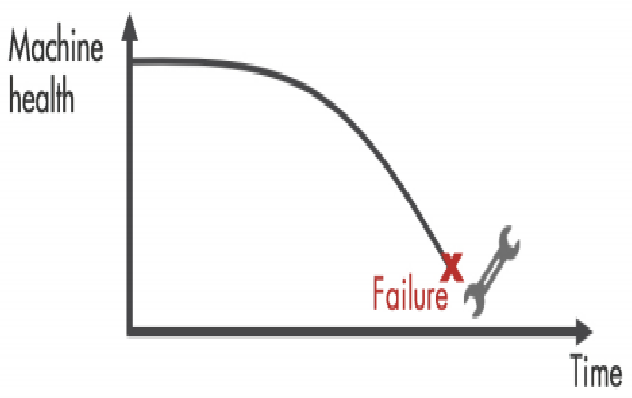
Artificial Intelligence Techniques for Predictive Maintenance

Maintenance

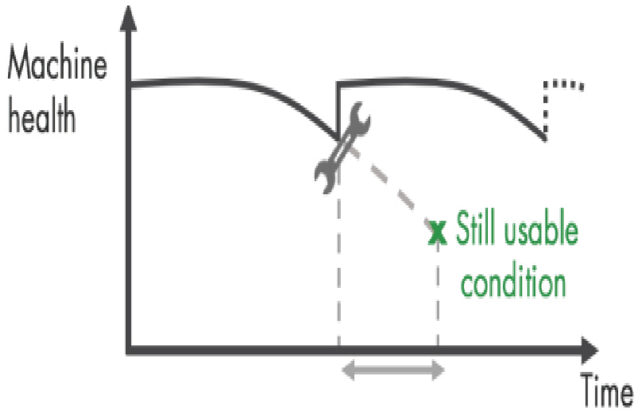
- Maintenance involves functional checks, servicing, repairing or replacing of necessary devices, equipment, machinery, building infrastructure, and supporting utilities in industrial, business, governmental, and residential installations.
- Types
  - Reactive Maintenance
    - Implemented right after a defect has been detected
    - Extremely costly to repair highly damaged parts
  - Preventive Maintenance
    - Catching and fixing problems before they happen
    - Challenge ? – When to do maintenance
    - Scheduling maintenance very early, you’re wasting machine life that is still usable, and this adds to your costs
  - Predictive Maintenance
    - Eliminate unplanned shutdown
    - Estimate time to failure of a machine
    - Find the optimum time to schedule maintenance

Maintenance - Comparison

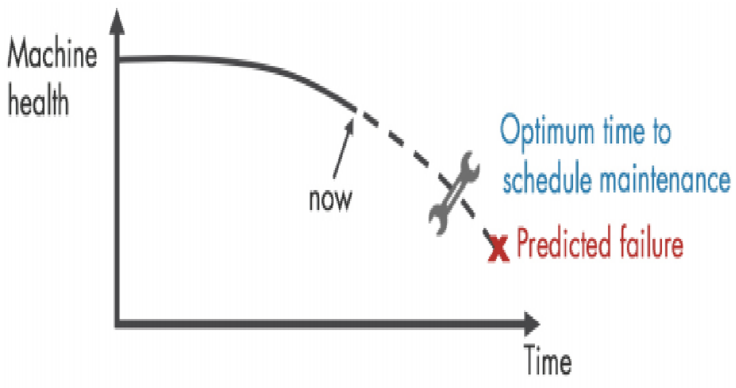
Reactive Maintenance



Preventive Maintenance



Predictive Maintenance



Predictive Maintenance: Traditional & AI - Implementation

- Industrial Equipment are constantly observed via sensors
- Sensors are attached to components of the equipment and feed constant, real-time data to CMMS (Computerized Maintenance Management System) software
- CMMS then interprets this data and warns technicians
- Using AI techniques along with CMMS – We can do following tasks
    - Detect the anomalies
    - Detect failure patterns
    - Provide early warnings

Predictive Maintenance - Advantages

- According to US Department of Energy
  - Return on investment is increased by 10 times
  - Maintenance costs are reduced by 25% to 30%
  - Equipment failure phenomenon is reduced by 70 to 75%
  - Equipment downtime is reduced by 35% to 45%
  - Output has increased by 20% to 25%
- Replace up to 30% of your Preventive Maintenance tasks
- According to Shell, AI-enhanced predictive maintenance can lead to savings of 20% or more on maintenance costs for key systems.

Anomaly Detection

- Anomaly detection is the key step for predictive maintenance
- Anomaly detection is the task of finding observations that do not conform to the normal
- Different Approaches
  - Univariate Anomaly Detection
    - One Class SVM
    - Isolation Forest
    - Local Outlier Factor
  - Multivariate Anomaly Detection
    - LSTM - Autoencoder

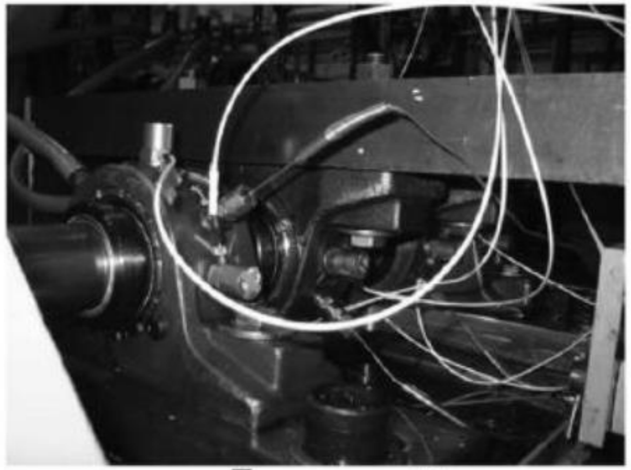
Dataset Description

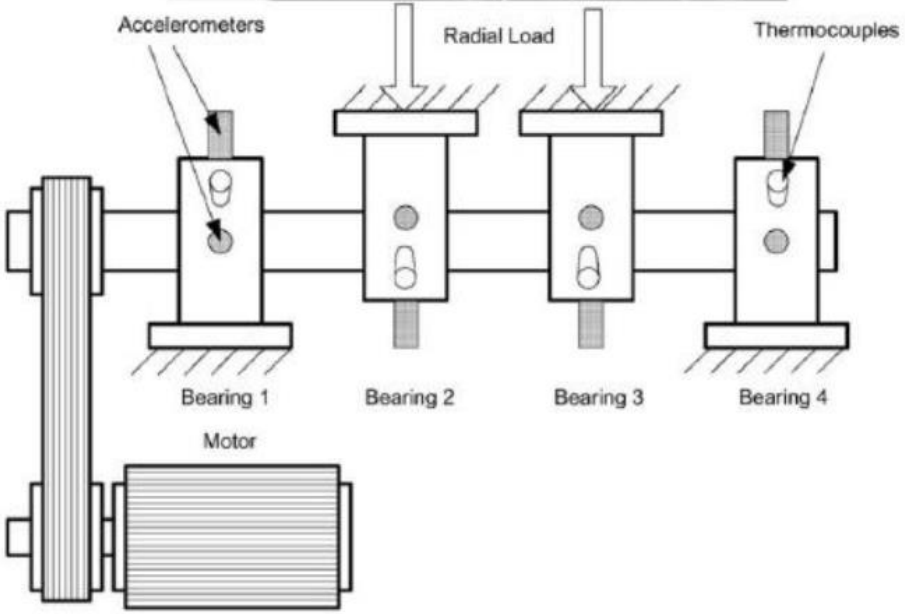
- IMS (Intelligent Maintenance Systems) Bearing Dataset
  - Also known as NASA PCoE (Prognastics Centre of Excellence) Bearing Dataset
  - Link: https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/
- Test Rig Setup
  - Four bearings were installed on a shaft. The rotation speed was kept constant at 2000 RPM by an AC motor coupled to the shaft via rub belts.
  - A radial load of 6000 lbs. is applied onto the shaft and bearing by a spring mechanism. All bearings are force lubricated.

Dataset Description

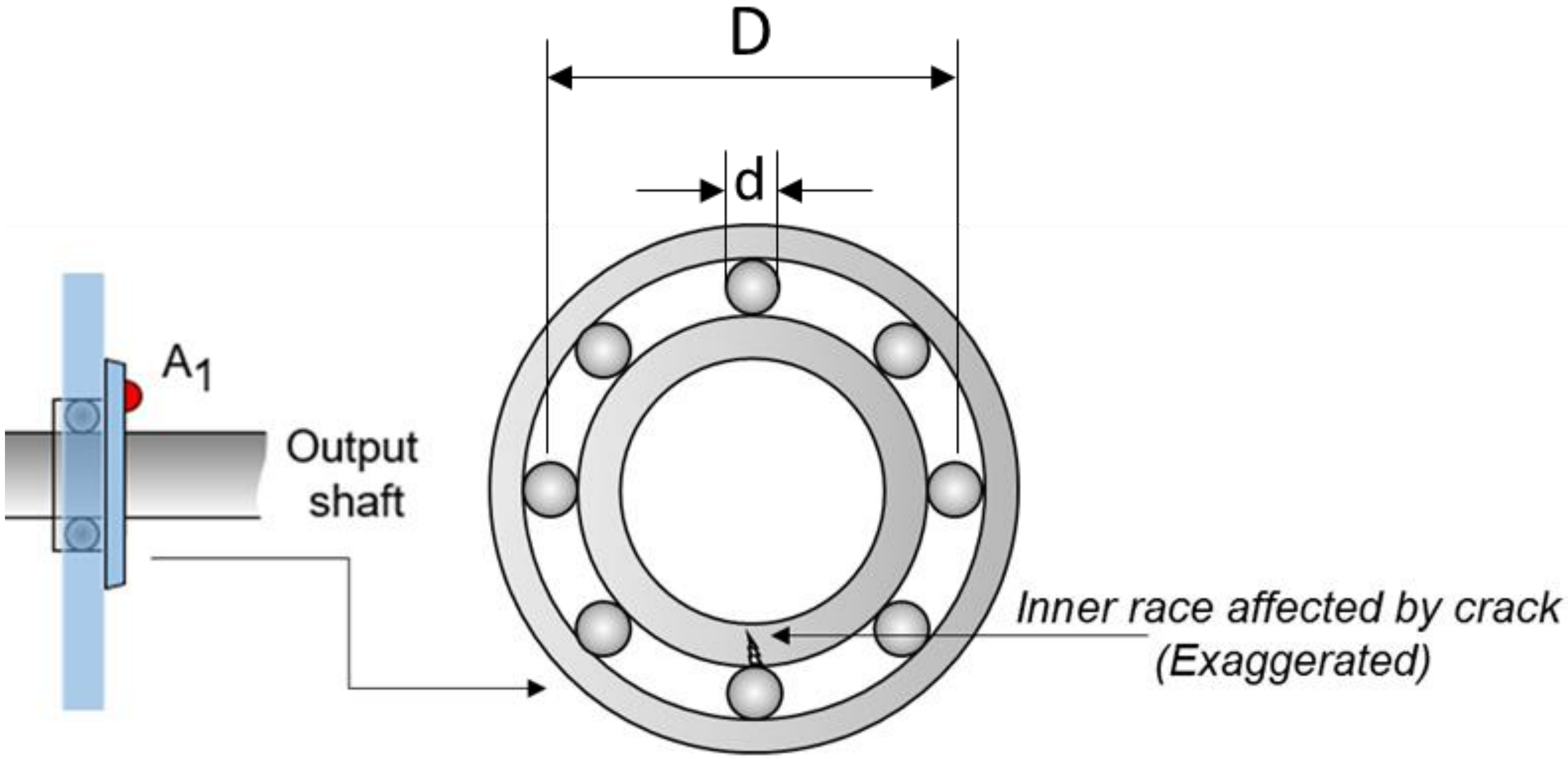
- Bearing : Rexnord ZA-2115 double row bearings
- Accelerometers: PCB 353B33 High Sensitivity Quartz ICP accelerometers
    - Accelerometers are used to sense vibrations
- Dataset describes test to failure experiments
- All failures occurred after exceeding designed life time of the bearing which is more than 100 million revolutions.

Test Rig Setup





Bearing - Image



Ball bearing cross−section (Magnified)

Dataset Description

- Dataset consists of 3 sets of Data. (3 Tests)
- Each data set consists of individual files that are 1-second vibration signal snapshots recorded at specific intervals.
- Each file consists of 20,480 points with the sampling rate set at 20 kHz.

