# Machine Learning

Kruthika Mysore Bhaskar

Word Count: 2999

## 1.Executive Summary

To reduce flaw rates and boost production effectiveness, a machine-learning approach was deemed essential for accurately forecasting the lifespan of metal components. Two strategies became considered: regression models—Random Forest and XGBoost—to estimate lifespan, and classification models—Random Forest and Support Vector Machine—to categorize parts as durable or non-durable.

The models were refined through tuning of hyperparameters and underwent stringent evaluation. Random Forest regression yielded highly accurate results with very minimal errors, while on the other hand, the best results were shown by the SVM classification model, yielding almost perfect precision and correctly classifying parts with respect to durability.

The model of SVM classification is advised for implementation because of its better precision and suitability for the task, offering an effective solution for reducing flaws and enhancing production results

## 2.Data Exploration

***Data Loading and Verification***

A range of production factors and material properties are on the dataset, which was loaded using pandas for efficient handling and analyses. The composition of the dataset was checked for validity (Fig.1): attributes, counts of non-null entries and value ranges across all 16 columns. This verification demonstrated that no missing values existed and ensured uniformity in the dataset for further analysis.

***Exploratory Analysis and Visualizations***

Key exploratory visualizations were created to find relations between production parameters and metal's lifespan. The correlation matrix in (Fig.2) shows the strong positive correlations between some

metal composition features, like Nickel%, with lifetime, whereas the Iron% and defect counts had poor or negative correlation with those features. This (Fig.1) suggests that a higher content of Nickel% will have a positive influence on the lifetime, while the level of Iron and defect count could inversely affect it.

```
Data columns (total 16 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Lifespan        1000 non-null   float64
 1   partType        1000 non-null   object
 2   microstructure  1000 non-null   object
 3   coolingRate     1000 non-null   int64
 4   quenchTime      1000 non-null   float64
 5   forgeTime       1000 non-null   float64
 6   HeatTreatTime   1000 non-null   float64
 7   Nickel%         1000 non-null   float64
 8   Iron%           1000 non-null   float64
 9   Cobalt%         1000 non-null   float64
 10  Chromium%       1000 non-null   float64
 11  smallDefects    1000 non-null   int64
 12  largeDefects    1000 non-null   int64
 13  sliverDefects   1000 non-null   int64
 14  seedLocation    1000 non-null   object
 15  castType        1000 non-null   object
```

*Figure.1:Dataset Overview*

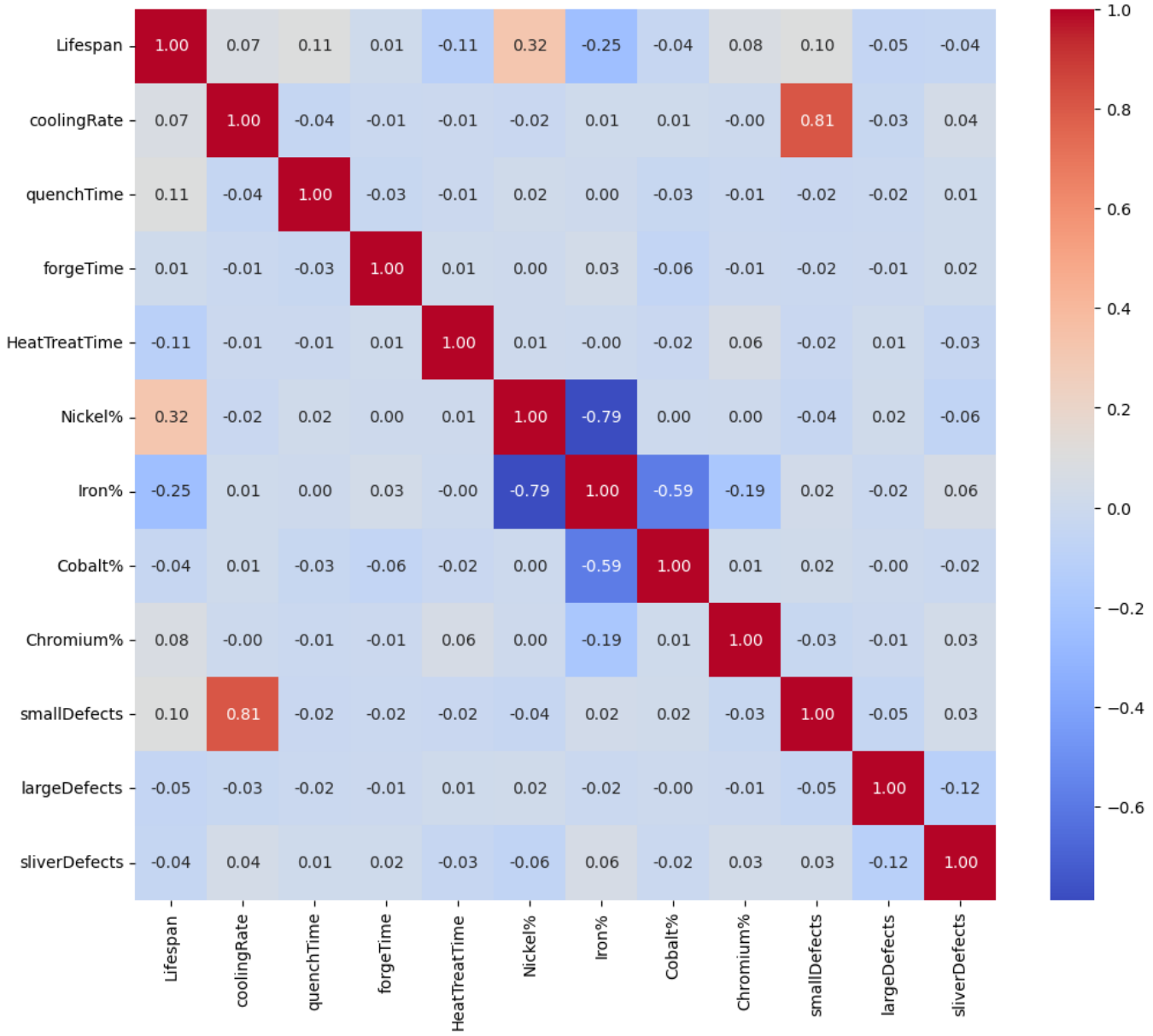


*Figure.2:Correlation Matrix of Numerical Features*

In addition, the spread of the variable lifespan was obtained by a histogram (Fig. 3), showing a central concentration, having a normal-like pattern with some striking lower-valued outliers. This histogram reinforces that the prediction of lifespan could take advantage of models considering the concentration and spread of data, especially around the central value. Furthermore, (Fig. 3) represents the overall lifespan distribution, which can give an indication for model development based on its mean.

