Comparison of classification algorithms

*An evaluation of kNN and classification trees

BS Steyn
(Department of Computer Science)
Stellenbosch University
Stellenbosch, South Africa
21740178@sun.ac.za

***Abstract*—This paper presents a comparative analysis of two supervised machine learning algorithms, *k*-Nearest Neighbours (kNN) and classification trees, applied to a dataset containing several data quality challenges. A comprehensive preprocessing pipeline was implemented to address missing values, redundant and noisy features, extreme outliers, mixed feature types, and class imbalance. The *k* parameter for the kNN algorithm was optimised using the elbow method, resulting in an optimal value of $k = 3$. The performance of the models was evaluated using 5-fold cross-validation to balance bias and variance. The classification tree achieved the highest mean accuracy of 94.78%, compared to 90.94% for the kNN algorithm. A Wilcoxon Signed-Rank Test with a significance level of $\alpha = 0.05$ indicated no statistically significant difference between the models ($p = 0.0625$). These results demonstrate that both algorithms provide robust predictive performance when supported by appropriate data preprocessing techniques.**

I. Introduction

The selection of an appropriate classification algorithm is a critical step in supervised machine learning, as different algorithms exhibit varying levels of sensitivity to data characteristics and preprocessing techniques. This paper focuses on a comparative analysis of two widely used algorithms: *k*-Nearest Neighbours (kNN) [1] and classification trees [3]. These models were selected due to their contrasting decision-making strategies. While kNN classifies data based on similarity within the feature space, classification trees partition the input space by recursively applying decision rules. Evaluating the relative performance of these models provides insight into their suitability for different data conditions.

The dataset used in this study presented several data quality challenges, including missing values, redundant and noisy features, extreme outliers, inconsistent feature scales, and imbalanced class distributions. Without proper preprocessing, these issues reduce predictive accuracy and increase model bias. To address these challenges, a structured preprocessing pipeline was implemented, consisting of median imputation, removal of redundant features, treatment of noisy records, outlier handling, one-hot encoding of categorical attributes, normalisation where required, and balancing of class distributions. This ensured that both models received consistent and representative inputs for training and evaluation.

The kNN algorithm was optimised using the elbow method to determine the optimal number of neighbours, with the best performance observed at $k = 3$. Classification trees were optimised through pruning to prevent overfitting and improve generalisation. Both models were evaluated using a 5-fold cross-validation approach to provide a reliable estimate of their predictive performance while balancing bias and variance. Statistical comparison was performed using the Wilcoxon Signed-Rank Test with a significance level of $\alpha = 0.05$ to determine whether the observed differences in performance were significant.

The remainder of this paper is separated into the following sections: Section II provides a background of the various models used. Section III outlines the preprocessing techniques used to enhance the dataset. Section IV provides an overview of the implementation. Section V provides an overview of the empirical procedure used to evaluate the models. Section VI provides the results, which are summarised and concluded in Section VII.

II. Background

This section provides a background on the individual algorithms.

A. K-Nearest Neighbour

The k-nearest neighbour algorithm (kNN, based on the work presented by Evelyn and Hodges [1], is a non-parametric supervised learning algorithm. It is mostly used in classification problems. KNN uses proximity to make classifications or predictions about the grouping of an individual data point. The algorithm functions on the assumption that similar points are positioned in a contiguous space. When attempting to classify an unlabelled query point, the neighbourhood of points around it is considered and subsequently estimated as a member of the majority class. This does, however, mean that the KNN algorithm is extremely sensitive to the distance metric used to define the k nearest points. The value of the constant k influences both the efficiency as well as the performance of the KNN algorithm. As the larger the value of k the larger, the subset of points that need to be considered with each classification increases. Hence, the larger k, the greater the computational cost of the algorithm. K also has a significant effect on the performance of the algorithm; if k is too large the algorithm any skew in the dataset is exaggerated,

causing the majority class to dominate. If k is too small, the algorithm becomes very sensitive to noise. Finding an optimal value for k depends highly on the training set of your data. KNN effectiveness is highly dependent on the metric used to define the subset of the k nearest points. Popular metrics like Euclidean distance have a bias towards large features in the dataset. To counter this, the dataset should be normalised to achieve the optimal performance of the KNN algorithm.

B. Classification Trees

Decision trees, proposed by Breiman *et al* [3], are a commonly used classification algorithm. Classification trees recursively partition the feature space by solving an optimisation problem which minimises inter-class impurity. This is achieved by developing a rule at each node to minimise the diversity of classes on each side. The metric used to measure the reduction in diversity is tantamount to classification success. To reduce the effect of over-fitting, pruning can be performed at the deep nodes.

III. Dataset Preprocessing

The dataset used in this paper contains various data quality issues. There are missing values, redundant features, noisy features, extreme outliers, normalisation, a numeric and categorical mixture, feature cardinality and data imbalance. To truly determine the effectiveness of the different algorithms, the dataset must first have these hindrances removed. As then, the trends in the underlying data will be more visible. The following subsections discuss how these data quality issues were resolved.

