Statistical Analysis for Enhanced Aircraft Engine Upkeep

A Comparative Assessment of Regression and Classification

s437451 – Julian Gerald Dcruz

Department: Applied Artificial Intelligence

Date: 25/11/23

Word Count: 3680

Information categorisation:
Open

Contents

Statistical Analysis for Enhanced Aircraft Engine Upkeep

A Comparative Assessment of Regression and Classification

1. Introduction

In the rapidly evolving aerospace industry, the prediction of equipment maintenance requirements before failure of components can save considerable time and resources. Traditional approaches like Corrective Maintenance Strategies, where maintenance is performed post-failure, and Preventive Maintenance Strategies, involving periodic maintenance cycles, although established, often incur excessive time and resource expenditure. This report delves into predictive maintenance strategies, a paradigm shift introduced by the advent of Industry 4.0. This revolution has integrated digital and physical systems, thereby making prognostics and health management (PHM) an essential aspect of smart manufacturing and big data frameworks. Predictive maintenance stands out by facilitating reliable solutions for the continuous monitoring of industrial equipment health, significantly enhancing the efficiency and reliability of aircraft engine upkeep. A key benefit of predictive maintenance is its ability to reduce maintenance costs while enhancing operator safety, increasing production, extending the remaining useful life of the components. This is achieved by leveraging statistical and machine learning algorithms to forecast equipment failures, thereby optimizing resource allocation, and minimizing unnecessary maintenance activities. Unlike traditional corrective and preventive maintenance strategies, predictive maintenance focusses on early fault detection and condition-based interventions. This method is based on the continuous monitoring of the state of the machine, with maintenance activities scheduled only when necessary. Such an approach ensures that maintenance is conducted in the final cycles of a component's life, leading to optimal resource utilization [1].

1.1 Problem Description

This report focuses on the application of advanced statistical methods to predict the time-to-failure of aircraft engines and classify their operational status, using a dataset that simulates a run-to-failure scenario. The report includes a comparative study between the two techniques: regression and classification, employed for the predictive maintenance of the aircraft engines from the provided data. The study's outcome is expected to contribute significantly to the field of predictive maintenance by providing insights into the effective use of regression and classification techniques in real-world scenarios.

1.2 Dataset Overview

The dataset contains information about 100 engines and their run-to-failure scenarios represented as the cycles and time-to-failure (TTF) cycle, also data from 4 sensors for each cycle of per engine till it runs to failure. Each engine also has a binary classification label attached to it, where the classification label is set to 1 when the TTF is within the last 30 cycles of engine operation. It is assumed that the engine progressing degradation pattern is reflected in its sensor measurements. A validation dataset has also been provided which will be used to assess the performance of each predictive maintenance technique chosen.

For this study, the training dataset and the test dataset is in a ".csv" format, a suitable format for data analysis.
The training dataset, provided as 'train_selected.csv' contains the following columns:

- 'id': represents the engine id.
- 'cycle': represents the engine cycle recorded till failure for each engine.
- 's1-4': represents the 4 sensors that are assumed to provide a pattern reflecting progressive engine degradation.
- 'ttf': represents the time-to-failure, which is the remaining useful life for each engine.
- 'label_bnc': represents a binary label which has value 1 when 'ttf' is within the last 30 cycles of engine operation.

The testing dataset was modified to include the similar columns as seen in training so that both regression and classification techniques could be conducted. The 'ttf' label from 'PM_truth.txt' was concatenated to the 'test_selected.csv' and the 'label_bnc' column was created for the test dataset by assigning value of 1 when the 'ttf' for each engine is within the last 30 cycles of engine operation.

2. Data Analysis

2.1 Preliminary Analysis

The training dataset has a total of 20,631 entries for each column, where the longest running engine has a maximum cycle number of 362. The testing dataset has a total of 100 entries, a smaller number than the training dataset, where the longest running engine has a maximum cycle of 303. This indicates that the evaluation dataset has a narrower range of operational cycles than the training dataset. It is also noticed that the minimum TTF in training dataset is 0, which means that all engines have been run to failure, whereas the minimum TTF in testing dataset is 7 which indicates the engines remaining operation cycle before reaching failure. The analysis suggests that performance assessment on the testing dataset could identify if the method used has overfit to training and help provide insight to generalizability of the method.

From a preliminary overview of the dataset, we identify that the 'id' column has no intrinsic predictive value or quantitative relationship with the sensor data or the cycles. Excluding non-informative variables like 'id' simplifies the interpretation of the model as it allows for clearer understanding of how input variables (sensor readings and cycles) are influencing output predictions or classifications.

2.2 Distribution Analysis

To analyse the distribution of readings for each sensor in the dataset, Histogram plot combined with a Kernel Density Estimate (KDE) plot was generated using the Seaborn [2] library in Python. This method allows for a visual examination of the data's distribution and underlying patterns. The KDE is a non-parametric way to estimate the probability density function of a continuous variable which is overlaid on the frequency distribution (histogram).

The analysis from the Figure 1 reveals the following insights about the dataset:

- The distributions for the sensor readings appear to closely resemble a Gaussian distribution [3], also known as a normal distribution, which is characterized by a symmetric bell-shaped curve.
- Sensor 2 histogram shows a left skew: most data is high-valued with some low-value outliers.
- Sensor 3 and 4 histograms have a right skew. Mainly low-valued data with high-value outliers.
- Overall, sensor readings are mostly gaussian but with slight skewness.

