Applied Machine Learning Project: TurboFan Engine Remaining Useful Life (RUL)

Roberto Cialini, Jaka Škerjanc, Riccardo Rossi-Erba, Jan Fiala, Chisom Okoye

Abstract

With technological advancements, turbofan engines, essential components of aircraft, have become more complex. The complexity of turbofan engines has increased significantly with technological advancements, making effective maintenance strategies crucial. About 60% of aircraft breakdowns involve turbofan engines, influenced by varying operating hours and conditions [1]. Traditional maintenance methods, such as corrective and preventive maintenance, are inadequate for ensuring optimal performance and reliability. This study explores Prognostics and Health Management (PHM) techniques, focusing on the prediction of the Remaining Useful Life (RUL) of turbofan engines using data-driven approaches.

The paper shows a variety of machine learning models, including Linear Regression, Elastic Net Regression, Random Forests Regression, Support Vector Machines Regression, and Neural Networks, to analyze historical sensor data and predict engine degradation. The results indicate that ensemble methods like Random Forests and advanced models such as Support Vector Machine Regression with RBF kernel provide superior performance compared to simpler models, demonstrating higher accuracy and robustness in RUL prediction.

The findings suggest that these data-driven models can significantly enhance the reliability of maintenance strategies for turbofan engines.

1. Introduction

Turbofan engines are the backbone of modern aviation, powering the majority of commercial and military aircraft. However, these complex machines are susceptible to degradation throughout their operational lifespan, leading to significant safety concerns and increased maintenance costs. Unscheduled engine failures pose serious threats to passenger and crew safety, with the Federal Aviation Administration (FAA) reporting that engine-related issues account for approximately 25% of all in-flight emergencies [2]. Reactive maintenance approaches based on fixed service intervals can be inefficient and costly, with the average cost of an unscheduled engine removal exceeding $1.5 million [3].

Traditional maintenance strategies like corrective and preventive maintenance have proven insufficient. Prognostics and Health Management (PHM), also known as condition-based or predictive maintenance, addresses these challenges by ensuring optimal system performance, preventing failures, reducing maintenance costs, and managing system health. PHM is critical in aviation, where reliability is paramount due to the severe consequences of accidents. This paper focuses on turbofan engines, the most complex and critical aircraft components, requiring high reliability and quality. Predicting Remaining Useful Life (RUL) is essential for developing cost-effective maintenance strategies and enhancing reliability.

More in detail, this paper explores the application of Machine Learning (ML) techniques for predicting the RUL of turbofan engines. By analyzing sensor data collected throughout the engine's lifecycle, including

Email addresses: rocia24@student.sdu.dk (Roberto Cialini), jaske24@student.sdu.dk (Jaka Škerjanc), riros24@student.sdu.dk (Riccardo Rossi-Erba), jafia24@student.sdu.dk (Jan Fiala), choko23@student.sdu.dk (Chisom Okoye)

parameters such as vibration, temperature, fuel consumption, and engine speed, ML models can identify patterns and trends indicative of impending degradation. This predictive capability enables proactive maintenance strategies, allowing for interventions to be scheduled before critical failures occur.

The benefits of implementing an RUL prediction system are multifaceted. Early detection of degradation facilitates targeted maintenance actions, minimizing the risk of catastrophic failures and enhancing overall operational safety. According to the International Air Transport Association (IATA), predictive maintenance can reduce unscheduled maintenance events by up to 30% and decrease maintenance costs by 20% [4]. This paper delves into the development and evaluation of various ML algorithms for RUL prediction, using real-world engine data sets from aircraft manufacturers and engine test rigs.

RUL prediction methods are categorized into model-based, data-driven, and hybrid approaches. Model-based methods use mathematical models to describe system degradation but are costly and require expertise. Data-driven methods analyze historical sensor data to understand degradation with less expertise needed. Hybrid approaches combine both methods but have their limitations.

In our project, we utilized data-driven models to predict the RUL of turbofan engines. The models implemented include Linear Regression, Elastic Net Regression, Random Forests Regression, Support Vector Machines Regression, and Neural Networks. These models leverage historical sensor data to accurately predict engine degradation and remaining useful life. By employing these data-driven approaches, we aim to enhance prediction accuracy, reduce maintenance costs, and improve the overall reliability of turbofan engines.

The remainder of this paper is organized as follows. Section 2 reviews related work addressing the same problem. Section 3 outlines the algorithms and methods used in this study. The results are presented in Section 4. The paper concludes with a summary in Section 5, followed by a discussion of contributions in Section 6.

2. Related Work

Predicting the remaining useful life (RUL) of turbofan jet engines has been extensively studied, leveraging various machine learning algorithms and datasets. Ali Alhamaly (2019)[5] explored a data-driven approach for predictive maintenance using the NASA C-MAPSS dataset, which simulates the degradation of commercial turbofan engines under various operational conditions. Alhamaly focused on preprocessing steps such as dimensionality reduction through PCA and the development of a health index to fuse sensor data, utilizing linear regression for trend analysis and exponential models for RUL prediction.

Similarly, Thakkar and Chaoui (2022)[6] employed deep learning models, specifically Deep Layer Recurrent Neural Networks (DL-RNN), to enhance RUL prediction accuracy. Their study compared DL-RNN with other neural network architectures like Multilayer Perceptron (MLP) and Nonlinear Auto Regressive Network with Exogenous Inputs (NARX), demonstrating superior performance of DL-RNN on the C-MAPSS dataset with significantly lower error rates (MAE and RMSE) compared to the alternative models.

In another study, Liu et al. (2023)[7] applied various machine learning algorithms to the NASA turbofan engine dataset, including linear regression, K-nearest neighbors (KNN), elastic net, random forests, support vector machines (SVM), and regression trees. Their research aimed to evaluate the effectiveness of these simpler models in RUL prediction, highlighting that ensemble methods like random forests and more sophisticated algorithms such as SVM provided competitive results compared to deep learning models.

Solis-Martin, Galan-Paez, and Borrego-Diaz [8] even went further to develop an approach to determine the RUL of jet engines that has been considered innovative and ranked in the third place of the 2021 PHM Conference Data Challenge. The technique involves tow Deep Convolutional Neural Networks stacked on two levels. While the first is used to extract low-dimensional feature vector based on the normalized raw data as input, the second DCNN injects a list of vectors and estimates RUL for the jet engines and the results show excellent outcomes.

Most of the existing literature focuses on advanced neural network models for RUL prediction. However, in our work, we chose to stick to simpler machine learning algorithms. This decision was driven by a desire to explore the effectiveness of these more interpretable and less computationally intensive models, providing a baseline for comparison against more complex approaches.

By integrating insights from prior studies and our experimental evaluations, we aim to contribute a comparative analysis of traditional machine learning methods for RUL prediction, providing a benchmark for future research and practical implementations in the field of prognostics and health management of aircraft engines.

3. Proposed Method

First we conducted a thorough analysis and preprocessing of the dataset to prepare it for modeling in section 3.1. In the data pre-processing phase the initial step involves removing columns with constant values, as these do not provide useful information for predictive modeling. Next, the RUL values are capped to a specific constant value to limit the range and effectively manage outliers. Columns that do not show a significant correlation with the target variable, RUL, are dropped to reduce dimensionality and improve model performance. Additionally, sensor data is smoothed to reduce noise and enhance the signal quality, aiding in more accurate predictions.

Following data pre-processing, the cleaned data is used to train various machine learning models. The models include linear and elastic net regression, random forest regression, regression tree, support vector machine (SVM) regressor with both linear and RBF kernels. Additionally, a basic neural network model is used to capture complex patterns in the data. This is the core of the section 3.2

The trained models are evaluated using cross-validation to ensure their robustness and generalizability. Four metrics are used to measure the performance of the models [9]:

1. Mean Absolute Error (MAE): measures the average magnitude of errors in a set of predictions, without considering their direction.
2. Mean Square Error (MSE): quantifies the error by squaring the difference between actual and predicted values.
3. Root Mean Squared Error (RMSE): provides the square root of the average squared differences between actual and predicted values, giving a more interpretable error measure.
4. R^2 Score: represents the proportion of the variance for the dependent variable that's explained by the independent variables in the model.

Figure 1 gives an overview about the workflow.

3.1. Project Analysis and Preprocessing

3.1.1. Data description

The datasets consist of multiple multivariate time series. Each time series represents a different engine, simulating a fleet of engines of the same type. Each engine begins with varying degrees of initial wear and manufacturing variation, which are unknown to the user. These variations are considered normal and not indicative of faults. The data includes three operational settings that significantly affect engine performance, along with sensor noise contamination.

At the start of each time series, the engines operate normally, but a fault develops over time. In the training set, this fault grows until the system fails, while in the test set, the time series ends before system failure. The competition's objective is to predict the remaining operational cycles before failure in the test set, i.e., the number of cycles the engine will continue to operate after the last recorded cycle. A vector of true Remaining Useful Life (RUL) values for the test data is also provided.

