Fault Detection in Factory Machines

190346A - Laksika T

190287R - Kajanan S

190277L - Jayaweera IGHI

190408R - Nanayakkara NKPI

Group - 03

Introduction

- Predictive maintenance is vital in industry, preventing machine faults, minimizing downtime, and enhancing safety.
- It's used in aviation, manufacturing, energy, transportation, and healthcare, ensuring efficiency and safety.
- We aims to ensure continuous machinery operation and reduce risks associated with unplanned failures.
- **Objective:** Create a specialized anomaly detection model for vibration data to improve operational efficiency and resource allocation.

1

Data Collection and Organization to Create a Combined Dataset

Introduction to Dataset

- Two distinct datasets:
  - Vibration dataset: Contains vibration measurements of M2 recorded from 4 sensors. [It might contain noise of other 20 machines]
  - Machine status dataset: Used 3 sensors to capture 21 machines’ status at 1 min time interval. [Sensor 1 - M1 to M5, Sensor 2 - M6 to M13, Sensor 3 - M14 to M21]
- Organized on a weekly basis, with each week comprising two essential data sources.
- Vibration measurements:
  - "OAVelocity", "Peakmg", "RMSmg", "Kurtosis", "CrestFactor", "Skewness", "Deviation", "PeaktoPeakDisplacement" are in X, Y, and Z directions.
- Expectation: Earlier Fault Detection for Machine 2

Dataset Preprocessing

Handling Missing Values

Duplicate Management

Enhance Data Interpretation

Dataset Preprocessing

- **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. To ensure data integrity, only the initial non-duplicate entries were retained.
- **Enhance data interpretation:** We transformed the ‘Timestamp’ column into a human-readable ‘DateTime’ format, ‘`%Y-%m-%d %H:%M:%S`’.

Merging the Vibration Dataset with Machine Status Dataset

```
mapped_df = (
    pd.merge_asof(
        pd.merge_asof(
            pd.merge_asof(
                vibration_df, sensor1, on="Timestamp", direction="nearest"
            ),
            sensor2, on="Timestamp", direction="nearest"
        ),
        sensor3, on="Timestamp", direction="nearest"
    )
)
```

Merging the Vibration Dataset with Machine Status Dataset

	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

```
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
```

15 min as tolerance

FINAL MERGED DATASET

```
<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
```

2

Exploratory Data Analysis

Data Analysis

- ❏ Analysis was conducted in following approaches.
  - ❏ Null and unique value analysis.
  - ❏ Correlation Analysis
  - ❏ Univariate time series analysis

- ❏ The dataset consisted of vibration data and machine status data.
- ❏ Analysis was performed on both types of data.

Null value Analysis

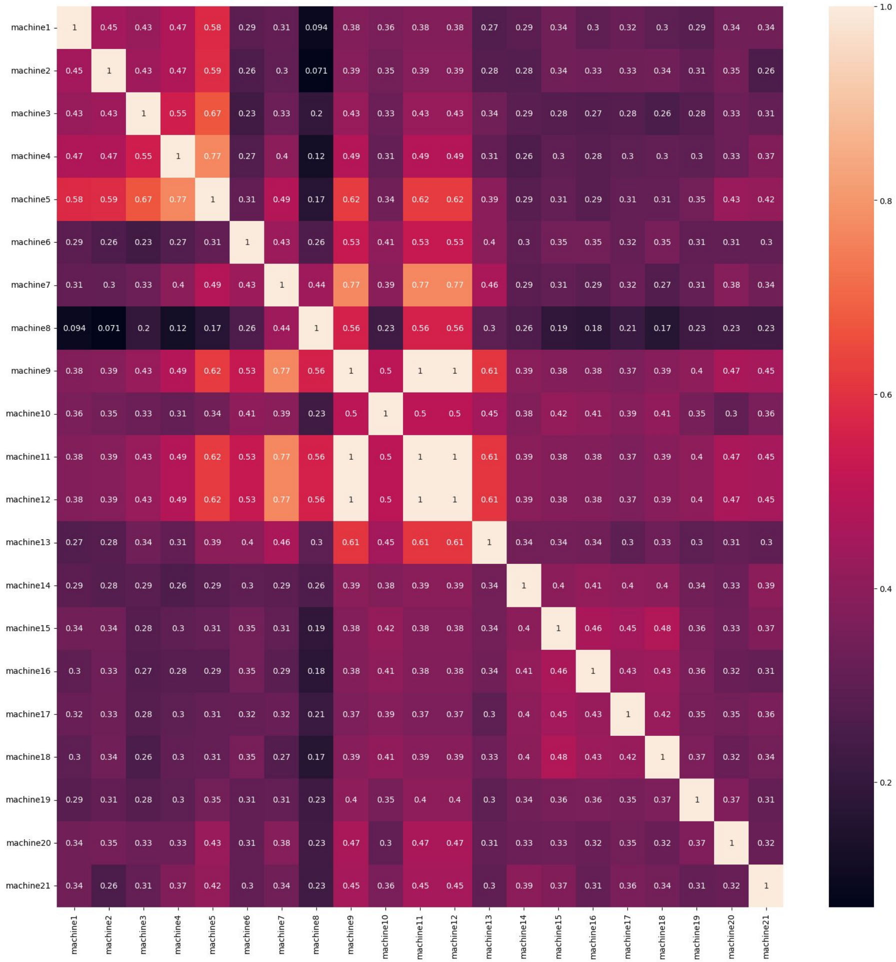
- ❏ Percentage of null values was calculated and the feature ‘SequenceNumber’ was removed due to having significantly higher null value percentage.

