COMP1801 - Machine Learning Coursework Report

Word Count: 3558

1. Executive Summary

The diverse ML methodologies were applied to two datasets: one containing processing parameters and measurements of metal parts, and the other comprising surface scans of these parts. The primary objective was twofold: predicting metal part lifespans and automating defect classification.Traditional regression models, including Linear, Lasso, Ridge, and Random Forest, were employed to predict lifespans, with Random Forest emerging as the most accurate model with accuracy of 97% in predicting lifespans. For defect classification, Binary Classification models and Convolutional Neural Networks (CNNs) were tested, with the Random Forest model excelling in classifying defects above 1500 hours of lifespan having a balanced accuracy of 94%, while CNNs showcased promise in identifying various defect types. Clustering analysis, employing the K-means algorithm with a selected cluster size of 2, highlighted distinct parameter groups.

2. Data Exploration

Before feeding the dataset into a model, the features, their distributions, descriptive statistics, correlations and relationship with the target variable (Lifespan) should be evaluated. In addition, identifying missing values, outliers, duplicate records or imbalances impact the prediction of the target variable.

2.1 Overview

	Lifespan	partType	microstructure	coolingRate	quenchTime	forgeTime	smallDefects	largeDefects	sliverDefects	seedLocation	castType
0	284.161690	Blade	colGrain	25	4.460592	7.937116	22	0	7	Top	Investment
1	1599.551748	Blade	singleGrain	9	1.425973	2.432948	2	0	0	Bottom	Die
2	768.311031	Nozzle	colGrain	26	2.508879	3.841211	25	0	0	Bottom	Investment

Figure 1: Head of the dataset containing the first three records

```
Feature list:

['partType',
 'microstructure',
 'coolingRate',
 'quenchTime',
 'forgeTime',
 'smallDefects',
 'largeDefects',
 'sliverDefects',
 'seedLocation',
 'castType']
```

Figure 2:Ten features of the dataset

```
Column data types:

Lifespan          float64
partType           object
microstructure     object
coolingRate         int64
quenchTime        float64
forgeTime         float64
smallDefects        int64
largeDefects        int64
sliverDefects       int64
seedLocation       object
castType           object
dtype: object
```

Figure 3: Data type of the target & each feature

The dataset consists of 1000 records, 6 numerical and 4 categorical features and a numeric target, 'Lifespan'(figure 3).

2.2 Missing values

As in figure 4, there are no missing values in the dataset.

```
Dataframe null value count:

Lifespan          0
partType          0
microstructure    0
coolingRate       0
quenchTime        0
forgeTime         0
smallDefects      0
largeDefects      0
sliverDefects     0
seedLocation      0
castType          0
dtype: int64
```

Figure 4: Count of missing values

2.3 Descriptive statistics of the data

```
Describe categorical data:
        partType microstructure seedLocation    castType
count       1000           1000         1000        1000
unique         4              3            2           3
top       Nozzle    singleGrain          Top  Continuous
freq         293            357          507         356
```

Figure 5: Descriptive statistics of the categorical features

partType	
Nozzle	293
Valve	241
Blade	236
Block	230

microstructure	
singleGrain	357
colGrain	328
equiGrain	315

seedLocation	
Top	507
Bottom	493

castType	
Continuous	356
Investment	333
Die	311

Figure 6:Value counts of categorical features

Describe numerical data:

	Lifespan	coolingRate	quenchTime	forgeTime	smallDefects
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000
mean	1366.373468	17.480000	2.786059	5.497136	13.37100
std	519.026551	7.557958	1.320935	2.613501	8.07047
min	115.120563	5.000000	0.501046	1.017799	0.00000
25%	960.976320	11.000000	1.608916	3.203739	8.00000
50%	1470.377014	17.000000	2.824488	5.510765	16.00000
75%	1757.165684	24.000000	3.902389	7.735951	20.00000
max	2380.142759	30.000000	4.990795	9.988511	33.00000

	largeDefects	sliverDefects
count	1000.000000	1000.000000
mean	0.117000	0.286000
std	0.565359	1.351307
min	0.000000	0.000000
25%	0.000000	0.000000
50%	0.000000	0.000000
75%	0.000000	0.000000
max	4.000000	10.000000

Figure 7: Descriptive statistics of the numerical features

Figures 5 and 6 outline categorical feature cardinality and their occurrences. The dataset includes four part types: ‘Nozzle’, ‘Valve’, ‘Blade’, and ‘Block’. 'Nozzle' is the most frequent, appearing 293 times, followed by 241, 236, and 230, respectively. Figure 7 presents descriptive statistics for features including 'Lifespan', ranging from 115.12 to 2380.14, with an average of about 1366.37 and a standard deviation of 519.03. Other features like 'CoolingRate', 'QuenchTime', 'ForgeTime', 'smallDefects', 'largeDefects', and 'sliverDefects' exhibit specific value ranges and corresponding averages with deviations.

2.4 Check for the duplicate records

The duplicate check reflected that the dataset contains 1000 unique records.

2.5 The impact of numerical features to the target

	Lifespan	coolingRate	quenchTime	forgeTime	smallDefects	largeDefects	sliverDefects
Lifespan	1.00	-0.79	0.07	0.03	-0.65	-0.02	-0.05
coolingRate	-0.79	1.00	0.03	-0.01	0.88	0.01	0.00
quenchTime	0.07	0.03	1.00	0.07	0.02	0.06	0.04
forgeTime	0.03	-0.01	0.07	1.00	-0.00	0.06	-0.01
smallDefects	-0.65	0.88	0.02	-0.00	1.00	0.01	-0.02
largeDefects	-0.02	0.01	0.06	0.06	0.01	1.00	-0.03
sliverDefects	-0.05	0.00	0.04	-0.01	-0.02	-0.03	1.00

Figure 8: Correlation heatmap

The heatmap illustrates correlations between features, indicating how strongly they're related on a scale from -1 to 1. A positive correlation means both features tend to increase together, while a negative correlation implies an inverse relationship. Focusing on 'Lifespan' as the target variable, it exhibits moderate negative correlations with 'coolingRate' (-0.79) and 'smallDefects' (-0.65), indicating a decrease in 'Lifespan' as these features increase. Moreover, 'coolingRate' and 'smallDefects' show a strong positive correlation (0.88), suggesting their tendency to vary together. Overall, most feature correlations are weak or close to zero.

