Assignment 2

L.W. Yates, 26065401
Department of Computer Science
Stellenbosch University
Stellenbosch, South Africa
Email: 26065401@sun.ac.za

Abstract—This study compares k-nearest neighbours (KNN) and classification trees on the Forest Cover dataset (581,012 instances, 58 features) which exhibits missing values, outliers, correlated features, and class imbalance. KNN preprocessing included standardization, mean imputation, and resampling techniques (SMOTE, Tomek Links). Classification trees used median imputation and cost-complexity pruning with entropy and Gini splitting criteria.

Hyperparameter optimization employed 5-fold stratified cross-validation. KNN was tuned across k values $\{1, 3, 5, 7, 9, 11, 13\}$ with Euclidean and Manhattan distance metrics. Classification trees were optimized using pruning path analysis. Final evaluation used 10-fold cross-validation with accuracy, F1-macro, precision, recall, specificity, and Cohen's kappa metrics.

Classification trees significantly outperformed KNN across all metrics. The optimal KNN configuration (k=3, Manhattan distance, Tomek Links) achieved 88.45% accuracy and 0.816 F1-macro score, while the best classification tree (entropy splitting, $\alpha = 1.32 \times 10^{-5}$) achieved 92.59% accuracy and 0.883 F1-macro score. Paired t-tests confirmed statistical significance ($p < 0.001$).

The superior performance of classification trees stems from their robustness to outliers and noise, effective handling of mixed-type features, and accommodation of non-linear relationships. These findings demonstrate the importance of algorithm selection based on dataset characteristics and highlight how preprocessing and hyperparameter tuning substantially improve performance on challenging datasets.

Index Terms—Machine Learning, KNN, K Nearest Neighbours, Classification Trees

I. Introduction

Classification is a fundamental task in machine learning, underpinning applications from medical diagnosis to fraud detection. Among the many algorithms available, KNN and classification trees stand out for their simplicity, interpretability, and non-parametric nature.

The dataset used in this assignment contains multiple types of data quality issues, including missing values, outliers, correlated features, and skewed class distributions. These issues complicate the modelling process and necessitate careful preprocessing tailored to each algorithm.

This report aims to investigate how KNN and classification trees perform on this dataset once these issues are addressed. The models are trained, tuned, and evaluated using stratified cross-validation with accuracy and F1-score as performance metrics. Results are compared and discussed in light of expectations derived from theory.

The remainder of this report is organised as follows: Section II provides background on the algorithms and expected effects of data quality issues. Section III outlines the methodology, including dataset description, preprocessing, and training. Section IV describes the empirical procedure used for evaluation. Section V presents the experimental results and discussion. Section VI concludes the report with findings and future directions.

II. Background

This section introduces the theoretical foundations of KNN and classification trees, and discusses expectations regarding the influence of data quality issues on these models. The section first describes the algorithms, followed by a discussion of data-quality sensitivity.

This assignment focuses on two supervised learning methods: k-nearest neighbours (KNN) and classification trees. Both belong to the family of non-parametric models, meaning they make minimal assumptions about the underlying data distribution, but they differ fundamentally in their inductive biases and in how they are affected by data quality issues. The following sections provide an in-depth description of each algorithm and discuss the influence of data quality issues, drawing primarily on Kelleher et al. [1].

A. K-nearest neighbours

KNN is an instance-based, or lazy, learning algorithm. Rather than building an explicit model during training, KNN stores all training instances and makes predictions at query time by identifying the k most similar instances in feature space. The core inductive bias is the assumption that instances close to each other in feature space are likely to belong to the same class. Classification is typically achieved through majority voting, though distance-weighted voting can improve robustness when class imbalance or noise is present.

A key design decision is the choice of similarity measure. For numerical attributes, Euclidean and Manhattan distances are most common; for categorical or binary attributes, overlap, Jaccard, and Hamming measures may be used. Mixed-type data may require composite measures, such as Gower's similarity, or appropriate preprocessing to ensure comparability across features.

