Predictive Maintenance Model: Leveraging Binary Classification Algorithms for Equipment Failure Prediction

Bethel Unwan, Ali Pirhosseinloo, Gediyon M.Girma

ECE1513 Project, Introduction to Machine Learning

Abstract
Predictive maintenance aims to anticipate equipment failures before they occur, allowing for timely repairs, unnecessary repair expenses and minimizing disruptions to operations. In this project, we employ various machine learning techniques to classify patterns and indicators of potential equipment failures.

1 Introduction

Many industries rely on machines and equipment for their operations. In today's dynamic industries, where downtime can translate to significant losses, predictive maintenance offers a smart approach to asset management, ensuring optimal operational efficiency and cost-effectiveness.

Traditional maintenance strategies often rely on scheduled inspections or reactive approaches, leading to costly downtimes and inefficient resource allocation. Moreover, implementing predictive maintenance entails navigating complex data streams, integrating sensor technologies, and developing robust algorithms capable of discerning meaningful patterns amidst noise.

In our approach, we employ three machine learning techniques: the Perceptron, Support Vector Machines (SVM), and Neural Networks. The Perceptron model serves as a fundamental building block, offering simplicity in capturing linear patterns within the data. Complementing this, SVM provides robustness in handling complex, nonlinear relationships by mapping data points into higher-dimensional spaces. Additionally, the versatility of Neural Networks allows for the exploration of intricate patterns and correlations, making them suitable for tasks that demand a deep understanding and adaptability to different datasets.

The dataset [1] we utilize consists of 10000 data points arranged in rows with 6 features in columns, including Product Type (Low, Medium, High), Air Temperature, Process Temperature, Rotational Speed, Torque and Tool Wear. In our analyses, we disregard the Failure Type due to the limited occurrences of each specific failure type in the dataset, making it unfeasible for our models to effectively classify them.

2 System Design

2.1 Data Pre-Processing

For data pre-processing, several common steps were undertaken across each algorithm to ensure consistency and comparability:

Mapping Dictionary Creation: A mapping dictionary was established to translate categorical variables ('L', 'M', and 'H') into numerical values (0, 1, and 2), respectively. This step facilitated the handling of categorical data within the algorithms.

Dataset Splitting: The dataset was split into training and testing subsets in a way that ensures a balanced representation of "no-

failure" and "failure" samples within each set. For the perceptron algorithm, an 80-20 split was employed, allocating 80% of the data for training and reserving 20% for testing. In contrast, for both the neural network and SVM algorithms, a more elaborate split was utilized. Here, 80% of the data was allocated for training, with 10% designated for validation and the remaining 10% set aside for testing. This strategy ensured adequate evaluation of model performance while avoiding data leakage.

SMOTE Oversampling: To address class imbalance, SMOTE (Synthetic Minority Over-sampling Technique) oversampling was employed for both the SVM and neural network algorithms. This technique artificially increased the representation of minority classes, specifically the 'failed' samples, thereby enhancing the models' ability to generalize and make accurate predictions.

2.2 Perceptron

Perceptron learning offers a straightforward yet powerful approach to binary classification tasks. By iteratively adjusting weights to minimize classification errors, perceptrons can effectively discern patterns in high-dimensional data, making them well-suited for simple decision-making processes in predictive maintenance scenarios. First, we carry out the data pre-processes, then we establish the seed for random number creation to ensure that the results are reproducible. The model is then trained using scikit-learn's Perceptron, and the seed for random number generation is reset for reproducibility. Then, using the training and testing data, we construct predictions about binary targets. The metrics module from scikit-learn is then used to construct the model's assessment metrics using training and testing data with binary targets. Then we compute the accuracy of solely failure samples for the binary target, which is very low because the model predicts largely zeros (i.e no-failure) for this unbalanced dataset. Finally, we compute and plot the Precision-Recall curve and the Area under the curve.