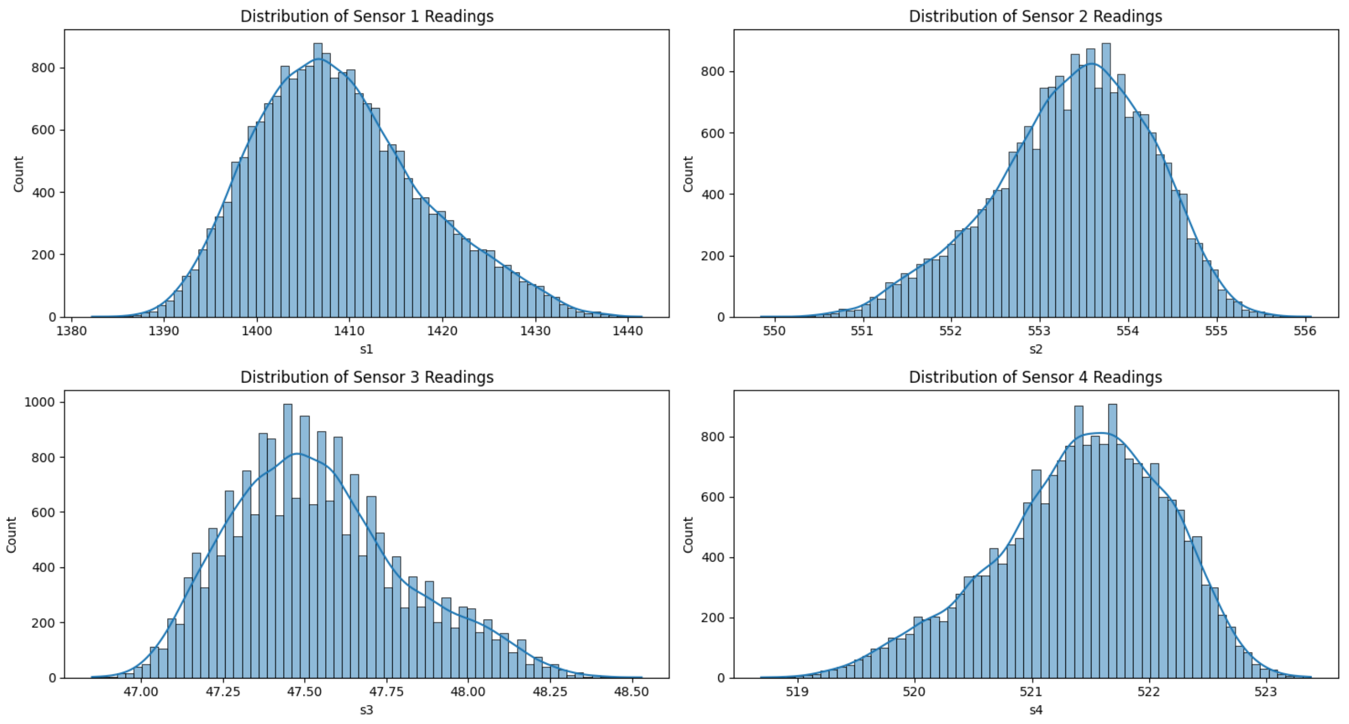

Figure 1: Histogram Plot with KDE plot for sensor data distribution analysis

The incorporation of 'cycle' as a variable alongside sensor data in regression and classification models is a methodically sound decision as it can contextualize the sensor data within the period of engine's operation, enabling a more comprehensive understanding of the data by the models. This temporal aspect is crucial for understanding the progression of the engine's condition over time. Using the Pandas library in python, I was able to quantitatively assess the skew and kurtosis values for the sensors and cycle columns. The Skew and Kurtosis provides insights into the shape of the distribution, and the presence and extremity of outliers in the dataset.

From Table 1 the following can be understood:

- Cycle feature: Slight positive skew indicates more engines with fewer cycles.
- Sensors 1 and 3: Positive skew, showing more engines with lower readings.
- Sensor 2 and 4: Negative skew, indicating more engines with higher readings.
- Cycle and Sensors: Negative kurtosis, fewer and less extreme outliers than normal distribution.

Table 1: Skew and Kurtosis Values for Sensors and Cycle Features

Feature Column	Skew Value	Kurtosis Value
Cycle	0.499904	-0.218539
S1	0.443194	-0.163681
S2	-0.394329	-0.157949
S3	0.469329	-0.172192
S4	-0.442407	-0.144917

2.3 Normality Analysis

By using the Shapiro-Wilk test and the D'Agostino's K-squared tests in this study a quantitative assessment of the normality of the data can be understood, which is a fundamental assumption that in many statistical modelling and inference techniques [3]. Both the tests provide insight into the specific nature of the distribution's deviation from normality, which is essential for the validity of parametric statistical tests and models that assume normality of the input data.

Table 2: Shapiro-Wilk test and D'Agostino K-Squared test performed on cycle and sensors for quantitative assessment of normality.