Data quality strongly influences KNN performance. Outliers can distort distances, although their effect is limited if k is small, since majority voting reduces their influence. Missing values can be handled by either ignoring the affected features in distance computation or by imputing them, with KNN itself frequently used as an imputer by averaging or taking the mode of neighbouring values. Noise presents a greater challenge: for small k, the algorithm tends to overfit to noise, while larger k values improve robustness at the cost of bias. Finally, KNN is sensitive to skewed class distributions, since neighbours from the majority class can dominate the vote. Solutions include weighted voting, undersampling (e.g., Tomek links), or oversampling (e.g., SMOTE). Normalisation or standardisation is also crucial, since features with larger numeric ranges otherwise dominate distance calculations.

B. Classification trees

Classification trees are supervised, information-based learners that use a recursive, divide-and-conquer strategy to partition the data into increasingly homogeneous subsets. The model is structured as a tree where internal nodes represent tests on descriptive features, branches represent outcomes of these tests, and leaves represent predicted class labels. The induction process seeks to reduce uncertainty in class distributions, usually measured by entropy or the Gini index, and proceeds until a stopping condition is met (e.g., homogeneous leaves, depth limit, or minimum node size). To improve generalisation, induced trees are often pruned to remove branches that capture noise or outliers.

Classification trees have the advantage of interpretability: the path from root to leaf can be expressed as a set of production rules, which makes the model explainable in contrast to black-box approaches. However, they are prone to overfitting if grown without constraints, as they can perfectly fit training data by forming highly specific rules. They are also subject to biases such as favouring attributes with many outcomes (many-values bias) or relying on greedy splitting criteria without global optimisation.

In terms of data quality, trees exhibit robustness to outliers and noise: such instances are often isolated in small leaves, which are subsequently pruned away. They are, however, sensitive to class imbalance. Minority classes typically appear in small leaves, which are pruned and absorbed into majority-class leaves, causing the minority to be misclassified. Missing values are handled more flexibly than in KNN: training cases with missing values can be fractionally assigned to multiple branches, and surrogate splits can be used when the primary splitting attribute is missing. Traditional methods such as mean and median imputation can also be used to great effect.

C. Data-quality issues and algorithmic sensitivity

The dataset supplied for this assignment exhibits several remaining data-quality issues (missing values, outliers, features with differing numeric ranges, correlated/redundant features, and class skew). Below we summarise how each issue typically impacts KNN and classification tree methods and give short practical directions for preprocessing.

a) Missing values: Missingness can bias model estimates and degrade predictive performance if treated incorrectly. The appropriate strategy depends on the type of missingness, of which there are three types. Missing completely at random (MCAR) occurs when the probability of missingness is not influenced by the available or unvailable data; missing at random (MAR) occurs when the missingness is related to the observed data but not the unobserved data; and missing not at random (MNAR) occurs when the missingness is related to the unobserved data [1]. For KNN, common strategies include:

- Imputation: For each missing value we impute a value using the mean/median.
- Distance-weighted imputation using nearest neighbours: We find the k nearest neighbours and use their values to impute the missing value, weighting by distance.
- Ignoring features with missing values in distance calculations (if few missing values): The chosen similarity measure is calculated by ignoring the missing values and scaling up the weight of the non-missing descriptive features.

Classification trees can handle missing values by using the following strategies:

- Surrogate splits: When a feature is missing, the tree can use another feature that is correlated with the missing feature to make a split.
- Imputation: Missing values can be imputed using the mean/mode of the feature or a more sophisticated method such as k-NN imputation.
- Ignoring missing values: If a feature has too many missing values, it can be ignored altogether.

b) Outliers: Outliers are data points that deviate significantly from the majority of the data, and can end up impacting model performance if they are not handled properly. For KNN, the sensitivity to outliers depends on the value K for the model. As K stays small, the probability for the model to select an outlier as a neighbour remains small, while larger values of K can include outliers in the majority vote. For classification trees, they are known to be robust to outliers. The reason for this is that outliers will tend to be isolated in their own leaf nodes, and after pruning, these leaf nodes are likely to be removed, thus reducing the impact of outliers on the overall model.

c) Correlated / redundant features: High correlation between features increases dimensionality without adding information; KNN suffers because redundant dimensions can distort distances, while classification trees typically select one of the correlated features for splitting (reducing interpretability but not always harming predictive power).

