Fault Detection in Factory Machines using Vibration Data Analytics

Final Report

Group 03

190277L - Hasitha Jayaweera

190287R - Kajanan Selvanesan

190346A - Laksika Tharmalingam

190408R - Pahan Nanayakkara

Table of Contents

Name	Index Number	Contribution
Hasitha Jayaweera	190277L	Business Impact Analysis, Data Security
Kajanan Selvanesan	190287R	Feature Engineering, Predictive Maintenance
Laksika Tharmalingam	190346A	Feature Engineering, Predictive Maintenance
Pahan Nanayakkara	190408R	Business Impact Analysis, Exploratory Data Analysis

Introduction

In today's industrial landscape, the seamless operation of factory machines is paramount to the productivity and safety of the entire sector. Any unexpected machine faults can lead to not only significant economic losses but also potential safety hazards. Predictive maintenance has emerged as a vital strategy to mitigate these risks and ensure the smooth functioning of industrial machinery. This project delves into the world of predictive maintenance, specifically focusing on utilizing vibration data analytics as a powerful tool to detect potential machine faults.

Predictive maintenance is the cornerstone of modern industrial operations, offering a proactive approach to ensure machinery's longevity and reliability. The significance of predictive maintenance can be best understood through its real-world applications:

- Minimizing Downtime: Predictive maintenance allows organizations to anticipate and address potential machine faults before they cause downtime. This leads to continuous production and minimizes revenue losses due to unplanned stoppages.
- Cost Reduction: By identifying and addressing faults in their early stages, predictive maintenance helps reduce repair costs and prevents major breakdowns that can be expensive to fix.
- Enhanced Safety: Ensuring the integrity of machinery through predictive maintenance is crucial for the safety of workers and the surrounding environment. It helps in preventing accidents and hazardous situations.
- Optimized Resource Allocation: By focusing maintenance efforts where they are needed most, organizations can optimize resource allocation, including labor and spare parts inventory.
- Efficiency and Productivity: A well-maintained machine operates at peak efficiency, which is vital for productivity, reducing waste, and maintaining product quality.

In various industries, predictive maintenance has revolutionized the way machinery is managed. Here are some examples:

- Aviation: Airlines use predictive maintenance to monitor aircraft engines' health, ensuring the safety of passengers and reducing unscheduled maintenance events.
- Manufacturing: Factories employ predictive maintenance to keep production lines running smoothly, reducing costly downtimes and production disruptions.
- Energy Sector: Power plants and utilities rely on predictive maintenance to ensure the continuous operation of turbines, generators, and other critical equipment.
- Transportation: Freight and passenger trains use predictive maintenance to maintain rail tracks, engines, and other components, ensuring the safety and efficiency of train operations.
- Healthcare: In the healthcare industry, predictive maintenance is used to maintain medical equipment like MRI machines, ensuring their uninterrupted service to patients.

Objectives

The primary goal of this project is to lead the way in the creation of an anomaly detection model that specializes in identifying patterns in vibration data that could indicate possible machine failures. It is intended for this model to serve as a fundamental resource for maintenance teams, providing them. This model promises to significantly improve operational efficiency, reduce downtime, and optimize resource allocation by anticipating and flagging potential issues. The goal of this project is to completely transform the way maintenance is carried out, opening the door for a proactive, data-driven strategy to guarantee the machinery's continuous operation.

Data Quality & Exploratory Data Analysis

Data quality is measured under the conditions such as accuracy, completeness, consistency, reliability and whether it is upto date.

Null Values

As an initial step, null value counts were checked. Null value percentage was calculated for each feature. ‘SequenceNumber’ had a high null value percentage.

Unique Values

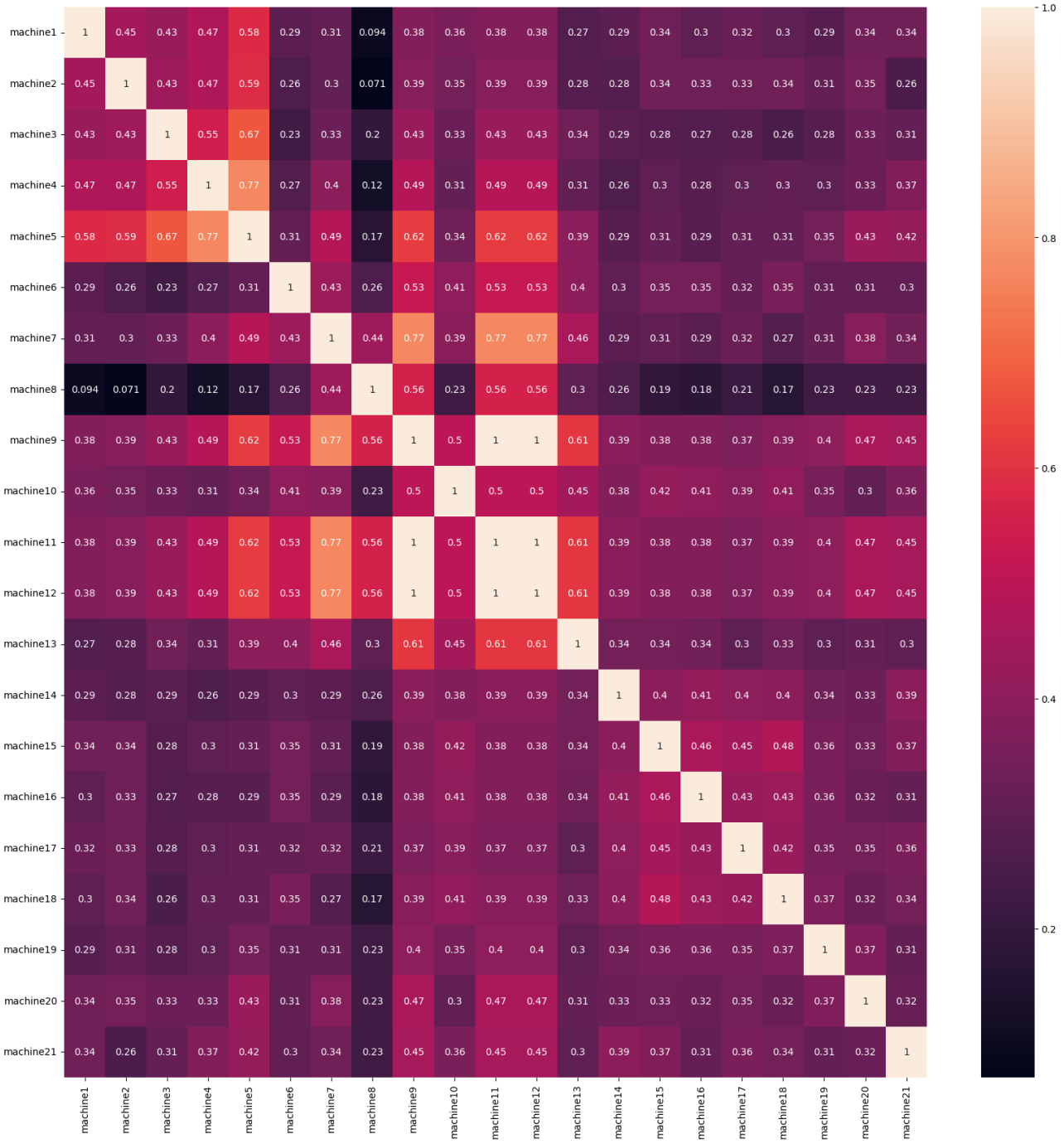
The number of unique values for each feature was checked. This is helpful in determining which features remained constant throughout the time series which is considered in the problem. Features with a single value can be neglected in time series analysis since they have no variation with respect to time.

First, the vibration data and machine status data were merged using the timestamp for the analysis. Since the timestamps were not exactly matching they were merged using the nearest value. After obtaining the merged data frame, data analysis was done.

