Predictive Maintenance with AI

Berat Kargin

May 2025

1 Introduction

In the era of Industry 4.0, the integration of Artificial Intelligence (AI) into industrial processes has become paramount. Predictive maintenance, a proactive approach to equipment upkeep, leverages AI to foresee potential machinery failures, thereby minimizing unplanned downtimes and optimizing operational efficiency. This project, developed in collaboration with the Scientific and Technological Research Council of Türkiye (TÜBİTAK), aims to harness AI techniques to predict machine failures in an industrial setting, enhancing maintenance strategies and ensuring seamless production workflows.

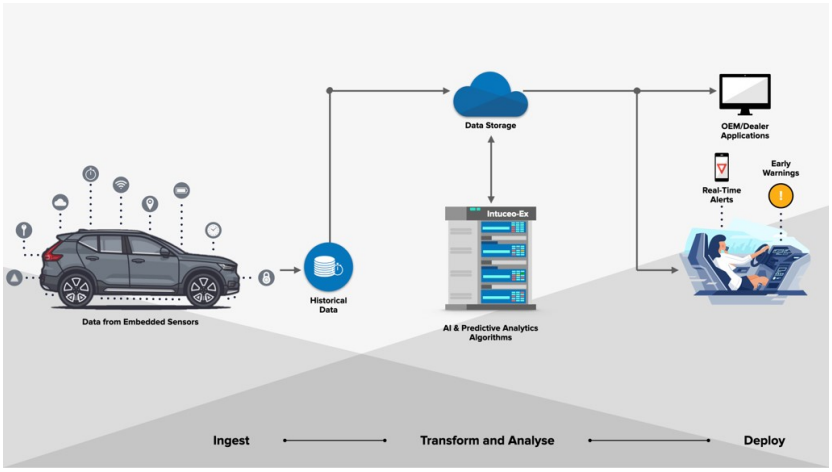


Figure 1: Predictive Maintenance

2 Problem Definition

Traditional maintenance strategies, such as reactive and preventive maintenance, often lead to either unexpected equipment failures or unnecessary maintenance activities, both of which can be costly and inefficient. The core problem

addressed in this project is the need for an intelligent system that can accurately predict machine failures before they occur. By analyzing sensor data and operational parameters, the goal is to develop a model that can identify patterns indicative of impending failures, enabling timely interventions and reducing maintenance costs.

The challenge lies in:

• Processing raw sensor readings and transforming them into meaningful features,

• Choosing appropriate models that can accurately detect failures,

• Dealing with class imbalance due to the rarity of failures compared to normal operations.

• The ultimate objective is to classify the general machine failure status (Machine failure) based on sensor inputs.

3 Literature Survey

Predictive maintenance has been a focal point in industrial research, with various methodologies explored to enhance its efficacy. Traditional statistical methods laid the groundwork, but the advent of machine learning and AI has revolutionized the field. Various approaches have been proposed in the literature for predictive maintenance, often involving:

• Statistical techniques like control charts and time-series anomaly detection.

• Machine learning models such as Support Vector Machines (SVMs), Random Forests, and Neural Networks.

• Deep learning approaches (especially for time-series data), including LSTMs and CNNs.

Research studies suggest that tree-based models and nearest-neighbor methods often perform well when labeled failure data is available and the dataset is tabular rather than sequential.

Inspired by this, our project implements and compares multiple traditional machine learning models to identify the most effective technique for this specific problem.

4 Methods to be Used and Compared

The project employs a comprehensive approach, integrating various data preprocessing, feature selection, and modeling techniques:

4.1 Data Preprocessing

• Handling missing values and outliers.
• Encoding categorical variables using One-Hot Encoding.
• Scaling features through normalization (MinMaxScaler) and standardization (StandardScaler).

4.2 Feature Selection

• Random Forest Feature Importance combined with correlation analysis to identify significant features.
• Recursive Feature Elimination (RFE) to iteratively select the most impactful features.

4.3 Modeling Techniques

In this project, we used and compared the following AI models:

- **K-Nearest Neighbors (KNN)**
- **Random Forest (RF)**
- **Decision Tree (DT)**

All models were trained on carefully preprocessed data, including normalization/standardization, categorical encoding, and feature selection (via RFE, and Random Forest Importance).

4.3.1 K-Nearest Neighbors (KNN)

KNN is a non-parametric model that classifies new data points based on the majority class among its 'k' nearest neighbors in the feature space.

Advantages:
• Simple to implement and interpret.
• Works well with smaller datasets.
• No assumptions about data distribution.