2.6 The impact of categorical features to the target

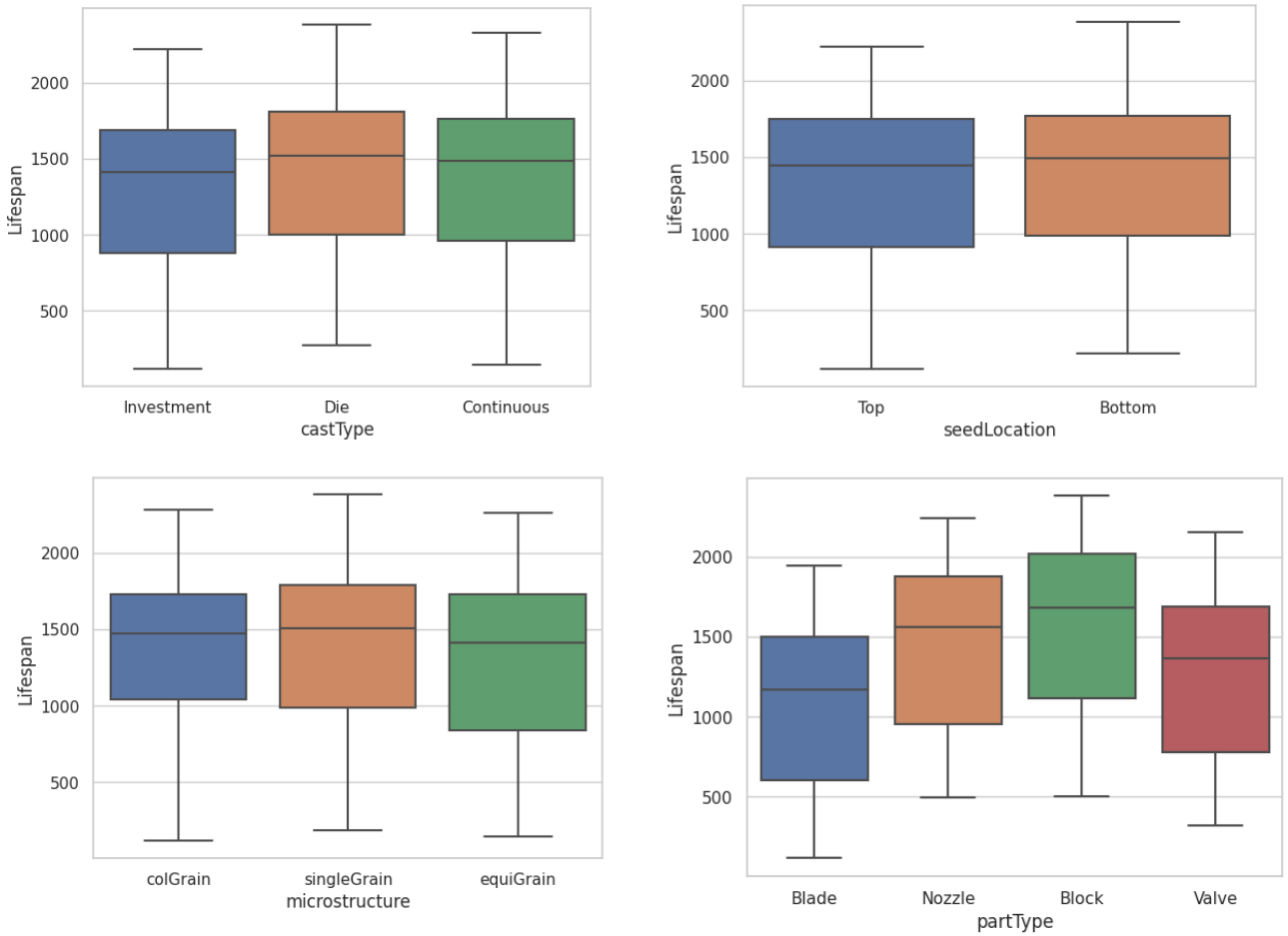


Figure 9: Boxplot representation of castType, seedLocation, microstructure and partType

Figure 9 shows the distribution of data for four categorical features related to the 'Lifespan' target variable. 'catType,' 'seedLocation,' and 'microstructure' display minimal variation across categories, indicating limited impact on the target. However, 'partType' shows some variation: 'Blade' tends to have a lower Lifespan, 'Block' exhibits a higher Lifespan, and 'Nozzle' follows 'Block' with a moderately higher Lifespan.

2.5 Outliers

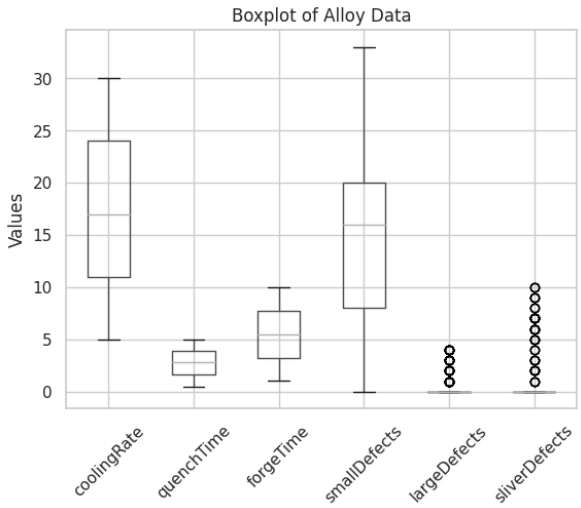


Figure10: Data distribution of the numerical features

Outliers, which represent unusual data points, were observed in 'largeDefects' and 'sliveDefects,'. They were kept in the dataset due to their potential predictive significance, which the model should capture.

2.6 Data Imbalances

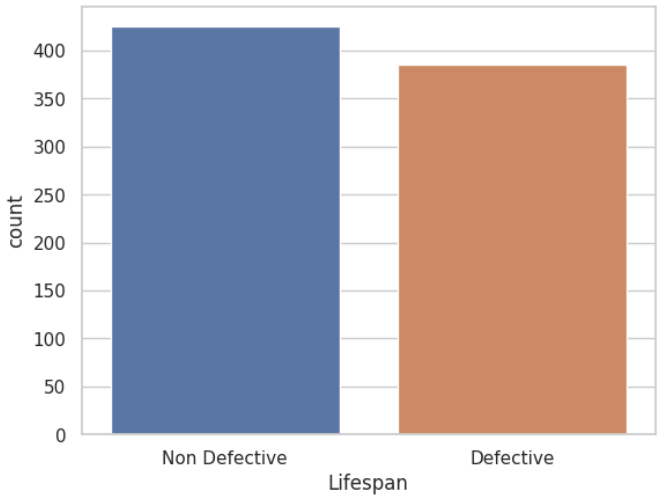


Figure 11: Class imbalance visualisation

There's a negligible difference in instances between the "Non-defective" and "Defective" classes, for binary classification.

2.7 Dataset split into train, validation and test

The dataset was partitioned as train, test, and validation with reasonable balanced allocation, 8:1:1 ratio to prevent overfitting. Shuffling non-time series data mitigates bias, and setting the random state to zero ensures reproducibility of the random split. In the regression problem, simple random sampling was employed for the random split, while in the classification problem, stratified sampling was used to maintain representative class proportions across the splits.

2.8 Target and feature encoding

Encoding involves converting categorical features into numerical representations as ML algorithms accept numerical values. This process should occur after data splitting to prevent data leakage (Cantrell, 2023). 'microstructure' and 'castType' were encoded using one-hot encoding, resulting in 17 new features. seedLocation is a binary category. However, for binary categories like 'seedLocation,' one-hot encoding doesn't provide any advantage since it ends up creating two columns where 1's and 0's are inverted creating redundant columns. In such cases, label encoding is more suitable. Once training data was fit to learn the parameters of encoding, all training, testing and validation features were transformed. During classification tasks, label encoding was preferred over one-hot encoding for the target variable to ensure unique numerical values.

2.9 Feature scaling

Standard scaler was used assuming the input is normally distributed. Min-max scaling wasn't preferred due to uncertainties associated with minimum and maximum feature values. Scaling operations were repeated whenever there was a modification in the feature matrix (Inside the training loop).

3. Regression Implementation

Linear Regression fits a linear relationship between the input features (x) and target ($\hat{y}$) by finding the coefficients (θ) that minimise the error between the predicted values and the actual values.

$$\hat{y} = \theta_0 + \theta_1 x_1 + \quad \theta_2 x_2 + \ldots + \quad \theta_n x_n$$

This equation represents a linear combination of input features weighted by their respective coefficients.

Lasso Regression adds an L1 regularisation penalty to the standard Linear Regression objective function.

$$Cost_{Lasso} = \sum_{i=1}^{m} \quad (y_i - \hat{y}_i)^2 + \alpha \sum_{j=1}^{n} \quad |\theta_j|$$

where m is the number of features,

n is the number of samples,

$\sum_{i=1}^{m} \quad (y_i - \hat{y}_i)^2$ is the sum of squared error over the predicted and actual output,

$\alpha \sum_{j=1}^{n} |\theta_j|$ absolute sum of the coefficients multiplied by a hyperparameter α.

