Optimal Thresholds for Classification Trees using Nonparametric Predictive Inference

Masad A. Alrasheedi[a], Tahani Coolen-Maturi[b,*], Frank P.A. Coolen[b]

[a] *Department of Management Information Systems, Taibah University, Madinah, Saudi Arabia*
[b] *Department of Mathematical Sciences, Durham University, Durham, UK*

Abstract

In data mining, classification trees are used to assign a new observation to one of a set of predefined classes based on the attributes of the observation. They are constructed recursively through a top-down approach using repeated splits of the training dataset, which is a subset of the full data. When the dataset includes continuous-valued attributes, it is necessary to select appropriate threshold values to determine the classes and split the data. In recent years, Nonparametric Predictive Inference (NPI) has been introduced for selecting optimal thresholds for two- and three-class classification problems, where the inferences are explicitly based on a given number of future observations and target proportions. The NPI-based threshold selection method has previously been applied in the context of Receiver Operating Characteristic (ROC) analysis, but not for building classification trees. Due to its predictive nature, the NPI-based threshold selection method is well suited to classification tree construction, as the primary goal of such trees is prediction. In this paper, we present new classification algorithms for building classification trees using the NPI approach for selecting optimal thresholds. We introduce a new procedure for selecting the optimal target proportions by optimising classification performance on test data. Various measures are used to evaluate and compare the performance of the NPI_2-Tree and NPI_3-Tree classification algorithms with other methods from the literature. Experimental results show that our proposed algorithms perform well, achieving accuracies of 83.84% for the NPI_2-Tree and 82.87% for the NPI_3-Tree. These results suggest that the proposed classification algorithms may serve as viable alternatives to existing methods.

Keywords: Nonparametric predictive inference, Classification, Classification trees, Optimal Thresholds

1. Introduction

In data mining techniques, classification is used to assign new observations to one of a set of predefined classes based on the attributes of the observations. The goal of classification is to predict the unknown class states (or labels) of observations when their attribute variable values are known. There are several classification methods available in the literature that one

*Corresponding author

Email addresses: mrshedi@taibahu.edu.sa (Masad A. Alrasheedi), tahani.maturi@durham.ac.uk (Tahani Coolen-Maturi), frank.coolen@durham.ac.uk (Frank P.A. Coolen)

might use to predict the class states of observations [30, 37]. Classification (or decision) trees are one of the most commonly used because their rules are easy to understand and interpret with minimal experience. Classification trees are constructed recursively by a top-down scheme using repeated splits of the training datasets. When a dataset contains a continuous-valued attribute, it is necessary to select an appropriate threshold to determine the classes and split the data accordingly. Consequently, several methods have been developed in the literature using various approaches to find the optimal threshold value [13, 33, 34]. However, these methods typically follow similar strategies for threshold selection, primarily aiming to maximise classification accuracy on the training datasets [22].

In recent years, Nonparametric Predictive Inference (NPI) has been developed as a statistical methodology based on imprecise probability theory. NPI is a statistical approach based on Hill's assumption $A_{(n)}$ [28], with the use of lower and upper probabilities to make inferences about one or more future observations [9]. Due to the predictive nature of the NPI approach, it has been introduced for different applications in statistics, finance, operations research, survival analysis and reliability [10, 16, 17, 18]. Alabdulhadi [6] and Coolen-Maturi et al.[19] introduced NPI for selecting optimal thresholds for two- and three-class in the context of Receiver Operating Characteristic (ROC) applications, where the inferences are explicitly in terms of a given number of future observations from each class. They present a direct criterion that enables one to choose target proportions that reflect the relative importance of one class over another in ROC applications. For example, if giving medication to people with a disease is critical, and this medication has no serious harmful effects for people without the disease, one can give more weight to the correctly classified for the diseased class than for the healthy class. It would be expected that this will increase the proportion of correctly classified people with the disease while decreasing the proportion of correctly classified people without it. The NPI-based threshold selection method has been implemented in the context of ROC analysis [6, 19], but not for building classification trees.

In this paper, we present new classification algorithms for building classification trees using the NPI approach for selecting the optimal thresholds [6, 19]. We first present a new classification algorithm, which we call the NPI_2-Tree algorithm, for building binary classification trees; we then extend it to build classification trees with three ordered classes, which we call the NPI_3-Tree algorithm. The method of building classification trees using the NPI_2-Tree and NPI_3-Tree algorithms is novel in that it builds classification trees by employing the NPI approach for selecting the thresholds for data with two and three classes using predictive inference. In order to build a classification tree using our algorithms, we introduce a new procedure for selecting the optimal values of target proportions by choosing that to maximise classification performance on unseen data (testing data sets). We carry out experimental analysis on different data sets to evaluate and compare the performance of the NPI_2-Tree and the NPI_3-Tree algorithms with other classification algorithms using several evaluation measures.