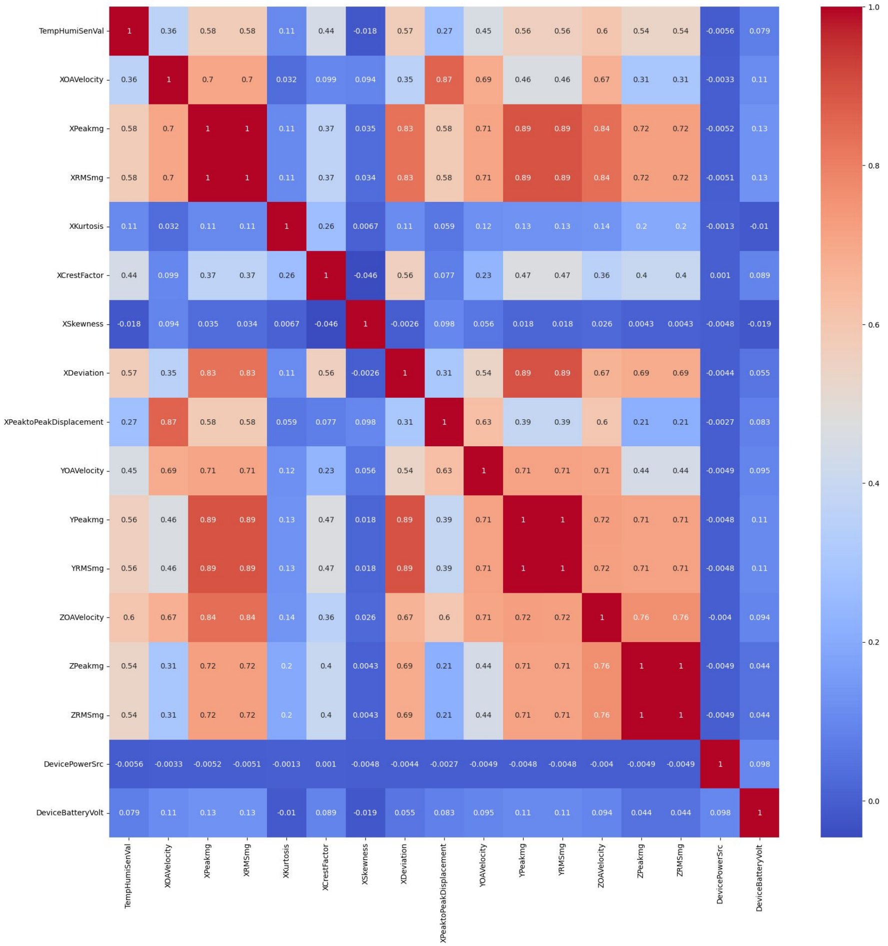
Unique values

- Number of unique values were counted for each feature.
- This is helpful in determining the values which do not vary across the time.
- These values can be neglected when doing time series analysis since there is no change with the time.

Correlation Analysis

- Correlation analysis is important in feature selection for modelling as well.
- Correlation was checked in vibration data and machine status data separately.


machine1
machine2
machine3
machine4
machine5
machine6
machine7
machine8
machine9
machine10
machine11
machine12
machine13
machine14
machine15
machine16
machine17
machine18
machine19
machine20
machine21
1.0
0.8
0.6
0.4
0.2

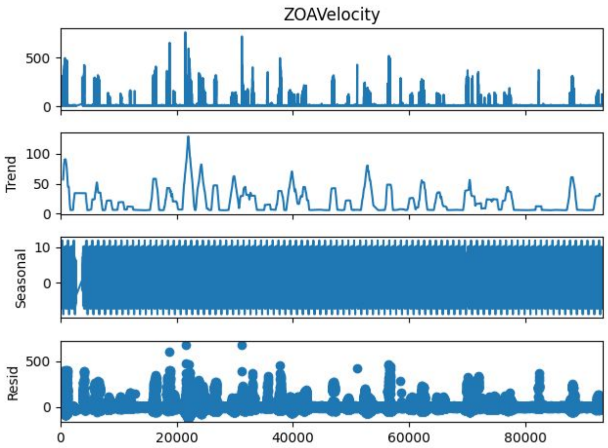

TempHumiSenVal
XOAVelocity
XPeakmg
XRMSmg
XKurtosis
XCrestFactor
XSkewness
XDeviation
XPeaktoPeakDisplacement
YOAVelocity
YPeakmg
YRMSmg
ZOAVelocity
ZPeakmg
ZRMSmg
DevicePowerSrc
DeviceBatteryVolt
1.0
0.8
0.6
0.4
0.2
0.0