Dataset – Directory Strucutre

Name	Date modified
2004.02.12.10.32.39	23-03-2004 14:12
2004.02.12.10.42.39	23-03-2004 14:12
2004.02.12.10.52.39	23-03-2004 14:12
2004.02.12.11.02.39	23-03-2004 14:12
2004.02.12.11.12.39	23-03-2004 14:12
2004.02.12.11.22.39	23-03-2004 14:12
2004.02.12.11.32.39	23-03-2004 14:12
2004.02.12.11.42.39	23-03-2004 14:12
2004.02.12.11.52.39	23-03-2004 14:12
2004.02.12.12.02.39	23-03-2004 14:12
2004.02.12.12.12.39	23-03-2004 14:12
2004.02.12.12.22.39	23-03-2004 14:12

2004.02.12.10.32.39 - Notepad

File Edit Format View Help

-0.049	-0.071	-0.132	-0.010
-0.042	-0.073	-0.007	-0.105
0.015	0.000	0.007	0.000
-0.051	0.020	-0.002	0.100
-0.107	0.010	0.127	0.054
-0.078	-0.212	0.042	-0.044
-0.020	-0.010	-0.144	-0.007
-0.046	0.112	0.034	0.034
-0.063	-0.154	0.071	0.076
0.068	0.044	-0.029	0.054
0.095	0.022	-0.090	-0.037
-0.007	0.007	-0.024	-0.095
-0.046	0.000	-0.122	-0.059
0.044	-0.002	-0.068	0.027
0.137	0.007	0.054	0.073
0.098	-0.032	0.088	-0.029
0.081	-0.081	-0.090	-0.105

The file name indicates when the data was collected. Each record (row) in the data file is a data point.

	Set No. 1	Set No. 2	Set No. 3
Recording Duration	October 22, 2003 12:06:24 to November 25, 2003 23:39:56	February 12, 2004 10:32:39 to February 19, 2004 06:22:39	March 4, 2004 09:27:46 to April 4, 2004 19:01:57
No. of files	2156	984	4448
Channel Arrangement (Accelerometer) (Ch – Channel)	Bearing 1 – Ch 1&2; Bearing 2 – Ch 3&4; Bearing 3 – Ch 5&6; Bearing 4 – Ch 7&8.	Bearing 1 – Ch 1; Bearing2 – Ch 2; Bearing3 – Ch3; Bearing 4 – Ch 4.	Bearing 1 – Ch 1; Bearing2 – Ch 2; Bearing3 – Ch3; Bearing 4 – Ch 4.
File Recording Interval:	Every 10 minutes		
Description	Inner race defect occurred in bearing 3 and roller element defect in bearing 4.	Outer race failure occurred in **Bearing 1**.	Outer race failure occurred in bearing 3.

Note: In this work, we are considering only **Set No. 2**

Proposed Techniques

- Predicts whether there is a possibility of failure (Anomaly)
- Predicts Remaining Useful Life (RUL)
  - Remaining useful life (RUL) is the length of time a machine is likely to operate before it requires repair or replacement.

Deliverables

- Anomaly Detection
  - Univariate
    - **User Input:** Either RMS Value or Mean Value of 1 sec Vibration Signal of Bearing 1
    - **Output:** Whether that test point is Normal or Anomaly
  - Multivariate
    - **User Input:** Mean Value of 1 sec Vibration Signal of 4 Bearings
    - **Output:** Whether that test point is Normal or Anomaly
- RUL Prediction
    - **User Input:** RMS & Kurtosis value of 1 sec vibration signal of Bearing 1at time t & t-1
    - **Output:** RUL_Class | Fraction Failing | RUL

RUL_Class	Fraction Failing (Range) (%)	RUL (%)
1	0 – 20 %	**80 %**
2	20 – 40 %	**40 %**
3	40 – 60 %	**60 %**
4	60 – 80 %	**20 %**
5	80 – 100 %	**< 20 %**

Project Workflow

Data Collection

- Collection of NASA IMS Bearing Dataset
- Understanding of Experimental setup & how data is collected?

EDA

- Visualization of 1 sec Vibration signal for all 4 Bearings
- Dataset description & statistics

Feature Extraction

- Mean value of 1 sec vibration signal
- RMS value of 1 sec vibration signal
- Kurtosis value of 1 sec vibration signal
- Visualization of all features

Feature Selection

- RMS & Mean value for Anomaly Detection
- RMS & Kurtosis value for Remaining Useful Life (RUL) Prediction

Univariate Anomaly Detection

- Algorithms: One Class SVM, Isolation Forest, Local Outlier Factor
- Feature used: RMS & Mean value
- Training: Train the model only with the normal data, as this data is only available at earlier stage
- Testing: Test the model with both normal and anomaly data

Metrics

- Labelling the data as normal or anomaly using Statistical Process Control ($\mu + 3\sigma$) & Heuristics
- Comparing different univariate anomaly detection using classification metrics
  - Accuracy score
  - Confusion Matrix – True Positive, True Negative, False Positive, False Negative

Contd.

Project Workflow

Multi-variate Anomaly Detection

- Algorithm: LSTM – Autoencoder
- Feature used: Mean value of all 4 Bearings
- Training: Train the model only with the normal data, as this data is only available at earlier stage
- Testing: Test the model with both normal and anomaly data

Metrics

- Reconstruction loss – Mean Absolute Error
- If the reconstruction loss is less than the threshold, then the test point is considered as normal
- If the reconstruction loss is greater than the threshold, then the test point is considered as anomaly
- Threshold is calculated from normal data – reconstruction loss distribution

RUL Calculation

- Weibull distribution is fitted with Shape parameter and Scale parameter
- The Shape parameter and Scale parameter is calculated from RMS value of Bearing 1 (Run to failure test)
- Weibull distribution is used to calculate the RUL (1 – fraction failing)
- Entire life of the bearing is classified into 5 classes / intervals (each interval - 20 % fail/RUL)

RUL Prediction

- Algorithms: k-NN, Decision Tree
- Feature used: RMS & Kurtosis value of 1 sec vibration signal at time t & t-1
- Target: Class 1 (80 % RUL), 2 (60 % RUL), 3 (40 % RUL), 4 (20 % RUL), 5 (<20 % RUL)
- Metrics: Accuracy Score & Confusion Matrix

Choosing Best Models

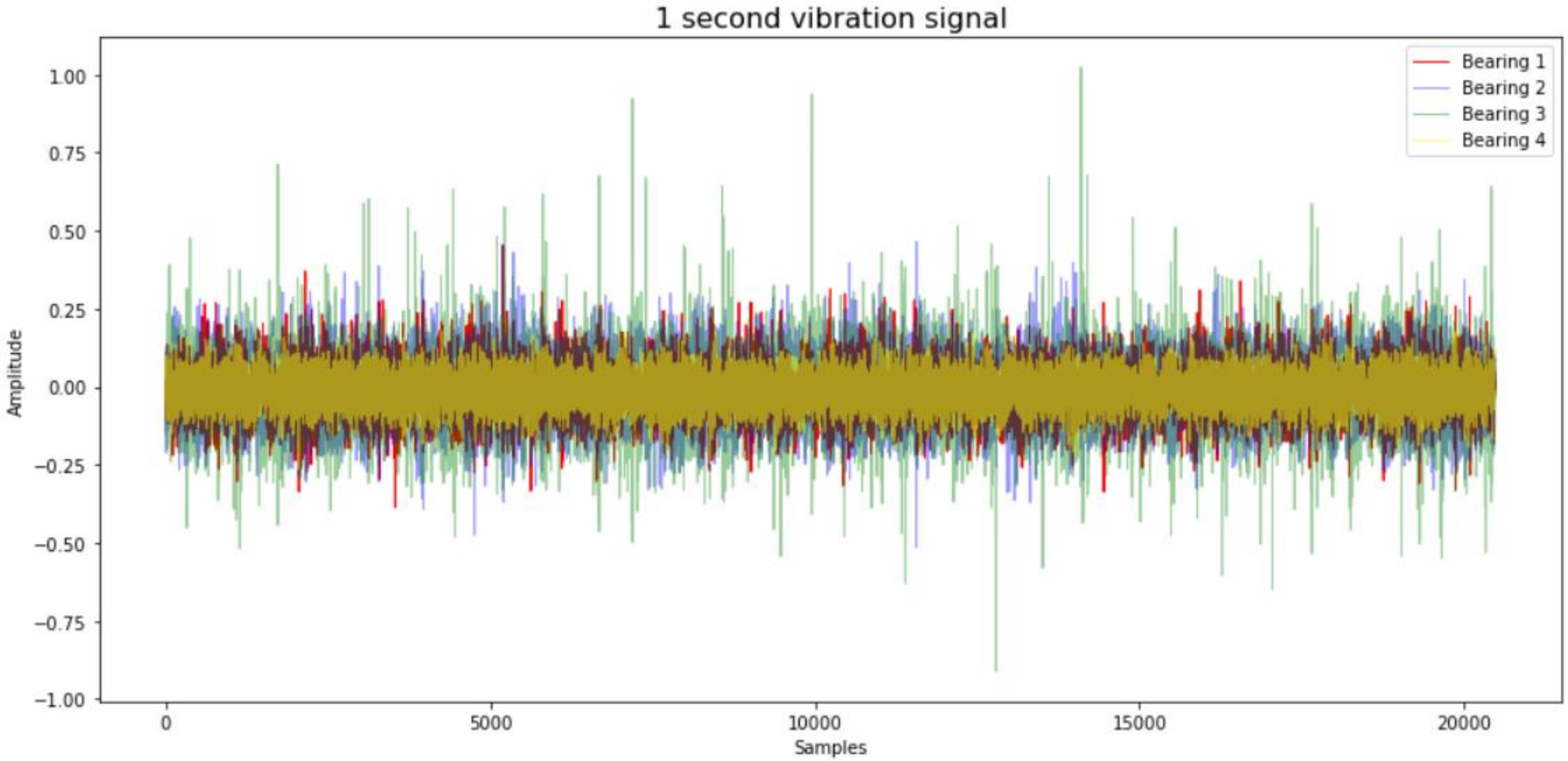
- Univariate Anomaly detection: Local Outlier Factor |feature: RMS value
- Multivariate Anomaly detection: LSTM Autoencoder
- RUL Prediction: Decision Tree Classifier

Model Deployment

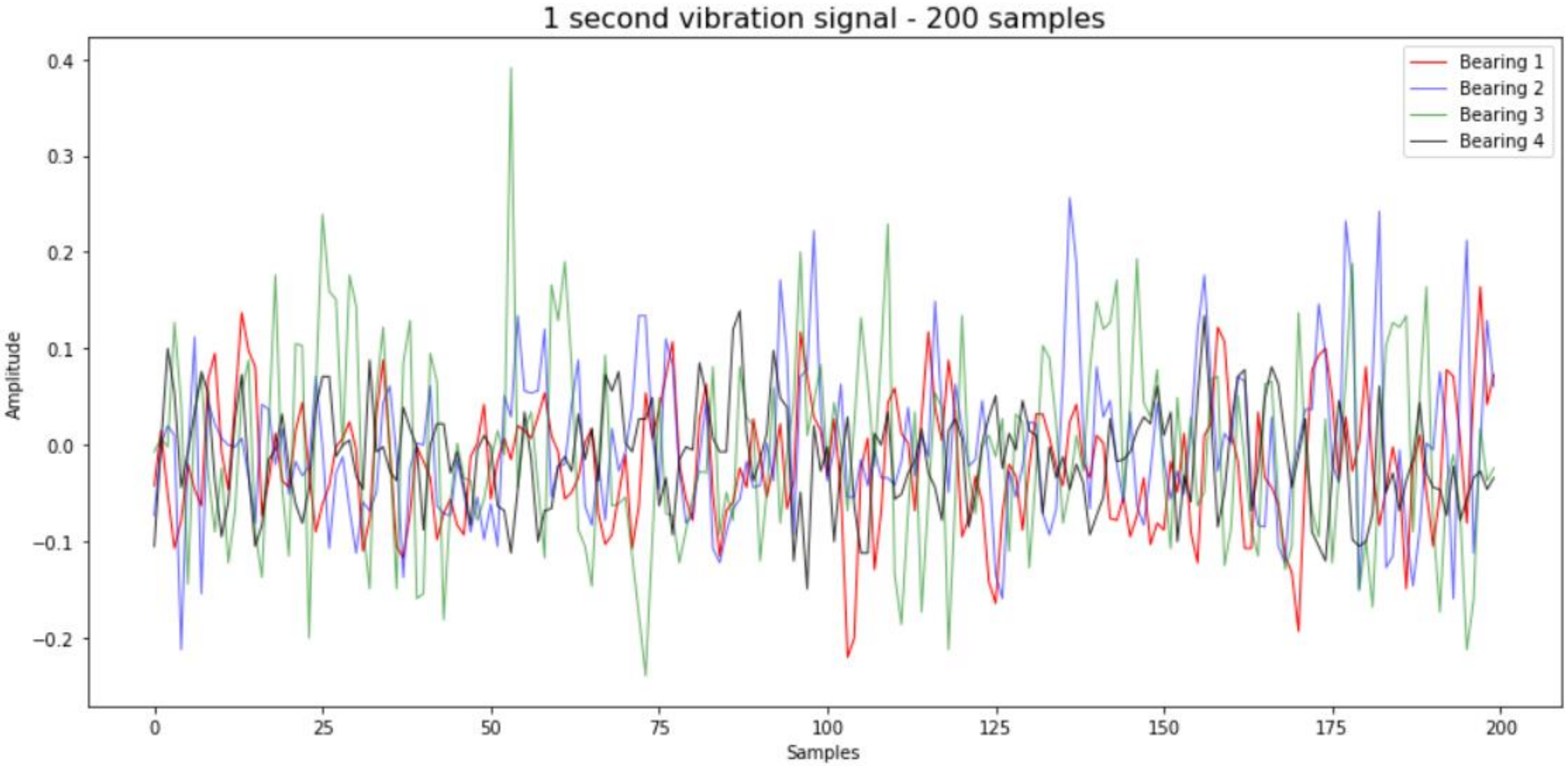
- Saving the best models as pickle / h5 file format
- Model deployment using FLASK-API & Heroku
- Containerization using Docker

