A black background with white text

Description automatically generated with low confidence

MSc Data Science Project

7PAM2002-0901-2024

Department of Physics, Astronomy and Mathematics

**Data Science FINAL PROJECT REPORT**

**Project Title:**

**Enhancing Equipment Reliability with Machine Learning-Based Predictive Maintenance**

**Student Name and SRN:**

Manikumar Duddipudi (22086809)

**Supervisor:** Carolyn Devereux

**Date Submitted:** 28-08-2025

**Word Count: 4939**

**GitHub Link :** https://github.com/Manikumar1999/Predictive\_maintenance\_project

DECLARATION STATEMENT

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science in Data Science at the University of Hertfordshire.

I have read the guidance to students on academic integrity, misconduct and plagiarism information at [Assessment Offences and Academic Misconduct](https://www.herts.ac.uk/__data/assets/pdf_file/0007/237625/AS14-Apx3-Academic-Misconduct-v17.0.pdf) and understand the University process of dealing with suspected cases of academic misconduct and the possible penalties, which could include failing the project module or course.

I certify that the work submitted is my own and that any material derived or quoted from published or unpublished work of other persons has been duly acknowledged. (Ref. UPR AS/C/6.1, section 7 and UPR AS/C/5, section 3.6).

I have not used chatGPT, or any other generative AI tool, to write the reportor code (other than where declared or referenced).

I did not use human participants or undertake a survey in my MSc Project.

I hereby give permission for the report to be made available on module websites provided the source is acknowledged.

Student SRN number: 22086809

Student Name printed: Manikumar Duddipudi

Student signature: Manikumar Duddipudi

UNIVERSITY OF HERTFORDSHIRE

SCHOOL OF PHYSICS, ENGINEERING AND COMPUTER SCIENCE

**Acknowledgement**

My deepest thanks goes out to everyone who helped and advised me during this project. I am incredibly grateful for the priceless advice, unwavering support, and helpful criticism I got at every turn during this research, all of which enabled me to polish my concepts and finish this project effectively.

I am also appreciative to my friends, classmates, and university professors for their support and encouragement during this academic endeavor. Lastly, I want to express my sincere gratitude to my family for their constant belief in me and support, which allowed me the will and courage to complete this project.

**Abstract**

Predictive maintenance has become an essential strategy in modern industries to minimize downtime, reduce costs, and improve equipment reliability. This project explores the application of machine learning models XGBoost, CatBoost, LightGBM, and MLPClassifier on the AI4I 2020 predictive maintenance dataset. Data preprocessing and balancing techniques were applied to address class imbalance and improve model reliability. The boosting-based models (XGBoost, CatBoost, and LightGBM) achieved 98% accuracy, while the MLPClassifier reached 96%. Comparative analysis revealed that although existing literature highlights XGBoost as a leading method, CatBoost produced the most reliable results with fewer misclassifications. These findings confirm that boosting algorithms are well-suited for predictive maintenance tasks, with CatBoost emerging as the most effective model for real-world industrial deployment.

**Contents**

[1. Introduction](#_3dy6vkm) 6

[1.1 Introduction](#_1t3h5sf) 6

[1.2 Inspiration](#_u5awihlk3qin) 6

[1.3 Aim](#_intzs3t5l6ee) 7

[1.4 Research questions](#_djvz9hp4khl) 7

[1.5 Objectives](#_n0xpq767dil4) 7

[1.6 Scope of the Project](#_6lvf2slrvvw1) 7

[1.7 Thesis organization](#_35nkun2) 8

[2. Literature review](#_x8j24ut9yxrj) 9

[3. Methodology](#_8v3md3h4quoy) 12

[3.1 An Overview in Brief](#_s4rch3nww71b) 12

[3.2 Description of the Data](#_4i7ojhp) 12

[3.3 Data Pre-Processing](#_nn2tn16fhq92) 12

[3.4 Classification Models](#_8zt14kege88b) 12

[3.4.1 XGBoost Classifier](#_40qm9h6bh0n6) 13

[3.4.2 CatBoost Classifier](#_6kpqhe614zgj) 14

[3.4.3 LightGBM Classifier](#_qtkjo0xeegdi) 15

[3.4.4 Multi-Layer Perceptron (MLPClassifier)](#_aiuq7himhfe) 16

[3.5 Model Evaluation Techniques](#_jsubp16jtwx6) 17

[3.6 Consideration of Ethical, Social, Legal, and Professional Issues](#_nxg6qkwp2vo1) 18

[4. Results](#_kpil99jou7ac) 19

[4.1 Comparative Model Performance](#_xh8km5ccuh3g) 19

[4.2 Comparison with Literature](#_sexbfvgfped6) 20

[4.3 Effect of Dataset Balancing](#_gz028uxee3ki) 21

[4.4 Real-World Application](#_uruve8q5wgiz) 21

[5. Conclusion](#_qf0cz5rr5kp3) 22

[6. References](#_tfpl1sm83acy) 23

[7. Appendix](#_9t77zaxpm5us) 25

**List of Figures:**

Fig 1: XGBoost classifier's architecture………………………………………………...……12

Fig 2: CatBoost classifier's architecture ……………………………………………………..13

Fig 3: LightGBM classifier's architecture…………………………………………………....14

Fig 4: Multi-Layer Perceptron classifier's architecture………………………………………15

Fig 5: MLP Accuracy and Loss Curves……………………………………………………...19

Fig 6: Confusion matrix for Catboost Classifier……………………………………………..19

**List of Tables:**

Table 1: Performance Comparison of ML models………………………………………...…18

# **Introduction**

## **1.1 Introduction**

Equipment failure in contemporary industrial settings can lead to serious financial losses, delays in production, and possible safety hazards. Conventional maintenance techniques like reactive maintenance, which fixes equipment when they break down, and preventive maintenance, which conducts planned inspections regardless of the state of the machine, are either ineffective or costly. These approaches often fail to leverage the growing availability of real-time sensor and operational data from industrial machinery.

Predictive Maintenance (PdM) offers a data-driven alternative by forecasting potential equipment failures before they occur. PdM helps industries anticipate the likelihood of machine failures and take preventative action by evaluating variables like temperature, rotational speed, torque, and tool wear (Musril et al., 2023). This improves the overall dependability and lifespan of machines, lowers maintenance expenses, and decreases unscheduled downtime.

