Failure prediction for APU's on a Metro System

Aaryadev Ghosalkar

November 2, 2023

Abstract

Predictive maintenance is vital for ensuring the reliable operation of metro transportation systems, enhancing passenger safety, and minimizing service disruptions. In this research, we focus on the challenging task of predicting failures on the Air Processing Unit (APU) on a metro system at least two hours in advance. An APU is a component on the metro responsible for compressing and storing compressed air for use on various functions on the metro such as the suspension. Our study encompasses a comprehensive comparison of various machine learning models, including Support Vector Machines (SVM), Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM) networks, and transformer-based models. These models are evaluated for their performance in predicting impending failures using various metrics such as accuracy, precision, recall and F1-Score for classification and Mean Squared Error (MSE) for Regression, providing valuable insights into their applicability in this context.

Additionally, we introduce a practical demonstration of a web API designed to showcase how the model would be deployed in a real world situation. This API serves as a testament to the feasibility of deploying predictive maintenance models in production settings.

It is important to note that while our research offers valuable insights into model performance and deployment readiness, we refrain from proposing a final production model. This limitation arises from the complex domain knowledge required for selecting an optimal model and the to keep the paper more general since the requirements of various transportation companies need not be the same. Nevertheless, our findings contribute to the ongoing efforts to bolster metro system reliability and safety through proactive failure prediction.

Keywords: Metro system, predictive maintenance, failure prediction, machine learning, deep learning, model deployment.

1 Literature Review

The initial phase of data extraction was undertaken by the work conducted by Veloloso et al [1]. In their work they expounded upon the operational principles of diverse sensors and articulated the central objective of predicting metro system failures. Moreover, they explained the precise criteria stipulated by the railway company for the assessment of the predictive model, thus serving as an instrumental starting point for the subsequent research endeavor.

The findings of Azure ML Team at Microsoft [2], show that there exist two primary approaches to tackle our problem: binary classification or failure prediction, where the objective is to predict whether the metro system will fail within the next two hours (yes or no), and regression, which in the context of our research pertains to predicting the number of cycles or amount of time before the metro system experiences a failure, or needs to be repaired. Also known as Remaining Useful life (RUL) Prediction. This paper will undertake a comparative analysis of both these approaches.

Davari et al [3] have conducted a survey on data-driven or Machine Learning (ML) and Deep Learning (DL) approaches to PdM. In their work they have compared both approaches to we will expand on this by providing more insight into which approaches are viable in the context for metro systems.

1.1 Failure Prediction

Numerous traditional ML algorithms are applied in the realm of failure prediction. As detailed in the works of Chaudhuri et al [4], where Support Vector Machines (SVM) and various SVM variants were employed to classify vehicles into three distinct risk categories: Immediate, Long-term, and Medium-term. This approach is simple and the results are easy to understand.

Convolution Neural Networks (CNN) can be applied to time series data, typically used for images, requires a method for converting time series into image-like representations using GAF (Gramian Angular Field) feature transformation, which facilitates the conversion of one-dimensional time series data into images. Leveraging this transformation for PdM Silva reported an accuracy rate of 93% on the SF103 dataset [5].

Long Short-Term Memory (LSTM) networks have gained prominence for failure prediction. Nguyen and Medjaher employed LSTM to predict failures by quantifying the probability of system failure within a specified time window [6]. This approach holds promise for our problem, as it not only offers the potential for interpretability but also grants flexibility to the company in defining the severity thresholds for failure prediction.

1.2 Remaining Useful Life Estimation

An alternative perspective on the problem involves estimating the Remaining Useful Life (RUL), which offers a versatile approach. This approach allows for the setting of thresholds to define significant risk levels based on the transportation companies needs this is usually done by monitoring specific variables that serve as reliable indicators of system deterioration however this approach requires domain knowledge about the features that can be used to predict the degradation of the system, due to this limitation we will be calculating the time to failure, the number of hours before failure based on the known failures.

Prior to 2011, a significant portion of research in Remaining Useful Life (RUL) prediction predominantly relied on statistical methods. Si et al.'s study exemplifies the utilization of various statistical techniques that were commonly employed, including linear regression and probabilistic models such as Markov models, in the context of RUL prediction [7].