The dataset is made by 26 features. Each row represents a snapshot of data taken during a single operational cycle, with each column representing a different variable. The columns correspond to the unit number, time in cycles, three operational settings, and twenty-one sensor measurements.

The dataset, labeled FD001, includes 100 train trajectories all under one condition at sea level and exhibiting one fault mode, specifically HPC (high-pressure compressor) degradation.

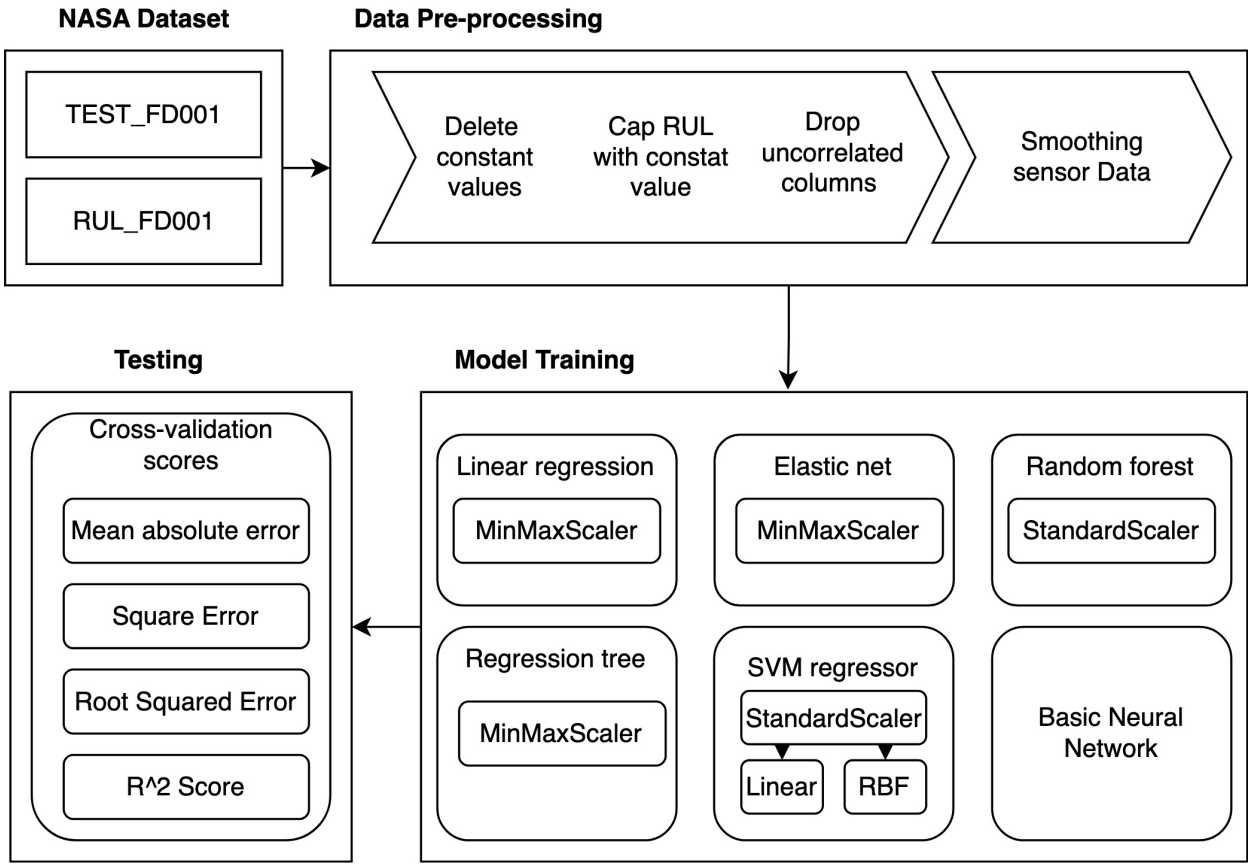

Figure 1: Project workflow.

3.1.2. Data analysis

Initially, we imported the necessary libraries and loaded the dataset. The column names were defined for better readability, which included operational settings and sensor measurements with added the RUL column, that was calculated based on the maximum cycle per engine and the current cycle.

We then provided a summary of the dataset to understand its basic statistics and computed the correlation matrix (figure 2) to identify relationships between different variables.

This step was crucial for feature selection, as it helped identify which features were most relevant to the RUL prediction.

To visualize the sensor data, we plotted the signals from the sensors for a subset of engines. This visualization helped us understand the patterns and trends within the sensor data, revealing significant insights into engine behavior over time.

3.1.3. Data cleaning

In the data cleaning phase, we identified and removed columns with constant values, as they do not contribute to the model's predictive power. Additionally, we removed columns that were not correlated with RUL, focusing only on the relevant features. This step ensured that the dataset was concise and contained only useful information for training the model.

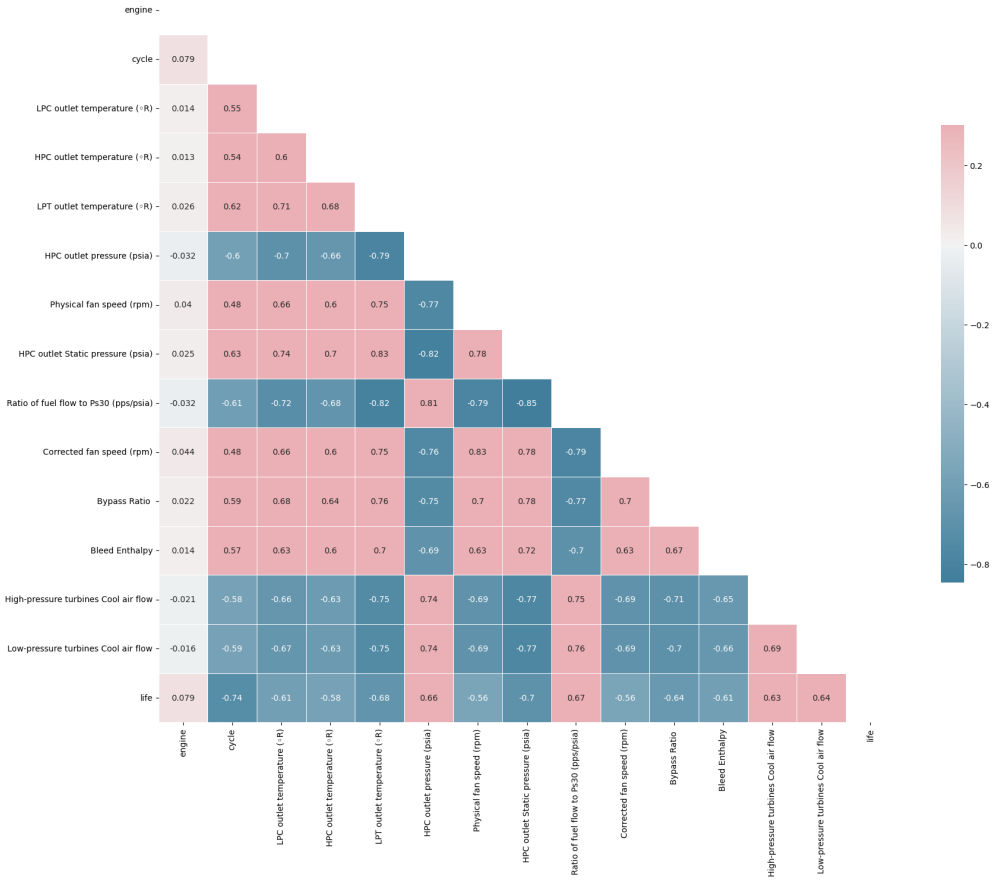

Figure 2: Correlation matrix.

3.1.4. Data smoothing

After analyzing noise in the data, we applied a smoothing technique using an exponentially weighted moving average. A comparison between smoothed and non-smoothed sensor data is shown in Figure 3. This technique was chosen to balance the trade-off between noise reduction and data lag. By smoothing the data, we improved the quality of the sensor readings, making them more reliable for predictive modeling. We visualized the smoothed data alongside the original data, which clearly showed the reduction in noise and highlighted the underlying trends more effectively. This visualization reaffirmed the importance of smoothing in enhancing the dataset's quality. By combining these preprocessing steps we prepared the dataset for effective modeling, ensuring that the features were relevant, the noise was minimized, and the data was ready for subsequent machine learning tasks.

3.1.5. Clipping Remaining Useful Life

Our initial assumption for the Remaining Useful Life (RUL) in the training set is that it declines linearly to zero. However, sensor signals often exhibit a 'bend,' indicating the onset of degradation. Given the lack of information on initial wear and tear, this linear decline might not accurately represent the true RUL behavior.