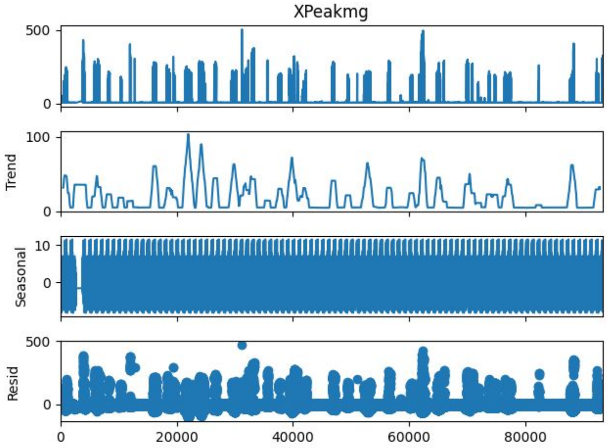
Analysis with machine 2

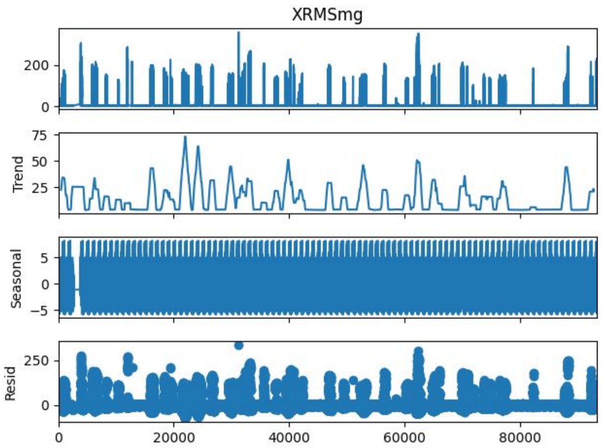
- Different phase of analysis was done taking into account the status of machine 2.
- Status of any machine can be one of,
  - 1 : on
  - 0 : off
  - -1 : unknown
- When considering the correlation between vibration data and status of machine 2 following features showed the highest correlation.
  - ZOAVelocity: 0.630355
  - XPeakmg: 0.592414
  - XRMSmg: 0.592159
  - TempHumiSenVal: 0.58641
  - ZPeakmg:0.585569

Univariate time series analysis

- ❑ The selected set of features were further taken into consideration for univariate time series analysis.
- ❑ Seasonal decomposition was used to decompose the time series data.
- ❑ The intention of this analysis was to observe trend and seasonality after decomposition.
- ❑ There were no significant patterns observed in trend and seasonality.







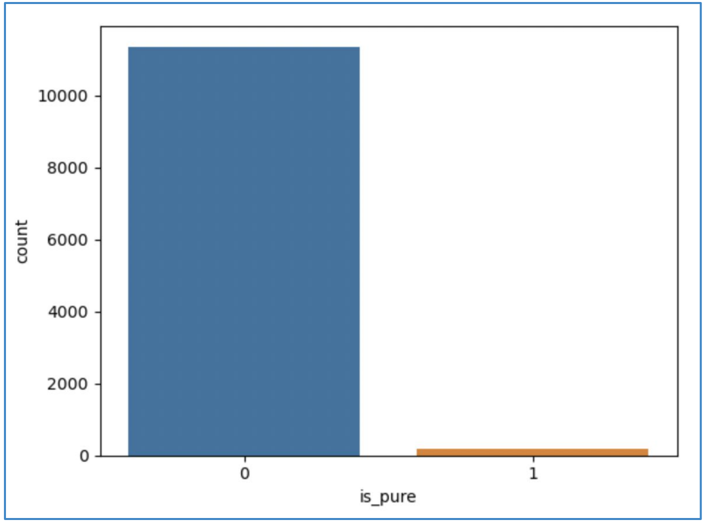
3

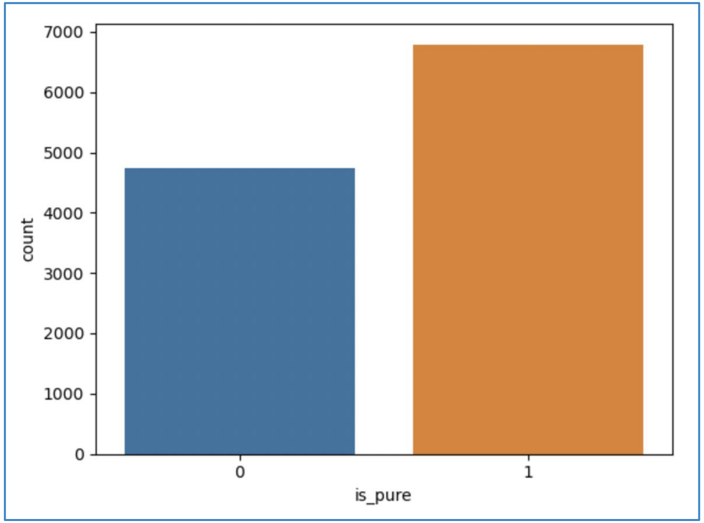
Preparing Machine 2 Dataset including Pure & Noisy Data

Data Filtering

- Our analysis on machine 2
- Machine 2 should be in the “on” (1) mode
- Other rows were omitted
- Differentiated resulting dataset into
  - “pure” (only machine 2's vibrations)
  - “noisy” (affected by other machines)
- Created a binary feature called “is_pure” (0 for noisy, 1 for pure data)

Data Filtering






```
is_pure = (M2 == 1) & (M1, M3 - M21 == 0)
```

```
is_pure = (M2 == 1) & (M1, M3 - M5 == 0)
```