In 2016 with the rise of Deep Learning, Wang et al. compared a more data driven approach such as Deep Neural Networks (DNN) and Shallow Neural Networks (SNN) in predicting lubricant pressure for wind turbines within a farm [8]. It's worth noting that selecting the appropriate variable for monitoring often requires domain expertise.As mention earlierwe will take a slightly different approach in our study thus circumventing the need for domain-specific knowledge in variable selection.

Several models originally designed for failure prediction have found applicability in RUL estimation. For instance, Boujamza and Elhaq explored the utilization of an attention-based LSTM model for RUL estimation using the C-MAPSS Dataset [9]. This adaptation suggests the possibility of exploring other attention-based models, such as transformers, for similar tasks in RUL estimation, expanding the scope of potential approaches for our research.

2 Background

The dataset used in this study was provided by a company and contains sensor readings from the APU on metro trains. The data consists of sensor readings collected at a frequency of 1 Hz (1 reading per second) from 20 sensors within the APU. True labels for the observed events are also included in the dataset. The following features have been extracted from the data for analysis:

Table 1: Summary Statistics and Data Types of Sensor Readings

Column	Data Type	Mean	Median	Min	Max
timestamp	DateTime	NA	NA	NA	NA
TP2	Float	1.0103	-0.008	-0.03	10.806
TP3	Float	8.9730	8.99	0.938	10.38
H1	Float	7.9479	8.86	-0.034	10.368
DV_Pressure	Float	-0.0192	-0.026	-0.036	8.11
Reservoirs	Float	1.5892	1.606	1.35	1.726
Oil_Temperature	Float	66.2864	67.2	20.8	80.175
Flowmeter	Float	20.2065	19.0123	18.8347	37.0083
Motor_Current	Float	2.1342	3.665	-0.01	9.3375
COMP	Bool	0.8829	1	0	1
DV_Electric	Bool	0.1171	0	0	1
TOWERS	Bool	0.9413	1	0	1
MPG	Bool	0.8829	1	0	1
LPS	Bool	0.0079	0	0	1
Pressure_Switch	Bool	0	0	0	0
Oil_Level	Bool	0.0031	0	0	0
Caudal_Impulses	Bool	-7.7623	0	0	1
gpsLat	Float	37.0052	-8.6583	-8.6941	41.2401
gpsLon	Float	9.9405	41.1855	0	216
gpsSpeed	Float	0.8984	0	0	1
gpsSignal	Float	NA	1	0	1

It is crucial to acknowledge that these values are approximations and are solely intended to provide a broad overview of the data distribution. Furthermore, it is essential to emphasize that the dataset is does not contain any missing values. Additional information about the data is provided by the original authors veloso et al[1]

3 Data preprocessing

In the data preprocessing stage, two distinct copies of the dataset are being created, each tailored for a different Machine Learning (ML) task:

3.1 Multiclass Classification

In this task, the primary objective is to classify data into three distinct states: normal operational conditions, pre-failure (signifying a state occurring two hours before an impending failure), and failure. A data point is designated as a "failure" state if it falls within the time window of documented failure events. Conversely, if a data point is at least two hours from any documented failure event, it is categorized as "normal" state. with these transformation we have a target column for a multiclass classification problem.

We choose this approach due to the inherent challenge of predicting a continuous timestamp directly using a machine learning model. Therefore, a process of discretization is essential. However, it is important to highlight the significance of the chosen threshold, denoted as N_{Hours} .It is worth noting that setting this threshold too high may lead to a potential decrease in accuracy, as the model might encounter difficulties distinguishing between normal and pre-failure states. We use $N_{Hours} = 2$ since the dataset description mentioned the need to predict failures atleast 2 hours in advance

after applying the above mentioned transformation we obtain the following class distribution.

Table 2: Class Distribution

Class	Instances
0 (Normal)	10539882
1 (Pre Failure)	212106
2 (Failure)	21600

Table 2 reveals a significant imbalance within the dataset, which poses a risk of the model favoring one class and achieving a high accuracy by simply predicting that class. In our case there are 10 539 882 Data points which accounts for 97.83% in other words if the model classifies all points as "Normal" the accuracy will be 97.8%.

