Classification of Dry Beans Using k-Nearest Neighbours and Classification Trees

Franco Uys (23583584)
Stellenbosch University

***Abstract*—This report presents an analysis of the detection of data quality issues and how they relate to the development process of two classification models for classifying dry bean types. The project aimed to identify and rectify data quality issues within a large dataset comprising 13 611 instances and 19 descriptive features, followed by the implementation of k-Nearest Neighbours (k-NN) and Classification Tree classifiers to predict the different bean types. An exploratory data analysis (EDA) is performed to characterize the features of the dataset and to identify quality issues such as missing values, outliers, incorrect data entries, and potential class skew. Different preprocessing techniques such as data imputation and removal are used to address these issues. After the data is properly formatted for the models both the k-NN and the Classification Tree are trained using 5-fold cross-validation, with hyperparameter tuning. The models are evaluated based on the weighted F1-score; both achieve a weighted F1-score of 0.97. Furthermore, base models are developed without these data quality corrections and scored 0.79 and 0.98 for the k-NN and the Classification Tree, respectively. These results demonstrate the inherent relationship between the chosen machine learning model and the quality of the data that they receive. This study provides insights for handling model-specific data quality issues and optimizing machine learning models for high-quality predictions.**

***Index Terms*—Data Quality, Machine Learning, k-Nearest Neighbours, Classification Trees, Classification, Cross-Validation**

I. Introduction

The performance of machine learning models is inherently related to the quality of the data they are trained on. This study aims to investigate the data quality issues present in the Dry Beans dataset and how the data quality issues relate to two specific classification models. The classification models considered are the k-NN [1] and the Classification Tree, which is a special case of the general Decision Tree proposed by Quinlan [2]. The dataset consists of 13 611 instances and 19 descriptive features and one target variable: the class of the dry bean.

To achieve the objectives of this study, an EDA is conducted, examining the data for erroneous data, class skew, potential outliers, and missing values. Koklu [3] also deals with the classification of dry beans, and tests are performed to verify whether the relationships between the variables in their dataset are consistent with those in the dataset of this study. After confirming that the variables were calculated identically, many of the data quality issues are easily resolved.

After performing the EDA, the data quality issues are addressed. Where possible, the missing and erroneous data are calculated using the formulae presented in Koklu, the remaining missing values are either imputed with the median, the mean or removed. The outliers are identified using Tukey's Rule [4] on the univariate distributions of each feature with respect to the target class.

The dataset is split into an 80-20 train-test split, and the models are trained using 5-fold cross-validation [5] on the training set. For benchmarking, both the processed and unprocessed data are used for training and evaluation. Hyperparameter tuning is performed during the 5-fold cross-validation using a grid search of parameter configurations. The performance of the models is measured by the weighted F1-score [6] over a 5-fold cross-validation over the test set, using the optimal parameters found over the training set.

II. Background

1) Overview of supervised classification: Supervised machine learning aims at learning a functional mapping from the input space composed of the descriptive features of the dataset to the output space, that consists of the target value. An arbitrary classifier is generally represented as:

$$f : \mathbb{R}^p \rightarrow \{1, \ldots, K\}, \quad \text{where } p \in \mathbb{Z}^+ \text{ and } K \in \mathbb{Z}^+ \tag{1}$$

where K is the number of classes the target variable can take, $\mathbb{R}^p$ represents a p-dimensional real-valued input space, and f is the classifier. In the context of the dataset presented in this study p = 19 and K = 7. To better facilitate the learning process of f, various techniques are used to improve the quality of the dataset, gain insights, and enhance model performance. Techniques relevant to this study and their implementations are discussed in the following sections.

2) Nomenclature:

- Overfitting [7]: The production of an analysis that corresponds too closely or exactly to a particular set of data, failing to fit additional data or predict future observations reliably.
- Underfitting: The production of an analysis that fails to generalize well to unseen data because it could not effectively learn the functional mapping on the training set
- Feature Leakage: A problem where the model learns the mapping from the input space to the output space and is evaluated on target variables that it has already been exposed to, leading to unreliable results.

A. Distance Metrics

Distance metrics play a pivotal role in distance-based classification algorithms such as k-NN. They allow for measuring the similarity and, by extension, the dissimilarity of two data points in a dataset.

One versatile and generalizable distance metric is the Minkowski distance defined in [8]. It defines the distance between two vectors $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_n$ in a n-dimensional space as:

$$D(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}} \tag{2}$$

where $p \geq 1$ is an adjustable parameter that changes the behaviour of the Minkowski distance. For example, larger values of p place a greater emphasis on distances between points since the element-wise distance is raised to the p^{th} power. Special cases of the Minkowski distance include the Manhattan distance (3), known as the L_1 distance with $p = 1$, and the Euclidean distance (4), known as the L_2 distance with $p = 2$.

$$D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i| \tag{3}$$

$$D(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} (x_i - y_i)^2 \right)^{\frac{1}{2}} \tag{4}$$

In the context of k-NN, the adjustable Minkowski distance measures the similarity of different points in the dataset. By adjusting the value of p one can control the sensitivity of the distance metric to outliers.

B. k-NN

k-NN is a non-parametric, instance-based classifier that is commonly used in supervised tasks. k-NN is non-parametric because it does not make any assumptions about the underlying distribution of the data. Instead, the only inductive bias is that it relies on a specific distance metric to measure the similarity between two data points.

The main idea of a k-NN classifier is to classify a new observation based on the majority class among its k nearest neighbors from the training set. Given a training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, where $\mathbf{x}_i \in \mathbb{R}^p$ represents the feature vectors and $y_i \in \{1, ..., K\}$ represents the class labels, the prediction for a new observation $\mathbf{x}$ is given by:

$$\hat{y}(\mathbf{x}) = \arg \max_{y \in \{1,2,\ldots,K\}} \sum_{i \in \mathcal{N}_k(\mathbf{x})} \mathbb{I}(y_i = y) \tag{5}$$

where $\mathcal{N}_k(x)$ indicates the indices of the nearest neighbours of $\mathbf{x}$ and $\mathbb{I}(\cdot)$ indicates the indicator function such that

$$\mathbb{I}(y_i = y) = \begin{cases} 1 & \text{if } y_i = y \\ 0 & \text{otherwise} \end{cases}$$

Algorithm 1 k-NN

```
Require: Training data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, new observation $\mathbf{x}$, integer $k$, distance metric $d(\cdot, \cdot)$
Ensure: Predicted class $\hat{y}(\mathbf{x})$ for the new observation $\mathbf{x}$
1: for each training instance $(\mathbf{x}_i, y_i) \in \mathcal{D}$ do
2:     Compute the distance $d_i = d(\mathbf{x}, \mathbf{x}_i)$
3: end for
4: Sort all distances $\{d_i\}$ in ascending order
5: Select the indices of the $k$ smallest distances to form the set $\mathcal{N}_k(\mathbf{x})$
6: Compute the predicted class using (5):
7: return $\hat{y}(\mathbf{x})$
```

The k-NN algorithm (Algorithm 1) relies heavily on the choice of k, which plays a significant role in its ability to generalize to unseen data. A small k can make the model more sensitive to noise, while a large k can smooth out the majority vote by including neighbors that are further away, leading to underfitting. Hyperparameter tuning on a validation set is typically used to select the correct value of k.