In practice, removing near-constant or highly redundant features, or using dimensionality reduction methods, can improve KNN performance and simplify tree structure [1].

d) Class imbalance (skew): When class distributions are highly skewed, simple accuracy is misleading. Techniques to address imbalance include using more informative performance metrics (precision, recall, F1, AUC), and applying resampling strategies (oversampling the minority class, undersampling the majority class). A widely used oversampling method is SMOTE (Synthetic Minority Over-sampling Technique), which generates synthetic minority examples. A commonly used undersampling method is Tomek Links, which removes examples from the majority class that are close to the minority class.

In the empirical work that follows we adopt preprocessing choices motivated by the above sensitivities: scaling prior to KNN, careful imputation for missing values (with justification based on observed missingness patterns), outlier detection and conservative treatment, and evaluation metrics and (where appropriate) resampling to handle class imbalance. These choices and their justifications are described in detail in Section III and Section IV.

III. Methodology

In this section we will describe the dataset, discuss the preprocessing steps taken for each model with justifications for each data quality issue resolution, and finally we will outline the model training and evaluation process.

A. Dataset description

The dataset used in this assignment is the Forest Cover Type dataset, which contains 581,012 instances and 58 features. The task is to predict the forest cover type (the target variable) based on these features. The dataset also has a mix of different data quality issues, refer to Table I for an overview.

TABLE I
Key data quality issues in selected features.

Feature	Data Issue
Slope	Missing (0.05%)
Facet	Correlated
Inclination	Noisy
Various	Outliers
Various	Range differences
Various	Mixed types
Water Level	Single value
Obs. ID	Unique
Target	Imbalanced

B. Preprocessing Steps

For each model we applied different preprocessing steps to address the data quality issues identified in the dataset. Below we outline the steps taken for each model along with justifications.

For KNN we first decided to drop Facet, Water_Level, and Observation_ID. Facet was dropped because it is highly correlated with Aspect, which can distort distance calculations in KNN. Water_Level was dropped because it has only one unique value, which does not provide any useful information for the model. Observation_ID was dropped because it is a unique identifier for each observation and does not contribute to the predictive power of the model. We also ensured that Soil Type 1 was hot encoded with the negative values encoded as 1 and the positive encoded as 0. To address the missing values in the Slope feature, we imputed the mean value of the column. The option of adjusting the distance calculation was also considered, but due to the minute impact of the missing values (0.0513%), it was decided that mean imputation would be a simpler and effective solution. We then standardized the features to have zero mean and unit variance. As KNN is sensitive to the scale of the features, this step was crucial to ensure that all features contribute equally to the distance calculations. Finally to deal with class imbalances we tested SMOTE (Synthetic Minority Over-sampling Technique) to generate synthetic samples for the minority class, and Tomek Links to remove overlapping samples between classes. All data transformations and resampling operations (SMOTE, Tomek Links) were performed within each training fold to avoid data leakage. The results of these techniques will be discussed in the results section V. Other data quality issues such as outliers, and noise, are not dealt with by preprocessing. Rather we depend on the K value we will select in order to balance the impact of these two issues on our model.

For classification trees, we performed similar data preprocessing steps as for KNN, with the exception of feature scaling and missing values. For missing values we simply imputed with the median instead of the mean, as it is more robust to outliers. With regards to feature scaling, classification trees are not always sensitive to the scale of the features, as they make decisions by finding the best split points on individual features, comparing values within each feature to determine thresholds. With regards to this dataset, this step was not necessary. There is no need to specifically address outliers or noise, as classification trees are robust to these issues. This is because the outliers will be all by themselves at the leaf nodes, and the leaves will be pruned, and all instances combined to form a subset for the parent node. This will result in outliers being in the minority, and thus not selected by the tree. Noisy patterns will also be isolated in their own leaf nodes, and similarly pruned away, meaning the noise will not affect the majority class. Though classification trees are robust to both outliers and noise, they are sensitive to class imbalance, as minority classes can be pruned away during tree construction. Therefore, addressing class imbalance was a priority in our preprocessing steps. We used the same methods to deal with class imbalances, the results of which will be discussed in the results section V. We trained the model using information gain (entropy), where we split the trees until we had a homogeneous set of samples in each leaf node. We then pruned the tree, as at a homogeneity of 1 we are perfectly fitted to the training set. Using sklearn's 'cost_complexity_pruning_path' function, we found all alpha values for each path, using no sampling, SMOTE and Tomek. We then tested each of these collections of alpha values on the a test set, the results of which can be seen in the results section V. Once the optimal alpha value for each method was found, we retrained the model on the combined training and validation set, and evaluated it on the test set.