To address this issue, we must employ data resampling techniques. Undersampling involves reducing the number of instances in the majority class, while oversampling entails increasing the instances in the minority class.

In our case, due to the dataset's substantial size and the pronounced class imbalance, we have choose undersampling. This decision is based on the understanding that oversampling, would generate many redundant data points, thereby increasing the risk of overfitting. [10].

3.2 Regression

To prepare the data for regression, we calculated the temporal distance between each data point and the nearest failure event within the dataset. It is worth noting that this method, while serviceable, assumes a linear degradation pattern for components and may benefit from more sophisticated approximations to enhance model performance. Nevertheless, it serves as a foundational step in our analysis. done with the following algorithm.

Algorithm 1 Find Time Till Failure

```
for $(start, end)$ in $failure\_periods$ do
    if $curr\_time < start$ then
        $Hours\_till\_Failure \leftarrow \frac{(start - curr\_time).total\_seconds()}{3600}$
        return $Hours\_till\_Failure$
    end if
end for
$Hours\_till\_Failure \leftarrow 0$

for each row in $Reg\_df$ do
    $curr\_time \leftarrow row["timestamp"]$
    $Reg\_df[row]["Hours\_till\_Failure"] \leftarrow find\_time\_till\_failure(curr\_time)$
end for
```

Where $failure\_periods$ is a list of pairs $(start, end)$, it is important to note that $failure\_periods$ is sorted according to $start$ (The first element in the pair).

In the context of the regression approach, the use of undersampling is typically unnecessary, as it does not pertain to a classification problem. Instead, our primary focus is on evaluating the model's mean squared error (MSE). Given our objective of predicting a failure at least 2 hours in advance, we aim for the mean squared error to be approximately around 4 (2^2), which signifies that, on average, the model is deviating by 2 hours in its predictions. This MSE threshold of 4 serves as a meaningful benchmark for our regression model.

4 Machine Learning for Failure Prediction

As mentioned in the abstract, the comparative analysis will encompass several machine learning (ML) models and deep learning models. It is important to note that all figures presented in this study are computed using the test subset of the resampled data. Specifically, 30% of the entire resampled dataset was designated for testing purposes. Furthermore a random seed of 42 was employed to ensure consistency in the selection of the test data.

Prior to delving into the model evaluations and results, it is imperative to establish a theoretical baseline. This baseline ensures that the models have learned meaningful patterns.

We three classes, each having approximately equal instances after resampling, thus we can reasonably assume that the probabilities of encountering each class are roughly equal, denoted as $P(Class_1) \approx P(Class_2) \approx P(Class_3)$. therefore, a model making random guesses would have an expected accuracy of 0.33, as it should have a $\frac{1}{3}$ chance of correctly predicting the class. Therefore, any accuracy significantly surpassing 0.33 on the testing data would signify that the model has successfully identified discernible patterns and is not relying on random chance for predictions.

4.1 Support Vector Machines

4.1.1 Background

The Support Vector Machine (SVM) is a classification algorithm that seeks to delineate data points using a hyperplane. By default, SVM is utilized to establish a linear decision boundary. However, through using the kernel trick, where a kernel transformation is applied to all data points before training the SVM, enabling the model to learn non-linear decision boundaries [11]. For this problem we are using Gaussian RBF kernel defined as follows:

$$\phi_\gamma(x, l) = \exp(-\gamma \|x + l\|) \tag{1}$$

In the absence of a kernel, the SVM classifies data points based on a decision function, which is defined as follows:

$$\hat{y} = \begin{cases} 0 & \text{if } w^T x + b < 0 \\ 1 & \text{if } w^T x + b \geq 0 \end{cases} \tag{2}$$

Where the weight vector w controls the width of the decision boundary, the smaller the value of w the larger the decision boundary, thus we would like to minimize $\|w\|$. However, if we also want to avoid any margin violation (hard margin), then we need the decision function to be greater than 1 for all positive training instances, and lower than –1 for negative training instances. This denotes a Hard Margin SVM.

