COMP1801 - Machine Learning Coursework Report

VENKATA SAMBA SIVAREDDY CHILAMURU – 001410264

Word Count: 3370

# 1. Executive Summary

This report aims at classifying the lifespan of metal parts in an industry to anticipate proactive maintenance and saving on costs. Toward this, two approaches were made here: one as Logistic Regression and the other as XGBoost, a non-linear model, for machine learning. The data would be first pre-processed through feature engineering, scaling, and encoding. Hyperparameter tuning along with cross-validation was rigorously applied in search of the best configurations. The results depict that Logistic Regression reached an accuracy of 40%, where it was limited by data complexity, whereas XGBoost reached an accuracy of 88%, which it excelled in capturing intricate patterns within the data. On these metrics of accuracy, precision, and recall, XGBoost ends up being the dominant model to deploy. Hence, these results emphasize that XGBoost is capable of handling complex classification tasks and supports its usage in predictive maintenance for metal parts.

# 2. Data Exploration

## 2.1 Loading the Dataset

The dataset was exported using ‘pandas’ which is a popular data analysis library. The command ‘df = pd.read\_csv(‘COMP1801\_Coursework\_Dataset.csv’)’ was used to load the dataset into a DataFrame called df.

## 2.2 Overview of the Dataset Structure and Missing Values

Missing values can be checked using df.isnull().sum() ensuring the model does not provide predicted analysis. This returns zero for all columns, so the dataset is complete.

## 2.3 Categorical Variable Distribution

The dataset was explored using Exploratory Data Analysis (EDA). EDA distributions of categorical variables give insight into dominating classes and potential imbalances.

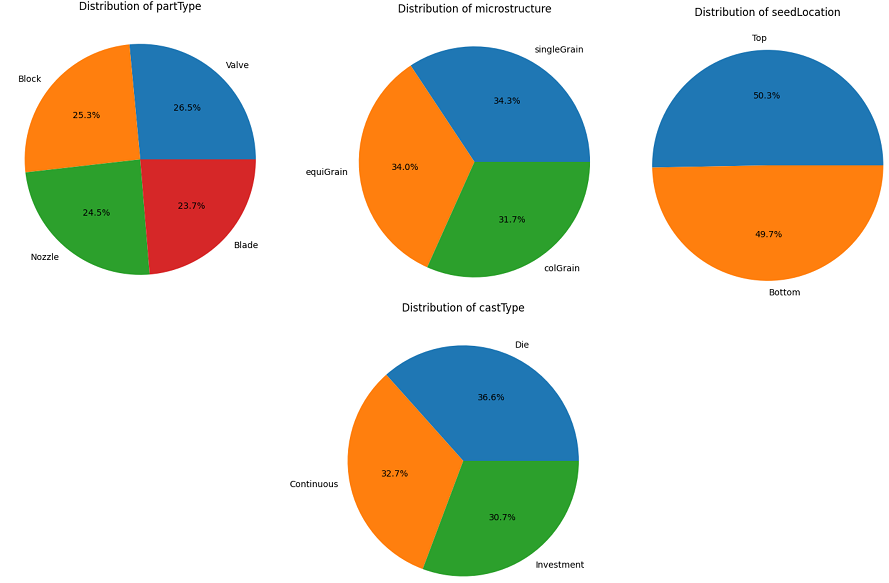


Figure 1: EDA

Figure 1 presents the visualization conducted through the EDA technique following the distributions for variables such as partType, microstructure, seedLocation, and castType is done by using pie charts created with df[col].value\_counts().plot.pie():

## 2.4 Summary of Numerical Variables

For numeric attributes, I will consider mean, median, and range. Observing the distribution and central tendency of columns like coolingRate, quenchTime, Nickel%, and Iron% will be useful as well because, in case of an outlier value or a high variance in those columns, that could impact the model.

## 2.5 Patterns and Relationships

* **Metal Composition**: Percentages of Nickel%, Iron%, and Cobalt% may be significant because they can interfere with its capability to resist wear and corrosion, influencing life expectancy.
* **Defects**: Characteristics like smallDefects, largeDefects, and sliverDefects may directly tie to durability; higher counts of defects probably link to shorter lifetimes.
* **Manufacturing Processes**: The castType, coolingRate, quenchTime, and forgeTime correspond to different manufacturing processes and heat treatment operations; such parameters could be important in predicting the lifetime.

