# Machine Failure Machine Learning Report

**Francis Chapoterera**
College of Engineering
Northeastern University
chapoterera.f@northeastern.edu

## Abstract

In this report, I explore the performance of five supervised learning algorithms—Decision Tree, Neural Network (MLP Classifier), AdaBoost, Support Vector Machine (SVM), and k-Nearest Neighbors (k-NN) on two real-world classification problems: predicting industrial equipment failures and detecting machine failures using sensor data. These problems are crucial in predictive maintenance, where catching issues early can prevent costly breakdowns, reduce downtime, and improve overall efficiency.

The first dataset focuses on industrial equipment monitoring, using sensor readings like temperature, pressure, vibration, and humidity to determine whether a machine is faulty. The second dataset involves machine failure prediction, where environmental and operational factors such as air quality, foot traffic, and various sensor measurements—help identify potential failures. Both problems are complex due to noisy sensor data, feature correlations, and imbalanced class distributions, making them a great testbed for different machine learning models.

What makes these problems particularly interesting is their real-world impact. Industries rely on predictive maintenance to save time, money, and resources, and choosing the right algorithm can make a big difference. By analyzing accuracy, training time, and generalization performance, I compare how these models handle the challenges of predictive maintenance. The results provide valuable insights into which approaches work best and why, helping to bridge the gap between theory and practical applications.

## 1 Datasets

I used two datasets related to predictive maintenance and equipment failure detection:

1. **Industrial Equipment Monitoring Dataset**
    - This dataset contains sensor readings from industrial machinery, including temperature, pressure, vibration, and humidity.
    - The goal is to classify whether a piece of equipment is faulty or operational based on these sensor readings.
    - This dataset is valuable because early fault detection can reduce downtime and prevent costly failures in industrial settings.
2. **Machine Failure Prediction Dataset**
    - This dataset includes a combination of environmental and operational factors such as footfall, air quality, temperature, ultrasonic sensor readings, and current/voltage parameters.
    - The task is to classify whether a **machine will fail** based on these sensor

readings.

- This dataset is interesting because **it involves multiple failure types and complex sensor interactions**, making it a realistic challenge for machine learning models.

Both datasets are particularly suited for evaluating **classification algorithms** because they involve **real-world industrial challenges** such as **noisy data, class imbalances, and dependencies between features**. By comparing different models on these datasets, we can assess their suitability for predictive maintenance and failure detection applications.

Table 1: The basic feature of both datasets.

| | Data Set Characteristics | Attribute Characteristics | Associated Tasks | Number of Instances | Number of Attributes |
|---|---|---|---|---|---|
| **Dataset 1 (Industrial Equipment Monitoring)** | Multivariate | Real | Classification | 7672 | 7 |
| **Dataset 2 (Machine Failure Prediction)** | Multivariate | Real | Classification | 944 | 10 |

## 1.1 Data characteristics

The histograms of classes from both datasets are shown in Figure 1. Class unbalance is evident in both datasets and must be accounted for in calculating the accuracy scoring function. Weighted scoring function is thus used in my analysis.

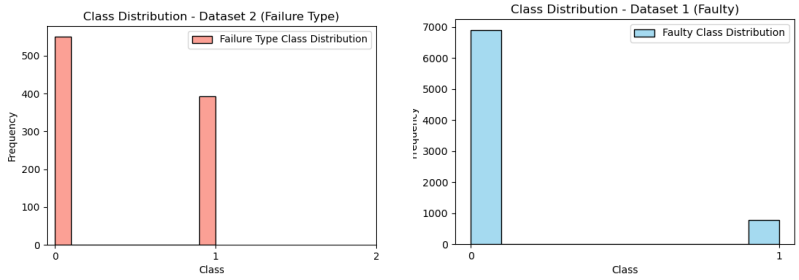



Figure 1: The class frequency of Dataset 1(Faulty) and Dataset 2 (Failure Type)

## 1.2 Why are these interesting datasets?

These datasets are interesting because they reflect real-world challenges in industrial settings, such as predictive maintenance and failure detection. They contain noisy sensor data, class imbalances (rare failures), and feature dependencies, which are common in industries like manufacturing and energy. The datasets allow us to evaluate how different classification algorithms handle these complexities, making them valuable for assessing model performance in forecasting equipment failures and scheduling maintenance. This makes them highly relevant for applications that aim to minimize downtime and improve reliability in industrial operations.