In this project, a predictive maintenance system is developed using sophisticated machine learning methods. In addition to target labels indicating whether a machine failed and what kind of failure it was, the dataset contains features like air temperature, process temperature, rotational speed, torque, and tool wear. Extreme Gradient Boosting (XGBoost), CatBoost, Light Gradient Boosting Machine (LightGBM), and Multi-Layer Perceptron (MLP) are four potent models that are used and assessed in order to produce precise predictions (Kane et al., 2022). According to Bundasak et al. (2022), these models were chosen because of their shown capacity to manage complicated, high-dimensional, and unbalanced data.

The objective of this project is to compare the performance of these algorithms in predicting machine failures and classifying failure types. The results demonstrate how ensemble-based gradient boosting methods and neural networks can be leveraged for industrial predictive maintenance applications, contributing to improved decision-making, optimized maintenance schedules, and enhanced operational efficiency.

## **1.2 Inspiration**

The inspiration for this project stems from the challenges industries face with costly downtime, safety risks, and inefficiencies in traditional maintenance approaches. With the rise of IIoT and real-time sensor data, there is a strong opportunity to apply machine learning for Predictive Maintenance (PdM). This project explores advanced models XGBoost, CatBoost, LightGBM, and MLP to bridge the gap between conventional practices and intelligent, data-driven solutions that reduce failures, minimize costs, and move toward smarter, automated maintenance systems.

## **1.3 Aim**

The aim of this project is to build a predictive maintenance system that uses machine learning models (XGBoost, CatBoost, LightGBM, and MLPClassifier) to accurately predict machine failures, reduce downtime, and optimize maintenance scheduling.

## **1.4 Research questions**

* How can machine learning be applied to improve the early prediction of equipment failures and support proactive maintenance scheduling in industrial environments?
* Which machine learning models achieve the best performance and reliability for predictive maintenance under varying equipment types and operational conditions?

## **1.5 Objectives**

**To collect and preprocess the AI4I 2020 dataset**, ensuring data quality through cleaning, normalization, and feature engineering for predictive maintenance tasks (Predictive Maintenance Dataset (AI4I 2020), 2022).

**To address class imbalance** using appropriate resampling techniques to improve model learning on rare failure events.

**To apply dimensionality reduction with PCA** for efficient feature representation and reduced model complexity.

To create machine learning models for industrial settings in order to optimize maintenance schedules and forecast equipment faults.

**To evaluate and compare model performance**, focusing on ensemble methods (XGBoost, LightGBM, CatBoost) and neural networks (MLPClassifier) in terms of accuracy, reliability, and robustness.

To do a thorough analysis utilizing metrics like F1-score, recall, accuracy, and precision.

## **1.6 Scope of the Project**

This project focuses on developing a predictive maintenance framework using the AI4I 2020 dataset by performing data preprocessing, handling class imbalance, and applying PCA for dimensionality reduction. It involves implementing and comparing advanced machine learning models, including XGBoost, LightGBM, CatBoost, and MLPClassifier, to predict equipment failures in industrial settings. The evaluation is carried out using metrics such as accuracy, precision, recall, F1-score. The scope is limited to experimental analysis on the given dataset and does not extend to real-time deployment, but it provides a strong foundation for future applications in industrial environments.

## **1.7 Thesis organization**

The thesis introduces predictive maintenance, highlighting its importance, motivation, and objectives. It reviews prior research, outlines the methodology including data preprocessing and model selection, and explains the experimental setup with evaluation metrics. Results are presented using performance measures and confusion matrices, followed by a discussion, conclusion and directions for future research.

# **Literature review**

Paolanti et al. (2018) introduced a predictive maintenance framework for detecting and forecasting motor failures in industrial environments, relying primarily on the Random Forest algorithm. A data processing tool installed on the Azure Cloud platform was used in their study to evaluate real-time data collected from sensors, programmable logic controllers (PLCs), and industrial communication protocols. The study showed that Random Forest was highly effective in distinguishing machine conditions with strong predictive accuracy, demonstrating its reliability for condition monitoring tasks. Compared to conventional simulation-driven approaches, this method achieved superior performance, validating its practical applicability. Nonetheless, the research emphasized certain limitations, such as the complexity of integrating diverse data sources and the challenges of aligning acquisition systems with cloud-based analytics. The authors recommended future studies to focus on enhancing preprocessing techniques, testing alternative machine learning models for benchmarking, and building scalable solutions to extend predictive capabilities across a wider range of industrial scenarios.

A case study was carried out by Satwaliya et al. (2023) to assess the use of machine learning techniques such as Random Forest, Gradient Boosting, and Deep Learning for predictive maintenance in manufacturing management. The objective was to determine the capability of these models in anticipating equipment malfunctions before they escalated into critical issues. Findings indicated that integrating machine learning substantially enhanced the accuracy of failure prediction, leading to notable reductions in unplanned downtime and maintenance costs, while improving overall production efficiency. However, the research also identified several challenges, such as the reliance on high-quality labeled datasets, difficulties in embedding ML systems into existing maintenance frameworks, and the heavy computational requirements of deep learning techniques. The authors concluded that predictive strategies driven by machine learning can provide significant advantages for manufacturers, particularly when aligned with data-driven decision-making practices, and stressed the necessity of ongoing model monitoring and refinement to maintain effectiveness in dynamic industrial environments.

With an emphasis on the Random Forest algorithm, Paolanti et al. (2018) presented a predictive maintenance framework for electric motors and industrial machinery. A real-world industrial case study was used to evaluate their methodology. In this case study, data was collected from various sources, such as sensors, programmable logic controllers (PLCs), and communication protocols. Microsoft Azure Cloud services were then used to process the data. Data collection, system analysis, and machine learning model deployment were among the phases of the project. Results showed that Random Forest had a high predictive accuracy in determining machine states, which enhanced dependability and reduced unplanned equipment breakdowns. In addition to improving maintenance scheduling, this proactive approach lessened the financial impact of downtime. However, the study emphasized the constraints of using cloud-based infrastructures and the problems of managing heterogeneous data coming from various industrial systems. The authors came to the conclusion that predictive maintenance systems can be greatly strengthened and more reliable industrial operations supported when scalable cloud solutions like Azure are combined with strong machine learning techniques.

Samatas et al. (2021) Conducted an extensive survey to investigate new developments in predictive maintenance made possible by the convergence of Internet of Things (IoT), artificial intelligence (AI), and machine learning technologies. Their analysis showed that, with 54.55% of the studies examined across several sectors, the manufacturing sector dominated research efforts. Among the most widely adopted machine learning models were Artificial Neural Networks (ANNs) (28.95%), Support Vector Machines (SVMs) (18.42%), and Random Forests (RFs) (14.47%). The study further classified twelve sensor categories employed in IoT-driven predictive maintenance systems, with temperature sensors (60.71%) and vibration sensors (46.42%) emerging as the most common. These outcomes underline the strong industrial relevance of predictive maintenance, especially where sensor-based data streams are integrated with AI approaches. A significant challenge highlighted was the difficulty of synchronizing AI models with heterogeneous sensor data in real-time production environments. To address this, the authors recommended ongoing research into more robust and scalable AI–IoT integration strategies capable of adapting to varied industrial domains.