Disadvantages:
• Computationally expensive with large datasets (slow at prediction time).
• Sensitive to irrelevant features and scaling.
• Performance depends heavily on the choice of 'k' and the distance metric.

4.3.2 Random Forest Classifier

Random Forest is an ensemble learning method that builds multiple decision trees and merges their outputs to improve accuracy and control overfitting.

Advantages:
• High accuracy and robustness to overfitting.
• Handles both categorical and numerical data well.
• Provides feature importance metrics.

Disadvantages:
• Less interpretable than a single decision tree.
• Slower training time with large number of trees.
• Can be memory-intensive.

4.3.3 Decision Tree Classifier

A Decision Tree splits the data based on feature values to create a tree structure for decision-making.

Advantages:
- Easy to visualize and interpret.
- Requires little data preprocessing.
- Fast to train and predict.

Disadvantages:
- High variance — prone to overfitting.
- Small changes in data can result in a completely different tree.
- Generally less accurate than ensemble models like Random Forest.

4.4 Model Evaluation

- Utilizing GridSearchCV for hyperparameter tuning.
- Assessing model performance through metrics such as Accuracy, Precision, Recall, F1 Score, Confusion Matrix, and ROC-AUC curves.

This multifaceted methodology ensures a robust analysis, facilitating the development of an AI-driven predictive maintenance system tailored for industrial applications.

5 Experiments and Results

5.1 Experimental Setup

The dataset consists of various sensor readings capturing machine conditions in a manufacturing environment. The target variable, Machine failure, is a binary indicator representing the occurrence of a failure. Additionally, the dataset includes multiple failure types (TWF, HDF, PWF, OSF, RNF) that specify the nature of the failure.

The data preprocessing pipeline involved:
- Handling missing values (if any),
- Encoding categorical features (Type) using One-Hot Encoding,
- Applying **Normalization (MinMaxScaler)** for KNN,
- Applying **Standardization (StandardScaler)** for tree-based models (Random Forest, Decision Tree).

After preprocessing, three feature selection techniques were applied independently to create three distinct datasets for each model:
- **Random Forest Importance + Correlation Analysis**,
- **Recursive Feature Elimination (RFE)**.

5.2 Model Training and Hyperparameter Tuning

The following machine learning models were trained:
- **K-Nearest Neighbors (KNN)**: Applied on the normalized dataset.

• **Random Forest**: Applied on the standardized dataset.
• **Decision Tree**: Applied on the standardized dataset.

Each model was trained on preprocessed datasets and underwent hyperparameter tuning using GridSearchCV to optimize model performance.

5.2.1 Hyperparameter Tuning with GridSearchCV

GridSearchCV is a technique that performs exhaustive search over a specified parameter grid for a model to identify the optimal hyperparameters. The parameters are selected based on the highest F1 score obtained during cross-validation, ensuring a balanced focus on both precision and recall.

For each model, the following parameter grids were defined:

- **K-Nearest Neighbors (KNN)**:
  - Dataset: Normalized (MinMaxScaler)
  - Parameter Grid:
    - n_neighbors: [3, 5, 7]
    - weights: ['uniform', 'distance']
- **Random Forest**:
  - Dataset: Standardized (StandardScaler)
  - Parameter Grid:
    - n_estimators: [50, 100, 150]
    - max_depth: [None, 10, 20]
    - min_samples_split: [2, 5, 10]
- **Decision Tree**:
  - Dataset: Standardized (StandardScaler)
  - Parameter Grid:
    - max_depth: [None, 10, 20]
    - min_samples_split: [2, 5, 10]

```
Best Parameters for KNN : {'n_neighbors': 3, 'weights': 'uniform'}
Best Parameters for Random Forest : {'max_depth': None, 'min_samples_split': 2, 'n_estimators': 50}
Best Parameters for Decision Tree : {'max_depth': None, 'min_samples_split': 2}
```

Figure 2: Results of GridSearchCV

5.3 Evaluation Metrics

Models were evaluated using the following metrics:

- **Accuracy**: Proportion of correctly classified samples.
- **Precision**: Ratio of true positives to the sum of true and false positives.
- **Recall**: Ratio of true positives to the sum of true positives and false negatives.
- **F1-Score**: Harmonic mean of precision and recall.
- **ROC-AUC**: Area under the ROC curve, indicating the model's ability to distinguish between classes.
- **Confusion Matrix**: Visual representation of true vs. predicted labels.