In Ridge regression the L2 regularisation penalty term is $\alpha \sum_{j=1}^{n} \theta_j^2$.

Both regularisation techniques prevent overfitting by penalising large coefficients, aiming to control the strength of regularisation and fine-tune the model's complexity. The primary purpose of regularisation is to prevent complex models from fitting noise in the data and to encourage simpler, more generalised models that perform well on unseen data.

Experiments performed:

The dataset was loaded into the colab environment, accessing the file from Google Drive. Tasks 2.1 to 2.8 were done regarding EDA and data preprocessing. The feature scaling is done inside the for loop, because the changing of the feature matrix (degree of polynomial) needs to re-perform scaling. as polynomial features are sensible to the data they are using.

Feature selection is essential to enhance model performance by reducing computational complexity and preventing overfitting,often carried out through filter or wrapper methods. Here, filter methods were selected for computationally less expensiveness and absence of additional model training. But it cannot guarantee the best subset of features. Therefore the experiment was done using 2 sets of feature sets.

Feature set 1: All encoded features

Feature set 2: Categorical features impacting the target (task 2.6) and all numerical features.Numerical features except coolingRate and smallDefects exhibit lower correlations.Their removal negatively impacts the model performance.

Following procedure was done to feature set 1 and 2.

Begins by exploring polynomial regression with degrees from 1 to 7, begins by exploring polynomial regression with degrees ranging from 1 to 7 to determine the most suitable degree by calculating MSE for each degree on both training and validation sets to obtain the best degree with the lowest validation MSE. Subsequently, it conducts Simple Linear, Lasso, and Ridge Regression. For Ridge and Lasso, it iterates through various alpha values to determine the best regularisation weight, evaluating their performance on training and validation datasets based on MSE. The chosen optimal alpha is then used to retrain the model. Finally, the best-degree polynomial regression model is trained and evaluated on the test dataset, measuring its performance via MSE and R2 score to assess its predictive accuracy.

In figure 12, the threshold point 0.1 has the minimum MSE. making it the selected alpha. Next figure, the overlap between the predicted and true values suggests the model is successful.

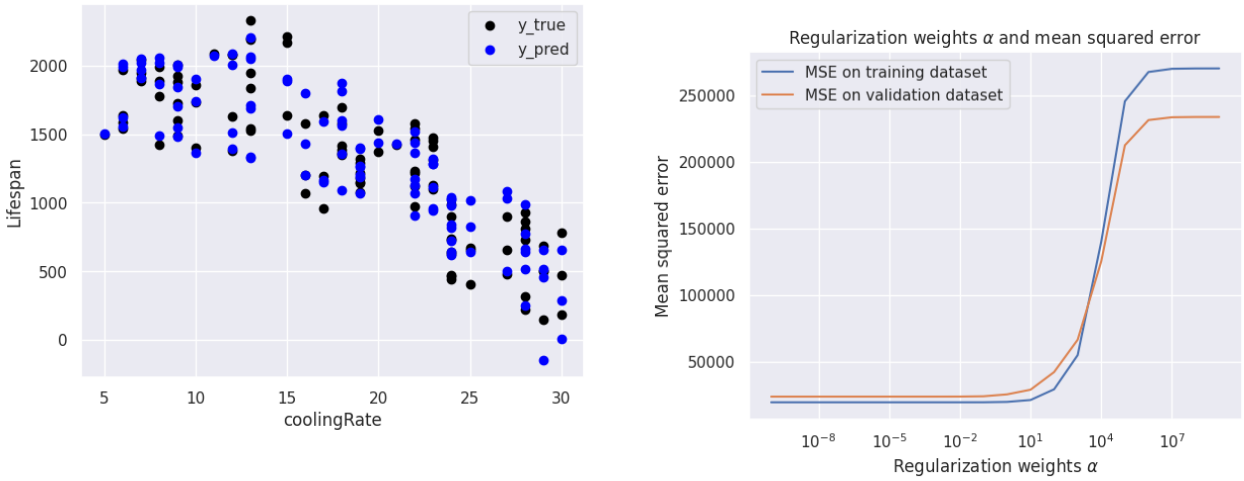


Figure 12 : MSE vs alpha for Ridge Regression and scatter plot of predictions and true targets for the feature set 1

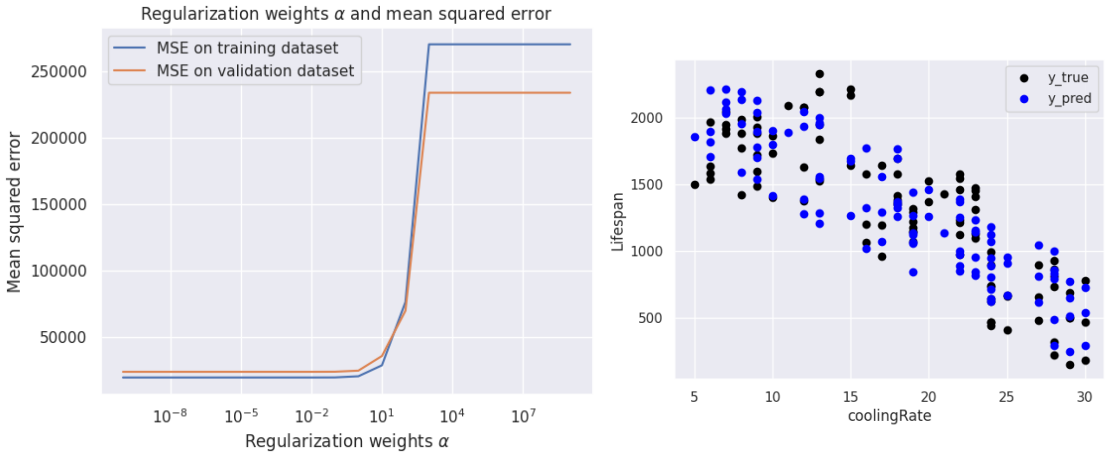


Figure 13 : MSE vs alpha for Ridge Regression and scatter plot of predictions and true targets for the feature set 1

Table 1: Chosen hyperparameter for the Ridge and Lasso for Feature set 1 & 2

	Feature set 1	Feature set 2
Best Degree	2	1
Best alpha (Lasso)	0.1	0.1
Best alpha (Ridge)	0.001	0.1

Table 2: Performance metrics of all the models

Feature					Feature			

set1					set2			
Model	R-squared score	MSE-train	MSE-validate	MSE-test	R-squared score	MSE-train	MSE-validate	MSE-test
Simple Linear Regression	0.9136	19853.26	24854.67	24944.28	0.8567	43820.90	58015.54	41373.87
Ridge Regression	0.9147	19472.30	23738.13	24631.85	0.8956	27841.97	28168.77	30150.02
Lasso Regression	0.8561	23737.89	23735.27	41562.74	0.8529	27841.96	28168.77	42491.36
Random forest Regression	**0.9799**	769.35	**5833.62**	**5783.66**	0.9567	3187.71	12758.59	12496.89

MSE measures prediction errors, R-squared, ranging from 0 to 1, indicates how well the model fits the data, with higher values suggesting a better fit. If validation and test MSE are approximately equal,it suggests that we are not overfitting on the validation set.

Conclusion and Best model:

Feature set 1 was chosen for model evaluation due to its higher R-squared score in Table 2.

Similarity in performance metrics (Table 2) across Linear, Lasso, and Ridge regressions, indicate that alterations in Lasso and Ridge did not impact their performance compared to the simpler Linear regression. Random Forest regression, an alternative strategy, was explored for its ability to handle nonlinear relationships and intricate feature interactions, potentially offering a unique approach.