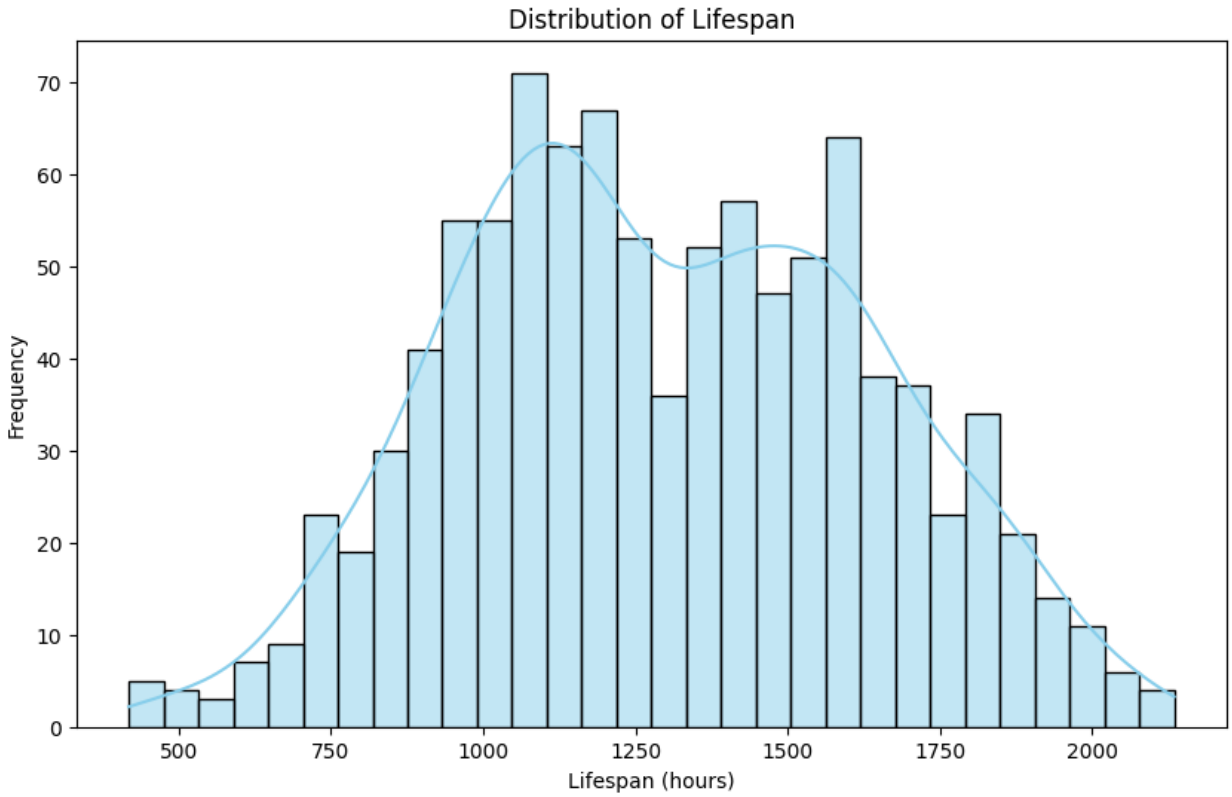


*Figure.3:Histogram of Lifespan Distribution*

***Feature Selection and Model Justification***

Feature selections based on these findings are Nickel%, Iron%, coolingRate, and imperfection features. They come in agreement with the association with these observations that Nickel% is positively correlated with lifespan, while the counts of imperfections have the potential to have a negative impact. And from the strong association with these features, it might be helpful for regression tasks to estimate the lifespan.

***Model Selection***

Since the linear relationships, distribution spread, and correlation structure observed in the data are manifold, approaches shall also be manifold. Regression models, such as Random Forest or XGBoost, might be indicated for continuous variables to predict lifespan. However, the task of classification into classes based on lifespan is also open to a consideration of classification models such as SVM.

This broad data exploration now uncovers actionable insights that will inform feature selection and model choice for a robust foundation in predictive analytics.

# 3. Regression Implementation

## 3.1 Methodology

***Model Selection***

Characteristics of the dataset and the nature of the problem led to the selection of following regression models: Random Forest Regressor and XGBoost Regressor. Both models are good for this task since they allow solving nonlinear relationships and high-dimensional data.

*Random Forest Regressor* (RF): It is an ensemble chosen for its robustness in handling feature interactions and, at the same time, because it prevents overfitting by aggregating the predictions of a set of decision trees. This has been widely applied in industrial prediction tasks where relationships among features ae complicated and not strictly linear (Liaw & Wiener, 2002).

*XGBoost Regressor:* Known for its performance in regression problems with large datasets, XGBoost (Chen & Guestrin, 2016) was chosen for its ability to efficiently handle large amounts of data and its ability to fine-tune hyperparameters for optimal performance. XGBoost has become a go-to model in machine learning competitions due to its flexibility and high predictive power.

Both are powerful models that allow for effective parallelization and have been shown to provide very accurate predictions for effective parallelization and have been shown to provide very accurate predictions for continuous outcomes, it would be found in predicting the lifespan of various parts.

.

***Preprocessing Routine***

There were a few steps taken to preprocess this dataset and prepare the input features for modeling:

1. *Feature Encoding*: OneHotEncoder was used for encoding categorical features like partType, microstructure, and seedLocation. In this technique, the categorical data is converted into a format that can be provided as input to machine learning algorithms.

2*. Feature Scaling*: As the dataset comprises both numeric and categorical data, the numeric features like coolingRate, Nickel%, and Iron% were scaled using StandardScaler to ensure that the model does not give undue importance to features with larger numerical ranges.

3. *Train-Test Split*: The data was split into 80% for training and 20% for testing using train_test_split from the sklearn.model_selection module. It helps evaluate model performance on unseen data.

4.*Data Balancing*: Given that the dataset did not present class imbalances for lifespan prediction (regression problem), balancing techniques were not required. However, in case of classification tasks, balancing methods like SMOTE would be explored to ensure fairness if needed.

***Hyperparameter Tuning Framework***

The most important point towards the optimization of the model performance relies on the tuning of the hyperparameters. Perform hyperparameter tuning using Bayesian optimization with BayesSearchCV from the skopt library. It selects the best values of hyperparameters concerning performance and allows an efficient search of parameter space.

1.Random Forest Regressor Hyperparameters:

*n_estimators*: The number of trees in the forest. Values from 100 to 500 were explored(Fig.4). The more trees usually perform better, increasing computational costs as well. The optimum value of 351 was chosen after exploring the performance through grid search and cross-validation.

*max_depth:* The maximum depth of the trees. Greater values are capable of capturing more intricate relationships but may be prone to overfitting. A depth of 30 was decided upon using the performance from cross-validation.

*min_samples_split:* The minimum number of samples required to split an internal node. Lower values allow the model to build finer divisions of the data. For this problem, a value of 2 was used.