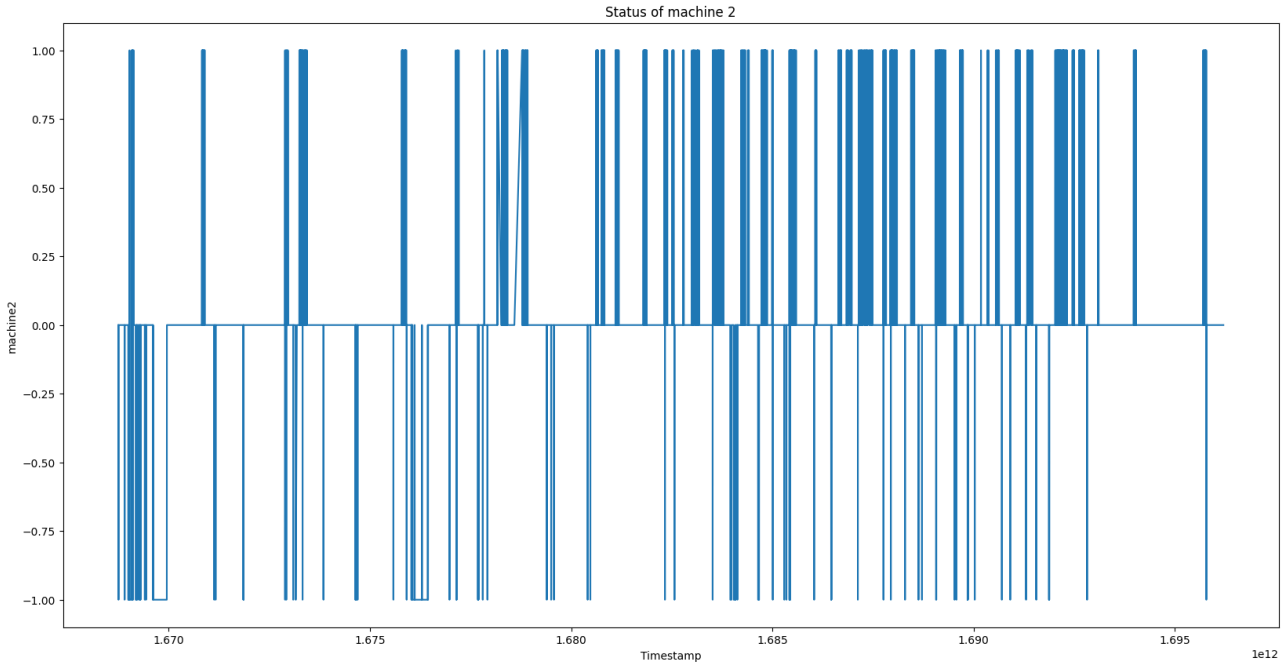
Analysis of features

Machine status

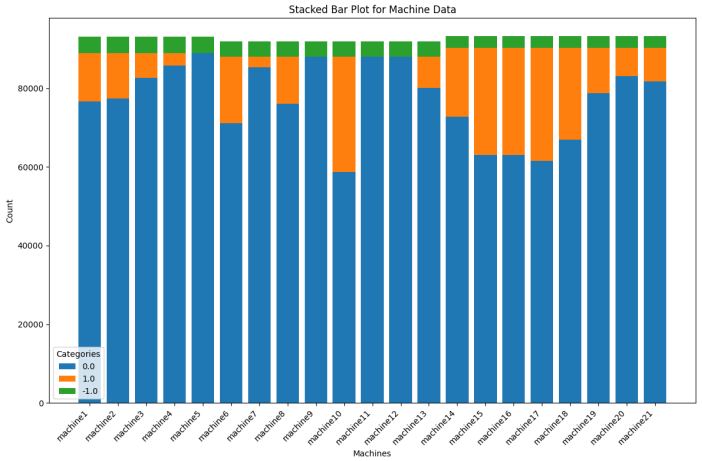
The status of 21 machines in the dataset can contain three values which are 1,0 and -1 representing ‘on’, ‘off’, and ‘unknown’ status respectively. A correlation check was done only on machine status data.



The functionality of machine 2 is important since it is equipped with 4 sensors attached and 20 other machines. The functioning of machine 2 is as follows.



Here the plot shows how the machine status is distributed through all the machines.



It can be observed that machine 2 has spent most of its time switched off (status=0)

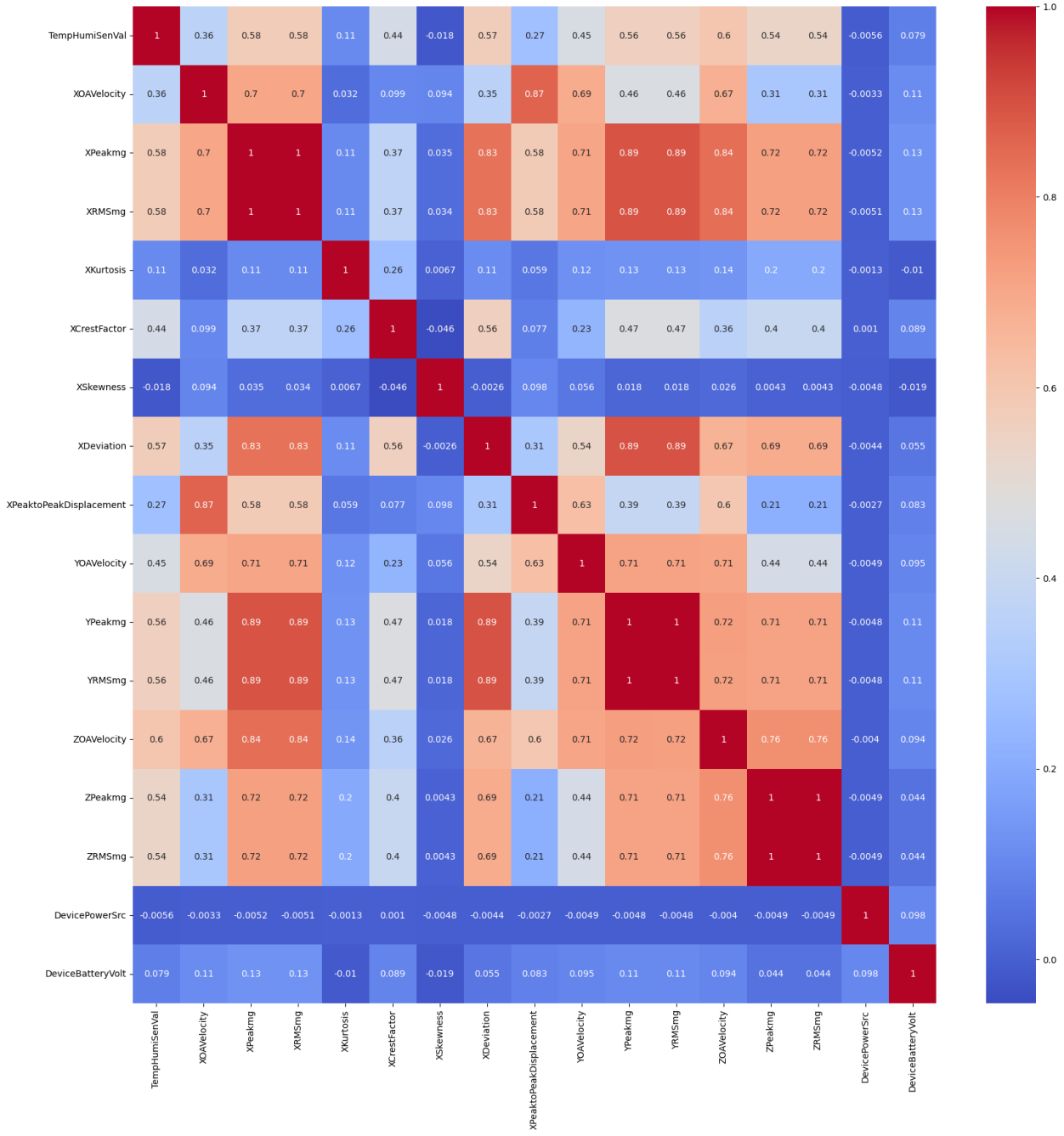
Vibration Data Analysis

After filtering features with higher null value percentages and constant values across time, the following features were selected for further analysis.

'TempHumiSenVal', 'XOAVelocity', 'XPeakmg', 'XRMSmg', 'XKurtosis', 'XCrestFactor', 'XSkewness', 'XDeviation', 'XPeaktoPeakDisplacement', 'YOAVelocity', 'YPeakmg', 'YRMSmg', 'ZOAVelocity', 'ZPeakmg', 'ZRMSmg', 'DevicePowerSrc', 'DeviceBatteryVolt'

Correlation

Correlation values between these selected features were as follows.

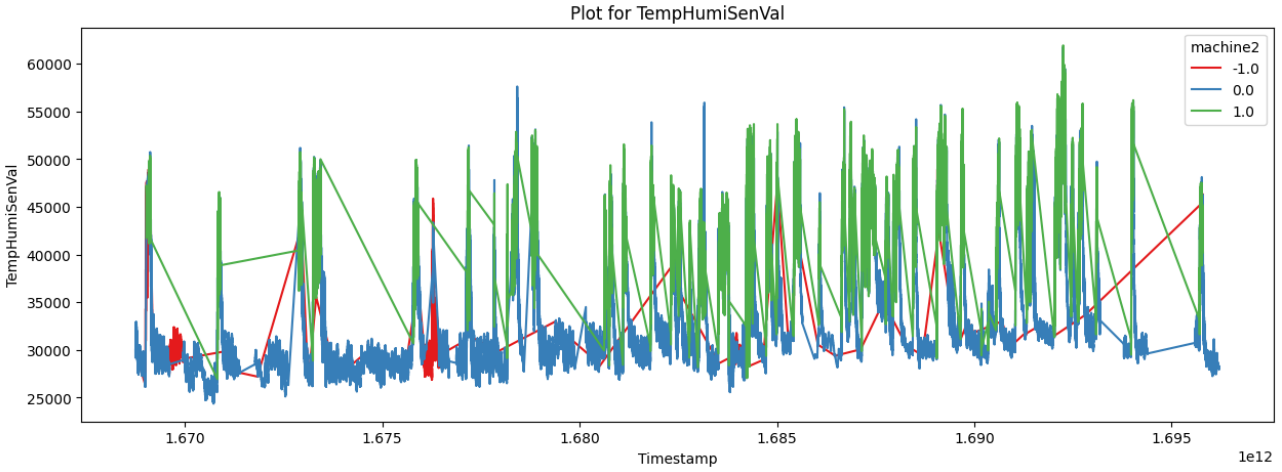


The following features which are further analysed are highly correlated with the status of machine 2.