To better reflect the observed data, we assume that the RUL remains constant initially and only begins to decline linearly after a certain point. This assumption has two main advantages:

- Improved Correlation with Sensor Signals: By keeping the RUL constant initially, it aligns better with the stable mean of the sensor signals, providing a more realistic representation of the engine's life.

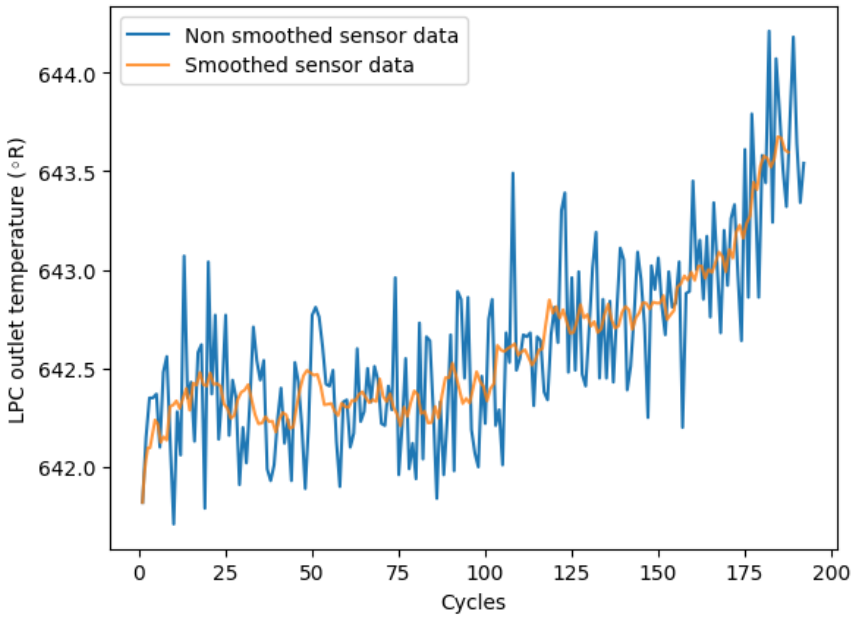

Figure 3: Comparison of smoothed and non-smoothed sensor data.

- Reduced Target Variable Spread: Lower peak RUL values result in a reduced spread of the target variable, making it easier to fit predictive models.

To implement this, we clip the RUL at a predefined value (e.g., 125 cycles), ensuring it remains constant up to this point before starting its linear decline. This clipping process is illustrated in Figure (figure 4), which shows how the RUL remains constant initially and then decreases linearly. This approach matches the patterns observed in the sensor data more closely, resulting in more accurate and reliable RUL predictions.

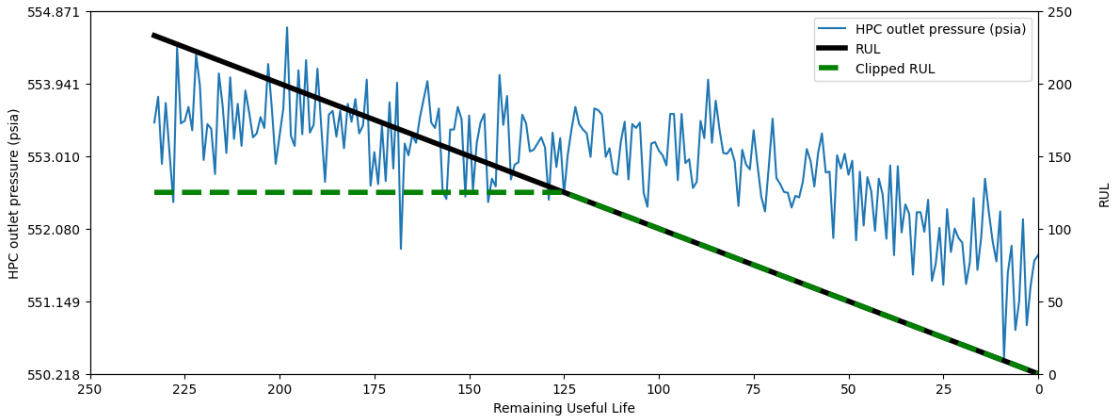

Figure 4: Clipping of RUL at value 125.

3.2. Machine Learning Methods

For every model we splitted the dataset into a two smaller dataset, the train dataset and the test dataset, respectively the 70% and 30 % of the original dataset.

3.2.1. Linear Regression

Linear Regression is a simple yet powerful method for predicting a dependent variable (y) based on one or more independent variables (x) [10]. At its core, linear regression aims to find the best-fitting straight line through a set of data points. This line, known as the regression line, represents the predicted values of the dependent variable based on the values of the independent variables.

The relationship is modeled using a linear equation:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \epsilon \tag{1}$$

Here, y is the dependent variable, x is the independent variable, β_0 is the y-intercept, β_i are the coefficients, and ϵ represents the error term, accounting for the variability in y that cannot be explained by the linear relationship with x. The coefficients (β_i) are estimated using the least squares method, which minimizes the sum of the squared differences between the observed and predicted values.

Our implementation

In our implementation, we created a pipeline that scaled the data with a MinMax Scaler before applying Linear Regression. The model was trained on the training data and evaluated using cross-validation (5 folds) on the four metrics described above. We visualized the predictions against the true values using a scatter plot (figure 5), which helped us see how accurately the model predicted the data.

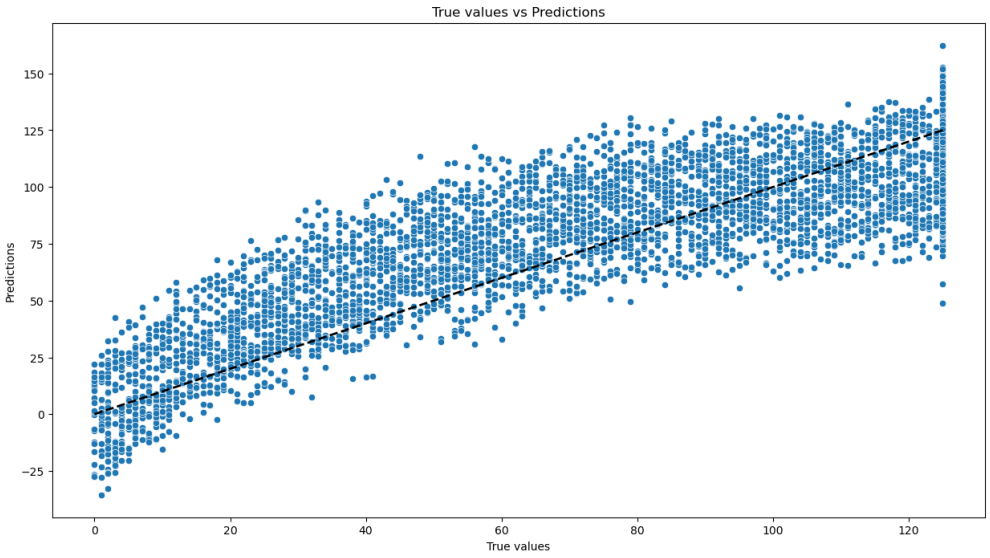

Figure 5: Comparison of linear model predictions.

3.2.2. Elastic Net

Elastic Net is a regularized regression method that linearly combines the L1 and L2 penalties of the Lasso and Ridge methods [11]. It addresses the limitations of both methods, providing a balance between feature selection and coefficient shrinkage. Elastic Net adds two types of regularization terms to the linear regression equation:

$$y = \beta_0 + \sum_{i=1}^{p} \beta_i x_i + \lambda_1 \sum_{i=1}^{p} |\beta_i| + \lambda_2 \sum_{i=1}^{p} \beta_i^2 + \epsilon \tag{2}$$

Here, y is the dependent variable, x_i are the independent variables, β_i are the coefficients, λ_1 and λ_2 are the regularization parameters, and ϵ is the error term. The regularization parameters are λ_1 and λ_2. λ_1 corresponds to the L1 norm (Lasso), promoting sparsity by encouraging some coefficients to be exactly zero. λ_2 corresponds to the L2 norm (Ridge), promoting smoothness by shrinking the coefficients towards zero, which helps handle multicollinearity and improve generalization[11][12]. The optimization problem of

Elastic Net regression involves finding the coefficients (β) that minimize both the residual sum of squares and regularization terms:

$$\min_{\beta_0,\beta} \frac{1}{2n} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} \beta_j^2 \tag{3}$$

In this equation, n represents the number of observations, y_i is the dependent variable for observation i, and x_{ij} is the value of the j-th independent variable for observation i. The intercept is β_0, and β_j are the coefficients.