C. Implementation choices

For the KNN & Classification Tree models we used the following libraries with a random seed of 42, where applicable, for reproducibility. The libraries used are as follows:

- import numpy as np
- import pandas as pd
- from sklearn.model_selection import train_test_split, StratifiedKFold
- from sklearn.preprocessing import StandardScaler
- from sklearn.neighbors import KNeighborsClassifier
- from sklearn.tree import DecisionTreeClassifier
- from sklearn.metrics import accuracy_score, f1_score, classification_report
- from imblearn.over_sampling import SMOTE
- from imblearn.under_sampling import TomekLinks as Tomek

IV. Empirical Procedure

Performance metrics are vital when determining the performance of a machine learning model. We focused on accuracy and F1-score as the primary metrics. Accuracy, defined as Accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$, measures the proportion of correctly classified instances, where TP, TN, FP, and FN represent true positives, true negatives, false positives, and false negatives, respectively [2]. While accuracy is simple to interpret, it can be misleading in the presence of class imbalance. To address this, we also measured the F1-score, which balances precision and recall. Precision and recall are defined as Precision $= \frac{TP}{TP+FP}$ and Recall $= \frac{TP}{TP+FN}$, while the F1-score, which balances both, is given by $\text{F1} = 2 \cdot \frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}$.

This makes the F1-score particularly suitable for the Forest Cover dataset, where some classes are underrepresented.

Cross-validation was used to estimate the generalisation ability of the models. In k-fold cross-validation, the dataset is divided into k disjoint folds; the model is trained on $k-1$ folds and validated on the remaining fold, with the process repeated k times [3]. The final performance is the average across folds. We applied stratified 5-fold cross-validation, ensuring that class distributions were preserved across folds. This is due to computational constraints, as in the final evaluation 10-fold CV was used. However, five folds were chosen as a balance between computational efficiency and variance reduction, providing robust performance estimates without excessive training overhead.

For KNN, two hyperparameters required tuning: the number of neighbours k and the distance metric. The parameter k controls the bias–variance trade-off: small k leads to low bias but high variance, while large k increases bias but reduces variance [1]. We evaluated values $k \in \{1, 3, 5, 7, 9, 11, 13\}$.

Two distance metrics were considered. Euclidean distance is defined as $d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$, which emphasises large differences across features. Manhattan distance is defined as $d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$, and is often more robust in high-dimensional or sparse feature spaces.

For Classification trees, we explored two splitting criteria: entropy (as used in C4.5 [4]) and Gini impurity (as used in CART [5]). Entropy is defined as $H(S) = -\sum_{i=1}^{c} p_i \log_2 p_i$, where p_i is the proportion of class i in subset S. The information gain of a split on attribute A is then $IG(S, A) = H(S) - \sum_{v\in Values(A)} \frac{|S_v|}{|S|} H(S_v)$, where S_v is the subset of S with attribute A taking value v.

The Gini impurity of a subset S is defined as $\text{Gini}(S) = 1 - \sum_{i=1}^{c} p_i^2$, where p_i denotes the proportion of class i in S.

Both criteria measure how well a feature split separates the data into pure subsets. To prevent overfitting, cost-complexity pruning was applied, where a complexity parameter α penalises tree size. The optimal α values were selected by evaluating their F1-scores. All preliminary experiments were run with fixed random seeds for reproducibility. Results were reported as mean and standard deviation across the five folds.

For our final test runs on our selected models, 10 folds were used instead of the original 5, we also decided to expand our collection of performance metrics to include precision, recall, specificity, as well as Cohens Kappa and the use of a confusion matrix.

Specificity, which measures the proportion of true negatives correctly identified, is defined as Specificity $= \frac{TN}{TN+FP}$, where TN and FP denote true negatives and false positives, respectively.