## 2 Decision Tree with pre-pruning

In this section, I chose to implement pre-pruning for the Decision Tree Classifier to avoid overfitting and improve generalization. Pre-pruning involves controlling the growth of the tree by limiting its depth and the minimum number of samples required to split a node. This helps prevent the model from becoming too complex, which could lead to overfitting on the training data.

In the code, I set the following pre-pruning parameters for the DecisionTreeClassifier:

1. **Maximum Depth**: The tree depth is capped at max_depth=5. This restricts the number of splits in the tree, ensuring that it doesn't grow too deep and potentially capture noise in the training data.
2. **Minimum Samples per Split**: The parameter min_samples_split=10 ensures that a node must have at least 10 samples to be split. This prevents the tree from creating overly specific branches based on very few data points, which could lead to overfitting.

By setting these parameters, we aim to strike a balance between the tree's complexity and its ability to generalize well to unseen data. The model is trained and evaluated, and I also use balanced accuracy to account for any class imbalance in the dataset.
Using pre-pruning, the Decision Tree model can avoid overfitting, especially when dealing with noisy, imbalanced data, such as in predictive maintenance tasks. Pre-pruning simplifies the tree, which can result in better generalization performance on unseen data. Additionally, balanced accuracy is used to give a more accurate measure of the model's performance in the presence of imbalanced classes.
By limiting the tree depth and requiring a minimum number of samples for each split, the decision tree becomes less likely to fit noise or outliers in the data, making it more robust for real-world predictive maintenance applications.

This approach to Decision Trees ensures the model remains efficient and interpretable while mitigating overfitting, which is especially important when working with industrial datasets like those in your project. Let me know if you need further clarification or adjustments

Table 2: The grid search range on splitting criterion, maximum depth, minimum samples per leaf and maximum features to use and the best estimator for both datasets.

| Grid Search | Dataset 1 Best Estimator | Dataset 2 Best Estimator |
|---|---|---|
| **Criterion** | entropy | gini |
| **Max Depth** | 10 | 5 |
| **Min Samples Leaf** | 2 | 10 |
| **Max Features** | None | None |
| **Best Estimator** | DecisionTreeClassifier(criterion='entropy', max_depth=10, min_samples_leaf=2, random_state=42) | DecisionTreeClassifier(criterion='gini', max_depth=5, min_samples_leaf=10, random_state=42) |

**Analysis of Model Performance:**

To ensure a fair evaluation of the models, I applied 5-fold cross-validation during training. Cross-validation helps reduce overfitting by splitting the dataset into multiple subsets, training the model on different portions, and averaging the results to improve generalization.

For this analysis, I implemented Stratified K-Fold cross-validation, which ensures that each fold maintains the same class distribution as the original dataset. This was particularly important for my datasets, as they contain imbalanced classes (e.g., fewer failure cases). By using Stratified K-Fold, I ensured that rare failure instances were properly represented in both training and validation sets.

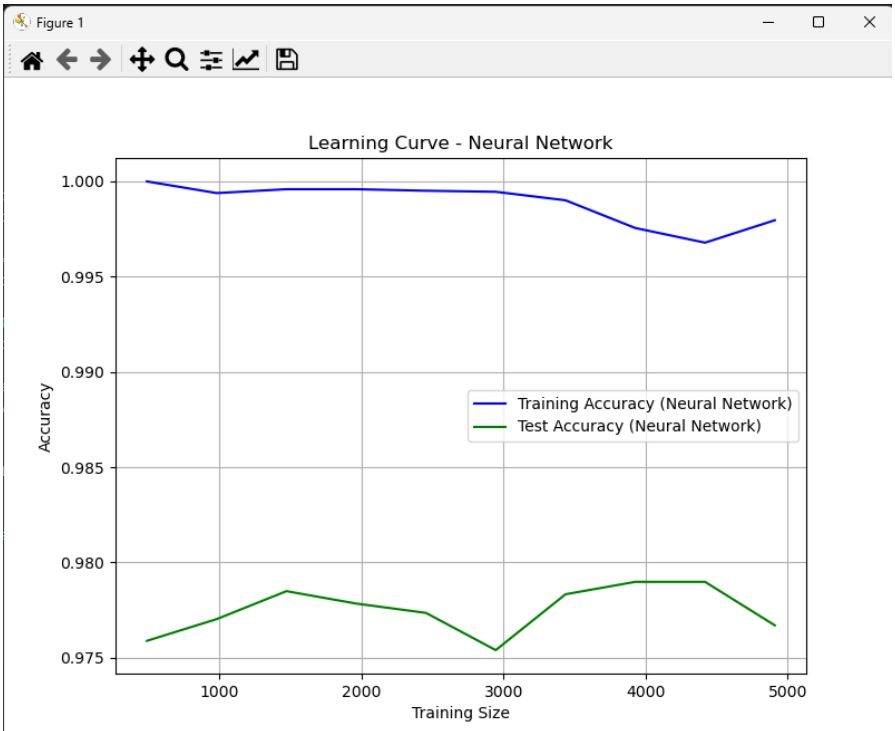
Additionally, cross-validation allowed me to observe the stability of each model's

