df['Acceleration_Time*'][outliers == -1], df['Acceleration'][outliers == -1],
 color='r', label='Anomaly')

plt.xlabel('Time')
plt.ylabel('Acceleration')
plt.ylabel('Acceleration vs Time with Anomaly Detection (Isolation Forest)')
plt.legend()
plt.gid(True)
plt.shov()



6. CONCLUSION

In conclusion, this research highlights the potential of machine learning techniques in automating bearing fault detection. By leveraging vibration signals and mathematical features, the proposed approach offers a data-driven solution for early fault detection and predictive maintenance. Future research directions include the integration of additional sensor data and the exploration of advanced machine learning algorithms for enhanced fault diagnosis.

ACKNOWLEDGMENT

We are thankful to our beloved HOD Mr. M.M. Bhoomkar sir for his guidance and cooperation for providing us with the DAQ setup and Labview Software. We are thankful to our project guides Prof. Ami Barrot ma'am and Prof. P.G. Kulkarni sir for providing us the guidance for the completion of our project.

REFERENCES

- [1] "VIBRATION ANALYSIS TECHNIQUES FOR GEARBOX DIAGNOSTIC: A REVIEW By Amit Aherwar. Md. Saifullah Khalid
- [2] "Detection of Fault in Gear Box System using Vibration Analysis Method" By LS.Dhamande, A.C.Pawar and V.I Suryawanshi
- [3] "A Review of Gearbox Condition Monitoring Based on vibration Analysis Techniques Diagnostics and Prognostics by Abdulrahman S. Sait, Yahya 1. Sharal Eldeen
- [4] "Vibration Analysis and Control in Mechanical Structures and Wind Energy Conversion Systems" by Paolo Gardonio and Leonardo Lanari
- [5] "Detection of Fault in Gearbox System Using Vibration Analysis Method" by Saurabh S. Shahapurkar and Others
- [6] "Dynamic Modelling and Vibration Analysis for Gear Tooth Crack Detection" by Omar Dawood Mohammed
- [7] "Spur Gear Fault Diagnosis using a Multilayer Gated Recurrent Unit Approach with Vibration Signal" by Ying Taol and Others



In []:

0.04