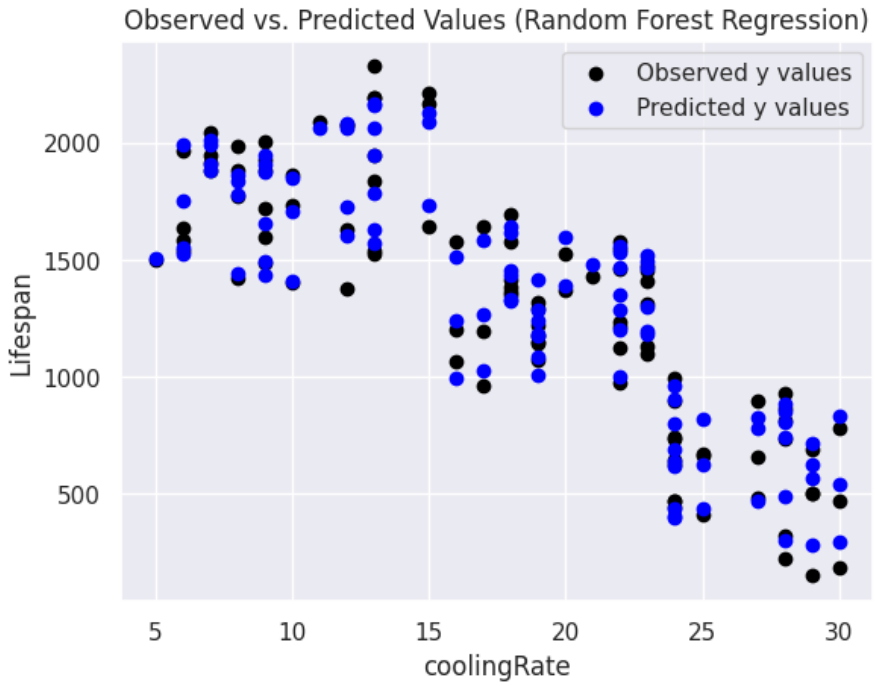
Therefore, Random Forest regression emerged as the best model. The Random Forest algorithm works by constructing multiple decision trees during training. Each tree is trained on a bootstrap sample of the dataset and utilises a random subset of features at each split. When making predictions, the algorithm aggregates the predictions from each tree to determine the final output. The randomness introduced in building each tree and combining their predictions helps in reducing overfitting and capturing complex relationships within the data.

The algorithm's performance was examined by tuning hyperparameters (number of trees (100,2002,300) in the forest and maximum depth (5,10,15) allowed for each tree in the forest) using grid search and assessing their performance using MSE and R-squared scores on the validation set. Best hyperparameters for Random Forest are 200 trees and max_depth=15. Tuning additional hyperparameters might introduce unnecessary complexity without significant improvements,

especially when these two parameters have a substantial impact on model accuracy and generalisation. The overlap between the predicted and true values in figure 14 suggests that the model's predictions are accurate.

Figure 14: True and predicted values of the 'Lifespan' based on the 'coolingRate' feature for the Random Forest Regression model on the test set

The *R*-squared score of 0.9799 indicates that the model explains nearly 97% of the variability in the



dataset, reflecting a high level of accuracy in predicting the target variable. MSE values of 5833.62 for validation and 5783.66 for testing are closely aligned, with minimal divergence which suggests that the model has not been overfitted.

4. Binary Classification Implementation

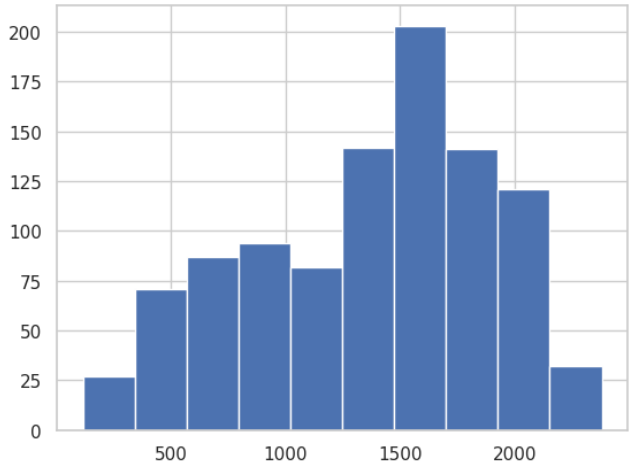


Figure 15: Distribution of Lifespan in Alloy Dataset

A binary target was implemented by assigning 'Non-defective' if 'lifespan' is greater than 1500, 'Defective' Otherwise. It is replaced with the old one as it acts as a leaky feature to the classification model.

After EDA and data-preprocessing (tasks 2.1 to 2.9), model preprocessors were initialised, and subsequent steps included training and validation of multiple models to identify the best-performing model based on the validation dataset. Finally, the chosen best model was evaluated using a test dataset.

Hyperparameter optimization:
Different machine learning algorithms have various hyperparameters that impact their performance. Here's an overview of some influential hyperparameters for the mentioned algorithms:(Hastie *et al*, 2009)

Table 3: Overview of influential hyperparameters for the classification algorithms

Logistic Regression	• Regularisation parameter (C): Controls overfitting by penalising large coefficient values. • Solver: The algorithm used for optimization (e.g., 'liblinear', 'lbfgs', 'sag', 'newton-cg'). • Penalty: Specifies the norm used in the penalization ('l1' or 'l2').
Support Vector Machine	• Kernel type: Selection of kernel function ('linear', 'rbf', 'poly', 'sigmoid').

	• C parameter: Penalty parameter for the error term, controlling the trade-off between decision boundary complexity and classification error. • Gamma (for RBF kernel): Defines how far the influence of a single training example reaches.
Naive Bayes	• Smoothing parameter (alpha): Controls additive (Laplace/Lidstone) smoothing, preventing zero probabilities in probability estimation.
k-Nearest Neighbors	• Number of neighbours (k): The number of nearest neighbours to consider. • Distance metric: The measure used to calculate the distance between instances (e.g., Euclidean, Manhattan, Minkowski).
Decision Tree	• Maximum depth: Maximum depth of the tree to control overfitting. • Minimum samples split: Minimum number of samples required to split a node. • Criterion: The function to measure the quality of a split ('gini', 'entropy').
Random Forest	• Number of trees (n_estimators): The number of trees in the forest. • Maximum depth: Maximum depth of each tree in the forest. • Minimum samples split: Minimum number of samples required to split an internal node. • Feature subsampling (max_features): The number of features to consider when looking for the best split.

Table 4: Results of hyper-parameter tuning

Model	Hyper-parameter pool	Best hyperparameters	Accuracy	Precision	Recall	f1-score
Logistic Regression	`'logistic__C': [0.001, 0.01, 0.1, 1, 10, 100], 'logistic__solver': ['liblinear', 'lbfgs', 'sag',`	`'logistic__C': 1, 'logistic__penalty': 'l2', 'logistic__solver': 'liblinear'`	0.81	0.79	0.81	0.81

	'newton-cg'], 'logistic__penalty': ['l1', 'l2']					
Support Vector Machines	'svc__kernel': ['linear', 'rbf', 'poly', 'sigmoid'], 'svc__C': [0.1, 1, 10], 'svc__gamma': [0.1, 1, 10]	'svc__C': 10, 'svc__gamma': 0.1, 'svc__kernel': 'rbf'	0.88	0.84	0.90	0.88
Naive Bayes	'gnb__var_smoothing': [1e-9, 1e-7, 1e-5]	'gnb__var_smoothing': 1e-09	0.79	0.79	0.74	0.79
KNN	'knn__n_neighbors': [3, 5, 7], 'knn__weights': ['uniform', 'distance'] 'knn__metric': ['euclidean', 'manhattan']	'knn__metric': 'manhattan', 'knn__n_neighbors': 7, 'knn__weights': 'uniform'}	0.88	0.84	0.90	0.88
Decision Tree	'tree__max_depth': [5, 10, 30], 'tree__min_samples_split': [2, 10], 'tree__min_samples_leaf': [2, 10], 'tree__criterion': ['gini', 'entropy']	'tree__criterion': 'entropy', 'tree__max_depth': 5, 'tree__min_samples_leaf': 10, 'tree__min_samples_split': 2	0.96	0.93	0.98	0.96