Feature Column	Shapiro Statistic	Shapiro P-Value	D'Agostino Statistic	D'Agostino P-Value
Cycle	0.9843	4.402e-42	645.508	6.755e-141
S1	0.9879	5.531e-38	523.130	2.533e-114
S2	0.9815	9.809e-45	717.064	1.956e-156
S3	0.9846	1.095e-41	637.143	4.425e-139
S4	0.9687	0.0	821.629	3.849e-179

Using the Scipy [4] library in Python, I was able to get the values from Table 2. The Shapiro-Wilk test gives the Shapiro statistic and the Shapiro p-value. The Shapiro-statistic, being close to 1 for all variables, suggests potential normal distribution, but their normality is disproven by Shapiro p-values below 0.05, particularly 'cycle' with a value of 0, leading to rejection of the normality hypothesis.

The D'Agostino's K-mean squared test performed using the Scipy library from python, gives a D'Agostino statistic and D'Agostino p-value. The test statistic assesses the normality of the data, a larger value indicates a deviation from normality. From Table 2 all the variables have large values, suggesting non-normality. D'Agostino p-values below 0.05 indicate non-normality for all variables, leading to rejection of normality. Despite attempts to normalize data using 'log' and 'sqrt' transformations, both Shapiro and D'Agostino tests confirm persistent non-normality in the distributions.

2.4 Corelation Analysis

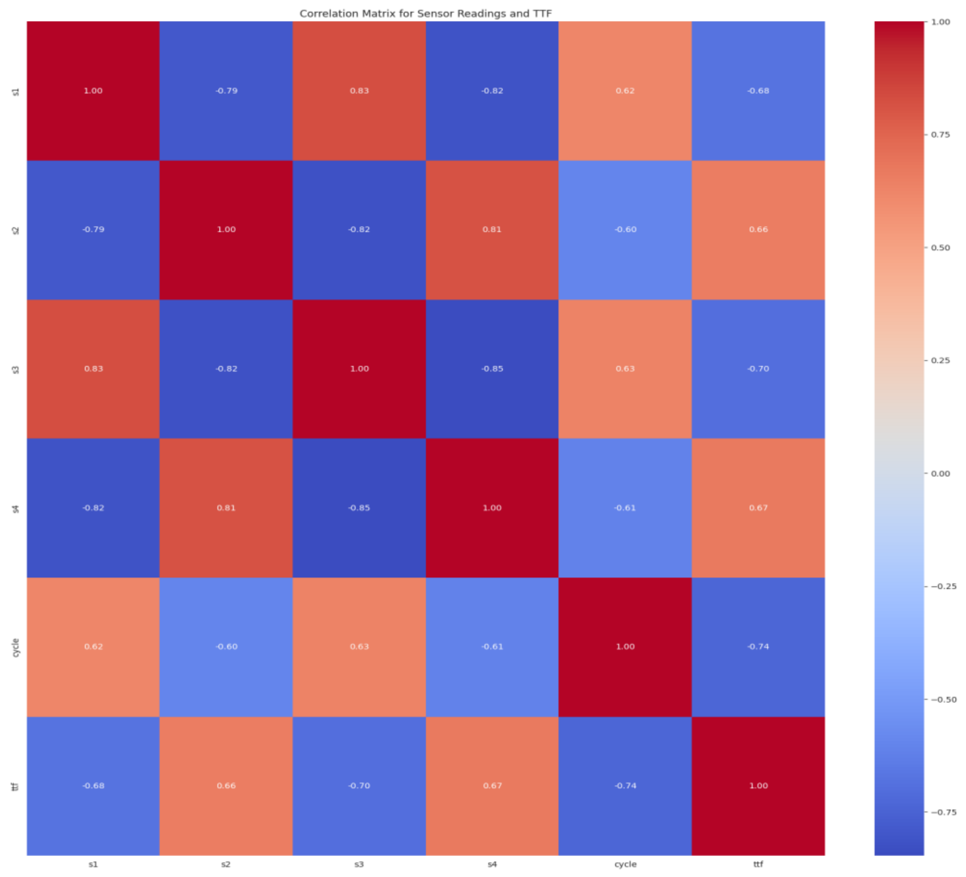

Figure 2: Corelation matrix of sensor readings and time-to-failure

An analysis into the corelation of the features with time-to-failure using the Pandas and Seaborn libraries from Python, Figure 2, revealed:

- Sensor 1 and 3 readings are strongly correlated, as are Sensor 2 and 4.
- Cycle has a positive correlation with Sensors 1 and 3, and a negative correlation with Sensors 2 and 4, reflecting engine wear and compensatory dynamics.
- Analysing sensor correlations with time-to-failure confirms insights on wear and tear.

2.5 Outlier Analysis

An outlier analysis on the sensor readings versus the cycle of operation of the engines, Figure 3, using Z-Scoring [5] to identify the outliers with a threshold set at 3 (which is the limit based on standard deviation) and then creating plots to identify normal readings and outliers in a scatter plot revealed the following:

- The scatter plots display the relationship between 'cycle' and each sensor's readings. The trend of the data points provides visual evidence of the corelations stated earlier: positive corelation with sensor 1 and sensor 3, negative corelation with sensor 2 and sensor 4 (an upward trend signifies a positive corelation, and a downward trend suggests a negative corelation).
- The red dots labelled as outliers indicate significant difference in the sensor reading from typical pattern. This could be indicative of abnormal engine behaviour and hence crucial for predictive maintenance.