Typically, two types of SVMs are employed: hard margin and soft margin SVMs. In a hard margin SVM, no data points are permitted within the margin. In contrast, a soft margin SVM allows for data points to be situated within the margin, and the extent to which this is allowed is regulated by a hyperparameter denoted as C A smaller value of C enforces a stricter margin, and $C = 0$ signifies a hard margin, where no points are permitted within the margin [11]. In your specific case, you have chosen $C = 1$. Hard margin SVM are more prone to overfitting. To train a softmargin SVM we generally add a slack variable to the minimization equation.

4.1.2 Results

Table 3: SVM Results

	Normal	**Pre Failure**	**Failure**
Precision	0.55	0.60	0.52
Recall	0.62	0.62	0.36
F1 - Score	0.58	0.61	0.43
Accuracy			0.57

The SVM model, without any hyperparameter optimization, has the worst performance, with an accuracy of 57%. However this accuracy is still higher than the theoretical baseline of 33.33% established at the beginning of Section 4. which means even this model is learning some pattern.

In terms of precision for class 1 (Pre-Failure), the SVM achieved a precision of 0.6, indicating that it correctly classifies 60% of instances as pre-failure. This performance is particularly useful for us.

It's worth noting that the SVM appeared to struggle with distinguishing between normal and pre-failure states, with a misclassification rate of approximately 32%. Additionally, it exhibited limited success in correctly predicting actual failure cases, achieving a correct prediction rate of only about 36%.

These observations are supported by the confusion matrix presented below, which highlights the model's performance with respect to different classes.

$$\begin{bmatrix} 0.61969674 & 0.26742227 & 0.11288099 \\ 0.32976412 & 0.61530106 & 0.05493482 \\ 0.37874659 & 0.26006661 & 0.3611868 \end{bmatrix}$$

4.2 Decision Trees

4.2.1 Background

A decision tree is a widely used machine learning algorithm for classification tasks. It takes the form of a tree-like structure, where internal nodes represent decisions or tests based on input features, branches represent the outcomes of these decisions or tests, and leaf nodes signify class labels assigned to data instances.

It's essential to emphasize that, in theory, finding the most optimal decision tree is an NP-Complete problem. This means that, as the size and complexity of the dataset and the tree structure increase, the computational resources required to find the best tree become prohibitively large. NP-Complete problems are a class of problems for which no efficient polynomial time algorithm exists to find the exact optimal solution, making it necessary to rely on heuristics and approximations when building decision trees for practical applications [12].

Decision trees can be trained using various algorithms, and among them are CART (Classification and Regression Tree) and ID3 (Iterative Dichotomiser 3). For this study, the decision tree model will be constructed using the CART algorithm which has a cost function given by the following equation:

$$J(k, t_k) = \frac{m_{left}}{m} \cdot G_{left} + \frac{m_{right}}{m} \cdot G_{right} \tag{3}$$

Where $G_{left/right}$ is the Gini impurity and $m_{left/right}$ is the number of instances on left or right.

Gini Impurity is defined as follows:

$$G_i = 1 - \sum_{k=1}^{n} P_{(i,k)}^2 \tag{4}$$

Where $P_{(i,k)}$ is is the ratio of class k instances among the training instances in the ith node

4.2.2 Results

Table 4: Decision Tree Results

	Normal	**Pre Failure**	**Failure**
Precision	0.70	0.69	0.51
Recall	0.68	0.70	0.52
F1 - Score	0.69	0.70	0.52
Accuracy			0.66

The decision tree model represents an improvement over the SVM, achieving an accuracy of 66%. The precision for pre-failure cases has also seen a substantial increase, reaching 69%. Despite these improvements, the decision tree still faces challenges in reliably predicting failure cases. However, it's an improvement of almost 17% compared to the SVM.

Additionally, the decision tree model demonstrates superior computational efficiency we will go over the computation complexity of the tested algorithms later, particularly for large datasets. It's important to mention that during hyperparameter tuning, different criteria were explored, including log_loss, entropy, and Gini impurity, but the impact on model performance was relatively minor.