The rest of the paper is organised as follows: Section 2 presents a general overview of classification trees and some classical methods for selecting the thresholds in classification trees. This is followed by background information about the NPI approach. Section 3 introduces the NPI method for selecting the optimal threshold for real-valued data and for the two-class classification scenario and provides an example to explain the overall methodology. In this section, a new classification algorithm, which we call the NPI_2-Tree algorithm, for

building binary classification trees using the NPI method for selecting the optimal threshold, is introduced. A new procedure for selecting the target proportions is presented to be used with the NPI_2-Tree algorithm for building the binary classification tree. Section 4 extends the method of building the classification tree from two classes to three ordered classes. A new classification algorithm, which we call the NPI_3-Tree algorithm, for building classification trees with three ordered classes, is introduced. Section 5 evaluates the performance of the NPI_2-Tree and the NPI_3-Tree algorithm on several data sets and compares the results with some other classification algorithms. Finally, Section 6 provides the conclusions and some interesting future works.

2. Preliminaries

2.1. Classification trees

A classification tree is a predictive model that provides a visual representation of the relationship between the attribute variables and the class variable. In general, a classification tree is built in the form of a tree structure that contains three main parts: a root node, internal nodes and leaf nodes [33, 34]. A root node is the topmost node in the tree with no incoming edges. An internal node has exactly one incoming edge and two or more outgoing edges. Finally, a leaf node has one incoming edge and no outgoing edges. In a classification tree, the internal nodes and the root node contain an attribute variable, each branch represents an outcome of the attribute variable, and each leaf node represents a class label. The paths from the root node to the leaf nodes denote the classification algorithm.

2.1.1. Split criteria

During the process of building the classification tree, a classification algorithm requires a split criterion, which is used to test all available attribute variables at each node of the tree and select the most useful one to split the data upon. The most commonly used classic split criteria are Information Gain (IG) and Information Gain Ratio (IGR), which were introduced by Quinlan [33, 34]. These split criteria are used to implement the ID3 [33] and the C4.5 [34] algorithms, respectively.

The IG is an impurity-based approach which uses entropy as an impurity measure. The formula of entropy [38], also called the Shannon Entropy, for a training data set S with a class variable C, is given by

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

where p_i is the proportion of the training data set S belonging to class C_i, for $i = 1, \ldots, K$. So, K is the total number of classes. The IG of an attribute variable X with different values $\{x_1, \ldots, x_r\}$, relative to the training set S and the class variable C is given by

$$\mathrm{IG}(C, X) = H(C) - H(C|X) \tag{2}$$

where $H(C|X)$ is the entropy of class C given attribute variable X, and is defined as

$$H(C|X) = \sum_{i=1}^{r} P(X = x_i) H(C|X = x_i) \tag{3}$$

In order to choose the best attribute variable for splitting the data at each node of the tree, the IG is used as a split criterion by the ID3 algorithm. It assigns the attribute variable for which the IG is maximum for the root node. Then, it splits the training set into two or more subsets based on the values of the chosen attribute, and then for each subset, it repeats the process. It has been proved that the IG split criterion is biased to attribute variables that have a large number of states, which could negatively affect the performance of the ID3 algorithm [33]. Therefore, the Information Gain Ratio (IGR) was introduced by Quinlan in 1993 [34] to overcome this weakness by using a normalisation of the IG. The IGR of an attribute X and a class variable C is given by:

$$\mathrm{IGR}(C, X) = \frac{\mathrm{IG}(C, X)}{\mathrm{SI}(X)} \tag{4}$$

where $\mathrm{IG}(C, X)$ is given by Equation (2), and $\mathrm{SI}(X)$ is called the split Information, which is the entropy of the variable which does not depend on C, it is given by:

$$\mathrm{SI}(X) = -\sum_{i=1}^{r} P(X = x_i) \log_2 P(X = x_i) \tag{5}$$

Unlike the ID3, the C4.5 can also handle numerical attributes. For a training data set S and a continuous-valued attribute X with n distinct values in the ordinal sequence $\{v_1, \ldots, v_n\}$. The C4.5 algorithm uses a binary split on X to evaluate each midpoint between adjacent values v_i and v_{i+1} (for $i = 1, \ldots, n-1$), by computing the IGR, as given by Equation (4). A threshold value that maximises the IGR criterion is selected as the optimal threshold for attribute X. After selecting the threshold value, the training data set is partitioned into two subsets based on the threshold value. The C4.5 algorithm continues recursively by evaluating each midpoint between adjacent values for each new subset and selecting new thresholds for each branch.

