Predictive Maintenance for Manufacturing

SGHIR Marwa

National school of applied sciences, AL HOCEIMA

marwa.sghir@etu.uae.ac.ma

Supervision: Pr KHAMJANE Aziz

Abstract

This research investigates the application of predictive maintenance using the AI4I 2020 dataset. By using machine learning techniques, the study aims to predict equipment failures and minimize unexpected downtime. The research evaluates the performance of various algorithms on the dataset and demonstrates the effectiveness of predictive maintenance in improving machine efficiency and reducing maintenance costs. The findings highlight the importance of selecting appropriate algorithms for optimal predictive maintenance outcomes. A synthetic predictive maintenance dataset generated by the School of Engineering at the University of Applied Sciences in Berlin, Germany, was used to test the performance of the suggested algorithms.

I. Introduction

Predictive maintenance is an important strategy in modern manufacturing that helps prevent equipment failures before they happen. By analyzing data from machines, this approach reduces unexpected downtime, schedules maintenance more effectively, and improves overall efficiency. With the rise of artificial intelligence (AI) and machine learning (ML), predictive maintenance has become even more powerful, making it easier to predict problems and take action in advance.

Classification models are essential for predictive maintenance because they help make accurate predictions quickly. These models use data to classify whether a machine is likely to fail or continue operating normally. In this study, we compare several popular classification models—Decision Tree, k-Nearest Neighbors (k-NN), Random Forest, Gradient Boosting, and Gaussian Naive Bayes—to understand how well they perform in terms of prediction accuracy, speed, and reliability.

This research highlights how machine learning can solve real-world problems in predictive maintenance. By evaluating these models based on accuracy, precision, recall, F1-score, training time, and prediction time, we provide insights into which model is best suited for different needs. This helps researchers and professionals choose the right tools to make maintenance more efficient and cost-effective.

II. Previous studies

In a study published in the Journal of Mechanical Science and Technology, Yang et al. (2019) proposed a data-driven approach to predict the remaining useful life of machine tool spindles [2]. The authors used a combination of principal component analysis (PCA) and extreme machine learning (ELM) to predict spindle

RUL from vibration data. The proposed method is validated using data collected from real machine tools, and the results show that the method can accurately predict spindle RUL.

Similarly, in a study by Wang et al. (2020), present a data-driven approach to predict remaining bearing life using vibration data [3]. The authors used a combination of wavelet packet decomposition (WPD) and deep belief network (DBN) to predict bearing RUL. The proposed method is validated using data collected from real wind turbines, and the results show that the method can accurately predict bearing RUL.

These studies have proposed a range of approaches, including machine learning, data-driven, and fuzzy logic-based techniques, that can be used to develop predictive maintenance systems for various types of machines. Future research in this area could investigate the development of hybrid approaches that combine multiple techniques to improve the accuracy and reliability of predictive maintenance systems.

III. Dataset and Methodology

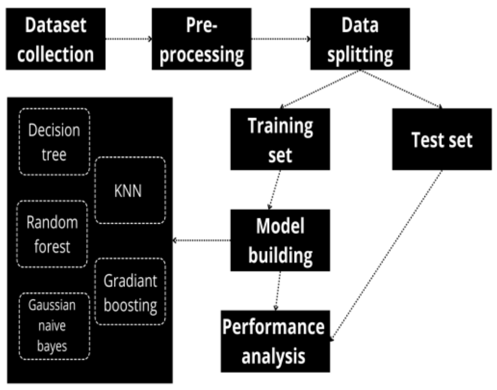


Figure1: Project Workflow

About the dataset:

Although predictive maintenance is a significant research field in the industrial sector, acquiring authentic datasets may prove challenging. In response to this challenge, a synthetic dataset was developed to emulate real predictive maintenance data observed in the industry.

The dataset is made up of 10,000 data points ordered as rows with the columns:

1. UID

2. Product ID

3. Type

4. Air temperature [K]

5. Rotational speed [rpm]

6. Tool wear [min]

7. Torque [Nm]

8. Process temperature [K]

The data set also contains a "Machine Failure" label, indicating if a machine has failed at that particular data point. Machine failure consists of the following independent failure modes.

1. Tool wear failure (TWF)

2. Power failure (PWF)

3. Random failures (RNF)

4. Heat dissipation failure (HDF)

5. Overstrain failure (OSF)

If any one of the above failure modes is true, the process fails and the machine error flag is set to 1. Therefore, the failure mode that causes the process to fail is opaque to machine learning methods

Data preprocessing / preparation:

There were no missing, null or duplicate values.

