Beaux Consunji, Lawrence Paul Marino, Joshua Zapico

CSCI 114: ML PREDICTIVE MAINTENANCE

I. INTRODUCTION

Predictive maintenance is a proactive strategy that leverages machine learning to predict when machinery is likely to fail, enabling timely maintenance and reducing unplanned downtime. This project focuses on developing a predictive maintenance model using a dataset sourced from Kaggle.

The goal is to optimize machine reliability and operational safety through accurate predictions. The primary objective of this project is to identify patterns in machine sensor data that signal failures. By doing so, we aim to minimize equipment downtime and maintenance costs. Predictive maintenance shifts the focus from reactive to proactive strategies, enhancing the reliability of industrial operations. Our study seeks to build and evaluate machine learning models capable of achieving these goals while ensuring practical applicability. Ideally, we visualize this model being implemented in the renewable energy sector, where reducing downtime and improving consistency can contribute significantly to the stability and reliability of energy sources in the Philippines.

II. METRICS AND GOALS

To evaluate the performance of the models, we primarily aim to predict at least 90% of the machine status correctly. In predictive maintenance, we want the model to perform well across all machine states to balance operational efficiency with failure detection. This is

because while detecting failures is critical, accurately identifying non-failures is also essential to avoid unnecessary downtime and maintenance costs.

III. DATASET

The dataset used in this project is sourced from Kaggle and includes machine sensor readings, operational settings, and failure events. This data provides a foundation for identifying patterns that predict failures. Preprocessing steps include data cleaning to handle missing values, feature engineering to derive new insights such as rolling averages, and normalization to scale features for improved model performance. While the dataset is comprehensive, it has certain limitations. These include class imbalance, as failure events are rarer than normal operations, and limited machine diversity, which may restrict the generalizability of the models to other industries or equipment types. In addition, the dataset is also synthetic, which means that it may not be representative of data from machines in reality.

Pre-Processing

Old Dataset

Name of Column	Data Type	Expected Values	Description
UDI	int64	Unique positive integers (e.g., 1 to 10,000).	Unique identifier for each entry.
Product ID	object	Strings representing unique product IDs.	Identifier for the specific product.

Type	object	H, L, or M.	Categorical feature denoting the type of the product.
Air temperature [K]	float64	Positive decimals, ranging approximately from 290 to 310 Kelvin.	Temperature of the air surrounding the manufacturing environment.
Process temperature [K]	float64	Positive decimals, ranging approximately from 300 to 320 Kelvin.	Temperature during the manufacturing process.
Rotational speed [rpm]	int64	Positive integers, ranging from approximately 800 to 3,000 RPM.	Speed of rotation in revolutions per minute.
Torque [Nm]	float64	Decimals, ranging approximately from 0.5 to 80 Newton-meters.	Torque applied by the machine.
Tool wear [min]	int64	Positive integers, ranging from 0 to 250 minutes.	Duration of tool wear.
Target	int64	Binary values: 0 (No failure) or 1 (Failure).	Indicates failure occurrence: 0 for no failure, 1 for failure.
Failure Type	object	None, Heat Dissipation Failure, Power Failure, Overstrain Failure, etc.	Categorical description of the failure type, or None if no failure occurred.

1. Column Removal:
    - Removed Columns: UDI, Product ID, Failure Type, and Type from the original dataset were removed, likely to reduce irrelevant or unnecessary identifiers for predictive modeling.
    - Justification: These columns are contributing little to no predictive analytics or and are often just redundant.
2. Row Handling:
    - Removed Nulls: Any rows with null values in relevant columns were likely dropped. This reduced the data from 10,000 rows to 9,973.
3. Feature Scaling:
    - Continuous variables (Air temperature [K], Process temperature [K], Rotational speed [rpm], Torque [Nm], Tool wear [min]) have been normalized/standardized.

New Dataset

Name of Column	**Data Type**	**Expected Values**	**Description**
Air temperature [K]	float64	Decimals, ranging approximately from -2.35 to 2.25.	Temperature of the air surrounding the manufacturing environment.
Process temperature [K]	float64	Decimals, ranging approximately from -2.90 to 2.56.	Temperature during the manufacturing process.

Rotational speed [rpm]	float64	Decimals, ranging approximately from -2.06 to 7.50.	Speed of rotation in revolutions per minute.
Torque [Nm]	float64	Decimals, ranging approximately from -3.63 to 3.67.	Torque applied by the machine in Newton-meters.
Tool wear [min]	float64	Decimals, ranging approximately from -1.69 to 2.27.	Duration of tool wear, in minutes.
Target	int64	Binary values: 0 (No failure) or 1 (Failure).	Indicates failure occurrence: 0 for no failure, 1 for failure.