A. Missing Data

The dataset used in this paper is beset with "?" to indicate missing values. Missing "?" values must be dealt with as if they remain; the models may learn to identify patterns in the occurrence frequencies of these values in each of the respective classes. The simplest solution would be to drop these values; however, this leads to a loss of data, which may skew the data. A solution to this problem is to replace the missing values ("?") with the median values of their respective column. This technique was chosen as it is fast to implement. This method is not without its issues. The main issue is that it alters the underlying distribution of the dataset by reducing the variance. However, the true values of the "?" values are impossible to determine, and the number of missing values to be replaced by the mean is small enough not to alter the underlying distribution too severely. This method, however, is not used for missing values in the target feature, as replacing these would lead to false associations being created between the target classes and the data. For missing values in the target feature, that entry is dropped in its entirety. These were performed for both kNN and classification trees.

B. Redundant Features

There is no use in training a classification model on features that convey irrelevant information in relation to predicting the class. If these fields are left in the data, they will lead to the models identifying trends in the records of these redundant fields. The models will then make predictions based on these identified trends, which in reality have no relation to which class an entry belongs to. Examples of these redundant fields from the dataset used in this paper are the *Observation ID*, *Water Level*, *Facet* and *Aspect*. Water Level is redundant as all the entries are the same; hence, this field provides no information. In a similar sense, the *Observation ID*, which is assigned sequentially, has no bearing on the class of the entry. *Facet* and *Aspect* have a high correlation, meaning that there is no point in keeping both features as they effectively provide the same information. This is done for both algorithms.

C. Noisy Features

Noisy features are those containing erroneous or highly inconsistent data that introduce unnecessary variability into the dataset and can mislead the classification models. These features may arise due to data entry mistakes, sensor errors, or inconsistencies in recording procedures. If left unaddressed, noisy features can reduce the predictive performance of both kNN and classification trees by introducing misleading patterns into the model. To mitigate this, a combination of statistical checks and manual inspection was performed. Features where a significant proportion of values showed inconsistent or unrealistic fluctuations were identified as noisy. For kNN, these noisy values were replaced using the median of the corresponding feature, as this approach preserves the central tendency while limiting the impact of extreme deviations. For classification trees, noisy records were left intact where possible, as tree-based models are generally more robust to minor variability. However, in cases where the inconsistency was too severe, the affected entries were removed entirely. This ensures that the dataset provides reliable and representative information to the models.

D. Extreme Outliers

The dataset used in this paper contains values that are clearly incorrectly recorded. There are occurrences where values are orders of magnitude larger than the other entries of the same feature. These records are treated as missing values and handled as discussed in the previous subsection. It is important to note that not all outliers are treated in this way; only those that are extreme outliers. Due to the difference in sensitivity of the kNN vs classification trees, these outliers are treated differently for each. For kNN, these entries are removed as kNN is highly sensitive to outliers due to the fact that distance is used in determining the associated class. For Classification trees, these outliers are rather just capped at the third and first quartiles of the feature.

E. Normalisation

Some models are very sensitive to large variances in the scale of different features in the dataset. If the variance differs by a large degree, the impact of the larger-scale features will dwarf that of the smaller-scale features. To resolve these

issues, the dataset is standardised so that all features have a range of [0,1]. This is only performed for kNN, due to the distance calculation used in kNN, it is highly sensitive to this data issue. Classification trees, however, split on thresholds; hence, scale doesn't matter.

F. Numeric and Categorical Mixture

The dataset used in this study consists of a mixture of numeric and categorical features, which poses challenges for preprocessing and model compatibility. Since kNN relies on distance-based metrics, categorical features must be encoded numerically to ensure meaningful comparisons between samples. To achieve this, one-hot encoding was applied to all categorical features. This technique prevents the creation of artificial ordinal relationships between categories while maintaining the interpretability of the dataset. For classification trees, however, categorical features were left in their original form, as the algorithm inherently handles categorical splitting based on class purity without requiring numerical transformation. This dual approach ensures that each algorithm receives an optimally formatted dataset while maintaining consistency across training and evaluation. By treating numeric and categorical features differently depending on the model, the preprocessing pipeline maximises predictive performance while preventing potential distortions in feature relationships.

G. Data Balancing

As mentioned in an earlier section, when there is an imbalance in the proportion of training data associated with a single class, the majority class can begin to dominate the minority class. Some models are more sensitive to this imbalance than others. However, to achieve the maximum out of all the models used, this imbalance must be addressed. In this paper, the imbalance is addressed by under-sampling the majority class and oversampling the minority class. Oversampling is done by increasing the number of data points associated with the minority class. This is done by randomly duplicating entries in the minority class until the desired number of entries is reached. Under-sampling is when only a subset of data points is taken from the majority class. This is done for both algorithms.

IV. Implementation

The algorithms discussed are implemented using the *Scikit-learn* [4] machine learning library. This library contains implementations for the classification models used in this paper. To visualise the results, the *Seaborn* [6] and *Matplotlib* [2] visualisation libraries are used. The dataset preprocessing is done using the *Pandas* [5] library.