2.3 SVM

Support Vector Machines (SVM) excel in handling complex datasets by identifying optimal hyperplanes that separate different classes with maximum margin. By harnessing SVM's ability to handle both linear and non-linear data, we can enhance the predictive accuracy of our maintenance classification model, particularly in scenarios where data exhibits intricate relationships.

To implement SVM model for our objective, we first tune the hyper-parameters that define the model specifically the model regularization term and kernel coefficient. While the model regularization term (C), controls the penalty for misclassification of the training examples, the kernel coefficient (gamma) determines the influence of a single training example on the decision boundary. Therefore, a parameter grid is defined for C: [0.1, 1, 10] and gamma: [0.1, 0.01, 0.001]. Next, an SVM classifier is instantiated, serving as the base estimator for the grid search process.

Subsequently, GridSearchCV from scikit-learn is utilized to conduct hyper-parameter tuning. The grid search is performed using 9-fold cross-validation and 'accuracy' as the scoring metric. The grid search is executed on the unbalanced training dataset. Once the search completes, the best combination of hyper-parameters is printed. This enables identification of the optimal hyper-parameters for the SVM classifier when trained on the unbalanced dataset.

Then a Support Vector Machine classifier is trained using the optimal hyperparameters obtained from the previous grid search. The classifier is trained with a radial basis function (RBF) kernel, with the probability parameter set to "True" to enable probability estimates.

After training the classifier, its performance is assessed by computing the training accuracy on the unbalanced training data, followed by evaluating its accuracy on the test dataset. Furthermore, separate accu-

racy scores are calculated for the failure-only and no-failure-only subsets of the test dataset to gauge the classifier's performance on each class individually.

Also, the F1 score, precision, and recall are computed to assess the classifier's overall performance. Additionally, the precision-recall curve and the area under the precision-recall curve (AUC-PR) are plotted to provide insight into the classifier's performance across different thresholds.

The performance of SVM models can be significantly affected by the balance of the dataset. SVM models trained on balanced data tend to perform well because they are not biased towards any particular class. To demonstrate this effect, a new SVM model was trained by repeating the above described process for a balanced dataset.

2.4 Neural Network

Neural Networks would serve as our primary solution as it offers unparalleled flexibility and adaptability to diverse datasets. By constructing deep neural architectures, we can capture complex patterns within the maintenance data, enabling more accurate predictions. The inherent ability of neural networks to learn complex representations from raw data makes them particularly suitable for predictive maintenance tasks, where the relationships between variables may be nonlinear and dynamic.

First, we carry out the data pre-processes. Afterwards, the data is normalized using the mean and standard deviation of the training data to ensure consistent scaling across all features. This normalization step helps improve the convergence and stability of the training process. The neural network model for binary classification is then defined. It consists of an input layer with 6 neurons (corresponding to the number of features), followed by two hidden layers with 64 and 32 neurons, respectively, and an output layer with 1 neuron. The model uses ReLU activation functions for the hidden layers and a sigmoid activation function for the output layer. During the training loop, the model is trained for 20 epochs using the Adam optimizer with a learning rate of 0.001. Binary cross-entropy loss is employed as the loss function to measure the discrepancy between predicted and actual labels.

After training, the model is evaluated on the validation and test datasets to assess its performance in terms of loss and accuracy. The evaluation is conducted using batch sizes of 8 for the unbalanced dataset and 16 for the balanced, ensuring efficient processing of data for both the validation and test data loaders. Additionally, the script includes a function to evaluate the model's performance specifically on failure samples within the test dataset. This evaluation provides insights into the model's ability to detect failures accurately.

Finally, the precision-recall curve is computed and plotted to evaluate the precision and recall trade-off of the model, providing additional insights into its performance. The determination of batch sizes and learning rates for both balanced and unbalanced datasets was conducted through the grid search algorithm. This method explored various combinations of hyper-parameters, selecting the optimal settings that minimized validation loss. This Neural Network design was applied to both the unbalanced and balanced datasets.