IV. MODEL DEVELOPMENT

Feedforward Neural Network

Rationale

Feedforward Neural Networks (FNNs) are well-suited for predictive maintenance because they can capture complex relationships in data. Equipment failures often result from various interacting factors, such as temperature changes, wear, and operational stress. FNNs can effectively model these interactions, allowing them to detect subtle patterns that may indicate potential failures. Additionally, FNNs are scalable and adaptable, making them a good

choice for predictive maintenance, where sensor data from machines is continuously collected and analyzed.

Description

A Feedforward Neural Network (FNN) is a type of neural network where information flows in one direction from input to output, without any feedback loops. The network consists of several layers: an input layer, multiple hidden layers, and an output layer. During training, the model learns the relationships between input features and the target by adjusting the weights of connections between neurons through backpropagation.

To enhance the FNN model's performance for predictive maintenance, several techniques were applied. First, the dataset was balanced by downsampling the majority class (non-failures), which improved the model's ability to detect failures. Feature normalization using StandardScaler ensured the data had consistent scales, promoting stable and faster training. The use of Binary Cross-Entropy Loss helped the model focus on optimizing the detection of failures. A higher classification threshold of 0.95 was set to reduce false positives by ensuring only highly confident predictions were classified as failures. The model architecture included multiple hidden layers and ReLU activation, enabling the network to learn more complex patterns and improving its ability to distinguish between failure and non-failure instances.

Results

Table 1. Table showing the evaluation metric results of the Feedforward Neural Network

Accuracy	Precision	Recall	f1
0.9488	0.4099	0.8584	0.5548

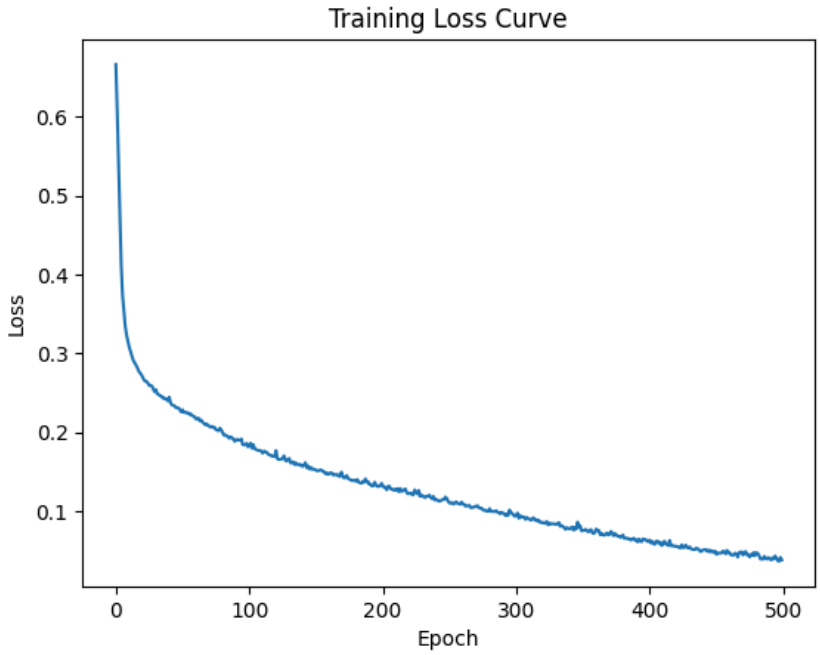

Figure 1. Table showing the table loss curve of the Feedforward Neural Network

The performance of the feedforward neural network demonstrates that the model is highly effective at identifying failure events. However, there is a significant issue with false positives, where the model mistakenly classifies non-failures as failures. This imbalance can be problematic in predictive maintenance, as false alarms may lead to unnecessary maintenance actions. To improve the model's performance, techniques such as adjusting the classification threshold or refining the model with additional features could be explored to reduce the occurrence of false positives.

Autoencoder

Rationale

Autoencoders are a type of unsupervised neural network designed to learn a compressed representation (latent space) of the features during the encoding phase and

reconstruct it during the decoding phase. Since labelled failure data is scarce, using autocoders can be ideal for predictive maintenance as it can be trained on the more common normal machine operation data to learn a baseline for non-failure behavior, enabling it to identify deviations as failures.