Visualizations & Results

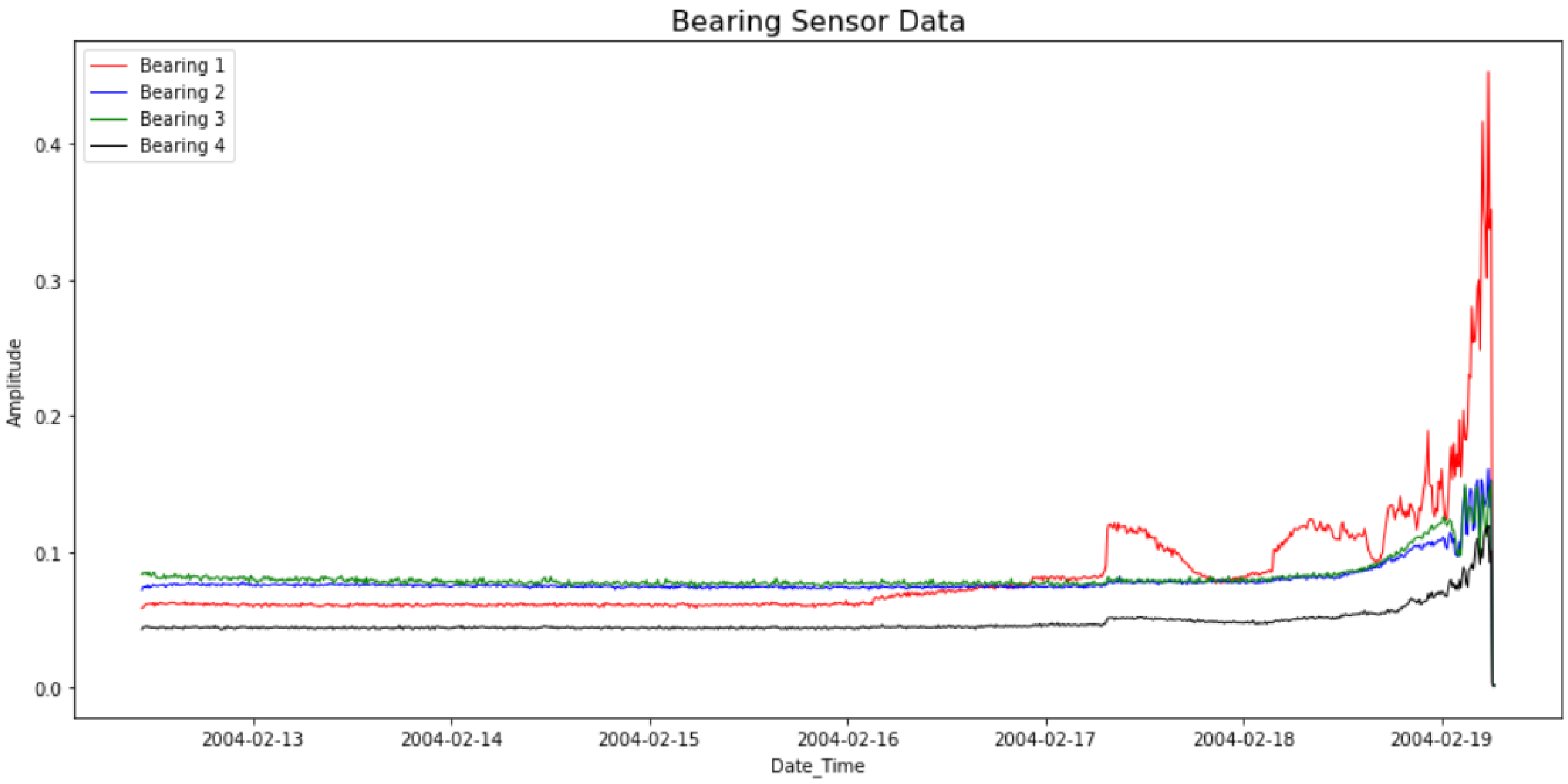
1 sec Vibration Signal – all 4 Bearings



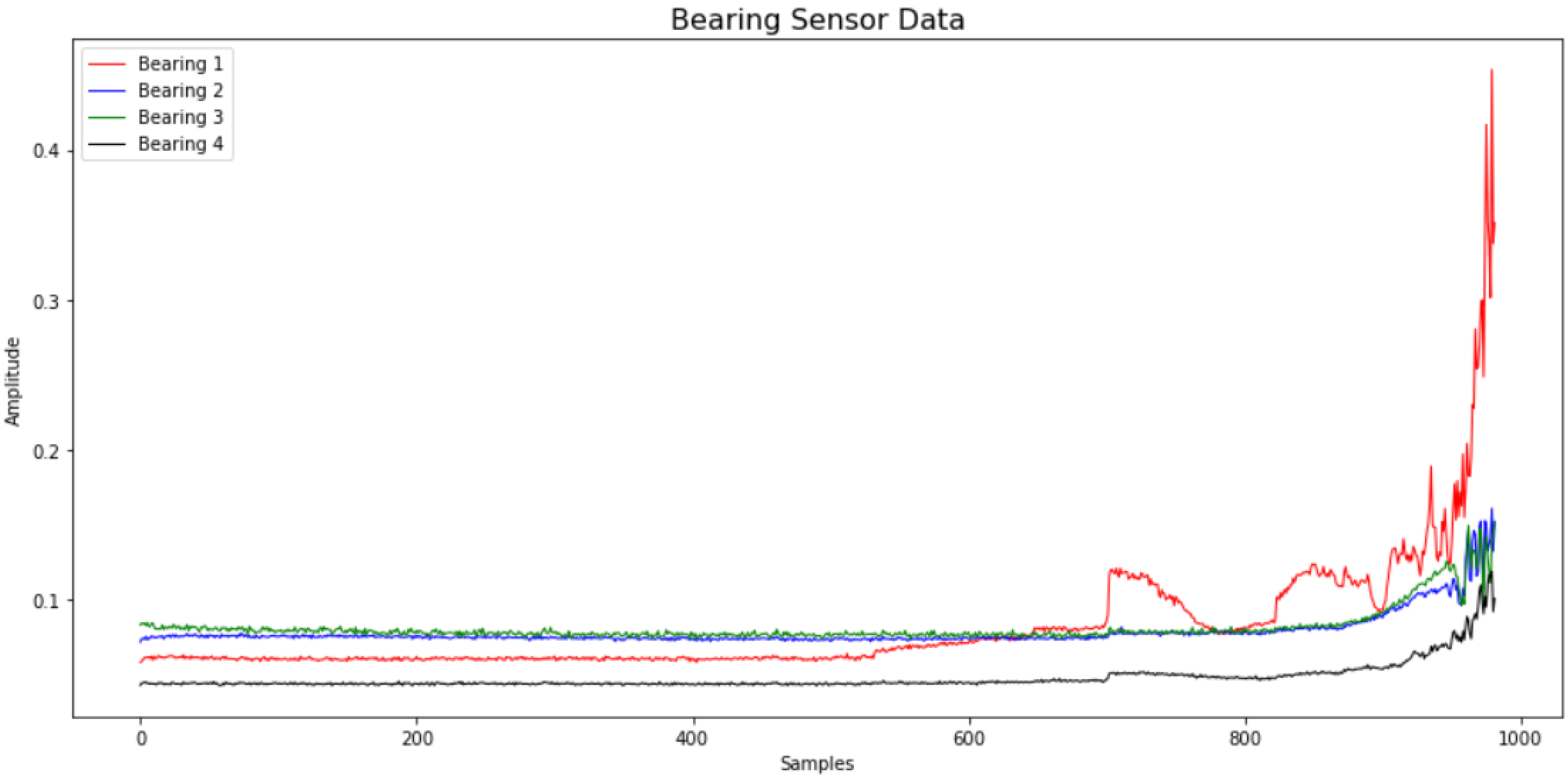
1 sec Vibration Signal – all 4 Bearings – first 200 samples



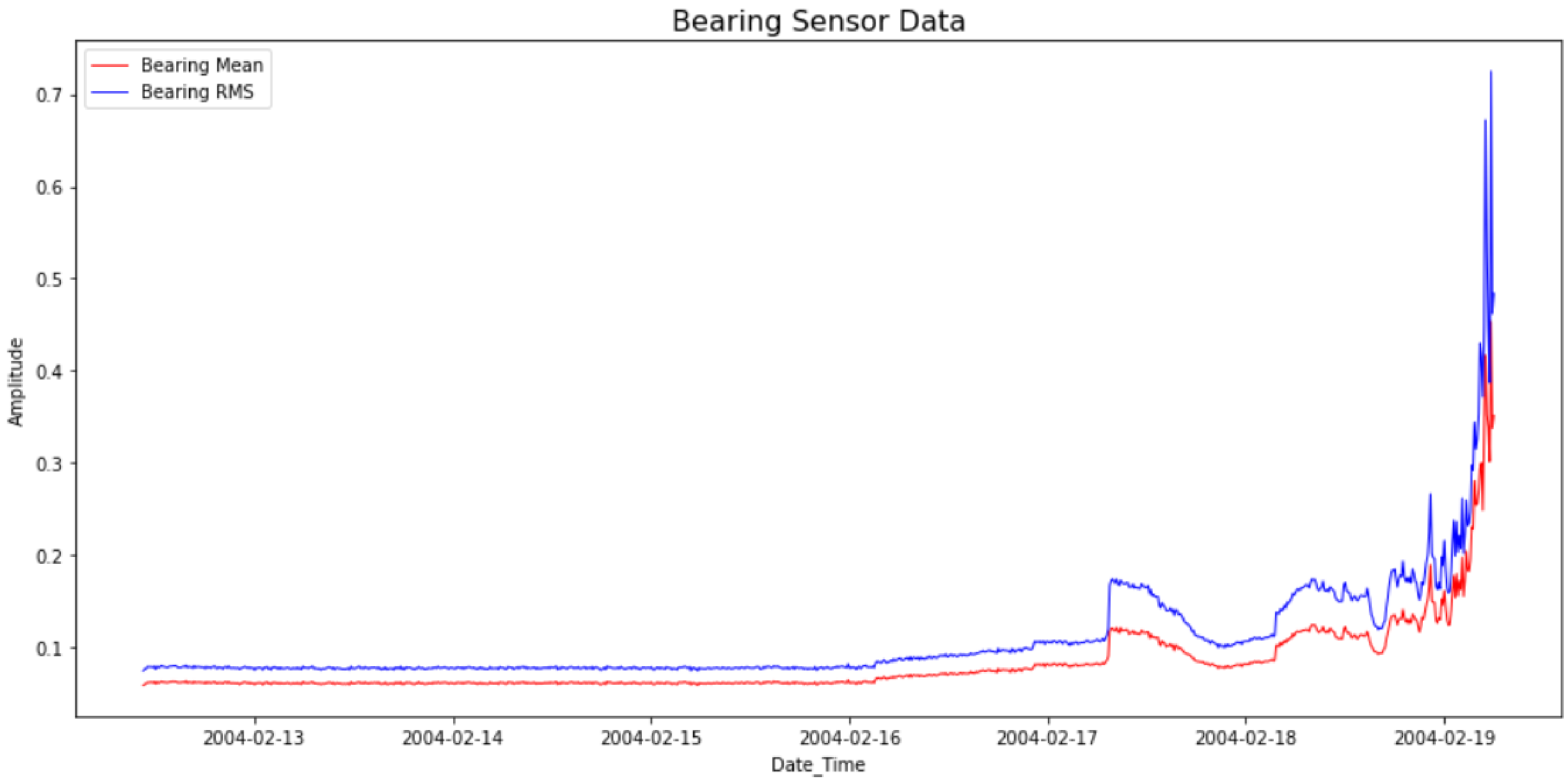
Mean Value of 1 sec vibration:
4 Bearings (Run to Failure Test)