Cohen's Kappa, which quantifies agreement between predicted and true labels beyond chance, is given by $\kappa = \frac{P_o - P_e}{1 - P_e}$, where P_o is the observed agreement and P_e is the expected agreement by chance.

As we are going to compare these two algorithms we require statistical tests to determine if the observed differences in performance are statistically significant.

Our formal hypothesis is as follows:

H0: There is no significant difference in predictive performance between KNN and Classification Tree classifiers on the Forest Cover dataset, as measured by F1 score.

H1: There is a significant difference in predictive performance between KNN and Classification Tree classifiers on the Forest Cover dataset, as measured by F1 score.

We use F1-macro score as our primary evaluation metric due to the class imbalance present in the Forest Cover dataset, where F1-macro provides equal weight to all classes regardless of their frequency.

We conducted a paired t-test using F1-macro scores from 10-fold cross-validation to test our hypothesis at significance level $= 0.05$. The pairing accounts for both algorithms being evaluated on identical data folds, reducing variance in the comparison.

The paired t-test statistic is calculated as $t = \frac{\bar{d}}{s_d/\sqrt{n}}$, where $\bar{d}$ is the mean of the differences between paired observations, s_d is the standard deviation of the differences, and n is the number of pairs.

Under the null hypothesis, this statistic follows a t-distribution with $n-1$ degrees of freedom.

This comprehensive evaluation framework allows for rigorous comparison of KNN and classification tree performance while accounting for the specific challenges presented by the Forest Cover dataset. The results of applying this methodology are presented in the following section, where we analyze both individual algorithm performance and comparative statistical significance.

V. Research Results

In this section we will discuss the results of our experiments, including preliminary algorithm exploration, the impact of preprocessing techniques, hyperparameter optimization, and final model performance.

A. Preliminary Algorithm Exploration

We tested different variations of algorithms ranging from no preprocessing, to testing different distance measures, and different sampling techniques (SMOTE and Tomek Links). It must be noted that the pruning value for the classification trees, was calculated before the testing of these configurations, the only configuration that does not use the pruning value is the baseline configuration. These pruning values can be found in the section V-B. For KNN we used a k value of 5, the reason for this is that it is computationally infeasible to run the range of k values on each variation. In Tables II and III, we show the results of KNN and Classification Trees with different configurations, with a heatmap in figure 1. The performance metrics we use for these preliminary results are accuracy and F1-macro score.

TABLE II
KNN preliminary results with k=5

Configuration	Accuracy	F1-macro
Baseline (Euclidean)	0.864 ± 0.001	0.779 ± 0.006
SMOTE (Euclidean)	0.844 ± 0.001	0.756 ± 0.006
Tomek (Euclidean)	0.863 ± 0.001	0.776 ± 0.006
SMOTE (Manhattan)	0.858 ± 0.002	0.772 ± 0.006
Tomek (Manhattan)	0.879 ± 0.001	0.803 ± 0.006

From the preliminary results, we can see that for KNN, using Tomek Links with the Manhattan distance metric provided the best performance in terms of both accuracy and F1-macro score. This suggests that removing overlapping samples between classes helps KNN better distinguish between classes, especially in the presence of class imbalance. The choice of distance metric also plays a significant role, with Manhattan distance outperforming Euclidean distance in this case. As for the classification tree, the results indicate that the choice of impurity measure (Entropy vs Gini) did not significantly impact performance, as both configurations yielded similar results. However, the use of SMOTE and Tomek Links did lead to a slight decrease in performance.

TABLE III
Classification Tree preliminary results

Configuration	Accuracy	F1-macro
Entropy (baseline)	0.929 ± 0.001	0.890 ± 0.003
Entropy + SMOTE	0.926 ± 0.001	0.875 ± 0.002
Entropy + Tomek	0.922 ± 0.001	0.879 ± 0.004
Gini (baseline)	0.929 ± 0.001	0.888 ± 0.001
Gini + SMOTE	0.923 ± 0.002	0.868 ± 0.006
Gini + Tomek	0.921 ± 0.001	0.871 ± 0.002

These results with respect to KNN are somewhat expected. Given the class imbalance of the dataset, undersampling with Tomek Links helps to create a more balanced training set, allowing KNN to better learn the decision boundaries between classes and results in a higher score. The Manhattan distance metric is also a major factor in these results. Across all variations the Manhattan distance performed consistently better than Euclidean distance. This could be due to the fact that Manhattan distance is less sensitive to outliers and better captures the structure of the data in high-dimensional spaces.

