COMP1801 - Machine Learning Coursework Report

Ozge Serap Elibol – 001411005

Word Count: 3249

Table of Contents

Table of Figures

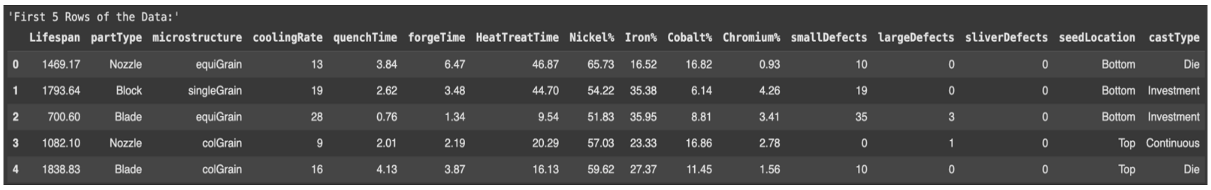
1. Executive Summary

This report presents the implementation of various machine learning models to address a challenge faced in the production of metal.
The dataset provided includes processing parameters and measurements from completed parts. Regression models were implemented to predict part lifespan, which are Random Forest Regression achieving the highest performance R²: 0.9368, while Polynomial Regression achived R²: 0.7738. For classification tasks, Gradient Boosting, Neural Networks and Decision Tree Classifier were evaluated to determine whether a part meets the company's minimum threshold of 1500 hours of lifespan. Gradient Boosting achieved higher performance with an accuracy of 91.50% and an F1-score of 0.85, while Neural Networks achieved an accuracy of 85.5% and an F1-score of 0.77.
The outputs recommend choosing Random Forest Regression for lifespan prediction and Gradient Boosting for classification. These models offer accurate, efficient, and interpretable solutions to improve production processes.

2. Data Exploration

The dataset was loaded using the pandas library's read_csv() function from a CSV file hosted on a GitHub repository. Loading the data from public GitHub repository guarantees that everyone have access to the data.

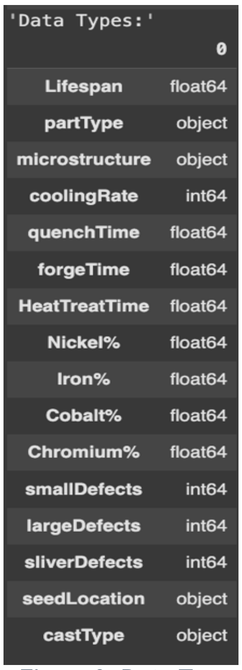
The dataset has the target variable, which is Lifespan, categorical features and numerical features.


'First 5 Rows of the Data:'

	Lifespan	partType	microstructure	coolingRate	quenchTime	forgeTime	HeatTreatTime	Nickel%	Iron%	Cobalt%	Chromium%	smallDefects	largeDefects	sliverDefects	seedLocation	castType
0	1469.17	Nozzle	equiGrain	13	3.84	6.47	46.87	65.73	16.52	16.82	0.93	10	0	0	Bottom	Die
1	1793.64	Block	singleGrain	19	2.62	3.48	44.70	54.22	35.38	6.14	4.26	19	0	0	Bottom	Investment
2	700.60	Blade	equiGrain	28	0.76	1.34	9.54	51.83	35.95	8.81	3.41	35	3	0	Bottom	Investment
3	1082.10	Nozzle	colGrain	9	2.01	2.19	20.29	57.03	23.33	16.86	2.78	0	1	0	Top	Continuous
4	1838.83	Blade	colGrain	16	4.13	3.87	16.13	59.62	27.37	11.45	1.56	10	0	0	Top	Die

Figure 1: First 5 Row of The Data

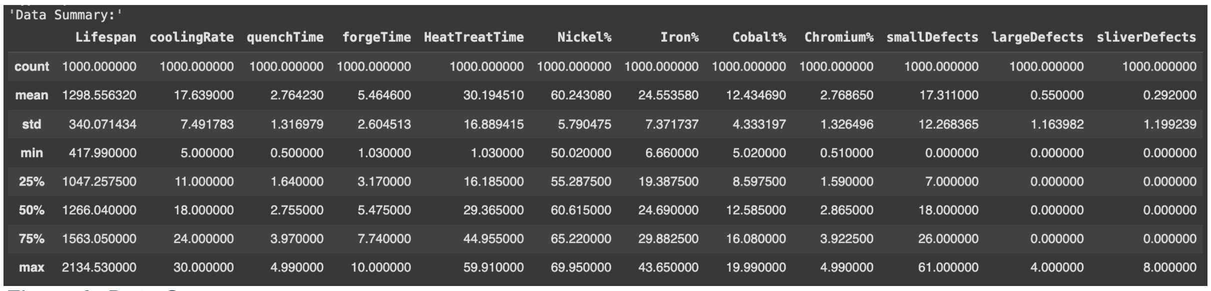
Figure 1 offers a quick overview of the dataset structure for further analysis.


'Data Types:'

	0
Lifespan	float64
partType	object
microstructure	object
coolingRate	int64
quenchTime	float64
forgeTime	float64
HeatTreatTime	float64
Nickel%	float64
Iron%	float64
Cobalt%	float64
Chromium%	float64
smallDefects	int64
largeDefects	int64
sliverDefects	int64
seedLocation	object
castType	object

Figure 2: Data Types

Figure 2 demonstrates that the dataset includes numerical (e.g., float64, int64) and categorical (object) variables.


'Data Summary:'

	Lifespan	coolingRate	quenchTime	forgeTime	HeatTreatTime	Nickel%	Iron%	Cobalt%	Chromium%	smallDefects	largeDefects	sliverDefects
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	1298.556320	17.639000	2.764230	5.464600	30.194510	60.243080	24.553580	12.434690	2.768650	17.311000	0.550000	0.292000
std	340.071434	7.491783	1.316979	2.604513	16.889415	5.790475	7.371737	4.333197	1.326496	12.268365	1.163982	1.199239
min	417.990000	5.000000	0.500000	1.030000	1.030000	50.020000	6.660000	5.020000	0.510000	0.000000	0.000000	0.000000
25%	1047.257500	11.000000	1.640000	3.170000	16.185000	55.287500	19.387500	8.597500	1.590000	7.000000	0.000000	0.000000
50%	1266.040000	18.000000	2.755000	5.475000	29.365000	60.615000	24.690000	12.585000	2.865000	18.000000	0.000000	0.000000
75%	1563.050000	24.000000	3.970000	7.740000	44.955000	65.220000	29.882500	16.080000	3.922500	26.000000	0.000000	0.000000
max	2134.530000	30.000000	4.990000	10.000000	59.910000	69.950000	43.650000	19.990000	4.990000	61.000000	4.000000	8.000000

Figure 3: Data Summary

Figure 3 indicates the Lifespan, averages 1298.56 hours with a range of 418 to 2134.53 and a standard deviation of 340.07, showing variability effected by multiple factors. Features like coolingRate and HeatTreatTime also show variability, highlighting the need for standardization.

