Predictive Maintenance

Using Machine Learning to Predict Machine Failures

Mohammad Basri | 15 Jan 2025

Introduction

- **Objective:** Predict machine failures using machine learning to improve operational efficiency.

- **Dataset:** AI4I 2020 Predictive Maintenance Dataset [Dataset]. (2020). UCI Machine Learning Repository. https://doi.org/10.24432/C5HS5C.

- **Target Variable:** Binary (Machine failure - 1 for failure, 0 for no failure).

- **Key Challenge:** Class imbalance (only 3.39% failures).

Environment Setup

- **Libraries Used:**
  - Pandas, NumPy (data manipulation).
  - Matplotlib, Seaborn (visualization).
  - Scikit-learn (modeling and evaluation).
  - XGBoost (advanced modeling).

Data Exploration

- **Dataset Overview:**
  - 10,000 rows, 10 columns.
  - No missing values.

- **Key Features:**
  - Numerical: Air temperature, Process temperature, Rotational speed, Torque, Tool wear.
  - Categorical: Type (L, M, H).

- **Target Variable:**
  - Imbalanced (3.39% failures).

Data Preprocessing

- **Steps:**

1. Dropped unnecessary columns (UDI, Product ID, Failure Type).
2. Encoded categorical variables (Type).
3. Renamed target column to Machine failure.
4. Split data into features (X) and target (y).
5. Performed train-test split (80% train, 20% test).

Model Building - Random Forest

- **Model:** Random Forest Classifier.
- **Performance:**
  - **Accuracy:** 98.45%.
  - **Confusion Matrix:**
    - ✓True Negatives: 1934
    - ✓False Positives: 5
    - ✓False Negatives: 26
    - ✓True Positives: 35
  - **Classification Report:**
    - ✓Precision (Class 1): 0.88
    - ✓Recall (Class 1): 0.57
    - ✓F1-Score (Class 1): 0.69

```
# Model evaluation
y_pred = rf_model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.9845
Confusion Matrix:
 [[1934    5]
 [  26   35]]
Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      1939
           1       0.88      0.57      0.69        61

    accuracy                           0.98      2000
   macro avg       0.93      0.79      0.84      2000
weighted avg       0.98      0.98      0.98      2000
```

Model Improvement - XGBoost

- **Issue:** Special characters ([,], <) in column names caused errors.
- **Solution:** Cleaned column names using regex.
- **Model:** XGBoost Classifier.
- **Performance:**
  - **Accuracy:** 98.5%.
  - **Confusion Matrix:**
    - ✓ True Negatives: 1930
    - ✓ False Positives: 9
    - ✓ False Negatives: 21
    - ✓ True Positives: 40
  - **Classification Report:**
    - ✓ Precision (Class 1): 0.82
    - ✓ Recall (Class 1): 0.66
    - ✓ F1-Score (Class 1): 0.73

Confusion Matrix

Column	Predicted 0	Predicted 1
Actual 0	1930	9
Actual 1	21	40

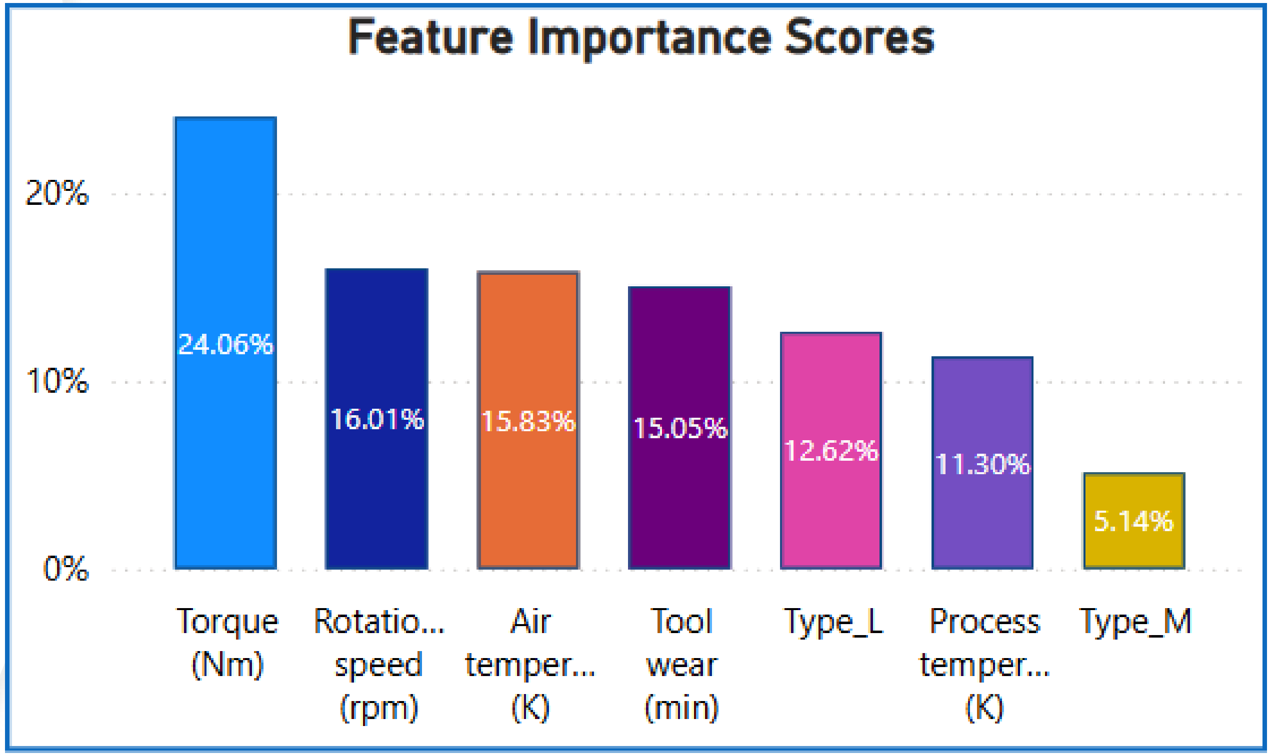
Model Performance Overview

Accuracy	F1-Score	Precision	Recall
0.99	0.73	0.82	0.66

Feature Importance

- **Top Features:**
  - Torque [Nm] (24.06%)
  - Rotational speed [rpm] (16.01%)
  - Air temperature [K] (15.83%)
  - Tool wear [min] (15.05%)

- **Visualization:** Bar plot showing feature importance.



Confusion Matrix Visualization

- **Heatmap:**
  - Visual representation of the confusion matrix.
  - Highlights correct and incorrect predictions for each class.

- **Insight:**
  - Model performs well on the majority class (no failure) but struggles with the minority class (failure).

Confusion Matrix

Column	Predicted 0	Predicted 1
Actual 0	1930	9
Actual 1	21	40

Key Insights

1. **Class Imbalance:**
    - Only 3.39% of samples represent failures.
    - Affects model performance on the minority class.

2. **Feature Importance:**
    - Torque and Rotational speed are the most important features.

3. **Model Performance:**
    - XGBoost performs slightly better than Random Forest.
    - Both models struggle with recall for the minority class.

Recommendations for Improvement

- **Handling Class Imbalance:**
  - Use SMOTE or class weighting.
- **Hyperparameter Tuning:**
  - Optimize model parameters for better performance.
- **Advanced Feature Engineering:**
  - Create new features (e.g., ratios, interactions).
- **Model Interpretability:**
  - Use SHAP values to explain predictions.

Conclusion

- The portfolio demonstrates a comprehensive approach to predictive maintenance using machine learning.

- Key challenges include class imbalance and improving recall for the minority class.