Comparative Analysis of Machine Learning Models for Machine Failure Prediction

By
Kasra Hashemi
and
Piyush Rawat

August 16, 2024

Abstract

In this project we explore the potential usage of machine learning techniques for predicting machine failures in manufacturing, a crucial aspect of predictive maintenance. Using a realistic synthetic dataset, we evaluate the performance of logistic regression, decision tree, and tuned random forest models. Despite challenges posed by class imbalance, the tuned random forest model emerges as the most effective, particularly in terms of the F1 score. Our findings suggest that while machine learning can aid in Stistical Process Control (SPC), careful model selection is essential due to the natural class imbalance, and future research could further refine these approaches by considering additional techniques and causal models.

Contents

1 Introduction

In manufacturing and industrial engineering, Statistical Process Control (SPC) has traditionally relied on conventional statistical methods to monitor and control production processes. However, with advancements in data science, there is growing interest in integrating machine learning techniques into SPC to enhance predictive capabilities.

This project investigates the potential of machine learning methods in predicting machine failure. The goal is to assess the effectiveness of these approaches in aiding predictive maintenance applications.

Given the challenges in acquiring real-world maintenance and manufacturing data due to proprietary restrictions and data sensitivity, we use a realistic synthetic dataset provided by S. Matzka [1]. This dataset simulates real-world conditions and allows us to rigorously test the performance of these machine learning models. We begin by employing logistic regression, a widely used traditional machine learning technique, followed by the more complex decision tree and random forest methods, to explore their potential in enhancing SPC practices. The models are created using the Scikit-Learn [2] and Imbalanced-Learn [3] libraries in Python 3.12.

2 Exploratory Data Analysis

Due to the synthetic nature of the dataset, no major cleaning/wrangling is required. The Python package YData Profiling is used to perform basic EDA. The dataset consists of the following variables:

- UDI: row index
- Type: defines the level of work to be done on the workpiece, consisting of high(H), medium(M), low(L)
- Product ID: the workpiece type (H/M/L) and a variant-specific serial number
- Air temperature [K]: ambient air temperature measured in Kelvin at the time of the process
- Process temperature [K]: temperature of the machine in Kelvin during manufacturing
- Rotational speed [rpm]: rotational speed of the cutting tool
- Torque [Nm]: torque applied to the workpiece by the cutting tool
- Tool wear [min]: how long the cutting tool has been in use measured in minutes
- Machine failure: a binary variable indicating whether the process failed or not which consists of five failure modes:
  - TWF: Tool Wear Failure
  - HDF: Heat Dissipation Failure
  - PWF: Power Failure
  - OSF: Overstrain Failure
  - RNF: Random Failure

If any of the five failure modes occur, then a machine failure occurs.
Below are descriptive statistics and distributions of the relevant variables:

	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	300.004930	310.005560	1538.776100	39.986910	107.951000
std	2.000259	1.483734	179.284096	9.968934	63.654147
min	295.300000	305.700000	1168.000000	3.800000	0.000000
25%	298.300000	308.800000	1423.000000	33.200000	53.000000
50%	300.100000	310.100000	1503.000000	40.100000	108.000000
75%	301.500000	311.100000	1612.000000	46.800000	162.000000
max	304.500000	313.800000	2886.000000	76.600000	253.000000

Table 1: Descriptive statistics for selected variables

From the pair plots, we see that air temperature and process temperature are highly correlated with each other, with a Pearson correlation coefficient of 0.876. We therefore drop the air temperature variable and only keep process temperature as it is more relevant in determining process failure.

Looking at the class balance, we find that the dataset is highly imbalanced:

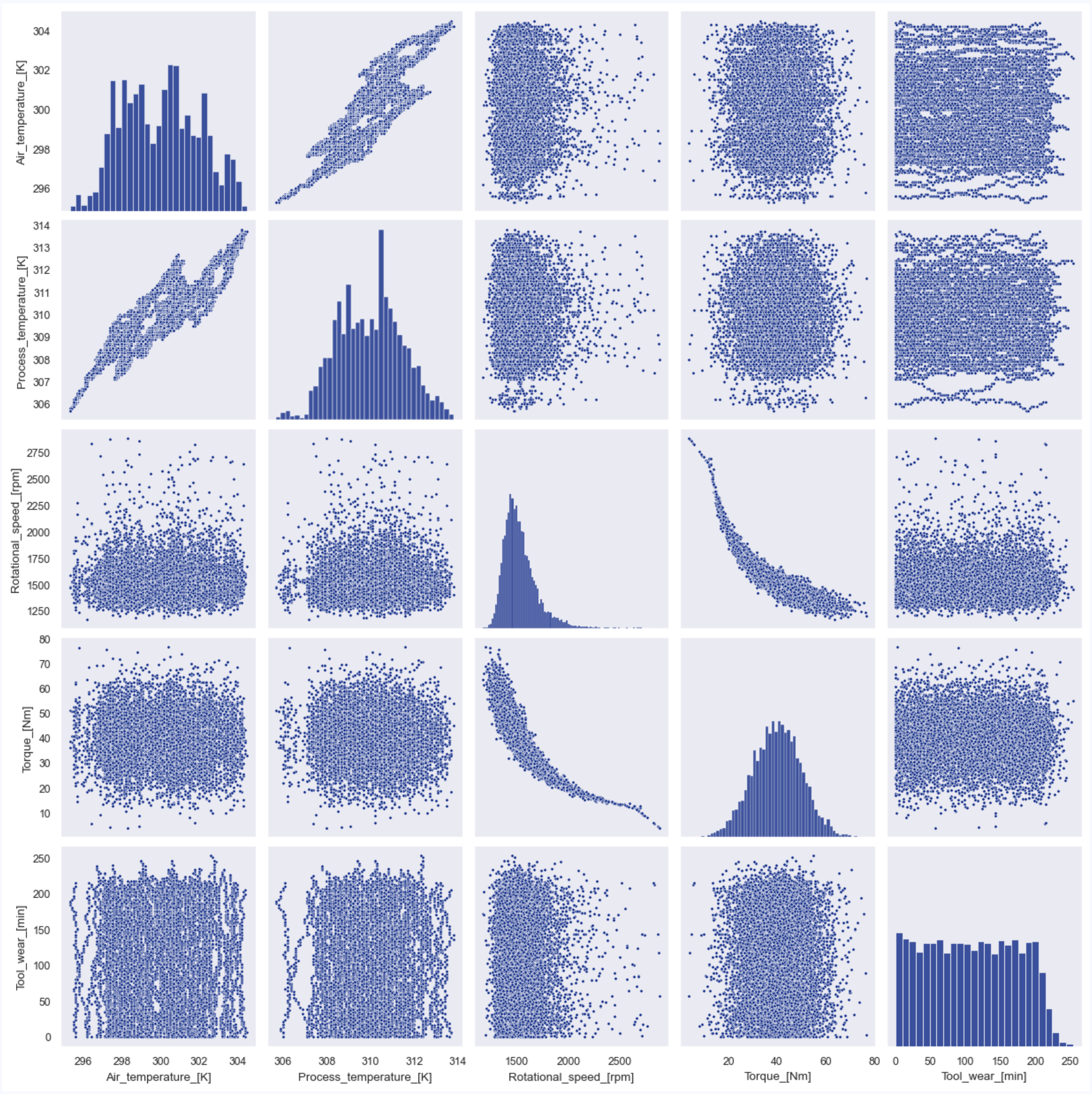

Figure 1: Pairplot

- 0: 9661 (78.6%)
- 1: 339 (21.4%)

It is also important to mention that it is possible for a process to have multiple simultaneous failure modes (e.g. `TWF`=`HDF`=1). In fact, in the used dataset, there are 24 instances of multimodal failures.

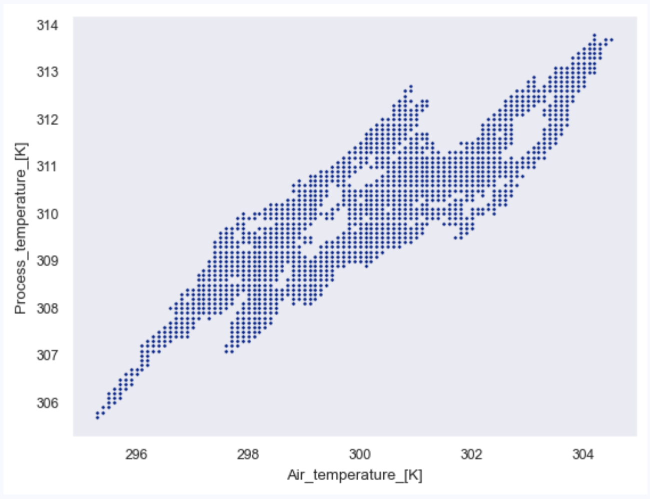

Figure 2: Air temperature vs. Process temperature

3 Logistic Regression Approach

We create two different logistic regression models, one to predict machine failure, and the other to predict the failure mode. In order to address the class imbalance, we use Imbalanced-learn's SMOTEENN to balance the dataset for model training. SMOTEENN combines SMOTE (Synthetic Minority Over-sampling Technique) with the ENN (Edited Nearest Neighbours) technique in order to achieve a balance of over- and undersampling of the minority class and the majority class respectively. Doing so results in the training dataset having a much better balance (from 96.4% to 47.7%).