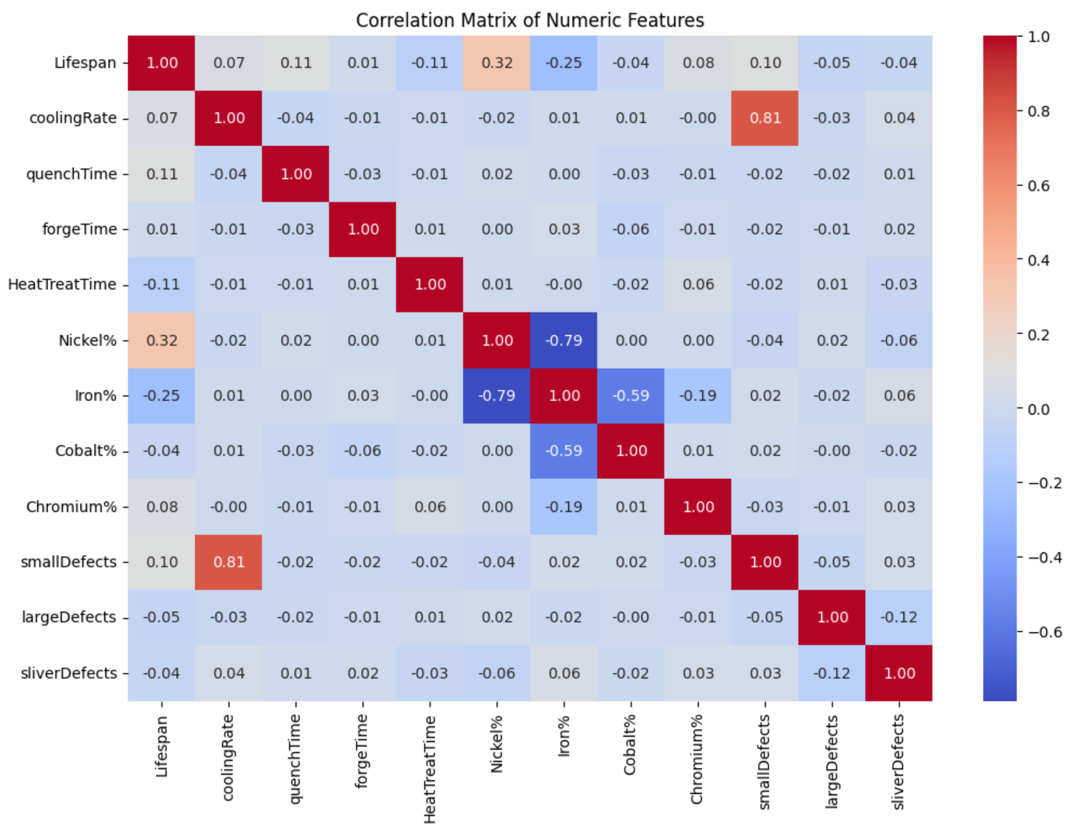


Figure 4: Correlation Matrix of Numeric Features

Figure 4 above was generated to show correlations between features, clearing some key relationships:

1. Target Variable (Lifespan):

- Nickel% (0.32) and smallDefects (0.10) are positively correlated.
- Iron% (-0.25) and HeatTreatTime (-0.11) are negatively correlated.

2. Feature Relationships:

- smallDefects and coolingRate (0.81) have strong positive correlation.
- Nickel% and Iron% (-0.79) have strong negative correlation.

Correlation-based feature selection removes irrelevant features, increasing prediction accuracy, leading to lower MSE and higher R² scores.

It was observed that the effects of the variables smallDefects, largeDefects and sliverDefects were not very high on their own and a new feature named total_defects was added to analyze the total effect of these variables on the target variable by performing a feature engineering. According to the analysis results, the total_defects variable did not have a significant effect on the target variable. In summary, it shows that the total number of defects is more meaningful than the individual defect types, but it still does not have much effect to be included in the modeling process.

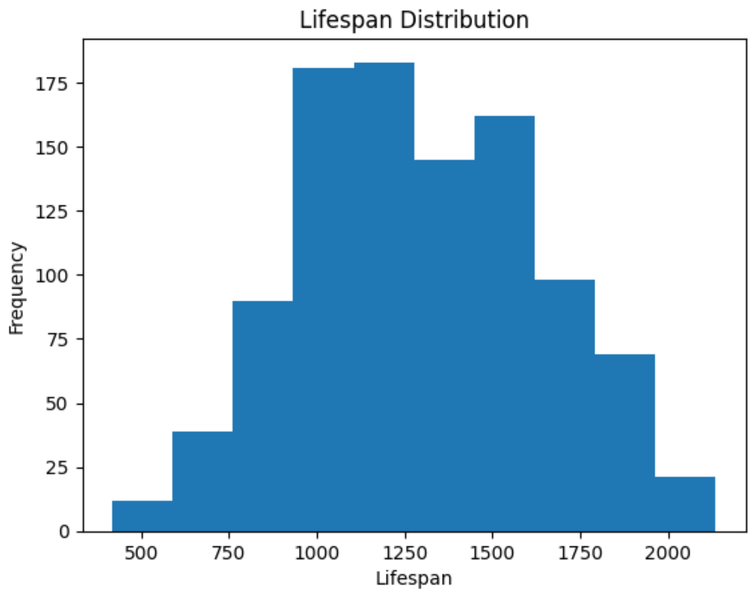


Figure 5: Lifespan Distribution

Figure 5 *was generated to* shows almost a symmetric Lifespan distribution, with most values between 1000 and 1600 hours, ranging from 500 to 2100. Its near-normal shape supports models handling linear and nonlinear patterns.

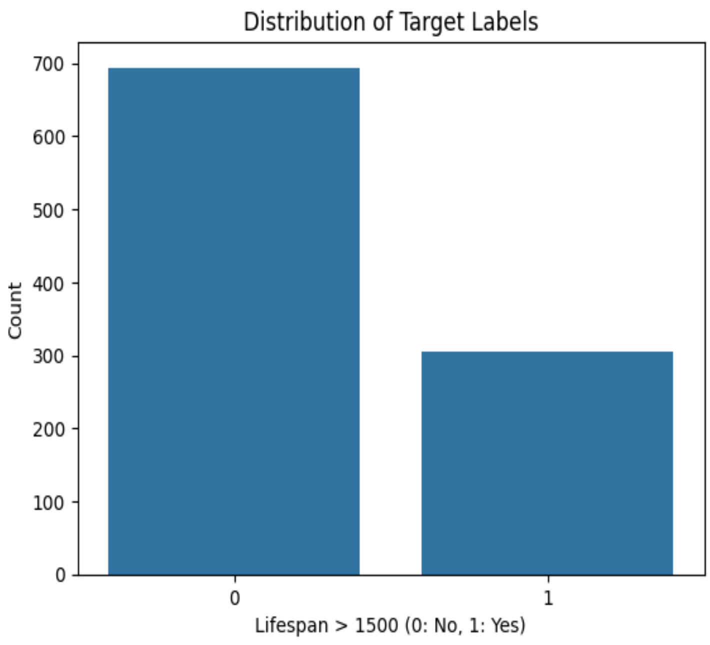


Figure 6: Distribution of Target Labels

Figure 6 was generated to visualize binary target, with 0 values which are much more than 1 values. This shows that most samples are defective, highlighting the need to address class imbalance. Handling imbalance in models like Gradient Boosting and Neural Networks is crucial to avoid majority class bias. Techniques like SMOTE, or class weighting can be used in the next steps.

Feature Selection

In the feature selection process, correlation analysis and other data exploration techniques are used:

- Key numerical features included Nickel% (has positive impact on Lifespan) and Iron% (has negative impact on Lifespan). Parameters like HeatTreatTime and forgeTime were chosen for their influence on microstructure and strength.
- Categorical features were One-Hot Encoded for model compatibility.
- Numerical features were standardized for consistent scaling.
- Features (e.g., Chromium%, quenchTime) which has less impact were excluded for their weak correlation with lifespan.

Discussion on Expected Approach and Model Suitability

From the initial analysis, Random Forest was expected to work well for predicting lifespan because it can handle complex relationships and different types of data. For classification, Neural Networks were thought to perform best due to their ability to find patterns and connections in complex data.

3. Regression Implementation

3.1 Methodology

3.1.1 Model Selection and Justification

1. Polynomial Regression

The histogram and scatter plots have indications that there are non-linear relationships in the dataset. Polynomial regression is well-suited to model these trends due to its ability to effectively capture non-linear relationships (Peckov, 2024). Furthermore, Polynomial Regression offers a moderate flexibility of shapes and interpretability, making it ideal for explaining how individual features affect predictions, which is particularly crucial in industrial applications (Peckov, 2024).

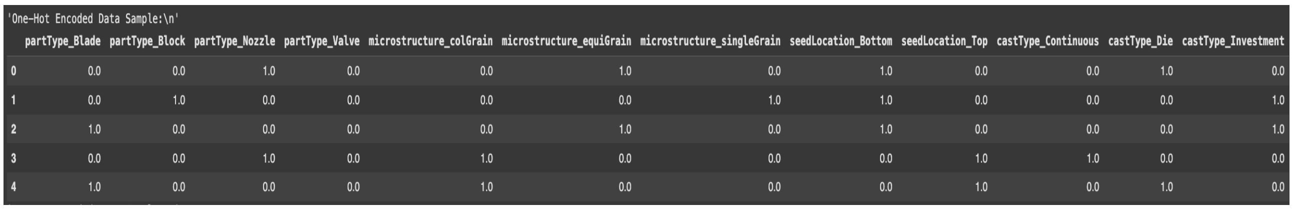
2. Random Forest Regression

Random forests are effective at modeling nonlinear dependencies, such as those observed between Nickel%, Iron%, smallDefects, and Lifespan. Because of the Law of Large Numbers, they do not overfit and are resistant to noise and outliers, which enhances their ability to generalize well to unseen data (Breiman, 2001). Injecting the right kind of randomness makes them accurate classifiers and regressors, while the

ensemble approach ensures robustness in handling complex relationships (Breiman, 2001).
Random Forest's ensemble approach ensures robustness and adaptability, resulting in accurate predictions even when the dataset has various scales and complex feature types.