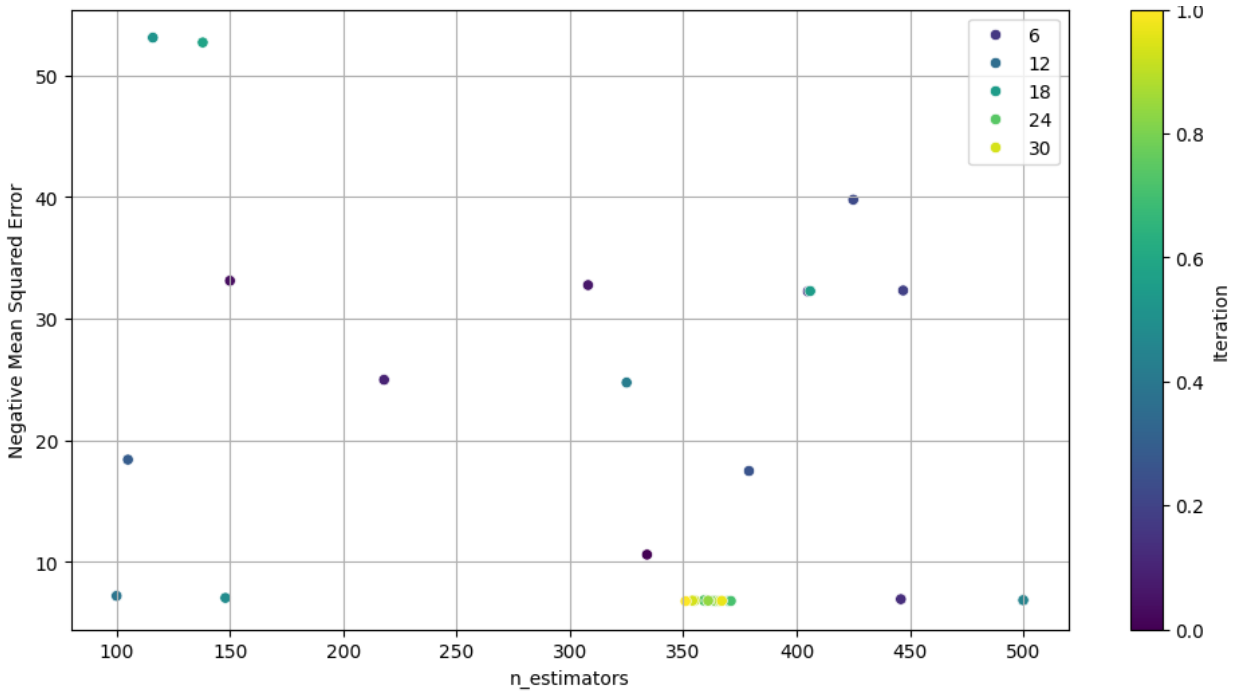


*Figure.4:Random Forest – Impact of n_estimators*

2. XGBoost Regressor Hyperparameters:

*n_estimators:* The number of boosts. After trying values between 100 and 500, 288 was chosen, as it offered optimal accuracy without excessive computational cost.

*learning_rate:* Control the quantity that the model adapts itself in each iteration of training. Here, a relatively small value, such as 0.0446, was chosen to be more probably convergent and to enhance model stability(Fig.5).

*max_depth:* This regulates the complexity of base trees. Here, it was set to 5 to avoid overfitting and, consequently, to avoid loss of generalization.

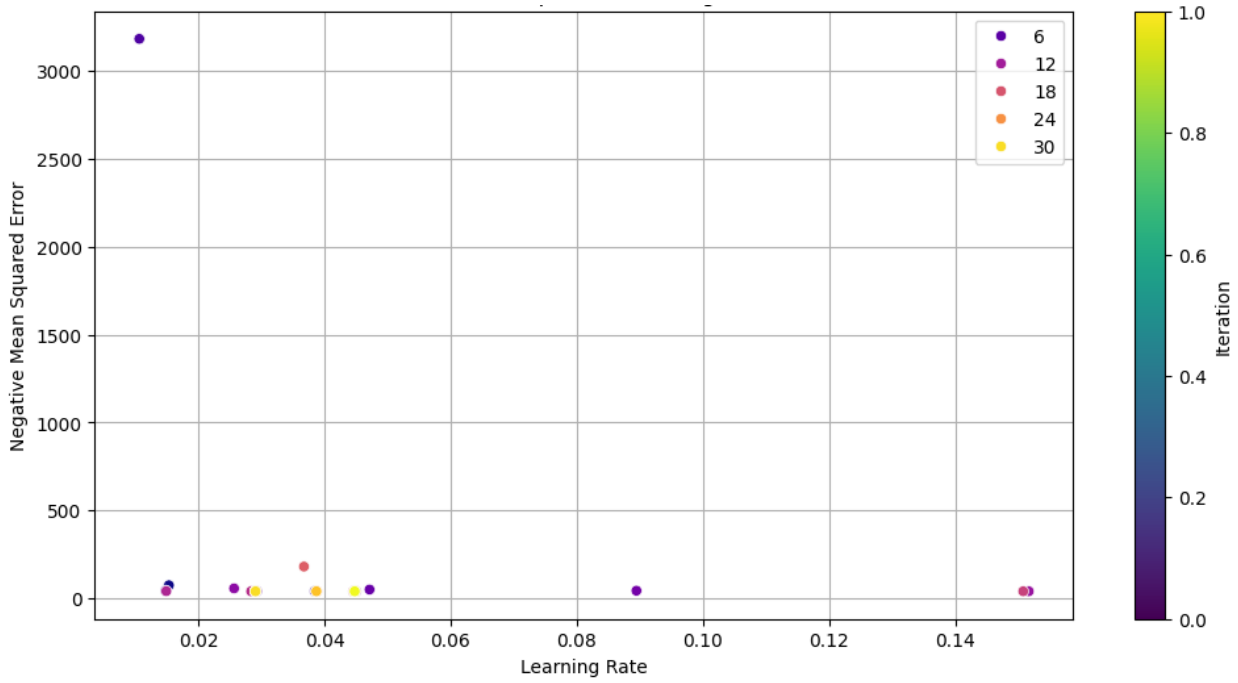


*Figure.5:XGBoost – Impact of Learning Rate*

| Model | Parameter | Search Space | Best Value |
|---|---|---|---|
| **Random Forest** | n_estimators | (100, 500) | 351 |
| | max_depth | (10, 30) | 30 |
| | min_samples_split | (2, 10) | 2 |
| **XGBoost** | n_estimators | (100, 500) | 288 |
| | max_depth | (0.01, 0.2) | 0.0446 |
| | min_samples_split | (3, 10) | 5 |

*Figurer.6:Hyperparameter Search Space and Best Parameters*

Hyperparameter tuning was done in both using cross-validation (CV) with folds of 5 each. The reason behind choosing this method for hyperparameter tuning is to ensure model parameters are chosen to avoid overfitting and hence maximize performance.

Both Random Forest and XGBoost regressors are tuned to their best performance by selected models with care, preprocessing routines, and optimizations of hyperparameters. The hyperparameters were chosen by systematic searches guided by cross-validation results, which balanced model complexity with predictive accuracy.

## 3.2 Evaluation

***Model Experiments and Optimization***

Both the Random Forest Regressor and XGBoost Regressor underwent extensive hyperparameter tuning to optimize accuracy for the task of predicting metal part lifespan. The models were tuned using Bayesian optimization with a cross-validation strategy (5-fold) to systematically identify the best hyperparameter combinations. Hyperparameter values for each model were iteratively tested, with each step recorded for comparative analysis, ensuring that only the best configurations were selected for final evaluation.

In the **Random Forest model**, three main hyperparameters, namely, n_estimators, max_depth, and min_samples_split, have been tuned. It follows that, within model performance gradually increases with an increase in the number of trees. Accordingly, it is observed(Fig.8) that the optimal n_estimators value of 351 balanced accuracy and efficiency, whereas max_depth with a value of 30 effectively captured complex patterns in the data. A minimum sample split of 2 was selected to enable the model to make deep splits in the data, hence capturing detailed relationships vital for the prediction of lifespan.