Data conditioning:

During the data conditioning process, inconsistencies in the dataset were identified and addressed. Specifically, rows were examined for cases where a machine failure was recorded without any associated failure mode being triggered, or where a failure mode was triggered but no machine failure was reported. Any rows exhibiting such errors were removed to ensure the dataset's consistency and reliability.

Dropping irrelevent features

In this section, we address the exclusion of irrelevant features like UID and Product ID. We made this decision due to their limited impact on our analysis

Exploratory data analysis

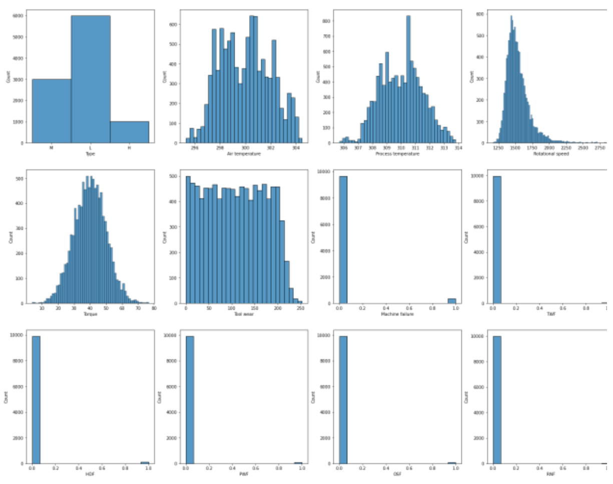


Figure2: the distribution for each attribute

After plotting the distribution for each attribute (Type, Machine failure, TWF, HDF, PWF, OSF, RNF), we can see that the data is imbalanced.

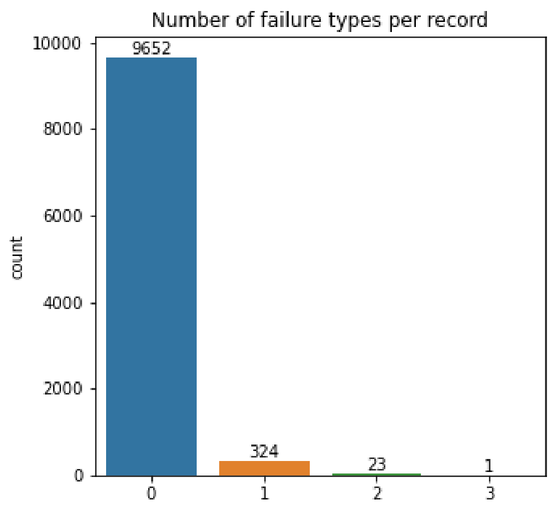


Figure3: number of failure types per record

As shown above, 24 records contain more than one type of failure, but their count is very small compared to the entire data set, so we will combine the failure types into one feature. The individual failure types are then dropped.

No failure = 0, TWF = HDF = PWF = OSF = RNF = 1

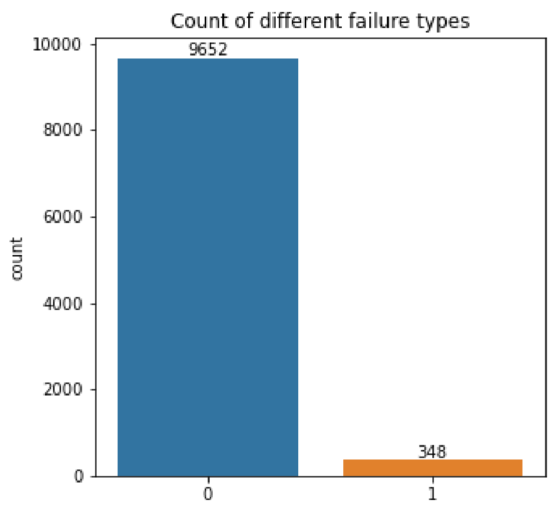


Figure4: Count of failure types

Data type conversion

We convert Type attribute into numbers, such that L = 0, M = 1, and H = 2, for “Type” attribute and we turned all columns into float for easier processing later.