Another split criterion is the Gini Index (GI) was introduced by Breiman [13] as a split criterion for the Classification And Regression Tree (CART) algorithm. The CART algorithm uses a binary split when building trees, meaning that each internal node in the classification tree can have only two branches. Finally, there is an imprecise split criterion, which is Imprecise Information Gain (IIG), and introduced by Abellán and Moral [2]. It is similar to the IG split criterion that is used in the ID3 algorithm, but the precise probabilities and entropy function have been replaced with imprecise probabilities and maximum entropy function. The IIG split criterion can use the maximum entropy distributions from the credal set obtained from the Imprecise Dirichlet Model (IDM) [1] or from Nonparametric Predictive Inference for Multinomial data (NPI-M) [3, 5]. In this paper, we refer to a classification tree created with the IIG and the IDM by the IDM algorithm, and with the IIG and the NPI-M by the NPI-M algorithm. Further details and explanations of these models are given in [1, 3]

2.2. NPI for real-valued observations

Nonparametric Predictive Inference (NPI) is a statistical methodology based on Hill's assumption $A_{(n)}$ [28], which gives direct probabilities for one or more future observations

based on n observed values of related random quantities. Inference based on $A_{(n)}$ is non-parametric and predictive. It was introduced particularly for situations where there is no prior information about the probability distribution for a random quantity of interest, or in cases where one explicitly does not want to use any such information. Let $X_1, \ldots, X_n, X_{n+1}$ be exchangeable real-valued random quantities. Suppose that the ordered observed values of $X_1, X_2, \ldots X_n$ are denoted by $x_1 < x_2 \ldots < x_n$, where the assumption is made that there are no ties between observations. For ease of notation, let $x_0 = -\infty$ and $x_{n+1} = \infty$. Note that $x_{n+1} = \infty$ is not an observation of the variable X_{n+1}. These n ordered observations divide the real-line into $n+1$ open intervals $I_j = (x_{j-1}, x_j)$, for $j = 1, 2, \ldots, n+1$. The assumption $A_{(n)}$ states that the future observation X_{n+1} falls equally likely in any interval (x_{j-1}, x_j), for each $j = 1, 2, \ldots, n+1$,

$$P(X_{n+1} \in I_j) = \frac{1}{n+1} \tag{6}$$

It is important to emphasise that Hill's assumption $A_{(n)}$ does not make any further assumptions on the distribution of probability $\frac{1}{n+1}$ within an interval I_j. NPI uses $A_{(n)}$ for predictive inferences about future observations in the form of lower and upper probabilities, also known as imprecise probabilities. Augustin and Coolen [9] introduced predictive lower and upper probabilities for events of interest based on assumption $A_{(n)}$, which is essentially an application of De Finetti's fundamental theorem of probability [20]. The lower and upper probabilities for the event $X_{n+1} \in B$, with $B \subset \mathbb{R}$, based on the intervals I_j, $j = 1, 2, \ldots, n+1$, and Hill's assumption $A_{(n)}$, are given by

$$\underline{P}(X_{n+1} \in B) = \frac{1}{n+1} \sum_{j=1}^{n+1} \mathbf{1}\{I_j \subseteq B\} \tag{7}$$

$$\overline{P}(X_{n+1} \in B) = \frac{1}{n+1} \sum_{j=1}^{n+1} \mathbf{1}\{I_j \cap B \neq \emptyset\} \tag{8}$$

where $\mathbf{1}\{A\}$ is equal to 1 if A is true and equal to 0 else. The NPI lower probability (7) is obtained by taking only probability mass into account that is necessary within B, which is only the case for the probability mass $\frac{1}{n+1}$ per interval I_j if this interval is totally contained within B. The NPI upper probability (8) is obtained by taking all probability mass into account that could possibly be within B, which is the case for the probability mass $\frac{1}{n+1}$ per interval I_j if the intersection of I_j and B is non-empty.

We are interested in $m \geq 1$ future observations, X_{n+i} for $i = 1, \ldots, m$ [14]. The data and future observations are linked by consecutive application of $A_{(n)}, A_{(n+1)}, \ldots, A_{(n+m-1)}$ [28]. These together are referred to as the $A_{(.)}$ assumptions, which can be considered a post-data version of a finite exchangeability assumption for $n+m$ random quantities. The $A_{(.)}$ assumptions imply that all possible orderings of n data observations and m future observations are equally likely, where the n data observations are not distinguished among each other and neither is the m future observations. Let $S_j = \#\{X_{n+i} \in I_j, i = 1, \ldots, m\}$, then assuming $A_{(.)}$ we have [14]