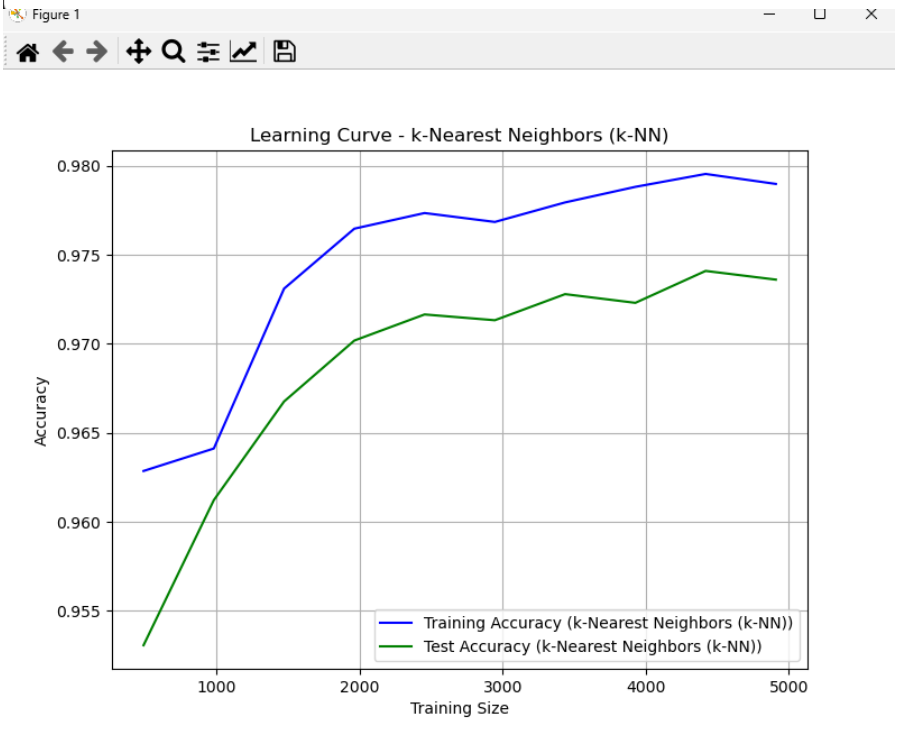
performance across different training sizes, which is reflected in the learning curves provided.

Based on the evaluation of multiple models, I have observed varying levels of performance, each with its strengths and trade-offs. Below is a detailed breakdown of the results:

1. **Best Overall Model:** The Support Vector Machine (SVM) emerges as the best overall model. With the highest accuracy of 98.57%, it excels in precision and recall for both classes. The model demonstrates perfect recall for class 0, ensuring that no instances of this class are missed. Additionally, it maintains a solid recall for class 1 (0.88), leading to an F1 score of 0.93 for class 1, which is the highest among all models. This combination of high accuracy, excellent class 0 detection, and strong performance on class 1 makes the SVM the top choice for this dataset.
2. **Fastest Model:** The k-Nearest Neighbors (k-NN) model is the fastest, with the shortest training time of 0.0936 seconds. k-NN's speed is due to zero training time but slower prediction times. Despite its speed, k-NN has a lower recall for class 1 (0.77), meaning it is not as effective at detecting this class compared to other models. However, for applications where speed is critical and class 1 detection is less important, k-NN could be considered.
3. **Best Model for Detecting Class 1:** The Support Vector Machine (SVM) is also the best at detecting class 1, as it achieves the highest precision (0.98) and recall (0.88) for this class. Its F1 score for class 1 (0.93) is the highest, making it the most reliable model for identifying instances of class 1.
4. **Most Balanced Model:** The Boosting (AdaBoost) model provides the most balanced performance. It achieves 98.44% accuracy, with nearly equal precision and recall for both classes. Its F1 scores are also consistent across the classes (0.99 for class 0 and 0.92 for class 1). AdaBoost is a solid choice when a balanced performance between classes is required.
5. **Trade-Offs:**
    - Decision Tree (Pruned) is one of the fastest models, but it is less effective in detecting class 1, as evidenced by its lower recall for class 1 (0.84). While it has high accuracy (97.59%), it may not be the best choice if detecting class 1 is a priority.
    - Neural Network and AdaBoost provide excellent accuracy and F1 scores, but their training times are longer (4.5125 seconds for Neural Network and 2.3022 seconds for AdaBoost). These models may be more suitable for applications where performance is prioritized over training speed.