Random Forest	'forest__n_estimators': [10, 30, 100], 'forest__max_depth': [5, 10, 30], 'forest__min_samples_split': [2, 10], 'forest__min_samples_leaf': [2, 10],	'forest__max_depth': 10, 'forest__min_samples_leaf': 2, 'forest__min_samples_split': 2, 'forest__n_estimators': 100	0.96	0.95	0.95	0.96

Best model:

While each model shows different strengths, certain limitations deter their suitability for this particular scenario. Logistic Regression, although interpretable and computationally efficient, may lack the capacity to capture complex relationships present in the data, making it less effective for intricate patterns. SVMs with the selected hyperparameters could potentially provide good generalisation, but the 'poly' kernel might not universally fit all datasets due to its complexity and susceptibility to overfitting. Naive Bayes assumes feature independence, which could compromise performance when this assumption is violated in real-world data. KNN can capture complex relationships but faces challenges in high-dimensional spaces and may become computationally expensive with larger datasets. Decision Trees perform well in capturing nonlinear relationships and providing interpretability but might lack robustness and be prone to overfitting. Despite Decision Trees' limitations, Random Forest emerges as the best choice by overcoming overfitting issues encountered in a single Decision Tree while maintaining good interpretability and robustness against noise and outliers.

Table 4 third column gives the best hyper parameter set for the chosen model and Table 3 explains them.

The Random Forest model was constructed with an ensemble of 100 decision trees, each limited to a maximum depth of 10, and requiring a minimum of 2 samples for node splitting and leaf formation. During training, these 100 trees were built in the forest by aggregating predictions from various decision trees, contributing to the final prediction by averaging their individual outputs.

Classification Report:

	precision	recall	f1-score
0	0.98	0.91	0.94
1	0.90	0.98	0.94
accuracy			0.94
macro avg	0.94	0.94	0.94
weighted avg	0.94	0.94	0.94

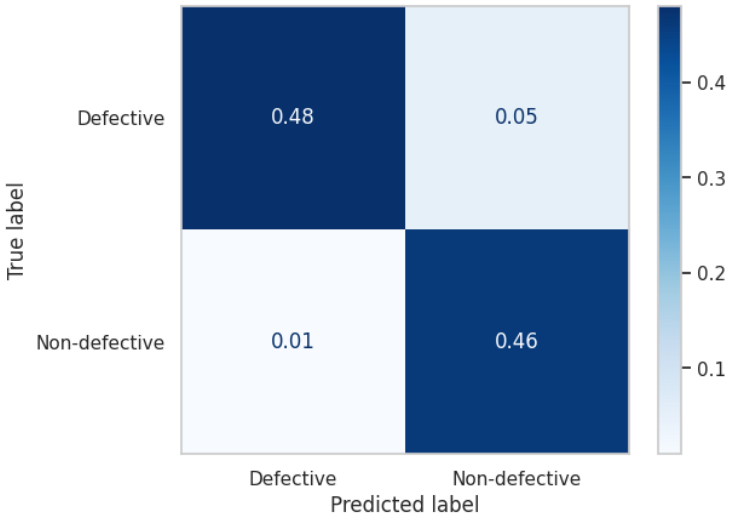


Figure 16: Evaluation with test data - Confusion Matrix and Classification report of the Random Forest Classifier

Figure 16 illustrates the model's performance metrics on test data. The balanced accuracy, 94%, reflects the percentage of correctly predicted instances. Precision and recall range between 90% to 98%, indicating a balanced identification of positive cases and capturing true positives. The F1-score, a blend of precision and recall, also reaches 94%, suggesting robust performance. Balanced accuracy and macro-averaged F1 scores confirm equilibrium across classes. Overall, the model exhibits strong performance in correctly identifying instances for both classes with balanced precision and recall.

Comparison with a naive baseline:

Classification Report:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	53
1	0.47	1.00	0.64	47
accuracy			0.47	100
macro avg	0.23	0.50	0.32	100
weighted avg	0.22	0.47	0.30	100

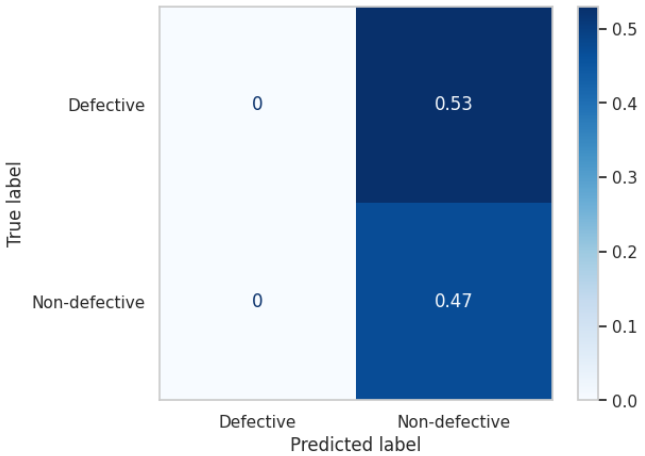


Figure 17: Random Forest Classifier comparison with an all test points positive baseline

Naive baseline approaches serve as benchmarks for evaluating complex models and demonstrating their effectiveness beyond simple majority class prediction. The evaluation of the model against a naive baseline approach, assuming all points as positive (Non defective), results in an accuracy of 47% with perfect recall (100%) but a notably low precision of 47%. , signifying the proportion of correctly identified positive cases among all cases predicted as positive.

5. Convolutional Neural Network Implementation

The image data was initially loaded from a CSV file to acquire labels and associated filenames. Images were then processed, resized to a uniform 100x100 pixel size, and pixel values were normalised between 0 and 1. The dataset was divided into training (80%), validation (10%), and test (20%) sets using a 72:8:20 ratio.

Layer (type)	Output Shape	Param #
input_22 (InputLayer)	[(None, 100, 100, 3)]	0
rescaling_21 (Rescaling)	(None, 100, 100, 3)	0
conv2d_86 (Conv2D)	(None, 100, 100, 64)	1792
max_pooling2d_86 (MaxPooling2D)	(None, 50, 50, 64)	0
conv2d_87 (Conv2D)	(None, 50, 50, 64)	36928
max_pooling2d_87 (MaxPooling2D)	(None, 25, 25, 64)	0
conv2d_88 (Conv2D)	(None, 25, 25, 64)	36928
max_pooling2d_88 (MaxPooling2D)	(None, 12, 12, 64)	0
conv2d_89 (Conv2D)	(None, 12, 12, 64)	36928
max_pooling2d_89 (MaxPooling2D)	(None, 6, 6, 64)	0
flatten_21 (Flatten)	(None, 2304)	0
dense_83 (Dense)	(None, 50)	115250
dense_84 (Dense)	(None, 50)	2550
dense_85 (Dense)	(None, 4)	204

Figure 18: Mode 1 Architecture

The CNN architecture involves four convolutional blocks, each with a Conv2D layer utilising 64 filters and (3, 3) kernel size, followed by MaxPooling2D for downsampling. Afterward, there's a final Conv2D and MaxPooling2D before flattening. Two dense blocks with 50 neurons and ReLU activation, possibly using 30% dropout, handle flattened features. The output layer, with a softmax

activation function, suits classification tasks with neurons aligning to classes. This architecture likely uses categorical cross-entropy for loss, ReLU for faster convergence, and dropout for regularisation.