$$P(\bigcap_{j=1}^{n+1} \{S_j = s_j\}) = \binom{n+m}{n}^{-1} \tag{9}$$

where s_j are non-negative integers with $\sum_{j=1}^{n+1} s_j = m$. Equation (9) implies that all $\binom{n+m}{n}$ orderings of m future observations among the n observations are equally likely. Let $X_{(r)}$, for $r = 1, \ldots, m$, be the r-th ordered future observation, so $X_{(r)} = X_{n+i}$ for one $i = 1, \ldots, m$ and $X_{(1)} < X_{(2)} < \cdots < X_{(m)}$. The probabilities given in Equation (10) are based on Equation (9) and derived by counting the relevant orderings, and hold for $j = 1, \ldots, n+1$, and $r = 1, \ldots, m$ [14],

$$P(X_{(r)} \in I_j) = \binom{j+r-2}{j-1}\binom{n-j+1+m-r}{n-j+1}\binom{n+m}{n}^{-1} \tag{10}$$

For this $X_{(r)} \in I_j$, NPI gives a precise probability, as each of the $\binom{n+m}{n}$ equally likely orderings of n past and m future observations has the r-th ordered future observation in precisely one interval I_j [15]. The event that the number of future observations in an interval (x_α, x_β), with $1 \leq \alpha < \beta \leq n+1$, and denoted by $S^m_{\alpha,\beta}$, is greater than or equal to a particular value $v \in \mathbb{N}$, has the following precise probability [7],

$$P(S^m_{\alpha,\beta} \geq v) = \sum_{i=v}^{m} \binom{n+m}{n}^{-1}\binom{\beta-\alpha-1+i}{i}\binom{n-\beta+\alpha+m-i}{m-i} \tag{11}$$

We will make use of these results in Sections 3 and 4 for selecting the optimal thresholds.

3. NPI-based binary classification trees

In this section, we first present the NPI method for selecting the optimal threshold for the two-class classification problem, and then illustrate the method with an example. Then, we introduce a new procedure for choosing the optimal values of target proportions in classification trees. Finally, we present our classification tree algorithm, which we call the NPI$_2$-Tree algorithm, and illustrate the method with an example.

3.1. NPI-based threshold selection for two classes

The NPI method for selecting the optimal threshold t for a real-valued random quantity and for a two-class classification scenario has been introduced by Alabdulhadi [6] and Coolen-Maturi et al. [19]. This method is different from the classical methods as it selects the optimal threshold value, which focuses on a number of future observations to which the threshold will be applied. Assume that we have a continuous random variable X whose values belong to two classes, C_1 and C_2, and small values of X are more likely to belong to class C_1, i.e. $X \leq t$, and large values of X are more likely to belong to class C_2, i.e. $X > t$. Let n_1 denote the number of observations for class C_1 and n_2 the number of observations for class C_2. It is assumed that there is full independence between the two classes, meaning that any information about the observations in one class does not contain information about the observations in the other class. In other words, any information about random quantities from one class does not affect any (lower and upper) probabilities for events involving only random quantities of the other class. Let $x^1_1 < x^1_2 < \cdots < x^1_{n_1}$ denote the ordered data from class C_1 and $x^2_1 < x^2_2 < \cdots < x^2_{n_2}$ denote the ordered data from class C_2. For ease of

notation, let $x_0^1 = x_0^2 = -\infty$ and $x_{n_1+1}^1 = x_{n_2+1}^2 = \infty$. The data for C_1 partition the real-line into $n_1 + 1$ intervals $I_i^1 = (x_{i-1}^1, x_i^1)$, for $i = 1, 2, \ldots, n_1 + 1$, and the data for C_2 partition the real-line into $n_2 + 1$ intervals, for $I_j^2 = (x_{j-1}^2, x_j^2)$, for $j = 1, \ldots, n_2 + 1$. Throughout this paper, it is assumed that there are no ties between the data observations, which occur when two or more observations have the same value. We can break the ties by adding a small amount to the tied observations, which tend to be zero. This is a popular method for breaking ties in statistics [27].

As the NPI inferences are based on multiple future observations, we consider m_1 future observations from class C_1, with data values denoted by $X_{n_1+r}^1$, where $r = 1, \ldots, m_1$, and m_2 future observations from class C_2, with data values denoted by $X_{n_2+s}^2$, where $s = 1, \ldots, m_2$. Let the m_1 ordered future observations from classes C_1 be denoted by $X_{(1)}^1 < X_{(2)}^1 < \cdots < X_{(m_1)}^1$, and the m_2 ordered future observations from class C_2 be denoted by $X_{(1)}^2 < X_{(2)}^2 < \cdots < X_{(m_2)}^2$. As the NPI-based inferences are in terms of multiple future observations, Alabdulhadi [6] and Coolen-Maturi et al. [19] have selected the threshold value t that gives the best classification based on the m_1 and m_2 future observations. To this end, the result of NPI for multiple future observations presented in Section 2.2 is used, but we need first to introduce further notation.