Our implementation

In our practical implementation of the Elastic Net model, we combined L1 and L2 regularization to balance feature selection and coefficient shrinkage. We set up a pipeline with a StandardScaler and the Elastic Net model. Then we defined a grid of hyperparameters for Elastic Net and used GridSearchCV to find the best parameters. We visualized the evolution of the mean test scores against the hyperparameters to understand their impact (figure 6). We got a result of

```
{'elasticnet__alpha': 0.0031622776601683794,
'elasticnet__l1_ratio': 0.001}
```

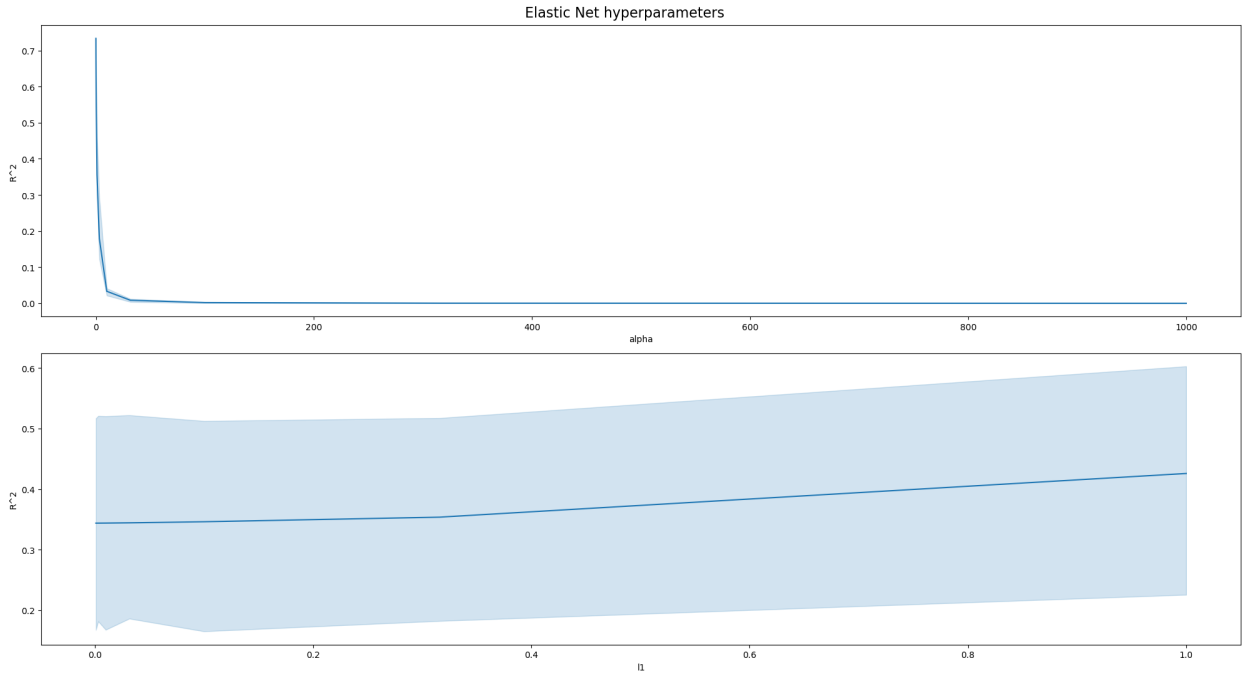

Figure 6: Comparison of elastic net parameters.

We used a cross validation (5 folds) to get the scores of the best model. In the end we visualized the predictions against the true values using a scatter plot (figure 7), as we did for the linear regression.

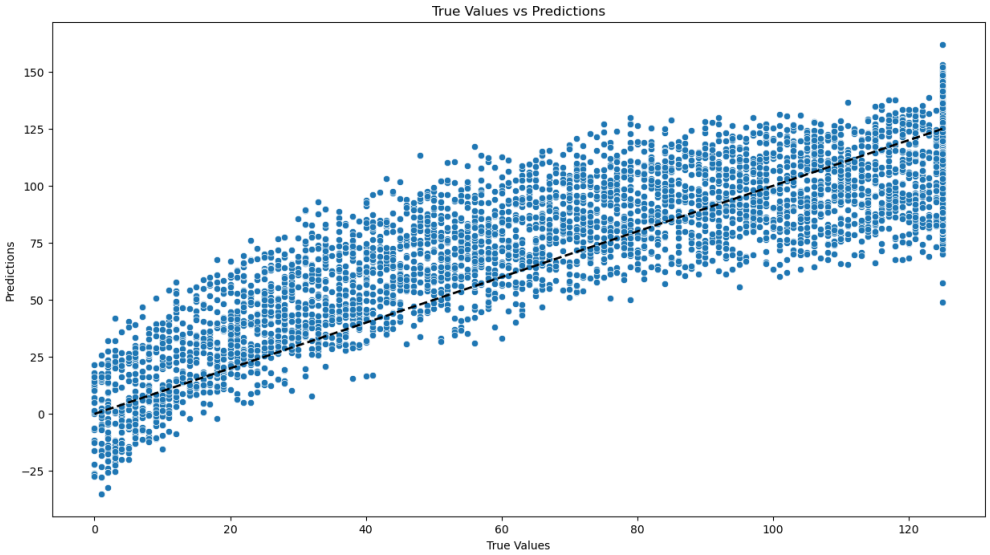

Figure 7: Comparison elastic net predictions.

3.2.3. Regression Trees

Regression trees are a type of decision tree specifically used for predicting continuous numerical outcomes. Regression Trees is a decision tree method used for predicting continuous values. Regression tree model is a non-parametric supervised learning method that recursively partitions the data space and fits a simple model (e.g., mean or median) within each partition, creating a tree-like model of decisions [13].

The Regression Tree algorithm involves the following steps:

1. Start with the entire dataset as the root.
2. At each node, split the dataset into two subsets based on the value of one of the independent variables. The split is chosen to minimize the mean squared error (MSE) within the resulting subsets.
3. Repeat the splitting process recursively for each subset until a stopping criterion is met (e.g., minimum number of samples per leaf or maximum tree depth).
4. Assign the mean value of the target variable within each terminal node (leaf) as the prediction for that region.

Our implementation

In our practical implementation of Regression Trees, we aimed to predict continuous values by recursively partitioning the data and fitting simple models within each partition. We set up a pipeline with a MinMaxScaler and a Decision Tree Regressor. We defined a grid of hyperparameters for the regression tree and used GridSearchCV to find the best parameters and visualized the evolution of mean test scores against various hyperparameters (figure 8) to understand their impact.

For the best hyperparameters we got a result of

```
{'regressor_tree__max_depth': 10,
'regressor_tree__max_features': 0.6,
'regressor_tree__min_samples_leaf': 20,
'regressor_tree__min_samples_split': 2}
```

Then, using a cross-validate (10 folds) on the four metrics described above, we calculated the scores on both train and test datasets.