Engelbrecht describes in [8] that the k-NN classification algorithm views target outliers as noise since they can affect the outcome but do not change the distance-based calculations. The algorithm is also robust to feature-based outliers since lower values of k will likely not select the outlier as a nearest neighbor, and for larger k, the effect of an outlier is mitigated through majority voting. The algorithm is sensitive to noise, especially for smaller values of k, since noisy neighbors become more likely. Missing values can either be excluded from the majority vote in (5) or imputed. Scaling is crucial since distance-based metrics rely heavily on element-wise distances.

For further reading on the algorithm, readers can consult The Elements of Statistical Learning [9].

C. Decision Trees and CART Algorithm

Decision Trees are another useful supervised machine learning tool that creates the functional mapping described in (1) through the use of Information Theoretic concepts such as Entropy and Information Gain [2]. The tree recursively divides the feature space into subsets, making each subset more homogeneous with respect to the target variable.

Homogeneity refers to the uniformity of instances within a subset concerning their class instances. A perfectly homogeneous subset has instances with the same class label. Various metrics are used to quantify homogeneity, including information gain, the Gini index [10], and entropy.

The Gini index is a metric used to measure the impurity of a dataset. It quantifies how often a randomly selected element from a subset would be incorrectly labeled if it were to be labeled according to the distribution of class labels within the subset.. The Gini index $G(T)$ is defined as follows:

$$G(T) = 1 - \sum_{i=1}^{C} p_i^2 \tag{6}$$

where p_i is the proportion of instances belonging to class i in the dataset T, and C is the total number of classes.

Entropy is a metric that measures randomness in the dataset and can be used to determine impurity in a dataset with respect to a class label. The entropy $H(T)$ of a dataset T is defined as:

$$H(T) = -\sum_{i=1}^{C} p_i \log_2(p_i) \tag{7}$$

where p_i is the proportion of instances belonging to class i in the dataset T.

The algorithm used in this study to construct the classification trees is the Classification and Regression Tree (CART) algorithm [9]. It is a binary recursive splitting algorithm that divides the data into two subsets based on a criterion that maximizes the homogeneity of the resulting subsets. It continues this process until predefined stopping criteria are met or every child node is pure.

The CART algorithm (Algorithm 2) can use any metric that measures the homogeneity of its resulting subsets; these include the Gini index and Entropy (7). The CART algorithm does not automatically prune the trees it induces, so it is not robust to outliers. Furthermore, the CART algorithm is not robust to missing values or target variable class skew because it does not have a mechanism to handle these cases.

The following pseudocode outlines the basic steps involved in constructing a Decision Tree using the CART algorithm:

Algorithm 2 Decision Tree Construction using CART

```
Require: Training data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, maximum tree depth $d_{\text{max}}$, minimum samples per leaf $m_{\text{min}}$
Ensure: A trained decision tree model
 1: function BUILDTREE($\mathcal{D}, d$)
 2:     if stopping criteria met (depth $d \geq d_{\text{max}}$ or samples $< m_{\text{min}}$) then
 3:         return leaf node with majority class label
 4:     end if
 5:     Find the best split for data $\mathcal{D}$ using Gini index or entropy
 6:     Split $\mathcal{D}$ into $\mathcal{D}_{\text{left}}$ and $\mathcal{D}_{\text{right}}$ using the best split
 7:     Create internal node with the best split
 8:     BUILDTREE($\mathcal{D}_{\text{left}}, d + 1$)
 9:     BUILDTREE($\mathcal{D}_{\text{right}}, d + 1$)
10: end function
```

D. Evaluation Metrics and Validation Techniques

Evaluation metrics are used in machine learning to validate a model's performance and confirm if it is generalizable. Choosing the correct evaluation metrics is crucial for the type of model used and the goals set. The following are some evaluation metrics used in the report. Validation Techniques ensure that the evaluation metrics are robust and offer a true reflection of the model's performance on unseen data.

1) Evaluation Metrics:

a) F1 Score: The F1 score is a measure described as the harmonic mean of the proportion of actual positives the model correctly identifies and the proportion of predicted positives that are correct.

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{8}$$

where

$$P = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{9}$$

and

$$R = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}. \tag{10}$$

The F1 score ranges from 0 to 1, where 1 indicates perfect precision and recall, and 0 indicates the worst performance.

The weighted F1 score is a variation of the F1 score that accounts for the different sizes of each class by averaging the F1 scores for each class and weighting each class's score weighted by its proportion in the dataset.

b) ROC Curve and AUC: The Receiver Operating Characteristic (ROC) curve [11] is a graphical representation that represents the ability of a binary classifier system as its classification threshold is varied. Mathematically, the ROC curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) for various threshold settings.

The True Positive Rate (TPR), also known as sensitivity or recall (10) The False Positive Rate (FPR) is defined as:

$$\text{FPR} = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}}, \tag{11}$$

where False Positives represents the number of false positives and True Negatives represents the number of true negatives.

The Area Under the Curve (AUC) is the area underneath the ROC curve. An AUC value closer to one indicates better model performance.

c) Confusion Matrix: A confusion matrix [12] is a table used to describe the performance of a classification model. It represents various metrics of the classifier in a compact way that allows for the diagnosis of classification reliability. The confusion matrix provides the following metrics:

- True Positives: The number of correctly predicted positive instances.
- True Negatives: The number of correctly predicted negative instances.
- False Positives: The number of negative instances incorrectly predicted as positive.
- False Negatives: The number of positive instances incorrectly predicted as negative.

Derivatives of these components such as the F1 score assess reliably help reliably assess the performance of a classifier.

2) Validation Techniques:

a) k-Fold Cross-Validation: k-Fold Cross-Validation is a robust validation technique used to evaluate the generalisation capability of a model by partitioning the dataset into k equally sized folds. The model is trained on $k-1$ folds and validated on the remaining fold. This process is repeated k times, with each fold serving as the validation set once. Performance metrics are averaged over k iterations to provide a stable estimate of the model's performance [5].

Mathematically, if we have a dataset $\mathcal{D}$ with n samples, the dataset is split into k disjoint subsets $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_k\}$. For each fold i, the model is trained on $\mathcal{D} \setminus \mathcal{D}_i$ and validated on $\mathcal{D}_i$. The k-fold CV estimate is computed by averaging the error metrics, denoted by Error_i, over all folds, resulting in [5]:

$$\text{CV}_{(k)} = \frac{1}{k} \sum_{i=1}^{k} \text{Error}_i. \tag{12}$$

K-Fold Cross-Validation provides a more reliable estimate of model performance by reducing variance from the random selection of training and validation sets [5].

E. Data Preprocessing Techniques

1) Standardisation: Standardisation [13], also referred to as Z-score normalization, standardises the range of features in the dataset. In many machine learning algorithms, especially those based on distance metrics, features with larger ranges can disproportionately influence the model. To prevent this, the data must be standardised, which transforms the data to have zero mean and unit variance for each feature. Mathematically, for a feature x, the standardised value z is computed as:

$$z = \frac{x - \mu}{\sigma} \tag{13}$$

where μ is the mean of the feature and σ is the standard deviation of the feature. This transformation ensures that all features contribute equally to the model and effectively deals with skewed distributions.