For a particular value of t, we denote the number of correctly classified future observations from class C_1 by L_t^1, that is those with data values $X_{n_1+r}^1 \leq t$, for $r = 1, \ldots, m_1$, and we denote the number of correctly classified future observations from class C_2 by L_t^2, that is those with data values $X_{n_2+s}^2 > t$, for $s = 1, \ldots, m_2$. Let a and b be any two values in the range $(0, 1]$ that are chosen to represent the relative importance of the correct classification of each one of the classes. We consider the general event of interest that the number of correctly classified future observations from class C_1 is at least am_1 and the number of correctly classified future observations from class C_2 is at least bm_2, that is $L_t^1 \geq am_1$ and $L_t^2 \geq bm_2$. Note that the choice of a and b depends on a person's beliefs of which class may be more important to be correctly classified than another; hence, the values of these proportions are not constrained except that they must be in $(0, 1)$. Of course, one can choose a and b to be equal if one gives the same importance of correct classifications to both classes, but choosing large or small values for a and b may not change classification performance, as shown in Example 3.1.1.

Due to the independence assumption of the two classes, the joint NPI lower and upper probabilities are derived as the products of the NPI corresponding lower and upper probabilities for the events that involve L_t^1 or L_t^2 as follows [6, 19]

$$\underline{P}\left(L_t^1 \geq am_1, L_t^2 \geq bm_2\right) = \underline{P}\left(L_t^1 \geq am_1\right) \times \underline{P}\left(L_t^2 \geq bm_2\right) \tag{12}$$

$$\overline{P}\left(L_t^1 \geq am_1, L_t^2 \geq bm_2\right) = \overline{P}\left(L_t^1 \geq am_1\right) \times \overline{P}\left(L_t^2 \geq bm_2\right) \tag{13}$$

The NPI lower and upper probabilities in Equations (12) and (13) are derived using NPI for multiple future observations as given in Section 2.2, in particular Equation (10), as shown below. It is noticed that the event $L_t^1 \geq am_1$ is equal to $X_{(\lceil am_1 \rceil)}^1 \leq t$, where $\lceil am_1 \rceil$ denotes the smallest integer greater than or equal am_1. Similarly, the event $L_t^2 \geq bm_2$ is equal to $X_{(m_2 - \lceil bm_2 \rceil + 1)}^2 > t$. To show how to use the Equation (10) of the NPI results for multiple future observations, we first consider class C_1 and then class C_2.

For $I_i^1 = (x_{i-1}^1, x_i^1), i = 1, \ldots, n_1 + 1$, and $t \in I_{i_t}^1 = (x_{i_t-1}^1, x_{i_t}^1)$, where $i_t = 2, \ldots, n_1$ is defined as that interval $I_{i_t}^1$ which contains t, the NPI lower and upper probabilities for the event $L_t^1 \geq am_1$ are given as follow [6, 19]:

$$\underline{P}(L_t^1 \geq am_1) = \underline{P}(X_{(\lceil am_1 \rceil)}^1 \leq t) = \sum_{i=1}^{i_t - 1} P(X_{(\lceil am_1 \rceil)}^1 \in I_i^1) \tag{14}$$

$$\overline{P}(L_t^1 \geq am_1) = \overline{P}(X_{(\lceil am_1 \rceil)}^1 \leq t) = \sum_{i=1}^{i_t} P(X_{(\lceil am_1 \rceil)}^1 \in I_i^1) \tag{15}$$

where the precise probabilities on the right-hand sides of Equations (14) and (15) are obtained from Equation (10). For $i_t = 1$, we have $\underline{P}(L_t^1 \geq am_1) = 0$ and $\overline{P}(L_t^1 \geq am_1) = P(X_{(\lceil am_1 \rceil)}^1 \in I_1^1)$, and for $i_t = n_1 + 1$, we have $\underline{P}(L_t^1 \geq am_1) = 1 - P(X_{(\lceil am_1 \rceil)}^1 \in I_{n_1+1}^1)$ and $\overline{P}(L_t^1 \geq am_1) = 1$. If t is equal to one of the observations x_i^1, i.e. $t = x_{i_t}^1$, then this event has precise probability,

$$P(L_t^1 \geq am_1) = P(X_{(\lceil am_1 \rceil)}^1 \leq t) = \sum_{i=1}^{i_t} P(X_{(\lceil am_1 \rceil)}^1 \in I_i^1) \tag{16}$$