Amer et al. (2023) developed a predictive maintenance framework that utilized supervised machine learning models to anticipate equipment malfunctions and limit unexpected operational halts. The research tested a range of algorithms Random Forest, Support Vector Machine, K-Nearest Neighbors, Decision Tree, Logistic Regression, Naïve Bayes, and XGBoost—on real-time industrial equipment data. Results highlighted that Random Forest and XGBoost consistently produced superior outcomes, with XGBoost showing distinct advantages for smaller datasets due to its efficiency and accuracy, while maintaining performance levels close to Random Forest when applied to larger data volumes. The experiments verified the system’s capacity to detect early warning signs of failure, thereby helping reduce downtime. A key issue tackled in the study was determining the most suitable algorithm across different dataset sizes and operational scenarios. For future directions, the authors recommended advancing model optimization techniques and investigating real-time applications in varied industrial environments to enhance reliability and streamline maintenance practices.

Purnachand et al. (2021) examined the rising demand for advanced predictive maintenance in industries where equipment downtime translates into heavy operational and financial losses. Instead of relying on routine schedules, the study stressed the importance of building models that can estimate the probability of machine failures at specific points, allowing organizations to optimize when and how maintenance is performed. While the authors did not limit their discussion to a single algorithm, they highlighted that the choice of modeling technique should be influenced by real-world operational parameters such as machine utilization rates and service duration. A key contribution of the work was the emphasis on correctly formulating the prediction task and systematically collecting reliable data from diverse business processes. One major challenge identified was the inherent complexity of industrial systems, which often consist of interdependent subsystems, making predictive planning more difficult. The paper further argued that the true value of predictive maintenance lies not only in the development of models but also in the continuous evaluation of their outputs and the transformation of predictions into actionable strategies. Ultimately, the authors recommended that modeling approaches should be selected with careful consideration of equipment operating behavior, lifecycle stage, and the broader industrial context.

# **Methodology**

## **3.1 An Overview in Brief**

The predictive maintenance project used the AI4I 2020 Predictive Maintenance Dataset to forecast equipment failures. After exploring the data, I applied preprocessing steps including outlier removal, label encoding, MinMax scaling, PCA for dimensionality reduction, and SMOTE for class balancing. Four models XGBoost, CatBoost, LightGBM, and MLP were trained and evaluated using classification reports and confusion matrices. This end-to-end pipeline showed how careful preprocessing and advanced algorithms improve the reliability of failure prediction in industrial machines.

## **3.2 Description of the Data**

The AI4I 2020 Predictive Maintenance Dataset, a synthetic dataset based on an actual milling machine, is used in this study. It covers machine breakdowns and operating circumstances and has 10,000 records with 14 attributes. A UDI, Product ID, and product quality type (Low 50%, Medium 30%, High 20%) are included with every entry.

Tool wear, torque, rotating speed, air temperature, and process temperature are important characteristics. Tool wear failure, heat dissipation failure, power failure, overstrain failure, and random failure are the five mechanisms that might cause a failure. The target variable is binary (failure/no failure).

Since the dataset only indicates failure occurrence (without specifying the exact type), it provides a realistic and challenging benchmark for predictive maintenance classification.

## **3.3 Data Pre-Processing**

Before model training, I focused on thorough preprocessing to improve data quality. Outliers in rotational speed and torque were removed using the IQR method since they could mislead models with unrealistic patterns. The categorical “Type” column was label-encoded to make it numerical.

I applied Min-Max scaling so that continuous features were on the same range, preventing any one variable from dominating. To further optimize the dataset, I used PCA, which reduced the feature space to five components. This step helped remove redundancy, reduced noise, and prevented overfitting while still preserving most of the variance.

Since failure cases were much fewer than non-failures, I balanced the target variable with SMOTE to ensure the models learned both classes effectively. Finally, I split the dataset into 70–30 train-test sets.

These steps made the dataset clean, balanced, and efficient, allowing models like XGBoost, CatBoost, LightGBM, and MLP to achieve reliable performance.

### **3.4 Classification Models**

For this project, I tested four models XGBoost, CatBoost, LightGBM, and MLP. I chose them to compare advanced gradient boosting methods with a neural network approach, as each offers distinct strengths for capturing complex patterns in predictive maintenance data. This allowed me to identify the most reliable model for the dataset.

### **3.4.1 XGBoost Classifier**

In this project, I implemented the Extreme Gradient Boosting (XGBoost) Classifier as one of the machine learning models. I chose XGBoost because predictive maintenance requires handling complex, non-linear relationships among sensor readings and operational variables. XGBoost is widely recognized for its accuracy, efficiency, and robustness on tabular data, making it a natural fit for this task.

XGBoost is an ensemble algorithm based on gradient boosting, where multiple decision trees are built sequentially to correct errors from previous trees. What makes it stand out is its use of both L1 (Lasso) and L2 (Ridge) regularization to reduce overfitting, along with optimizations like parallel processing, sparsity awareness, and efficient handling of missing values. These features make it highly suitable for practical applications such as predictive maintenance.

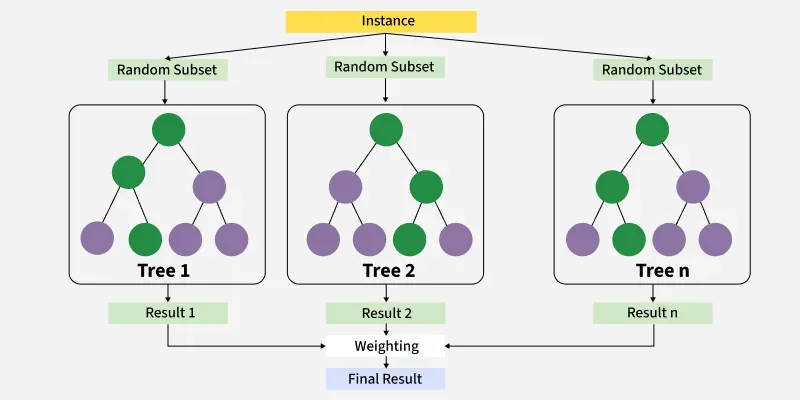


Fig 1: XGBoost classifier's architecture (Source: https://www.geeksforgeeks.org/machine-learning/xgboost/)

For implementation, I used the **XGBClassifier** from the Python library with default parameters. My goal was to establish a baseline before comparing it with CatBoost, LightGBM, and MLP. The workflow was simple: fit the model on the training set, predict on the test set, and evaluate using accuracy, precision, recall, and F1-score. Even with default settings, XGBoost performed strongly, confirming its reliability as an out-of-the-box solution.