Layer (type)	Output Shape	Param #
input_11 (InputLayer)	[(None, 100, 100, 3)]	0
rescaling_10 (Rescaling)	(None, 100, 100, 3)	0
conv2d_40 (Conv2D)	(None, 100, 100, 64)	1792
max_pooling2d_40 (MaxPooling2D)	(None, 50, 50, 64)	0
dropout_63 (Dropout)	(None, 50, 50, 64)	0
conv2d_41 (Conv2D)	(None, 50, 50, 64)	36928
max_pooling2d_41 (MaxPooling2D)	(None, 25, 25, 64)	0
dropout_64 (Dropout)	(None, 25, 25, 64)	0
conv2d_42 (Conv2D)	(None, 25, 25, 64)	36928
max_pooling2d_42 (MaxPooling2D)	(None, 12, 12, 64)	0
dropout_65 (Dropout)	(None, 12, 12, 64)	0
conv2d_43 (Conv2D)	(None, 12, 12, 64)	36928
max_pooling2d_43 (MaxPooling2D)	(None, 6, 6, 64)	0
flatten_10 (Flatten)	(None, 2304)	0
dropout_66 (Dropout)	(None, 2304)	0
dense_39 (Dense)	(None, 100)	230500
dropout_67 (Dropout)	(None, 100)	0
dense_40 (Dense)	(None, 100)	10100
dropout_68 (Dropout)	(None, 100)	0
dense_41 (Dense)	(None, 100)	10100
dropout_69 (Dropout)	(None, 100)	0
dense_42 (Dense)	(None, 4)	404

Figure 19: Mode 2 Architecture

The model integrates convolutional blocks with 64 filters, ReLU activation, and max pooling, culminating in three dense blocks with 100 neurons and ReLU activation, potentially applying 30% dropout. Featuring a softmax output for multi-class classification, this model differs by implementing extensive data augmentation (rotation, shifts, flips, zoom) for enhanced robustness and varied data

exposure. This augmentation aims to improve generalisation. Modified dense layer depth enhances model complexity. Employing 'adam' optimizer, 'sparse_categorical_crossentropy' loss, and 'accuracy' metric, this model augments data diversity for better generalisation compared to the previous architecture.

These are tested for several batch sizes(8,16,24) and epochs(10,15,20). Larger batch sizes converge faster but promote overfitting; smaller batches generalise.

Best model:

Model 1 with batch size 32 and epoch 20 got the best validation accuracy of 0.93

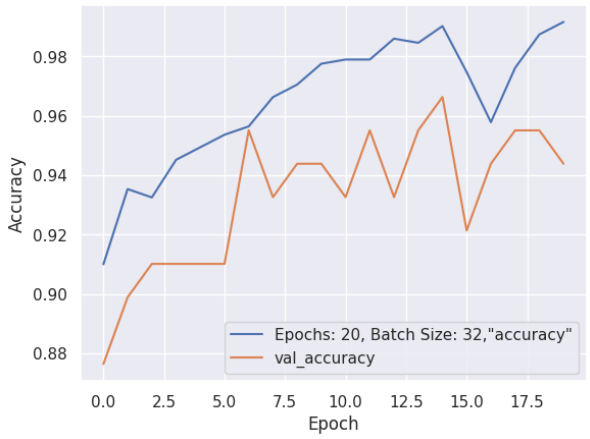


Figure 20:Train & validation accuracy over each epoch

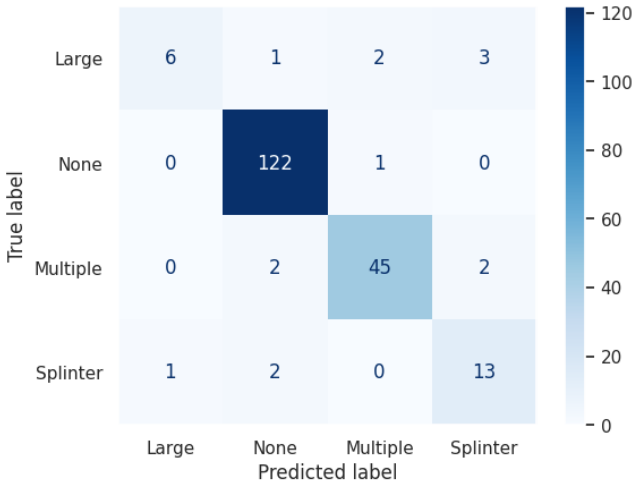
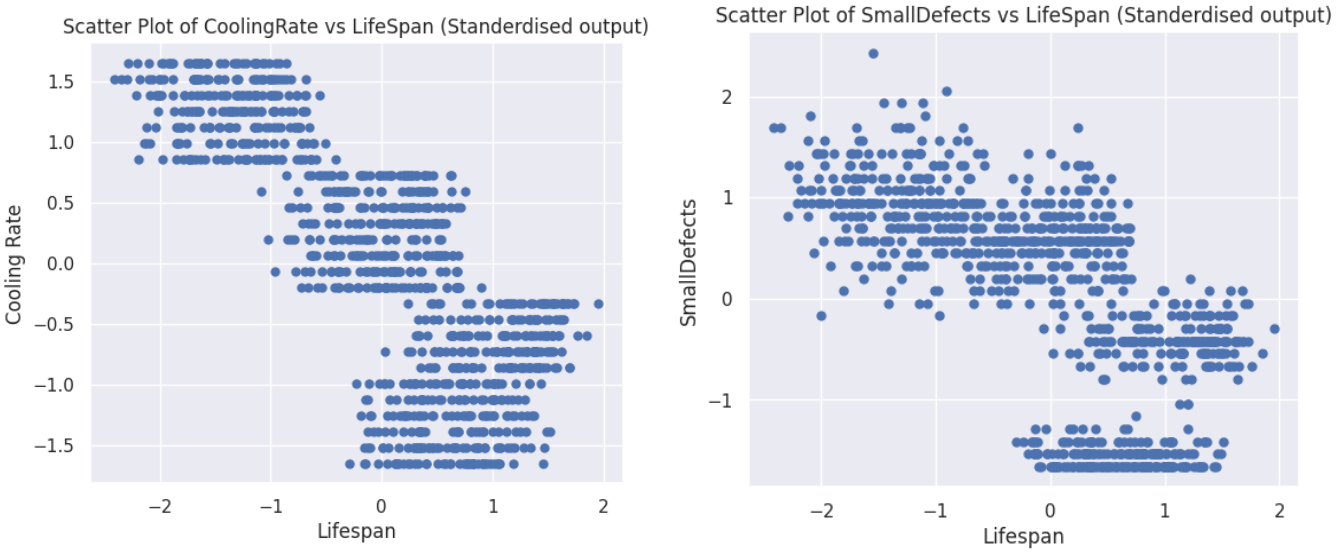


Figure 21: Confusion matrix for test data

It was tested with unseen data and achieved a 93% accuracy on test data, with balanced precision (87%) and recall (81%). The confusion matrix indicates that the model generally performs well in predicting 'None' and 'Splinter' categories but struggles more with 'Large' and 'Multiple' categories, often misclassifying instances between these classes. This suggests a need for further improvement in distinguishing between 'Large' and 'Multiple' categories to enhance the model's accuracy in these specific classifications.

6. Clustering Implementation

Clustering is a method that tries to split the data into similar groups, and can only be applied to numerical data. Numerical data were extracted from the dataset and scaled. coolingRate and smallDefects were highly correlated with Lifespan (figure 8). Therefore, they were selected to generate



the scatter plots.