Handling outliers: Density-Based Anomaly Detection (LOF)

Outliers can significantly affect the performance of machine learning models, so identifying and removing them is an essential step in the data preprocessing process. To detect outliers in the dataset, the **Density-Based Anomaly Detection** technique, specifically the **Local Outlier Factor (LOF)** algorithm, was used. The LOF model evaluates the density of data points in the feature space and assigns a score to each instance. Data points with a negative score are identified as outliers. After detecting the outliers, any rows with outlier values were removed from the dataset to maintain the integrity of the analysis and ensure more reliable predictions.

For more **visualization** we Generated a profile report that includes (outliers, missing values, distributions, etc.) using pandas-profiling

Feature engineering

Before further EDA, in this section, we engineer new features that could help in predicting machine failures more accurately. These features are derived from the existing data and are based on domain knowledge and hypotheses about what conditions might lead to machine failure. For example, the temperature difference could be an indicator of abnormal machine behavior, and the power is a product of torque and rotational speed which could affect the machine's operational efficiency.

Based on the following figures:

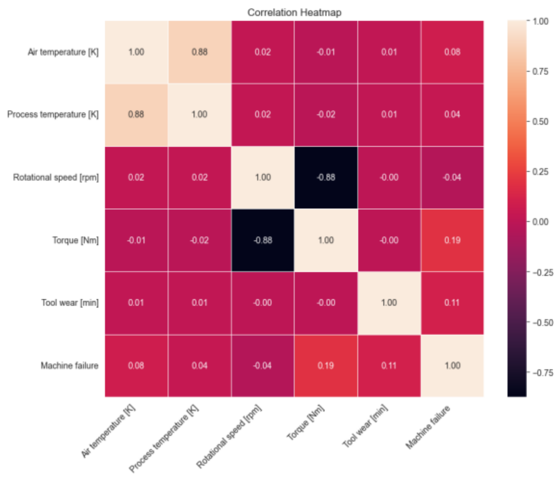


Figure5: Correlation heatmap

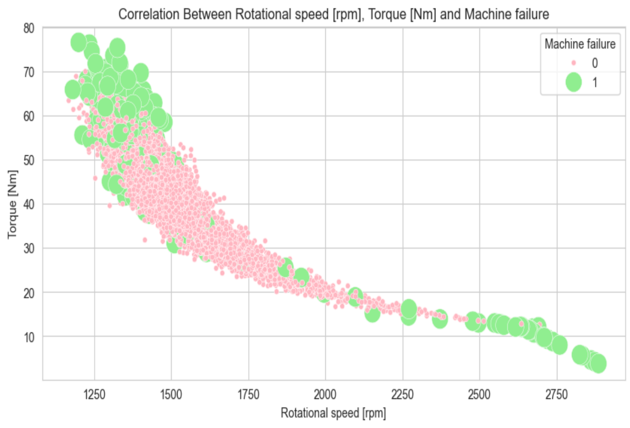


Figure6: Correlation between torque, rotational speed and machine failure

From the above graph, we can conclude most machine failure occurs When Torque [Nm] value is high and Rotational speed [rpm] is also high

So we can derive a new attributes using these formulas:

$$Power=Torque * Rotational\ speed$$

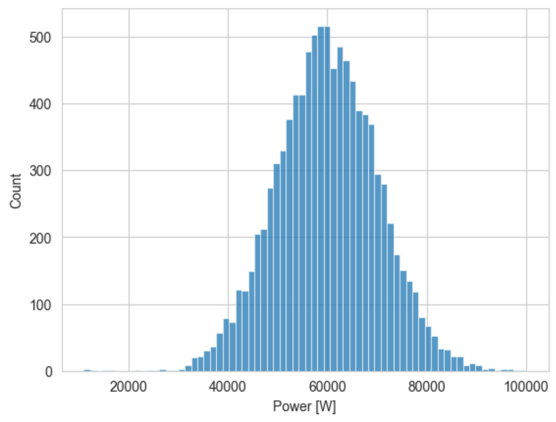


Figure7: Histogram of Power attribute

$$\textit{Temperature difference} = \textit{Process temperature - Air temperature}$$