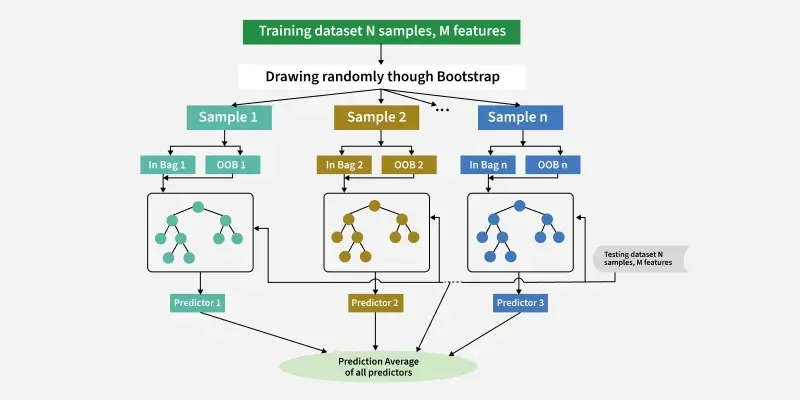
The model also provided feature importance, helping identify critical variables like torque, tool wear, and temperature in predicting failures. This interpretability is particularly valuable in real-world systems. While I did not tune hyperparameters such as n\_estimators, max\_depth, or learning\_rate, which could further improve results, the baseline implementation still proved effective.

Overall, XGBoost was a powerful, interpretable, and practical choice for predictive maintenance.

### **3.4.2 CatBoost Classifier**

Another model I implemented in this project was the **CatBoost Classifier**, a gradient boosting algorithm developed by Yandex. I chose CatBoost because of its ability to handle categorical features efficiently and deliver strong performance on tabular datasets. Although my dataset had limited categorical columns (mainly product type), I wanted to evaluate this model due to its growing popularity in predictive maintenance and industrial analytics.

CatBoost, short for Categorical Boosting, works on the principle of gradient boosting like XGBoost and LightGBM, but it introduces **ordered boosting** to prevent prediction shift and reduce overfitting. This technique makes CatBoost more robust in handling imbalanced datasets, which is often the case in predictive maintenance where machine failures are relatively rare.

Fig 2: CatBoost classifier's architecture (Source: https://www.geeksforgeeks.org/machine-learning/catboost-ml/)

Another major strength of CatBoost is its ability to process categorical variables without requiring one-hot encoding or heavy preprocessing. Instead, it internally converts categorical features using statistics-based encoding, making it practical for real-world datasets that include both numeric sensor data and categorical descriptors.

For implementation, I used the **CatBoostClassifier** from the catboost library, keeping the configuration simple with default parameters (and verbose=0 to suppress logs). I trained the model on the training dataset and evaluated predictions on the test set using accuracy, precision, recall, and F1-score. Even without hyperparameter tuning, CatBoost achieved strong results, showing its effectiveness as a low-effort, high-performance solution.

In summary, CatBoost was a valuable addition to my project. Its **ease of use, resistance to overfitting, and strong performance** made it a reliable model for predictive maintenance, suitable for both academic research and industrial deployment.

### **3.4.3 LightGBM Classifier**

Another classification model I implemented in this project was the Light Gradient Boosting Machine (LightGBM) Classifier, a highly efficient and scalable gradient boosting algorithm. I selected LightGBM because it is widely used for structured data and is well-suited for predictive maintenance tasks where both accuracy and speed are critical. Unlike XGBoost or CatBoost, LightGBM is specifically optimized for speed and memory efficiency, making it attractive for large-scale industrial datasets.

LightGBM builds on gradient boosting principles but introduces a **leaf-wise tree growth strategy**, where the leaf with the highest loss reduction is split first. This results in deeper trees that capture complex data patterns more effectively. It also uses innovations like histogram-based learning and exclusive feature bundling (EFB) to accelerate training and reduce memory consumption. These features make it ideal for predictive maintenance, which often involves continuous retraining on growing sensor data.

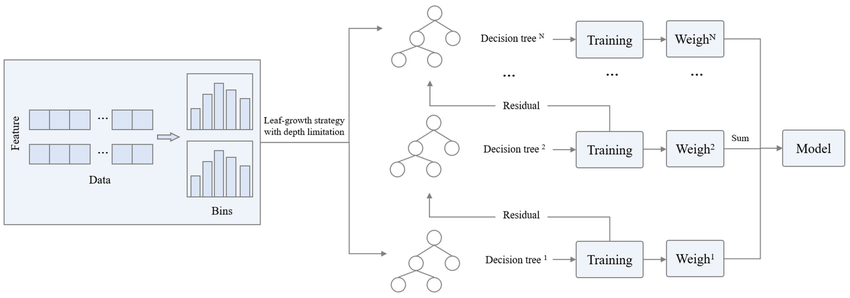


Fig 3: LightGBM classifier's architecture (Source: https://www.researchgate.net/figure/Schematic-diagram-of-LightGBM-algorithm\_fig4\_375811349)

Using largely default options, I implemented the LGBMClassifier from the lightgbm library. Metrics like accuracy, precision, recall, and F1-score were used to assess the model after it was trained on the AI4I predictive maintenance dataset. LightGBM demonstrated good performance right out of the box and offered competitive results even without parameter adjustment.

The main benefits were **fast training speed, efficiency, and scalability**, making it practical for real-world predictive maintenance systems that handle massive datasets. One limitation, however, is its tendency to overfit due to leaf-wise growth, particularly on smaller datasets. This can be controlled with parameters like max\_depth or num\_leaves.

In summary, LightGBM provided excellent performance, scalability, and efficiency, making it a strong candidate for predictive maintenance deployment.

### **3.4.4 Multi-Layer Perceptron (MLPClassifier)**

The Multi-Layer Perceptron (MLP) Classifier, a neural network-based method that can identify intricate, non-linear relationships in the data, was the last model I used for this project. Unlike tree-based models such as XGBoost, CatBoost, and LightGBM, which rely on decision tree ensembles, the MLP learns continuous feature representations. I chose this model because predictive maintenance data often contains subtle interactions between sensor readings and machine conditions that may not always be captured by boosting methods.