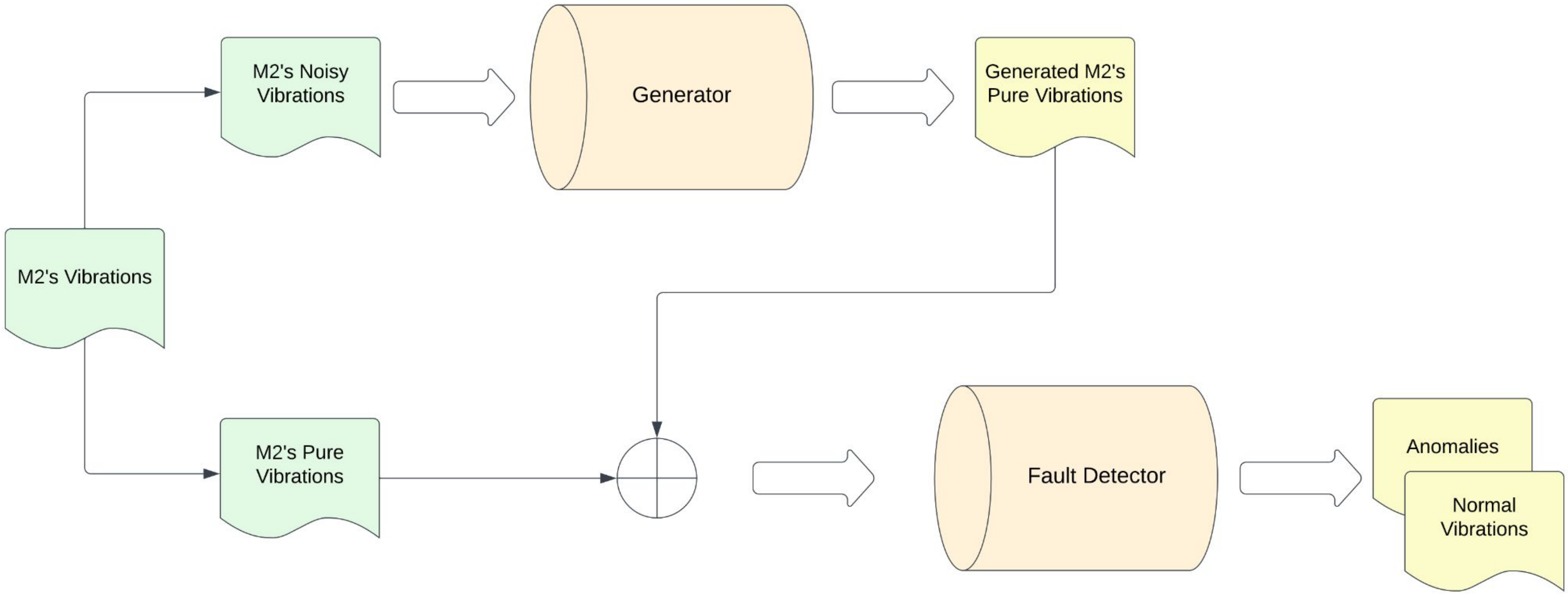
Assumptions on Data Filtering

- Machines 1 to 5 (M1 - M5) are physically close to machine 2 (M2) in the industrial setup. Based on,
  - Serial number
  - Same sensor has been used to capture the status of them
- When M2 operates, its vibrations are **mostly** affected by the neighboring machines (M1, M3 - M5) rather than far away machines 6 to 21.
- `is_pure = (M2 == 1) & (M1, M3 - M5 == 0)`

4

Model Development

Overview of Model Methodology



Overview of Model Methodology

Discriminator

Generator

Fault Detection

Discriminator Development: Penalizer of Generator

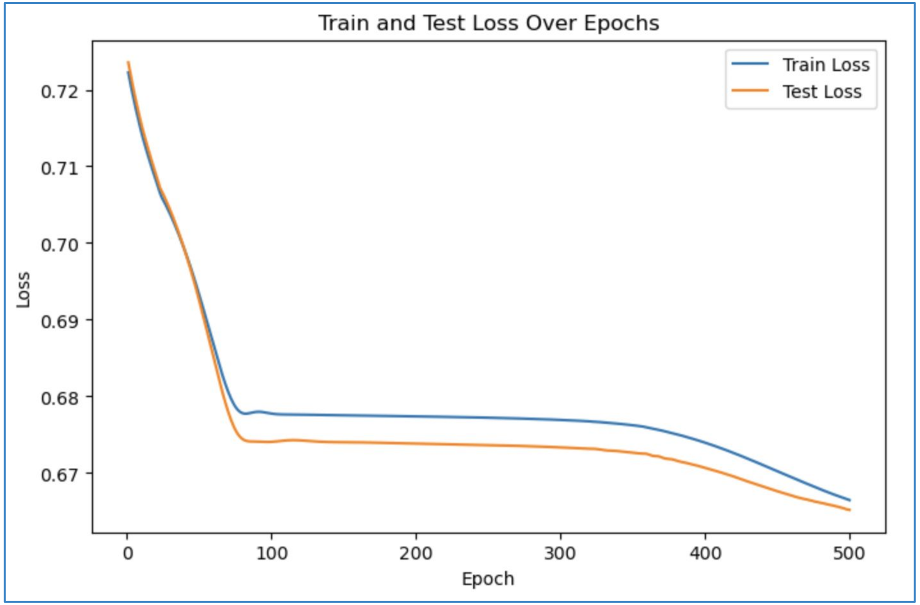
One input sample

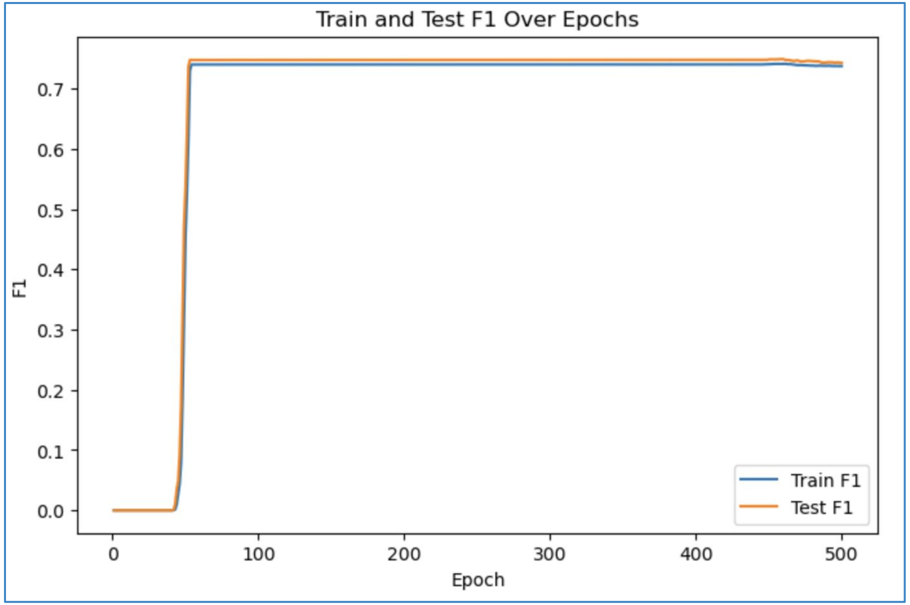
XOAVelocity	XPeakmg	XRMSmg	XKurtosis	XCrestFactor	XSkewness	XDeviation	XPeaktoPeakDisplacement
YOAVelocity	YPeakmg	YRMSmg	YKurtosis	YCrestFactor	YSkewness	YDeviation	YPeaktoPeakDisplacement
ZOAVelocity	ZPeakmg	ZRMSmg	ZKurtosis	ZCrestFactor	ZSkewness	ZDeviation	ZPeaktoPeakDisplacement