## 2.6 Modelling Approach

Since Lifespan is a continuous variable, regression would be the appropriate approach. The most basic kind of linear regression would be the simplest and easiest to interpret. However, relationships often nonlinear or interactional when it comes to material composition with defects, a more complex model, such as the Support Vector Regression, may better capture higher-order interactions and nonlinearities.

# 3. Regression Implementation

## 3.1 Methodology

The Linear Regression (LR) and Support Vector Regression (SVR) were selected to predict the lifetime of metal parts using machine learning.

## 3.1.1 Model selection and justification

**Linear Regression (LR)**

Linear Regression was chosen as a benchmark model because it's easy and interpretable and can be easy to implement. Maulud and Abdulazeez (2020) states that Linear Regression model can easily interpret results even when the relationship between features and the target variable is likely linear. This will determine whether a simple linear relationship exists between predictor features and the target (lifetime).

**Support Vector Regression (SVR)**

Support Vector Regression is specifically suited to capture complex and non-linear relationships; thus, it would be a good fit for this task. Malik et al. (2021) states that SVR uses kernels to lift the input features to higher dimensions; hence, it allows the capturing of intricate patterns in data by just focusing on pairwise patterns.

## 3.1.2 Data Pre-processing

* **Handling of Categorical Variables**: There are categorical variables in the dataset, which have been encoded using one-hot encoding so that every category has been turned into a numeric variable.
* **Scaling Features**: Scaling applied and ensured that features have equal magnitudes, which is crucial for SVR since this algorithm is very sensitive to scales of features. All features are normalized using a standard scaling; zero mean and unit variance.
* **Train-Test Split**: Since the dataset was divided into training and test subsets, it was partitioned using the typical 80-20 split. The fitted model was realized on the training set.
* **Outlier Treatment and Data Balancing**: I used the IQR method to detect outliers since the numerical features have outliers; thus, I tackled data points outside 1.5 times IQR, thereby minimizing extreme values that could distort the model.

## 3.1.3 Hyperparameter Tuning

**Linear Regression Hyperparameters:**

* **fit\_intercept**: Whether to calculate the intercept for this model. If set to False, no intercept will be used in calculations. Not that setting fit\_intercept=False will incur a small overhead while fitting the model.
* **positive**: Forces coefficients to be positive which can help interpretability when all coefficients are expected to be non-negative.

**Support Vector Regression Hyperparameters**:

* **kernel**: The choice of kernel function is of prime importance as it will define the decision boundary in the feature space transformed.
* **C**: It's the regularization parameter C, representing the hardness of the margin for SVR.
* **gamma**: This parameter specifies the influence radius that one training example has on the boundary, for non-linear kernels.

## 3.2 Evaluation

## 3.2.1 Experiment 1: Linear Regression Model Optimisation

Linear Regression model was trained and tuned with the help of GridSearchCV for identifying the best hyperparameters. The adoption of a 5-fold cross-validation approach was undertaken within the optimisation process of the model to evaluate the model’s generalisation capabilities.

* **fit\_intercept**: [True, False]
* **positive**: [True, False]

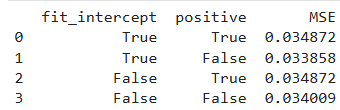


Figure 2: Hyperparameter Tuning for Linear Regression

Figure 2 displays the results from the hyperparameter tuning of the LR model. The results interpreted 0.033858 as the best Mean Squared Error (MSE).

Based on the results, the final Linear Regression model was trained with the help of the best hyperparameter obtained and prediction made during Hyperparameter Tuning for Linear Regression. The performance metrics obtained were.

* MSE: 0.031233
* RMSE: 0.1767
* R-squared: 0.113

## 3.2.2 Experiment 2: Support Vector Regression Model Optimisation

In the second experiment, Support Vector Regression Model optimisation was undertaken with the help of GridSearchCV. The technique assisted in identifying the best combination of hyperparameters.