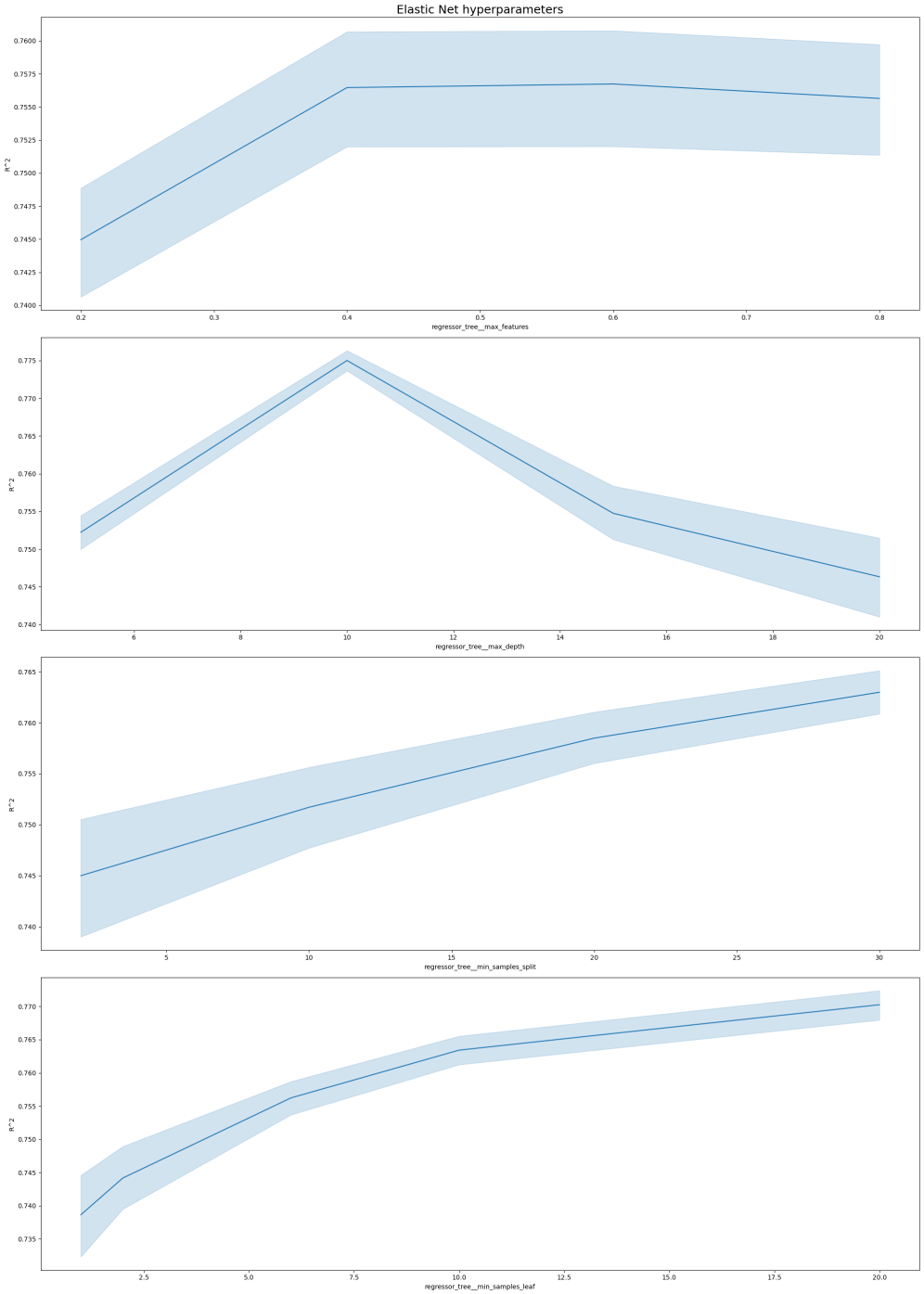

Figure 8: Comparison of regression tree parameters.

3.2.4. Support Vector Machines (SVM) regression

Support Vector Machines (SVM) is a classification and regression algorithm that finds the hyperplane maximizing the margin between different classes [14]. Support Vector Regression (SVR) is a supervised learning algorithm that is the regressional counterpart of Support Vector Machines (SVM) used for classification problems. While SVM aims to find a hyperplane that separates classes with the maximum margin,

SVR aims to find a hyperplane that fits as many data points as possible within a specified margin. This hyperplane is determined by a subset of training samples known as support vectors.

The optimization problem for a linear SVM is:

$$\min \frac{1}{2}||w||^2 \text{ s.t } y_i(w \cdot x_i + b) \geq 1, \text{ for all } i \tag{4}$$

Here w is the weight vector, b is the bias term, y_i is the class label and x_i is the feature vector

Similar to SVM, SVR can utilize kernel functions to transform the input data into a higher-dimensional feature space [15]. Kernels allow SVR to handle non-linear relationships between input and output variables by implicitly mapping them to a higher-dimensional space where they may become linearly separable. In our implementation we used both linear and RBF kernels to compare their performance.

Our implementation

We set up a pipeline with a StandardScaler and an SVM regressor with a linear kernel. Using GridSearchCV, we tuned the hyperparameters to identify the best model, and we visualized the impact of the hyperparameters (figure 9).

For the best model we got a result of

```
{'svm__C': 0.3981071705534973,
'svm__gamma': 0.01}
```

We evaluated the model's performance on the training and test datasets using cross-validation (10 folds),

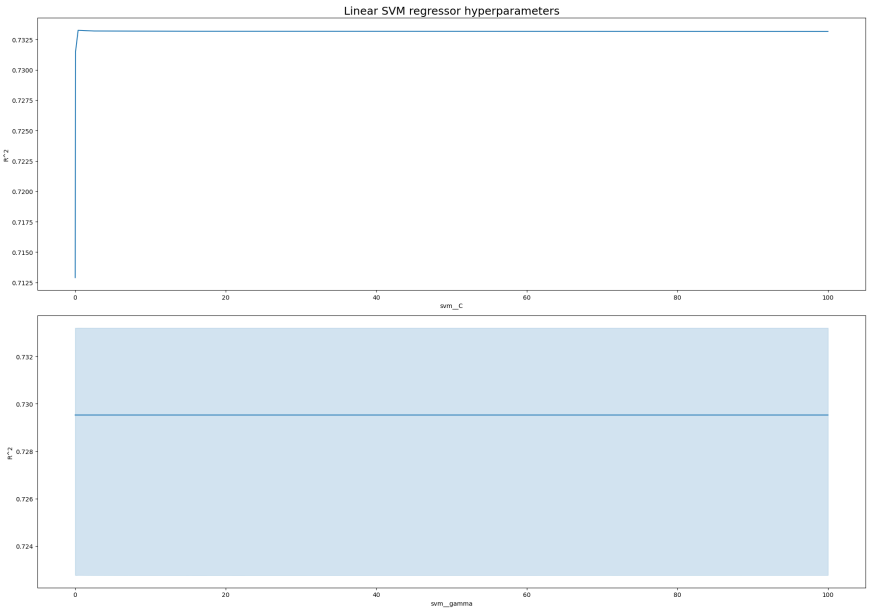

Figure 9: Comparison of linear SVM regressor parameters.

Similarly, we created a pipeline for the SVM regressor with an RBF kernel. After tuning the hyperparameters with GridSearchCV, we found the best model. We got a result of

```
{'svm__C': 100.0,
'svm__gamma': 2.5118864315095824}
```

As for the linear kernel SVR we visualized the parameter evolution to understand their effect on the model's performance (figure 10).

Then, using a cross validatiom (10 folds) we assessed the best model's performance on both the training and test datasets.

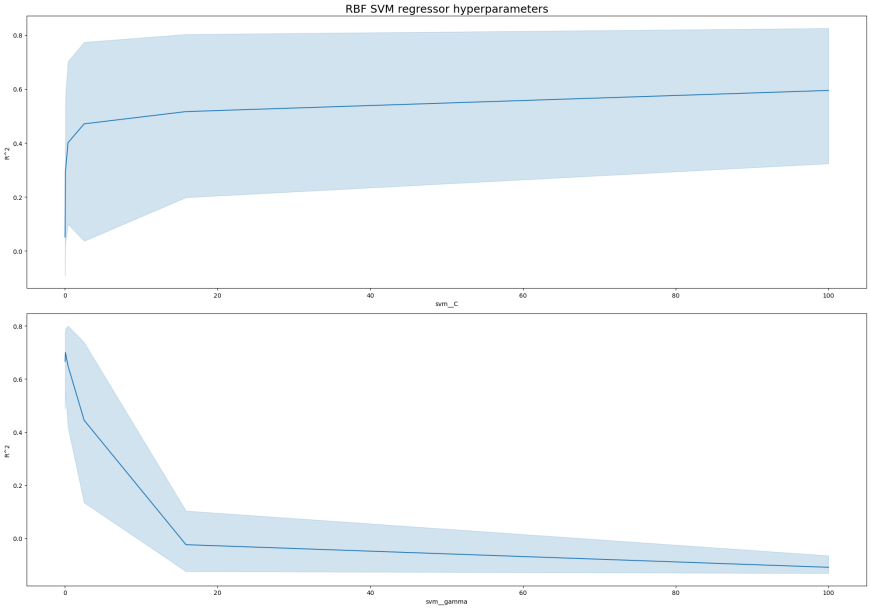

Figure 10: Comparison of RBF SVM regressor parameters.

3.2.5. Random Forests

Random Forests is an ensemble learning technique that constructs multiple decision trees during training and aggregates their outputs to improve predictive accuracy. Each tree in the forest is built from a random subset of the training data, and at each split in the tree, a random subset of features is considered [16].

More in detail Regression trees within a Random Forest are built using bootstrap samples of the training data. At each node, a random subset of features is selected, and the best split is chosen to minimize the mean squared error. This randomness helps create diverse trees, which collectively provide a more robust and generalized model. The predictions of all trees are averaged to produce the final output, which reduces variance and improves robustness compared to a single tree [17]. This ensemble approach mitigates the risk of overfitting.

The theoretical strength of Random Forest regression comes from its ability to manage the bias-variance tradeoff effectively, leading to lower generalization error. The combination of bootstrap sampling, feature randomness, and aggregation enhances model stability and accuracy [18].

Random Forests provide also a natural way to estimate feature importance by measuring the impact of each feature on the prediction accuracy [16]. The main steps of the Random Forest algorithm are:

1. Draw B bootstrap samples from the training data.
2. For each bootstrap sample, grow an unpruned decision tree:
   (a) At each node, select a random subset of m predictors.
   (b) Split the node using the best predictor among the subset.
3. Aggregate the predictions of the B trees to make the final prediction.

Our implementation

In our implementation of Random Forests, we started by defining a pipeline with a StandardScaler and a RandomForestRegressor, enabling the score to evaluate the model using out-of-bag samples.