This outlier analysis suggests not to exclude outliers in the sensor readings from the dataset during training the models, as they are crucial indicators of the abnormality in engine behaviour.

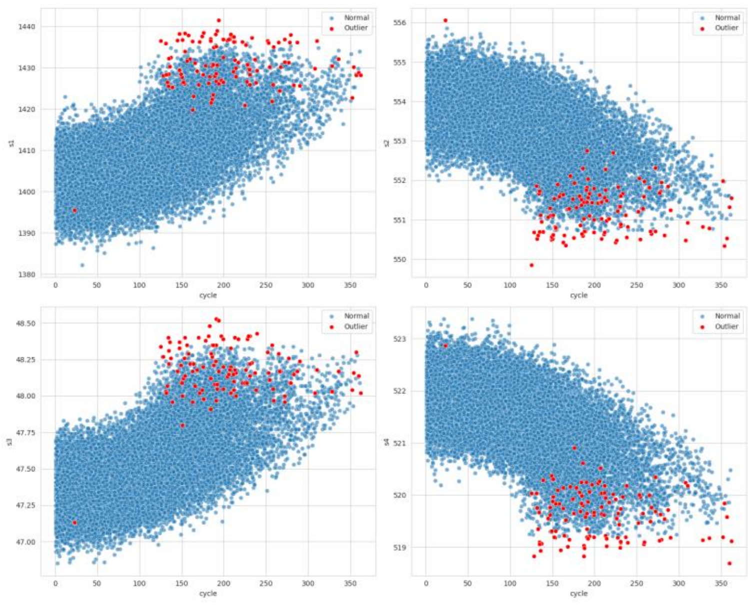

Figure 3: Outlier analysis for sensor readings and cycle of engine operation

3. Predictive Modelling

3.1 Feature Preparation

The analysis indicated skewness in the data, suggesting a need for normalization techniques. This study conducted experiments using various preprocessing methods:

- No Pre-processing: The data was used as is, without any scaling.
- Standard Scaler: Utilizing scikit-learn [6], this method standardizes features by removing the mean and scaling to unit variance, thus transforming each feature to have a mean of zero and a standard deviation of one. This technique is beneficial when dataset features are measured in different units, which might otherwise bias or skew the performance of machine learning algorithms.
- Min-Max-Scaler: Also from scikit-learn, this scaler normalizes features to a specified range, typically between 0 and 1. It adjusts each feature to fall within its minimum and maximum values, preserving the shape of the original distribution and the embedded information in the data. However, it is sensitive to outliers as extreme values compresses most data into narrow range.

These experiments were designed to analyse and compare he effectiveness of scaling approaches in enhancing model performance [7].

3.2 Model Selection

Data analysis of engine sensor measurements and operational cycles indicates excluding non-predictive ‘id’ feature for clarity. Selected features for modelling include all sensor readings and cycle counts. Skewness in sensor output suggests normalization is necessary for model adherence. Positive and negative correlations between cycle counts and sensors highlight the importance of thoughtful feature selection, especially for time-to-failure predictions. Outliers are valuable for anomaly detection. Model selection should prioritize techniques suitable for non-normal distributions and capable of interpreting complex patterns for diverse engine operations. Suitable models include non-parametric ones like decision trees, random forests, gradient boosting machines, and SVMs [8].

For this study the following models were selected for modelling regression and classification:
XGBoost

- XGBoost [9], or Extreme Gradient Boosting, is a sophisticated machine learning algorithm that leverages gradient boosting frameworks and improves model performance by applying regularization to control over-fitting, making it more focussed on the most salient features.
- The presence of notable outliers and slight skewness to the data is particularly suitable for XGBoost, its ability to sequentially build trees that learn from the errors of the previous ones allows for nuanced pattern detection in sensor data and operational cycles.
- Used the xgboost library in python.

Support Vector Machine (SVM) with RBF Kernel

- SVM with a radial basis function kernel enables to map input features to higher-dimensional spaces non-linearly, where it can find an optimal hyperplane that best separates the data into classes or fits a regression function.
- An SVM with RBF kernel is advantageous for this data due to its capacity to handle non-linearity effectively. It can construct a decision boundary that accurately captures the complex patterns in the data, essential for robust and reliable predictions.
- Used the scikit-learn library in python.

3.3 Evaluation Metrics

In the regression analysis for time-to-failure prediction, Mean Squared Error (MSE) and R2 Score are used for comprehensive error evaluation. MSE is key as it heavily penalizes larger errors, important given the serious impact of significant time-to-failure prediction deviations. R2 score measures how well the model captures variance in predictions, using sensor and cycle data; a higher R2 indicates better model fit. Together, MSE and R2 score assess error magnitude effectiveness in explaining time-to-failure variance [10].