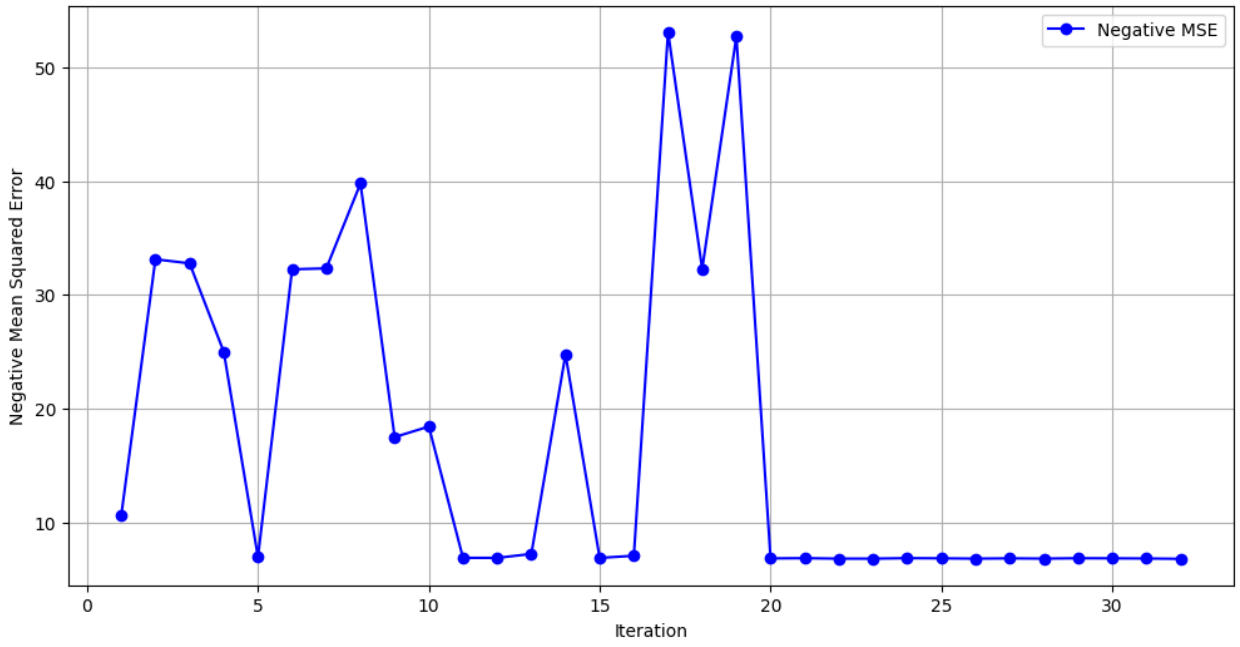


*Figure.7:Random Forest Hyperparameter Tuning Progression*

| Parameter | Value Range | Optimal Value |
|---|---|---|
| n_estimators | 100 - 500 | 351 |
| learning_rate | 10-30 | 30 |
| max_depth | 03-10 | 2 |

*Figure.8:Random Forest Hyperparameter Tuning Summary*

The **XGBoost Regressor** was tuned similarly, focusing on n_estimators, learning_rate, and max_depth. The iterative tuning revealed that an n_estimators value of 288 was optimal, balancing model complexity with efficiency. The learning_rate was adjusted to 0.0446, ensuring a stable yet responsive learning process. A maximum depth of 5 was selected, preventing overfitting and improving the model's ability to generalize to new data, as evidenced by cross-validation performance (Chen & Guestrin,2016).

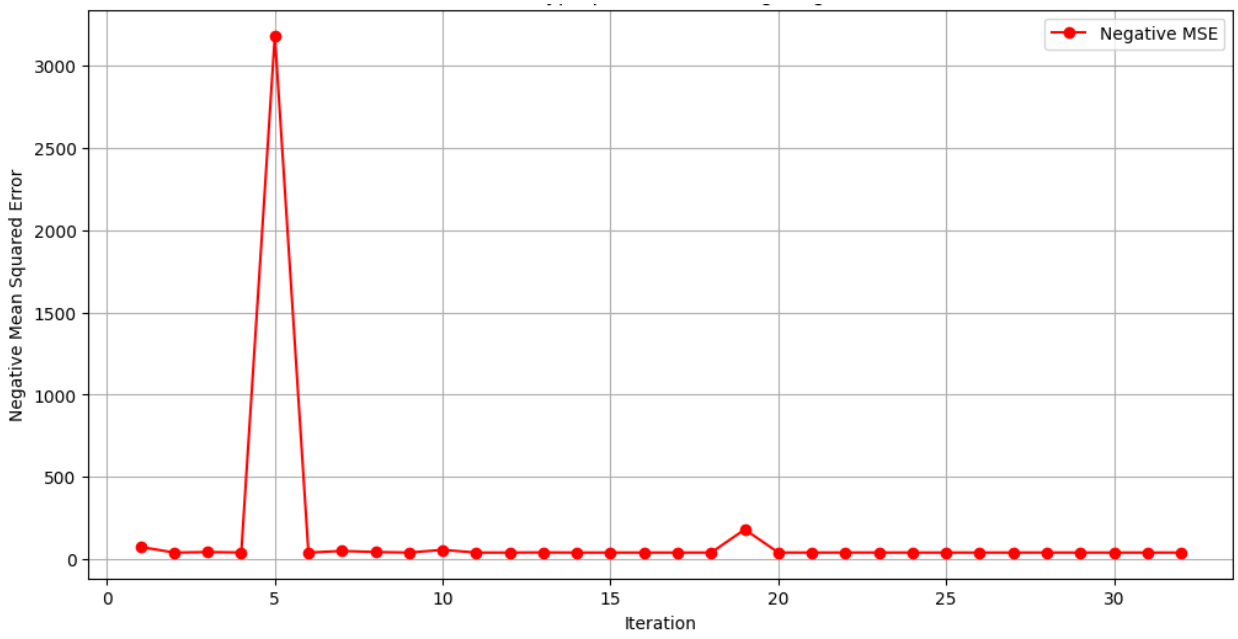


*Figure.9:XGBoost Hyperparameter Tuning Progression*

| Parameter | Value Range | Optimal Value |
|---|---|---|
| n_estimators | 100 - 500 | 288 |
| learning_rate | 0.01 - 0.2 | 0.0446 |
| max_depth | 03-10 | 5 |

*Figure.8: XGBoost Hyperparameter Tuning Summary*

***Performance Metrics***

The performance metrics of the models used are standard regression metrics in assessing predictive accuracy: Mean Squared Error-MSE, Root Mean Squared Error-RMSE, and R-squared-$R^2$. MSE gives a sense of the average prediction error, while RMSE means the magnitude of error, which is the square root of MSE; hence, the units of error and target variables match. $R^2$ exhibits the proportion of explained variance in the lifetime by the model. Obviously, a value closer to 1 indicates better performance. The results of the evaluation is in (Fig.9) below.

**Best Model Comparison**

Among the final model comparisons, the Random Forest Regressor outperformed XGBoost regarding the MSE and RMSE, which are important measures of accuracy for continuous variable predictions. Random Forest had much better MSE and RMSE, thus establishing that it has higher precision over the prediction of the lifespan of the metal part. Both models have achieved an $R^2$ score close to perfect, meaning each has captured almost all the variances in the data. The Error metrics, though, suggest that Random Forest is suitable for deployment since its overall prediction error is low, even though it has similar $R^2$ values.

```
Model Evaluation Results:
                        MSE       RMSE        R²
Random Forest      3.074251   1.753354  0.999974
XGBoost           36.376817   6.031320  0.999697
```