We created a grid of hyperparameters to identify the optimal combination of hyperparameters that maximizes the model's performance. We performed the tuning using GridSearchCV and we got as a result:

```
{'random_forest__max_depth': 20,
'random_forest__max_features': 10,
'random_forest__n_estimators': 300}
```

We visualized the performance trends by plotting the evolution of mean test scores as hyperparameters vary (figure 11). Using cross validation (10 folds), we evaluated the performance of the tuned Random Forest Regressor on both the training and test datasets on the performance metrics described above.

We displayed also the out-of-bag (OOB) score of the Random Forest Regressor. to estimate the performance of the model without the need for a separate validation dataset. This metric provides an estimate of the model's predictive accuracy using unseen data. We got as a result:

```
OOB Score: 0.84895
```

To gain further insights, we analyzed and plotted the feature importance, highlighting which features contributed most to the model's predictions (figure 12).

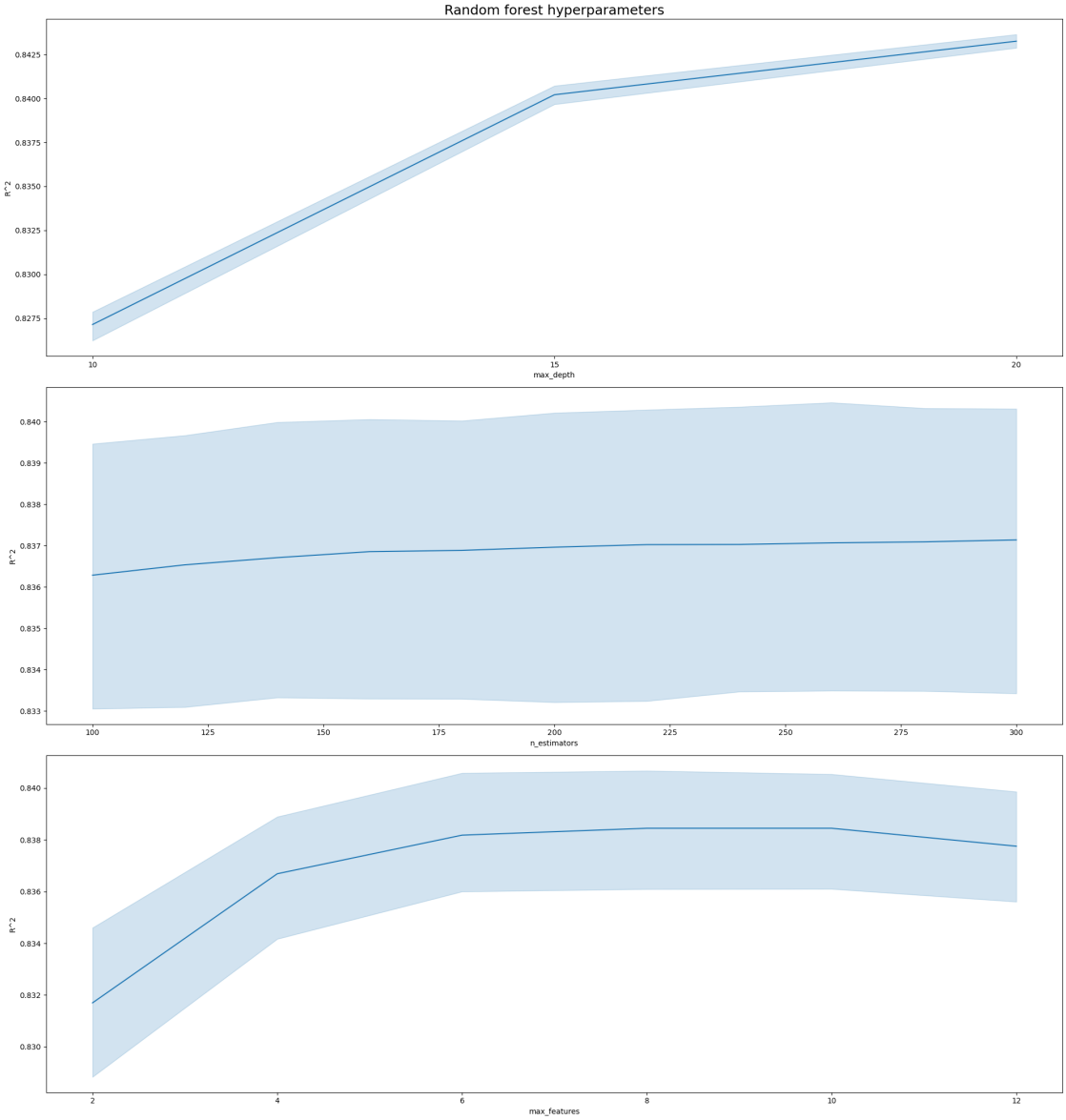

Figure 11: Comparison of random forest parameters.

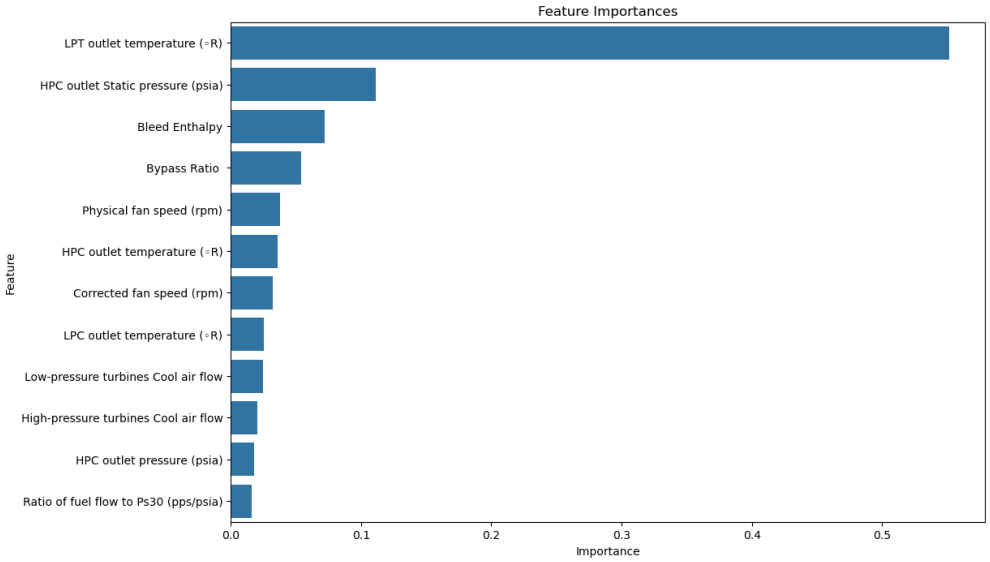

Figure 12: Analyzed feature importance for random forest model.

3.2.6. Neural Network

A neural network is a computational model inspired by the human brain, consisting of interconnected nodes (neurons) that process data in layers to recognize patterns and make predictions.

Our implementation

The best-performing model was composed by a network with two hidden layers, composed respectively by 32 and 16 neurons, as shown in the following summary (figure 13):

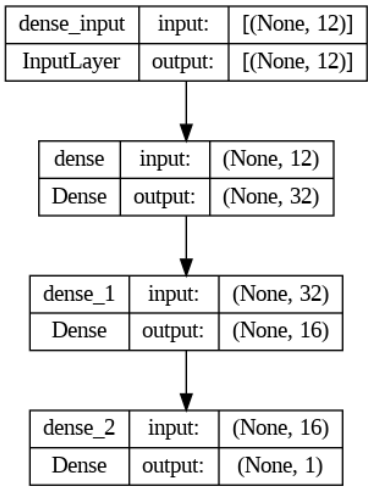

Figure 13: Neural network summary.

The model has been trained with a batch size of 16 for 250 epochs. It reserves 10% of the training data for validation and includes an early stopping callback that monitors the mean squared error. The training will stop if there is no improvement in the error for 20 consecutive epochs and will restore the model weights to the best observed during training. The following graphs shows the values of MSE (figure 14), RMSE (figure 15), MAE (figure 16) and R^2 (figure 17) of both training and validation splits during each training epoch:

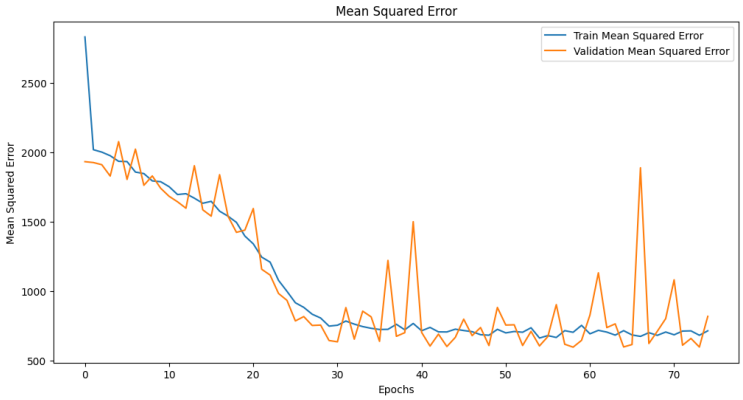

Figure 14: Neural Network Mean Square Error.