**Relevant Learning Curves**



185

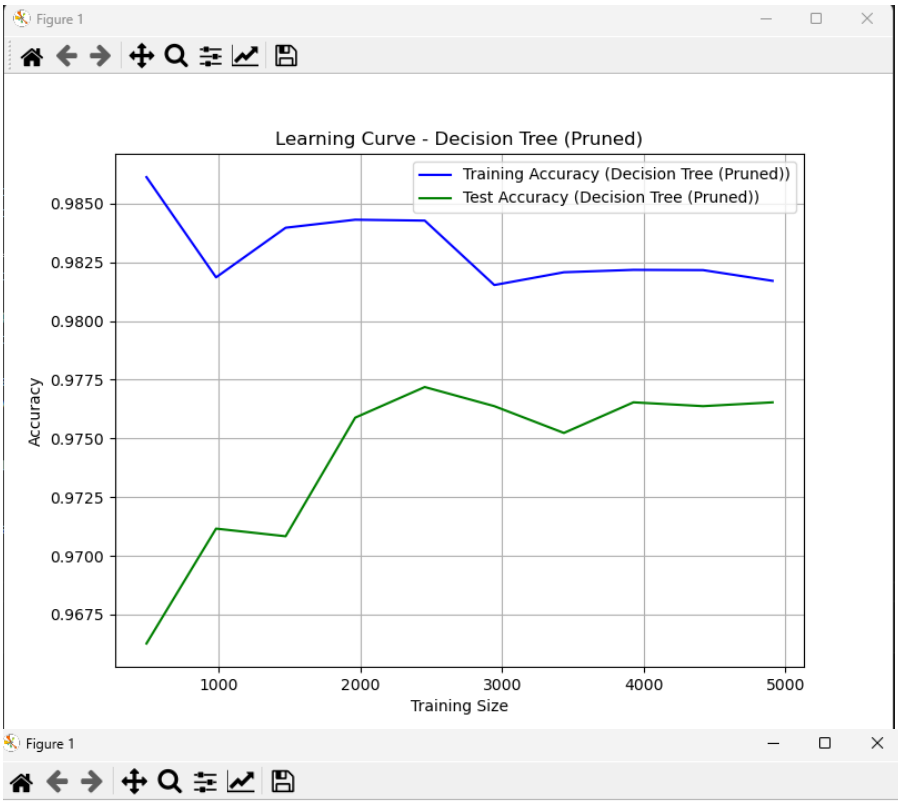
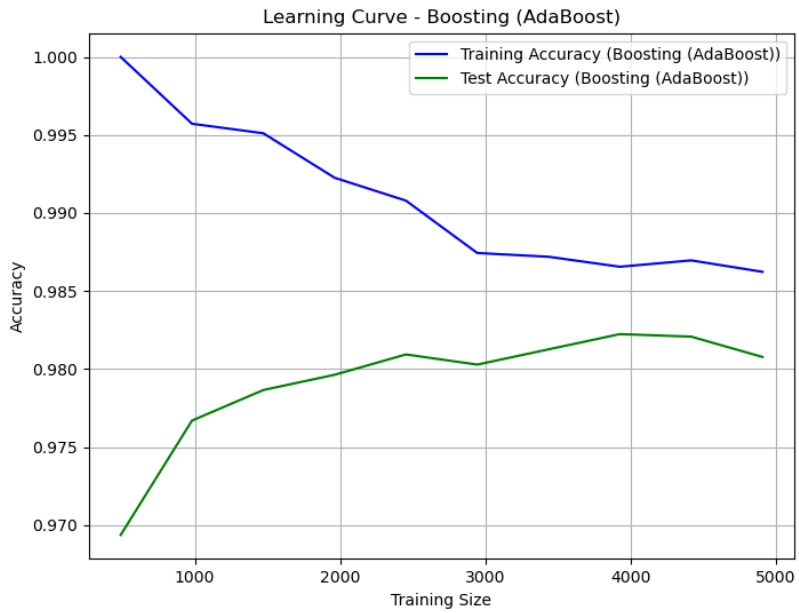