The confusion Matrix for the decision tree was as follows:

$$\begin{bmatrix} 0.68019605 & 0.18456119 & 0.13524276 \\ 0.18171943 & 0.70251397 & 0.1157666 \\ 0.22615804 & 0.25158946 & 0.5222525 \end{bmatrix}$$

4.3 Random Forest

4.3.1 Background

The Random Forest algorithm represents an improvement over decision trees classifier. It is classified as an ensemble technique, which means it uses predictions from multiple models. Specifically, the Random Forest algorithm involves training a "forest" of numerous decision tree classifiers. In this instance, 100 different decision trees are used. During inference, the algorithm determines the most common class among the predictions.

4.3.2 Results

Table 5: Random Forest Results

	Normal	**Pre Failure**	**Failure**
Precision	0.73	0.69	0.70
Recall	0.72	0.78	0.54
F1 - Score	0.72	0.73	0.61
Accuracy			0.70

The Random Forest indeed offers improvements over decision trees in various aspects. It has shown an increase in accuracy, precision, and recall.

During the hyperparameter optimization phase, a slight accuracy increase of 1% was achieved by experimenting with different criteria, such as entropy.

4.4 Neural Networks

4.4.1 Background

4.4.2 Results

4.5 LSTM Networks

4.5.1 Background

4.5.2 Results

5 Machine Learing for RUL Estimation

6 Deployment and future scope

References

[1] B. Veloso, R. P. Ribeiro, J. Gama, and P. M. Pereira, “The metropt dataset for predictive maintenance,” *Scientific Data*, vol. 9, no. 1, p. 764, Dec 2022. [Online]. Available: https://doi.org/10.1038/s41597-022-01877-3

[2] AzureML team (Microsoft). (2015, April) Predictive maintenance: Step 1 of 3 - data preparation and feature engineering. Accessed: 23/09/2023. [Online]. Available: https://gallery.azure.ai/Experiment/Predictive-Maintenance-Step-1-of-3-data-preparation-and-feature-engineering-2

[3] N. Davari, B. Veloso, G. Costa, P. Pereira, R. Ribeiro, and J. Gama, “A survey on data-driven predictive maintenance for the railway industry,” *Sensors*, vol. 21, p. 5739, 09 2021.

[4] A. Chaudhuri, “Predictive maintenance for industrial iot of vehicle fleets using hierarchical modified fuzzy support vector machine,” *arXiv preprint arXiv:1806.09612*, 2018.

[5] W. Silva, “Cnn-pdm: A convolutional neural network framework for assets predictive maintenance,” 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:260505498

[6] K. T. Nguyen and K. Medjaher, “A new dynamic predictive maintenance framework using deep learning for failure prognostics,” *Reliability Engineering & System Safety*, vol. 188, pp. 251–262, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0951832018311050

[7] X.-S. Si, W. Wang, C.-H. Hu, and D.-H. Zhou, “Remaining useful life estimation – a review on the statistical data driven approaches,” *European Journal of Operational Research*, vol. 213, no. 1, pp. 1–14, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221710007903

[8] L. Wang, Z. Zhang, H. Long, J. Xu, and R. Liu, “Wind turbine gearbox failure identification with deep neural networks,” *IEEE Transactions on Industrial Informatics*, vol. 13, no. 3, pp. 1360–1368, 2016.

[9] A. Boujamza and S. Lissane Elhaq, “Attention-based lstm for remaining useful life estimation of aircraft engines,” *IFAC-PapersOnLine*, vol. 55, no. 12, pp. 450–455, 2022, 14th IFAC Workshop on Adaptive and Learning Control Systems ALCOS 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896322007546

[10] C. Ellis, “Oversampling vs undersampling for machine learning,” 01 2023. [Online]. Available: https://crunchingthedata.com/oversampling-vs-undersampling/

[11] A. Geron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2nd ed. O'Reilly Media, Inc., 2019.

[12] H. Laurent and R. L. Rivest, "Constructing optimal binary decision trees is np-complete," *Information processing letters*, vol. 5, no. 1, pp. 15–17, 1976.