Before training the logistic regression models, we additionally drop the rotational speed variable as it is highly correlated with torque and therefore can potentially result in issues in the estimation of the models. Due to the correlation between rotational speed and torque, the estimations will still be able to capture the variability in the data.

We then use Scikit-learn to estimate the logistic regression models. Despite employing SMOTEENN, the model predicting the individual failure modes performs rather poorly. The model predicting machine failure performs marginally better, with the following results:

	precision	**recall**	**f1-score**	**support**
0	0.99	0.68	0.81	2428
1	0.07	0.04	0.07	72
accuracy			0.68	2500
macro avg	0.53	0.73	0.47	2500
weighted avg	0.96	0.68	0.79	2500

Table 2: Logistic Regression Model

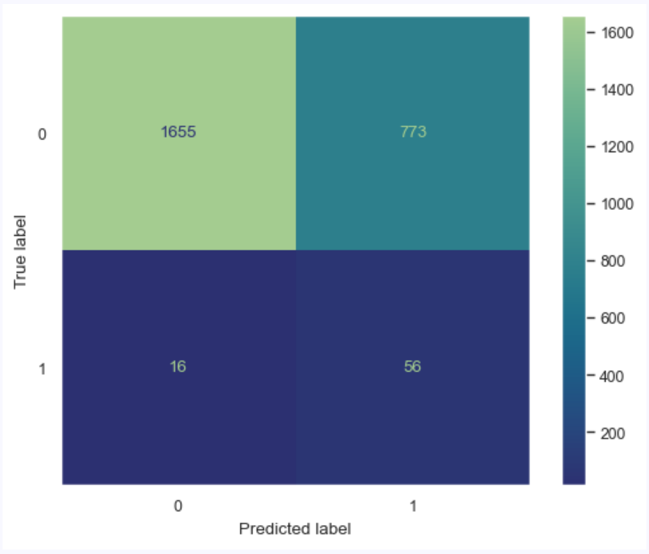

Figure 3: Logistic Regression Model Confusion Matrix

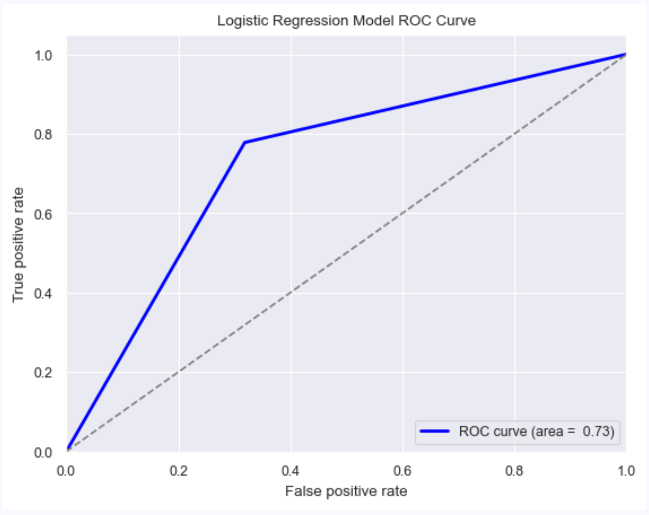

Figure 4: Logistic Regression Model ROC Curve

4 Decision Tree Approach

Decision trees are more robust to collinearity compared to logistic regression models, and as such, we include rotational speed in the independent variables as well. We also only consider machine failure and not failure modes in order to improve model performance. As before, the same SMOTEENN method is used in order to balance the classes. Additionally, balanced class weighting is used which adjusts weights inversely proportional to class frequencies in the input data. The results of the model are then tested using cross-validation to ensure robustness:

	precision	**recall**	**f1-score**	**support**
0	0.99	0.92	0.95	2428
1	0.40	0.30	0.34	72
accuracy			0.91	2500
macro avg	0.60	0.83	0.64	2500
weighted avg	0.97	0.91	0.94	2500