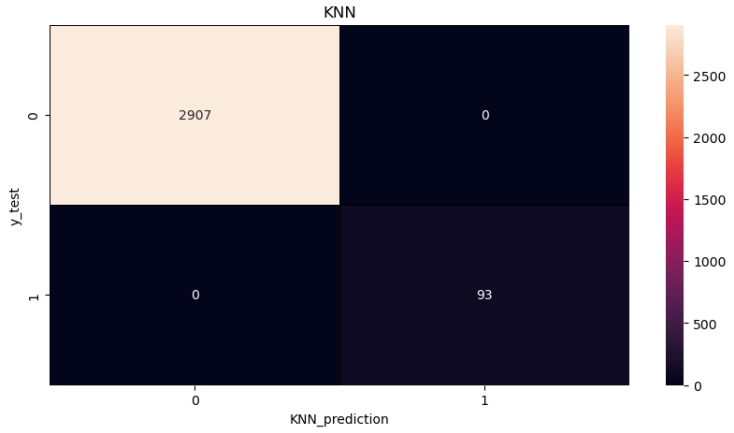
5.4 Results and Analysis

	Model	Method	Accuracy	Recall Score	Precision Score	F1 Score
0	KNN	RF(normalized)	1.0	1.0	1.0	1.0
1	Random Forest	RF(scaled)	1.0	1.0	1.0	1.0
2	Decision Tree	RF(scaled)	1.0	1.0	1.0	1.0