3.1.2 Data Preprocessing

Categorical features were converted to binary columns using One-Hot Encoding as shown in Figure 7:

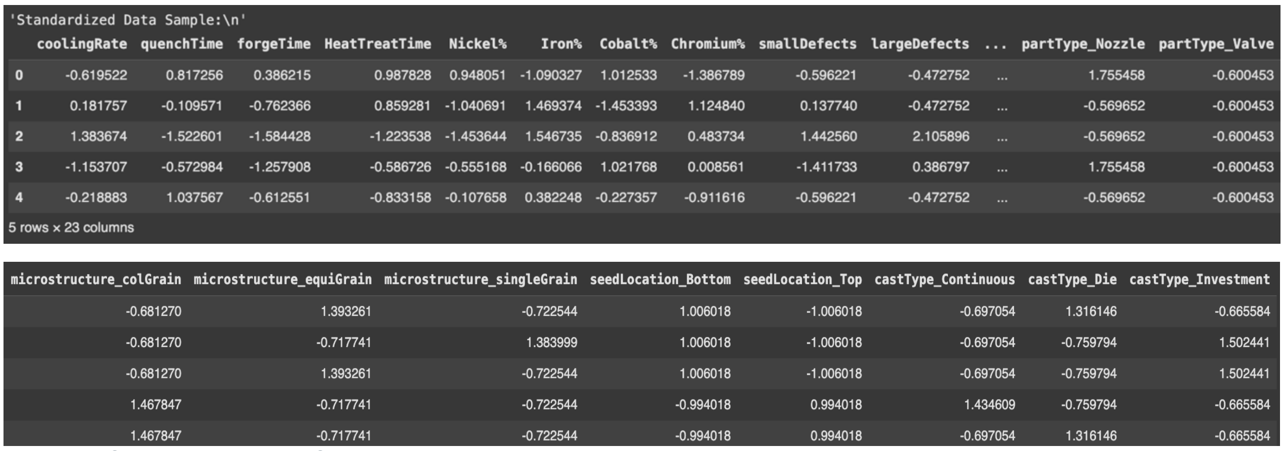


'One-Hot Encoded Data Sample:\n'

	partType_Blade	partType_Block	partType_Nozzle	partType_Valve	microstructure_colGrain	microstructure_equiGrain	microstructure_singleGrain	seedLocation_Bottom	seedLocation_Top	castType_Continuous	castType_Die	castType_Investment
0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0
1	0.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0
2	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0
3	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0
4	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0

Figure 7:One-Hot Encoded Data Sample

To ensure equal contribution from all features and to avoid bias due to different scales, numerical features were standardized using StandardScaler which has an output as shown in Figure 8. The data were transformed to have a mean of 0 and a standard deviation of 1.



'Standardized Data Sample:\n'

	coolingRate	quenchTime	forgeTime	HeatTreatTime	Nickel%	Iron%	Cobalt%	Chromium%	smallDefects	largeDefects	...	partType_Nozzle	partType_Valve
0	-0.619522	0.817256	0.386215	0.987828	0.948051	-1.090327	1.012533	-1.386789	-0.596221	-0.472752	...	1.755458	-0.600453
1	0.181757	-0.109571	-0.762366	0.859281	-1.040691	1.469374	-1.453393	1.124840	0.137740	-0.472752	...	-0.569652	-0.600453
2	1.383674	-1.522601	-1.584428	-1.223538	-1.453644	1.546735	-0.836912	0.483734	1.442560	2.105896	...	-0.569652	-0.600453
3	-1.153707	-0.572984	-1.257908	-0.586726	-0.555168	-0.166066	1.021768	0.008561	-1.411733	0.386797	...	1.755458	-0.600453
4	-0.218883	1.037567	-0.612551	-0.833158	-0.107658	0.382248	-0.227357	-0.911616	-0.596221	-0.472752	...	-0.569652	-0.600453

5 rows × 23 columns

microstructure_colGrain	microstructure_equiGrain	microstructure_singleGrain	seedLocation_Bottom	seedLocation_Top	castType_Continuous	castType_Die	castType_Investment
-0.681270	1.393261	-0.722544	1.006018	-1.006018	-0.697054	1.316146	-0.665584
-0.681270	-0.717741	1.383999	1.006018	-1.006018	-0.697054	-0.759794	1.502441
-0.681270	1.393261	-0.722544	1.006018	-1.006018	-0.697054	-0.759794	1.502441
1.467847	-0.717741	-0.722544	-0.994018	0.994018	1.434609	-0.759794	-0.665584
1.467847	-0.717741	-0.722544	-0.994018	0.994018	-0.697054	1.316146	-0.665584

Figure 8: Standardized Data Sample

The dataset was split into training, validation, and test sets to evaluate model performance accurately:

- 80-20 split for training and test sets. Further, the training set was split into 60% training and 20% validation for hyperparameter tuning and performance evaluation.
- Shuffling was applied to ensure that the splits were representative of the overall data distribution.

After performing the splitting process, code cell generated an output, to be sure the correctness of the data division process.

3.1.3 Hyperparameter Tuning Framework

Polynomial Regression: For the polynomial regression model, the main hyperparameter is the degree, which controls model complexity by setting the highest power of input features. Degrees 1, 2, 3 were tested to find the best balance. Given the small hyperparameter space, testing each degree with a for loop was feasible, and their performance was compared to determine the optimal value.

Random Forest Regressor: The Random Forest model's hyperparameters were tuned using GridSearchCV, which evaluates combinations via cross-validation to minimize validation error. This resulted in robust model for predicting metal part life.

Hyperparameters:

- n_estimators: The number of trees in the forest. Values tested were 50, 100, 200.
- max_depth: The maximum depth of each tree, with values None, 10, 20, 30 tested. The value None allows the nodes to expand until all leaves are pure or contain fewer than the minimum number of samples for splitting.
- min_samples_split: The minimum number of samples required to split an internal node, with values 2, 5, 10.
- min_samples_leaf: The minimum number of samples required to be at a leaf node, with values 1, 2, 4.

3.2 Evaluation

Polynomial Regression Performance:
To find the optimal degree for the polynomial regression model, degrees from 1 to 3 were tested by using a for loop. Each degree was evaluated using Mean Squared Error (MSE) and R^2 Score on the training and validation datasets.