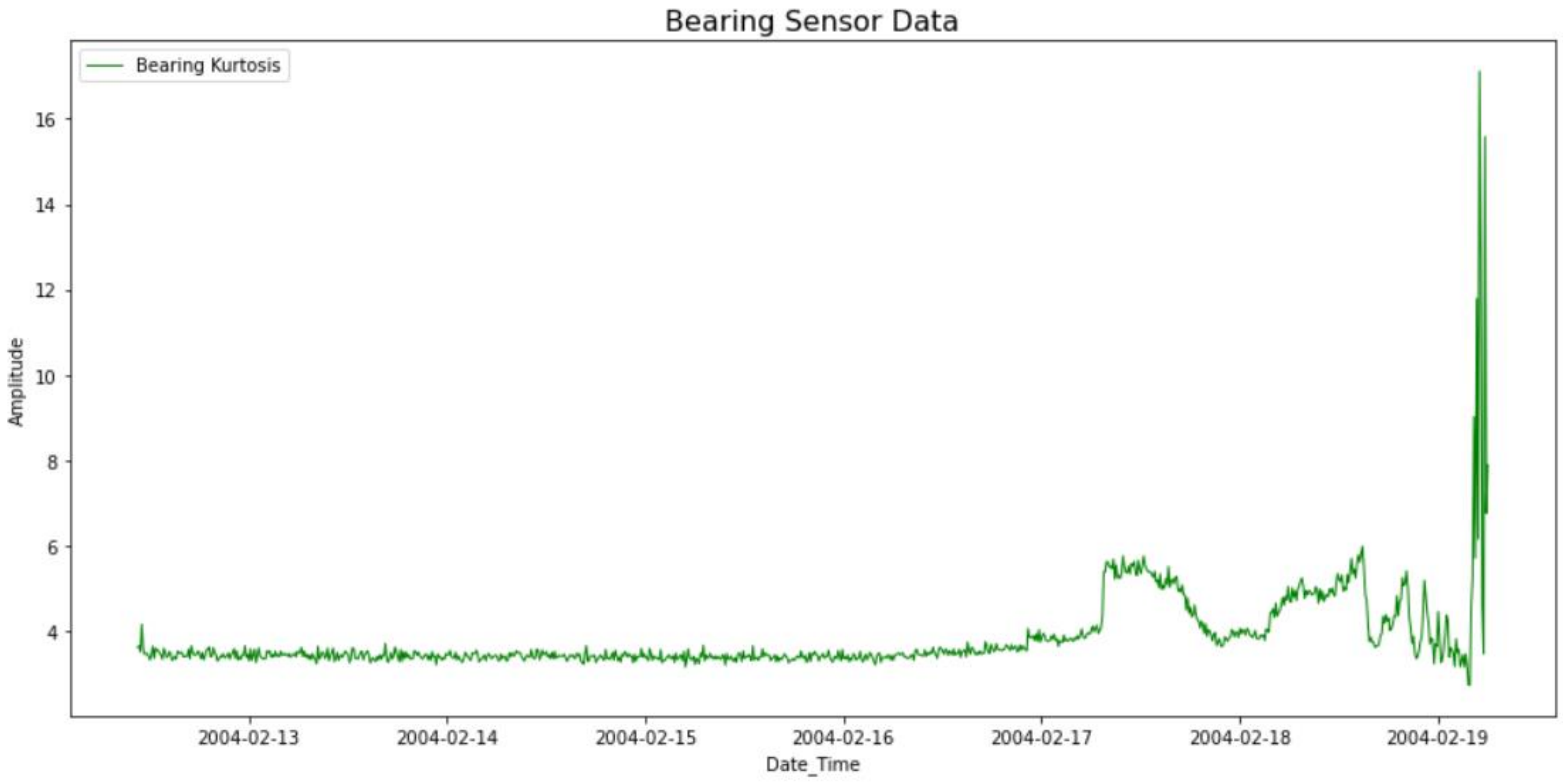
Mean Value of 1 sec vibration – Converted to Sample (Last 2 data points dropped)



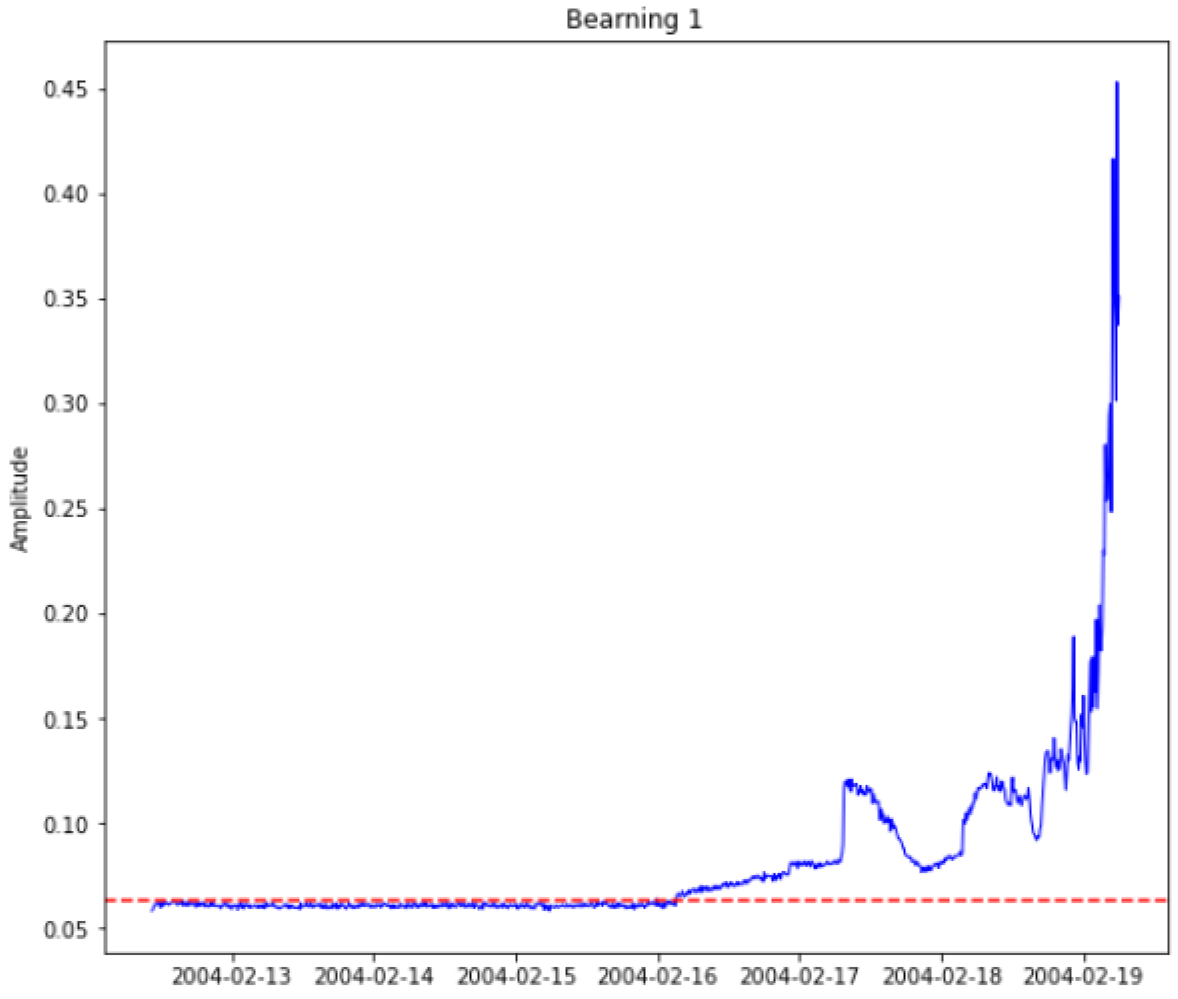
Mean vs. RMS Value of 1 sec Vibration signal
Bearing 1 - (Run to Failure Test)

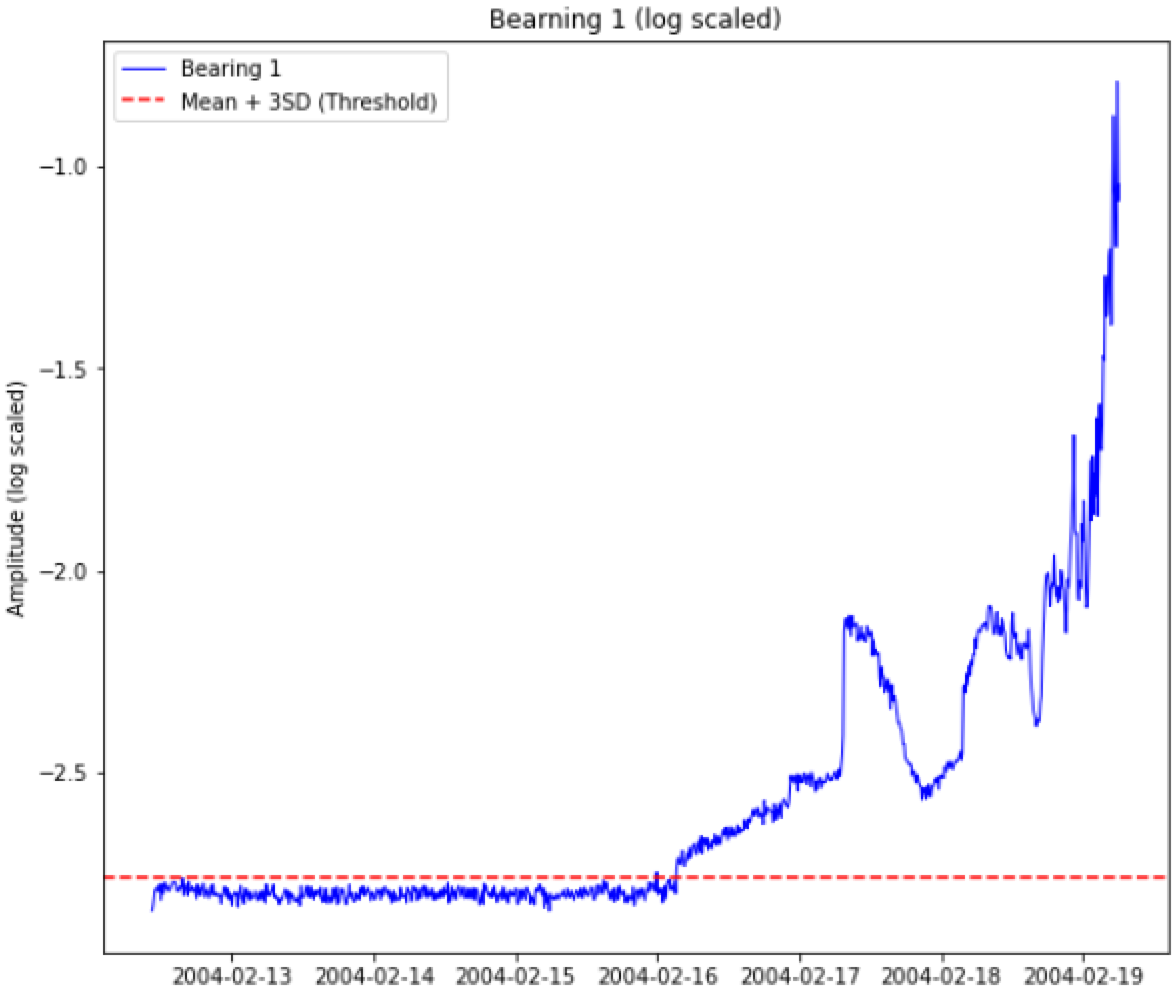


Kurtosis of 1 Sec Vibration Signal - Bearing 1 - (Run to Failure Test)