*Figure.9:Model Performance Metrics*

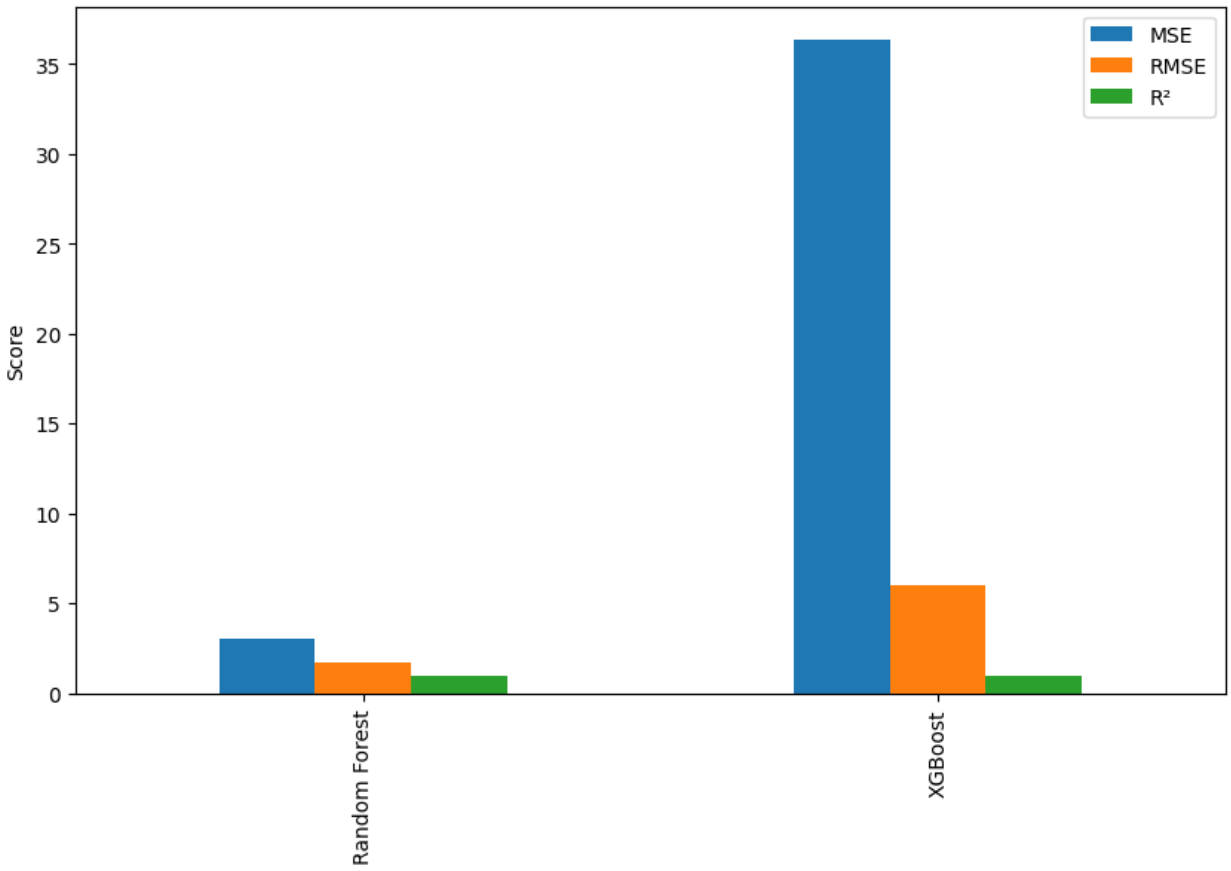


*Figure.10:Model Performance Comparison (MSE and RMSE)*

The presented comprehensive comparison underlines the efficiency of hyperparameter tuning in model performance and supports the main goal regarding the optimization of predictive accuracy for operational utility in an industrial context.

As shown in (Fig.10), Random Forest demonstrates better accuracy with lower MSE and RMSE on both training and testing sets, indicating a stronger fit and minimal overfitting. In contrast, XGBoost achieves a high $R^2$ of 1.00 but with higher error values, suggesting slightly less precision in predictions, detailed variation is shown in (Fig.9).

| Model | Metric | Training Set | Testing Set |
|---|---|---|---|
| Random Forest | MSE | 0.61 | 3.01 |
| | RMSE | 0.78 | 1.73 |
| | $R^2$ | 1 | 1 |
| XGBoost | MSE | 14.31 | 36.6 |
| | RMSE | 3.78 | 6.05 |
| | $R^2$ | 1 | 1 |

*Figure.11: Performance Metrics for Random Forest and XGBoost on Training and Testing Sets.*

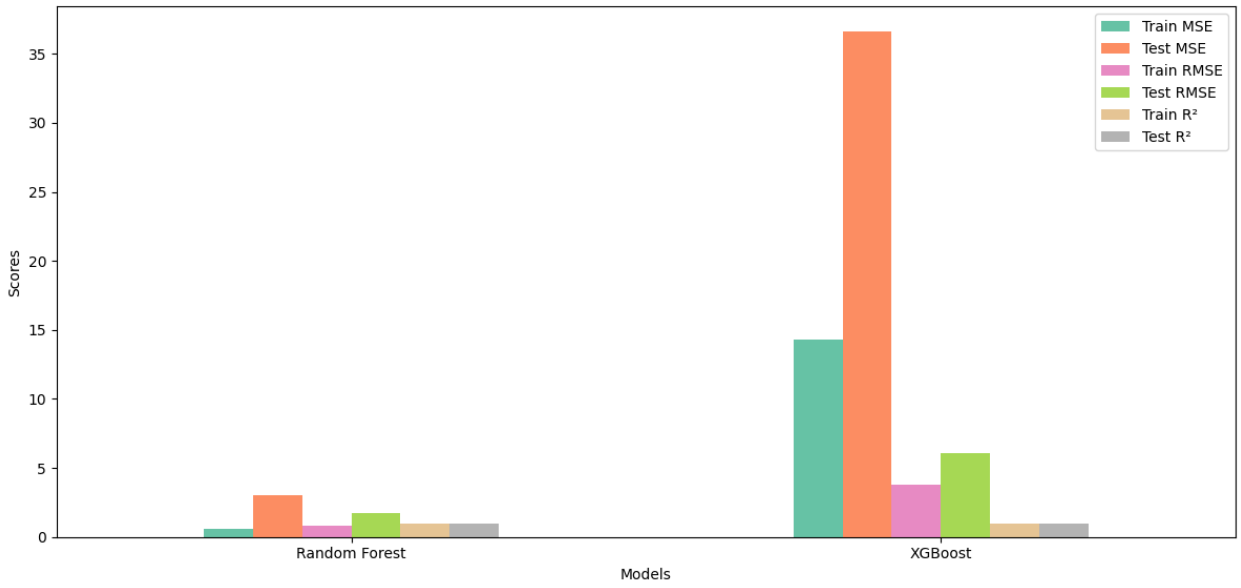


*Figure12:Training vs Testing : Model Performance Comparison*

## 3.3 Critical Review

***Strengths and Limitations***

The chosen regression models, Random Forest Regressor and XGBoost Regressor, showcased robust performance, effectively handling complex, nonlinear relationships within the dataset. The Random Forest model excelled with a lower Mean Squared Error (MSE) and Root Mean Squared Error (RMSE), indicating superior predictive accuracy for lifespan estimation. The Bayesian optimization strategy

further enhanced model performance by identifying the best hyperparameter values through an efficient, structured search across multiple dimensions.

However, limitations were also identified. Random Forest, though accurate, is computationally intensive, particularly with higher n_estimators, which may hinder scalability in real-time applications. XGBoost, while faster, exhibited a slightly higher error rate, potentially due to its sensitivity to parameter changes and the requirement for precise tuning to achieve stability. Additionally, both models, while excelling in this particular dataset, may overfit with more extensive or varied datasets if tuning is not recalibrated periodically.

***Alternative Approaches***

To address these limitations, Gradient Boosting Regressor is suggested as an alternative, as it combines boosting with simpler base models, reducing overfitting while maintaining predictive power (Chen & Guestrin, 2016). Another recommendation involves Principal Component Analysis (PCA) for dimensionality reduction, which could streamline data preprocessing and improve model interpretability by focusing on key features with high variance contributions (Jolliffe, 2002). Lastly, regularization techniques such as L2 regularization within an Elastic Net framework could further enhance generalization by penalizing complex model structures, especially beneficial for production scalability (Zou & Hastie, 2005).

*Table 1: Model Comparison Summary*

| Model | Strengths | Limitations |
|---|---|---|
| Random Forest | High accuracy, handles non-linear interactions | Computationally intensive, potential overfitting |
| XGBoost | Fast training, effective in boosting | Sensitive to parameter changes, risk of instability |

This review underscores the models' effectiveness for the current dataset while recommending alternative approaches to enhance scalability and efficiency in future applications.