Figure 22: Scatter plot of coolingRate and smallDefects with LifeSpan

The K-means clustering algorithm was utilised on coolingRate, SmallDefects and LifeSpan.To determine the optimal number of clusters three evaluation methods; visual inspection, silhouette score, and the elbow method were used.

6.1 Visual Inspection

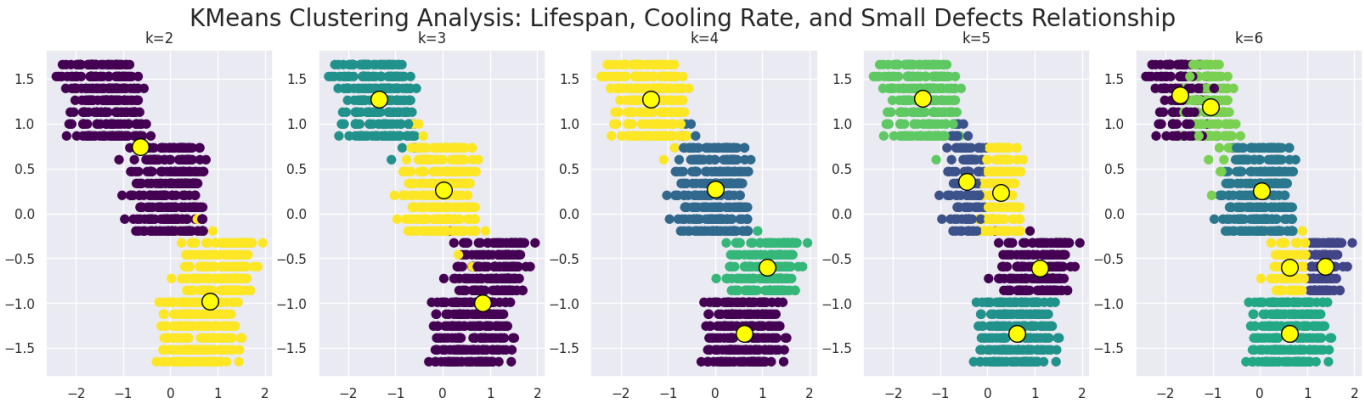


Figure 23: KMeans Clustering of "coolingRate" and "LifeSpan"

The plots illustrate the partitioning of the data points with clusters increasing from 2 to 6. Cluster centroids are marked as yellow circles. Based on figure 23, the clustering with k=2 appears more suitable due to well-separation of clusters observed in the scatter plots.

6.2 Silhouette Score

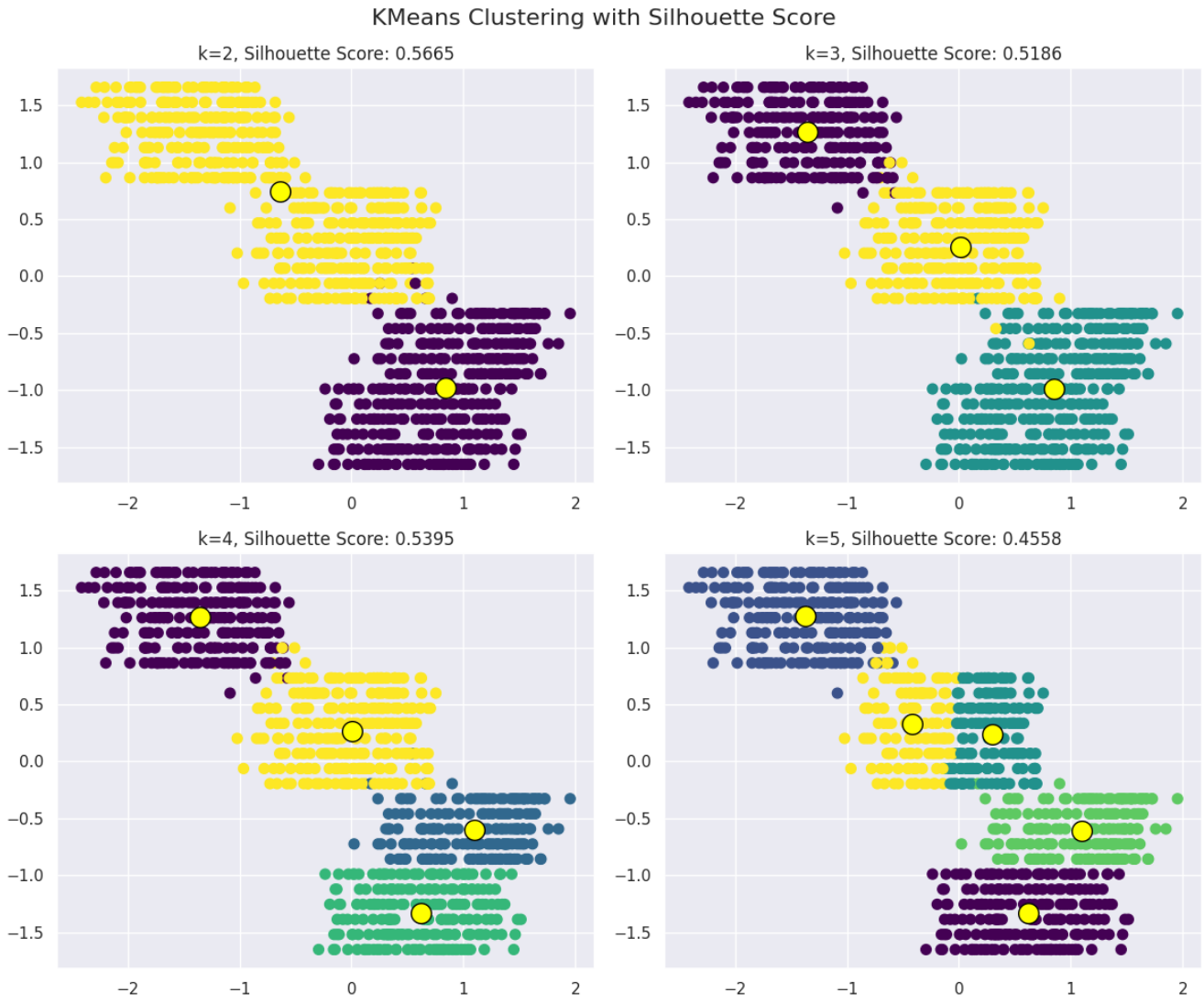


Figure 24: KMeans Clustering evaluation by Silhouette Score

The silhouette score ranges from -1 to 1, indicating cluster separation:

- Close to +1: Good separation between clusters.
- Around 0: Sample lies on cluster boundaries.
- Close to -1: Incorrect clustering.

The highest score (0.5665) is observed for k=2, suggesting better-defined separation than other 'k' values.