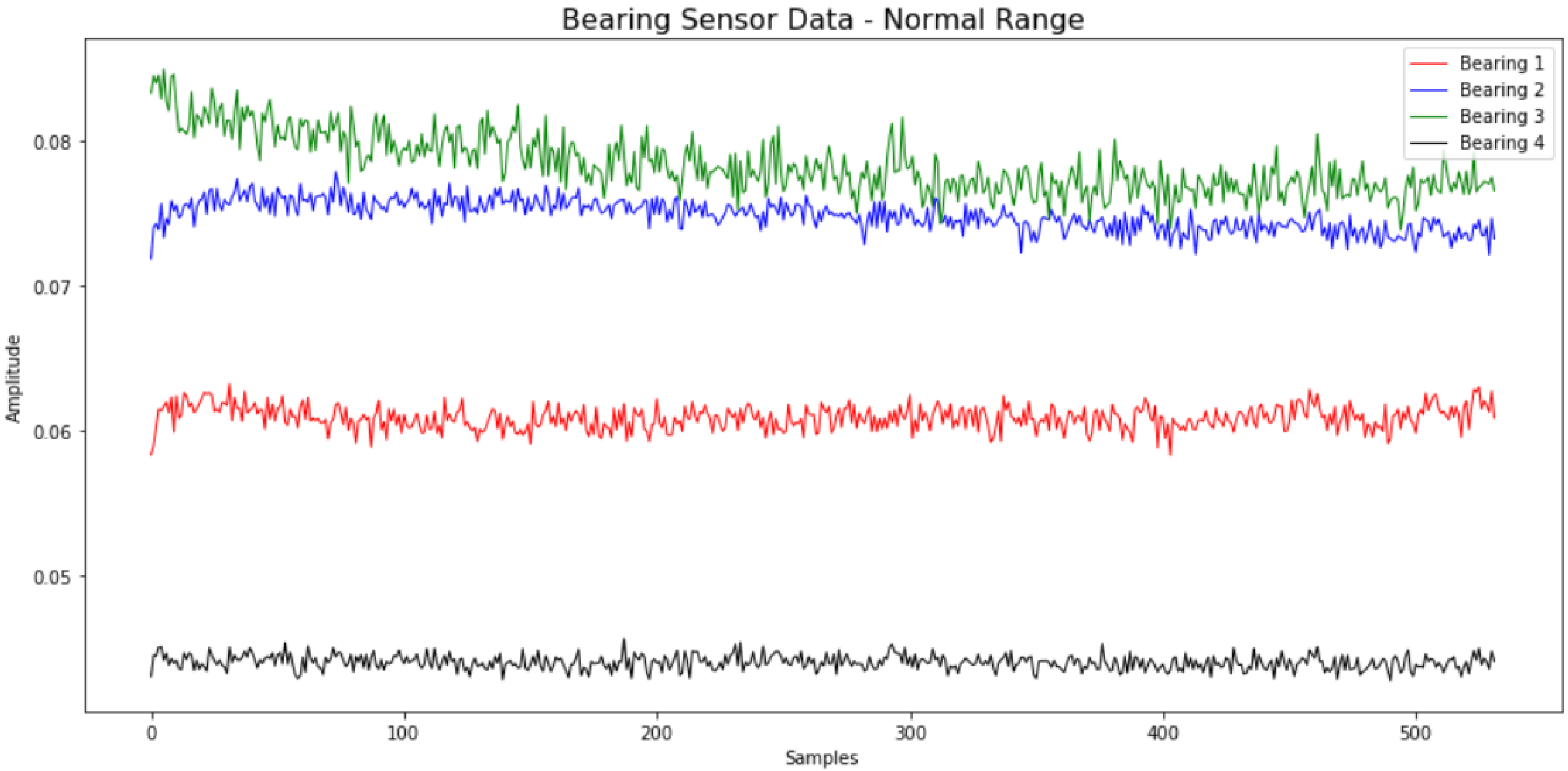


Anomaly Detection using μ + 3σ (SPC):
Mean value of 1 sec vibration signal of Bearing 1

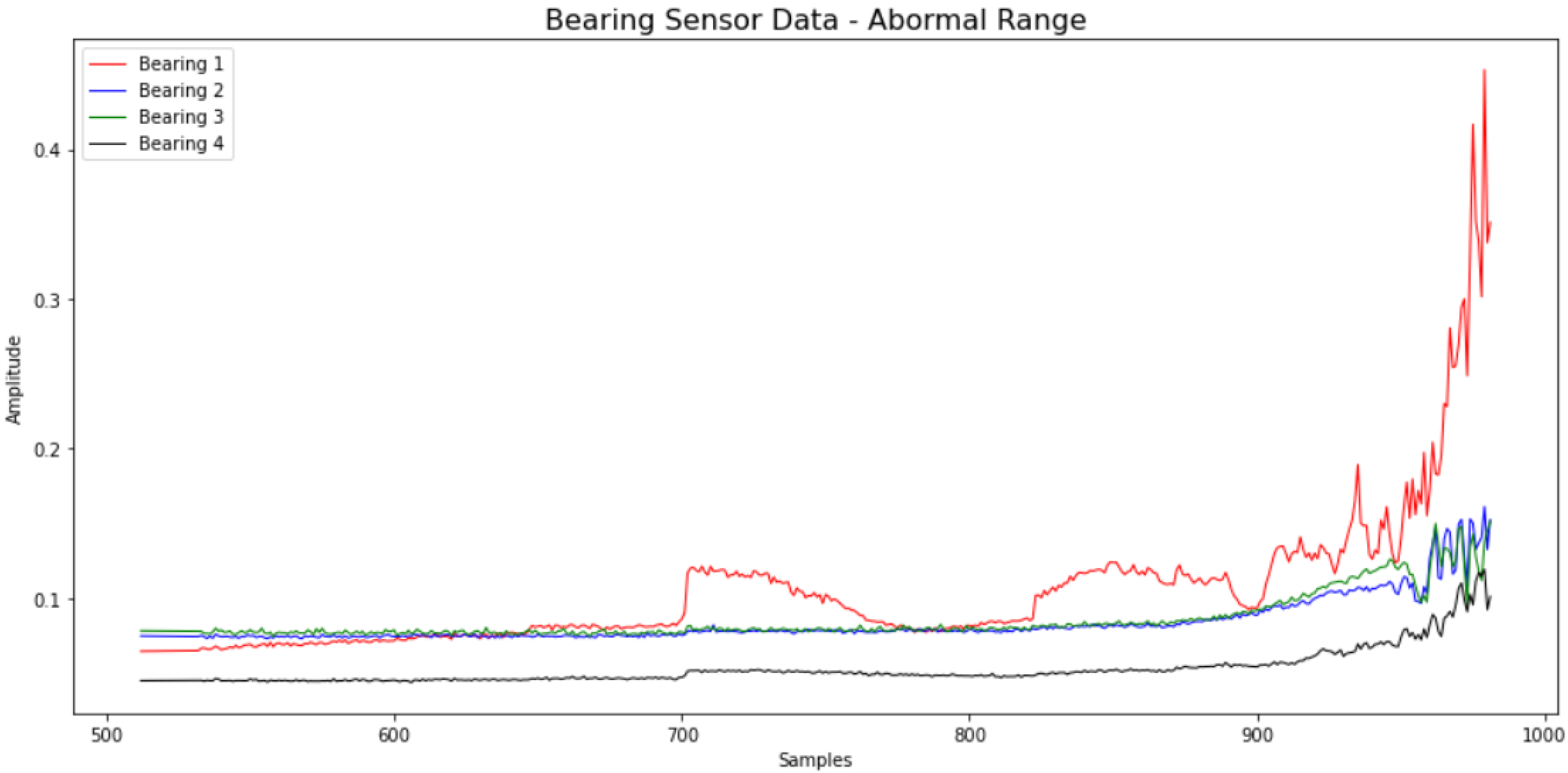




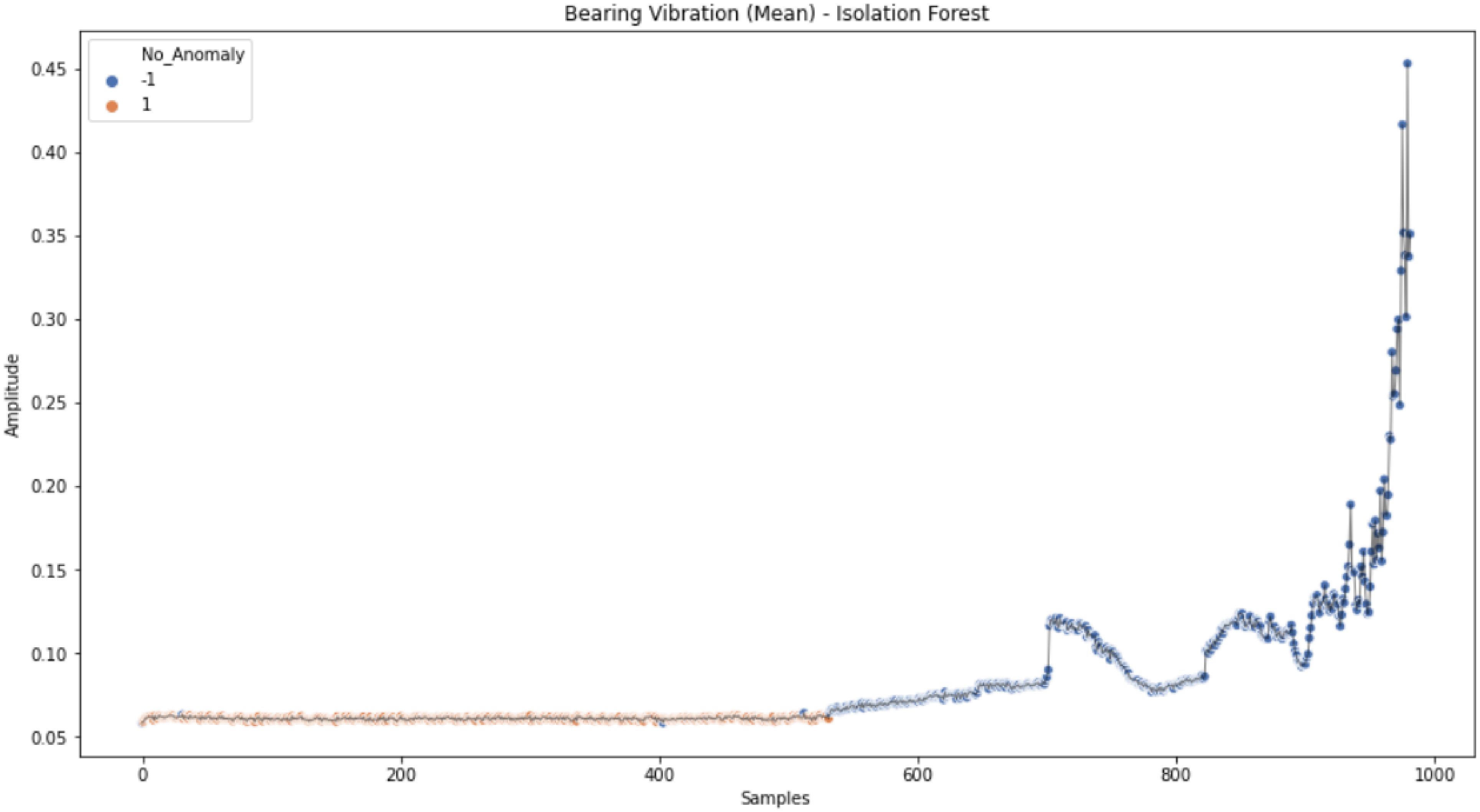
Visualizing Normal Range (Mean Value)