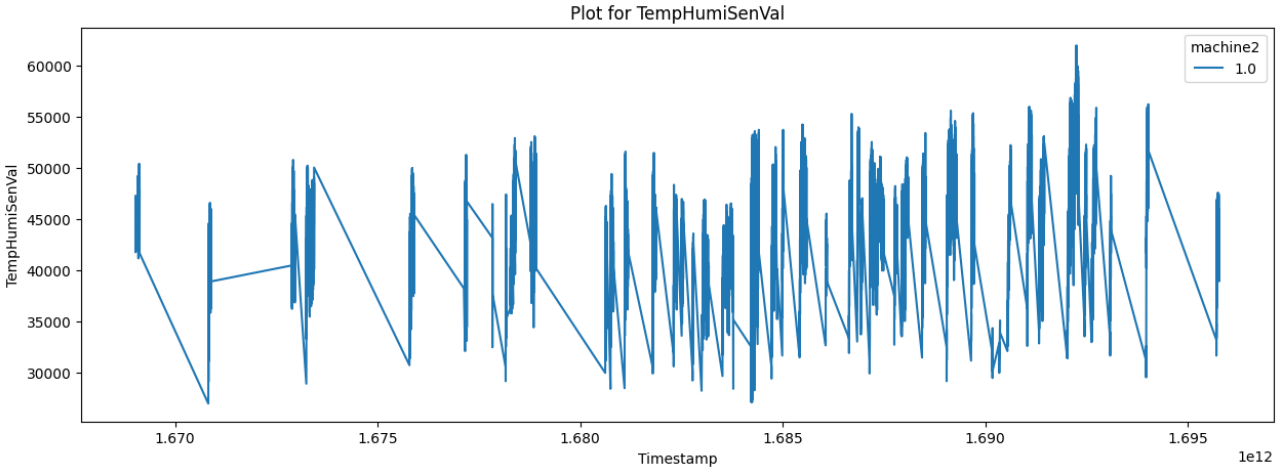
ZOAVelocity	0.630355
XPeakmg	0.592414
XRMSmg	0.592159
TempHumiSenVal	0.58641
ZPeakmg	0.585569
ZRMSmg	0.585509
YPeakmg	0.581767
YRMSmg	0.581529
XDeviation	0.542783
YOAVelocity	0.433045
XCrestFactor	0.422211
XOAVelocity	0.320183
XPeaktoPeakDisplacement	0.258324
XKurtosis	0.109285
DeviceBatteryVolt	0.025091
DevicePowerSrc	-0.027302
XSkewness	-0.042377

The following plots visualize the variation of those features under different status values of machine 2.

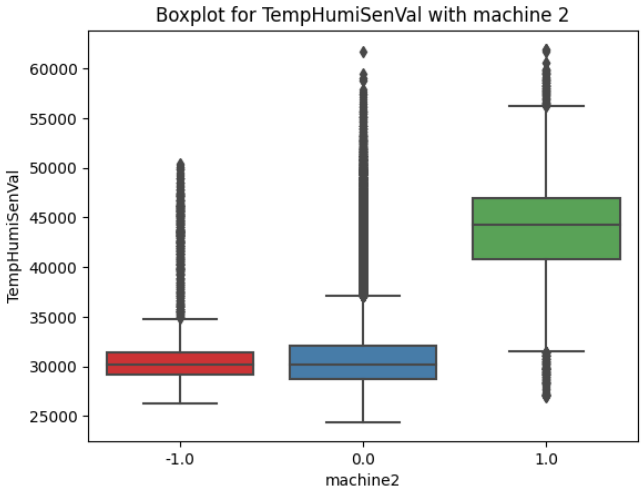
TempHumiSenVal



When the machine is 'on', the variation is as follows.

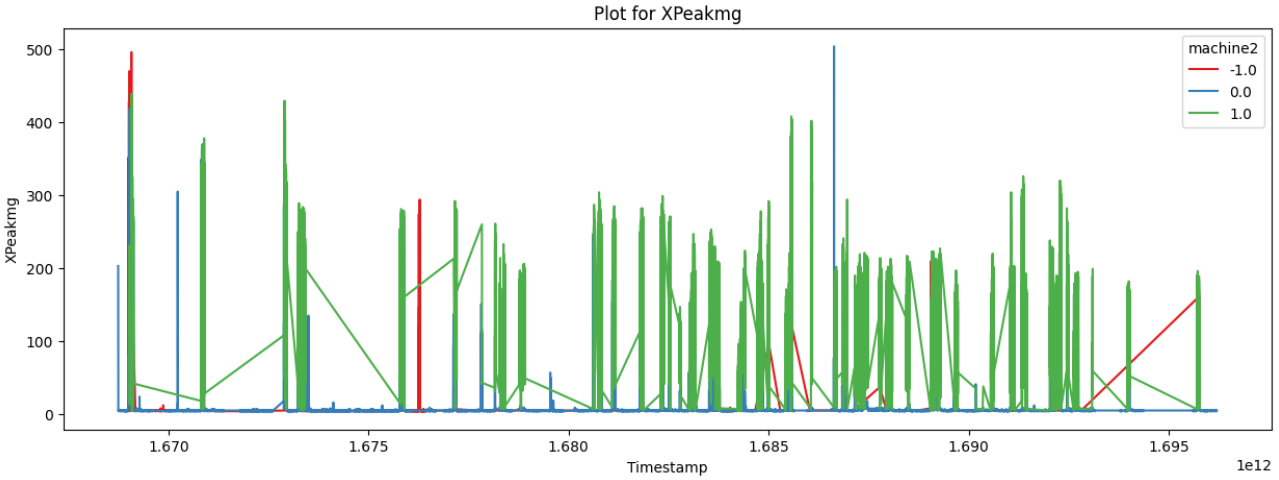


When considering the box plot for the same feature,



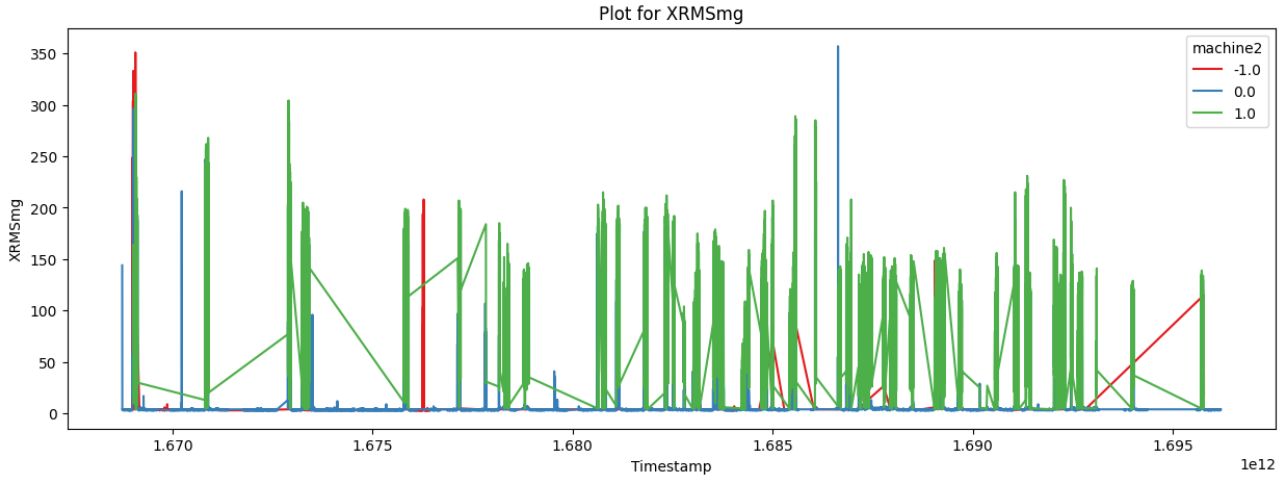
According to the box plot, it can be seen that there is a considerable amount of outliers.