6.3 Elbow Method

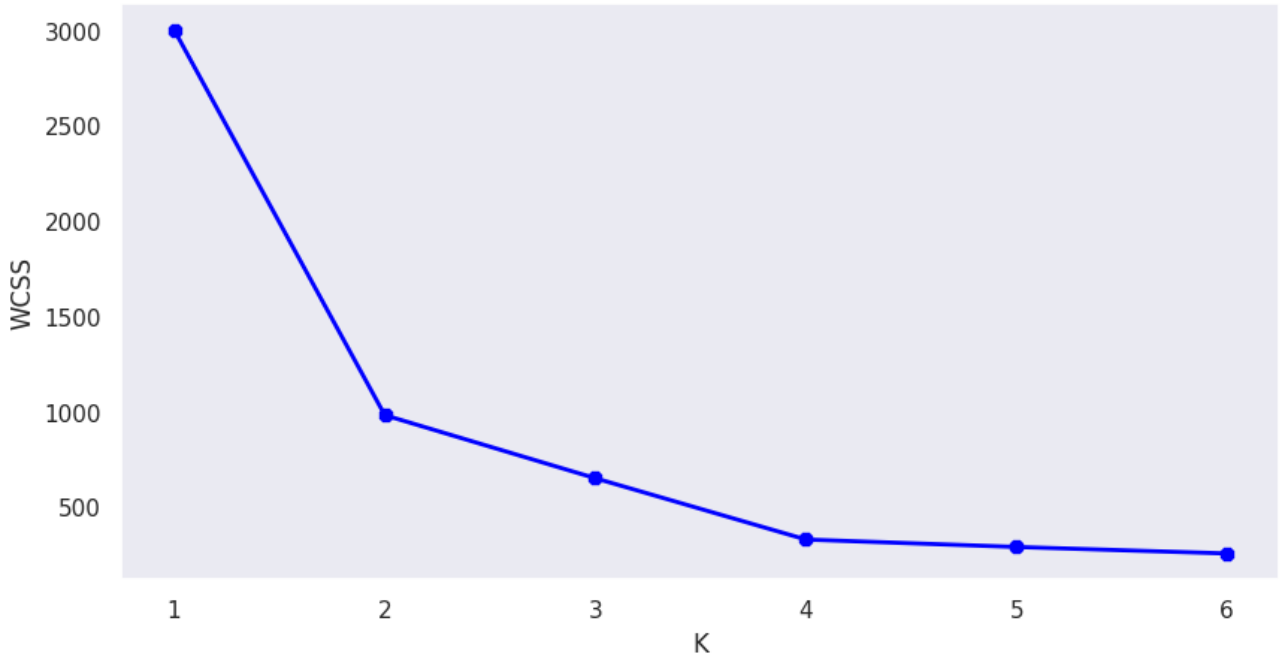


Figure 25: within-cluster sum of squares(WCSS) vs number of clusters(K)

The elbow method assesses the optimal cluster count by analysing the within-cluster sum of squares (WCSS) against the number of clusters(K). The "elbow" or the bending point represents the suitable k value. In this case, the exact point is ambiguous due to multiple possible bends in the plot.

Considering both visual inspection and quantitative evaluation methods, k=2 emerges as the preferred choice for clustering.

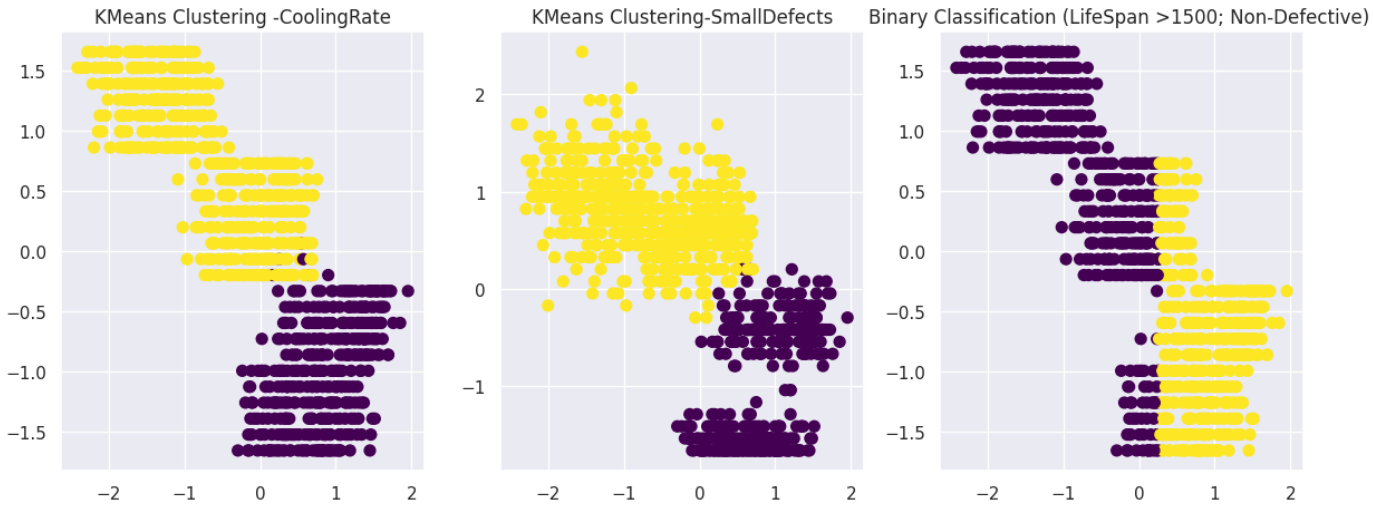


Figure 26: Comparison of KMeans Clustering and Binary Classification

Figure 26 illustrates the clusters formed by K-means and binary classification, portraying how they divide the data into two distinct groups. K-means exhibits a better separation between the clusters, contrasting with binary classification which uses a fixed threshold of 1500. This visualisation shows insightful relationships: for standardised coolingRate values ranging from 0 to -2, there is a tendency towards higher Lifespan, whereas from 0 to 2, the Lifespan tends to be lower. Similarly, standardised smallDefects from 0 to -2 tend to exhibit higher Lifespan, while 0 to 2 tends to have lower Lifespan. The K-means clustering is effective over binary classification in this context.

7. Recommendations

Based on the comprehensive evaluation of the regression and classification models for predicting the lifetime of metal parts, the recommended model is the Random Forest regression. It demonstrates exceptional performance with an R-squared score of approximately 0.9799 and closely aligned MSE values between validation and testing sets. This signifies the model's ability to explain nearly 97% of the dataset's variability and its accurate predictions, indicative of minimal overfitting. In contrast, while the Binary Classification model shows robust performance with high balanced accuracy, precision, recall, and F1-score, the superior explanatory power and minimal overfitting exhibited by the Random Forest regression make it the preferred choice for practical application in predicting the lifetime of metal parts.

CNN demonstrates promising accuracy, particularly in predicting 'None' and 'Splinter' categories, while facing challenges in differentiating 'Large' and 'Multiple' defects based on the confusion matrix. Despite these challenges, the CNN model showcases notable potential for practical deployment. However, to enhance its performance further, model refinements focusing on improving the differentiation between 'Large' and 'Multiple' defect categories, along with augmenting the dataset with more diverse examples for these categories, could be beneficial. Despite these areas for improvement, the CNN model's accuracy and balanced precision-recall trade-off make it a suitable candidate for practical deployment.

8. Reflections

The Regression Implementation, adopting wrapper methods for feature selection could have been beneficial to identify the best feature combinations without overfitting concerns, contrary to the filter method. Additionally, employing dimensionality reduction techniques could have enhanced feature engineering. A comprehensive approach involving all levels of hyperparameter tuning might have further optimised model performance. Moreover, exploring transfer learning CNN algorithms such as VGG16 could have offered insights into potential improvements for the CNN model. Lastly, considering alternative clustering methods like agglomerative clustering might have provided additional insights and potentially improved clustering results. These modifications, focusing on feature selection, dimensionality reduction, comprehensive hyperparameter tuning, and exploring different algorithms, could have potentially refined the models' performance in their respective tasks.

9. References

COMP1801 Lecture notes and tutorials.

Cantrell, E. (2023) *Prioritizing encoding timing: Before or after train-test split? - machine learning*, *CopyProgramming*. Available at: https://copyprogramming.com/howto/encoding-before-vs-after-train-test-split?utm_content=cmp-true (Accessed: 27 November 2023).

Hastie, T., Tibshirani, R., Friedman, J.H. and Friedman, J.H., 2009. *The elements of statistical learning: data mining, inference, and prediction* (Vol. 2, pp. 1-758). New York: springer.