The results for the classification trees show that the baseline configurations consistently outperformed those trained with SMOTE or Tomek Links. This finding appears counterintuitive, since classification trees are known to be sensitive to skewed class distributions: minority classes often end up in small leaves that are pruned and absorbed into majority classes. However, in this case the resampling strategies seem to have introduced additional complications. SMOTE interpolates synthetic minority examples that may not align with natural feature distributions, leading the tree to form less meaningful splits. Tomek Links, on the other hand, remove borderline samples that are critical for defining accurate decision boundaries. Even though pruning parameters were tuned for each resampled dataset, these structural changes reduced the tree's ability to generalise. As a result, the unaltered dataset produced stronger overall performance despite the imbalance.

B. Hyperparameter Optimization

For KNN, we tested different values of k (1, 3, 5, 7, 9, 11, 13) with 5 folds to ensure robust evaluation. We also tested the distance metrics previously mentioned (Euclidean and

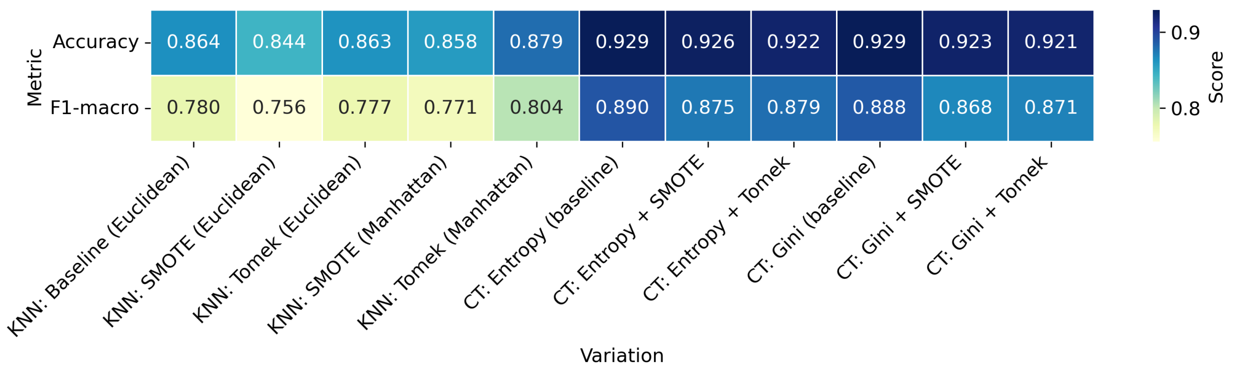

Fig. 1. Heatmap of Tuning Results

Manhattan) in the previous section V-A. The reason for not performing a grid search, random search or Bayesian optimization was due to the computational cost and time constraints. Due to this, distance metrics were tested separately, and then tested KNN on the best configuration found. After testing these value of k on the model using Tomek Links with the Manhattan distance metric, we found the results displayed in the table IV, and visualised in figure 2.

TABLE IV
KNN Grid Search Results - All K Values

K Value	Mean Accuracy	Std Accuracy	Mean F1 Macro	Std F1 Macro
1	0.876617	0.001983	0.804984	0.004821
3	0.880605	0.001724	0.809393	0.005518
5	0.879370	0.001100	0.803190	0.004942
7	0.876843	0.000756	0.798115	0.004811
9	0.874224	0.000944	0.793542	0.005429
11	0.871706	0.001029	0.788215	0.005070
13	0.869002	0.001005	0.783479	0.003756