For implementation, I used scikit-learn’s MLPClassifier with a three-layer pyramidal structure of 128, 64, and 32 neurons. The ReLU activation function was applied for efficient training, while the Adam optimizer was selected for adaptive gradient updates. To improve generalization and reduce overfitting, I included L2 regularization (alpha=0.0005), adaptive learning rates, and early stopping. Training was capped at 500 iterations, ensuring sufficient convergence without excessive computation.

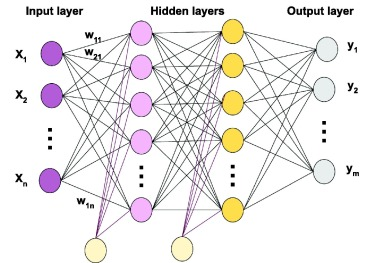


Fig 4: Multi-Layer Perceptron classifier's architecture (Source: https://www.sciencedirect.com/topics/computer-science/multilayer-perceptron)

The model was trained on the AI4I predictive maintenance dataset and evaluated using accuracy, precision, recall, and F1-score. The MLP achieved competitive results compared to boosting models, proving that deep learning can add value in this domain. Its main strength was flexibility: the network effectively captured non-linear dependencies and subtle variations in sensor data, which can be critical in identifying early failure signals.

However, the MLP required more computational effort and was less interpretable than boosting models an important consideration for real-world deployment. Despite these limitations, it complemented the other algorithms well and highlighted the potential of neural networks in predictive maintenance.

## **3.5 Model Evaluation Techniques**

In critical domains such as **predictive maintenance**, it is vital to thoroughly evaluate models to ensure that classification algorithms are both reliable and effective (Vujovic & Zeljko, 2021). Several performance metrics are commonly applied for this purpose, each providing unique insights into model behavior. The key measures used in this project are explained below, along with their mathematical representations.

**1. Accuracy:** Accuracy is defined as the model's proportion of overall correct predictions.

**Formula:**

**2. Precision (Positive Predictive Value):**

Precision reflects the fraction of correctly identified positive cases among all the instances the model predicted as positive.

**Formula:** Precision =

**3. Recall (Sensitivity, True Positive Rate):**

Recall indicates the model’s capability to correctly capture actual positive cases within the dataset.

**Formula:** Recall =

**4. F1 Score:**

By using their harmonic mean to combine precision and recall into a single figure, the F1-score offers a fair assessment.

**Formula:** F1 Score = 2 ×

This metric is highly beneficial when working with skewed class distributions, as it balances the trade-off between avoiding false positives and minimizing false negatives.

## **3.6 Consideration of Ethical, Social, Legal, and Professional Issues**

The Predictive Maintenance Project applies machine learning algorithms to forecast equipment failures and optimize maintenance using open-source Kaggle datasets. While the technical focus is on modeling and evaluation, it is also important to reflect on the ethical, legal, professional, and social aspects of such technologies.

**Ethical Considerations**Machine learning models may inherit bias if datasets are skewed or certain failure cases are underrepresented. In predictive maintenance, this could result in less accurate predictions for some equipment, affecting safety and efficiency (Kroll, 2018). The Kaggle dataset used contains only public, non-sensitive data, so privacy issues are minimal. However, imbalance in failure cases was addressed using SMOTE to improve fairness. As no personal data is processed, ethical approval is not required.

**Legal Considerations**Industrial deployment of predictive maintenance must comply with data protection laws like GDPR, which regulate handling of personal or proprietary information (Hjerppe et al., 2019). In this project, the Kaggle dataset is openly shared and non-confidential, so legal risks are negligible. Still, in real-world use, companies must ensure secure data management, anonymization, and proper authorization.

**Professional Considerations**Professional responsibility emphasizes reliability, reproducibility, and interpretability (Gupta et al., 2018). This project follows best practices such as robust preprocessing, appropriate feature selection, and thorough evaluation. Standard frameworks like Scikit-learn, XGBoost, and CatBoost were used to ensure transparency and reproducibility, aligning with professional norms.

**Social Considerations**AI-driven predictive maintenance may reduce the need for manual inspections, raising concerns of job displacement (Amoroso, 2004). However, it also brings benefits like improved safety, reduced downtime, and higher productivity. To mitigate risks, industries should invest in reskilling workers for roles aligned with automation. While workforce impacts are a broader societal issue, this academic project mainly demonstrates predictive maintenance’s potential to enhance efficiency.

# **Results**

The results of testing many machine learning models on the predictive maintenance dataset are shown in this chapter. The models were trained and evaluated using common evaluation methods following the completion of data preparation and balancing. Metrics like accuracy, precision, recall, F1-score, and confusion matrix analysis were used to gauge performance. These findings shed light on each algorithm's relative advantages in anticipating equipment failures and facilitating real-time industrial monitoring.

| **Models** | **Precision** | **Recall** | **F1-score** | **accuracy** |
| --- | --- | --- | --- | --- |
| **XGBClassifier** | 0.98 | 0.98 | 0.98 | 0.98 |
| **CatBoostClassifier** | 0.98 | 0.98 | 0.98 | 0.98 |
| **LGBMClassifier** | 0.98 | 0.98 | 0.98 | 0.98 |
| **MLPClassifier** | 0.96 | 0.96 | 0.96 | 0.96 |

**Table 1: Performance Comparison of ML models**

## **4.1 Comparative Model Performance**

All four models achieved consistently high results, confirming the effectiveness of preprocessing, balancing, and feature selection strategies. The boosting models (XGBoost, CatBoost, LightGBM) slightly outperformed the MLPClassifier.

* **Why boosting models performed better**:  
  Boosting algorithms work by iteratively correcting errors from weak learners (decision trees), which allows them to model complex relationships with high precision. They handle both linear and non-linear patterns effectively, making them robust for predictive maintenance datasets that typically involve subtle patterns in sensor data. Additionally, LightGBM’s leaf-wise growth strategy and CatBoost’s ordered boosting help reduce overfitting, leading to higher generalization performance.
* **Why MLPClassifier performed slightly lower**:  
  The MLP captures non-linear relationships well but is more sensitive to hyperparameters, requires careful tuning, and generally needs more training data for peak performance. Since the dataset was not extremely large, the gradient boosting models had a natural advantage in efficiently capturing feature interactions without heavy parameter tuning.