# 4.Classification Implementation

## 4.1 Feature Crafting

***Binary Classification Label for Lifespan:*** To aid in classification, a binary feature labeled "1500_labels" was created to indicate whether a component's lifespan exceeded 1500 hours. This additional label helped in distinguishing long-lasting components, which is critical for identifying factors affecting durability.

***KMeans Clustering***: KMeans clustering was applied to the principal components of the dataset, with clusters ranging from k=2 to k=10. Silhouette scores, which measure cohesion and separation, were calculated for each cluster count. A score of 0.347 suggested k=3 as optimal. The clustering (Fig.13) provided useful segmentation for the data, preserving major variances in a lower-dimensional space.

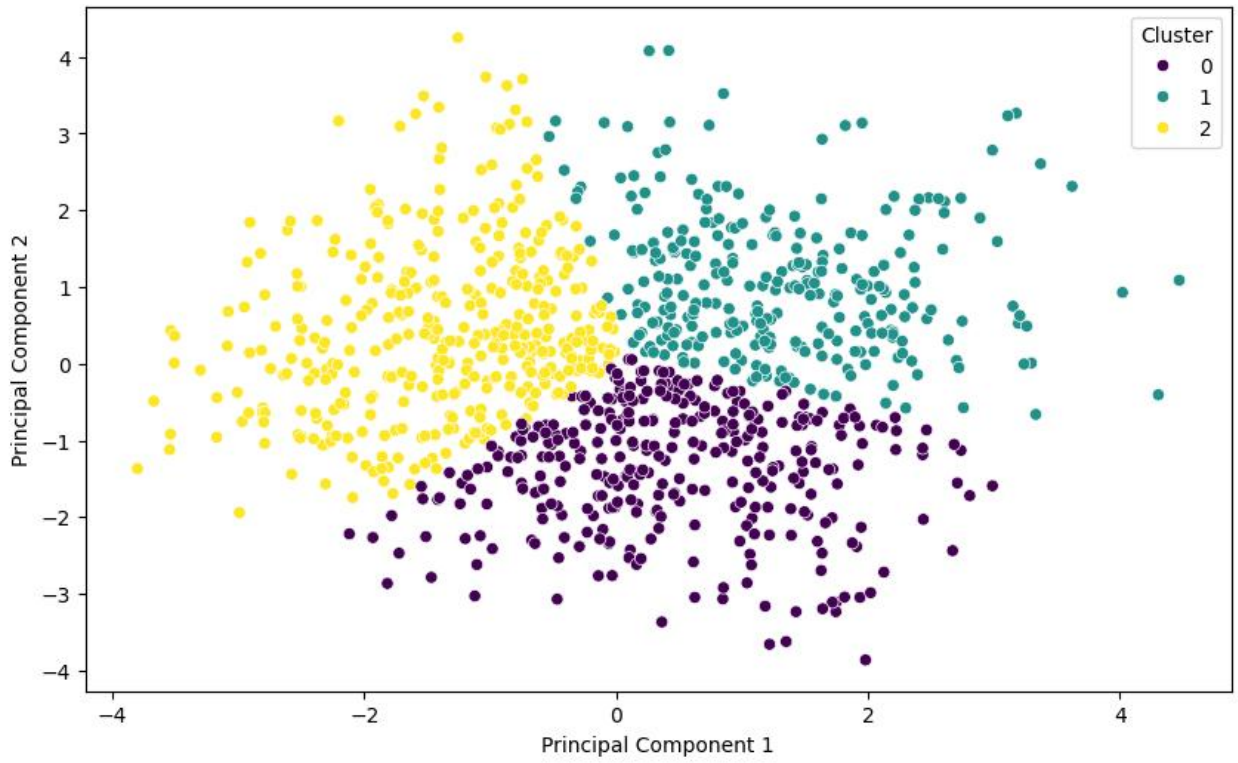


*Figure13:Cluster Formed by KMeans (K =3)*

***Elbow Plot and Silhouette Analysis***: An elbow plot and silhouette scores (Fig.14) were generated to validate k=3 as the best choice for achieving a balanced intra-cluster compactness and inter-cluster separation.

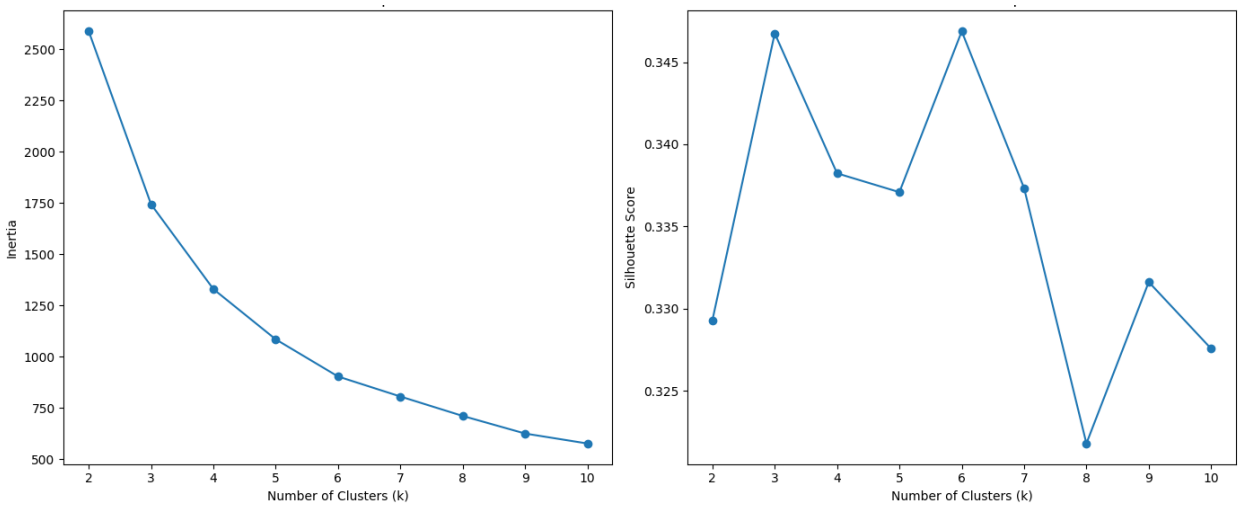


*Figure.14:Elbow Method and Silhouette score for Optimal k*

***Principal Component Analysis (PCA):*** PCA was applied to decrease the data to two primary dimensions. This dimensionality reduction preserved significant variance, streamlining clustering and enhancing computational efficiency during model training.

Feature Scaling: Numeric features were scaled using StandardScaler to ensure uniform input scales, essential for models like SVM.

Exclusion of Lifespan: Lifespan was excluded as an input to prevent data leakage during the prediction of defect status.

## 4.2 Methodology

***Model Selection and Data Split:*** The dataset was divided into training (70%) and testing (30%) sets to enable reliable evaluation of classification models. Two algorithms, Random Forest and Support Vector Machine (SVM), were selected for their robustness in handling complex, high-dimensional datasets.

Random Forest: Random Forest was trained with class weights to counter class imbalance. Using GridSearchCV for hyperparameter optimization, the final model achieved an accuracy of 96.0% and an AUC of 0.969, as seen in (Fig.15). The confusion matrix (Fig.16) revealed effective classification across all clusters, showing reliable performance in both high and low-class proportions.