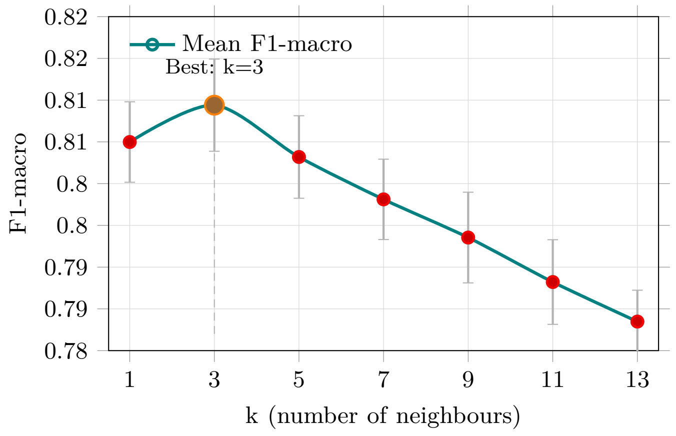

Fig. 2. Effect of k on mean F1-macro for KNN. Points show means, thin caps indicate one standard deviation; best-performing k is highlighted.

After examining these results we can see that the best performance was achieved with $k = 3$, yielding a mean accuracy of 0.880605 and a mean F1-macro score of 0.809393. This indicates that a smaller number of neighbours allows KNN to better capture local patterns in the data, leading to improved classification performance. As k increases, the model becomes more biased towards the majority class, resulting in decreased performance metrics.

While with classification trees we tested different alpha values to determine the best configuration for pruning. A variable in the sklearn DecisionTreeClassifier called ccp_alpha controls the complexity of the tree by pruning it. We tested unique alphas for each configuration as mentioned previously and obtained the alpha value by using the function cost_complexity_pruning_path. We compared these values using F1-score, the top 10 results of each can be seen in the tables V, VII, and VI.

TABLE V
Top 10 Alpha Values for Baseline Classification Tree

Rank	Alpha (α)	F1	Nodes	Depth
1	1.323202e-05	0.892363	26531	39
2	1.333474e-05	0.892345	26441	39
3	1.250807e-05	0.892266	27823	39
4	1.347537e-05	0.892261	26287	39
5	1.283619e-05	0.892247	27369	39
6	1.260523e-05	0.892243	27735	39
7	1.373624e-05	0.892229	25951	39
8	1.235023e-05	0.892222	27951	39
9	1.359969e-05	0.892194	26117	39
10	1.395084e-05	0.892153	25793	39

TABLE VI
Top 10 Alpha Values for Tomek Classification Tree

Rank	Alpha (α)	F1	Nodes	Depth
1	6.876173e-06	0.883191	30799	33
2	7.875869e-06	0.883156	30461	33
3	1.151093e-05	0.882873	26461	33
4	0.000000e+00	0.882619	31765	33
5	9.185940e-06	0.882328	29053	33
6	1.132054e-05	0.882311	26893	33
7	9.501301e-06	0.882299	28325	33
8	1.087512e-05	0.882143	27189	33
9	1.107787e-05	0.882014	27029	33
10	1.294098e-05	0.881955	24789	33

TABLE VII
Top 10 Alpha Values for SMOTE Classification Tree

Rank	Alpha (α)	F1	Nodes	Depth
1	0.000000e+00	0.884129	31845	41
2	1.130157e-05	0.884092	28793	39
3	9.477934e-06	0.884033	30349	41
4	1.083709e-05	0.883987	28925	39
5	8.543190e-06	0.883972	30571	41
6	1.018652e-05	0.883907	29143	39
7	1.596626e-05	0.883742	23699	38
8	1.242036e-05	0.883684	27493	39
9	1.204330e-05	0.883591	27757	39
10	1.673245e-05	0.883568	23143	38

C. Final Model Performance

For our final models we have chosen KNN with $k = 3$, using the Manhattan distance metric and Tomek Links as our sampling technique, and for classification trees we have chosen the baseline configuration using Entropy as our impurity measure, and an alpha value of $1.323202e{-}05$ for pruning. The results of these final models can be seen in the tables VIII and IX, as well as in the model comparison heatmap (Figure 3).