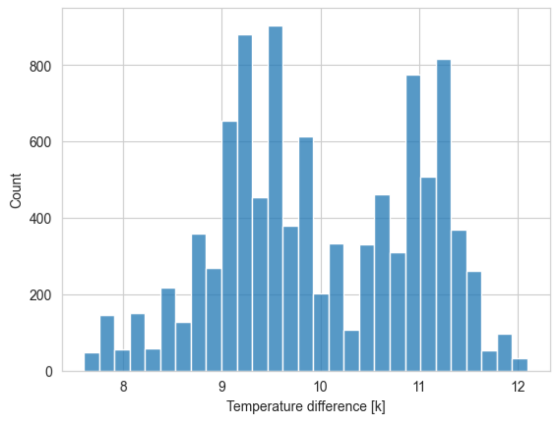


Figure7: Histogram of Temperature difference attribute

Feature Selection

To improve model performance and reduce computational complexity, feature selection was performed using the SelectKBest method with the Chi-Square (chi2) score function. This technique evaluates the relevance of each feature in relation to the target variable, "Machine failure." The dataset was divided into features (X) and the target variable (y), and the Chi-Square test was applied to score each feature based on its relationship with the target. The features were ranked by score, and the top features were visualized to identify the most significant variables. The results provided insights into which features had the highest predictive power for the model, allowing for more efficient and effective model training.

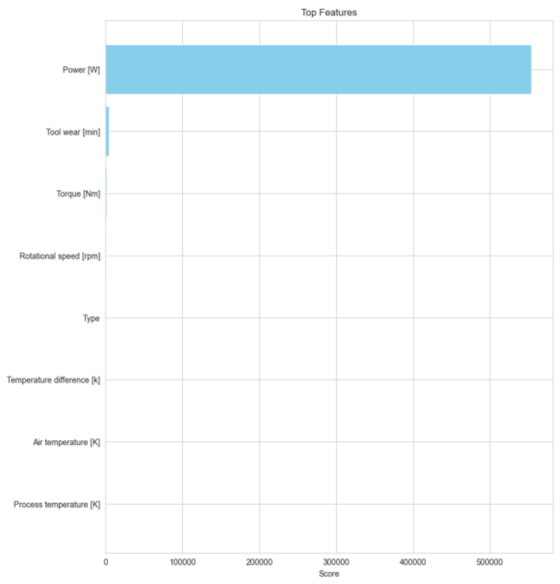


Figure8: Top features

Data splitting and sampling

This involves splitting the dataset into two partitions for training and testing, ensuring the model's effectiveness with new, unseen data. In our study, we divided the entire dataset into training and testing sets, maintaining 70%-30% ratio.

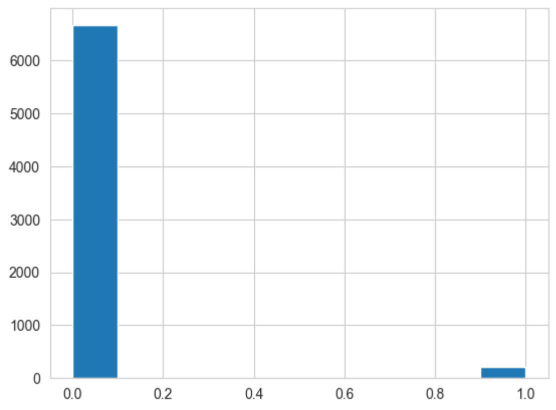


Figure9: Training dataset before oversampling

Because the data is imbalanced, we oversample the training set

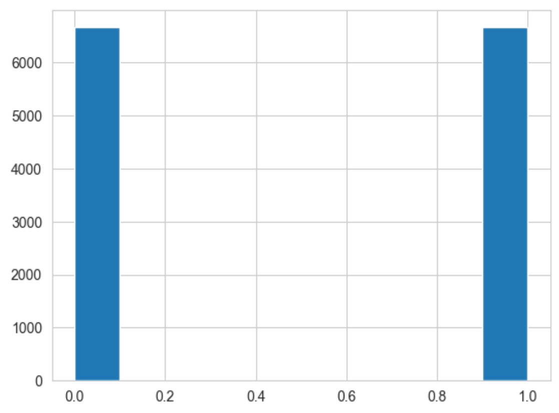


Figure10: Training dataset after oversampling

Handling Class Imbalance

To address the issue of class imbalance in the dataset, where the target variable "Machine failure" was disproportionately represented, **SMOTE (Synthetic Minority Over-sampling Technique)** was used. SMOTE works by generating synthetic samples of the minority class based on the feature space of existing minority class instances. This technique was applied to the training data to balance the class distribution, thereby preventing the model from being biased toward the majority class. The resampled dataset, with more balanced class representation, was then used for model training to improve the model's ability to generalize and make accurate predictions for both classes.