XPeakmg

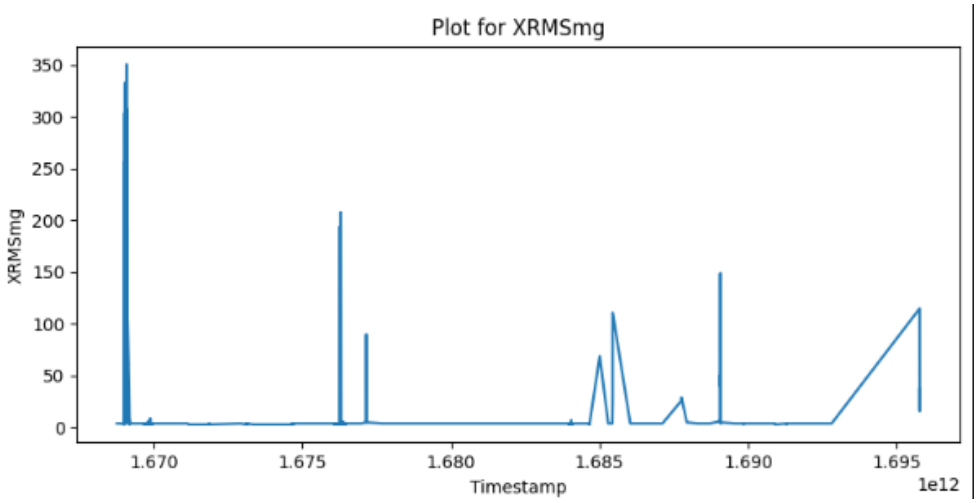


It can be observed that there are clear anomalies in *XPeakmg* when the status of machine 2 is unknown.

XRMSmg

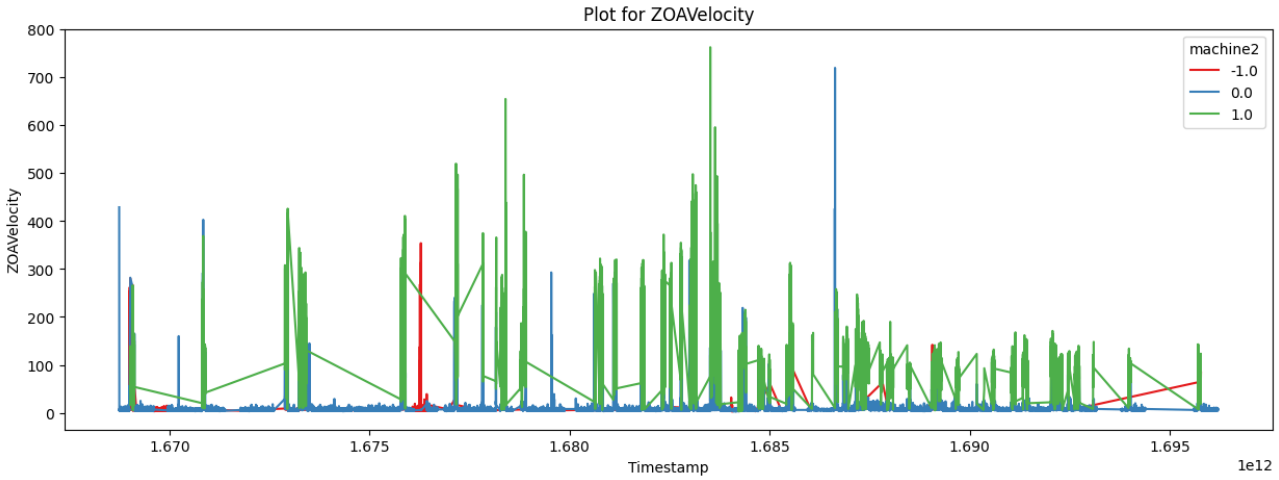


This also shows significant anomalies in certain timestamps when the status of machine 2 is unknown.



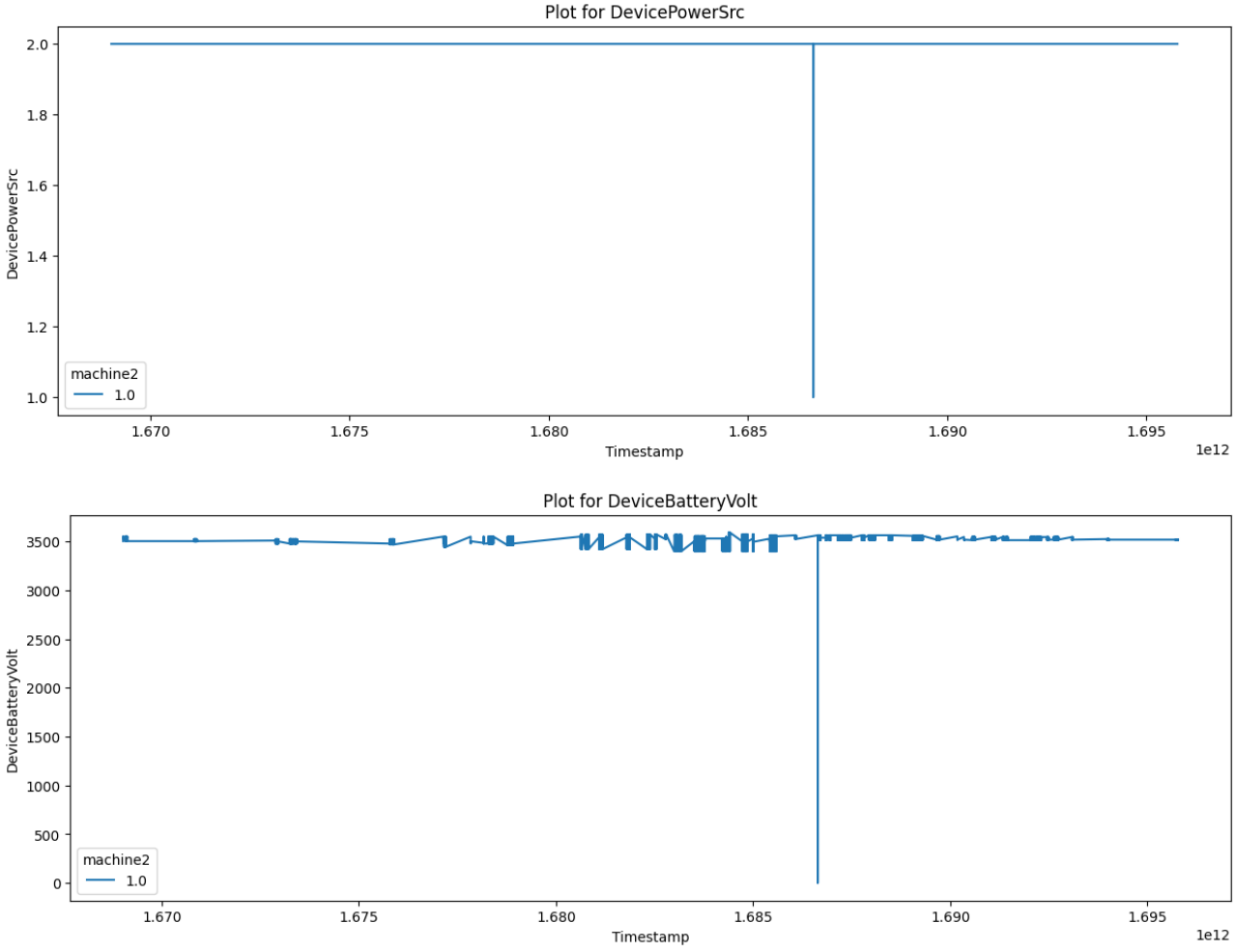
ZOAVelocity

This feature has a correlation of 0.630355 with the status of machine 2.



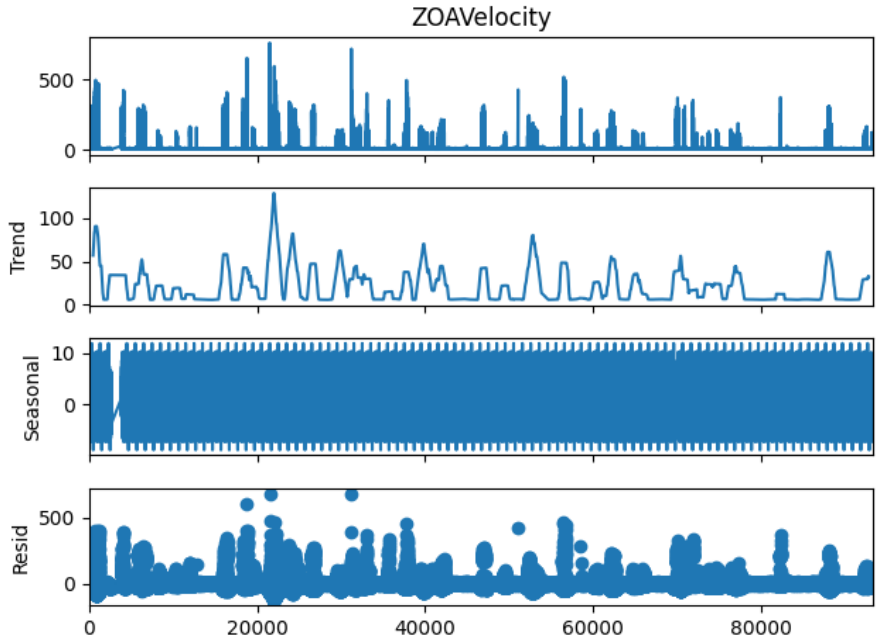
DevicePowerSrc and ***DeviceBatteryVolt***