Of course, this implies that we have for such a value of t that $\underline{P}(L_t^1 \geq am_1) = \overline{P}(L_t^1 \geq am_1) = P(L_t^1 \geq am_1)$, in this case. Similarly, the NPI lower and upper probabilities for the event $L_t^2 \geq bm_2$ are derived. For $I_j^2 = (x_{j-1}^2, x_j^2), j = 1, \ldots, n_2 + 1$, and $t \in I_{j_t}^2 = (x_{j_t-1}^2, x_{j_t}^2), j_t = 2, \ldots, n_2$, the NPI lower and upper probabilities for the event $L_t^2 \geq bm_2$ are

$$\underline{P}(L_t^2 \geq bm_2) = \underline{P}(X_{(m_2 - \lceil bm_2 \rceil + 1)}^2 > t) = \sum_{i=j_t+1}^{n_2+1} P(X_{(m_2 - \lceil bm_2 \rceil + 1)}^2 \in I_j^2) \tag{17}$$

$$\overline{P}(L_t^2 \geq bm_2) = \overline{P}(X_{(m_2 - \lceil bm_2 \rceil + 1)}^2 > t) = \sum_{i=j_t}^{n_2+1} P(X_{(m_2 - \lceil bm_2 \rceil + 1)}^2 \in I_j^2) \tag{18}$$

For $j_t = 1$, we have

$$\underline{P}(L_t^2 \geq bm_2) = 1 - P(X_{(m_2 - \lceil bm_2 \rceil + 1)}^2 \in I_1^2) \text{ and } \overline{P}(L_t^2 \geq bm_2) = 1.$$

While for $j_t = n_2 + 1$, we have

$$\underline{P}(L_t^2 \geq bm_2) = 0 \text{ and } \overline{P}(L_t^2 \geq bm_2) = P(X_{(m_2 - \lceil bm_2 \rceil + 1)}^2 \in I_{n_2+1}^2)$$

Furthermore, when $t = x_{j_t}^2$

$$P(L_t^2 \geq bm_2) = P(X_{(m_2 - \lceil bm_2 \rceil + 1)}^2 > t) = \sum_{i=j_t+1}^{n_2+1} P(X_{(m_2 - \lceil bm_2 \rceil + 1)}^2 \in I_j^2) \tag{19}$$

so

$\underline{P}(X_{(m_2 - \lceil bm_2 \rceil + 1)}^2 > t) = \overline{P}(X_{(m_2 - \lceil bm_2 \rceil + 1)}^2 > t) = P(X_{(m_2 - \lceil bm_2 \rceil + 1)}^2 > t)$, if $t = x_{j_t}^2$.

Now, we can obtain the optimal threshold t for the two classes by maximising either Equation (12) for the lower probability or Equation (13) for the upper probability. To search for the optimal threshold value t, one does not need to go through each of the $n_1 + n_2 + 1$ intervals produced by the data observations. The optimal threshold t can be only in intervals where the left-end point of the interval is an observation that belongs to class C_1, and the right-end point of the interval is an observation that belongs to class C_2 [6, 19]. It is important to clarify that the NPI lower and NPI upper probabilities, Equations (12) and (13), may lead to different optimal thresholds because they are different criteria. In this paper, we consider only the optimal threshold value based on the NPI lower probability, Equation (12), for building classification trees. This is because the NPI lower probabilities are based on evidence in favour of events while the NPI upper probabilities are based on evidence against events. Next, we provide an example illustrating the NPI method for selecting the optimal threshold. For more details, examples and discussions of NPI for selecting the optimal thresholds, we refer to [6, 19].

3.1.1. Example

Assume that we have a data set of 20 people, where 10 people from class C_1, i.e. $n_1 = 10$, and 10 people from class C_2, i.e. $n_2 = 10$. Suppose the data set that belongs to class C_1 are {25, 27, 28, 29, 30, 36, 37, 40, 63, 68 } and the data set that belongs to class C_2 are {48, 53, 67, 70, 73, 75, 82, 86, 89, 90}. In order to illustrate the NPI method for selecting the threshold values, we have presented the NPI method for $m_1 = m_2$ and for $m_1 \neq m_2$, and we have considered four different scenarios for target proportions a and b. Note that these target proportions were previously chosen to represent the relative importance of the correct classification of one class over another.

To select the optimal threshold t, we need to search within each of the n_1+n_2+1 intervals created by the data observations, and then we choose the value t that maximises the NPI lower probabilities method, Equation (12), or the NPI upper probabilities method, Equation (13). Note that as shown in [6, 19], the optimal threshold value t can only be found in intervals in where the left-end point of the interval is an observation that belongs to class C_1, and the right-end point is an observation that belongs to class C_2. The first and last intervals should also be considered. Thus, we just consider the intervals as shown in [6, 19]. Table 1 presents the optimal threshold value t obtained from the NPI lower probabilities method, Equation (12), and the NPI upper probabilities method, Equation (13), along with the corresponding lower and upper probabilities, for $m_1 \neq m_2$, while Table 2 presents the optimal threshold value t obtained from the NPI lower and upper probabilities method along with the corresponding lower and upper probabilities, for $m_1 = m_2$.