* **kernel**: ['linear', 'rbf', 'poly']
* **C**: [0.1, 1, 10]
* **gamma**: ['scale', 'auto']

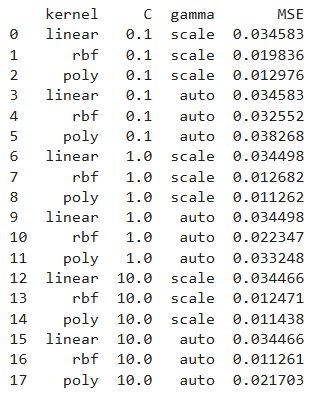


Figure 3: Hyperparameter Tuning for Support Vector Regression (SVR)

Figure 3 interprets the best hyperparameters as kernel= rbf, c= 10, and gamma= auto as determining MSE of 0.011261 in the test.

Based on the results final SVR model will be trained for the best configurations obtained under Hyperparameter Tuning for Support Vector Regression. The performance metrics obtained;

* MSE: 0.009348
* RMSE: 0.0967
* R-squared: 0.7340

## 3.2.3 Comparing models

After the optimisation process of the models, the performance of the best version of Linear Regression and Support Vector Regression is;

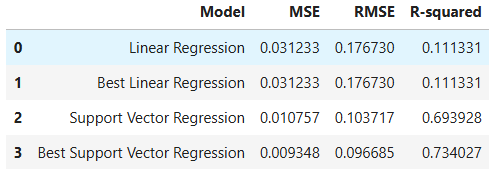


Figure 4: Comparative analysis table of models

Figure 4 evaluates both models optimised and compares them based on Mean Square Error (MSE), Root Mean Squared Error (RMSE), and R-squared.

**Evaluation**

* **MSE**: MSE determines the average of squared differences between actual and predicted values. The smaller the MSE, the better a given model performs at prediction (Hodson, Over and Foks, 2021). Similarly, SVR by itself remarkably outperforms Linear Regression, with an MSE improvement of more than 0.02.
* **RMSE**: RMSE, being the square root of MSE, is better interpretable because it retains the same unit as the target variable. The RMSE is lower for SVR (0.0967) than for Linear Regression (0.1767) and therefore reflects a higher degree of accuracy in prediction.
* **R-squared**: R-squared is the part of the variance of the target variable explained by the model. The higher the R-squared value implies that the model succeeded in explaining more variation (Hayes, 2021). Now, the R-squared for the SVR model is at 0.7340, which is so much better than that of the Linear Regression model - 0.1113.

## 3.3 Critical Review

**Strengths**

SVR was used as a first benchmark that is interpretable to address the non-linear relations in the data, where improvement in metrics is observable. Key strengths include:

* **Hyperparameter Tuning is Effective**: The trick applied was doing a GridSearchCV along with 5-fold cross-validation such that the best model parameter was selected to achieve maximum generalization.
* **Outlier-Handling and scaling**: outlier handling and scaling of features do have proper pre-processing, which is crucial, especially because SVR is sensitive to the scales of the features and outliers.

**Areas of Improvements**

* **Parameter Range Exploration**: Even though GridSearchCV was used, it may not span a range that is quite as comprehensive as those methods such as RandomizedSearchCV or Bayesian optimization, which sometimes find much better parameters with far fewer iterations and computational resources.
* **Outlier Detection**: This outlier handling used the IQR method; however, more advanced methods like Isolation Forest or Local Outlier Factor (LOF) can properly identify and handle outliers to have less distortion in the distribution of data.

**Alternative approaches for future exploration**

* **Cross-validation Techniques**: Another area to test could be using time-based splitting or rolling cross-validation if data at least partially shows a temporal structure. While probably not applicable here, it would bring robustness to the model evaluation on data with a temporal dependency in any future task.
* **Neural Network Regression**: The best way to add more complex patterns might be the concept of implementing a rather simple neural network model. Hyperparameter adjustments to the number of layers, neurons within the layer, and activation functions for the neural networks might yield better fits of the non-linear relationships in the data.