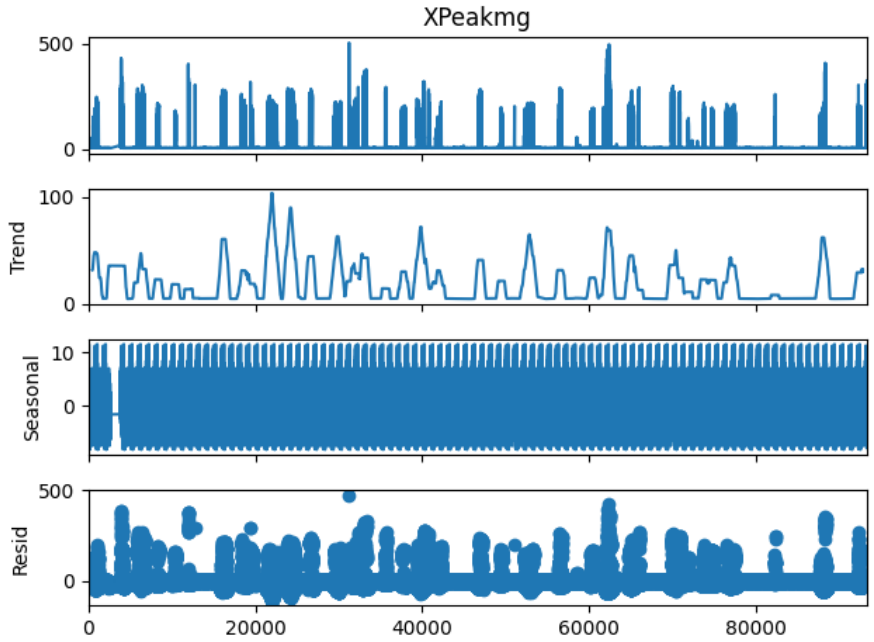
The features *DevicePowerSrc* and *DeviceBatteryVolt* show a significant anomaly in their distributions. This is when the machine 2 is switched on.

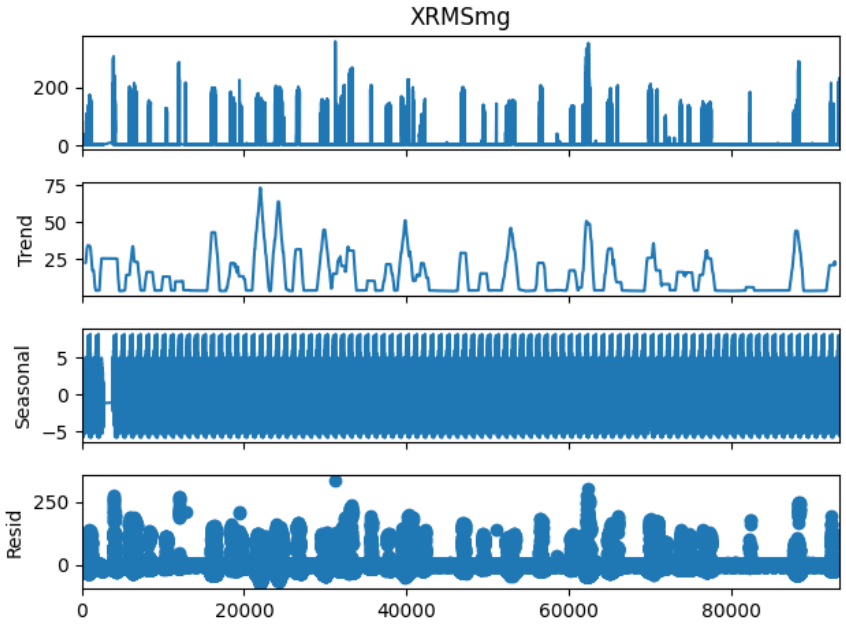


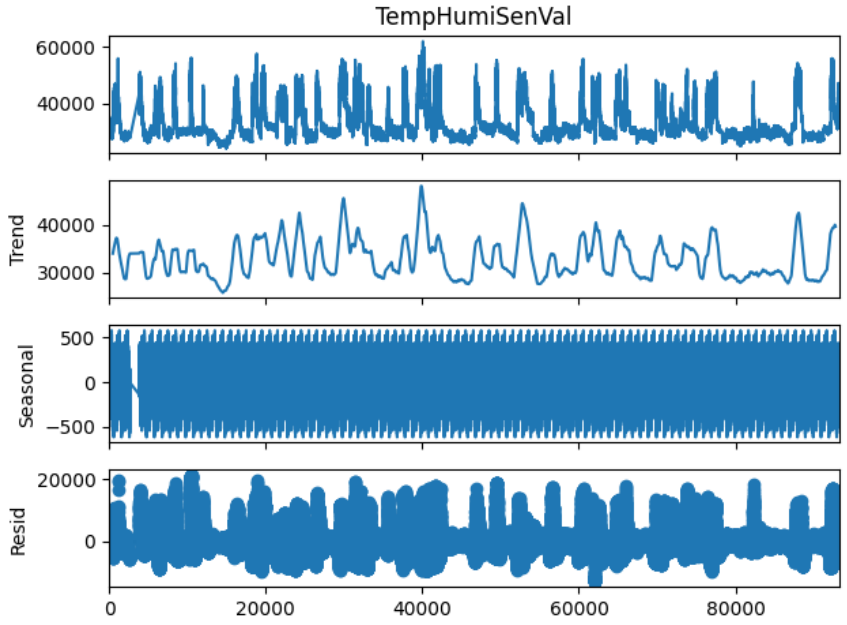
Univariate time series

Seasonal decomposition was performed on each of the selected features in order to obtain insights from trends and seasonality. There were no significant observations by analysis after seasonal decomposition. The following plots show the seasonal decomposition which was done on above mentioned highly correlated features.









Feature Engineering

Data Collection and Organization to Create a Combined Dataset

Dataset Introduction

In this project, we were provided with two distinct datasets: vibration data and machine status data. These datasets were organized on a weekly basis, with each week comprising two essential data sources.

- Vibration Data: This dataset contained detailed information about vibration measurements of m2 recorded from multiple sensors.
- Machine Status Data: The machine status dataset was divided into three sensors, each capable of monitoring eight different machines. These sensors played a crucial role in understanding the status and performance of all 21 machines.

The data was structured systematically, with a folder designated for each week. Inside these weekly folders, we found two key datasets: one for vibration data and another for machine status data. This weekly arrangement provided a logical and organised structure for our analysis. To gather the data, we methodically traversed through the weekly folders. Our objective was to collect both the vibration data and the machine status data from each week, ensuring that we captured all relevant information.

Data Preprocessing

Before we started analysing the data, we realised that getting the data ready for analysis is crucial. So, we took some important steps to prepare the data, such as:

- Handling Missing Values: Rows with missing timestamps were removed, and any inconsistencies in timestamp formatting were rectified.
- Duplicate Management: We identified duplicate records by considering a combination of 'Timestamp' and relevant identifiers ('SensorID' for vibration data and 'UID' for machine status data). To ensure data integrity, only the initial non-duplicate entries were retained.
- To enhance data interpretation, we transformed the 'Timestamp' column into a human-readable 'DateTime' format, '%Y-%m-%d %H:%M:%S'.

Dataset Merging

Since the machine status dataset includes three sensors, each of which can monitor up to eight different machines, it was crucial to combine all this information systematically. Initially, we collected machine status data for each sensor, with each sensor responsible for monitoring a specific subset of the machines. Sensor 1, Sensor 2, and Sensor 3 are used to monitor machines 1 to 5, 6 to 13, and 14 to 21, respectively.

We recognized the importance of timestamps as a common reference point to merge the datasets. The first step was to map the timestamps from the vibration data to the closest corresponding timestamps in each sensor's machine status data. This ensured that for each specific timestamp in the vibration data, we had a nearest matching timestamp in the machine status data of all three sensors. This consolidation allowed for a holistic view of the machines' statuses at that specific point in time.

Outlier Machine Status Handling

For data points where the time difference between vibration and machine status data exceeded a predefined tolerance (15 minutes), we took specific actions. We identified outliers for each sensor category ('sensor1', 'sensor2', and 'sensor3').