Architecture of Discriminator

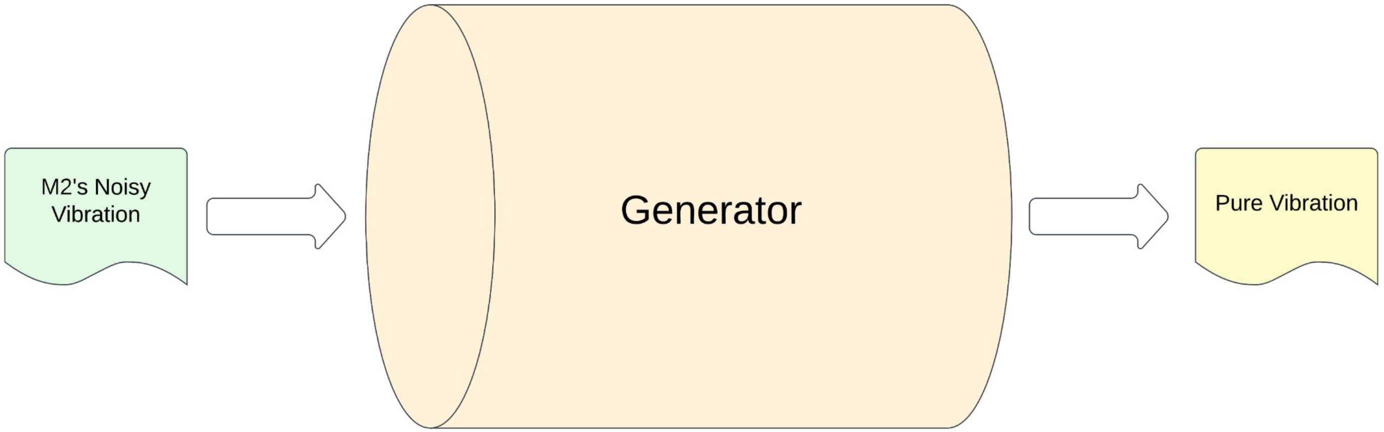
```
Discriminator(
  (discriminator): Sequential(
    (0): Conv1d(3, 24, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): ReLU()
    (2): Conv1d(24, 8, kernel_size=(3,), stride=(1,), padding=(1,))
    (3): ReLU()
    (4): MaxPool1d(kernel_size=4, stride=2, padding=0, dilation=1, ceil_mode=False)
    (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()
  )
)
```

num_epochs = 500, lr = 0.001





Generator Development



One input sample

XOAVelocity	XPeakmg	XRMSmg	XKurtosis	XCrestFactor	XSkewness	XDeviation	XPeaktoPeakDisplacement
YOAVelocity	YPeakmg	YRMSmg	YKurtosis	YCrestFactor	YSkewness	YDeviation	YPeaktoPeakDisplacement
ZOAVelocity	ZPeakmg	ZRMSmg	ZKurtosis	ZCrestFactor	ZSkewness	ZDeviation	ZPeaktoPeakDisplacement

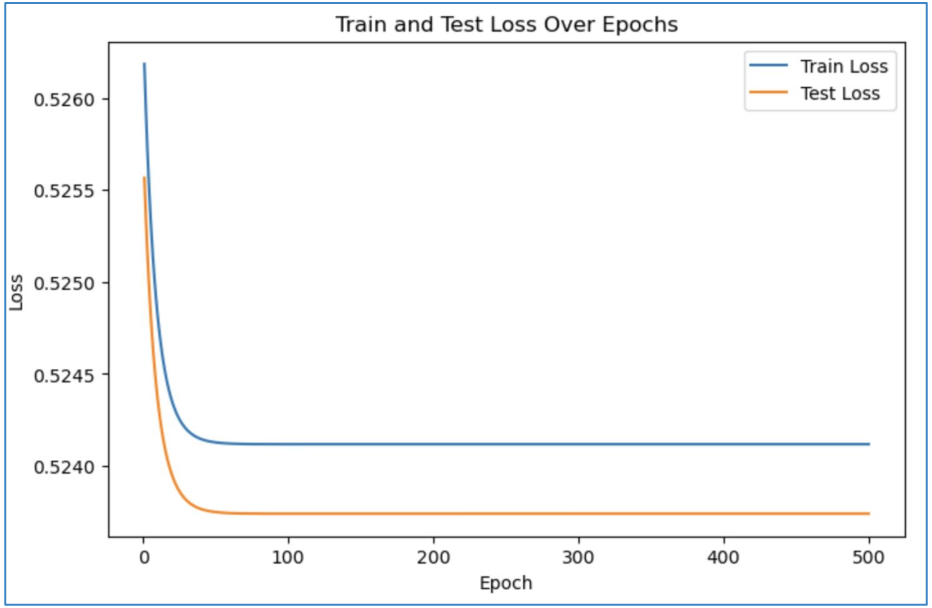
Architecture of Generator

```
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()
  )
)
```