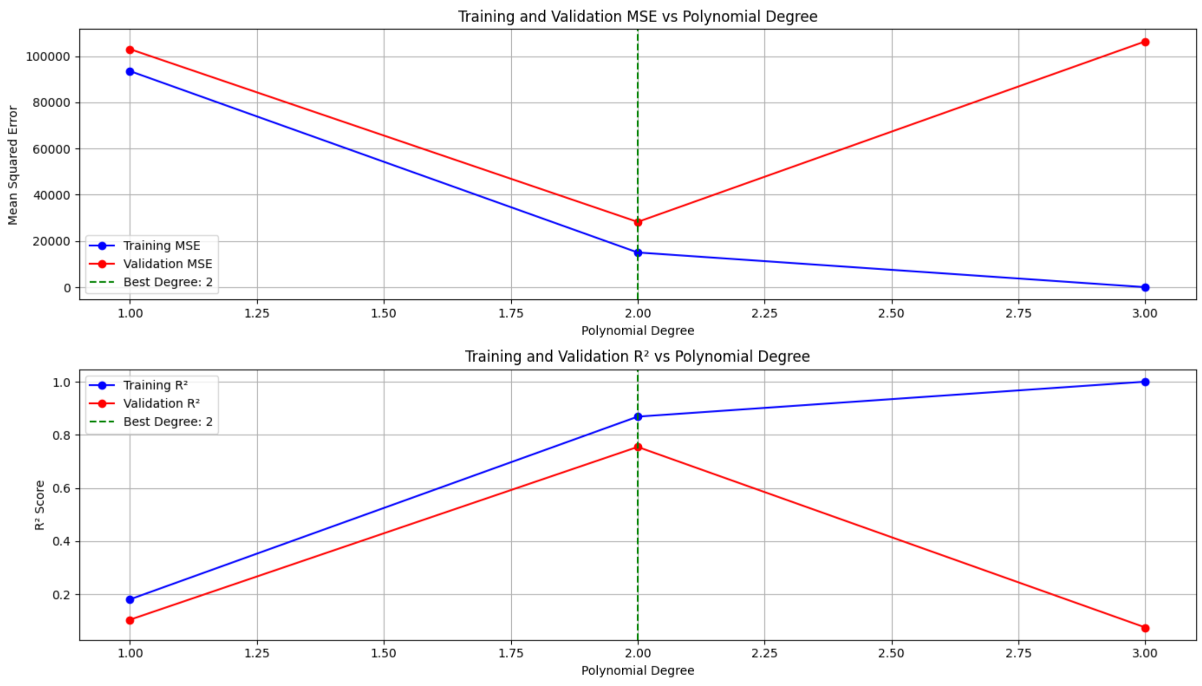


Figure 9: Comparision between Polynomail degree and MSE/R² scores

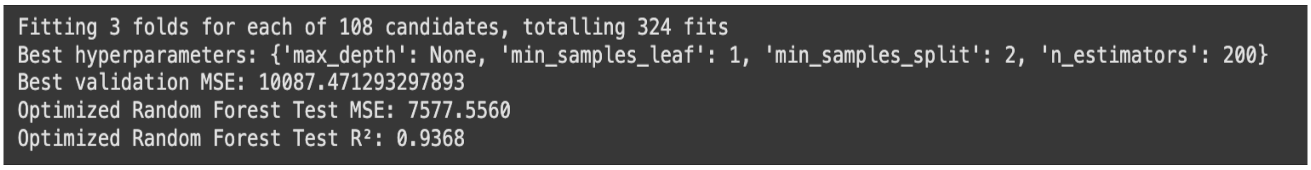
Figure 9 was generated to demonstrate the relationship between the polynomial degree and the MSE & R² for both the training and validation datasets. Key observations are as follows:

Degree 1: The model underperformed on both training and validation sets, and this is a sign of underfitting. The training R² score was 0.1810, and the validation R² score was 0.1044. The training MSE was 102,341.45, and the validation MSE was 98,452.12.

Degree 2: The model performed well on both training and validation sets, achieving the best balance between complexity and generalization. The training MSE was 15018.5812, and the validation MSE was 27142.1366. The training R² score was 0.8685, and the validation R² score was 0.7544. These results indicate that the model captures the key patterns in the data without overfitting.

Degree 3: Although it perfectly fit the training set (R²: 1.0000), it showed a sharp performance drop on the validation set (R²: 0.0750), indicating overfitting.

Random Forest Performance:



```
Fitting 3 folds for each of 108 candidates, totalling 324 fits
Best hyperparameters: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}
Best validation MSE: 10087.471293297893
Optimized Random Forest Test MSE: 7577.5560
Optimized Random Forest Test R²: 0.9368
```

Figure 10: Results of Random Forest Regressor

GridSearchCV was used for hyperparameter tuning of the Random Forest model. Totally, 108 combinations were tested with 3-fold cross-validation, resulting in 324 fits.

Random Forest Regression results as shown in Figure 10:
The best hyperparameters:

- n_estimators: 200
- max_depth: None
- min_samples_split: 2
- min_samples_leaf: 1

Final Model Evaluation:

- Test MSE: 7,577.56
- Test R^2: 0.9368

These outputs show that the optimized Random Forest model generalizes well to unseen data. The Test MSE is significantly lower than the validation error, and the Test R^2 score of 0.9368 shows that the model explains 93.68% of the variance in the target variable.

Final Model Comparison and Recommendation
Summary of Results

- Polynomial Regression:
    - Test MSE: 27142.1366
    - Test R^2: 0.7738

- Random Forest Regression:
    - Test MSE: 7577.56
    - Test R^2: 0.9368

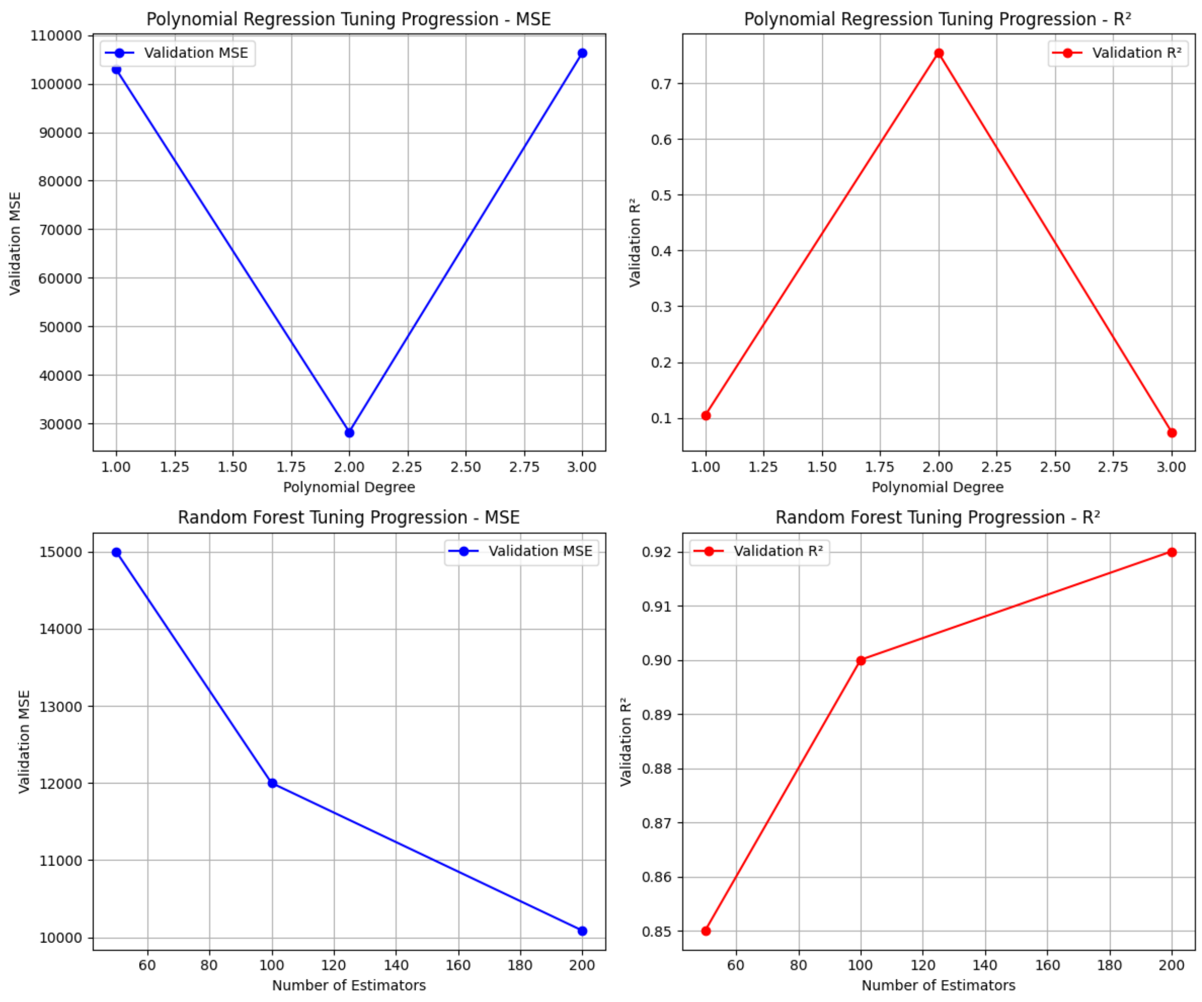


Figure 11: Polynomial Regression and Random Forest Tuning Results

For polynomial regression, the validation MSE minimizes at Degree 2, confirming it as the optimal degree, while Degree 3 shows overfitting. The validation R² also peaks at Degree 2, further supporting it as the best choice. For random forest, the validation MSE decreases with more estimators, reaching the lowest at 200, while the validation R² increases consistently, achieving the highest score at 200 estimators which was generated to show in Figure 11. The Random Forest Regression model is recommended due to its superior performance and reliability.

3.3 Critical Review

This methodology demonstrated strengths such as comprehensive hyperparameter tuning for Random Forest with GridSearchCV, effective preprocessing with appropriate feature selection and scaling, and fair comparisons through consistent evaluation metrics and train-test splits. However, Polynomial Regression showed high levels of overfitting, which could have been mitigated with regularization, and cross-validation could have provided a more robust degree selection. Although Random Forest delivered relatively good results, interpretability could be improved using tools like SHAP. Future work could explore models like XGBoost for better efficiency, regularized regression to balance complexity and interpretability.