For classification analysis in fault detection, I employ various metrics, each significant for different performance aspects.:

- Precision: Proportion true positives in positive predictions, vital for minimizing false positives.
- Recall: Proportion of actual positives identified, crucial in high-skates fault detection.
- Accuracy: Overall model correctness, often misleading due to dataset imbalance.
- F1 Score: harmonic mean of precision and recall, informative for understanding false positives and negatives in fault classification.
- Confusion Matrix: Detailed representation of true and false positives/ negatives, essential for specific model improvements.
- ROC and AUC: True positive rate vs false positive rate, with AUC measuring class discrimination capability.

These metrics collectively enhance understanding of model performance, with an emphasis on improving F1 score to reduce false positives and negatives [11].

3.4 Hyperparameter Tuning

Hyperparameter tuning is essential in developing regression and classification models, enhancing convergence speed and reducing computational demand. It addresses overfitting and underfitting by adjusting factors like regularization strength and tree depth, ensuring models align with dataset specificities. This report will discuss the best outcomes from hyperparameter tuning, comparing two predictive modelling techniques for aircraft engine maintenance.

Utilizing the Optuna [12] library in python, parameter grids were established for regression and classification models. Optuna employs Bayesian optimization for efficient hyperparameter space navigation, often yielding superior results with fewer trials compared to traditional GridSearchCV methods. An automated mechanism evaluated the impact of different scaling approaches and feature combinations on model performance, with the results and discussions presented in subsequent report sections.

4. Discussion and Results

4.1 Regression Analysis

Table 3 is the tabulation of the evaluation results for prediction of time-to-failure of the engines on the testing dataset using regression models: Support Vector Machine and Extreme Gradient Boosting Trees algorithms. The training of the models is carried out in an automated search using custom feature and normalization method search and using ‘optuna’ library for finding best parameters for the models using 100 trials for the hyperparameter search and a mean absolute error for the optuna objective function to compare and find the best parameter grid for the models. Normalization methods were only applied on the SVM model due to its sensitivity to feature scales that can have an affect on the model’s performance, especially in kernel methods and regularization. The XGBoost tree-based approach are robust to feature scales reducing the necessity for scaling as a preprocessing step.

Experiments were designed to use different combinations of the features from the dataset to evaluate how well the model can learn the time-to-failure patterns of the engine. From the data analysis, I found that the S1 and S3 sensors are highly corelated to the cycle of operation of the engine, and hence the wear and tear of the engines can be understood from the readings of these sensors.

The SVM model using a Min-Max Scaler outperformed the XGBoost Models for experiments using all sensor readings and cycle, and for the experiments using only S1 and S3 sensor readings with cycle. This is expected as the superior performance of the SVM model can be attributed to its effective handling of feature scale, robustness to noise and outliers, ability to capture non-linear patterns in smaller feature sets, and inherent regularization preventing overfitting.

Table 3: Evaluation of the regression models for prediction of time-to-failure using ‘optuna’ hyperparameter tuning on ‘SVR’ and ‘XGBRegressor’

Model	Features Used	Normalization Used	R2 Score	MSE
SVR	cycle_s1_s2_s3_s4	min-max	0.618	659.057
SVR	cycle_s1_s2_s3_s4	standard	0.406	1024.360
XGB-Regressor	cycle_s1_s2_s3_s4	none	0.579	726.42
SVR	cycle_s1_s3	min-max	0.649	605.14
SVR	cycle_s1_s3	standard	0.422	997.685
XGB-Regressor	cycle_s1_s3	none	0.608	675.242

The plots of predicted time-to-failure vs the ground truth time-to-failure for the 2 best experiments of SVM, can help us understand how well the model has generalized and is able to predict time-to-failure on the evaluation set.

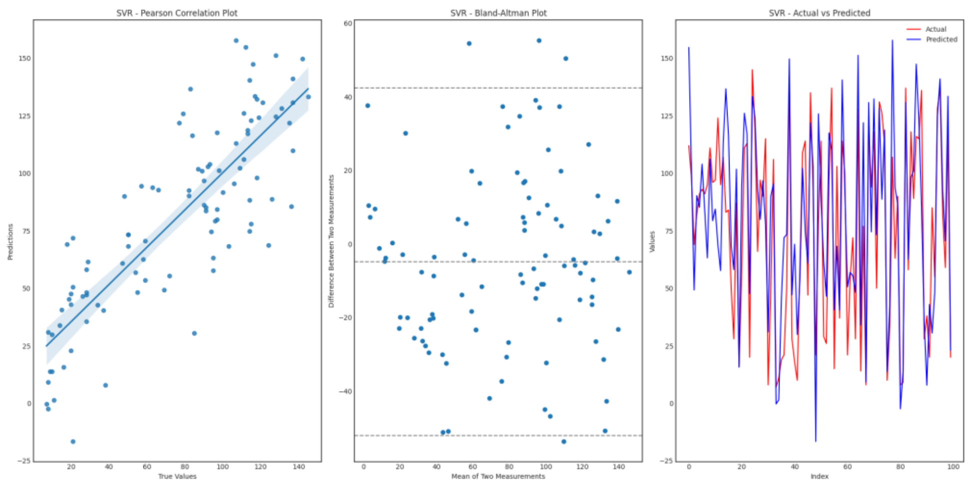

Figure 4: Evaluation plot of SVM model using only S1 and S3 sensors with cycle to predict for time-to-failure.