Table 3: Decision Tree Model

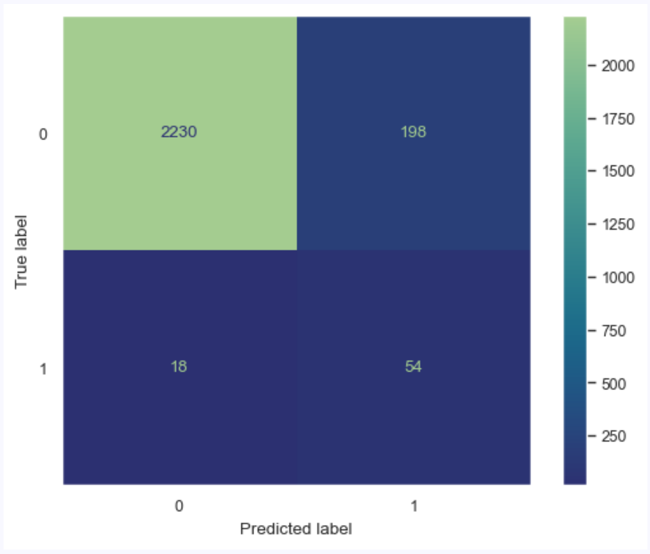

Figure 5: Decision Tree Confusion Matrix

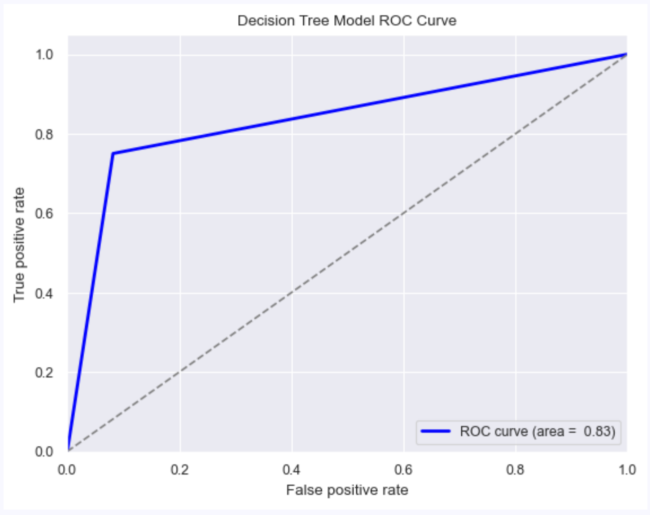

Figure 6: Decision Tree Model ROC Curve

5 Random Forest Approach

Lastly, we use an ensemble technique, namely a random forest model. As before, SMOTEENN is used to balance the dataset, followed by an exhaustive grid search to find the optimal combination of hyperparameters for the random forest model and then use these parameters to estimate the model. We then use stratified k-fold cross-validation to confirm that the results are robust.

	precision	**recall**	**f1-score**	**support**
0	0.99	0.90	0.95	2428
1	0.08	0.41	0.54	72
accuracy			0.90	2500
macro avg	0.60	0.89	0.64	2500
weighted avg	0.97	0.90	0.93	2500