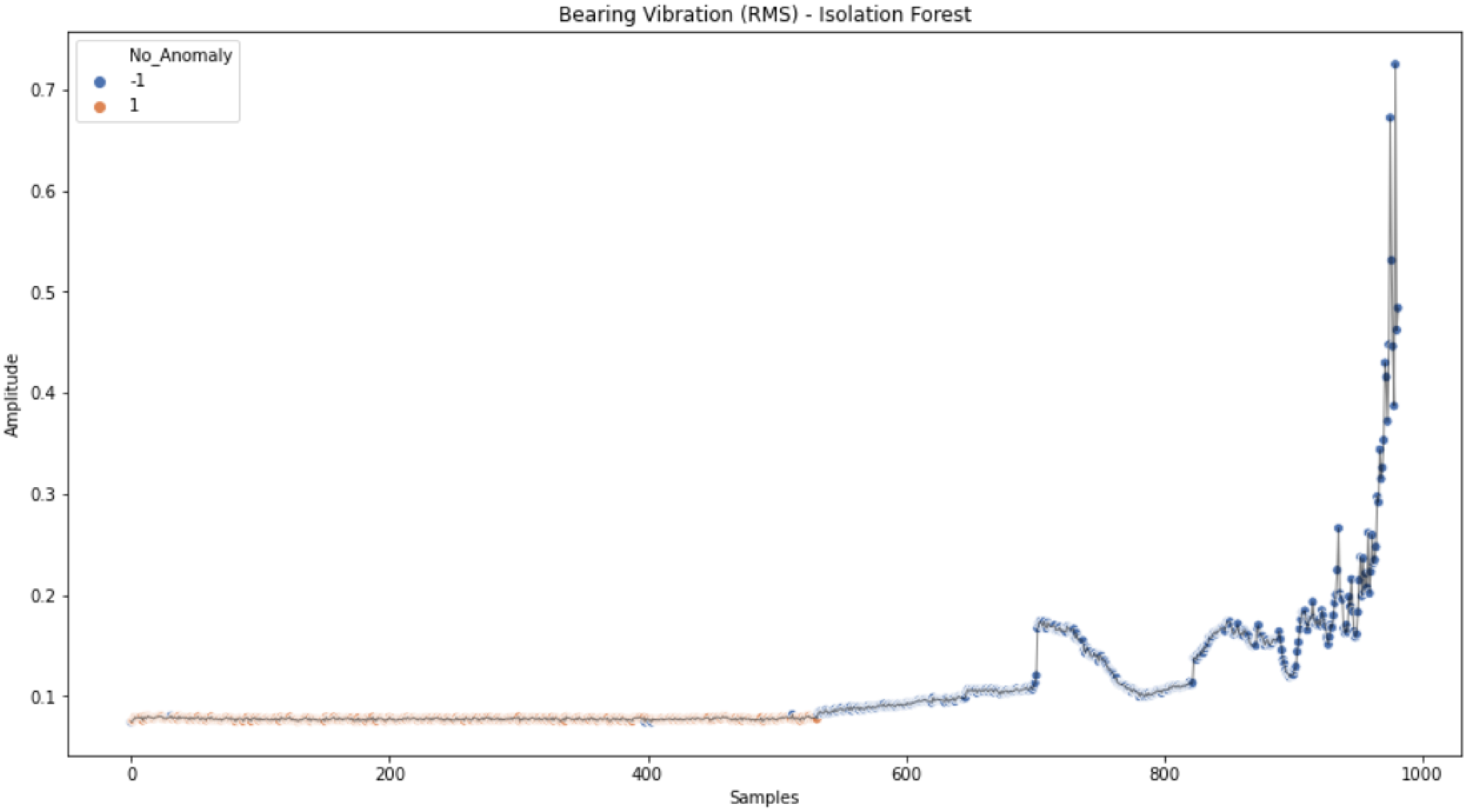
Visualizing Abnormal Range (Mean Value)



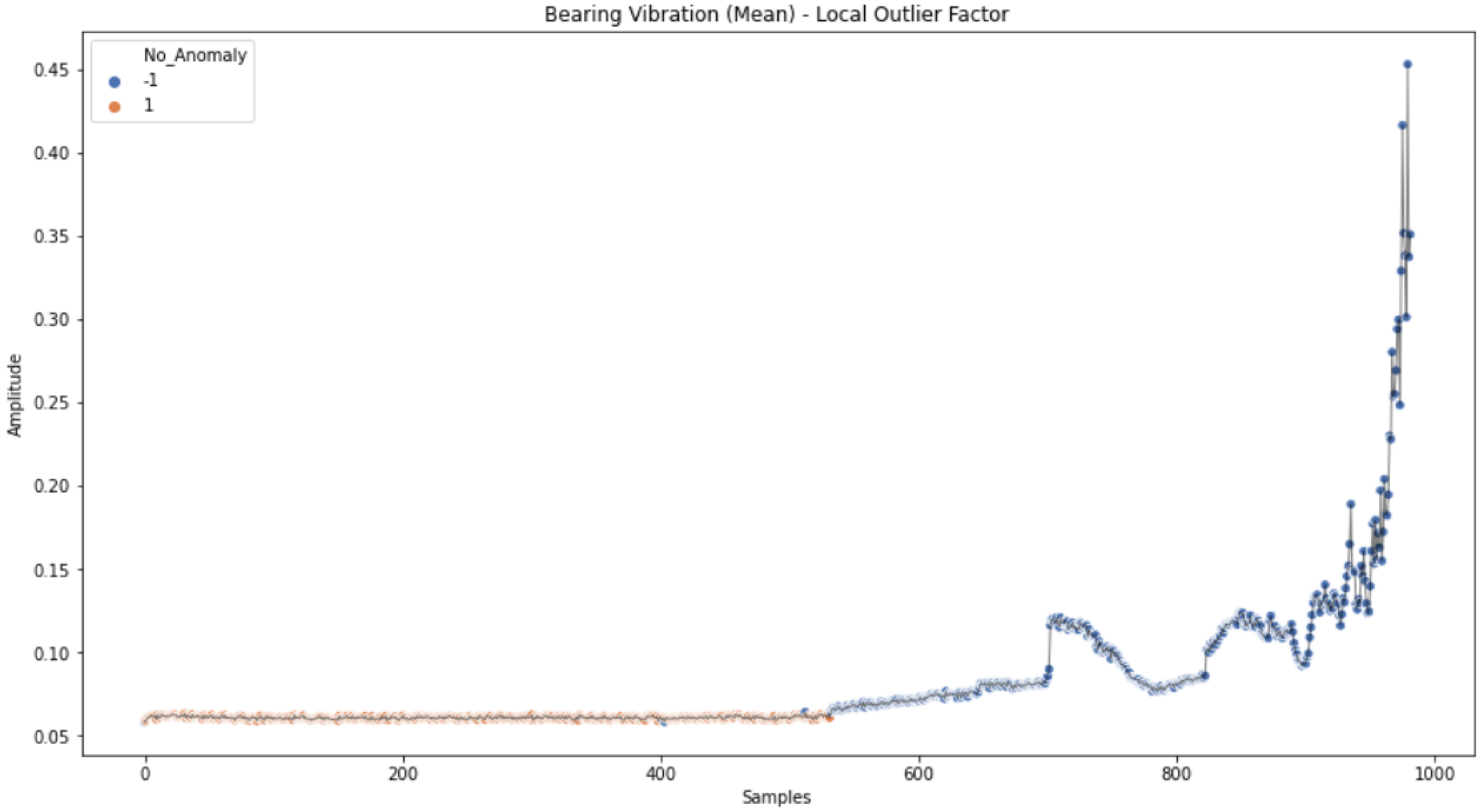
Univariate Anomaly Detection :
Isolation Forest – Mean value of 1 sec vibrations – Bearing 1



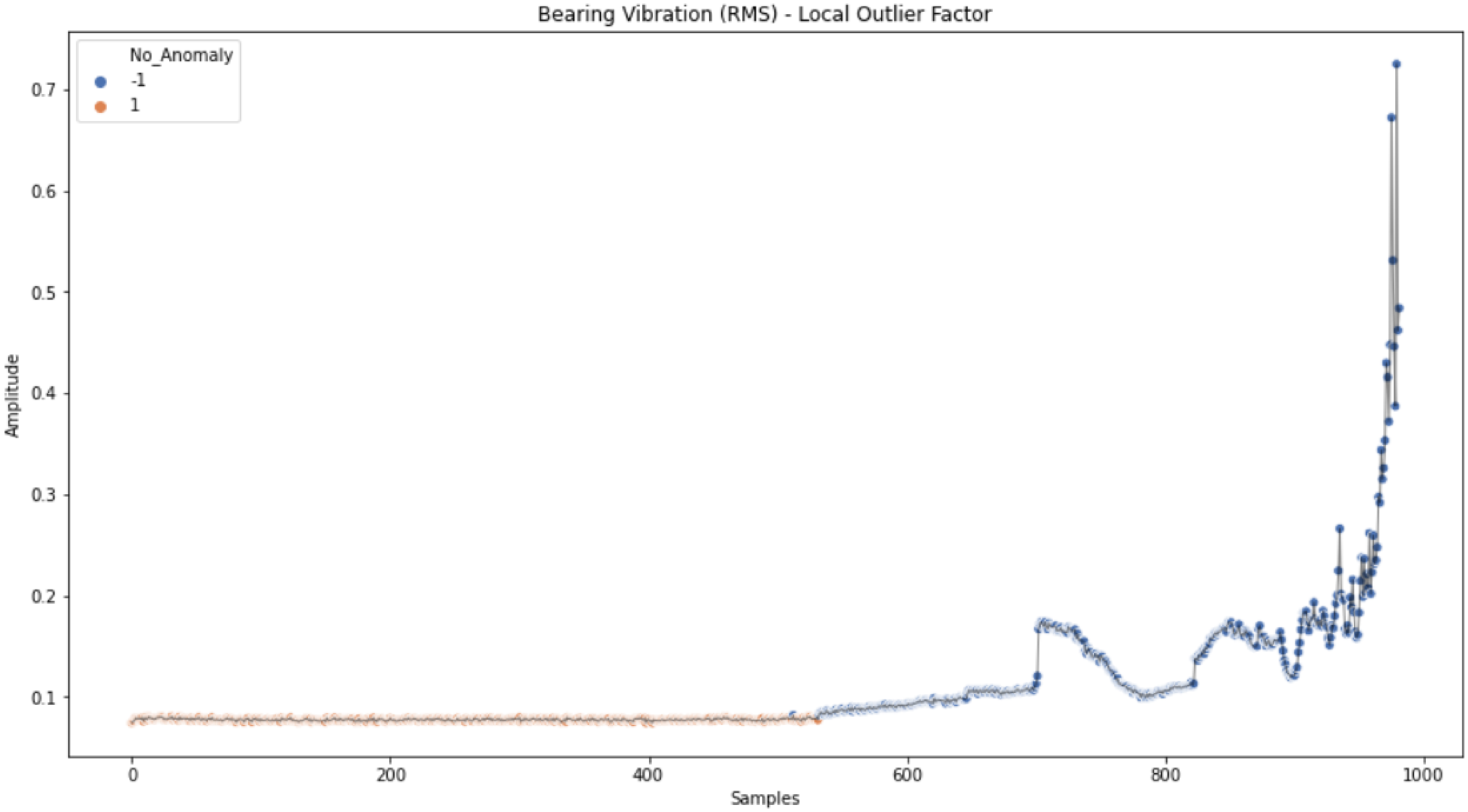
Univariate Anomaly Detection :
Isolation Forest – RMS value of 1 sec vibrations – Bearing 1



Univariate Anomaly Detection :
Local Outlier Factor – Mean value of 1 sec vibrations – Bearing 1



Univariate Anomaly Detection :
Local Outlier Factor – RMS value of 1 sec vibrations – Bearing 1