From the Figure 4 and Figure 5 the predicted time-to-failure readings have been plotted against the ground truth time-to-failure readings. The Pearson-Corelation plot, seen on the left in the figures, shows how well the predicted and ground truth readings are close to each other. The Bland-Altman plot, seen on the middle of the figures, gives a visual representation of the differences between the predicted and ground truth time-to-failure readings. The predicted vs actual plot is a line plot representation showing the red as the actual reading and blue as the predicted readings for time-to-failure.

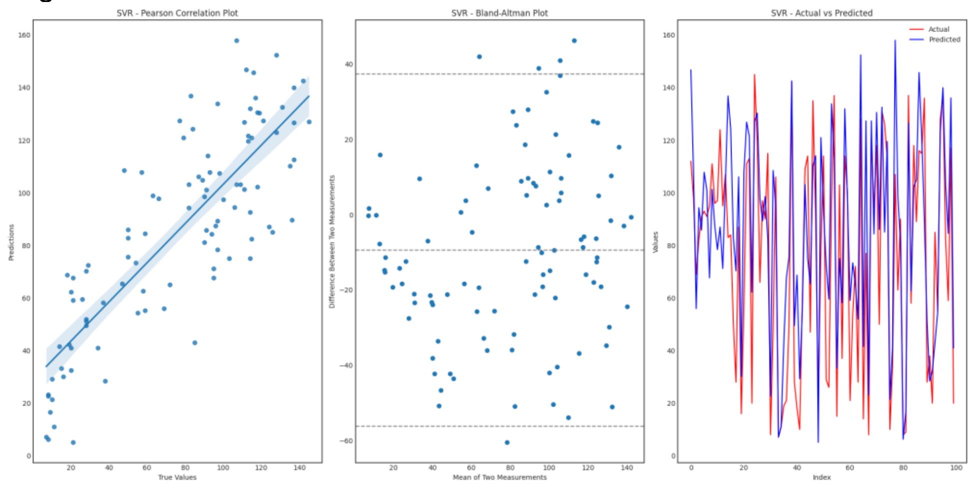

Figure 5:Evaluation plot of SVM Model using all Sensor readings with cycle to predict for time-to-failure.

From Figure 4, the corelation plot analysis shows a statistically significant positive corelation in the prediction of TTF, suggesting the model can capture the general trend in the data. However, the variability in the Bland-Altman plot and the divergence observed in the Actual vs Predicted plot highlight that the model's performance has room for improvement, particularly in terms of consistency and precision across the full range of predictions. The model seems to perform well for a portion of the data, but there are noticeable instances where the model's predictions deviate from the actual TTF. Comparing with the plots from the Figure 5 for the model using all sensor readings with cycle for predicting TTF, the SVM model using only S1 and S3 sensor readings with cycle has a better corelation and the readings of actual vs predicted are more closer in the plot.

Optuna was used for hyperparameter tuning of the SVM model, with the following parameter grid:

- Regularization parameter ‘C’: Logarithmic scale of 0.1 to 10, balancing smooth decision boundaries and accurate training point classification.
- Kernel Coefficient ‘gamma’: Logarithmic scale of 0.01 to 1, assessing non-linear relationship handling.
- Epsilon: Logarithmic scale of 1e-5 to 1e-1, determining error margin tolerance and fitting precision.
- Kernel type: Radial Basis Function (RBF) to adapt to data complexity and structure.

Optimal parameters found:

- SVM (S1, S3, Cycle): ‘C’=0.221161, ‘gamma’=0.122645, ‘epsilon’=0.005714.
- SVM (S1, S2, S3, S4, Cycle): ‘C’=0.111048, ‘gamma’=0.133073, ‘epsilon’=0.020128.

4.2 Classification Analysis

Table 4: Evaluation of fault classification for SVC and XGB-Classifier on testing dataset

Model	Features Used	Normalization Used	Accuracy	F1 Score	Precision	Recall
XGB-Classifier	cycle_s1_s2_s3_s4	none	0.92	0.826	0.904	0.76
SVC	cycle_s1_s3	min-max	0.91	0.830	0.785	0.88
XGB-Classifier	cycle_s1_s3	none	0.89	0.755	0.85	0.68
SVC	cycle_s1_s2_s3_s4	min-max	0.89	0.792	0.75	0.84

For fault classification, the binary target ‘label_bnc’ in the testing dataset was compared with predictions from SVM and XGBoost models. These models’ performance metrics are detailed in Table 4. Optuna was employed for hyperparameter tuning, using 100 trials and F1 score as the assessment metric.

The F1 score analysis provides insights into the models’ handling of false positives and false negatives. The SVC model, using S1, S3 and Cycle, achieved a high F1 score of 0.830 but a lower precision of 0.785, indicating a tendency towards false positives. In contrast, the XGB-Classifier, utilizing all sensors and cycle, showed a robust F1 score of 0.826 and higher precision of 0.904, indicating better management of false positives.

Further analysis revealed that the SVC model had higher precision and F1 scores for healthy engine status prediction, but lower precision for faulty status, suggesting a higher false positive rate. The model’s recall of 0.88 implies a lower likelihood of misclassifying faulty engines as healthy.