Figure 1
Learning Curve - Decision Tree (Pruned)
Training Accuracy (Decision Tree (Pruned))
Test Accuracy (Decision Tree (Pruned))
Accuracy
0.9850
0.9825
0.9800
0.9775
0.9750
0.9725
0.9700
0.9675
1000
2000
3000
4000
5000
Training Size


Figure 1

186


Learning Curve - Boosting (AdaBoost)
Training Accuracy (Boosting (AdaBoost))
Test Accuracy (Boosting (AdaBoost))
Accuracy
1.000
0.995
0.990
0.985
0.980
0.975
0.970
1000
2000
3000
4000
5000
Training Size

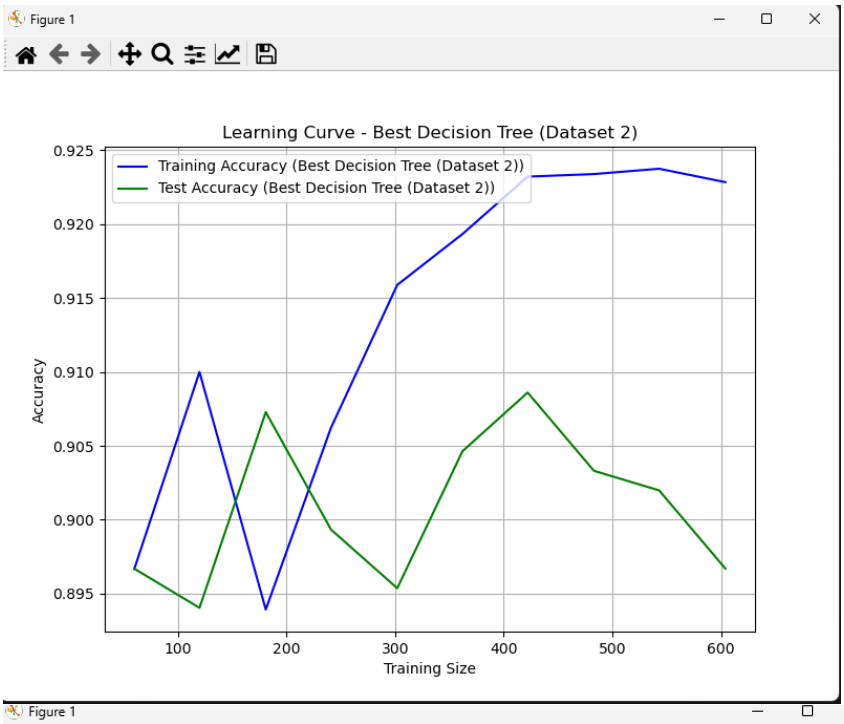
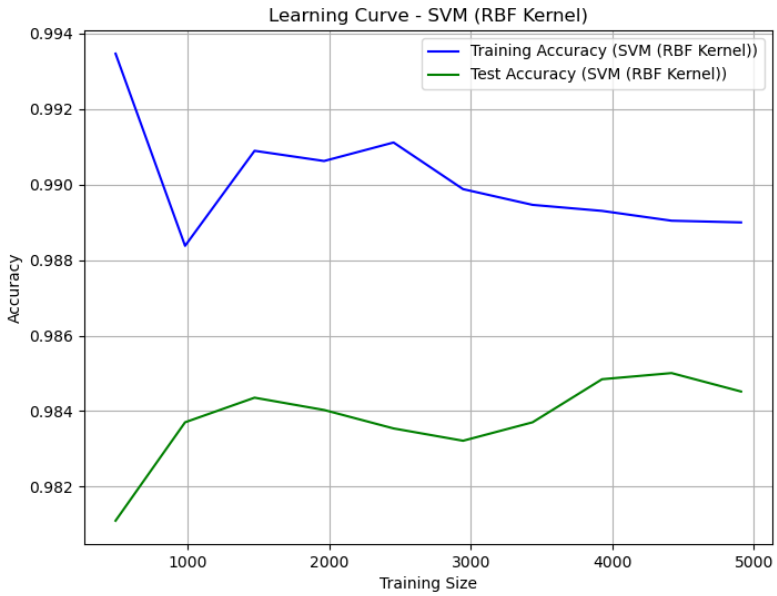