4. Classification Implementation

4.1 Feature Crafting

To determine whether a metal part is defective, a binary feature 1500_labels was created. Parts with a lifespan greater than 1500 hours were labeled as 1 (not defective), while parts with a lifespan of 1500 hours or less were labeled as 0 (defective) which was generated in Figure 12. This threshold is based on company standards for acceptable lifespan.

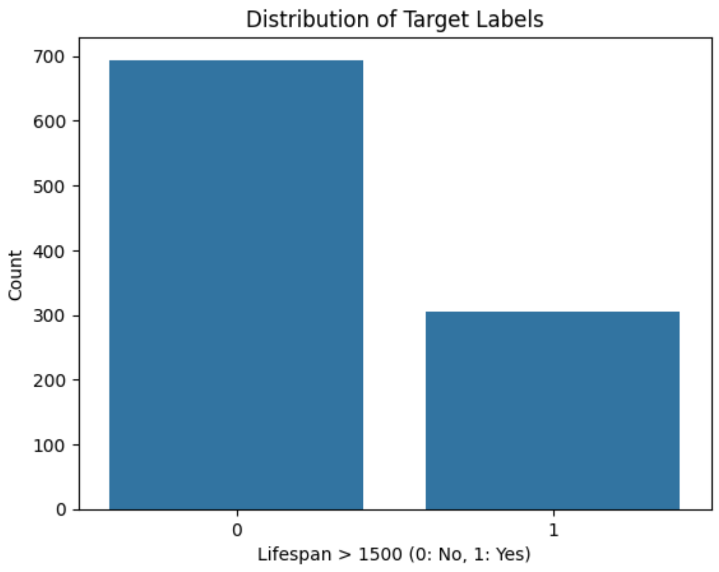


Figure 12: Distribution of Target Labels

4.2 Methodology

4.2.1 Model Selection and Justification

Neural Networks were chosen for their flexibility in modeling complex, non-linear interactions, such as material composition and defects. Given the dataset's diverse features, ranging from categorical to numerical, Neural Networks are highly effective at capturing complex relationships. Particularly, recent research has showed that Neural Networks are highly effective in detecting defects in composite materials, successfully handling complex application scenarios (Research on Defect Detection Method, 2024). A model includes multiple hidden layers and uses L2 regularization and Dropout to prevent overfitting, making it suitable for exploring intricate feature patterns and expected to deliver successful results.

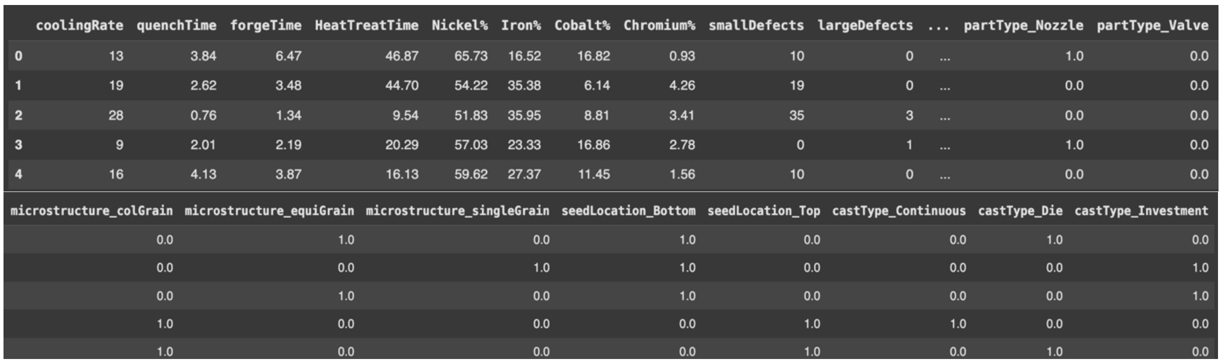
Gradient Boosting was selected for its effectiveness in capturing non-linear relationships and feature interactions. Its robustness to imbalanced data and ability to generate feature importance scores made it valuable. Recent studies have shown that

Gradient Boosting models effectively manage imbalanced datasets and provide interpretable feature importance, which aids in identifying critical factors in predictive tasks, making them well-suited for defect prediction in manufacturing environments (Chen & Guestrin, 2016).

Decision Tree Classifier was implemented as a benchmark model, with its depth tuned to prevent overfitting while maintaining generalizability. Although it was tested, the decision tree was not included in the final report.

4.2.2 Data Preprocessing

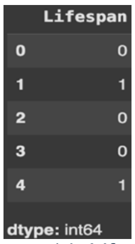
Categorical features were transformed with One-Hot Encoding. Using drop_first=True reduced multicollinearity by removing redundant columns, helping to mitigate overfitting risks, and displayed the first 5 rows of encoded data, as shown in Figure 13 & Figure 14.



	coolingRate	quenchTime	forgeTime	HeatTreatTime	Nickel%	Iron%	Cobalt%	Chromium%	smallDefects	largeDefects	...	partType_Nozzle	partType_Valve
0	13	3.84	6.47	46.87	65.73	16.52	16.82	0.93	10	0	...	1.0	0.0
1	19	2.62	3.48	44.70	54.22	35.38	6.14	4.26	19	0	...	0.0	0.0
2	28	0.76	1.34	9.54	51.83	35.95	8.81	3.41	35	3	...	0.0	0.0
3	9	2.01	2.19	20.29	57.03	23.33	16.86	2.78	0	1	...	1.0	0.0
4	16	4.13	3.87	16.13	59.62	27.37	11.45	1.56	10	0	...	0.0	0.0

microstructure_colGrain	microstructure_equiGrain	microstructure_singleGrain	seedLocation_Bottom	seedLocation_Top	castType_Continuous	castType_Die	castType_Investment
0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0
0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0
0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0
1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0
1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0

Figure 13: Encoded Data



	Lifespan
0	0
1	1
2	0
3	0
4	1

dtype: int64

Figure 14: Lifespan

To ensure equal contribution from all features and to avoid bias due to different scales, numerical features were standardized using StandardScaler.

To ensure that sufficient data was available for training, fine-tuning, and evaluation purposes the dataset split was managed with 60%/20%/20% train-validation-test. We excluded the Lifespan column to avoid data leakage and ensure reliability of our model. Various methods were implemented to tackle the class imbalance with the kind of model:

• Gradient Boosting SMOTE (under-sampling the majority class by synthesizing from minority)

• Neural Networks: Class weights were applied during training, giving more importance to the minority class to ensure effective learning from both classes without bias towards the majority, because when the class weights are calculated, it is observed that: class weights: {0: 0.7075471698, 1: 1.7045454545}.

4.2.3 Hyperparameter Tuning Framework

Gradient Boosting (GB) Hyperparameters

1. **Number of Estimators (n_estimators)**: It controls the number of boosting stages to perform. Selecting a high value may improve the model's performance. A range of [50, 100, 150] was chosen to balance these considerations.
2. **Learning Rate (learning_rate)**: It determines the contribution of each tree to the final prediction. Smaller values make the model learn more slowly but can improve generalization. A range of [0.05, 0.1, 0.2] was tested.
3. **Maximum Depth (max_depth)**: This parameter controls the complexity of each decision tree by setting the maximum depth. Limiting depth prevents overfitting and ensures interpretability. Values of [3, 4, 5] were chosen as they typically balance bias and variance effectively.

   A GridSearchCV approach with 5-fold cross-validation was utilized to find the optimal combination of hyperparameters.

Neural Network Hyperparameters

1. **Learning Rate:** The model was tuned using the Adam optimizer for stable convergence, with a grid search over values [0.01, 0.001, 0.0001, 0.0005]. The optimal value, 0.001, balanced speed and precision in weight updates. Firstly SGD was tried, but because of unsatisfactory results, Adam is applied due to its, more efficient learning. This transition improved the model's learning process and overall performance.

2. **Dropout Rate:** Dropout rates of [0.2, 0.3, 0.4] were tested, with 30% selected to mitigate overfitting by randomly deactivating neurons during training. This regularization technique enhanced generalization.

3. **L2 Regularization:** L2 regularization was applied to the Dense layers with a penalty term of 0.01 to discourage overly complex weights. This helped reduce overfitting and improved validation performance.