2) Encoding Categorical Data: k-NN and classification trees require numerical input, meaning that categorical features in the dataset must be represented numerically. Methods for doing this are described below:

a) Label Encoding: In label encoding [14], an integer is assigned to each category, provided the number of existing categories is known.

b) One-Hot Encoding: One-hot encoding [14] transforms each category into a separate binary feature, resulting in a matrix of 0s and 1s, where the number of columns (or rows) is determined by the number of distinct values the categorical feature can take on.

3) Outlier Detection:

a) Tukey's Method: Tukey's method [4], also known as Tukey's fences, identifies outliers based on the interquartile range (IQR). The IQR is the range between the first quartile (Q_1) and the third quartile (Q_3) and represents the middle 50% of the data. Tukey's method defines outliers as data points that fall outside the following bounds

$$\text{Lower Bound} = Q_1 - k \times \text{IQR} \tag{14}$$

$$\text{Upper Bound} = Q_3 + k \times \text{IQR} \tag{15}$$

The value of k is adjusted depending on the severity of the outliers to classify; however, the convention is $k = 1.5$

4) Tomek Links: Tomek Links [15] is an undersampling technique that aims to remove overlapping data points between classes to remove noise and target variable skew. A Tomek Link between two instances, $(\mathbf{x}_i, y_i)$ and $(\mathbf{x}_j, y_j)$, from different classes exists if they are each other's nearest neighbors:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \min_{\mathbf{x}_k \in \mathcal{D}} d(\mathbf{x}_i, \mathbf{x}_k) \quad \text{and} \quad d(\mathbf{x}_j, \mathbf{x}_i) = \min_{\mathbf{x}_l \in \mathcal{D}} d(\mathbf{x}_j, \mathbf{x}_l)$$

where $d(\mathbf{x}_i, \mathbf{x}_j)$ represents the distance between instances $\mathbf{x}_i$ and $\mathbf{x}_j$, and $\mathcal{D}$ is the dataset. Plainly stated, a Tomek Link is formed by two points if they are closer to each other than any other point in the dataset.

Tomek Links are useful for cleaning class boundaries by removing ambiguous samples that are close to the decision boundary.

F. Shepard's Method

Shepard's Method [16] is a technique for interpolation that uses a weighted average of known values based on their inverse distances from the point of interest. The weights are inversely proportional to the power of the distance $d_i(x)$ between the point of interest and the known points:

$$w_i(x) = \frac{1}{d_i(x)^p}, \tag{16}$$

where $d_i(x)$ is the Euclidean distance between x and xi, and p is a positive parameter that controls the rate of decay of the weight with distance. In the context of k-NN, Shepard's method can be used to mitigate the effects of imbalanced classes as points close to the point of interest will be allocated a higher weight. This means that even if the majority of neighbours have a particular class that the element may be assigned to some other class depending on proximity.

G. Dimensionality Reduction and Visualization

Dimensionality reduction [17] techniques are essential tools in machine learning and data analysis, particularly for visualizing high-dimensional data in a more interpretable, low-dimensional space. They allow for some form of information preservation in the original high-dimemsional space and aims to represent that information in a low-dimensional context.

H. Simple Linear Regression

A linear regression model [9] aims to describe the relationship between a response variable y and one or more predictor variables x. In a regression model, both the predictor and the response variable must be numeric continuous values. In the simplest form, a simple linear regression model assumes a linear relationship between a single predictor variable x and a single response variable y. The model can be mathematically represented as follows:

$$y = \beta_0 + \beta_1 x + \epsilon \tag{17}$$

where:

- y is the response variable,
- x is the predictor variable,
- β_0 is the intercept, representing the expected value of y when $x = 0$,
- β_1 is the slope, indicating the change in y for a one-unit change in x,
- ϵ is the error term, representing the noise in the data that cannot be explained by x.

The goal of fitting a linear model is to estimate the parameters β_0 and β_1 such that the sum of squared residuals (differences between observed and predicted values) is minimized.

The coefficient of determination, R^2 [18], is often used to assess the goodness-of-fit of a linear model. It is defined as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{18}$$

where:

- y_i are the observed values,
- $\hat{y}_i$ are the predicted values from the linear model,
- $\bar{y}$ is the mean of the observed values.

The R^2 value is often interpreted as meaning the proportion of variance of the target variable explained by the predictor variable so a value of one indicates that the predictor variable perfectly explains the target

I. Kernel Density Estimation

Kernel Density Estimation (KDE) [19] is a non-parametric method to estimate the probability density function of a random variable. KDE provides a smooth estimate of the distribution of data points based on a kernel function that weights the contribution of each data point according to its distance from a given point.

The KDE of a dataset $\{x_1, x_2, \ldots, x_n\}$ is given by:

$$\hat{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{19}$$

where:

- $\hat{f}(x)$ is the estimated density at point x,
- n is the number of data points,
- h is the bandwidth, a smoothing parameter that determines the width of the kernel,
- K is the kernel function, which is usually a symmetric, non-negative function such as the Gaussian kernel.

The choice of kernel function and bandwidth h greatly influences the KDE. The bandwidth controls the smoothness of the estimated density: a small h results in a rough estimate, while a large h produces a smoother estimate.

III. METHODOLOGY

A. Data Characterization

Table I presents a preliminary table used to characterize features in the dataset. This table was generated with missing values were removed to, facilitate the search for other data abnormalities, such as outliers and erroneous data.

It is evident from Table I that the majority of the features are numeric, and three features are categorical: Constantness, Class, and Colour. Moreover, some entries need further investigation. For instance, ConvexArea has a minimum value of -30 and EquivDiameter has a maximum value of 3014441.24 despite the mean value of 475.25.

B. Feature Verification and Recalculation

Upon some further investigation it is clear that the features in Koklu's study largely overlap with those in this study [3], with the exception of *ShapeFactor6*, *SortOrder*, *ShapeFactor5*, and *Constantness*. To ensure that the features of the dataset in this study are formulated identically to those in Koklu, the features in Koklu that were present in this dataset and had reproducible formulae defined in Koklu were recalculated, and then compared with those those reported by Koklu. The relevant formulae and their corresponding features are as follows:

- **Equivalent Diameter (Ed):** The diameter of a circle having the same area as a bean seed area.

$$\text{Ed} = \sqrt{\frac{4A}{\pi}} \tag{20}$$

- **Solidity (S):** The ratio of the area of the bean to the area of its convex hull.

$$\text{S} = \frac{A}{C} \tag{21}$$

- **Extent (Ex):** The ratio of the pixels in the bounding box to the bean area.

$$\text{Ex} = \frac{A}{A_B} \tag{22}$$

- **Roundness (R):** Measures the roundness of an object.

$$\text{R} = \frac{4\pi A}{P^2} \tag{23}$$

- **Compactness (CO):** Measures the compactness of an object.

$$\text{CO} = \frac{\text{Ed}}{L} \tag{24}$$

- **Shape Factors:**

$$\text{ShapeFactor1 (SF1)} = \frac{L}{A} \tag{25}$$

$$\text{ShapeFactor2 (SF2)} = \frac{l}{A} \tag{26}$$

$$\text{ShapeFactor3 (SF3)} = \frac{A}{\left(\frac{L}{2} \times \frac{l}{2} \times \pi\right)} \tag{27}$$

$$\text{ShapeFactor4 (SF4)} = \frac{A}{\left(\frac{L}{2} \times \frac{l}{2} \times \pi\right)} \tag{28}$$

TABLE I: Table I provides an overview of the features in the dataset, including data types, the number of unique values, and statistical measures such as mean, standard deviation, minimum, maximum, and median. It highlights both numeric and categorical features, with notable variations in value ranges and distributions.