Figure 15: Neural Network Root Mean Square Error.

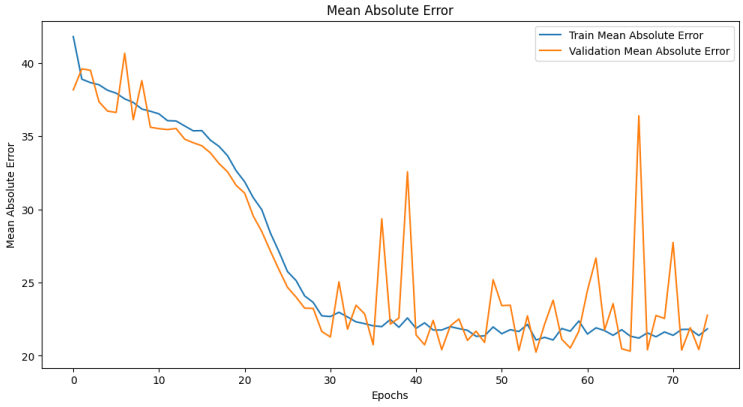

Figure 16: Neural Network Mean Absolute Error.

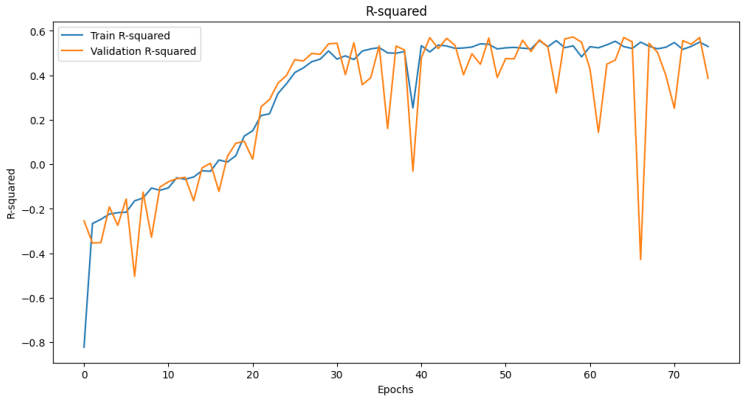

Figure 17: Neural Network R^2.

3.3. Software

The models were developed using the Python libraries Pandas, NumPy, and scikit-learn (SKlearn) [19][20][21]. SKlearn was the primary contributor, offering a comprehensive array of tools essential for machine learning, including pre-processing utilities, various machine learning algorithms, and evaluation metrics. For data visualization and plotting, seaborn and matplotlib were utilized [22][23].

4. Results and Discussion

Our project evaluated several machine learning models for predicting the remaining useful life (RUL) of turbofan engines, including Linear Regression, Elastic Net, Random Forests, Support Vector Machines (SVM), and Regression Trees. On the given dataset, we utilized data from sensors with preprocessing steps such as data cleaning, smoothing, and feature selection to enhance model performance. For better comparison on all tested models, we listed cross-validation methods for train set in table (table 1) and (figure 18). Detailed test results of our experiments are also evaluated in table (table 2) and comparison chart (figure 19).

Model	**R^2**	**Mean Absolute Error**	**Root Mean Squared Error**
Neural network	0.650	19.444	23.927
SVM regression - linear kernel	0.733	17.602	21.611
Linear Regression	0.734	17.569	21.579
Elastic Net Models	0.734	17.573	21.574
Regression Tree	0.782	13.796	19.526
Random forest	0.845	11.011	16.480
SVM regression - rbf kernel	0.869	10.275	15.114

Table 1: Comparison of Model Performance on train set.

Model	R^2	Mean Absolute Error	Root Mean Squared Error
Neural network	0.630	19.592	24.107
SVM regression - linear kernel	0.722	17.627	21.693
Elastic Net	0.723	17.626	21.678
Linear Regression Models	0.723	17.607	21.678
Regression Tree	0.757	14.267	20.292
SVM regression - rbf kernel	0.801	13.062	18.346
Random forest	0.814	12.904	17.742

Table 2: Comparison of Model Performance on Test Set

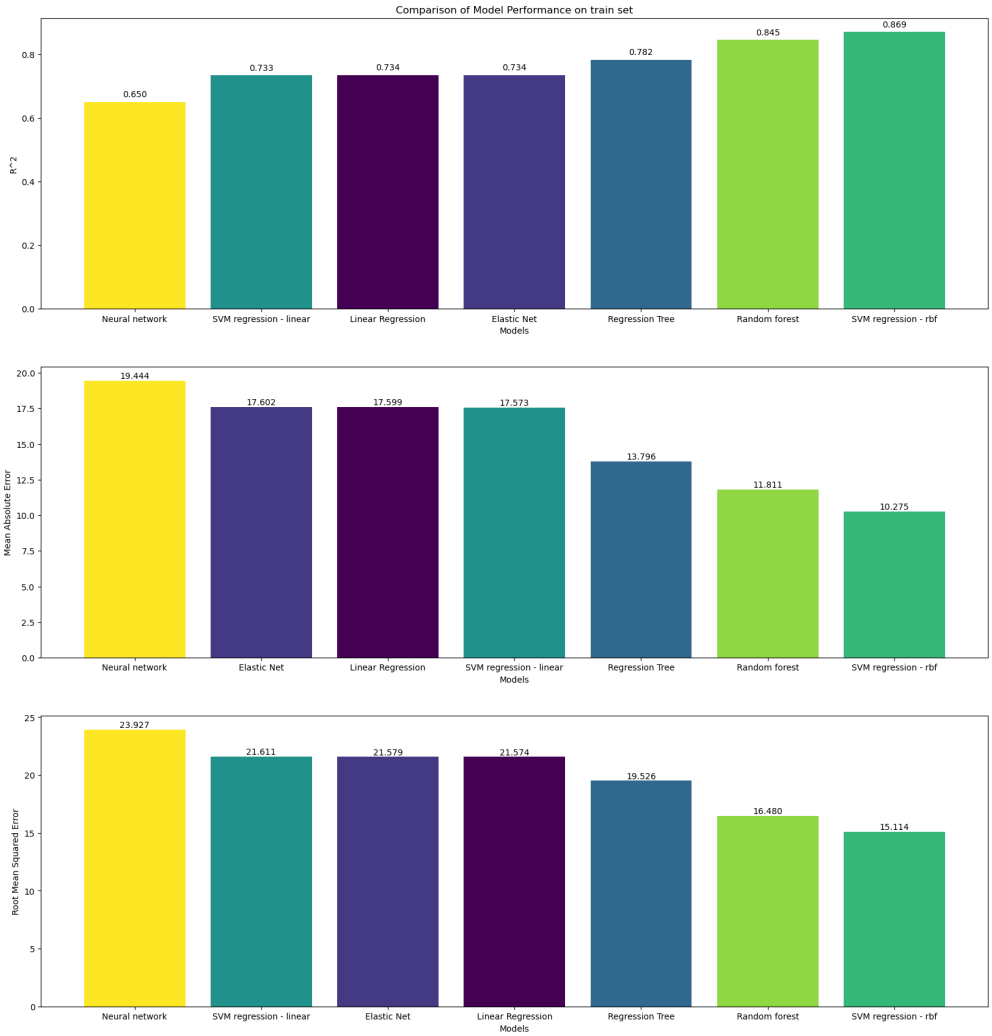

Figure 18: Comparison of Model Performance on train set.

The Random Forest model exhibited the best overall performance across the test sets and the second best performance on the train set, achieving high R^2 values (0.845 on training and 0.814 on test) and the lowest errors (MAE: 11.011 on training and 12.904 on test; RMSE: 16.480 on training and 17.742 on test). This suggests that Random Forests are highly effective at capturing the complex relationships in the data that are highly non-linear.