3 Results

In comparing the algorithms, key metrics including test accuracy for failure samples, F1 score on the test set, and the Area under the precision-recall curve were employed. While training and overall test accuracy are commonly used, they can sometimes present misleading impressions, especially in cases of class imbalance. These metrics were selected for their ability to provide insightful evaluations, particularly in scenarios of class imbalance. By incorporating these measures, a comprehensive assessment was achieved, showing the

models' capacity to accurately identify failure cases, overall predictive performance, and balance between precision and recall. Below, the algorithm results are presented with accuracy and f1-score values expressed as percentages for clarity.

3.1 Perceptron

Metric	Value
Training Accuracy	96.9
Testing Accuracy	97.35
Test Accuracy (Failure Samples)	36.07
F1 Score (Test set)	45.36
AUC-PR	0.4196

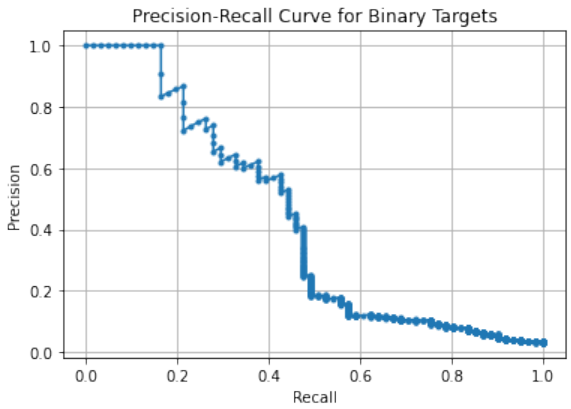

Figure 1: Perceptron Model

3.2 SVM

The best parameters obtained for the unbalanced training data were C = 10 and γ = 0.001. Below is the table of results:

Table 1: Performance Metrics for Unbalanced Training Data

Metric	Value
Training Accuracy	70
Test Accuracy	97.50
Test Accuracy (Failure Samples)	35
F1 Score (Test set)	48.97
AUC-PR	0.5019

For the balanced training data, the best parameters were C = 10 and γ = 0.01.

Table 2: Performance Metrics for Balanced Training Data

Metric	Value
Training Accuracy	99
Test Accuracy	93.90
Test Accuracy (Failure Samples)	44
F1 Score (Test set)	32.96
AUC-PR	0.2856

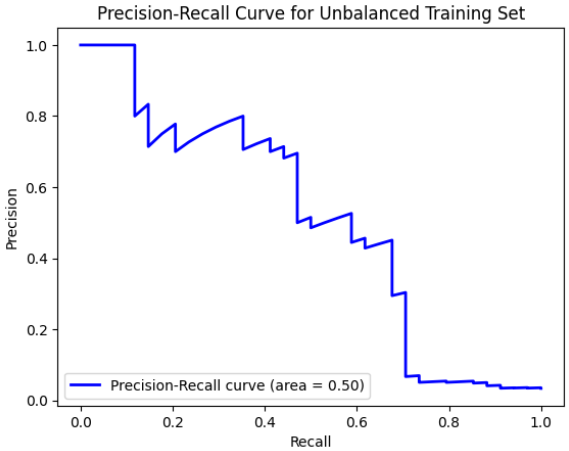

Figure 2: SVM Unbalanced

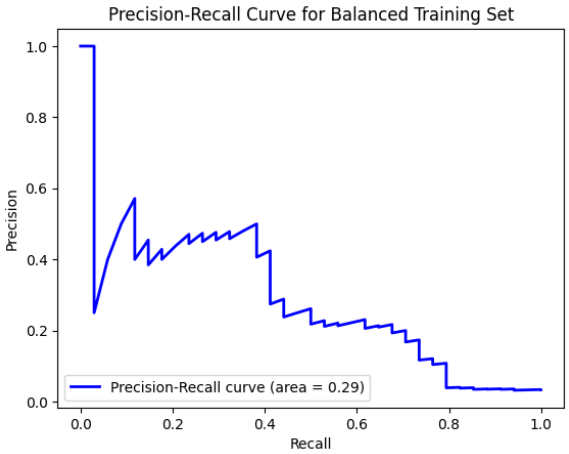

Figure 3: SVM Balanced

These results demonstrate the impact of balancing the training dataset on the performance of the SVM classifier. Despite achieving lower accuracy on the balanced dataset,