The analysis of the XGB-Classifier model on a per-class analysis reveals that the model has a precision of 0.92, a f1 score of 0.94 and a recall of 0.97, for predicting the healthy status of the engine, whereas a precision of 0.90, a f1 score of 0.82 and a recall of 0.76, for predicting the faulty status of the engines. This indicates that the model has a lower rate of predicting False positives, which means the model is less likely to predict healthy engines as faulty, however a few false negatives suggest that a few of the faulty engines will be classified as healthy engines which is a risk to safety as it could result in a missed opportunity for prevention of failure.

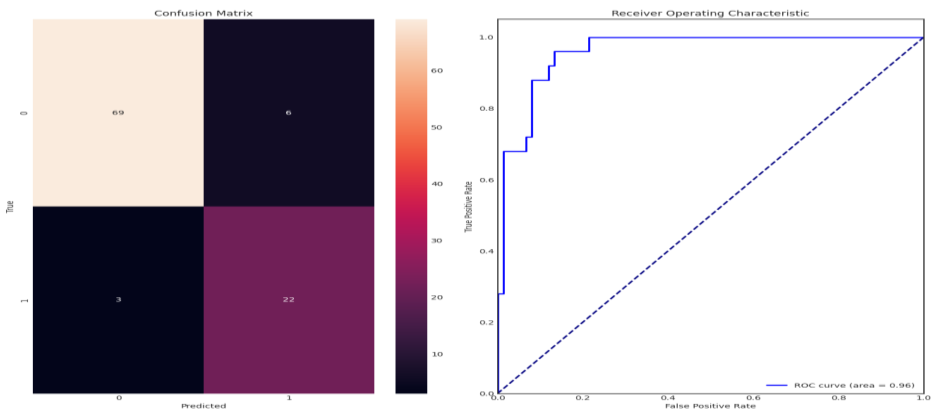

Figure 6: Evaluation plot of Confusion matrix and ROC AUC curve for the SVC model using only the S1 and S3 sensors with cycle for faulty classification.

Figure 6 shows the confusion matrix plot and the ROC-AUC curve for the SVC model, and Figure 7 shows the confusion matrix plot and ROC-AUC curve for the XGB-Classifier model. From the confusion matrix it is seen that the SVC model predicts fewer False Negatives compared to the XGB-Classifier model. However, we notice the XGB-Classifier model predicts fewer False Positives compared to the model.

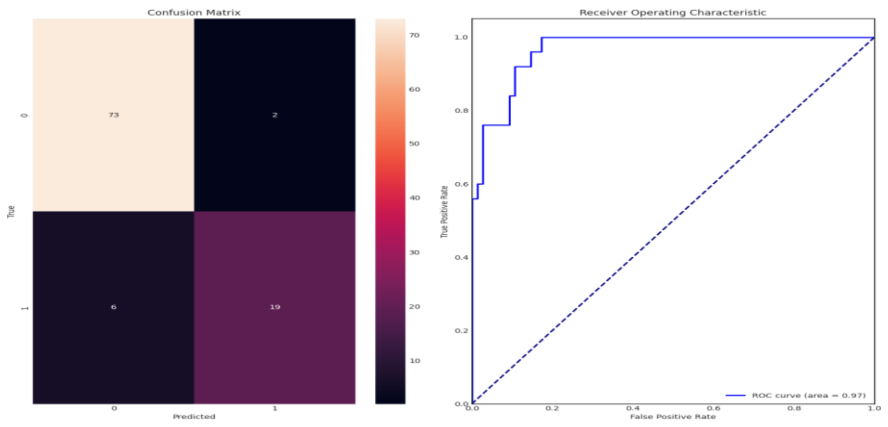

Figure 7: Evaluation plot of Confusion matrix and ROC-AUC curve for XGB-Classifier model using all sensor readings and cycle for faulty classification.

Further analysis of the ROC-AUC curve suggests that both the models perform well, the XGB-Classifier model rises more steeply towards the top-left corner of the plot compared to the plot for the SVC model, this suggests that the XGB-Classifier model achieves a higher true positive rate with lower false positive rate for more threshold values, which is desirable. Both models have a strong discriminative ability for both the classes, which is essential for real-world applications where the ability to correctly identify true positives and true negatives is crucial while minimizing false positives and false negatives.

To find the best parameters for the two models I used optuna library with a parameter grid of:

Support Vector Classifier

- Regularization parameter ‘C’: Logarithmic scale from 0.1 to 10 for smooth decision boundary and accurate classification balance.
- Kernel Coefficient ‘gamma’: Logarithmic scale from 0.01 to 1 to evaluate non-linear relationship handling.
- Kernel type: Categorical options of ‘linear’, ‘poly’, ‘rbf’, and ‘sigmoid’.

XGBoost Classifier

- Number of trees: Range of 10 to 1000 for exploring model complexity and balancing fitting.
- Shrinkage factor ‘learning_rate’: Range from 1e-10 to 1 to control learning step size and generalization.
- Maximum tree depth: Range of 2 to 32 for managing model complexity and overfitting risks.
- Subsample ratio of training instances: Range of 0.5 to 1.0 for sample use per tree, balancing overfitting and underfitting.
- Subsample ratio for columns per tree: Range of 0.5 to 1.0 for feature use per tree, enhancing robustness.