As shown in Table 1, the optimal threshold value t differs for different values of a and b. In Scenario 1, for $a = b = 0.25$, the optimal threshold value is $t = 40$. In this scenario, the values of lower and upper probabilities for the NPI method are quite high because the required proportions seem easy to achieve. In Scenario 2, we put more emphasis on the number of correctly classified future observations from class C_1 than from class C_2, that is, $a = 0.50$ and $b = 0.25$. As we can see from Table 1, the optimal threshold value increased to

Scenario	Target proportions		NPI lower method		NPI upper method	
#	a	b	t	value	t	value
$m_1 = 4$, $m_2 = 6$						
1	0.25	0.25	40	0.98	40	0.99
2	0.50	0.25	63	0.93	63	0.98
3	0.85	0.20	68	0.67	68	0.97
4	0.70	0.70	40	0.28	40	0.51
$m_1 = 100$, $m_2 = 80$						
1	0.25	0.25	40	0.99	40	1.00
2	0.50	0.25	68	0.99	68	1.00
3	0.85	0.20	68	0.80	68	0.99
4	0.70	0.70	40	0.60	40	0.84

Table 1: Optimal threshold t and corresponding value of the NPI lower and upper probabilities, for $m_1 \neq m_2$.

$t = 63$ in order to optimise the lower and upper probabilities, compared to Scenario 1. For Scenario 3, for $a = 0.85$ and $b = 0.20$, we again need this scenario to give a higher proportion of correctly classified future observations from class C_1 than from class C_2. The optimal threshold value has increased to $t = 68$ compared to Scenario 2, while the corresponding NPI lower probability is smaller than the one in Scenario 2. This indicates that achieving these proportions a and b jointly is more difficult. In Scenario 4, for $a = b = 0.70$, the optimal threshold value is the same as for Scenario 1, where we have large values of $a = b$. Of course, the corresponding values of the NPI lower and upper probabilities are lower than those in Scenario 1 as a and b here are larger. For $m_1 = 100$ and $m_2 = 80$, with respect to the optimal threshold, the optimal thresholds are found to be the same as for $m_1 = 4$ and $m_2 = 6$ regardless of the values of a and b, except for $a = 0.50$ and $b = 0.25$, the optimal threshold is $t = 68$. The corresponding lower and upper probabilities are very high compared to $m_1 = 4$ and $m_2 = 6$.

The results of the NPI method for selecting the optimal threshold value t, as well as their corresponding NPI lower and upper probabilities for $m_1 = m_2$, are presented in Table 2. We have used the same scenarios as in Table 1. For $m_1 = m_2 = 8$, the results are quite similar to the results in Table 1. With respect to the optimal threshold, we found that the optimal threshold for $a = b = 0.25$, $a = 0.85$, $b = 0.20$, and $a = b = 0.70$ are the same in both tables. The corresponding NPI lower and upper probabilities in Table 2 are slightly larger than in Table 1. A change in the NPI lower and upper probabilities is here due to the nature of the event we consider; for example, for $m_2 = 6$ and $m_2 = 8$, and $b = 0.25$, we need at least 2 good classifications in both cases, which is easier for $m_2 = 8$ than for $m_2 = 6$, so then for the latter, the lower and upper are probabilities smaller. We can also note from Tables 1 and 2 that the NPI lower and the upper methods provide the same optimal threshold regardless of the a and b values considered, which is likely because the data sets used in Example 3.1.1 are small with little overlapping. For $m_1 = m_2 = 100$, the results are the same as for $m_1 = 100, m_2 = 80$, given in Table 1, for all scenarios of a and b.

Scenario	Target proportions		NPI lower method		NPI upper method	
#	a	b	t	value	t	value
$m_1 = 8$, $m_2 = 8$						
1	0.25	0.25	40	0.99	40	1.00
2	0.50	0.25	68	0.97	68	0.99
3	0.85	0.20	68	0.79	68	0.99
4	0.70	0.70	40	0.31	40	0.58
$m_1 = m_2 = 100$						
1	0.25	0.25	40	0.99	40	1.00
2	0.50	0.25	68	0.99	68	1.00
3	0.85	0.20	68	0.80	68	0.99
4	0.70	0.70	40	0.60	40	0.84

Table 2: Optimal threshold t and corresponding value of the NPI lower and upper probabilities, for $m_1 = m_2$.