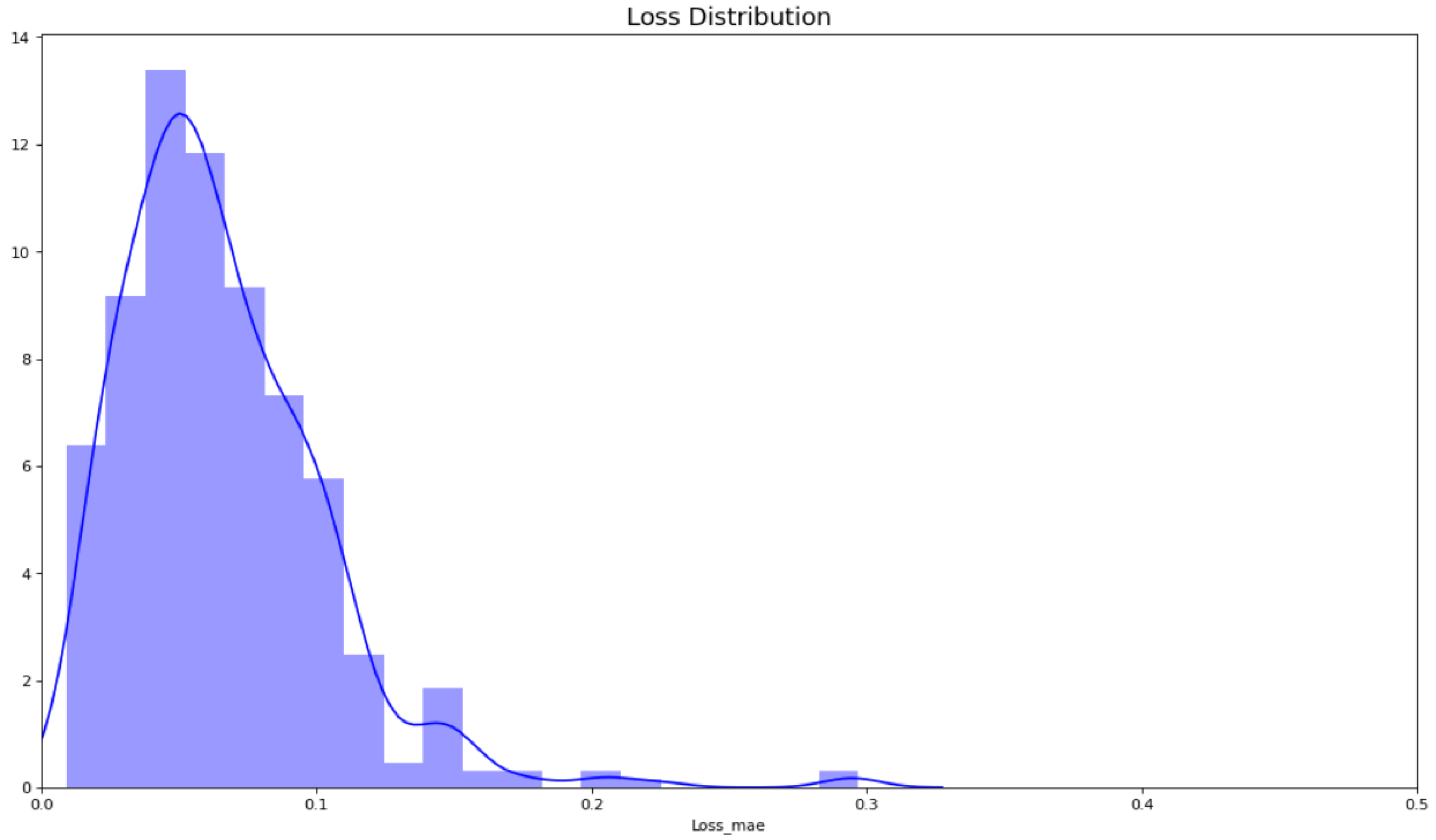
Univariate Anomaly Detection :
Comparison of Isolation Forest vs. Local Outlier Factor | Mean vs. RMS

	True Negative	False Positive	False Negative	True Positive	Accuracy
Isolation_Forest_Mean	451	0	3	528	0.996945
Isolation_Forest_RMS	451	0	4	527	0.995927
LOF_Mean	451	0	2	529	0.997963
LOF_RMS	451	0	0	531	1.000000

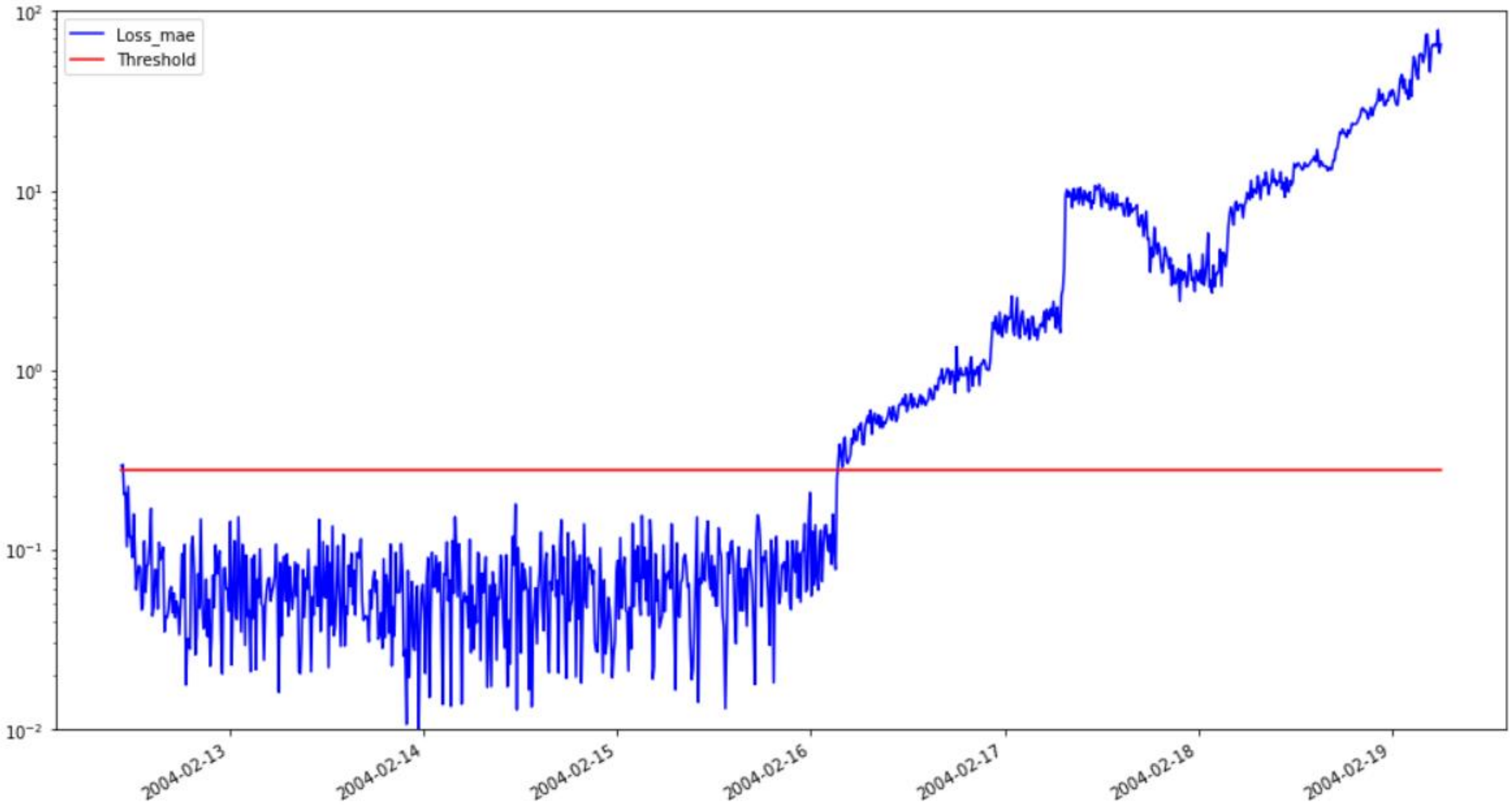
Inference

- -1 (Anomaly - Negative)
- 1 (Normal - Positive)
- The metrics True Negative and False Positive indicates the detection of outlier (anomaly). All these algorithms detects the anomalies correctly.
- False Negatives indicates false alarms i.e. The actual value is normal (1), but the predicted value is Abnormal (-1)
- Local Outlier Factor based methods reduces the false alarms (False Negatives)
- Local Outlier Factor on RMS Value performs well when compared with other algorithms

Multivariate Anomaly Detection :
Reconstruction Loss (MAE) Distribution – used in threshold calculation



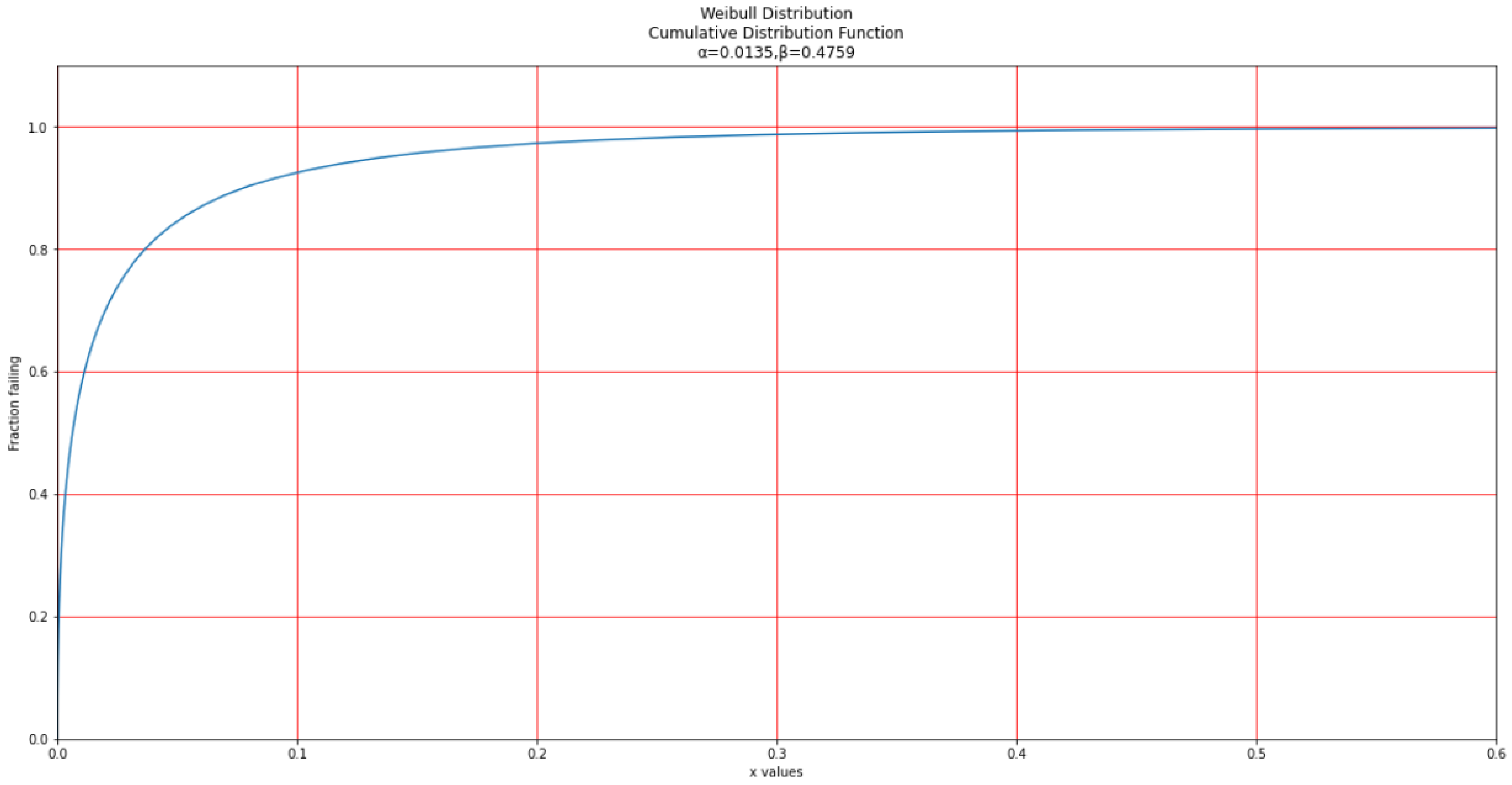
Multivariate Anomaly Detection : Reconstruction Loss (log scaled)



Inference

- For each test point we have to calculate reconstruction loss (MAE)
  - If reconstruction loss is less than threshold, the test point is marked as normal
  - If reconstruction loss is greater than threshold, the test point is marked as Anomaly