Description

The autoencoder model has 1 encoder layer, which compresses the 5 input features into a latent representation of 2 dimensions, and 1 decoder layer, which reconstructs the original features with 5 nodes from the latent representation. It uses a Stochastic Gradient Descent (SGD) as an optimizer to minimize the reconstruction loss. The reconstruction loss is calculated using Mean Squared Error (MSE) and the model aims to make encodings that minimize this for non-failure data, so that it can classify failures as those with a reconstruction loss that is higher than a certain threshold. This threshold is tuned by evaluating the model when setting the threshold to the i-th quantile of the error values for 10 iterations and selecting the threshold which maximizes the average accuracy of the predictions. The model is also trained over 10 epochs with a batch size of 1.

Results

Table 2. Table showing the evaluation metric results of the autoencoder model

Accuracy	Precision	Recall	f1
0.90	0.58	0.71	0.60

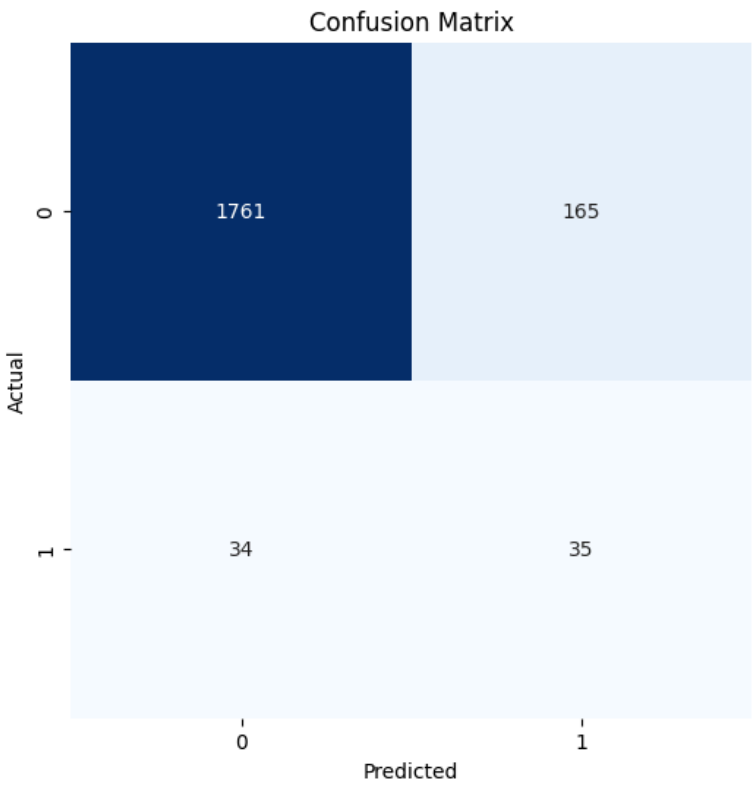

Figure 2. Confusion matrix showing the number of true negatives, false positives, false negatives, and true positives of the autoencoder model

The model had an accuracy of 0.90 which means that it achieves our goal of predicting at least 90% of machine states correctly.

Decision Tree

Description

This project uses a Decision Tree Classifier, a type of machine learning model that helps us make predictions by asking a series of yes-or-no questions based on the data. The dataset includes measurements like air temperature, process temperature, rotational speed, torque, and

tool wear, all of which are used to predict whether an event, such as equipment failure, will occur. The data was split into two parts: one for teaching the model and another for testing how well it performs. We adjusted the model's settings to make it as reliable and balanced as possible, especially since there are fewer failure events compared to normal operations. The model's performance was evaluated by how well it identifies potential failures while minimizing unnecessary alerts, ensuring it provides actionable and trustworthy results.

Rationale

The Decision Tree was chosen because it is easy to understand, as it also shows its decisions in a way people can make sense of. It is especially useful in tasks such as predicting failure of equipment where knowing why that prediction was made can be used to improve operations. This type of data fits well with the model, and the relationships between factors like temperature and tool wear are very complicated, but the model handles it. We fine tuned the settings and addressed the uneven data problem (one outcome happens much more than the other) to make that model effective and practical for real world use.