the classifier shows improved performance in terms of precision, recall, and F1 score compared to the unbalanced dataset. Moreover, the AUC-PR indicates the effectiveness of the classifier in capturing the trade-off between precision and recall.

3.3 Neural Network

Table 3: Performance Metrics for Unbalanced Dataset

Metric	Value
Train Accuracy	99.2
Test Accuracy	98.00
Test Accuracy (Failure Samples)	58.82
F1 Score (Test set)	66.67
AUC-PR	0.6922

Table 4: Performance Metrics for Balanced Dataset

Metric	Value
Training Accuracy	97.81
Test Accuracy	94.81
Test Accuracy (Failure Samples)	76.47
F1 Score (Test set)	52.75
AUC-PR	0.7206

These results demonstrate the effectiveness of the neural network algorithm in handling both unbalanced and balanced datasets, with particular success in capturing failure samples, as indicated by the higher F1 score and accuracy on failure samples compared to the previous algorithms.

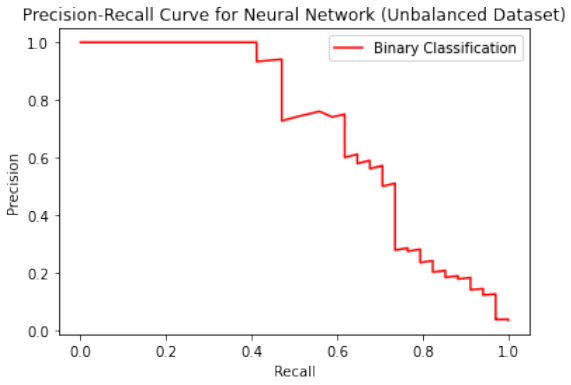

Figure 4: Neural Network Unbalanced

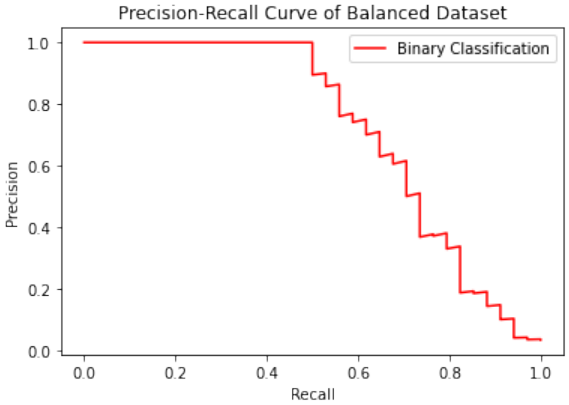

Figure 5: Neural Network Balanced

4 Conclusion

Our analysis illustrates the superior performance of neural networks compared to other algorithms such as perceptron learning and support vector machines (SVM) in predictive maintenance tasks, as shown by higher F1 scores and AUC-PR values. While the perceptron algorithm exhibits limited efficacy in handling complex datasets due to its inherent simplicity, it often struggles with imbalanced class distributions. Similarly, the SVM algorithm demonstrates suboptimal performance in this context. In contrast, the robustness and adaptability of neural networks make them highly effective tools for forecasting maintenance needs.

References

[1] S. Matzka. Predictive maintenance dataset. *International Conference on Artificial Intelligence for Industries*, 2020. doi: 10.24432/C5HS5C. URL https://archive.ics.uci.edu/ml/datasets/AI4I+2020+Predictive+Maintenance+Dataset.