V. Empirical Methodology

This section provides details on how the individual algorithms are first optimised and then evaluated to determine which method performs the most optimally on the dataset used in this paper.

A. Algorithmic Optimisation

Before evaluating the success of the respective models, they must first be optimised. The main model that is optimised is the KNN algorithm. For the KNN algorithm, values in the range [1:21] are selected. A kNN model is trained using each of these values, and then the trained models are used to evaluate the test dataset. The elbow method is then used to determine the optimal value for k.

B. Evaluation Metric

To determine if there is any significant statistical difference between the models, a 5-fold method is used to collect the test accuracy values of the various models and ensembles. The set of test accuracies for each of the models is then tested for a significant statistical difference using a Wilcoxon Signed-Ranked test. A 5-fold method was chosen as larger values lead to higher variance and lower values lead to higher bias, where 5 strikes a good balance between these extremes.

VI. Results

The results are presented in the following two subsections. Subsection A provides the outcome for the optimisation of the models. Subsection B provides the success of all models on the test dataset.

A. Optimal Model Parameters

For the KNN algorithm, the average of 10 runs where the train-test split of the data was randomly performed was used to determine that the optimal performance parameter is $k = 3$ for the dataset used in this paper. Consider the figure:

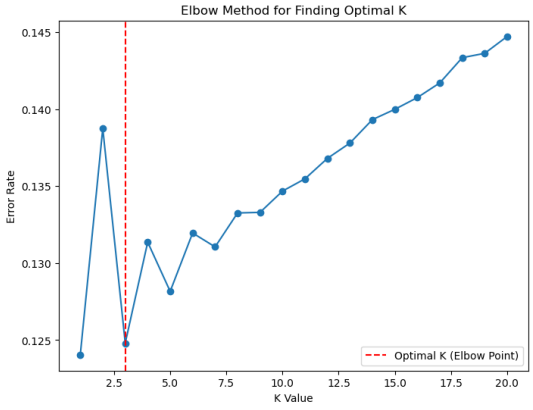

Fig. 1. Each data point represents a pairing of a k value with its average error rating

In Figure 1, it can be observed that the k values of 1 and 3 are the values with the lowest error rating. It can be observed that the k value of 3 is where the model starts to stabilise, hence why it was chosen as the optimal k value. Figure 71 also shows that larger values of k tend to have an increased error rating, but also that they tend to settle.

B. Model Performance on the Test Data

The performance of each of the models is tested to determine which model has the best performance. These tests are done using a 5-fold method. It is determined that the best-performing model is the classification tree model. Using a Wilcoxon Signed-Ranked statistical test with an α value of 0.05, it was determined that there is no statistically significant difference between the results of the models. Since the p-value determined using the test is 0.0625, which is not less than the α value. The following tables show the results for the two models.

Fold	Accuracy
Fold 1	94.72%
Fold 2	94.80%
Fold 3	94.67%
Fold 4	94.82%
Fold 5	94.90%
Mean	94.78%

TABLE I
5-FOLD MEAN TEST ACCURACIES FOR CLASSIFICATION TREE

Fold	Accuracy
Fold 1	91.02%
Fold 2	90.88%
Fold 3	90.99%
Fold 4	90.94%
Fold 5	90.87%
Mean	90.94%

TABLE II
5-FOLD MEAN TEST ACCURACIES FOR KNN

The tables show that the models all perform well on the dataset; however, if we compare the two tables, it can be seen that the classification performs better.

VII. CONCLUSION

This paper presented a comparative evaluation of the *k*-Nearest Neighbours (kNN) algorithm and classification trees on a dataset containing several data quality challenges. A comprehensive preprocessing pipeline was implemented to address missing values, redundant and noisy features, extreme outliers, mixed feature types, and class imbalance. The kNN algorithm achieved optimal performance at $k = 3$, determined using the elbow method, while classification trees were optimised through pruning to improve generalisation. Using 5-fold cross-validation, classification trees achieved a mean accuracy of 94.78%, outperforming kNN with 90.94%. However, a Wilcoxon Signed-Rank Test with a significance level of $\alpha = 0.05$ showed no statistically significant difference between the models ($p = 0.0625$). These findings demonstrate that both algorithms provide reliable predictive performance when supported by appropriate data preprocessing techniques.

REFERENCES

[1] Evelyn Fix and J. L. Hodges. "Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties". In: *International Statistical Review / Revue Internationale de Statistique* 57.3 (1989), pp. 238–247. ISSN: 03067734, 17515823. URL: http://www.jstor.org/stable/1403797 (visited on 2024-10-24).

[2] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[3] R.A. Olshen Leo Breiman Jerome Friedman and Charles J. Stone. "Classification and Regression Trees". In: Taylor & Francis Group, 1984, p. 368. DOI: https://doi.org/10.1201/9781315139470.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[5] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[6] Michael L. Waskom. "seaborn: statistical data visualization". In: *Journal of Open Source Software* 6.60 (2021), p. 3021. DOI: 10.21105/joss.03021. URL: https://doi.org/10.21105/joss.03021.