TABLE VIII
KNN final model performance (mean $\pm$ std)

Metric	Value
Accuracy	0.8845 ± 0.0026
F1-macro	0.8159 ± 0.0071
F1-weighted	0.8839 ± 0.0026
Precision	0.8373 ± 0.0067
Recall	0.7977 ± 0.0081
Specificity	0.9728 ± 0.0006
Cohen's Kappa	0.8137 ± 0.0042

TABLE IX
Classification Tree final model performance (mean $\pm$ std)

Metric	Value
Accuracy	0.9259 ± 0.0017
F1-macro	0.8828 ± 0.0037
F1-weighted	0.9258 ± 0.0017
Precision	0.8877 ± 0.0047
Recall	0.8785 ± 0.0039
Specificity	0.9828 ± 0.0004
Cohen's Kappa	0.8809 ± 0.0027

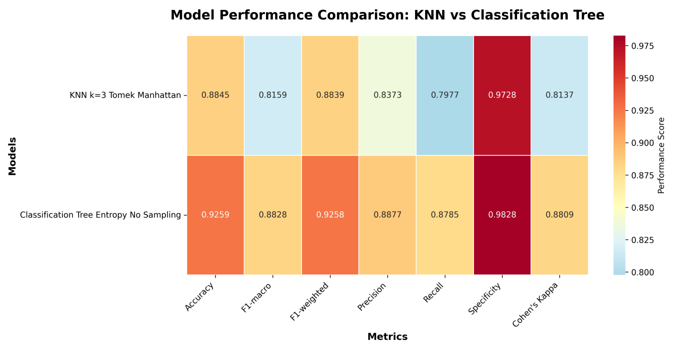

Fig. 3. Model Comparison Heatmap: KNN vs Classification Tree

To determine which model performed best we conducted a paired t-test as described in section IV. The results of this test can be seen in table X.

TABLE X
Paired t-test results for F1-macro

Statistic	Value	Notes
Metric	f1-macro	
KNN mean	0.815891	mean across folds
DT mean	0.882839	mean across folds
Mean difference (KNN - DT)	-0.066948	
Paired t-statistic	-29.7741	df = 9
p-value	2.6569×10^{-10}	two-tailed

Based on our formal hypothesis testing framework, we can now evaluate the statistical evidence. Our null hypothesis H_0 states that there is no significant difference in predictive performance between KNN and Classification Tree classifiers on the Forest Cover dataset, as measured by F1-macro score. The alternative hypothesis H_1 posits that there is a significant difference in performance between these classifiers.

The paired t-test results provide compelling evidence to reject H_0 at the $\alpha = 0.05$ significance level. With a p-value of 2.6569×10^{-10}, which is substantially less than our predetermined significance threshold, we have strong statistical evidence against the null hypothesis. The test statistic $t = -29.7741$ with 9 degrees of freedom (n-1 = 10-1) falls well within the critical rejection region, confirming our decision to reject H_0.

Therefore, we accept the alternative hypothesis H_1 and conclude that there is a statistically significant difference in predictive performance between KNN and Classification Tree classifiers on the Forest Cover dataset. Specifically, the negative mean difference of -0.066948 indicates that the classification tree consistently outperforms KNN across all cross-validation folds in terms of F1-macro score.

VI. Conclusion

This study compared the performance of k-nearest neighbours (KNN) and classification tree classifiers on the Forest Cover dataset, with a focus on addressing data quality issues such as missing values, outliers, correlated features, and class imbalance. Both models were trained and evaluated using cross-validation, with accuracy, F1 score, and additional metrics guiding the analysis.

The results showed that classification trees consistently outperformed KNN across all evaluated metrics. KNN achieved its best performance with $k = 3$ and the Manhattan distance metric combined with Tomek Links, but it remained more sensitive to class imbalance and high-dimensional feature interactions. In contrast, classification trees demonstrated greater robustness to noise and outliers, and careful pruning further improved their generalisation ability. Statistical testing confirmed that the observed performance difference was significant, providing strong evidence that classification trees are better suited to this dataset.

In conclusion, while KNN benefits from simplicity and interpretability, classification trees offered higher predictive performance and stronger resilience to the data quality issues present in the Forest Cover dataset.

References

[1] J. D. Kelleher, B. Mac Namee, and A. D'Arcy, Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies. MIT Press, 2015.

[2] D. M. W. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 2020. [Online]. Available: https://arxiv.org/abs/2010.16061

[3] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2, ser. IJCAI'95. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, p. 1137–1143.

[4] J. R. Quinlan, C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.

[5] L. Breiman, J. Friedman, R. Olshen, and C. Stone, Classification and Regression Trees (1st ed.). Chapman and Hall/CRC, 1984.