Generator Development

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

num_epochs = 500, lr=0.001, alpha = 1, beta = 2




```python
with torch.no_grad():
    test_X_hat = generator(test_X_tensor)
    test_is_pure = discriminator(test_X_hat)
    test_is_pure_class = test_is_pure.round()

    print(f"mse as reconstruction_error: {generator_cost(test_X_hat, test_X_tensor)}")
    print(f"accuracy_score: {accuracy_score(torch.ones(test_is_pure_class.shape), test_is_pure_class)}")
```

```
mse as reconstruction_error: 0.021645912900567055
accuracy_score: 1.0
```

Fault Detection

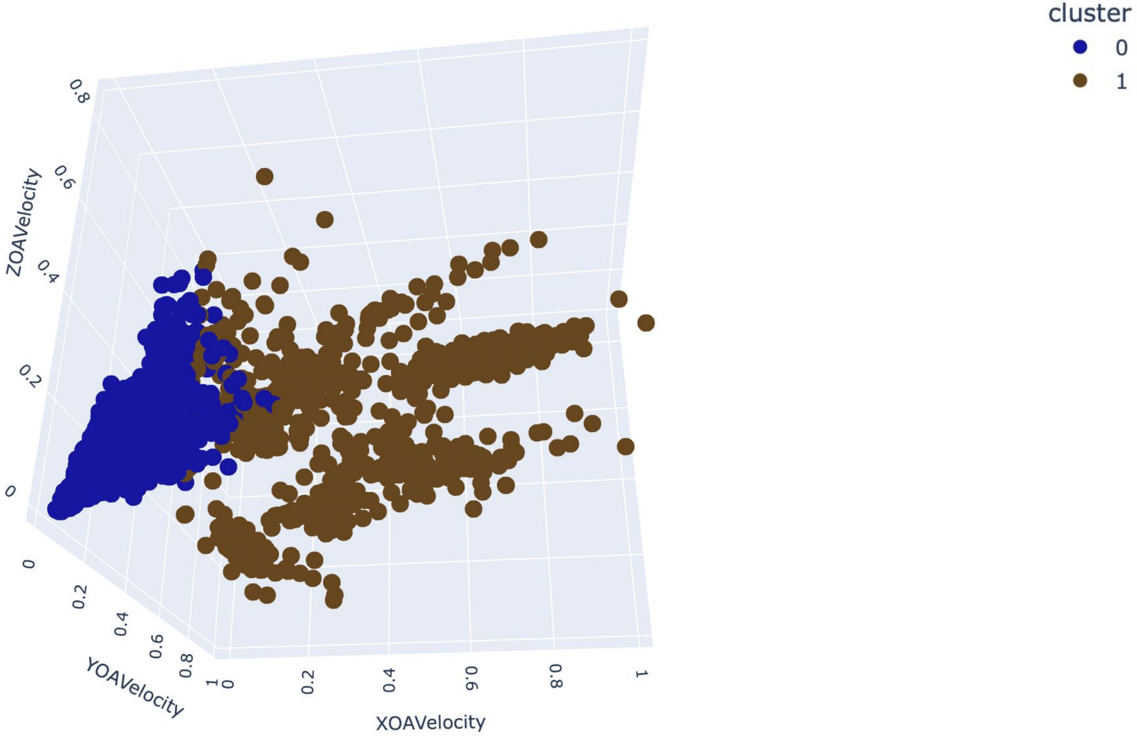
- Our project revolves around the core principle of unsupervised anomaly detection.
- Dataset lacks predefined labels for normal or anomalous data points, making unsupervised methods a natural choice.
- We thoughtfully selected three specific unsupervised approaches:
  - K-means clustering - Clustering approach
  - Isolation Forest
  - One-Class SVM

Anomaly detection approaches

K-Means Clustering: Understanding Faulty States

- K-Means clustering, an unsupervised method, is a crucial component of our approach.
- K-Means' capability allows us to categorize data into "faulty" and "non-faulty" states without the need for predefined labels.
- We used below of them are the selected features,
  - XOAVelocity, YOAVelocity, ZOAVelocity