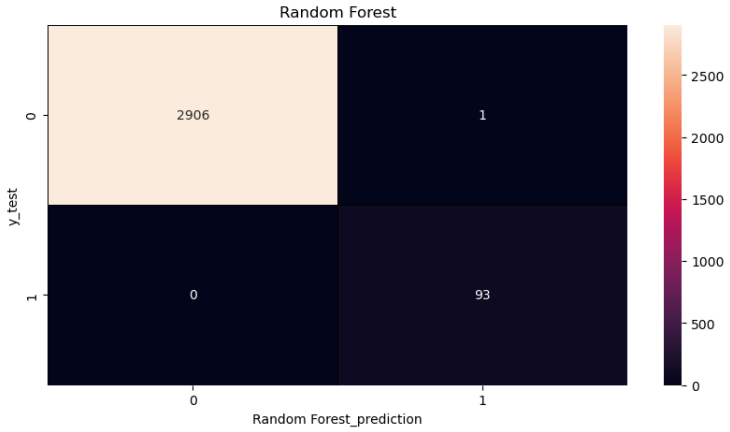
Figure 3: The Interpreted Results

5.5 Confusion Matrix Analysis

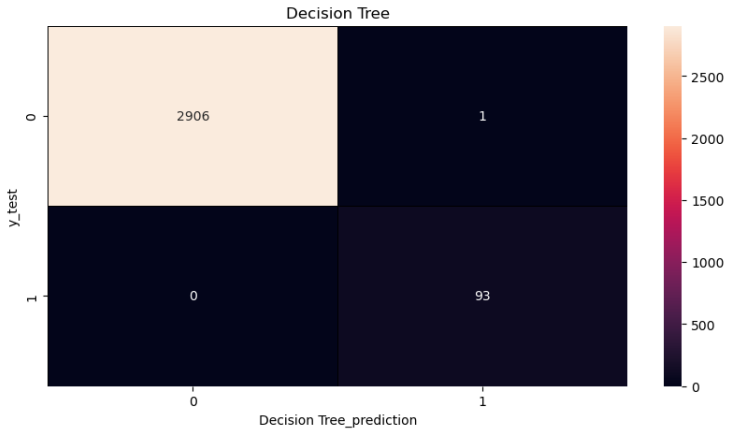
5.5.1 KNN Confusion Matrix



5.5.2 Random Forest Confusion Matrix

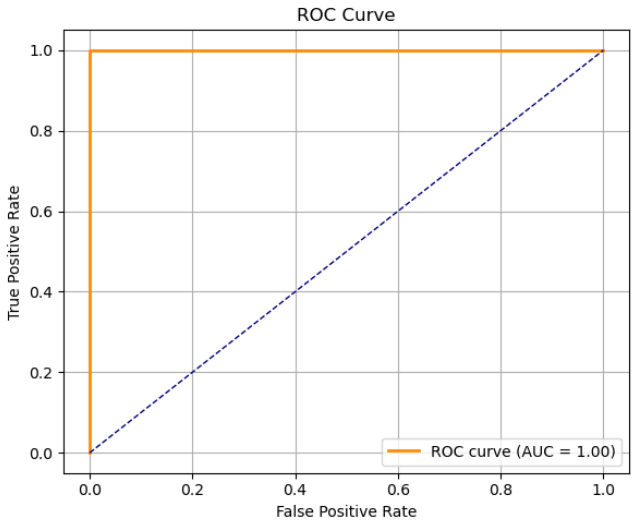


5.5.3 Decision Tree Confusion Matrix

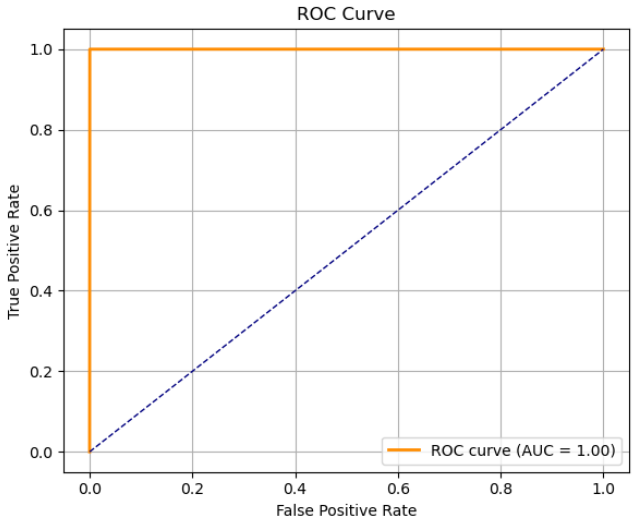


5.6 ROC-CURVE Analysis

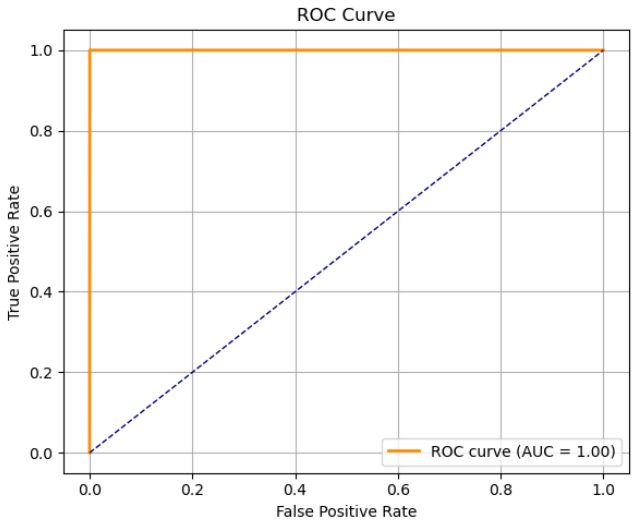
5.6.1 KNN ROC-Curve



5.6.2 Random Forest ROC-Curve



5.6.3 Decision Tree ROC-Curve



6 Conclusion

This project successfully demonstrated the implementation of predictive maintenance using AI techniques in a manufacturing environment. By analyzing sensor data such as air temperature, process temperature, rotational speed, torque, and tool wear, the objective was to predict potential machine failures before they occur.

The project was developed in collaboration with **TÜBİTAK**, emphasizing the significance of AI in enhancing maintenance strategies and minimizing production downtimes.

Three machine learning models — **K-Nearest Neighbors (KNN)**, **Random Forest**, and **Decision Tree** — were applied and compared. Each model underwent hyperparameter tuning using **GridSearchCV**, optimizing their predictive performance based on the F1-score.

The experimental results indicated that:

• The **Random Forest** model achieved the best overall performance, with the highest accuracy and F1-score, as well as the most balanced confusion matrix and ROC-AUC curve. Its ensemble nature effectively handled both categorical and numerical data, leveraging feature importance to reduce overfitting.

• The **KNN model**, while relatively simple to implement, was computationally intensive due to its reliance on distance calculations. However, it still performed adequately with normalized data.

• The **Decision Tree** model provided interpretability and quick training time but was more prone to overfitting, especially when the tree depth was not

properly controlled.

Future Work:

• Integrating additional sensor data to capture more complex machine behaviors.

• Implementing deep learning models (e.g., LSTM or CNN) for temporal pattern analysis in case the dataset is extended with time-series data.

Developing a real-time deployment system that integrates predictive maintenance alerts with factory monitoring dashboards.

In conclusion, the project effectively demonstrated the potential of AI-driven predictive maintenance systems in reducing unexpected equipment failures, optimizing maintenance schedules, and minimizing operational costs in industrial settings.