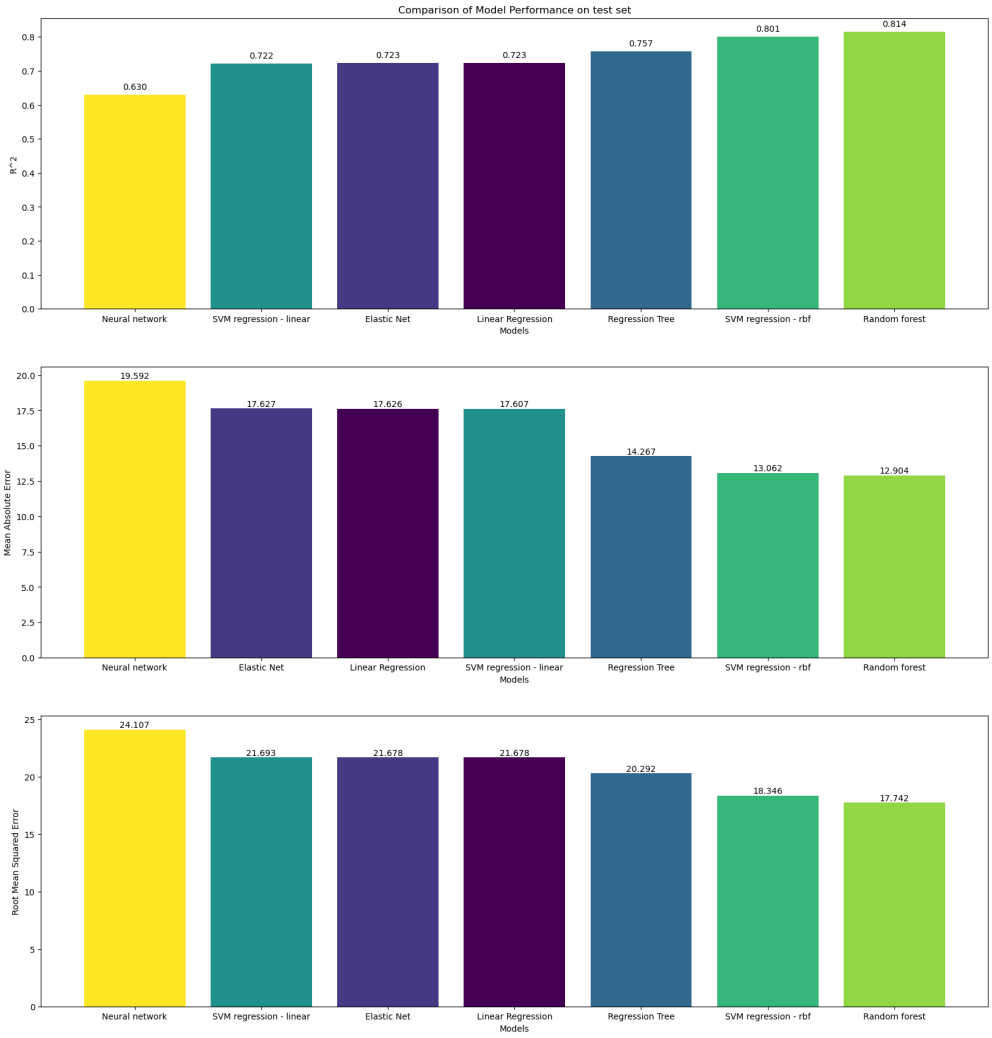

Figure 19: Comparison of Model Performance on test set.

The SVM with RBF kernel also performed exceptionally well, particularly in capturing non-linear relationships, as evidenced by its highest R^2 value on the train dataset and the second highest score in the test dataset (0.869 on training and 0.801 on test) and low errors (MAE: 10.275 on training and 13.062 on test; RMSE: 15.114 on training and 18.346 on test). This model's ability to handle non-linearity makes it a strong candidate for RUL prediction in turbofan engines.

Linear models, including Linear Regression and Elastic Net, performed moderately well, with similar R^2 values around 0.723 on the test set and MAE/RMSE around 17.6/21.7, indicating that while they capture some relationships in the data, their simplicity limits their predictive power compared to more complex models. Linear and Elastic Net regression have similar values, that shows that the regularization parameters didn't affect much the Linear model.

Neural Networks, despite their flexibility, did not perform as well as expected, likely due to the limited size and complexity of our dataset. The Neural Network model had the lowest R^2 values (0.650 on training and 0.630 on test) and higher errors compared to Random Forest and SVM models.

Regression Trees provided a good balance between simplicity and performance, with better results than linear models but not as strong as Random Forest or SVM with RBF kernel.

4.1. Reflections on the Results

Pre-processing the dataset is effective because by applying feature selection and correlation analysis, data cleaning, and data smoothing steps on the dataset, it ensured that the models used the most relevant features for prediction. It also reduced noise and improved accuracy of the model.

Potential Improvements could be to apply advanced feature engineering, removing and cleaning out anomalous readings would improve the performance of the model.

Hyperparameter tuning, especially for non-linear complex models like Random Forests and SVMs led to improvement in results.

5. Conclusions

Our study demonstrates that for predicting the remaining useful life of turbofan engines, ensemble methods like Random Forests and sophisticated models like SVM with RBF kernel offer superior performance over simpler models such as Linear Regression and Elastic Net. The Random Forest model based on the achieved result shows high accuracy and robustness, making it a highly suitable choice for this task. These high-performance numbers were achieved mainly with max thresholding of RUL with the given number before it started going nonlinear, as explained in figure 4.

Future work could explore further enhancements such as more detailed hyper-parameters tuning and the integration of more advanced deep learning models, like Recurrent Neural Networks. Additionally, expanding the models with the use of other available sub-datasets with slightly different features, could improve model generalization and accuracy.

6. Contributions

The authors Roberto Cialini, Jaka Škerjanc, Riccardo Rossi-Erba, Jan Fiala, and Chisom Okoye all partook and contributed equally to this project.

References

[1] J. Wei, P. Bai, D. Qin, T. C. Lim, P. Yang, H. Zhang, Study on vibration characteristics of fan shaft of geared turbofan engine with sudden imbalance caused by blade off, Journal of Vibration and Acoustics 140 (4) (2018) 041010.

[2] Federal Aviation Administration (FAA), In-flight emergencies statistics, FAA Safety Briefing (2022).

[3] International Air Transport Association (IATA), Economic Performance of the Airline Industry, IATA Annual Review, 2021.

[4] International Air Transport Association (IATA), Benefits of predictive maintenance in aviation, in: IATA Maintenance Cost Conference, 2023.

[5] A. Alhamaly, Jet engine remaining useful life (rul) prediction, accessed: 2024-05-28 (2022).
URL https://medium.com/@hamalyas_/jet-engine-remaining-useful-life-rul-prediction-a8989d52f194

[6] U. Thakkar, H. Chaoui, Remaining useful life prediction of an aircraft turbofan engine using deep layer recurrent neural networks, Actuators 11 (2022) 67. doi:10.3390/act11030067.

[7] K. Ensarioğlu, T. İnkaya, E. Emel, Remaining useful life estimation of turbofan engines with deep learning using change-point detection based labeling and feature engineering, Applied Sciences 13 (21) (2023) 11893. doi:10.3390/app132111893.

[8] . B.-D. J. Solis-Martin D., Galán-Páez J., A stacked deep convolutional neural network to predict the remaining useful life of a turbofan engine., arXiv preprint arXiv:2111.12689. (2021).

[9] F. Niaz, Understanding evaluation metrics in machine learning: R-squared is a pain!, linkedIn article (2022).
URL https://www.linkedin.com/pulse/understanding-evaluation-metrics-machine-learning-r-squared-pain

[10] D. N. Gujarati, Linear Regression: A Mathematical Introduction, Sage Research Methods, SAGE Publications, Inc, 2019.
URL https://methods.sagepub.com/book/linear-regression

[11] H. Zou, T. Hastie, Regularization and variable selection via the elastic net, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67 (2) (2005) 301–320.

[12] J. Friedman, T. Hastie, R. Tibshirani, Regularization paths for generalized linear models via coordinate descent, Journal of Statistical Software 33 (1) (2010) 1–22.

[13] L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone, Classification and Regression Trees, Wadsworth International Group, 1984.

[14] M. Awad, R. Khanna, Support Vector Regression, Apress, Berkeley, CA, 2015, pp. 67–80. doi:10.1007/978-1-4302-5990-9_4.
URL https://doi.org/10.1007/978-1-4302-5990-9_4

[15] D. Team, Kernel functions - introduction to svm kernel examples, [Blog post] (2017).
URL https://data-flair.training/blogs/svm-kernel-functions/

[16] L. Breiman, Random forests, Machine Learning 45 (1) (2001) 5–32. doi:10.1023/A:1010933404324.
URL https://doi.org/10.1023/A:1010933404324

[17] A. Liaw, M. Wiener, et al., Classification and regression by randomforest, R news 2 (3) (2002) 18–22.

[18] G. Biau, E. Scornet, A random forest guided tour, TEST 25 (2) (2016) 197–227. doi:10.1007/s11749-016-0481-7.
URL https://doi.org/10.1007/s11749-016-0481-7

[19] N. Developers, NumPy User Guide, NumPy Project (2024).
URL https://numpy.org/doc/stable/user/index.html

[20] W. McKinney, pandas: User Guide, PyData (2010).
URL https://pandas.pydata.org/docs/user_guide/index.html

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Penvén, J. Vanderplas, R. Weiss, et al., scikit-learn: Machine Learning in Python (2011).
URL https://scikit-learn.org/stable/

[22] S. developers, seaborn: statistical data visualization, PyData (2024).
URL https://seaborn.pydata.org/

[23] M. developers, Matplotlib: User Guide, Matplotlib project (2024).
URL https://matplotlib.org/stable/users/index.html