# 4. Classification Implementation

## 4.1 Feature Crafting

**Binary Classification based on a 1500-hour Threshold**

Within the binary classification model, binary label 1500\_label was created within the dataset that helps in identifying parts with a lifespan above 1500 hours. The dataset was classified where;

* Label 0 represents parts with a lifeline ≤ 1500 hours and were considered defective.
* Label 1 represents parts with a lifeline >1500 hours and were considered non-defective.

**Advanced grouping with K-Means Clustering**

By acknowledging the limitations of the basic binary classification model, the K-Means clustering technique is applied providing additional grouping patterns within the lifespan data. Within the process of implementing the K-Means clustering, numerical features were normalised with the use of MinMaxScaler which ensures the clustering algorithm treats each feature equally.

An Elbow Method plot was used to determine the optimal number of clusters;

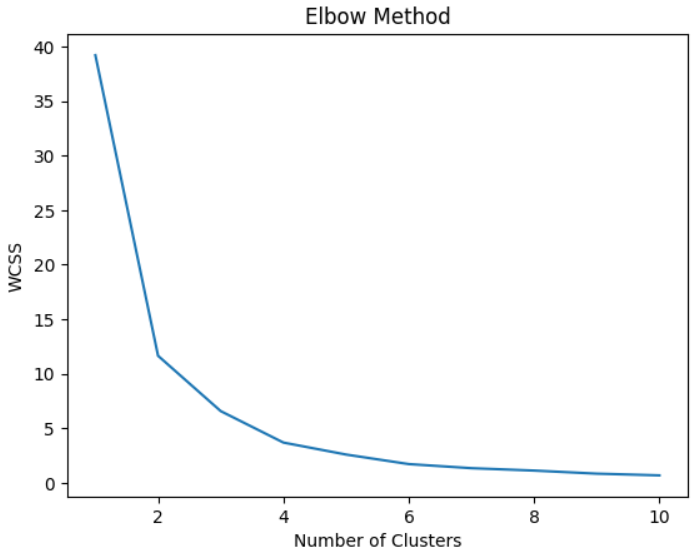


Figure 5: Defining clusters using the Elbow Method

Figure 5 interprets three clusters (k=3) defining the lifeline of parts in the dataset.

Rationales for k=3 clusters selected;

The Elbow Method is widely known in clustering for choosing the best k-balancing explained variance with technique (Asazuma et al. 2023). Plots of WCSS for different numbers of clusters from 1 to 10 indicated a sudden fall of Within-Cluster Sum of Squares (WCSS) up to k=3. After that, the WCSS's fall was not as steep, indicating that a further increase in the number of clusters would not enhance clustering quality much.

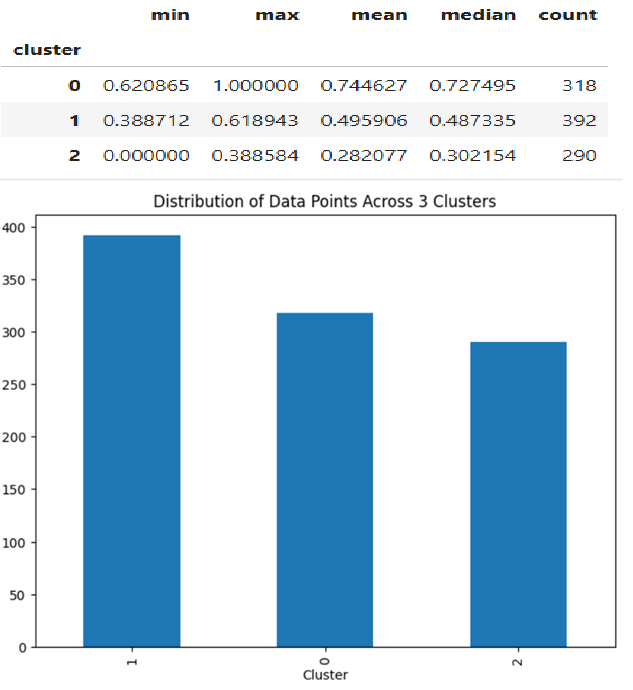


Figure 6: Data distribution within the clusters

Figure 6 provides a clustering analysis of grouped parts within the three distinct clusters identified based on the lifeline of the parts;

* Cluster 0: Part lifeline between 62.1% - 100% (High)
* Cluster 1: Part lifeline between 38.9% - 61.9% (Mid)
* Cluster 2: Part lifeline between 0% - 38.38% (Low)

Clusters provided a more granular view of the part lifespans. Three separate lifespans instead of one large binary will be identified. This can guide better variations in part durability and process adjustments by the company for different lifespan groups.