RUL Prediction – Fraction Failing: Weibull Distribution Fitted on RMS_Feature

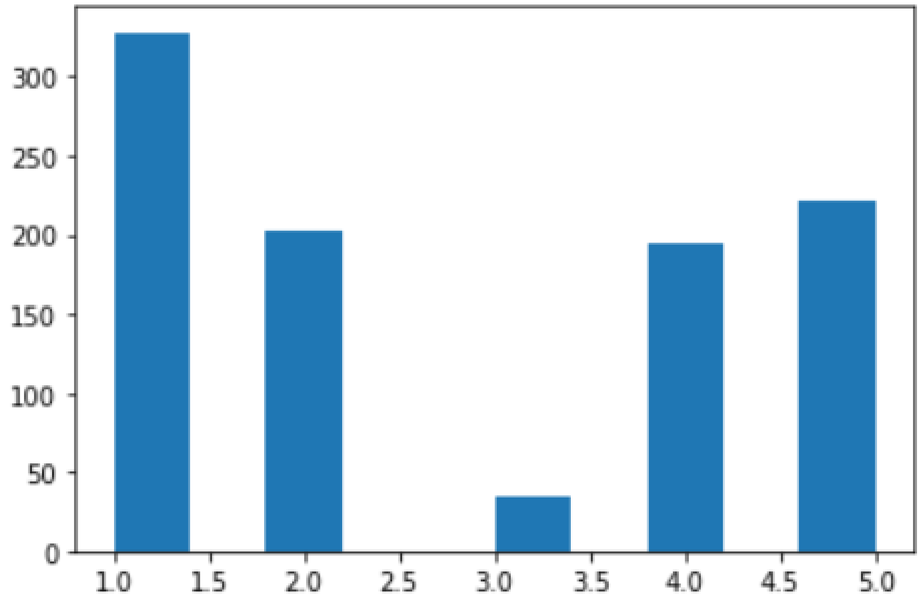


RMS_Feature = abs(RMS – (mean of Normal range))

Classification - Fraction Failing - RUL

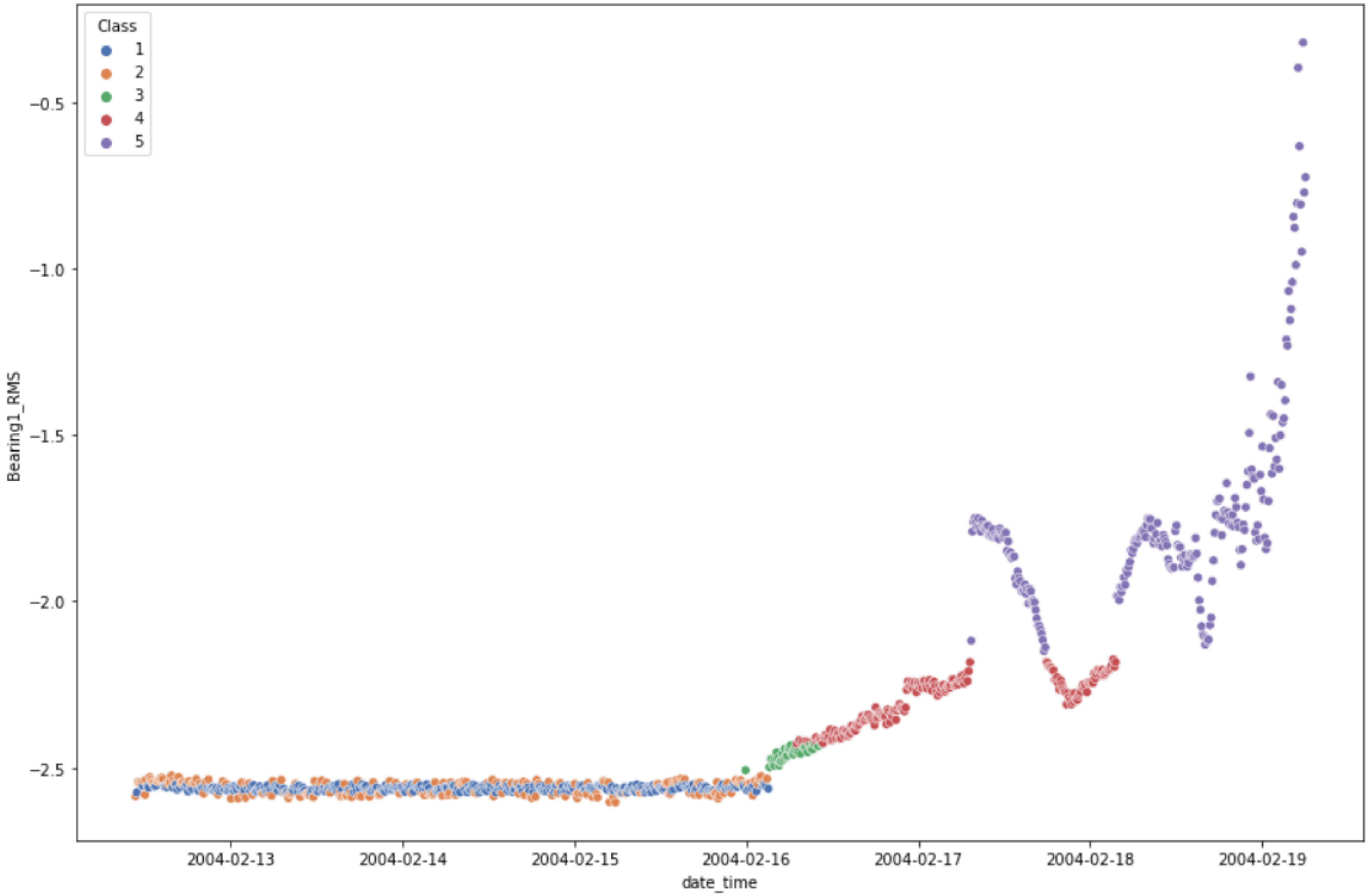
Class	RMS_Feature (Range)	Fraction Failing	RUL
1	0.0-0.001	0-20%	80%
2	0.001-0.0035	20-40%	60%
3	0.0035-0.011	40-60%	40%
4	0.011-0.037	60-80%	20%
5	> 0.037	80-100%	< 20%

Histogram – Class Frequency



1	328
5	222
2	202
4	194
3	35

RUL - Classification



Decision Tree Classifier – Test Set - Performance

	precision	recall	f1-score	support
1	1.00	1.00	1.00	103
2	1.00	1.00	1.00	59
3	1.00	1.00	1.00	7
4	1.00	1.00	1.00	57
5	1.00	1.00	1.00	69
accuracy			1.00	295
macro avg	1.00	1.00	1.00	295
weighted avg	1.00	1.00	1.00	295

Note: 30% of total data is used as Test set

API Test – Univariate Anomaly Detection

```
url = "http://127.0.0.1:5000/Ano_Det_Uni"
input_data = {"rms":0.0782,"mean":0.0619}
json_data = json.dumps(input_data)
headers = {'Content-Type': 'application/json'}
response = requests.request("POST", url, headers=headers, data = json_data)
print(response.text)
```

```
{
  "Output": "Normal"
}
```

```
url = "http://127.0.0.1:5000/Ano_Det_Uni"
input_data = {"rms":0.1585,"mean":0.1148}
json_data = json.dumps(input_data)
headers = {'Content-Type': 'application/json'}
response = requests.request("POST", url, headers=headers, data = json_data)
print(response.text)
```

```
{
  "Output": "Anomaly"
}
```

API Test – Multivariate Anomaly Detection

```
url = "http://127.0.0.1:5000/Ano_Det_Multi"
input_data = {"Bearing1": 0.0619, "Bearing2": 0.0751, "Bearing3": 0.0825, "Bearing4": 0.0438}
json_data = json.dumps(input_data)
headers = {'Content-Type': 'application/json'}
response = requests.request("POST", url, headers=headers, data = json_data)
print(response.text)
```


```
{
  "Output": "Normal"
}
```


```
url = "http://127.0.0.1:5000/Ano_Det_Multi"
input_data = {"Bearing1": 0.1156, "Bearing2": 0.0802, "Bearing3": 0.0818, "Bearing4": 0.0512}
json_data = json.dumps(input_data)
headers = {'Content-Type': 'application/json'}
response = requests.request("POST", url, headers=headers, data = json_data)
print(response.text)
```


```
{
  "Output": "Anomaly"
}
```

API Test – RUL Prediction

```
url = "http://127.0.0.1:5000/RUL_Predict"
input_data = {"Bearing1_RMS":0.0790,"Bearing1_Kurt":3.5062,"Bearing1_RMS_Prev":0.0789,"Bearing1_Kurt_Prev":3.5963}
json_data = json.dumps(input_data)
headers = {'Content-Type': 'application/json'}
response = requests.request("POST", url, headers=headers, data = json_data)
print(response.text)
```


```
{
  "Fraction Failing": "20-40%",
  "RUL": "60%",
  "RUL_Class": 2
}
```


```
url = "http://127.0.0.1:5000/RUL_Predict"
input_data = {"Bearing1_RMS":0.0,"Bearing1_Kurt":0.0,"Bearing1_RMS_Prev":0.0,"Bearing1_Kurt_Prev":0.0}
json_data = json.dumps(input_data)
headers = {'Content-Type': 'application/json'}
response = requests.request("POST", url, headers=headers, data = json_data)
print(response.text)
```


```
{
  "Fraction Failing": "No Proper Input",
  "RUL": "No Proper Input",
  "RUL_Class": "No Proper Input"
}
```

Instructions to Execute – Local Computer

- Clone / Download the Github Repository
  - **Link:** https://github.com/selvasundar93/AI_PdM
- Run app.py by executing the command **python app.py** in terminal
- Use Jupyter Notebook or any Script to send & receive data from this API
- **Univariate Anomaly Detection**
  - URL - Endpoint = "127.0.0.1:5000/Ano_Det_Uni"
  - Load **"{"rms":value,"mean":value}"** as JSON data
    - Ex: *"{"rms": 0.0782, "mean": 0.0619}"*
  - API will respond back with JSON data in the form "{"Output": output_value}"
    - Ex: *"{"Output": Normal}"*
- **Multivariate Anomaly Detection**
  - URL - Endpoint = "127.0.0.1:5000/Ano_Det_Multi"
  - Load **"{"Bearing1":value,"Bearing2":value,"Bearing3":value,"Bearing4":value}"** as JSON data
    - Ex: *"{"Bearing1": 0.0619, "Bearing2": 0.0751, "Bearing3": 0.0825, "Bearing4": 0.0438}"*
  - API will respond back with JSON data in the form "{"Output": output_value}"
    - Ex: *"{"Output": Normal}"*

Instructions to Execute

- **Remaining Useful Life Estimation**
  - URL - Endpoint = "127.0.0.1:5000/RUL_Predict"
  - Load **"{"Bearing1_RMS":value,"Bearing1_Kurt":value,"Bearing1_RMS_Prev":value,"Bearing1_Kurt_Prev":value}"** as JSON data
    - Ex: *"{"Bearing1_RMS":0.0790,"Bearing1_Kurt":3.5062,"Bearing1_RMS_Prev":0.0789,"Bearing1_Kurt_Prev":3.5963}"*
  - API will respond back with JSON data in the form "{"RUL_Class":output_value,"Fraction Failing":output_value, "RUL":output_value}"
    - Ex: *"{"RUL_Class":2,"Fraction Failing":"20-40%", "RUL":"60%"}“*
- **Note:**
  - Three outputs are possible (Anomaly Detection): Normal, Anomaly, No Proper Input
    - **Normal** : Vibrations in Normal Range - Healthy State
    - **Anomaly**: Vibrations exceeds Normal Range - Unhealthy / Possibility of failure in near future
    - **No Proper Input** : If all inputs are "0" - Sensor Fault / Machine is turned off
  - For accurate prediction, input values must be given with atleast four decimal places (Ex: 0.0825)
  - Refer Tests folder for examples
  - Refer Tests/Day_Wise_Data folder for feature engineered dataset
    - Link: https://github.com/selvasundar93/AI_PdM/tree/main/Tests/Day_Wise_Data

GitHub Repositories

- Project Deployed using FLASK API & Containerization using Docker
  - Link: https://github.com/selvasundar93/AI_PdM
- Project Deployed using FLASK API & Heroku
  - Link: https://github.com/selvasundar93/AI_PdM_Heroku
- Project Deployed using Heroku with File Input
  - Link: https://github.com/selvasundar93/AI_PdM_File_Reader
- Project Notebooks
  - Link: https://github.com/selvasundar93/Springboard_Capstone