Feature	Data Type	Number of Unique Values	Mean	Standard Deviation	Min	Max	Median
Area	int64	12011	53048.28	29324.10	20420.00	254616.00	44652.00
Perimeter	float64	13351	855.28	214.29	524.74	1985.37	794.94
MajorAxisLength	float64	13543	320.14	85.69	183.60	738.86	296.88
MinorAxisLength	float64	13543	202.27	44.97	122.51	460.20	192.43
AspectRatio	float64	13543	1.58	0.25	1.02	2.43	1.55
Eccentricity	float64	13543	0.75	0.09	0.22	0.91	0.76
ConvexArea	int64	12066	53765.69	29778.01	-30.00	263261.00	45178.00
Constantness	int64	2	0.90	0.30	0.00	1.00	1.00
EquivDiameter	float64	12012	476.25	25836.87	0.16	3014441.24	238.44
Colour	object	4	–	–	–	–	–
Extent	float64	13529	0.75	0.05	0.56	0.87	0.76
Solidity	float64	13526	0.99	0.00	0.92	0.99	0.99
Roundness	float64	13543	0.87	0.06	0.49	0.99	0.88
Compactness	float64	13525	0.80	0.06	0.64	0.99	0.80
ShapeFactor1	float64	13543	0.01	0.00	0.00	0.01	0.01
ShapeFactor2	float64	13543	0.00	0.00	0.00	0.00	0.00
ShapeFactor3	float64	13543	0.64	0.10	0.41	0.97	0.64
ShapeFactor4	float64	13611	2.37	0.87	0.70	3.97	2.37
ShapeFactor5	float64	13543	1.00	0.00	0.95	1.00	1.00
ShapeFactor6	float64	13606	89.36	51.84	0.00	178.99	88.77
Class	object	7	–	–	–	–	–
Sort order	float64	13611	0.50	0.29	0.00	1.00	0.50

The rest of the features cannot be directly calculated or are explicitly measured, as specified in Koklu's study.

Table II shows that the data in this study adhere to the same formulae as those presented in Koklu's study, and indicates consistent feature calculations between the datasets

TABLE II: Validation of Feature Calculation Accuracy and Error Indices

Feature	Incorrect Indices	Correct Percentage
EquivDiameter	[2762, 4780, 5884]	99.98%
ConvexArea	[3941]	99.99%
AspectRatio	None	100.00%
Solidity	[3941]	99.99%
Roundness	None	100.00%
Compactness	[2762, 4780, 5884]	99.98%
ShapeFactor1	None	100.00%
ShapeFactor2	None	100.00%
ShapeFactor3	None	100.00%
ShapeFactor4	None	100.00%

C. Handling Missing Values and Erroneous Data

After verifying the calculations, the handling of the missing values and erroneous data becomes trivial. Upon further inspection of the indices that do not align with the formulae specifically for 'EquivDiameter', 'Compactness', 'Solidity', and 'ConvexArea'— as shown in Table IV it is clear that the values can be recalculated.

Since 'Compactness' (24) is a function of 'EquivDiameter' (20), which in turn is a function of the 'Area', it is challenging to determine the source of the error. The inspection reveals that the error originates from the incorrect calculation of 'EquivDiameter' (20), as shown in Table IV. Furthermore, Table IV also shows errors for 'ConvexArea' and 'Solidity' for the indicies shown in Table II. The negative value for 'ConvexArea' clearly indicates an error, as such a value is not sensible..

D. Regression Analysis for Convex Area

This means that the values for 'EquivDiameter' and 'Compactness' in Table IV can be recalculated using (24) and (24). However, since no formula was explicitly defined for `ConvexArea`, a different approach was used. A simple linear regression model is fit to the data, using `Area` as the predictor variable and `ConvexArea` as the response variable. The regression line shown in Figure 1 is defined by the equation:

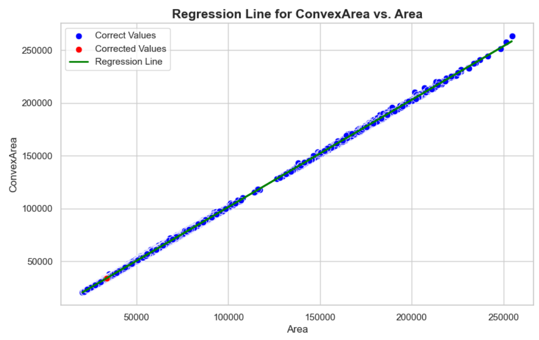


Fig. 1: Linear regression model of ConvexArea using Area as the predictor variable

$$\text{ConvexArea} = 1.0153 \times \text{Area} - 92.3497 \tag{29}$$

The model exhibited a high R^2, $R^2 = 0.9999$, indicating a nearly perfect linear relationship between `Area` and `ConvexArea`. This allowed for the imputation of the 'ConvexArea', and by extention Solidity (21), by using the known 'Area' value at the incorrect index.

TABLE III: Summary of Incorrect Feature Calculations and Data Entries Highlighting Anomalies in ‘Convex Area’ and ‘Solidity’

Index	Area	Convex Area	Equivalent Diameter	Colour	Compactness	Solidity
2762	45750	46099	24100.35	brown	0.791616	0.9924
4780	71368	72557	3014441.24	brown	0.779597	0.9832
5884	20464	20772	0.1614174	black	0.844016	0.9865
3941	33739	-30	207.262705	brown	0.810239	-1.1246

TABLE IV: Incorrect Feature Calculations and Data Entries

Index	Area	Convex Area	Equivalent Diameter	Colour	Compactness	Solidity
2762	45750	46099	24100.35	brown	0.791616	0.9924
4780	71368	72557	3014441.24	brown	0.779597	0.9832
5884	20464	20772	0.1614174	black	0.844016	0.9865
3941	33739	-30	207.262705	brown	0.810239	-1.1246

Referring to Table V it is seen that there are also 18 missing values for the ‘Compactness‘ column, to address this, the values of ‘EquivDiameter‘ corresponding to the missing indices of ‘Compactness‘ are used in (24).

TABLE V: Number of Missing Values for Each Feature

Feature	Number of Missing Values
Colour	6
Extent	6
Compactness	18
ShapeFactor6	5
Class	17

E. Imputation Strategy for Missing Values

For the missing values present in the ‘Extent‘ column, the ‘Extent‘ feature is a function of ‘Area‘ and an attempt is made to perform a regression on ‘Area‘, but as shown in Figure 2, there is no relationship between these two variables. Instead, using the Kernel Density Estimation plot in Figure 3, it is decided to use the median for imputation because the distribution is slightly skewed to the right and the median is robust to this. Similarly, for ‘ShapeFactor6‘, the mean is used for imputation as Figure 4 indicates a uniform distribution.