4. **Layer Sizes and Depth:** The architecture was tuned with two to three hidden layers and units ranging from [10, 20, 50] per layer. The final configuration chosen was two hidden layers with 20 and 10 neurons, respectively.

5. **Batch Size and Epochs:** A batch size of 100 and early stopping (with 10 patience epochs) were used to prevent overtraining. This approach effectively ensured a consistent decrease in validation loss over epochs.

4.3 Evaluation

Gradient Boosting

Tuning Experiments:

- Parameters Tuned: n_estimators, learning_rate, and max_depth.
- Approach: A grid search was used to evaluate combinations of these hyperparameters.

GridSearchCV Execution: The model was trained and validated on multiple combinations of n_estimators, learning_rate, and max_depth using 5-fold cross-validation. The performance was measured in terms of validation accuracy. The output revealed that the best parameters were:

- n_estimators: 150
- learning_rate: 0.1
- max_depth: 5

Validation Accuracy Results: With the optimal hyperparameters, the validation accuracy achieved was **90.50%**, indicating a well-balanced model.

Test Evaluation:

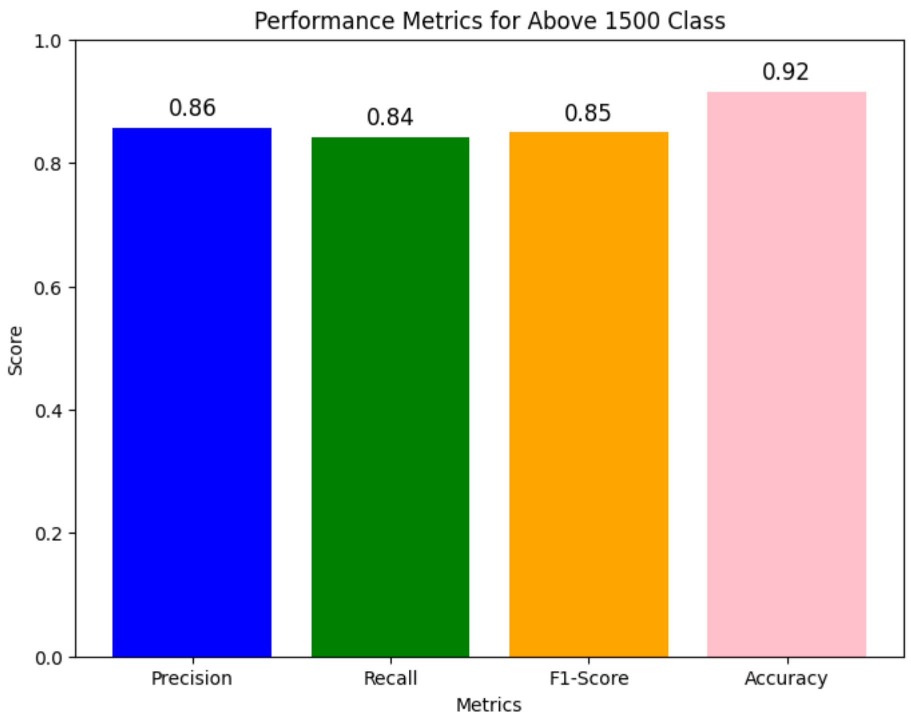


Figure 15: Performance Metrics

Figure 15 was evaluated on the test dataset, yielding the following results:

- **Test Accuracy**: 91.50%
- **Precision**: 86%
- **Recall**: 84%
- **F1-Score**: 85%

These metrics demonstrate that the model performs well in distinguishing between classes, especially for imbalanced datasets.

Confusion Matrix Analysis:

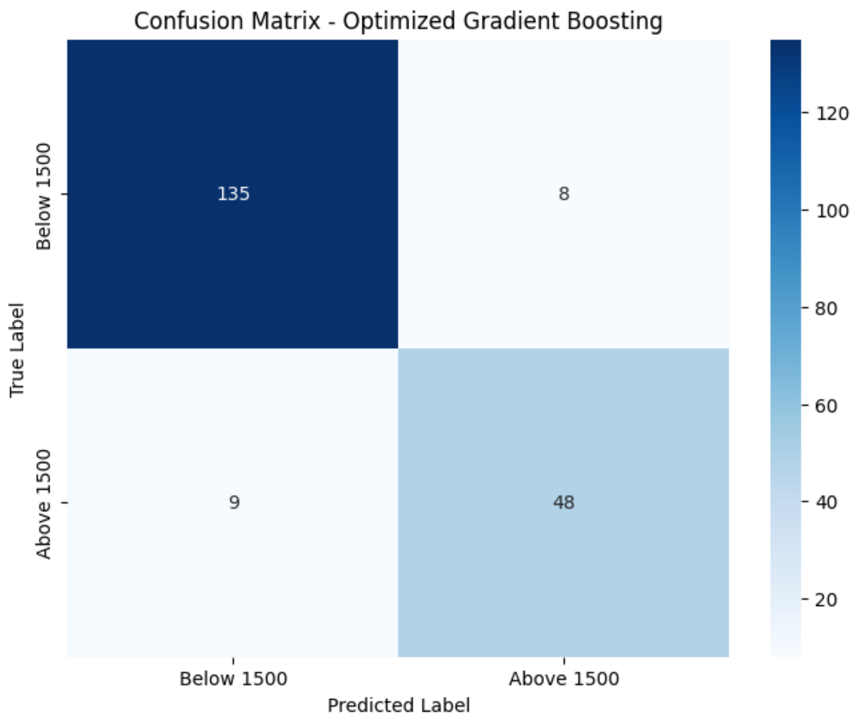
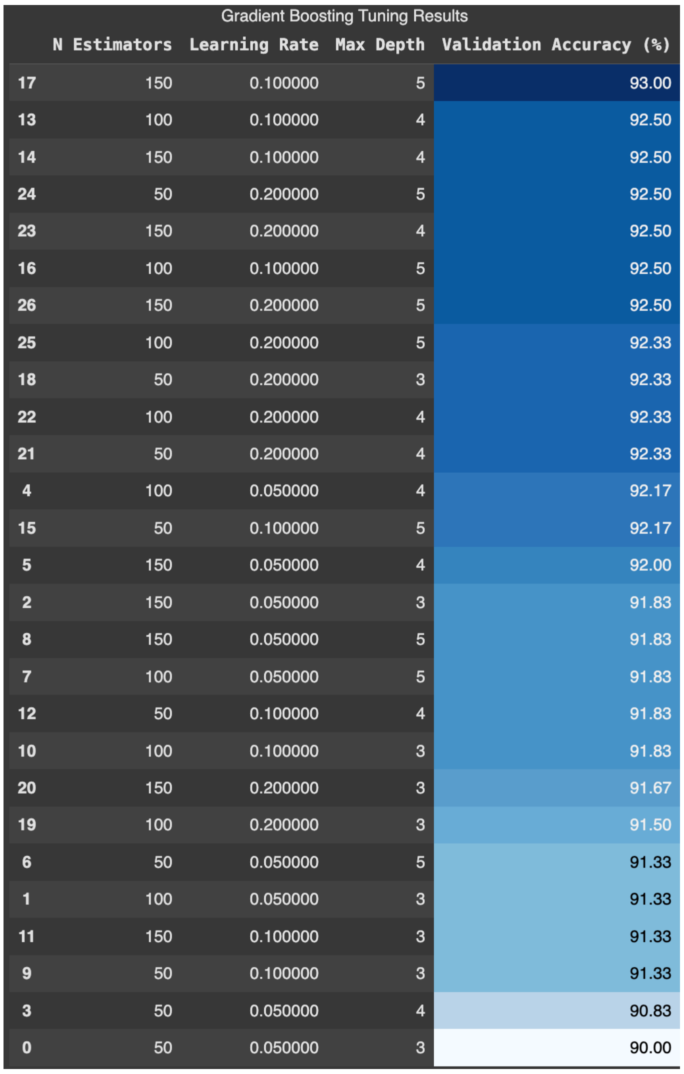


Figure 16: Confusion Matrix

Figure 16 was generated, showcasing the following:

- True Negatives: 135
- False Positives: 8
- False Negatives: 9
- True Positives: 48 This indicates a balanced performance, with only minor misclassifications in each class.

Visualization of Hyperparameter Results: Figure 18 was generated to show the validation accuracies for all tested hyperparameter combinations.