Random Forest Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.95 | 0.99 | 0.97 | 94 |
| 1 | 0.99 | 0.92 | 0.95 | 85 |
| 2 | 0.95 | 0.97 | 0.96 | 121 |
| accuracy | | | 0.96 | 300 |
| macro avg | 0.96 | 0.96 | 0.96 | 300 |
| weighted avg | 0.96 | 0.96 | 0.96 | 300 |

Figure.15:Random Forest Classification Report

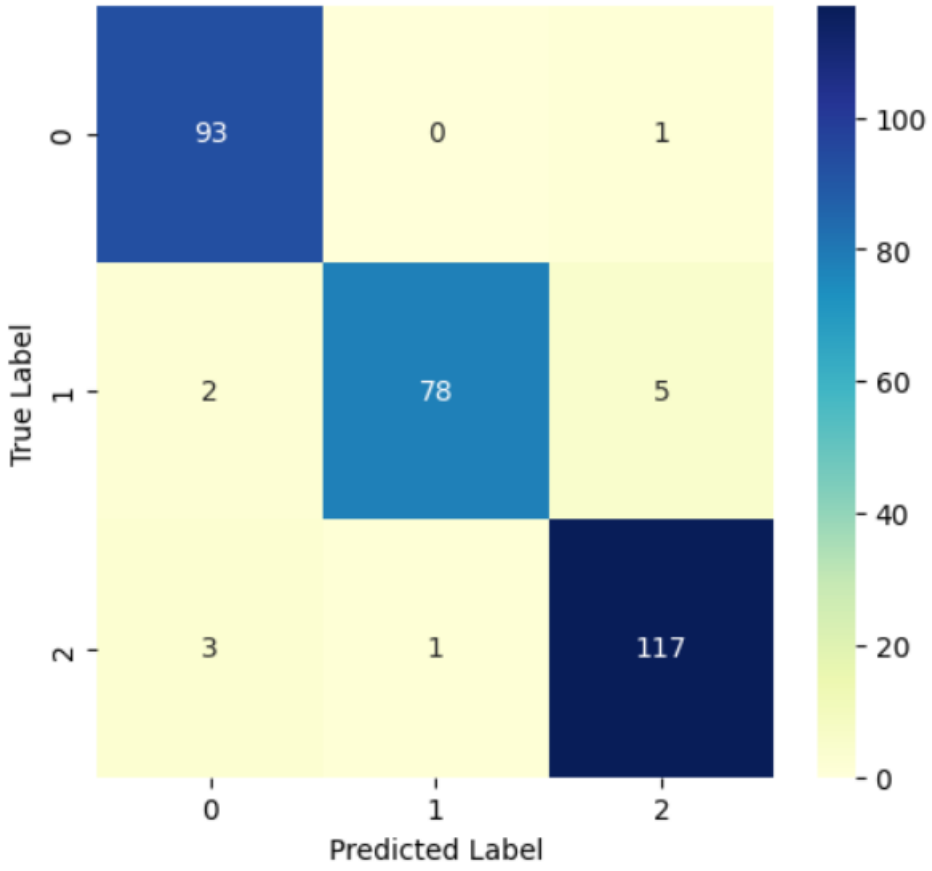


*Figure.16:Confusion Matrix for Random Forest*

**Support Vector Machine (SVM)** demonstrated strong results with an accuracy of 99.7% and an AUC of 0.997. Hyperparameter tuning via GridSearchCV further optimized its performance. The confusion

matrix (Fig.18) revealed near-perfect classification, and the classification report (Fig.17) showed weighted precision, recall, and F1-score all reaching 1.00, underscoring SVM's precision in this classification problem.

SVM Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 1.00 | 1.00 | 94 |
| 1 | 1.00 | 0.99 | 0.99 | 85 |
| 2 | 0.99 | 1.00 | 1.00 | 121 |
| accuracy | | | 1.00 | 300 |
| macro avg | 1.00 | 1.00 | 1.00 | 300 |
| weighted avg | 1.00 | 1.00 | 1.00 | 300 |

*Figure.17:SVM Classification Report*

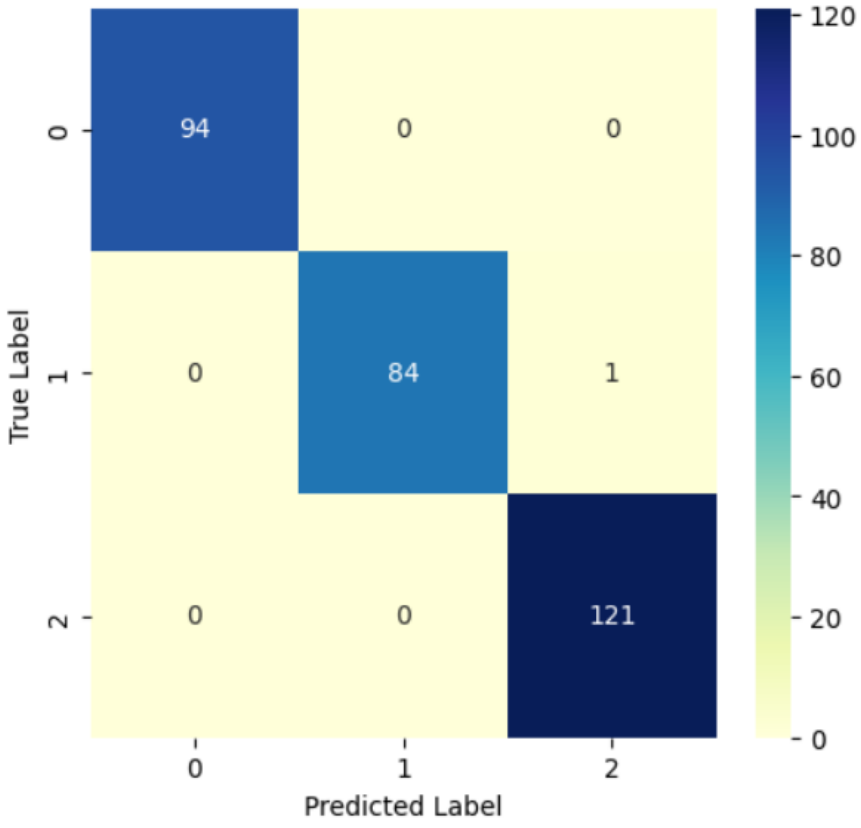


*Figure.18: Confusion Matrix for SVM*

## 4.3 Evaluation

***Model Comparison and Performance Metrics***: (Fig.20) provides a summary of the accuracy, AUC, precision, recall, and F1-score for both Random Forest and SVM models. SVM outperformed Random Forest across all metrics, with a slight edge that makes it preferable for this classification task.

***Performance Visualization***: A bar chart (Fig.21) was created to compare model performance metrics visually, making it clear that SVM achieved superior accuracy, AUC, and F1-scores. These findings highlight SVM's capability in handling the dataset's complexity, making it an optimal choice for this specific classification challenge.

SVM outperforms Random Forest in testing accuracy and AUC, demonstrating better generalization with minimal performance gap between training and testing sets. Its robust performance on both sets, without overfitting, makes SVM the preferred model for real-world deployment, as shown in (Fig.22) and the accompanying (Fig.23).

```
Model Performance:
Random Forest - Accuracy: 0.960, AUC: 0.969
SVM - Accuracy: 0.997, AUC: 0.997
```

*Figure.19:Model Performance*

| Model | Accuracy | AUC | Precision | Recall | F1-Score |
|---|---|---|---|---|---|
| Random Forest | 0.96 | 0.969 | 0.96 | 0.96 | 0.96 |
| SVM | 0.997 | 0.997 | 1 | 1 | 1 |

*Figure.20:Comparison of Model Performance (Accuracy, AUC, Precision, Recall, and F1-Score) for Random Forest and SVM*