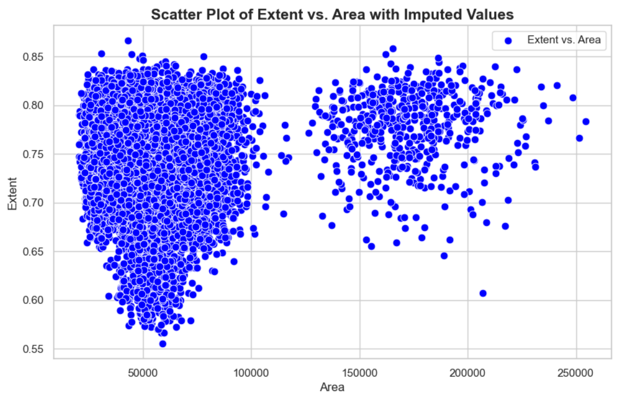


Fig. 2: Scatter plot showing no significant relationship between ‘Extent‘ and ‘Area‘.

Finally, ‘Colour‘ feature is imputed with the mode value, which is brown. The missing values for the ‘Class‘ feature are dropped since it is the target variable and noise in the target variable affects the model more than noise in a single feature in a multivariate setup. The effects of noise in the target variable cannot be smoothed out as they can be for the feature variables, because errors in the target variable directly distort the learning process.

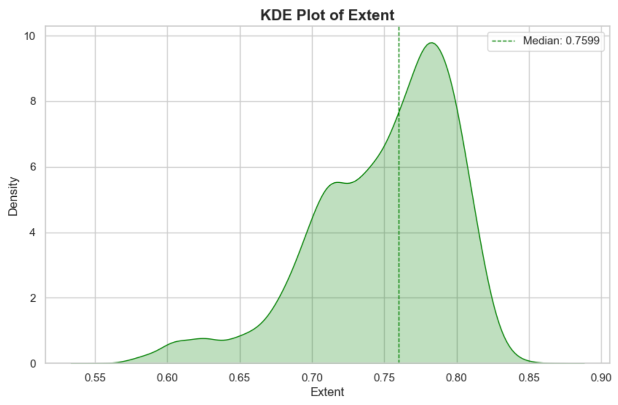


Fig. 3: Kernel density plot for ‘Extent‘, showing a slightly right-skewed distribution. The dashed green line is the median

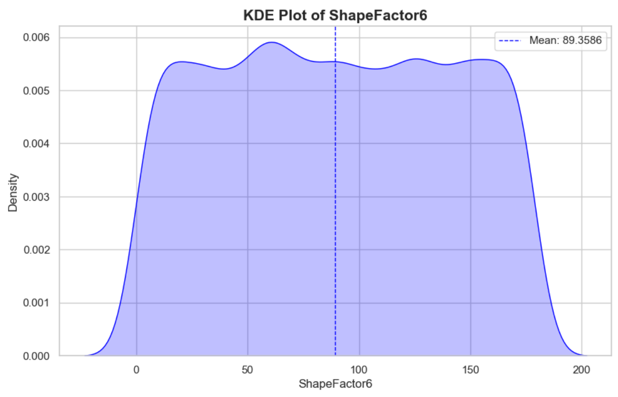


Fig. 4: Kernel density plot for ‘ShapeFactor6‘, showing a uniform distribution. The dashed blue line is the mean

F. Handling Target Variable Skew

To evaluate the dataset for outliers and class distribution skewness, a bar plot of the class distribution was created, as shown in Figure 5. The plot illustrates the number of instances in each class after the removal of missing values. Figure 5 shows a significant class imbalance. The ‘DERMASON‘ class has the highest number of instances, followed by ‘SIRA‘ and ‘SEKER‘. In contrast, the ‘BOMBAY‘ class has the fewest instances. This imbalance affects the modelling process of the Classification Tree and k-NN algorithms.

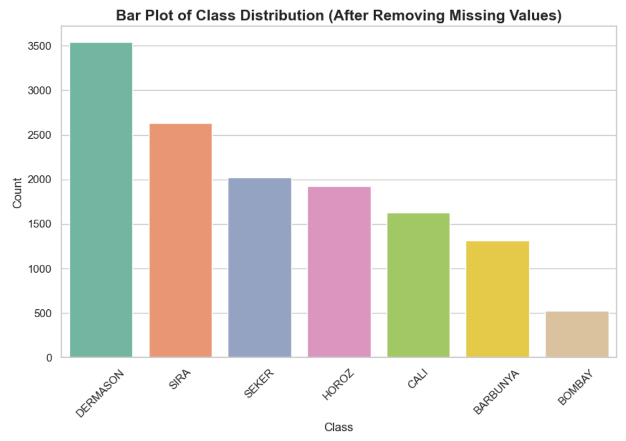


Fig. 5: Bar plot of class distribution after removing missing values. The plot shows the number of instances for each class in the dataset, indicating a significant class imbalance with ’DERMASON’ having the most instances and ’BOMBAY’ having the fewest.

G. Data Splitting and Preparation

The dataset is divided into training and testing subsets using an 80-20 split ratio. A stratified approach is employed to maintain the distribution of the target variable, ‘Class‘, across both subsets. This ensures that each subset will be a true reflection of the dataset as a whole, which is important because it ensures reliability of each model performance. The training set is used for model training and hyperparameter tuning, and the test set is reserved for final performance evaluation.

H. Outlier Detection and Handling

Outlier detection is performed only on the training set to ensure that the test set remains an accurate representation of the original dataset. To identify feature-based outliers, Tukey’s Rule is applied to each numeric feature within each target class. This method calculates the interquartile range (IQR) for each feature and defines outliers as data points falling outside the range defined by 1.5 times the IQR below the first quartile (Q1) and above the third quartile (Q3).

The distributions of the features with respect to each class, along with the identified outliers, are visualized using histograms. Figure 6 provides an example of these distributions for the ‘Area‘ feature. Instances identified as outliers are removed only for Classification Trees since the k-NN algorithm is inherently robust to outliers.

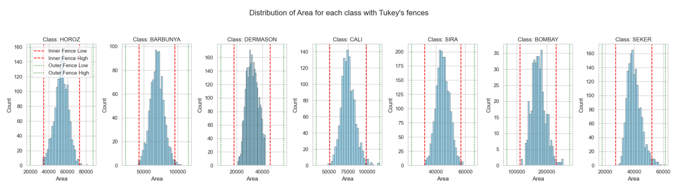


Fig. 6: Feature distributions before and after outlier removal. The inner fences, indicated by red dashed lines, represent 1.5 times the IQR below the first quartile (Q1) and above the third quartile (Q3). The outer fences, shown as green dotted lines, represent 3 times the IQR from Q1 and Q3.

I. Handling Class Imbalance

Given that there is a class skew in the target variable, two approaches were used to address this:

- **Tomek Links:** This technique is employed to undersample the majority class by removing examples that are deemed redundant. Tomek links are applied to both the k-NN and Classification Tree models to create a more balanced training set.
- **Weighted Approaches:** For k-NN, the `weights` parameter is set to `distance`, implementing Shepard’s method, which gives closer neighbors more influence in the classification. For the Classification Tree, a class-weighted criterion is used by setting `class_weight` to `balanced`, adjusting weights inversely proportional to class frequencies in the training data.