IV. Modeling

In this study, we applied several classification models to predict machine failures based on sensor data. The models chosen for evaluation include **Decision Tree**, **k-Nearest Neighbors (k-NN)**, **Random Forest**, **Gradient Boosting**, and **Gaussian Naive Bayes**. These models were selected for their varying complexity, computational efficiency, and ability to handle different aspects of the classification problem.

1. Decision Tree Classifier

The **Decision Tree** is a non-parametric model that recursively splits the data into subsets based on feature values. The splits aim to reduce the impurity in the subsets (often using Gini impurity or entropy as a criterion). We applied hyperparameters like `max_depth=8`, `min_samples_split=10`, and `min_samples_leaf=5` to prevent overfitting. Although the model performed well on the majority class, the imbalance led to poor performance on the minority class, where precision and recall were lower for machine failure predictions. Decision trees are prone to overfitting if left unpruned, which is why hyperparameter tuning is essential to control the depth and leaf nodes of the tree.

2. k-Nearest Neighbors (k-NN)

k-NN is a simple, instance-based algorithm that assigns a label based on the majority class among the k nearest neighbors in the feature space. In our experiment, we used a range of k values and selected the best one through **GridSearchCV**. The model's performance was impacted by the imbalance in the data, resulting in poor recall and precision for the minority class. One of the challenges with k-NN is its sensitivity to the feature scale and its high computational cost during prediction. As the number of neighbors (k) increases, the model becomes more general, reducing the risk of overfitting.

3. Random Forest Classifier

The **Random Forest** is an ensemble learning method that aggregates multiple decision trees trained on random subsets of features and samples. It is less prone to overfitting than a single decision tree due to the randomness introduced in the training process. In this study, we applied a Random Forest model with hyperparameters like `n_estimators=50` and `max_depth=8`. While the Random Forest performed well on the majority class, it showed similar challenges with the minority class. Its robustness against overfitting is generally attributed to averaging across multiple trees, which reduces variance and improves generalization.

4. Gradient Boosting Classifier

Gradient Boosting is another ensemble technique that builds trees sequentially, where each tree corrects the errors made by its predecessor. It focuses on reducing bias by fitting to the residuals of previous models. We applied this model without heavy tuning, and it showed comparable results to Random Forest, with slightly better precision but still lower recall on the minority class. Overfitting can occur in Gradient Boosting models when the trees are too deep or too many, leading to overfitting on the training data, but this is mitigated by early stopping or tuning the number of trees.

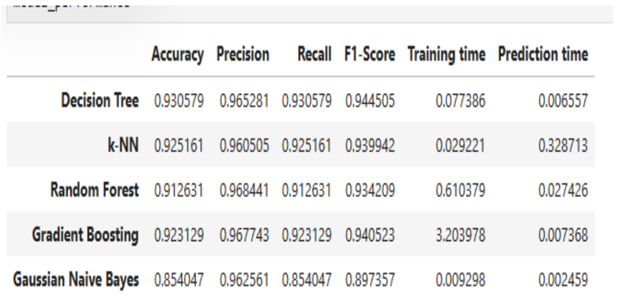
5. Gaussian Naive Bayes

The **Gaussian Naive Bayes** model is a probabilistic classifier based on Bayes' theorem, assuming that the features are conditionally independent given the class label. While it performed well for the majority class, its performance on the minority class was less robust, resulting in a low F1-score for machine failure predictions. Despite its simplicity, Naive Bayes can be prone to overfitting if the features are highly correlated, as the assumption of independence becomes less valid.

Overfitting Concerns

Throughout the modeling process, overfitting was a key concern, especially with more complex models such as Decision Tree and Gradient Boosting. Overfitting occurs when a model learns not only the underlying patterns in the training data but also the noise, leading to poor generalization to unseen data. To mitigate overfitting, we employed strategies such as pruning the decision tree (`max_depth`, `min_samples_split`, `min_samples_leaf`), limiting the depth of Random Forest and Gradient Boosting models, and balancing the dataset using **SMOTE**. Cross-validation was also applied to ensure robust evaluation and prevent overfitting to the training data.