Results

Table 3. Table showing the evaluation metric results of the decision tree model

Accuracy	Precision	Recall	f1
0.9804	0.7076	0.6969	0.7022

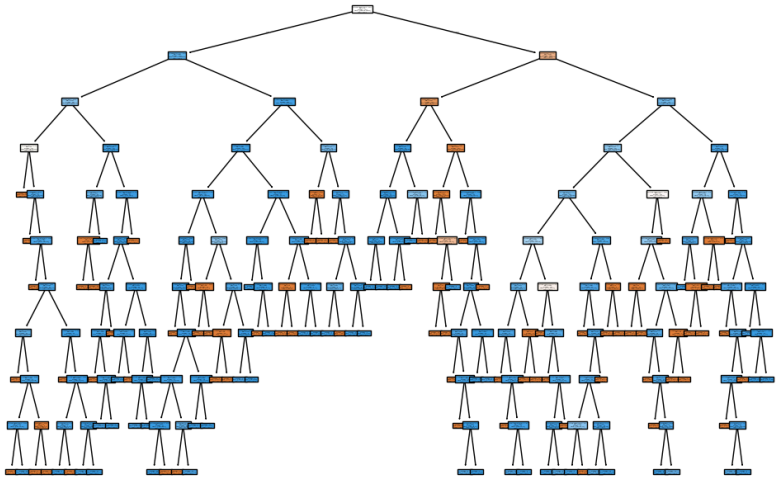

Figure 3. Decision Tree plot of the decision tree model

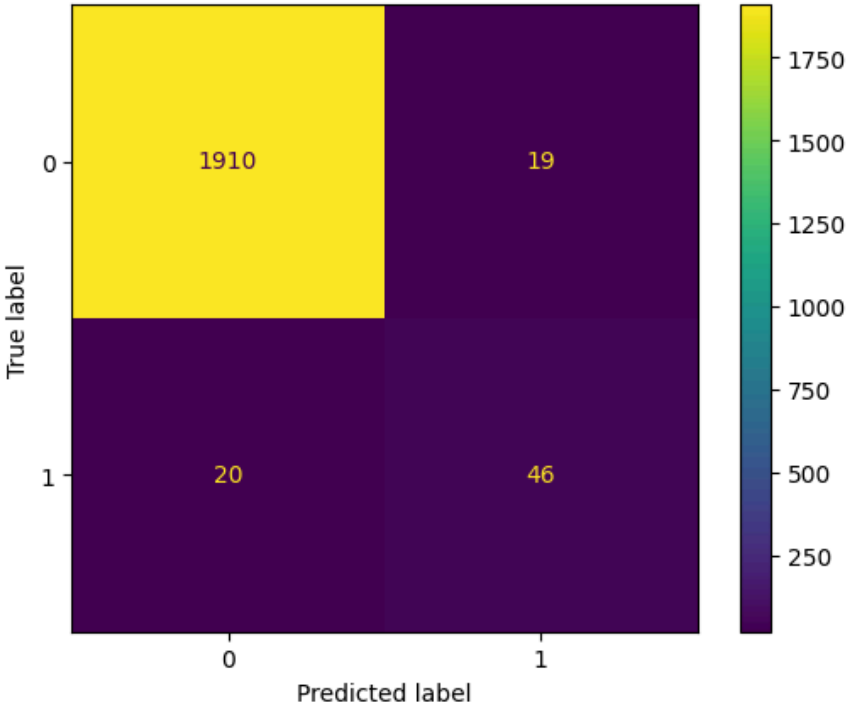

Figure 4. Confusion matrix showing the number of true negatives, false positives, false negatives, and true positives of the decision tree model

The results show that the model works very well at correctly identifying most of the data points in the dataset. It works well at predicting when the equipment is liable to fail or not, but

has a hard time with a handful of adverse cases. The model performs well when it predicts a failure, but there may be some cases where it sends out an alert that fails. That means the system is typically good at finding possible problems without being overly concerned. And it also does a good job in making sure it doesn't miss too many critical failures, which reduces the risk of unannounced downtime. Overall, this balance of catching true problems without generating too many alarms makes the system usable for stakeholders that need to see consistent and actionable insights about the performance of equipment.

XGBoost

Rationale

One of the main advantages of XGBoost in predictive maintenance is its ability to handle both continuous and categorical data efficiently. In predictive maintenance scenarios, datasets often include a mix of numerical sensor readings and categorical variables, such as machine history. XGBoost's ability to work seamlessly with mixed data types allows it to leverage all the information, improving accuracy on failure predictions. XGBoost also has the ability to provide insights into feature importance, which is valuable in predictive maintenance, as understanding which variables influence the likelihood of failure the most can help guide maintenance actions.

Description

XGBoost is a machine learning algorithm that improves predictions by building decision trees in sequence, with each tree correcting the mistakes of the previous one. This approach enhances the model's performance, especially in tasks like classification and regression. Known for its speed, efficiency, and accuracy, XGBoost is widely used in real-world applications.