J. Modelling and Evaluation

Two models are developed for this study: k-NN and a Classification Tree. The data is only scaled for the k-NN model as the Classification Tree is inherently not sensitive to unscaled data. Model training and hyperparameter tuning is conducted using k-fold cross-validation on the training set, a grid search approach was used to identify the best parameters. Afterward, the models are evaluated on a test set using k-fold cross-validation with the best parameters found after the hyperparameter search. The main evaluation metrics used were the confusion matrix and the ROC curve.

K. Baseline

In both cases a baseline model is also developed, following the same procedure with the exception of data preprocessing and hyperparameter tuning. This is done to ensure a point of reference and allows for evaluating the impact of the data quality issues had on model performance.

IV. Empirical Procedure

A. Experimental Setup

To evaluate the performance of the models, a series of tests are performed that ensured each model’s results are a true reflection of their performance on unseen data. This involved splitting the data into a training and test sets which accurately reflected the distribution of the target variable and using cross-validation techniques, and avoiding feature leakage.

B. Cross-Validation and Hyperparameter Tuning

Hyperparameter tuning for both the k-NN and Classification Tree models is conducted using 5-fold cross-validation on the training set. The motivation for $k = 5$ is to strike a balance in the computational expense and the reliability of the performance evaluation.

1) k-NN: For the k-NN model, the hyperparameters that were tuned are shown in VI:

TABLE VI: k-NN Hyperparameters and Their Search Ranges

Hyperparameter	**Search Range**
`n_neighbors`	Odd values from 1 to $\sqrt{n}$
`p`	{1 (Manhattan), 2 (Euclidean)}
`weights`	{`uniform`, `distance`}

The parameter grid searched includes odd values for `n_neighbors` from 1 to the square root of the number of samples, `p` values of 1 (Manhattan distance) and 2 (Euclidean distance), and `weights` options of `uniform` and `distance`.

2) Classification Tree: For the Classification Tree model, the hyperparameters that were tuned are shown in VII:

TABLE VII: Classification Tree Hyperparameters and Their Search Ranges

Hyperparameter	**Search Range**
`criterion`	{`gini`, `entropy`}
`max_depth`	{None, 20, 30}
`min_samples_split`	{2, 10, 20}
`min_samples_leaf`	{1, 5, 10}
`class_weight`	{None, `balanced`}

A comprehensive grid search is performed over these parameters to identify the optimal configuration for the Classification Tree model.

C. Model Evaluation on Test Set

After identifying the best hyperparameters through cross-validation, the models are retrained on the entire training set using these optimal settings. The final evaluation is then conducted on the test set to assess the models' generalisation performance. The same scaling and class balancing techniques applied to the training data are also applied to the test data to ensure consistency. The evaluation on the test set is also done using 5-fold cross-validation to ensure reliability of results.

Performance metrics used for evaluation include the confusion matrix, ROC curve, and weighted F1-score. The weighted F1-score is the primary metric for hyperparameter optimisation, as it effectively balances precision and recall across all classes, particularly in the presence of class imbalance.

V. Research Results

A. Discussion of k-NN Results

Improvements observed in the k-NN model when using Tomek Links and Shepard's method over the base model are as a result from several data preprocessing techniques applied to enhance data quality issues and mitigate class imbalance. The base k-NN model, which does not use any scaling or preprocessing methods, achieves a weighted accuracy and F1-score of 0.79, as shown in Table VIII. In contrast, both the k-NN models with Tomek Links and Shepard's method achieve much higher weighted accuracy and F1-scores of 0.98, respectively. The following data quality issues contribute to the performance boost over the base model:

- **Handling Missing Values and Erroneous Data:** By imputing the missing values with the prescribed formulae in Koklu, the dataset was enriched and reflected an accurate representation of the relationships between the features. Furthermore by using the mean and median imputation methods, the same effect was achieved.
- **Handling Class Imbalance:** Class imbalance was a significant issue in the dataset, as evident from the class distribution shown in Figure 5. By using Tomek Links and Shepard's method to address the target class imbalance, the k-NN algorithm was able to effectively learn the functional mapping from the feature space to the target space.

TABLE VIII: Performance Metrics of k-NN Models, Including Base Model, Tomek Links, and Shepard's Method, Evaluated by Weighted Accuracy and Weighted F1-Score.

Model	**Weighted Accuracy**	**Weighted F1-score**
k-NN Base	0.79	0.79
k-NN Tomek Links	0.98	0.98
k-NN Shepard's Method	0.98	0.98

The ROC and confusion matrix plots for the Tomek Links version of the k-NN model (Figures 9 and 10) ffurther demonstrate the enhanced model performance. These figures illustrate a clear improvement in the model's ability to distinguish between classes, compared to the base model's plots (Figures 7 and 8). Similar results obtained using Shepard's method, not shown due to their redundancy, also confirm that both techniques effectively address class imbalance and contribute to better model accuracy and reliability.

B. Classification Tree Results Discussion

The performance of the Classification Tree models, as shown in Table IX, demonstrates high accuracy and F1-score across all variations. The base Classification Tree model achieves a weighted accuracy and F1-score of 0.98, indicating that the model performs well even without additional preprocessing techniques. Both the Classification Tree models using Tomek Links and a weighted approach result in slightly lower weighted accuracy and F1-scores of 0.97. This minor decrease suggests that while the preprocessing methods are effective in managing data quality and class imbalance, they

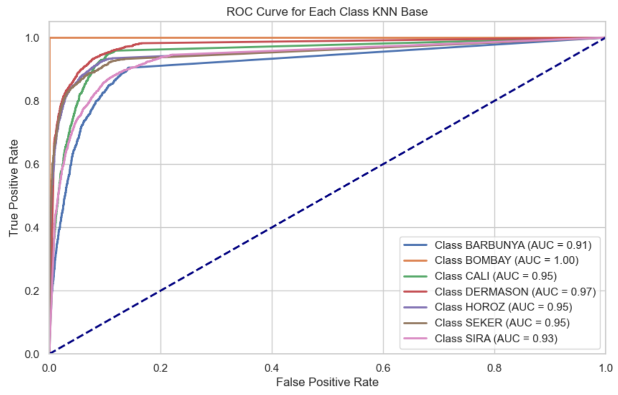
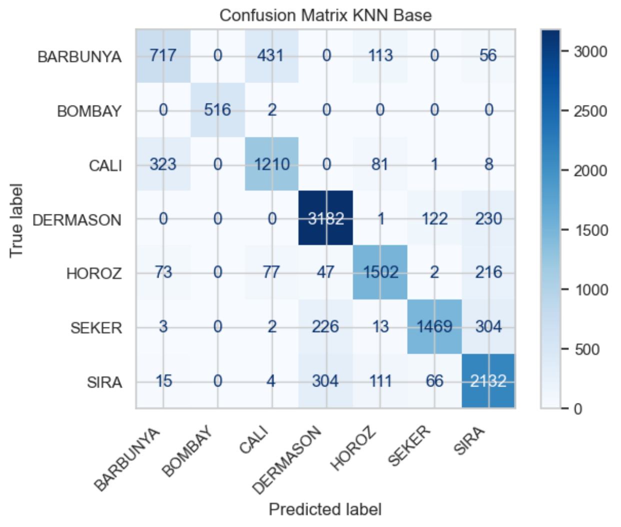


Fig. 7: Receiver Operating Characteristic (ROC) Curve for Each Class of the k-NN Base Model.