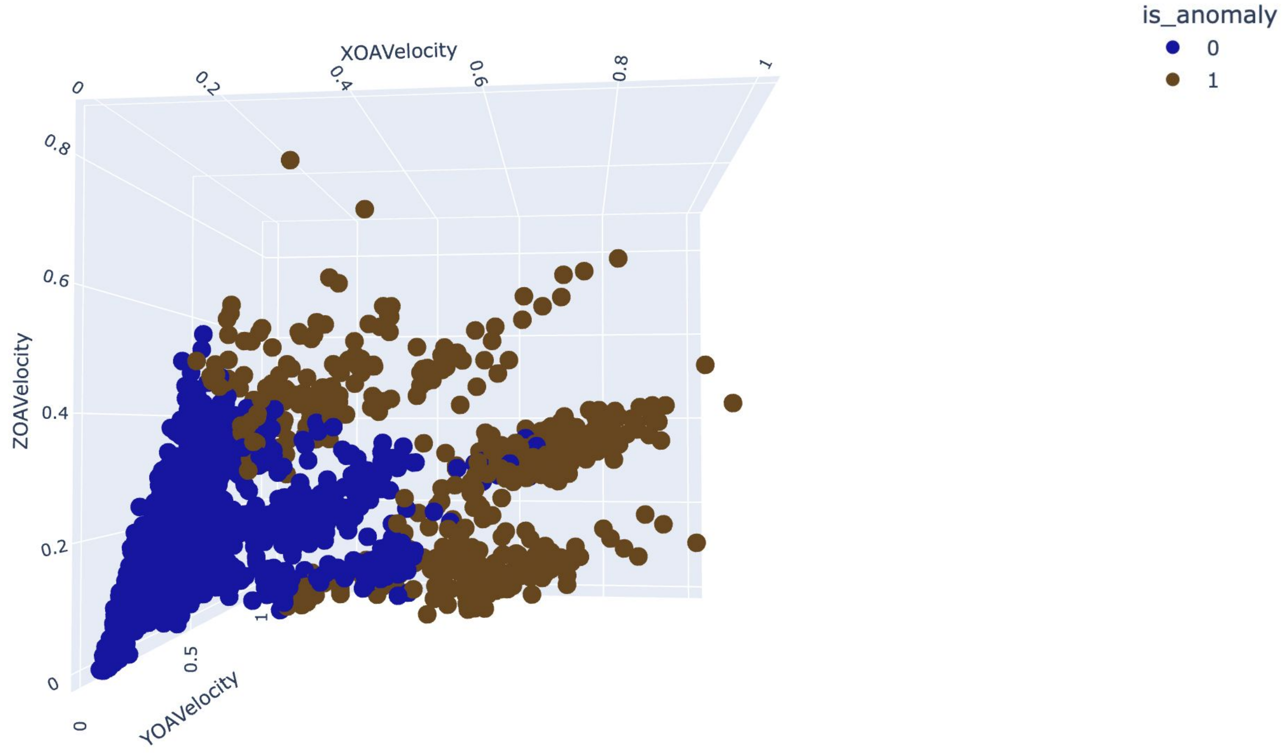
K-Means Clustering



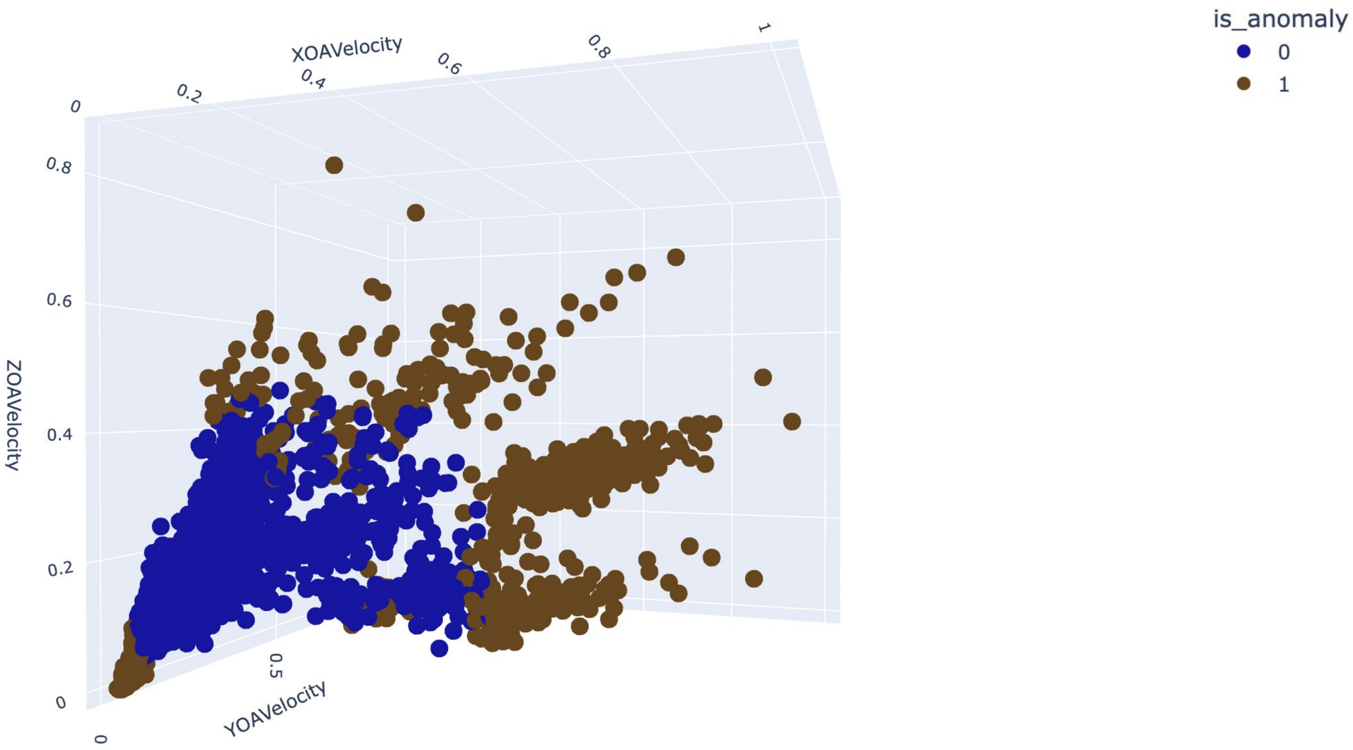
Isolation Forest and One-Class SVM for Anomaly Detection

- We used the Isolation Forest and the One-Class Support Vector Machine (SVM) as our next approaches.
- These unsupervised techniques excel at identifying anomalies without the need for predefined labels, which is vital for detecting early machine faults.
- We used below of them are the selected features,
    - XOAVelocity, YOAVelocity, ZOAVelocity