**Feature Encoding and Data Preparation for Model Training**

With LabelEncoder, the categorical features are encoded to convert them into suitable numeric values used by the machine learning models. After encoding, the dataset was split into training and testing sets, with an 80:20 ratio, where new cluster labels serve as target variables (y) and the remaining features as the input variables (X)

## 4.2 Methodology

**Model Selection**

**XGBoost (Extreme Gradient Boosting)**

XGBoost has been chosen as the first-level classification model because it is quite strong and resistant to complex relationships between features and their target. XGBoost is an ensemble method where multiple trees are grown iteratively (Bhati et al., 2021). It uses gradient boosting to allow high predictive accuracy; hence, it is suitable for this problem where complex interactions of many features likely involve the target class.

**Logistic Regression**

Logistic Regression is a linear classifier that provides a means to understand how well features relate to the target classes, which is very good for baseline analysis (Abbas et al., 2022). On the other hand, Logistic regression might fail if it comes up against patterns that are complex and rather non-linear (Babu Nuthalapati and Nuthalapati, 2024). However, the model is applied here for a comparison with the more advanced models. Logistic Regression would be used as its performance on binary and multiclass classification problems is mostly optimal, especially when the class separation is not significantly challenging so this should be a starting choice to set the baseline performance.

**Data pre-processing**

* **Feature Exclusion**: Feature Lifespan is removed from both the train set and the test set, so data would not leak because the purpose was to predict the threshold lifespan based on other feature values.
* **Feature Scaling**: Normalizing the numeric features so that all the features receive a similar weightage to train the model.
* **Categorical Encoding**: With categorical encoding, the categorical features are encoded with the help of one-hot encoding to transform them into numerical forms so that they can be efficiently used within the models.
* **Data Splitting**: Dividing the set into train and test, with 80% data of the total for model training and 20% for model testing.
* **Data Balancing**: Even though unbalanced data may bias models toward frequently occurring classes, techniques such as Synthetic Minority Over-sampling Technique (SMOTE) or class weights can compensate for class imbalances.

**Hyperparameter Tuning Framework**

**Logistic Regression**

* **Regularisation Parameter (C)**: C is a regularization parameter that controls how strong the regularization is. The lower the value for C, the stronger the regularization.
* **Penalty**: One of the following types of penalty must be used: l1, which tries to drive coefficients zero; and l2, which tries to shrink all coefficients uniformly.
* **Solver**: Amongst the most important hyperparameters is the solver; a choice either between liblinear and saga that improves convergence during training. The data size chosen will depend on the type of penalty hence, the solver is dependent on both computation efficiency and model performance.

**XGBoost Hyperparameter**

* **Learning Rate**: It defines how big the step size is at each iteration. With a small number of learning rate iteration is more prone to improve the performance but at a high cost in terms of the computation done.
* **Max Depth**: This is the maximum depth of each tree. The more the depth, the more complex patterns can be captured by the model but tends to overfit.
* **Number of Estimators**: An optimal range determined between accuracy and efficiency.

## 4.3 Evaluation

With the help of a predefined hyperparameter tuning framework, systematically testing multiple configurations was done through the application of Grid Search.