	Timestamp	s1_Timestamp_minutes_diff	s2_Timestamp_minutes_diff	s3_Timestamp_minutes_diff
count	1.072540e+05	107254.000000	107254.000000	107254.000000
mean	1.682959e+12	1734.691476	1746.994771	1734.630649
std	8.688836e+09	6227.885838	6225.911193	6227.902302
min	1.668730e+12	0.000017	0.000000	0.000000
25%	1.675262e+12	0.139687	0.140950	0.138033
50%	1.682539e+12	0.285533	0.288067	0.283400
75%	1.690486e+12	0.430229	0.435100	0.429017
max	1.698624e+12	40311.339017	40311.341983	40311.341767

If a data point was deemed an outlier, the corresponding machine status values were set to 0. This process was crucial in maintaining data consistency and reliability.

```
s1_m = [f"machine{i}" for i in range(1, 6)]
s2_m = [f"machine{i}" for i in range(6, 14)]
s3_m = [f"machine{i}" for i in range(14, 22)]

def handle_outlier_machine_status(row):
    if row["s1_Timestamp_minutes_diff"] > tolerance:
        row[s1_m] = 0
    if row["s2_Timestamp_minutes_diff"] > tolerance:
        row[s2_m] = 0
    if row["s3_Timestamp_minutes_diff"] > tolerance:
        row[s3_m] = 0

    return row
```

By adopting these methods, we effectively collected, organized, and preprocessed the data, ensuring it is well-prepared for further exploration to extract valuable insights for our project.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 107254 entries, 0 to 107253
Data columns (total 63 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   Timestamp                 107254 non-null  int64
 1   DateTime                  107254 non-null  object
 2   SensorID                  107254 non-null  object
 3   SequenceNumber            107254 non-null  object
 4   TotalLength               107254 non-null  float64
 5   SourceAddress             0 non-null       float64
 6   TempHumiRange             107254 non-null  float64
 7   TempHumiStatus            107254 non-null  float64
 8   TempHumiEvent             107254 non-null  float64
 9   TempHumiSenVal            107254 non-null  float64
 10  XSenEvent                 107254 non-null  float64
 11  XOAVelocity               107254 non-null  float64
 12  XPeakmg                   107254 non-null  float64
 13  XRMSmg                    107254 non-null  float64
 14  XKurtosis                 107254 non-null  float64
 15  XCrestFactor              107254 non-null  float64
 16  XSkewness                 107254 non-null  float64
 17  XDeviation                107254 non-null  float64
 18  XPeaktoPeakDisplacement   107254 non-null  float64
 19  YSenEvent                 107254 non-null  float64
 20  YOAVelocity               107254 non-null  float64
 21  YPeakmg                   107254 non-null  float64
 22  YRMSmg                    107254 non-null  float64
 23  YKurtosis                 107254 non-null  float64
 24  YCrestFactor              107254 non-null  float64
 25  YSkewness                 107254 non-null  float64
 26  YDeviation                107254 non-null  float64
 27  YPeaktoPeakDisplacement   107254 non-null  float64
 28  ZSenEvent                 107254 non-null  float64
 29  ZOAVelocity               107254 non-null  float64
 30  ZPeakmg                   107254 non-null  float64
 31  ZRMSmg                    107254 non-null  float64
 32  ZKurtosis                 107254 non-null  float64
 33  ZCrestFactor              107254 non-null  float64
 34  ZSkewness                 107254 non-null  float64
 35  ZDeviation                107254 non-null  float64
 36  ZPeaktoPeakDisplacement   107254 non-null  float64
 37  LogIndex                  107254 non-null  float64
 38  DeviceEvents              107254 non-null  float64
 39  DevicePowerSrc            107254 non-null  float64
 40  DeviceBatteryVolt         107254 non-null  float64
 41  DeviceTime                107254 non-null  float64
 42  machine1                  107254 non-null  float64
 43  machine2                  107254 non-null  float64
 44  machine3                  107254 non-null  float64
 45  machine4                  107254 non-null  float64
 46  machine5                  107254 non-null  float64
 47  machine6                  107254 non-null  float64
 48  machine7                  107254 non-null  float64
 49  machine8                  107254 non-null  float64
 50  machine9                  107254 non-null  float64
 51  machine10                 107254 non-null  float64
 52  machine11                 107254 non-null  float64
 53  machine12                 107254 non-null  float64
 54  machine13                 107254 non-null  float64
 55  machine14                 107254 non-null  float64
 56  machine15                 107254 non-null  float64
 57  machine16                 107254 non-null  float64
 58  machine17                 107254 non-null  float64
 59  machine18                 107254 non-null  float64
 60  machine19                 107254 non-null  float64
 61  machine20                 107254 non-null  float64
 62  machine21                 107254 non-null  float64
dtypes: float64(59), int64(1), object(3)
memory usage: 51.6+ MB
```

Preparing Machine 2 Dataset including Pure & Noisy Data

Data Filtering

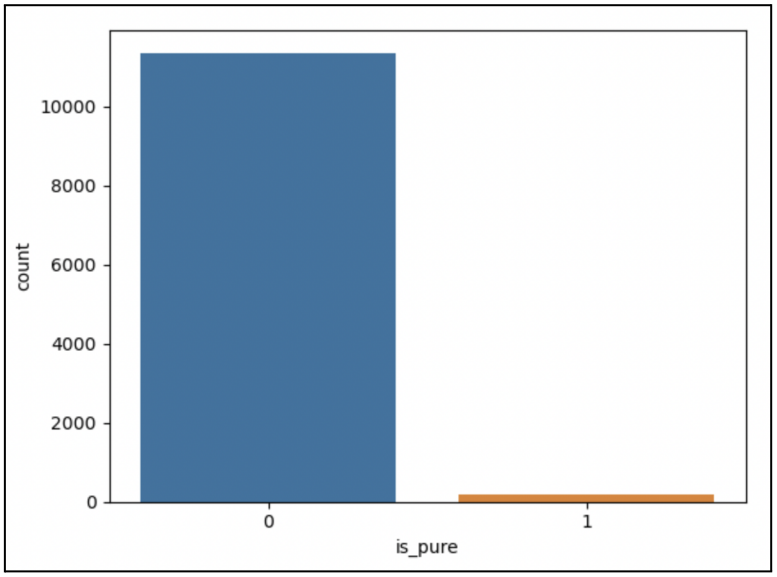
We initiated the data processing by isolating data when machine 2 was in the ‘on’ mode, specifically when machine 2 was equal to 1. This selection was made to focus exclusively on the periods when machine 2 was operational, as this is crucial for understanding its behavior. Data, where machine 2 is off (with machine 2 as 0) or in other states (with machine 2 as -1/NA), were intentionally omitted because they weren't relevant to our analysis.

The resulting dataset's vibration details may include noise from all other 20 machines. Therefore, we attempted to classify each data point as either a pure data point or a noisy data point. In this context, a pure data point signifies that it contains only the vibration data from machine 2, whereas noisy data points have been influenced by the vibration of other machines as well.

Filtering for Pure or Noisy Data

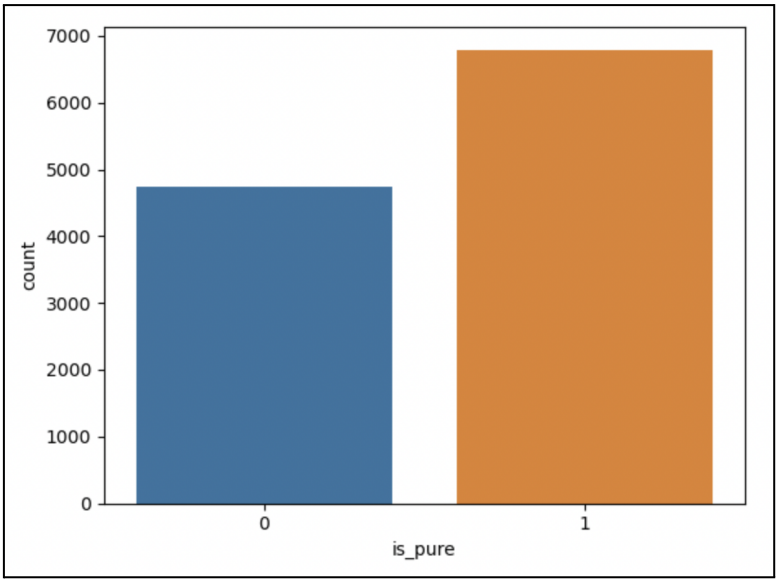
A critical aspect of this analysis was identifying noisy data points. We defined ‘noisy’ as data points where any machine status except machine 2 was either 1 or -1. These values were considered indicative of potential noises in the vibration data. We determined whether this noisy

condition was present for each row and created a binary ‘is_pure’ column, where 0 indicated noisy data and 1 indicated pure/non-noisy data. As a result, we got very few pure data points.



If we assume that the vibration data contains noise from all 20 other machines, which makes it difficult to separate and analyze the vibrations specifically from machine 2. Therefore, we used a different approach with some assumptions. Our approach centered on machines 1 to 5 (m1 - m5) because we assumed that they were physically close to machine 2 (m2) in the industrial setup. We assumed that when m2 operates, its vibrations might be mostly affected by the neighboring machines (m1, m3 - m5) rather than far away machines 6 to 21. And, to monitor these five machines collectively, they used one sensor (sensor 1) as it was capable of capturing data from all of them (m1 - m5) simultaneously. We believed that sensor 1 would be the most informative source because it was monitoring the machines closest to m2, making it an ideal choice for assessing how m2's captured vibration details might be influenced by its nearby machines.

Therefore, we defined ‘noisy’ as data points where the status of any nearby machine to m2 was either 1 or -1.



To enhance the clarity of the dataset and prepare it for further analysis, we removed unnecessary columns, including the ‘machine1’ to ‘machine21’ columns, which were no longer required since we created a new feature called ‘is_pure’. Additionally, we sorted the dataset by the ‘Timestamp’ column to ensure that the data was ordered chronologically.

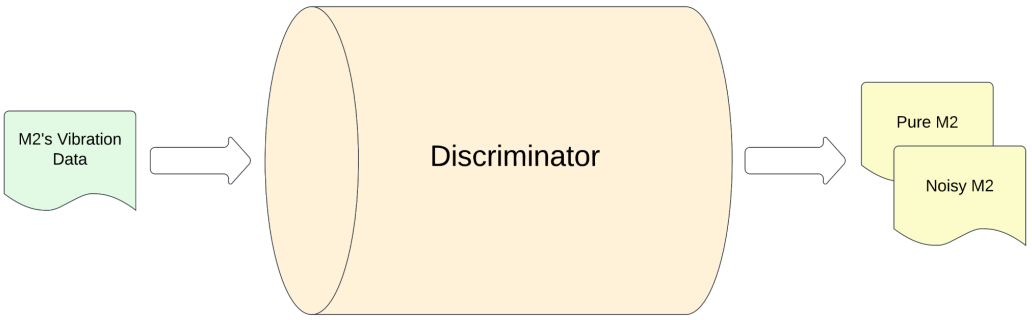
Model Development

Now, we have the vibration data from machine 2, which includes both pure and noisy data. Using a combination of pure and noisy data to train a fault detection algorithm is not ideal. As a solution, we have decided to generate pure data from the noisy data. This generated pure data, when combined with the actual pure data, can be used to train a fault detection algorithm. To generate pure data from noisy data, we have adopted an approach similar to a GAN, which includes a discriminator and a generator. Therefore, the development of this model can be divided into three main parts: the development of the discriminator, the generator, and the fault detection algorithm.

Discriminator

In our project, a discriminator has been meticulously developed to fulfill a critical role in our methodology. It will be used to train the generator by penalizing it for failing to generate pure data from the provided noisy data. This approach allows us to obtain an effective generator capable of removing noise from M2’s noisy vibrational data. Therefore, the accuracy of the generator is highly contingent on the discriminator's performance.

The primary objective of this discriminator is to classify M2’s vibrational data as either ‘pure’ or ‘noisy’. To train the discriminator, we can utilize a dataset that has been labeled as pure or noisy in previous steps.



Our approach involves training the discriminator using advanced machine learning techniques. During the training process, the discriminator learns to differentiate between pure and noisy data based on specific features and patterns within the vibration signals. The representation of one input sample (which has shape (3, 8)) is shown below.

XOAVelocity	XPeakmg	XRMSmg	XKurtosis	XCrestFactor	XSkewness	XDeviation	XPeaktoPeakDisplacement
YOAVelocity	YPeakmg	YRMSmg	YKurtosis	YCrestFactor	YSkewness	YDeviation	YPeaktoPeakDisplacement

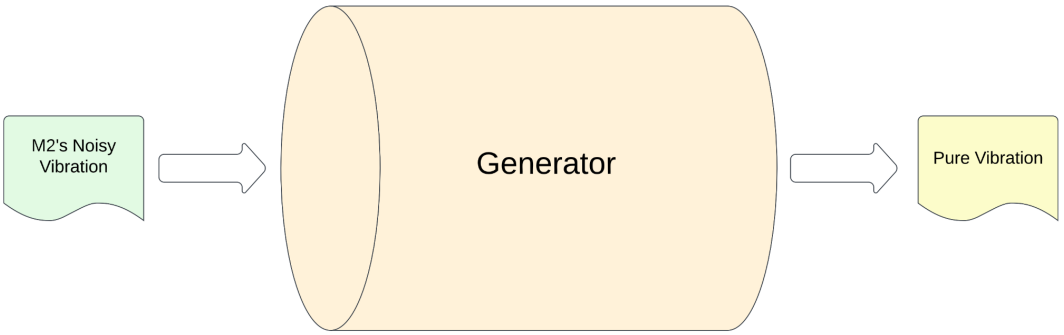
ZOAVelocity	ZPeakmg	ZRMSmg	ZKurtosis	ZCrestFactor	ZSkewness	ZDeviation	ZPeaktoPeakDisplacement

Architecture of Discriminator:

```
Discriminator(
  (discriminator): Sequential(
    (0): Conv1d(3, 24, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): ReLU()
    (2): Conv1d(24, 8, kernel_size=(3,), stride=(1,), padding=(1,))
    (3): ReLU()
    (4): MaxPool1d(kernel_size=4, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): Flatten(start_dim=1, end_dim=-1)
    (6): Linear(in_features=24, out_features=8, bias=True)
    (7): ReLU()
    (8): Linear(in_features=8, out_features=3, bias=True)
    (9): ReLU()
    (10): Linear(in_features=3, out_features=1, bias=True)
    (11): Sigmoid()
  )
)
```

Generator

As mentioned earlier, an effective generator capable of removing noise from M2's noisy vibrational data. The generator served as a transformative tool for M2's noisy vibrational data. Our goal is generating M2's pure vibration from M2's noisy vibration.



Our approach involved training the generator to perform the transformation task. During this training, we penalized the generator for its failure to generate pure data from the provided noisy data. To implement this penalization, we utilized the already trained discriminator. Simultaneously, to minimize deviation from the original noisy data, we also applied a penalty based on the reconstruction error.

Cost = α * [-log(reconstruction_is_pure).mean()] + β * MSE(reconstructed_data, original_data)

The representation of its one input sample is similar to that of the discriminator's representation.

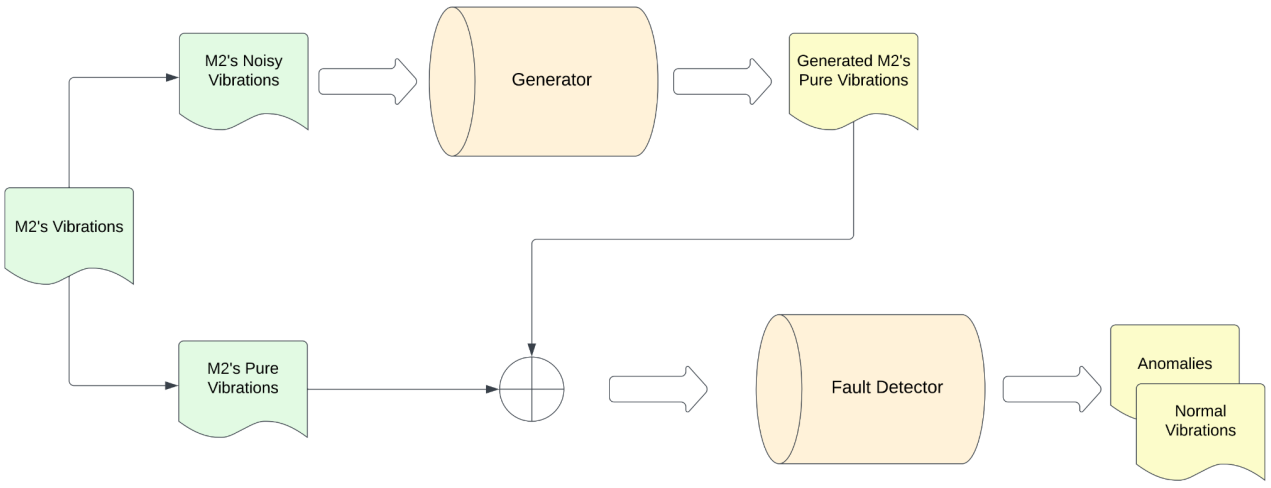
Architecture of Generator:

```
Generator(
  (encoder): Sequential(
    (0): Conv1d(3, 24, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): LeakyReLU(negative_slope=0.2)
    (2): Conv1d(24, 16, kernel_size=(3,), stride=(1,), padding=(1,))
    (3): LeakyReLU(negative_slope=0.2)
    (4): Conv1d(16, 8, kernel_size=(3,), stride=(1,), padding=(1,))
    (5): LeakyReLU(negative_slope=0.2)
    (6): Flatten(start_dim=1, end_dim=-1)
    (7): Linear(in_features=64, out_features=3, bias=True)
    (8): LeakyReLU(negative_slope=0.2)
  )
  (decoder): Sequential(
    (0): Linear(in_features=3, out_features=8, bias=True)
    (1): LeakyReLU(negative_slope=0.2)
    (2): Linear(in_features=8, out_features=16, bias=True)
    (3): LeakyReLU(negative_slope=0.2)
    (4): Linear(in_features=16, out_features=24, bias=True)
    (5): ReLU()
  )
)
```

With the generator's assistance, we achieved a substantial improvement in data quality. It effectively transformed unreliable, noisy information into data that could be trusted. This outcome was vital because pure data is essential for informed decision-making and a wide range of analytical purposes.

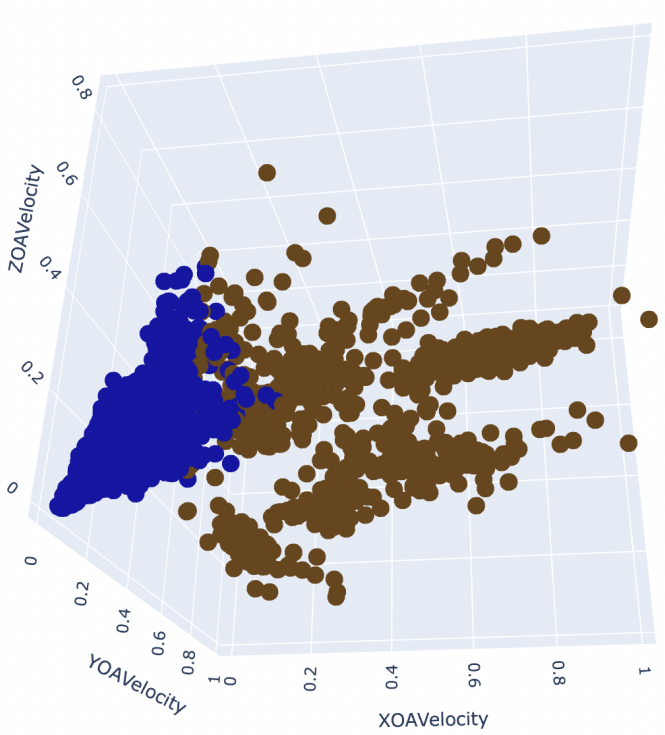
Machine Fault Detection

The final part of this project is driven by the core principle of unsupervised anomaly detection. The dataset we work with lacks predefined labels for normal or anomalous data points, making unsupervised methods a natural choice. Among different unsupervised techniques, we thoughtfully selected three specific approaches, each for its unique strengths and capabilities, to address different facets of machine fault detection.

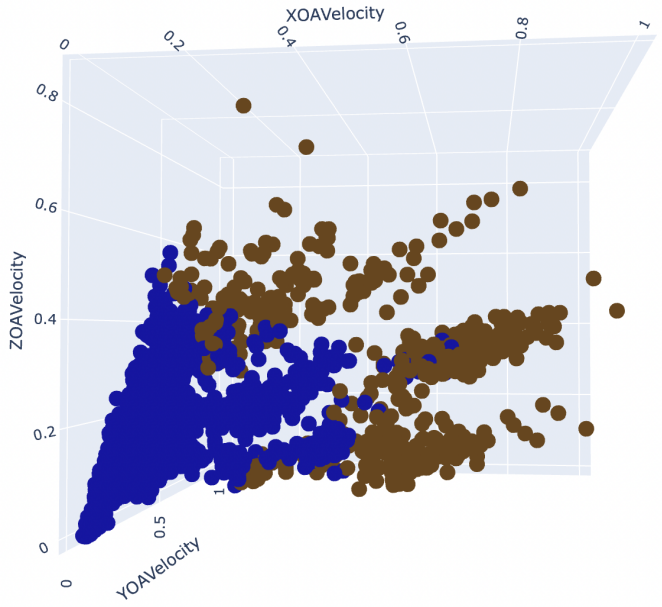


We initiated our exploration with K-Means clustering, an unsupervised method. K-Means excels at grouping similar data points together without the need for predefined labels, making it ideal for our dataset that lacks such labels. We used the power of K-Means to categorize the data into two clusters, distinguishing between “faulty” and “non-faulty” states. Visualizing these clusters

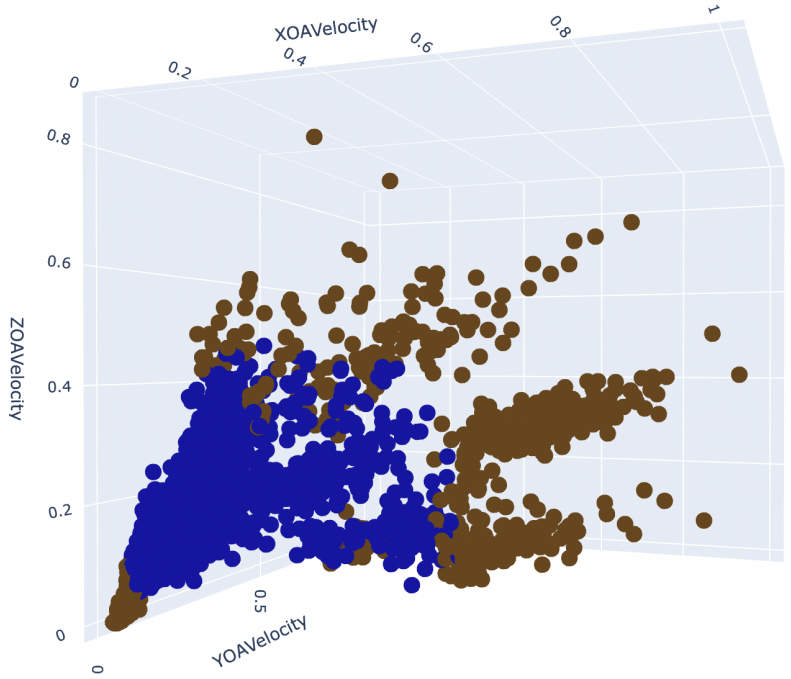
in a 3D scatter plot allowed us to identify patterns in the data, enabling a deeper understanding of the machine's operational condition.



After that, we tried the Isolation Forest method as our next approach. This unsupervised technique stood out as an excellent choice for detecting anomalies, which are data points that behave differently without the need for predefined labels. We selected the Isolation Forest because it excels in identifying anomalies that could signal early machine faults. Visualizing this in a 3D scatter plot made it straightforward to see irregular behavior and potential problems.



Eventually, we arrived at the One-Class Support Vector Machine (SVM), another unsupervised method. As the Isolation forest, this method is also excellent at spotting strange behavior in our data without needing labels to guide it. Unusual behavior, particularly when it signifies abrupt machine faults, can be significantly disruptive. One-class SVM emerged as a compelling choice due to its effectiveness in identifying these irregularities in the data.



Algorithm	Silhouette Score
K-Means clustering	0.6954547999165038
Isolation Forest	0.6290147240351824
One-Class SVM	0.43722974102782125

Data Security & Ethical Considerations

Data Security

When dealing with machine data, the sensorIDs have been ignored. And as the analyzers we are not allowed to share the data with anyone and we have ensured that by signing a nondisclosure agreement.

Ethical Considerations

In our perspective, the machine fault prediction model designed for machine 2 (M2) will be developed with a strong commitment to adhering to all relevant constraints and ethical considerations.

Transparency is a cornerstone of our approach. We recognize that transparent data handling and model operation build trust and enhance collaboration. By providing clear and understandable explanations of our model's operations to all, we are fulfilling ethical obligations. Fairness in machine fault prediction for M2 means that the model should provide equitable predictions for M2 and other machines in the industrial setting. We understand that fair and equitable predictions are not only ethically necessary but also essential for business sustainability. To ensure fairness, We carefully curate and preprocess training data to eliminate bias. Also choose algorithms known for fairness and non-discrimination.

Protecting the privacy of machine-specific data, especially that of M2, is one of our top priorities. Our model will be designed in a way that preserves the confidentiality of proprietary information about M2's configurations and operations. We restrict data access and ensure sensitive details are disclosed only when necessary. We recognize the importance of data ownership and the responsibilities that come with it. We will ensure that data is handled in strict compliance with these contractual obligations, respecting the interests of the data provider. This ensures that data ownership rights are upheld.

Business Impact Analysis

Predictive maintenance approaches can save businesses money by avoiding costly emergency repairs, optimizing maintenance plans and increasing productivity and profitability. By scheduling repairs during pre-arranged downtime, businesses can limit the impact on output and reduce the need for costly emergency repairs.

Financial Savings

Avoiding unplanned downtime is one of the most significant financial benefits. Downtime can lead to lost production. Predictive maintenance helps in optimizing maintenance schedules,

reducing costly emergency repairs, and extending the lifespan of machinery. Additionally, by replacing components before they fail, we can avoid costly secondary damages that may occur if a faulty component is not addressed in time.

Operational Efficacy

By leveraging predictive maintenance, the factory can move from a reactive maintenance approach to a proactive one. This means that maintenance activities can be scheduled minimizing unplanned downtimes. The early detection of potential faults through vibration data allows for timely intervention, reducing the risk of failures and ensuring continuous operation of the machinery.

Data-Driven Decision-Making

The analysis of vibration data and implementing predictive maintenance are inherently data-driven approaches. This allows for continuous improvement based on historical data and real-time data.

We use data analysis to identify potential issues before they occur, reducing downtime and increasing productivity and profitability.