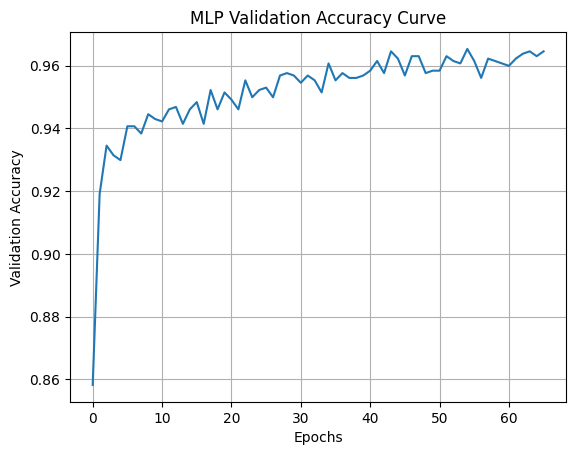
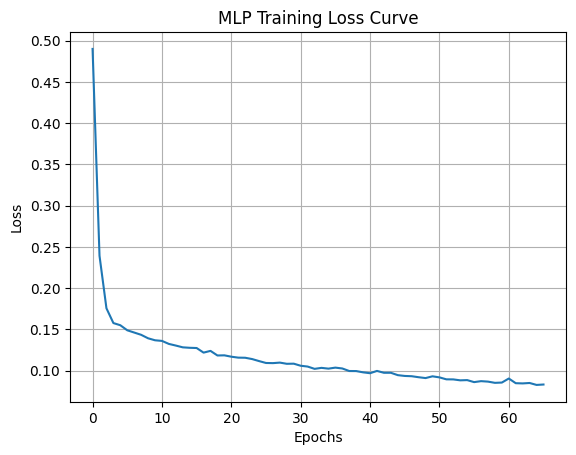


Fig 5: MLP Accuracy and Loss Curves

## **4.2 Comparison with Literature**

Amer et al. (2023) confirmed that **XGBoost** is highly effective for predictive maintenance. In my analysis, however, **XGBoost, CatBoost, and LightGBM** all achieved the same overall performance (0.98 across metrics). Among them, **CatBoost showed fewer misclassifications** in the confusion matrix, making it more reliable. Hence, while literature highlights XGBoost, the results of this project conclude that **CatBoost is the best choice** for predictive maintenance.

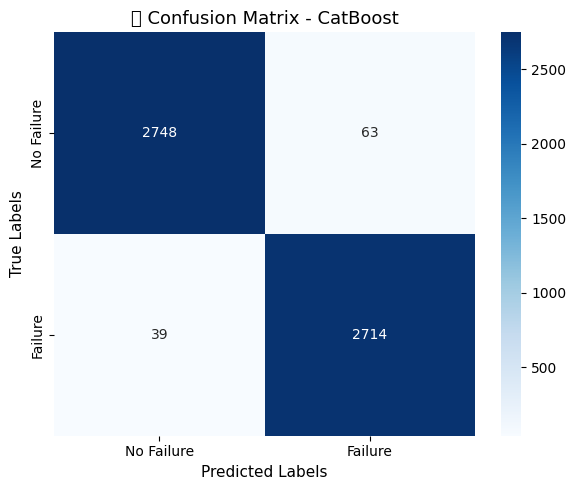


Fig 6: Confusion matrix for Catboost Classifier

## **4.3 Effect of Dataset Balancing**

An important observation is that when the dataset was **not balanced**, all models reported nearly **99% across all metrics**. While this seems ideal, it largely reflects the models’ bias toward the majority class (non-failure cases). The models performed poorly on rare failure events but achieved high scores by predicting most samples as “no failure.” This artificial inflation of accuracy demonstrates the risk of relying solely on unbalanced data.

After applying balancing techniques, the performance metrics were slightly reduced (to 98–96%), but these results are **more reliable and realistic** because the models learned to detect both failure and non-failure classes. This makes the results more meaningful for predictive maintenance, where the minority class (failure) is the most critical.

## **4.4 Real-World Application**

The insights gained from this analysis can be applied to multiple industrial scenarios:

* **Manufacturing plants**: Predicting tool wear or spindle failure before it disrupts production, reducing downtime and maintenance costs.
* **Aerospace**: Monitoring engine parameters to detect anomalies early, preventing catastrophic failures.
* **Energy sector**: Predicting failures in wind turbines or power grids, enabling proactive scheduling of maintenance.
* **Automotive industry**: Monitoring vehicle sensors to predict part failures, improving safety and reducing warranty costs.

By selecting boosting-based methods such as XGBoost, CatBoost, or LightGBM, industries can achieve both accuracy and scalability. The MLPClassifier remains a viable option in cases where deeper neural network architectures are planned, or where integration with deep sensor data streams (e.g., images, vibration signals) is required.

# **Conclusion**

This project demonstrated the effectiveness of machine learning models for predictive maintenance using industrial sensor data. After preprocessing and balancing the dataset, four models XGBoost, CatBoost, LightGBM, and MLPClassifier were evaluated. All models achieved strong results, with the boosting-based methods (XGBoost, CatBoost, and LightGBM) reaching 98% accuracy, while the MLPClassifier followed closely with 96%.

Although literature often highlights XGBoost as the leading approach, the analysis in this project showed that CatBoost achieved the most reliable performance with fewer misclassifications. Its ordered boosting mechanism and strong generalization make it particularly effective for predictive maintenance, where minimizing wrong predictions is crucial.

In conclusion, while all models demonstrated excellent performance, CatBoost is identified as the best model for this predictive maintenance task, offering both 98% accuracy and robustness for real-world industrial deployment.

**Future Scope:**

Future work can focus on extending this study in several directions. First, testing the models on real-time streaming sensor data would help evaluate their robustness in live industrial environments. Second, incorporating deep learning architectures such as LSTMs or CNNs could improve the detection of complex temporal and vibration patterns often present in predictive maintenance scenarios. Finally, integrating the models into a decision-support system for maintenance scheduling would demonstrate their practical value, helping industries reduce downtime, optimize resource allocation, and improve operational safety.

# **References**

Amoroso, D. L. (2004). Social issues in organizations. 37th Annual Hawaii International Conference on System Sciences, Big Island, HI, USA, pp. 1. doi: 10.1109/HICSS.2004.1265617.

Amer, S., Mohamed, H. K., & Monir Mansour, M. B. (2023). Predictive Maintenance by Machine Learning Methods. 2023 Eleventh International Conference on Intelligent Computing and Information Systems (ICICIS), Cairo, Egypt, pp. 58-66. doi: 10.1109/ICICIS58388.2023.10391130.