Isolation Forest



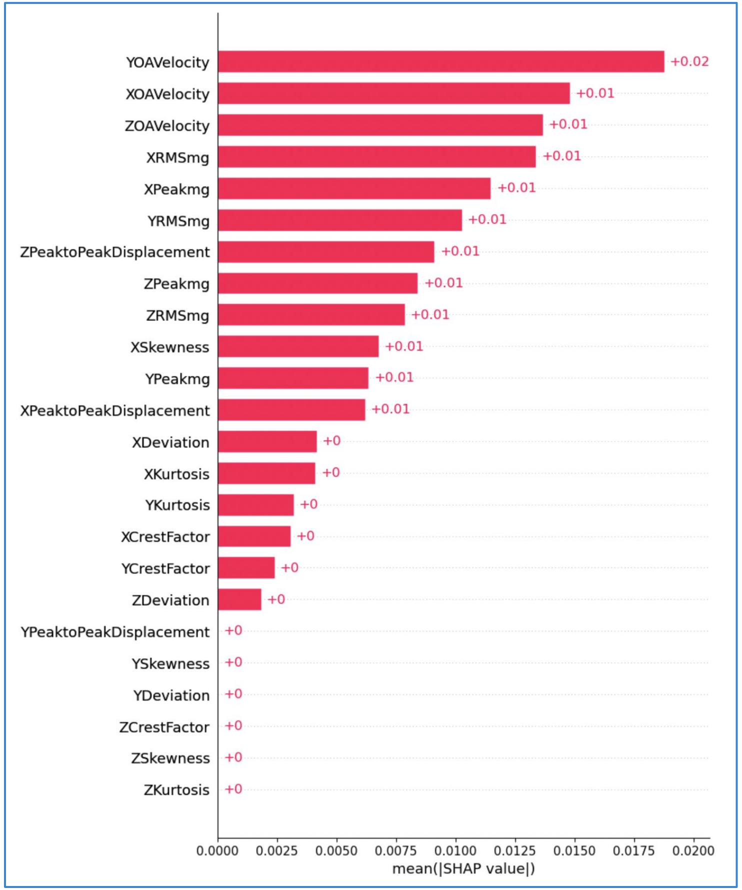
One-Class SVM

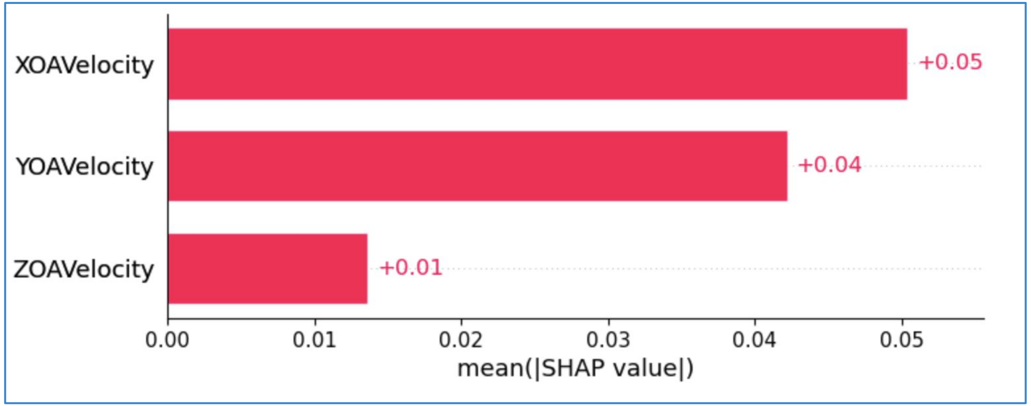


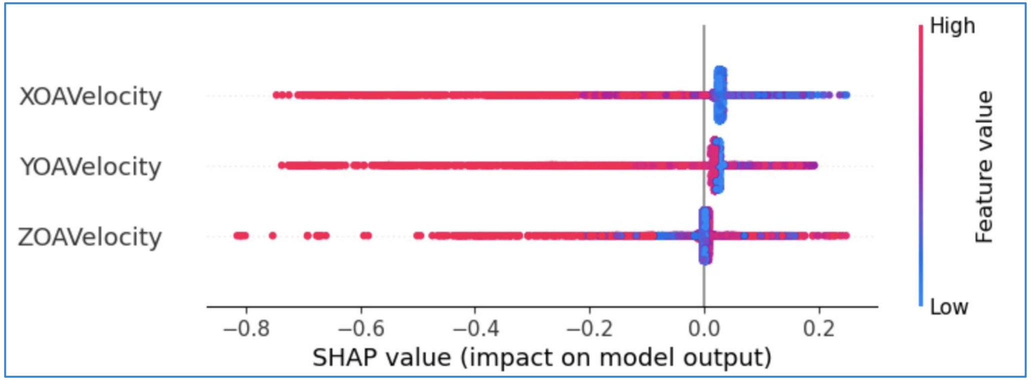
Fault Detection

Algorithm	**Silhouette Score**
K-Means clustering	0.731
Isolation Forest	0.727
One-Class SVM	0.563

Model Explainability - SHAP







Security & Ethical Consideration

- We ensure not to share the data with anyone by signing a nondisclosure agreement.
- The machine learning model built for machine 2 has been adhered to all relevant constraints and ethical considerations.
- Protect the privacy machine data
- Restrict data access and disclose sensitive data only when necessary
- To maintain the fairness of the model, the data has been preprocessed with eliminating the bias.

Business Impact

- Financial Decision Making
- Operational Efficacy
- Data-Driven Decision-Making

References

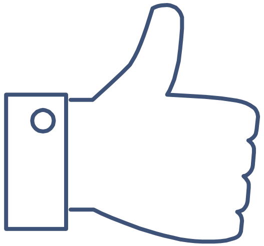
https://youtu.be/ufINa-XGX2M?si=zj3rsVCsiJZMz2-9

https://www.analyticsvidhya.com/blog/2023/01/learning-different-techniques-of-anomaly-detection/

https://www.intellspot.com/anomaly-detection-algorithms/

https://towardsdatascience.com/a-novel-approach-to-feature-importance-shapley-additive-explanations-d18af30fc21b

https://medium.com/analytics-vidhya/anomaly-detection-using-generative-adversarial-networks-gan-ca433f2ac287



THANK YOU!

Any Questions?