V. Evaluation



	Accuracy	Precision	Recall	F1-Score	Training time	Prediction time
Decision Tree	0.930579	0.965281	0.930579	0.944505	0.077386	0.006557
k-NN	0.925161	0.960505	0.925161	0.939942	0.029221	0.328713
Random Forest	0.912631	0.968441	0.912631	0.934209	0.610379	0.027426
Gradient Boosting	0.923129	0.967743	0.923129	0.940523	3.203978	0.007368
Gaussian Naive Bayes	0.854047	0.962561	0.854047	0.897357	0.009298	0.002459

Figure10:

The models were evaluated based on key metrics: accuracy, precision, recall, F1-Score, training time, and prediction time. Here's a brief comparison:

- **Decision Tree**: Achieved high accuracy (93.06%) and a good balance between precision (96.53%) and recall (93.06%). It also had fast training (0.077s) and prediction times (0.007s).
- **k-NN**: Similar to the Decision Tree in terms of accuracy (92.52%), precision (96.05%), and recall (92.52), but with longer prediction time (0.329s).

- **Random Forest**: Slightly lower accuracy (91.26%) but high precision (96.84%) and decent recall (91.26%). It had a longer training time (0.61s) but fast prediction time (0.027s).
- **Gradient Boosting**: Similar performance to the others but with a significantly longer training time (3.20s).
- **Gaussian Naive Bayes**: Had the lowest accuracy (85.40%) but was very fast in training (0.009s) and prediction (0.002s).

Best Model: Based on overall performance, **Decision Tree** is the best model to use. It strikes a good balance between accuracy, precision, recall, and efficiency, making it the most reliable choice for this task.

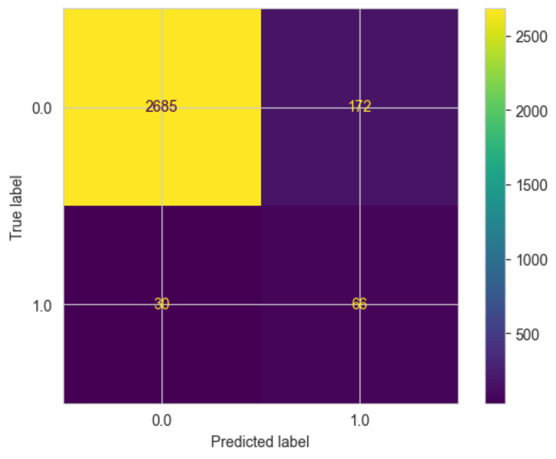


Figure11: Confusion matrix for the decision tree model

VI. Limitations of the Approach

The approach has several limitations, particularly in certain scenarios where models may perform poorly:

1. **Class Imbalance**: Despite using SMOTE, models like **k-NN** may still struggle with imbalanced datasets, leading to inaccurate predictions for rare classes.
2. **Noisy Data**: Models like **Random Forest** and **Gradient Boosting** can overfit to noisy data, reducing their ability to generalize.
3. **Simplified Assumptions**: **k-NN** and **Naive Bayes** may fail to capture complex relationships in the data, resulting in suboptimal performance.

Extensions:

- Using more advanced techniques like **cost-sensitive learning**, **deep learning**, or **robust decision trees** could improve performance, especially in noisy or imbalanced scenarios.

VII. Conclusions

The study found that **Decision Tree** and **k-NN** models performed well for predicting machine failures, with **Decision Tree** achieving the best balance of accuracy, precision, and computational efficiency.

Key points:

- The models showed strong accuracy (up to 96%) and f1-scores.
- **SMOTE** effectively addressed class imbalance.
- **Decision Tree** is the best model, but improvements can be made to handle noise and imbalanced data.

In conclusion, while the approach is effective, further improvements are possible, particularly in dealing with noise and complex datasets.

VIII. References

1. Yang, L., et al. (2019). A data-driven approach for predicting the remaining useful life of machine tool spindle based on PCA and ELM. Journal of Mechanical

Science and Technology, 33(2), 785-795.

2. Wang, Q., et al. (2020). Predicting remaining useful life of rolling bearings using wavelet packet decomposition and deep belief network. Journal of Mechanical Science and Technology, 34(1), 79-88.
3. S. Matzka, "Explainable Artificial Intelligence for Predictive Maintenance Applications," 2020 Third International Conference on Artificial Intelligence for Industries (AI4I), 2020, pp. 69-74, doi: 10.1109/AI4I49448.2020.00023.
4. https://www.kaggle.com/datasets/stephanmatzka/predictive-maintenance-dataset-ai4i-2020
5. https://ijsdr.org/papers/IJSDR2305088.pdf
6. https://www.sciencedirect.com/science/article/pii/S2213846323001864