**Logistic Regression Tuning Results**

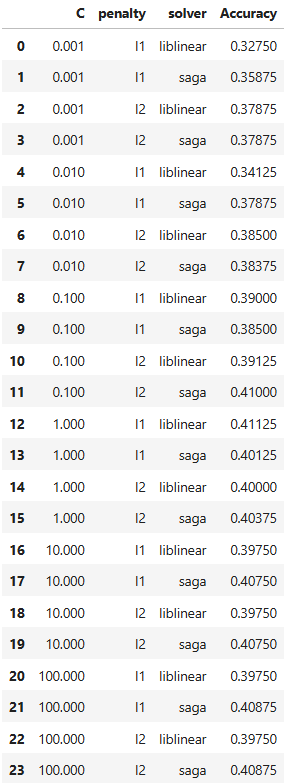
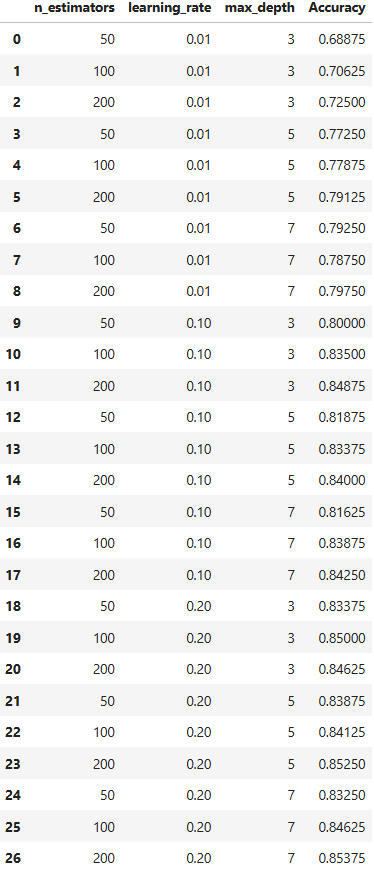


Figure : Best hyperparameters for Logistic Regression

Figure 7 interprets the best accuracy for Logistic Regression when using the regularization parameter C=1.0 along with the l1 penalty and the use of the solver liblinear, which had a cross-validation accuracy of about 41.1 percent.

**XGBoost Tuning Results**



**Figure 8: Best hyperparameters for XGBoost**

Figure 8 interprets XGBoost tuning using the learning rate of 0.2 maximum depth of 7, and 200 estimators, one had a higher cross-validation accuracy at about 85.4 percent.

**Final Model Version**

Accuracy, precision, recall, F1-score Metrics. For the test set, both models will be tested using accuracy, precision, recall, and F1-score. These metrics allow us to look at model performance from several angles:

* Accuracy. It measures the general correctness of the predictions.
* Precision: The number of correctly predicted positives out of all the positive predictions, useful when false positives are costly.
* Recall: it reflects the ability of a model to capture actual positives, important in those tasks where false negatives are critical.
* F1 Score: Harmonic mean of precision and recall, the mean will put both values at par to give an overall score to its performance.

**Model Performance**

**Logistic Regression**

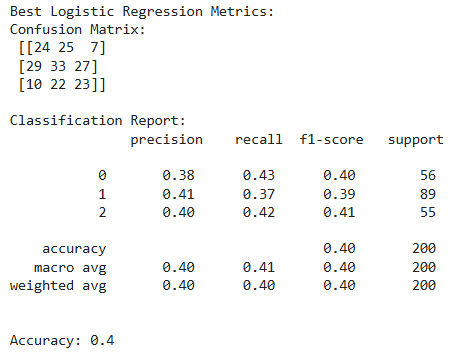


Figure : Confusion Matrix to evaluate model performance

Figure 9 interprets the accuracy of 40% obtained while carrying a confusion matrix based on best model configurations after hyperparameters.

**XGBoost**

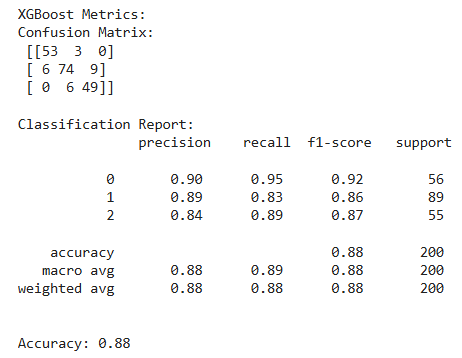


Figure : Confusion Matrix to evaluate model performance

Figure 10 interprets the accuracy of 88% obtained while carrying a confusion matrix based on best model configurations after hyperparameters.