True label \ Predicted label	BARBUNYA	BOMBAY	CALI	DERMASON	HOROZ	SEKER	SIRA
BARBUNYA	717	0	431	0	113	0	56
BOMBAY	0	516	2	0	0	0	0
CALI	323	0	1210	0	81	1	8
DERMASON	0	0	0	3182	1	122	230
HOROZ	73	0	77	47	1502	2	216
SEKER	3	0	2	226	13	1469	304
SIRA	15	0	4	304	111	66	2132

Fig. 8: Confusion Matrix for the k-NN Base Model.

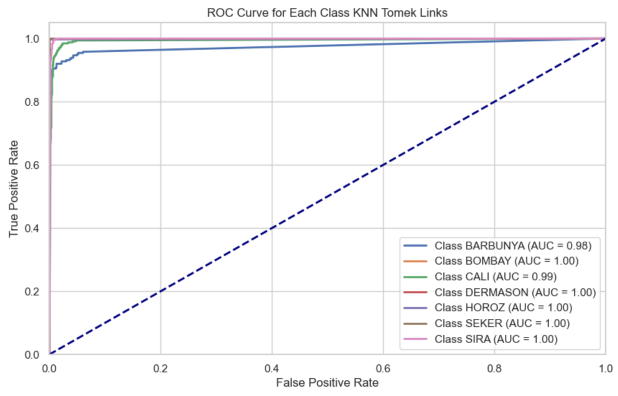
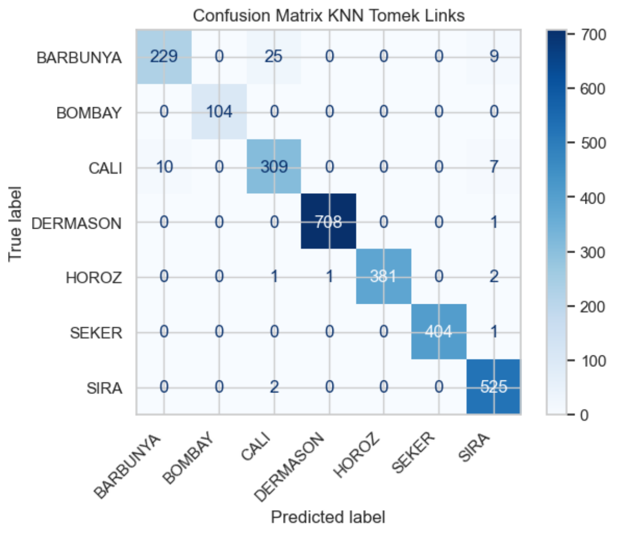


Fig. 9: Receiver Operating Characteristic (ROC) Curve for Each Class of the k-NN Tomek Links Model.

may inadvertently introduce changes that are not beneficial for the Classification Tree model. This could be due to the version of CART used, where the extensive list of hyperparameters that can be tuned allows the decision tree to generalize well



True label \ Predicted label	BARBUNYA	BOMBAY	CALI	DERMASON	HOROZ	SEKER	SIRA
BARBUNYA	229	0	25	0	0	0	9
BOMBAY	0	104	0	0	0	0	0
CALI	10	0	309	0	0	0	7
DERMASON	0	0	0	708	0	0	1
HOROZ	0	0	1	1	381	0	2
SEKER	0	0	0	0	0	404	1
SIRA	0	0	2	0	0	0	525

Fig. 10: Confusion Matrix for the k-NN Tomek Links Model.

to the data even without preprocessing.

TABLE IX: Performance Metrics of Classification Tree Models, Including Base Model, Tomek Links, and Weighted Variants, Evaluated by Weighted Accuracy and Weighted F1-Score.

Model	Weighted Accuracy	Weighted F1-score
Class. Tree Base	0.98	0.98
Class. Tree Tomek Links	0.97	0.97
Class. Tree Weighted	0.97	0.97

The best parameter for the minimum samples per leaf was 10 in the base model, suggesting that the outliers were smoothed out by a majority voting system similar to that of k-NN. This means that the processing to remove outliers and class skewness might have adversely affected the model as information was lost.

ROC curves and confusion matrices for both the base and Tomek Links models (Figures 11 and 12 for the base model, and Figures 13 and 14 for the Tomek Links model) show that although the preprocessing technique version of the Classification Tree performs well on the dataset, the base model still outperforms it. The extensive list of hyperparameters provided by the CART algorithm allows the induced classification tree to be robust to many data quality issues.

VI. Conclusion

In this study, the impact of various data preprocessing techniques was explored on the performance of two classification models, namely k-NN and the Classification Tree, using the Dry Beans dataset.The aim was to identify data quality issues and address them in reference to each of the machine learning models used.

The experimental results demonstrate that improving data quality significantly enhances the performance of the k-NN model. The results also suggest that the Classification Tree is

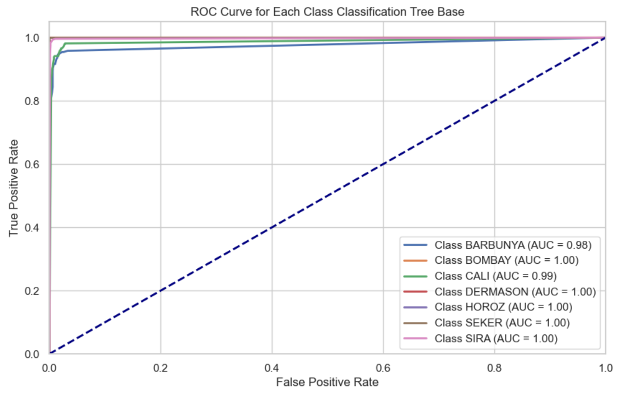
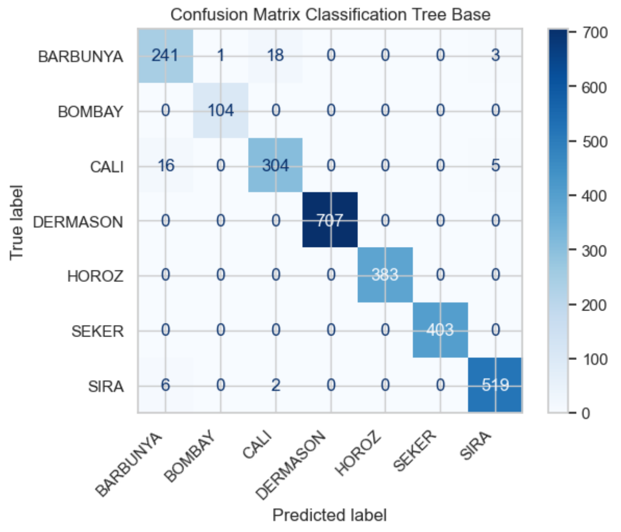


Fig. 11: Receiver Operating Characteristic (ROC) Curve for Each Class of the Classification Tree Base Model.