Table 4: Tuned Random Forest Model

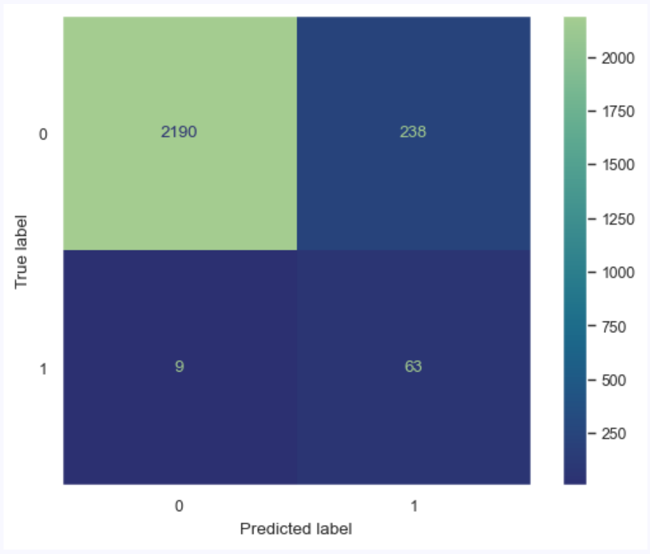

Figure 7: Random Forest Confusion Matrix

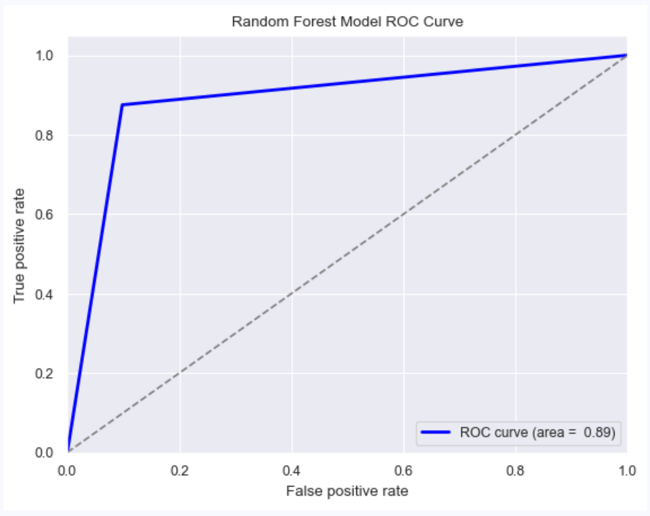

Figure 8: Random Forest Model ROC Curve

6 Results and Evaluation

As can be seen from the model results, the highly imbalanced nature of the dataset makes it rather challenging to build accurate classification models. Nevertheless, we observe a significant improvement in the random forest model compared to the other two models.

Metric	**Class**	**Precision**	**Recall**	**F1-Score**
Logistic Regression	0	0.99	0.68	0.81
	1	0.07	0.04	0.07
	Accuracy		0.47	
Decision Tree	0	0.99	0.92	0.95
	1	0.40	0.30	0.34
	Accuracy		0.64	
Tuned Random Forest	0	1.00	0.90	0.95
	1	0.80	0.41	0.54
	Accuracy		0.64	

Table 5: Model Performance Comparison

With regards to feature importance, the decision tree and the random forest models align well:

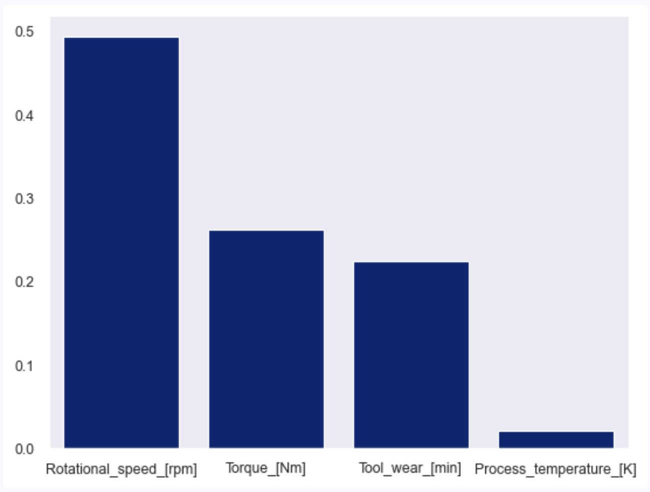

Figure 9: Decision Tree Feature Importance

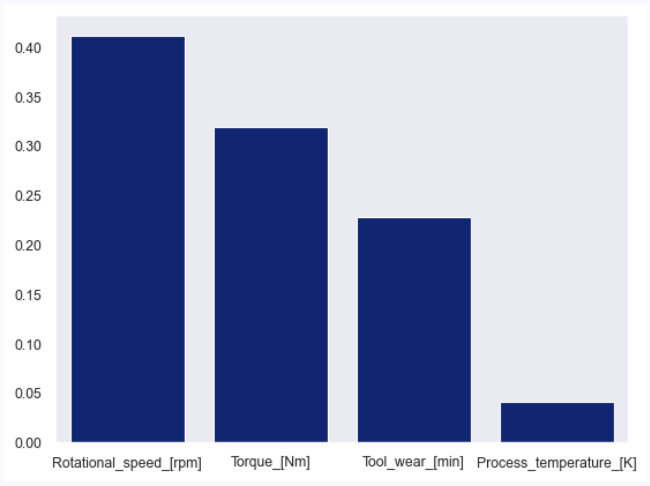

Figure 10: Random Forest Feature Importance

There is a clear relationship between the rotational speed, torque, and machine failure. This result is expected, because

- there is already a strong known correlation between rotational and torque
- the correct setting of rotational speed and torque in all manufacturing operations is a known factor when it comes to premature tool wear and workpiece finish quality

7 Conclusion

In this project, we demonstrated that the tuned random forest model outperforms the other two models, particularly when evaluated using F1 and ROC AUC scores—an important consideration given the significant class imbalance. Future studies could explore alternative modeling techniques to enhance effectiveness. However, as noted in [1], predictive models in the context of predictive maintenance are generally expected to be explainable, which makes more complex approaches like neural networks less suitable, despite their potential for higher predictive power. Instead, simulation techniques may offer more accurate predictions and better explainability.

References

[1] S. Matzka, "Explainable artificial intelligence for predictive maintenance applications," in *2020 third international conference on artificial intelligence for industries (ai4i)*, pp. 69–74, IEEE, 2020.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017.