**Critical evaluation of both models**

The performance of both algorithms is determined based on the result that reflects how accurate XGBoost is in terms of overall accuracy compared to Logistic Regression at 40%. Therefore, the higher F1-score and recall values for XGBoost, strongly imply an excellent ability to handle complex patterns within the dataset, and hence, make up a very reliable model fit for deployment.

## 4.4 Critical Review

**Strengths of the Methodology**

* **Effective Hyperparameter Tuning**: The use of Grid Search with cross-validation for the tuning of hyperparameters resulted in all models achieving their optimal configuration.
* **Comprehensive evaluation metrics**: Precise, Recall, and F1-score and Accuracy Evaluation of Every Model provided a multi-dimensional evaluation of the performance of the model.

**Areas of Improvements**

* **Hyperparameter Tuning Complexity**: Implementing Random Search or Bayesian Optimization may provide a more efficient hyperparameter tuning process that would enable easier exploration of hyperparameters and therefore possibly better configurations.
* **Data Preprocessing Coherence**: Including scale normalization and encoding, other more specific operations, like feature selection or extraction, would make a difference in the model's performance.

**Alternative Approaches for Future Investigations**

* **New Model Architectures**: More experiments in the future would include ensemble models, such as Random Forests, or even advanced neural network architectures like Convolutional Neural Networks.
* **Other Hyperparameter Tuning Strategies**: Using Bayesian Optimization or evolutionary algorithms might make the tuning process much more efficient, especially for models with large parameter spaces like XGBoost.

# 5. Conclusions

From the testing of Logistic Regression and XGBoost, it is concluded that XGBoost performed better than Logistic Regression when considering the classification of metallic parts, with an accuracy of 88% whereas logistic regression did 40%. This outcome supports the general assumption that since XGBoost captures complex relations more than Logistic Regression, the former would perform better for this task of classification. In contrast, compared to what I discussed in the Data Exploration section (Part 2), where I identified possible trends and patterns in the data, XGBoost exploited them much better. Based on the results the effective model that will be deployed for classification tasks is XGBoost. This model has higher accuracy and can apply complex feature interaction perfectly, making it the right model for determining whether a part is usable or not. It is concluded that XGBoost had 88% whereas Logistic Regression was at 40%.

# 6. References

Abbas, A., Khan, M.A., Latif, S., Ajaz, M., Shah, A.A. and Ahmad, J., (2022). A new ensemble-based intrusion detection system for internet of things. *Arabian Journal for Science and Engineering*, pp.1-15.

Asazuma, Y., Funayama, H., Matsubayashi, Y., Mizumoto, T., Reisert, P. and Inui, K., (2023). Take No Shortcuts! Stick to the Rubric: A Method for Building Trustworthy Short Answer Scoring Models. In *International Conference on Higher Education Learning Methodologies and Technologies Online* (pp. 337-358). Cham: Springer Nature Switzerland.

Babu Nuthalapati, S. and Nuthalapati, A., (2024). Accurate weather forecasting with dominant gradient boosting using machine learning. *Int. J. Sci. Res. Arch*, *12*(2), pp.408-422.

Bhati, B.S., Chugh, G., Al‐Turjman, F. and Bhati, N.S., (2021). An improved ensemble based intrusion detection technique using XGBoost. *Transactions on emerging telecommunications technologies*, *32*(6), p.e4076.

Hayes, T., (2021). R-squared change in structural equation models with latent variables and missing data. *Behavior Research Methods*, *53*(5), pp.2127-2157.

Hodson, T.O., Over, T.M. and Foks, S.S., (2021). Mean squared error, deconstructed. *Journal of Advances in Modeling Earth Systems*, *13*(12), p.e2021MS002681.

Malik, A., Tikhamarine, Y., Sammen, S.S., Abba, S.I. and Shahid, S., (2021). Prediction of meteorological drought by using hybrid support vector regression optimized with HHO versus PSO algorithms. *Environmental Science and Pollution Research*, *28*, pp.39139-39158.

Maulud, D. and Abdulazeez, A.M., (2020). A review on linear regression comprehensive in machine learning. *Journal of Applied Science and Technology Trends*, *1*(2), pp.140-147.