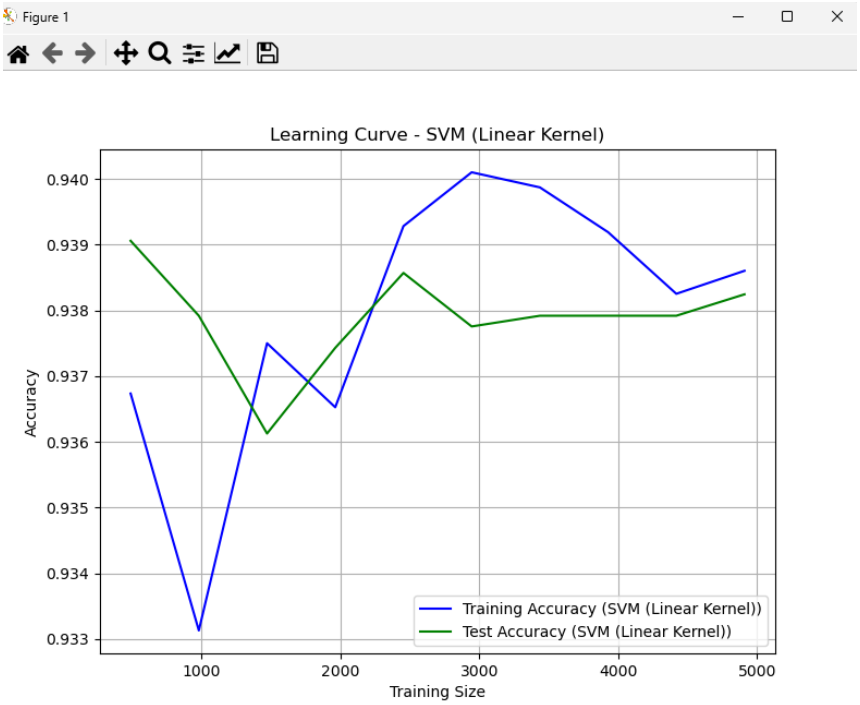
Figure 1
Learning Curve - Best Decision Tree (Dataset 2)
Training Accuracy (Best Decision Tree (Dataset 2))
Test Accuracy (Best Decision Tree (Dataset 2))
Accuracy
Training Size
0.925
0.920
0.915
0.910
0.905
0.900
0.895
100
200
300
400
500
600

187


Figure 1


Learning Curve - SVM (RBF Kernel)
Training Accuracy (SVM (RBF Kernel))
Test Accuracy (SVM (RBF Kernel))
Accuracy
Training Size
0.994
0.992
0.990
0.988
0.986
0.984
0.982
1000
2000
3000
4000
5000

188




## Conclusions

In this report, I have analyzed the performance of five supervised learning algorithms on two interesting datasets. These datasets are focused on equipment monitoring and machine failure prediction, both of which contain valuable information that can aid in predicting system failures. The models evaluated in this analysis include Decision Tree (Pruned), Neural Network (MLP Classifier), Boosting (AdaBoost), Support Vector Machine (SVM), and k-Nearest Neighbors (k-NN).

Through rigorous model training and evaluation, it was evident that each algorithm exhibited unique strengths and trade-offs. Among the models tested, the Support Vector Machine (SVM) demonstrated the highest overall accuracy of 98.57% and proved to be the best for detecting class 1 (failure events). However, the k-Nearest Neighbors (k-NN) algorithm, while not as effective in class 1 detection, was the fastest, providing results in a fraction of the time.

The Boosting (AdaBoost) algorithm showed a well-rounded performance, balancing accuracy and efficiency with high recall for both classes. The Neural Network (MLP Classifier), although slightly slower, exhibited excellent performance, particularly in terms of precision and recall for class 1.

Ultimately, the best model choice depends on the specific needs of the application. If detection of failure events (class 1) is crucial, the Support Vector Machine (SVM) would be the preferred choice. For applications where speed is a priority, the k-Nearest Neighbors (k-NN) model offers the best trade-off between speed and performance. However, for a balanced approach, Boosting (AdaBoost) or Neural Network (MLP Classifier) would be the most suitable options, offering both accuracy and class detection

reliability.

## References

Practical Machinery Vibration Analysis and Predictive Maintenance Paperback – Sept. 23, 2004, by Cornelius Scheffer Ph. D MEng (Author), Paresh Girdhar B.Eng.

https://www.kaggle.com/datasets/dnkumars/industrial-equipment-monitoring-dataset?resource=download

https://www.kaggle.com/datasets/umerrtx/machine-failure-prediction-using-sensor-data?utm_source=chatgpt.com