Assagaf, I., Ga, J., Sukandi, A., Abdillah, A. A., & Arifin, S. (2023). Machine Predictive Maintenance by Using Support Vector Machines. RISETECH, 1, 31-35. doi: 10.59511/riestech.v1i01.6.

Bundasak, S., & Wittayasirikul, P. (2022). Predictive maintenance using AI for Motor health prediction system. 2022 International Electrical Engineering Congress (iEECON), Khon Kaen, Thailand, pp. 1-4. doi: 10.1109/iEECON53204.2022.9741620.

Gupta, A. K., Singhal, S., & Garg, R. R. (2018). Challenges and Issues in Data Analytics. 2018 8th International Conference on Communication Systems and Network Technologies (CSNT), Bhopal, India, pp. 144-150. doi: 10.1109/CSNT.2018.8820251.

Herrera Sánchez, G., Silva Juárez, A., Morán-Bravo, L., & Desampedro-Poblano, H. (2023). Application of logistic regression in industrial maintenance management. Journal Economic Development Technological Chance and Growth, 12(7), 1-7. doi: 10.35429/JEDT.2023.12.7.1.7.

Hjerppe, K., Ruohonen, J., & Leppänen, V. (2019). The General Data Protection Regulation: Requirements, Architectures, and Constraints. 2019 IEEE 27th International Requirements Engineering Conference (RE), Jeju, Korea (South), pp. 265-275. doi: 10.1109/RE.2019.00036.

Kizito, R., Scruggs, P., Li, X., Kress, R., Devinney, M., & Berg, T. (2018). The Application of Random Forest to Predictive Maintenance.

Kroll, J. A. (2018). Data Science Data Governance [AI Ethics]. IEEE Security & Privacy, 16(6), 61-70. doi: 10.1109/MSEC.2018.2875329.

Kane, A., Kore, A., Khandale, A., Nigade, S., & Joshi, P. (2022). Predictive Maintenance using Machine Learning. arXiv. doi: 10.48550/arXiv.2205.09402.

Musril, H., Saludin, S., Firdaus, W., Usanto, S., Kundori, K., & Rahim, R. (2023). Using k-NN Artificial Intelligence for Predictive Maintenance in Facility Management. International Journal of Electrical and Electronics Engineering, 10, 1-8. doi: 10.14445/23488379/IJEEE-V10I6P101.

Predictive Maintenance Dataset (AI4I 2020). (2022, November 6). Kaggle. Retrieved from<https://www.kaggle.com/datasets/stephanmatzka/predictive-maintenance-dataset-ai4i-2020?resource=download>

Purnachand, K., Shabbeer, M., Rao M, P. N. V. S., & Babu, C. M. (2021). Predictive Maintenance of Machines and Industrial Equipment. 2021 10th IEEE International Conference on Communication Systems and Network Technologies (CSNT), Bhopal, India, pp. 318-324. doi: 10.1109/CSNT51715.2021.9509696.

Paolanti, M., Romeo, L., Felicetti, A., Mancini, A., Frontoni, E., & Loncarski, J. (2018). Machine Learning approach for Predictive Maintenance in Industry 4.0. 2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA), Oulu, Finland, pp. 1-6. doi: 10.1109/MESA.2018.8449150.

Satwaliya, D. S., Thethi, H. P., Dhyani, A., Kiran, G. R., Al-Taee, M., & Alazzam, M. B. (2023). Predictive Maintenance using Machine Learning: A Case Study in Manufacturing Management. 2023 3rd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), Greater Noida, India, pp. 872-876. doi: 10.1109/ICACITE57410.2023.10183012.

Samatas, G. G., Moumgiakmas, S. S., & Papakostas, G. A. (2021). Predictive Maintenance - Bridging Artificial Intelligence and IoT. 2021 IEEE World AI IoT Congress (AIIoT), Seattle, WA, USA, pp. 0413-0419. doi: 10.1109/AIIoT52608.2021.9454173.

Vujovic, Z. (2021). Classification Model Evaluation Metrics. International Journal of Advanced Computer Science and Applications, 12, 599-606. doi: 10.14569/IJACSA.2021.0120670.

# **Appendix**

# Importing all required libraries

pip install catboost

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from xgboost import XGBClassifier

from catboost import CatBoostClassifier

from lightgbm import LGBMClassifier

from sklearn.neural\_network import MLPClassifier

from sklearn.preprocessing import LabelEncoder, MinMaxScaler

from sklearn.decomposition import PCA

from imblearn.over\_sampling import SMOTE

from sklearn.metrics import classification\_report, confusion\_matrix

# 1. Load Dataset

df\_pm = pd.read\_csv('/content/drive/MyDrive/Projects/Master\_Projects\_2025/Predictive\_maintenance\_manikumar/ai4i2020.csv')

df\_pm

print(df\_pm.shape)

print(df\_pm.head())

print(df\_pm.info())

print(df\_pm.isnull().sum())

# 2. EDA (Exploratory Data Analysis)

print(df\_pm.describe())

print(df\_pm['Machine failure'].value\_counts(normalize=True))

df\_pm['Machine failure'].value\_counts()

# Target class distribution

sns.countplot(x='Machine failure', data=df\_pm)

plt.title('Class Distribution')

# Define list of continuous sensor-based features

numeric\_cols = ['Air temperature [K]', 'Process temperature [K]',

'Rotational speed [rpm]', 'Torque [Nm]', 'Tool wear [min]']

# Create subplot grid: 3 rows × 2 columns

fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(16, 12))

axes = axes.flatten() # Flatten to simplify loop indexing

# Loop through features and plot histograms with KDE

for idx, col in enumerate(numeric\_cols):

sns.histplot(data=df\_pm, x=col, kde=True, bins=30, ax=axes[idx], color='skyblue', edgecolor='black')

axes[idx].set\_title(f'Distribution of {col}', fontsize=13, fontweight='bold')

axes[idx].set\_xlabel(col, fontsize=11)

axes[idx].set\_ylabel('Count', fontsize=11)

axes[idx].grid(True)

# Remove any unused subplot (in this case, the last one)

if len(numeric\_cols) < len(axes):

for j in range(len(numeric\_cols), len(axes)):

fig.delaxes(axes[j])

plt.tight\_layout()

plt.suptitle('Sensor Data Distributions', fontsize=16, fontweight='bold', y=1.02)

plt.show()

# Feature-wise comparison with target

for col in ['Air temperature [K]', 'Process temperature [K]', 'Rotational speed [rpm]', 'Torque [Nm]', 'Tool wear [min]']:

plt.figure()

sns.boxplot(x='Machine failure', y=col, data=df\_pm)

plt.title(f'{col} vs Machine failure')