Gradient Boosting Tuning Results

	N Estimators	Learning Rate	Max Depth	Validation Accuracy (%)
17	150	0.100000	5	93.00
13	100	0.100000	4	92.50
14	150	0.100000	4	92.50
24	50	0.200000	5	92.50
23	150	0.200000	4	92.50
16	100	0.100000	5	92.50
26	150	0.200000	5	92.50
25	100	0.200000	5	92.33
18	50	0.200000	3	92.33
22	100	0.200000	4	92.33
21	50	0.200000	4	92.33
4	100	0.050000	4	92.17
15	50	0.100000	5	92.17
5	150	0.050000	4	92.00
2	150	0.050000	3	91.83
8	150	0.050000	5	91.83
7	100	0.050000	5	91.83
12	50	0.100000	4	91.83
10	100	0.100000	3	91.83
20	150	0.200000	3	91.67
19	100	0.200000	3	91.50
6	50	0.050000	5	91.33
1	100	0.050000	3	91.33
11	150	0.100000	3	91.33
9	50	0.100000	3	91.33
3	50	0.050000	4	90.83
0	50	0.050000	3	90.00

Figure 17: Tuning Results

Neural Network

Tuning Experiments:

- Parameters Tuned: Layer sizes, dropout rate, L2 regularization, and learning rate.
- Approach: Iterative testing of different architectures, leveraging early stopping to prevent overfitting.
- Best Configuration:
  - Two hidden layers with 20 and 10 units , respectively.
  - Dropout rate: 30% in the first layer.
  - L2 regularization: 0.01 for both layers.
  - Optimizer: Adam, learning rate 0.001.

Hyperparameters like Dropout Rate (30%) and L2 Regularization were manually tuned through experimentation to balance generalization and performance.

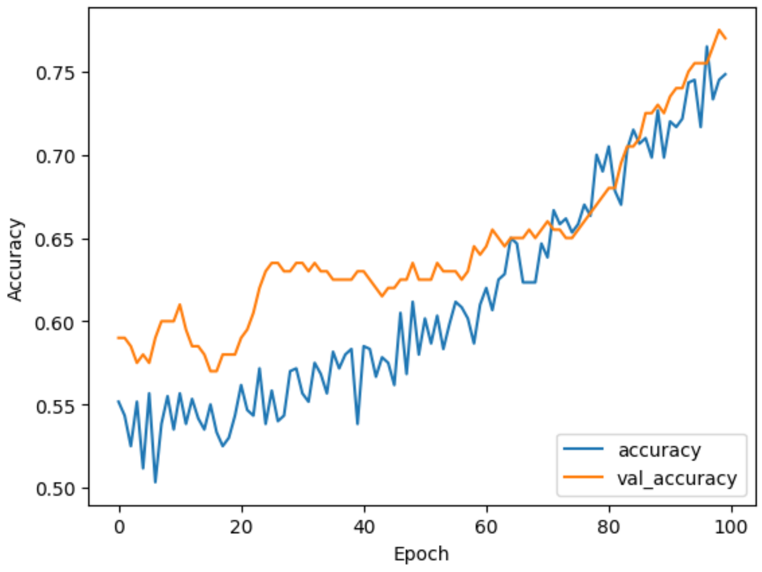


Figure 18: Results of Trying to Add Layers

Figure 19 shows an experiment with an **extended Neural Network**, where a **third hidden layer** was added, units were increased, dropout rates were adjusted for improved learning, and Adam optimizer was reduced from **to 0.0005** to improve convergence and prevent overshooting the optimal solution. Despite these changes, the gains did not surpass the initial model version, and **overfitting** was evident. Therefore, the simpler model was retained.

Performance Analysis

Training & Validation Accuracy:

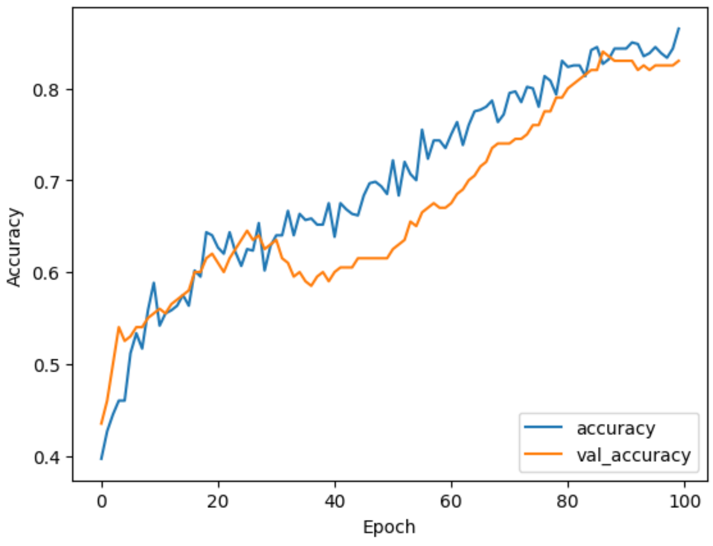


Figure 19: Neural Network Accuracy

Figure 20 shows training accuracy steadily improving to over 85%, with validation accuracy stabilizing at around 83%.

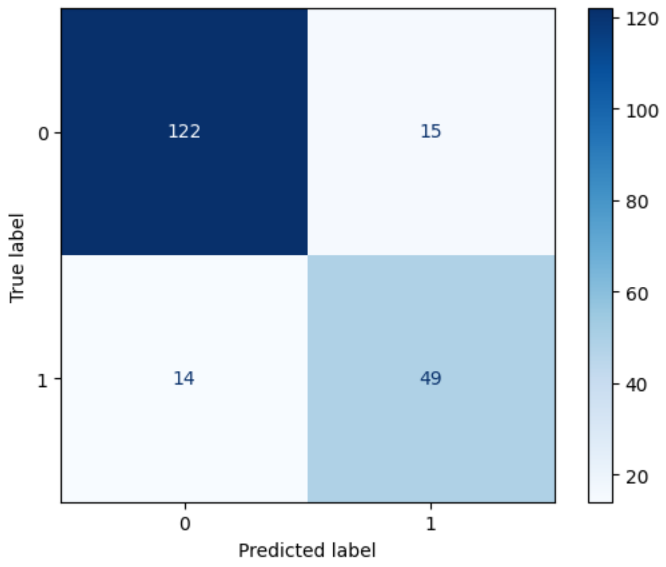


Figure 20: NN Confusion Matrix

Figure 21 was generated to indicaate that:

- True Negatives (TN): 122 - Correctly classified as non-defective.
- False Positives (FP): 15 - Overestimated defect presence.
- False Negatives (FN): 14 - Missed detecting defects, which can be costly.
- True Positives (TP): 49 - Correctly caught defective parts.

Final Evaluation on Test Set for NN:

- Binary Accuracy: 0.8336
- Cross-Entropy Loss: 0.4406
- Cross-Entropy Loss on Test Data: 0.434
- Accuracy on Test Data: 85.5%