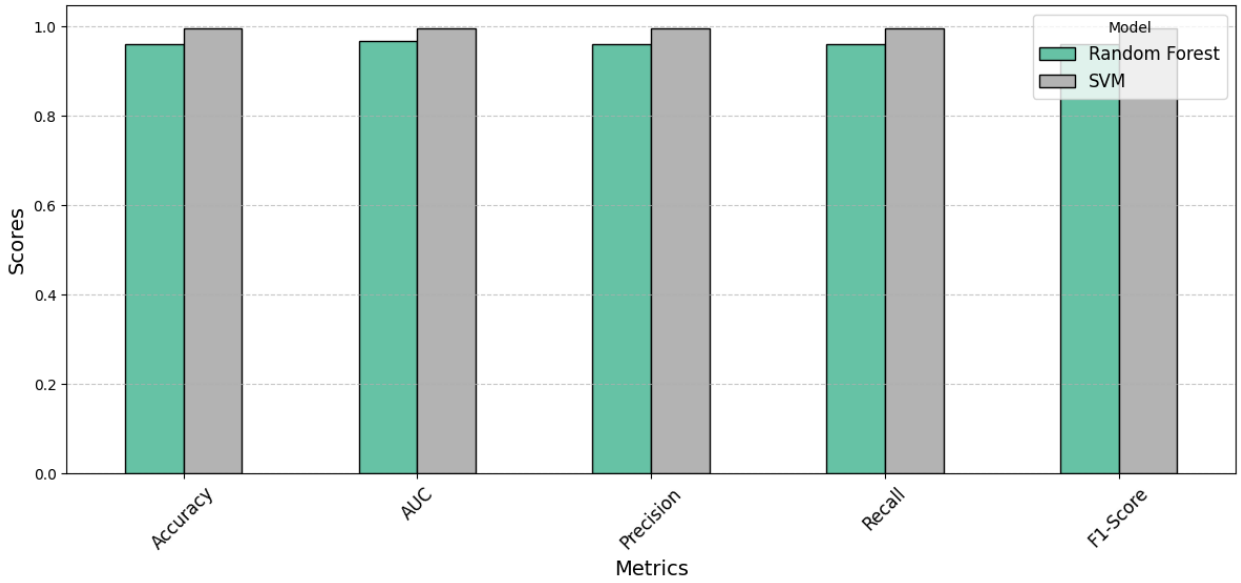


*Figure.21:Model Performance Comparison*

```
Random Forest - Testing Accuracy: 0.960, AUC: 0.969
SVM - Testing Accuracy: 0.997, AUC: 0.997
Random Forest - Training Accuracy: 1.000, AUC: 1.000
SVM - Training Accuracy: 0.999, AUC: 0.999
```

*Figure.22:Training and Testing Performance Comparison: SVM vs. Random Forest*

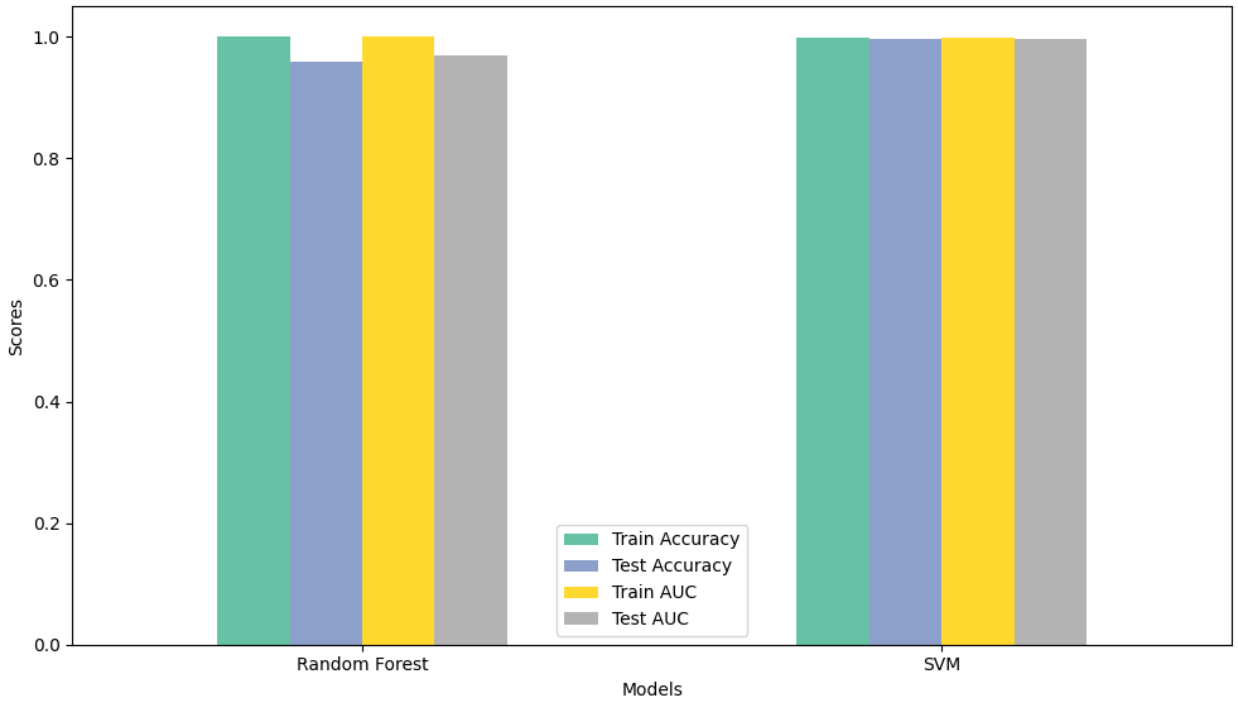


*Figure.23:Training vs Testing : Model comparison*

## 4.4 Critical Review

***Model Performance and Suitability:*** While both models performed well, the SVM's high accuracy and robustness make it more suitable for this task. SVM's high performance, particularly in complex, high-dimensional data, suggests it would generalize well to similar datasets.

***Feature Selection and Engineering:*** Feature engineering, including clustering and dimensionality reduction, played a vital role in enhancing model performance. These techniques simplified the data while retaining essential variance, improving both Random Forest and SVM results.

***Opportunities for Improvement and Recommendations:*** Despite the models' high performance, some limitations persist. Future work could explore the impact of additional feature engineering techniques to enhance the model's ability to capture more complex patterns in the data. Alternative clustering methods may also provide valuable insights and improve the model's ability to handle diverse datasets. Furthermore, investigating other classification models, such as gradient-boosting algorithms, could offer further improvements, potentially increasing classification accuracy and reliability when applied to similar datasets.

# 5.Conclusions

Experiments in both regression and classification showed strong model performance in predicting metal part lifespans. The Random Forest Regressor outperformed XGBoost, with lower MSE, RMSE, and nearly perfect $R^2$ scores. The Support Vector Machine (SVM) achieved near-perfect accuracy and AUC, effectively classifying parts based on durability.

***Model Recommendation***

The SVM model is recommended for deployment. While regression models predicted lifespan accurately, the SVM better meets operational needs by classifying parts as suitable or defective, simplifying decision-making.

**Justification**

SVM's classification simplifies quality control, reduces downtime, and minimizes production costs. Its high accuracy and AUC ensure reliability, and classification offers actionable insights more aligned with business needs than regression. SVM is the most suitable model, ensuring proactive quality management.

# 6.References


Liaw, A. and Wiener, M. (2002) 'Classification and regression by randomForest' R News, 2(3), pp.18-22. Available at: https://journal.r-project.org/articles/RN-2002-022/RN-2002-022.pdf

Chen, T. and Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794). ACM.Available at: https://dl.acm.org/doi/10.1145/2939672.2939785

Jolliffe,I.T.(2002) Principal Component Analysis. 2nd ed. New York, Springer-Verlag.

Zou, H. and Hastie, T. (2005) 'Regularization and variable selection via the elastic net', Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67(2), pp.301-320.Available at: https://academic.oup.com/jrsssb/article/67/2/301/7109482