Optimal Hyperparameters:

- SVC (S1, S3, Cycle): ‘C’=2.247, ‘gamma’=0.125, Kernel=’poly’.
- XGB-Classifier (S1, S2, S3, S4, Cycle): ‘n_estimators’=283, ‘max_depth’=20, ‘learning_rate’=0.944, ‘subsample’=0.845, ‘colsample_bytree’=0.845.

4.3 Comparative Analysis

Based on the results from classification and regression analysis using support vector machine and gradient boosting algorithms, we find that the classification models are more suited for the practical application of predicting time-to-failure by fault classification. The results from fault classification show that the model can strongly distinguish both the classes while minimizing false positives and false negatives. However, the model from regression analysis is not suited for the practical application due to its higher error value from ground truth data, making it more likely to have many false positives and false negatives.

This study set out to compare the two techniques that can be used for predictive maintenance of aircraft engines in a practical setting, from the results and analysis it can be concluded that using a classification technique outperforms the regression. Conversely, regression models, although conceptually appropriate for predicting the precise time-to-failure, exhibit limitations as seen in the results, where the predictions have greater deviations from the ground truth data, implying a higher propensity for prediction errors. Such errors could manifest as incorrect estimations of an engines remaining useful life (RUL), resulting in potentially costly and unsafe misjudgements regarding maintenance scheduling.

Classification models not only provide a binary indicator of the engine health, but also exhibit a higher predictive accuracy that is crucial in operational settings. Therefore, the deployment of classification techniques is recommended for organizations seeking to implement a predictive maintenance strategy that is both dependable and operationally viable.

5. Conclusion

In the conclusions of this study, the data analysis played a crucial role in guiding the model selection, which in turn influenced the observed performance outcomes. The meticulous data analysis revealed inherent characteristics of the dataset, such as the skewness and distribution patterns of sensor readings and their correlations with the engine cycles. These insights directly informed the choice of predictive models, leading to the selection of those equipped to handle the data's specific nuances. The models that demonstrated the best performance were those that could accommodate the temporal aspect encapsulated by the 'cycle' variable, manage the non-linear relationships evident from the corelation analysis, and robustly incorporate the information conveyed by outliers. Such models, including Support Vector Machines and Extreme Gradient Boosting algorithms, were adept at capturing the complex dynamics of engine wear and degradation, as evidenced by their string predictive accuracy. This alignment of data-driven model selection with high model performance in fault classification underlines the effectiveness of the methodical approach to predictive maintenance in the aviation industry. The classification models were validated as not only theoretically sound but practically superior for the early detection of engine faults, thereby reinforcing their value in operational settings. Improved model performance could be achieved in regression by using more complicated algorithms like LSTMs or Deep Neural Nets which can learn non-linear relationships better, also using strategies like rolling-window approach and robust normalization strategies could achieve better results.

6. Supplementary Material

The Code and Data used for this study has been uploaded and will be maintained in the below attached GitHub repository.

GitHub repo: https://github.com/juliangdz/SLM_Aircraft_Predictive_Maintenance.git

7. References

[1] A. A. N. Q. Z. O. K. Zeki Murat Çınar, "Machine Learning in Predictive Maintenance towards sustainable smart manufacturing in Industry 4.0," *sustainability,* vol. 8211, p. 12, 2020.

[2] M. L. Waskom, "seaborn: statistical data visualization," *The Journal of Open Source Software,* vol. 6, no. 60, p. 3021, 2021.

[3] A. G. a. S. Zahediasl, "Normality Tests for Statistical Analysis: A Guide for Non-Statisticians," *International Journal of Endocrinology and Metabolism,* vol. 10, no. 2, pp. 486-489, 2012.

[4] R. G. T. E. O. M. H. T. R. D. C. E. B. P. P. W. W. J. B. S. J. v. d. W. M. B. J. W. Pauli Virtanen, "SciPy 1.0: fundamental algorithms for scientific computing in Python," *Nature Methods,* vol. 17, pp. 261-272, 2020.

[5] S. Seo, "A Review and Comparison of Methods for Detecting Outliers," University of Pittsburgh, 2006.

[6] F. a. V. G. a. G. A. a. M. V. Pedregosa, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, 2011.

[7] M. A. P. M. Md Manjurul Ahsan, "Effect of Data Scaling Methods on Machine Learning Algorithms and Model Performance," *technologies,* vol. 52, p. 9, 2021.

[8] S. Raschka, "Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning," 2020.

[9] C. G. Tianqi Chen, "XGBoost: A Scalable Tree Boosting System," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* pp. 785-794, 2016.

[10] W. M. J. G. Chicco D, "The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE, and RMSE in regression analysis evaluation," *PeerJ Computer Science,* vol. 7, 2021.

[11] G. C. O. Varoquaux, "Evaluating Machine Learning Models and Their Diagnostic Value," in *Machine Learning for Brain Disorders*, New York, NY, Humana, 2023.

[12] T. a. S. S. a. Y. T. a. O. T. a. K. M. Akiba, "Optuna: A Next-Generation Hyperparameter Optimization Framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Anchorage, AK, USA, Association for Computing Machinery, 2019, p. 2623–2631.