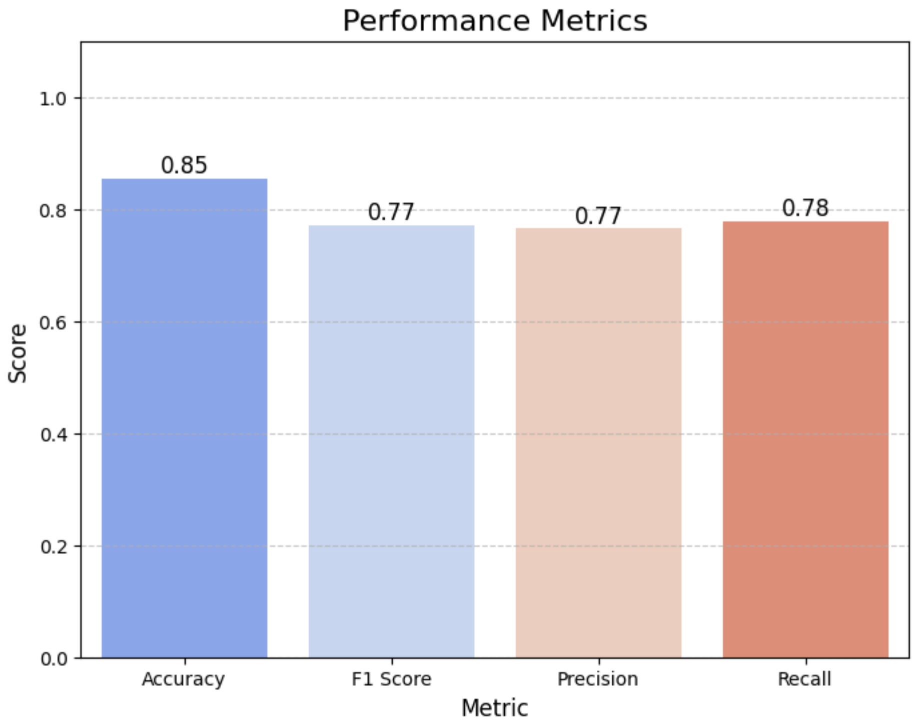
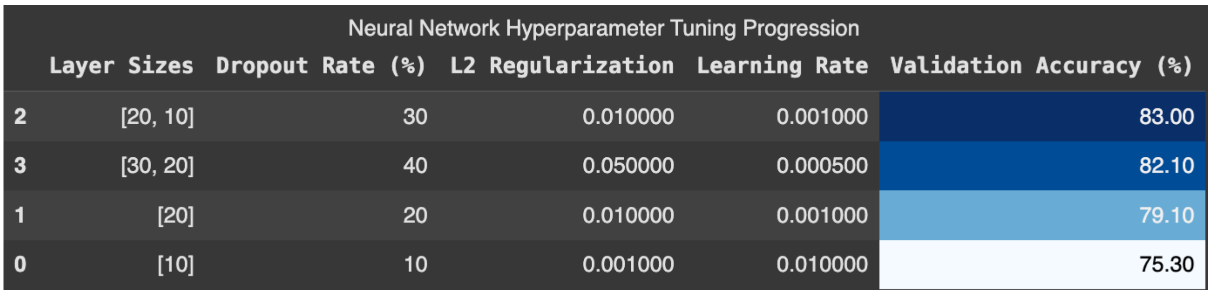


Figure 21: Performance Metrics For NN Model

Figure 22 was generated to express the performance metrics for Neural Network in the final model evaluation, developed during the coursework.

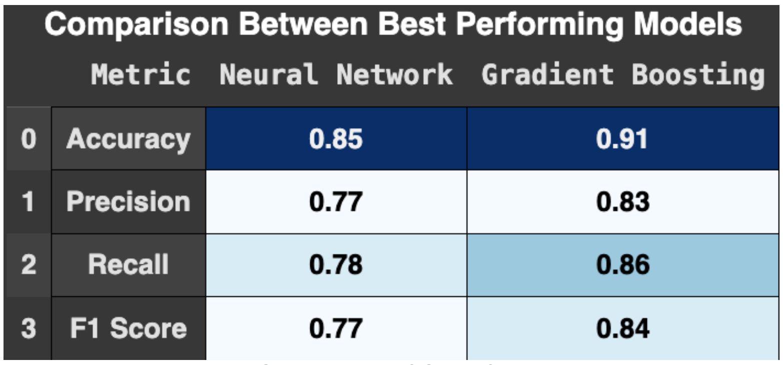


Neural Network Hyperparameter Tuning Progression

	Layer Sizes	Dropout Rate (%)	L2 Regularization	Learning Rate	Validation Accuracy (%)
2	[20, 10]	30	0.010000	0.001000	83.00
3	[30, 20]	40	0.050000	0.000500	82.10
1	[20]	20	0.010000	0.001000	79.10
0	[10]	10	0.001000	0.010000	75.30

Figure 22: Hyperparamter Tuning Progress for NN

Figure 23 shows the progression of hyperparameter tuning for the Neural Network. The best setup used two layers ([20, 10]), a 30% dropout rate, L2 regularization of 0.01, and a learning rate of 0.001, achieving 85% validation accuracy.


Comparison Between Best Performing Models

	Metric	Neural Network	Gradient Boosting
0	Accuracy	0.85	0.91
1	Precision	0.77	0.83
2	Recall	0.78	0.86
3	F1 Score	0.77	0.84

Figure 23: Comparison of Classification Models

As shown in Figure 24, Gradient Boosting outperformed the Neural Network across all metrics, making it more reliable in reducing misclassification risks. Therefore, Gradient Boosting is recommended for deployment due to its relatively high performance in predicting defective parts.

4.4 Critical Review

Strengths

- Balanced Methodology: Class weights solved class imbalance effectively, benefiting the minority "Above 1500" class.
- Comprehensive Evaluation: Metrics like accuracy, precision, recall, and F1-score were analyzed with visual support.

Areas for Improvement

- Hyperparameter Search Space: Expanding tuning for Neural Networks (layer sizes, dropout rates) and Gradient Boosting (learning rate, max depth) using techniques like Bayesian Optimization could enhance performance.
- Interpretability: Gradient Boosting offered feature importance, but the Neural Network can be improved, although different techniques were tried.

Proposed Alternative Approaches

- Exploration of Other Models:
  - Future studies could explore Support Vector Machines (SVMs) for imbalanced classification.
  - Decision Tree models were also tested; corresponding code is available for further analysis as a potential baseline.

- Unsupervised Feature Reduction:
  - Dimensionality reduction methods like PCA or Autoencoders could enhance feature effectiveness.

- Data Augmentation and Ensemble Methods:
  - Ensemble methods combining Neural Networks and Gradient Boosting via stacking or blending could improve predictive.

5. Conclusions

Summary of Findings

1. Regression Results:
   - Random Forest Regression outperformed Polynomial Regression with an MSE of 7,577.56 and an R² of 0.9368, effectively modeling non-linear interactions and providing feature importance insights.
2. Classification Results:
   - Gradient Boosting outperformed other models (Neural Network, Decision Tree) with 91.50% accuracy, 0.86 precision, 0.84 recall, and 0.85 F1 score. Its ability to handle class imbalance using SMOTE and its interpretability made it the best classification model.

Comparison with Initial Assessment

The experiments showed that Gradient Boosting outperformed Neural Networks (NN) in the binary classification task of predicting part usability.Initial expectations suggested that Neural Networks would excel due to their ability to model complex relationships, but attempts to improve NN performance, such as adding layers and changing dropout rates, did not made significant gains and even led to overfitting. Gradient Boosting achieved higher accuracy (91.50%) and F1-score (0.85), and effectively handled the class imbalance. These strengths make it a better fit for predicting part usability while offering actionable insights to improve manufacturing processes.

Final Recommendation

- Gradient Boosting is recommended for deployment to accurately predict defective parts. Its superior classification performance minimizes misclassification and provides interpretability through insights into key predictors.
- While Random Forest Regression was effective in predicting lifespan, the business's focus on identifying defects makes Gradient Boosting a more practical choice for deployment.

6. References

Breiman, L. (2001) ‘Random Forests’. *Machine Learning*. Available at: https://link.springer.com/article/10.1023/A:1010933404324 (Accessed: 8 November 2024).

Research on Defect Detection Method for Composite Materials Based on Deep Learning. *Applied Sciences*, 14(10), 4161. Available at: https://www.mdpi.com/2076-3417/14/10/4161 (Accessed: 17 November 2024).

Chen, T., & Guestrin, C. (2016) ‘XGBoost: A Scalable Tree Boosting System’. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Available at: https://doi.org/10.1145/2939672.2939785 (Accessed: 15 November 2024).

Peckov, A. (2024) 'Polynomial Models and Polynomial Regression'. *Doctoral Dissertation*. Available at: https://kt.ijs.si/wp-content/uploads/2021/11/phd_aleksandar_peckov.pdf (Accessed: 15 November 2024).

Lee, S., & Kim, H. (2019) 'Advances in Polynomial Regression for Complex Feature Interactions'. *Journal of Applied Mathematics and Computation*, 34, 140-155. Available at: https://pdfs.semanticscholar.org/e4fd/4f1e6aef60ded196b22ae4ce9575e9506852.pdf (Accessed: 10 November 2024)

Scikit-learn (2024) 'Tree-based models'. Available at: https://scikit-learn.org/stable/modules/tree.html (Accessed: 13 November 2024).

Scikit-learn (2024) 'DecisionTreeClassifier'. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier (Accessed: 16 November 2024).

Scikit-learn (2024) 'GridSearchCV'. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html (Accessed: 12 November 2024).

Scikit-learn (2024) 'Ensemble methods - Random Forest'. Available at: https://scikit-learn.org/stable/modules/ensemble.html#forest (Accessed: 10 November 2024).

Scikit-learn (2024) 'OneHotEncoder'. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html#sklearn.preprocessing.OneHotEncoder (Accessed: 10 November 2024).