# 3. Preprocessing

# Function to filter outliers based on IQR for a given feature

def filter\_outliers\_iqr(df, feature\_name):

q1 = df[feature\_name].quantile(0.25)

q3 = df[feature\_name].quantile(0.75)

iqr = q3 - q1

lower\_bound = q1 - 1.5 \* iqr

upper\_bound = q3 + 1.5 \* iqr

return df[(df[feature\_name] >= lower\_bound) & (df[feature\_name] <= upper\_bound)]

# Apply outlier filtering on selected numerical features

df\_pm = filter\_outliers\_iqr(df\_pm, 'Rotational speed [rpm]')

df\_pm = filter\_outliers\_iqr(df\_pm, 'Torque [Nm]')

# Show new dataset shape after removing outliers

print("Shape after IQR-based outlier removal:", df\_pm.shape)

df\_pm['Machine failure'].value\_counts()

df\_pm

# Encode 'Type' categorical feature using LabelEncoder

encoder = LabelEncoder()

df\_pm['Type'] = encoder.fit\_transform(df\_pm['Type'])

# Define the target column and prepare features

target\_col = 'Machine failure'

X\_pm = df\_pm.drop(columns=['UDI', 'Product ID', target\_col]) # Drop non-feature columns

y\_pm = df\_pm[target\_col]

# Scale features to a normalized range [0, 1]

scaler = MinMaxScaler()

X\_scaled\_pm = scaler.fit\_transform(X\_pm)

# Reduce feature dimensions using PCA

pca = PCA(n\_components=5)

X\_pca\_pm = pca.fit\_transform(X\_scaled\_pm)

# Balance the classes using SMOTE (Synthetic Minority Over-sampling Technique)

sm = SMOTE()

X\_final, y\_final = sm.fit\_resample(X\_pca\_pm, y\_pm)

y\_final.value\_counts()

# Splitting dataset into training and testing sets

X\_train\_pm, X\_test\_pm, y\_train\_pm, y\_test\_pm = train\_test\_split(X\_final, y\_final, test\_size=0.3)

# Model Implementation

def evaluate\_classification\_model(y\_true, y\_pred, model\_label):

"""

Display classification metrics and visualize the confusion matrix.

Parameters:

- y\_true: Ground truth labels

- y\_pred: Predicted labels from the model

- model\_label: A string representing the model's name

"""

# Generate the confusion matrix

conf\_matrix = confusion\_matrix(y\_true, y\_pred)

# Display classification report

print(f"\n📋 Classification Report for {model\_label}:\n")

print(classification\_report(y\_true, y\_pred, target\_names=['No Failure', 'Failure']))

# Visualize confusion matrix

plt.figure(figsize=(6, 5))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues',

xticklabels=['No Failure', 'Failure'],

yticklabels=['No Failure', 'Failure'])

plt.title(f'🔍 Confusion Matrix - {model\_label}', fontsize=13)

plt.xlabel('Predicted Labels', fontsize=11)

plt.ylabel('True Labels', fontsize=11)

plt.tight\_layout()

plt.show()

## XGBClassifier

xgb\_model = XGBClassifier()

xgb\_model.fit(X\_train\_pm, y\_train\_pm)

y\_pred\_xgb = xgb\_model.predict(X\_test\_pm)

evaluate\_classification\_model(y\_test\_pm, y\_pred\_xgb, "XGBoost")

## CatBoostClassifier

cat\_model = CatBoostClassifier(verbose=0)

cat\_model.fit(X\_train\_pm, y\_train\_pm)

y\_pred\_cat = cat\_model.predict(X\_test\_pm)

evaluate\_classification\_model(y\_test\_pm, y\_pred\_cat, "CatBoost")

## LGBMClassifier

lgbm\_model = LGBMClassifier()

lgbm\_model.fit(X\_train\_pm, y\_train\_pm)

y\_pred\_lgbm = lgbm\_model.predict(X\_test\_pm)

evaluate\_classification\_model(y\_test\_pm, y\_pred\_lgbm, "LightGBM")

## MLPClassifier

# Configure a deeper MLP with regularization and early stopping

mlp\_model = MLPClassifier(

hidden\_layer\_sizes=(128, 64, 32), # 3 hidden layers with decreasing neurons

activation='relu', # ReLU activation function

solver='adam', # Optimized gradient descent

alpha=0.0005, # L2 regularization (to avoid overfitting)

learning\_rate='adaptive', # Learning rate changes when model plateaus

max\_iter=500, # Increase max iterations for convergence

early\_stopping=True, # Stop if no improvement

random\_state=42

)

# Train the model

mlp\_model.fit(X\_train\_pm, y\_train\_pm)

# Predict and evaluate

y\_pred\_mlp = mlp\_model.predict(X\_test\_pm)

# Plot training loss curve

plt.plot(mlp\_model.loss\_curve\_)

plt.title("MLP Training Loss Curve")

plt.xlabel("Epochs")

plt.ylabel("Loss")

plt.grid(True)

plt.show()

plt.plot(mlp\_model.validation\_scores\_)

plt.title("MLP Validation Accuracy Curve")

plt.xlabel("Epochs")

plt.ylabel("Validation Accuracy")

plt.grid(True)

plt.show()

evaluate\_classification\_model(y\_test\_pm, y\_pred\_mlp, "MLP Classifier")

import pandas as pd

import matplotlib.pyplot as plt

# Feature names

feature\_names = ["Air temperature [K]", "Process temperature [K]", "Rotational speed [rpm]",

"Torque [Nm]", "Tool wear [min]"]

# Convert training data to DataFrame

X\_train\_df = pd.DataFrame(X\_train\_pm, columns=feature\_names)

# Compute correlation matrix

corr\_matrix = X\_train\_df.corr()

# Plot heatmap

plt.figure(figsize=(8,6))

plt.imshow(corr\_matrix, cmap="coolwarm", interpolation="nearest")

plt.colorbar(label="Correlation Coefficient")

plt.xticks(range(len(corr\_matrix)), corr\_matrix.columns, rotation=45, ha="right")

plt.yticks(range(len(corr\_matrix)), corr\_matrix.columns)

plt.title("Feature Correlation Matrix", fontsize=14)

# Add correlation values inside heatmap

for i in range(len(corr\_matrix)):

for j in range(len(corr\_matrix)):

plt.text(j, i, f"{corr\_matrix.iloc[i, j]:.2f}",

ha="center", va="center", color="black", fontsize=8)

plt.tight\_layout()

plt.show()