	BARBUNYA	BOMBAY	CALI	DERMASON	HOROZ	SEKER	SIRA
BARBUNYA	241	1	18	0	0	0	3
BOMBAY	0	104	0	0	0	0	0
CALI	16	0	304	0	0	0	5
DERMASON	0	0	0	707	0	0	0
HOROZ	0	0	0	0	383	0	0
SEKER	0	0	0	0	0	403	0
SIRA	6	0	2	0	0	0	519

Fig. 12: Confusion Matrix for the Classification Tree Base Model.

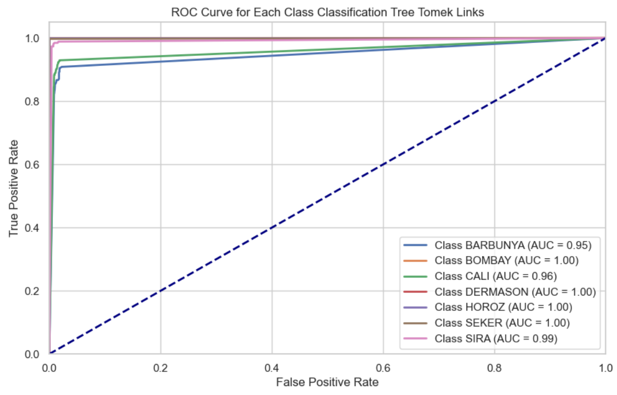
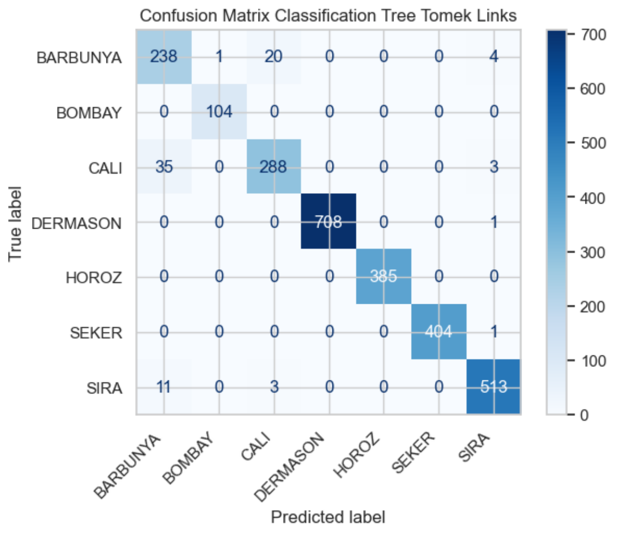


Fig. 13: Receiver Operating Characteristic (ROC) Curve for Each Class of the Classification Tree Tomek Links Model.

robust to many of the data quality issues present because of its extensive hyperparameter list. Therefore, understanding the strengths and weaknesses of each model regarding data quality is vital for effective model selection and development.



	BARBUNYA	BOMBAY	CALI	DERMASON	HOROZ	SEKER	SIRA
BARBUNYA	238	1	20	0	0	0	4
BOMBAY	0	104	0	0	0	0	0
CALI	35	0	288	0	0	0	3
DERMASON	0	0	0	708	0	0	1
HOROZ	0	0	0	0	385	0	0
SEKER	0	0	0	0	0	404	1
SIRA	11	0	3	0	0	0	513

Fig. 14: Confusion Matrix for the Classification Tree Tomek Links Model.

Future work may include applying these techniques to other datasets, including regression models. Furthermore, an analysis of more sophisticated outlier detection techniques and data imputation methods could be analysed.

Overall, this study highlights the critical role of data quality and preprocessing in machine learning, and underscores the importance of preprocessing in relation to the specific model being applied.

Source code can be found here [20]

References

[1] E. Fix and J. L. Hodges, “Discriminatory analysis. nonparametric discrimination: Consistency properties,” *International Statistical Review / Revue Internationale de Statistique*, vol. 57, no. 3, pp. 238–247, 1989. [Online]. Available: http://www.jstor.org/stable/1403797

[2] J. R. Quinlan, “Induction of decision trees,” *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[3] M. Koklu and I. A. Ozkan, “Multiclass classification of dry beans using computer vision and machine learning techniques,” *Computers and Electronics in Agriculture*, vol. 174, p. 105507, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169919311573

[4] J. W. Tukey, *Exploratory Data Analysis*. Addison-Wesley, 1977.

[5] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013. [Online]. Available: https://faculty.marshall.usc.edu/gareth-james/ISL/

[6] Y. Sasaki, “The truth of the f-measure,” *Teach Tutor Mater*, 01 2007.

[7] Oxford English Dictionary, “Overfitting,” https://web.archive.org/web/20171107014257/https://en.oxforddictionaries.com/definition/overfitting, 2024.

[8] A. Engelbrecht, “Similarity-based learning,” South Africa, 2024, slides, Machine Learning 441/741. [Online]. Available: https://engel.pages.cs.sun.ac.za/

[9] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[10] L. Breiman, J. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, 1st ed. Chapman and Hall/CRC, 1984. [Online]. Available: https://doi.org/10.1201/9781315139470

[11] M. R. Junge and J. R. Dettori, “Roc solid: Receiver operator characteristic (roc) curves as a foundation for better diagnostic tests,” *Global Spine Journal*, vol. 8, no. 4, pp. 424–429, June 2018, epub 2018 May 23.

[12] S. V. Stehman, “Selecting and interpreting measures of thematic classification accuracy,” *Remote Sensing of Environment*, vol. 62, no. 1, pp. 77–89, 1997. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0034425797000837

[13] E. Kreyszig, H. Kreyszig, and E. J. Norminton, *Advanced Engineering Mathematics*, 10th ed. Hoboken, NJ: Wiley, 2011.

[14] K. Potdar, T. Pardawala, and C. Pai, “A comparative study of categorical variable encoding techniques for neural network classifiers,” *International Journal of Computer Applications*, vol. 175, pp. 7–9, 10 2017.

[15] “Two modifications of cnn,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-6, no. 11, pp. 769–772, 1976.

[16] D. Shepard, “A two-dimensional interpolation function for irregularly-spaced data,” in *Proceedings of the 1968 23rd ACM National Conference*, ser. ACM ’68. New York, NY, USA: Association for Computing Machinery, 1968, p. 517–524. [Online]. Available: https://doi.org/10.1145/800186.810616

[17] L.-Y. Deng, M. Garzon, and N. Kumar, *Dimensionality Reduction in Data Science*. Springer, 2022.

[18] S. Glantz and B. Slinker, *Primer of Applied Regression & Analysis of Variance*. McGraw-Hill Education, 2000. [Online]. Available: https://books.google.co.za/books?id=fzV2QgAACAAJ

[19] M. Rosenblatt, “Remarks on Some Nonparametric Estimates of a Density Function,” *The Annals of Mathematical Statistics*, vol. 27, no. 3, pp. 832 – 837, 1956. [Online]. Available: https://doi.org/10.1214/aoms/1177728190

[20] F. Uys, “Classification of dry beans,” https://github.com/FrancoUysp/Classification-of-Dry-Beans, 2024, source code.