To improve the XGBoost model's performance, several strategies were used. First, the issue of class imbalance was addressed by adjusting the model to give more importance to the minority class (failures). Next, the model's ability to classify failures was enhanced by tuning specific parameters. A higher confidence threshold was set for predicting failures, ensuring more accurate predictions. The model's settings were fine-tuned to improve overall performance. Finally, the model was evaluated using various metrics to ensure its reliability in predicting failures, minimizing errors, and making it well-suited for predictive maintenance tasks.

Results

Table 4. Table showing the evaluation metric results of the XGBoost model

Accuracy	Precision	Recall	f1
0.9855	0.7937	0.7576	0.7752

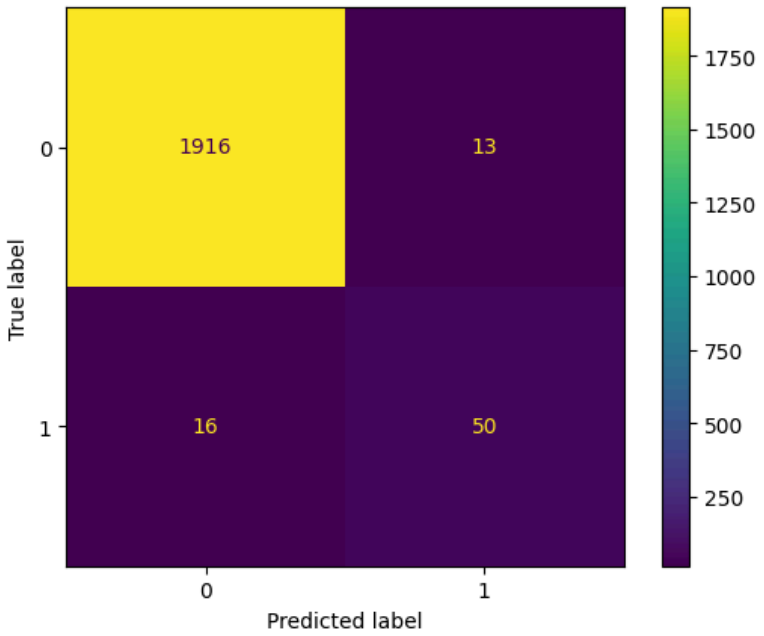

Figure 4. Confusion matrix showing the number of true negatives, false positives, false negatives, and true positives of the XGBoost model

The XGBoost model performs well, correctly predicting most outcomes. It is quite accurate when it predicts a failure, predicting failures 98.55% of the time. Though, there is still some room to reduce incorrect positive predictions. It also does a good job of identifying most actual failures, but it misses a few. Overall, the model strikes a good balance, making it a reliable choice for predictive maintenance tasks, offering a sensible trade-off between catching true failures and avoiding false alarms.

V. CONCLUSIONS AND RECOMMENDATIONS

The XGBoost model outperforms the other models, providing the best overall performance in predicting both positive and negative outcomes. It is particularly effective at balancing the detection of both classes. The decision tree follows closely behind, performing well but slightly less balanced than XGBoost. The feedforward neural network shows a tendency to focus more on predicting one class, leading to a less balanced performance. The autoencoder, while having decent results, performs the weakest among all the models.

In summary, XGBoost is the top choice for reliable predictions. If needed, the decision tree could be further improved, while the feedforward neural network and autoencoder might require additional adjustments for better performance.

SOURCES

Brownlee, J. (2020, November 26). How to tune the number and size of decision trees in XGBoost with Python. *Machine Learning Mastery*. https://machinelearningmastery.com/tune-number-size-decision-trees-xgboost-python/

Science Publishing Group. (2024). *A study on machine learning and its applications*. *Modern Machine Learning Research*, 2(4), 15. https://sciencepublishinggroup.com/article/10.11648/j.mlr.20240902.15

Stormborn, A. (n.d.). *Anomaly detection with autoencoders*. GitHub. Retrieved December 7, 2024, from https://github.com/AarnoStormborn/anomaly-detection-with-autoencoder/blob/main/anomaly-detection-with-autoencoders.ipynb

Zhu, Y., & Ding, X. (2019). Data-driven quality management for predictive maintenance in manufacturing: A review and research directions. *International Journal of Quality & Reliability Management*, 36(9), 1509-1535. https://doi.org/10.1108/IJQRM-04-2019-0131