3.2. Selecting the target proportions for two classes

Instead of prefixing the target proportions a and b, as in Example 3.1.1, we introduce a data-driven approach for selecting a and b, which aims to improve the classification performance. Consider the NPI method for selecting the optimal threshold which is based on the NPI lower probability, Equation (12),

$$\underline{P}(L_t^1 \geq am_1, L_t^2 \geq bm_2) = \underline{P}(L_t^1 \geq am_1) \times \underline{P}(L_t^2 \geq bm_2)$$

where $\underline{P}(L_t^1 \geq am_1)$ and $\underline{P}(L_t^2 \geq bm_2)$ are given in Equations (14) and (17), respectively, and $a, b \in (0, 1]$. We now consider a and b as parameters instead of desirable target proportions, and we aim to choose values for a and b that maximise classification performance. Note that there is no conflict between saying these values represent one's beliefs of which class may be important to classify correctly, and choosing these by optimisation. The first approach is useful if one would prefer to give one class more importance than another, whereas the second approach works if one aims to maximise the total classification accuracy. The main questions are how to find or select these values a and b and how to validate their performance in classification trees. To this end, we suggest using two stages of the k-fold cross-validation (k-fold CV) procedure, also known as double k-fold cross-validation [39]. In k-fold CV procedure, the classification algorithm is trained and evaluated using several different subsets of the data set instead of one. The data sets are randomly divided into k subsets of approximately equal size, called folds. Each fold of the k folds is used as a testing set to evaluate the performance of the classification algorithm, and the $k-1$ remaining folds are mixed together to use as a training set. This method is performed k times so that training and testing are performed k times. In the end, the classification accuracy is computed by taking the average of the k classification accuracies attained from the k test sets. This procedure enables us to train our classification method in which the values of a and b also need to be optimised. Without this procedure, one can use the same data for finding the optimal values of a and b and simultaneously evaluate their performance, which may result in a biased evaluation of the algorithm [39].

Figure 1 presents the diagram of the proposed procedure, which is the two stages of the k-fold cross-validation procedure, where $k = 5$. We have chosen $k = 5$ because it reduces

the required computation as done in [11]. As shown in Figure 1, there is an outer 5-fold cross-validation loop, which is used to validate our method of selecting the parameters a and b. In addition to the outer loop, there is an inner 5-fold cross-validation loop that is used to optimise the parameters a and b. The outer loop is repeated 5 times, producing five different training and testing sets resulting from the entire dataset. Each fold of the outer training set is again divided into 5 folds, and the inner loop is repeated 5 times as well. The inner loop will return only the model with the most optimal values for a and b to the outer loop, which will use its testing set to evaluate the model's quality. In the outer loop, we will get 5 different performances that can be averaged to obtain the final performance. We then extract the optimal parameters a and b from the outer folds and use them as target proportions for the data set.

For more explanation, we present the method step by step as follows. First, we randomly divide the data into five folds, $k = 5$, each containing a training and testing set. Secondly, each outer training fold (starting with outer training fold 1, as in the red box in A) is again divided into five inner folds, each containing training and testing set as in B. In the inner folds, as in B, we discover possible optimal values of a and b using optimisation techniques. So, we have five possible values of parameters a and b. After that, we choose the values for the parameters a and b from the inner folds that give the best classification accuracy on inner testing folds to test on the outer test fold in A (for example, test fold 1). There are many optimisation methods in the literature that can be used to discover these optimal values, where a Genetic Algorithm (GA) [25, 26] is one of the most commonly used optimisation methods in the literature. The GA is a search-based optimisation technique based on the rules of genetics and natural selection to provide solutions to problems. We use the GA as additional tuning in order to discover the best values of a and b in the inner stage. More details about how the GA method works to tune the values of a and b are given in [8]. Thirdly, as in C, we record the result of this outer fold, including the values of a and b and the classification accuracy. Then, we repeat this process for the remaining outer folds. Finally, as in D, we choose the best values of a and b that give the highest classification accuracy; we then use these values as the optimal values for the data set. Note that we build the whole classification tree in this process using all the available attributes, not based on a single attribute. So this is a joint optimisation problem, where at each stage we build a full tree. If the number of such attributes increases, then we probably have an exponential increase in the computation time for the optimisation. Therefore, it would be of interest to investigate the use of suitable fast optimisation techniques; this is left as a topic for future research.

3.3. NPI$_2$-Tree algorithm

Having explained the process of choosing the values of a and b, we now present a new algorithm, which we call the Nonparametric Predictive Inference for Binary Classification Trees (NPI$_2$-Tree) algorithm. This algorithm is used for building binary classification trees using the NPI approach for selecting the optimal thresholds and the proposed method for choosing a and b. Note that this algorithm can only be used to build trees with two classes. For this reason, we added the number 2 to the abbreviation. The procedure for building classification trees using the NPI$_2$-Tree algorithm is quite similar to Quinlan's C4.5 algorithm

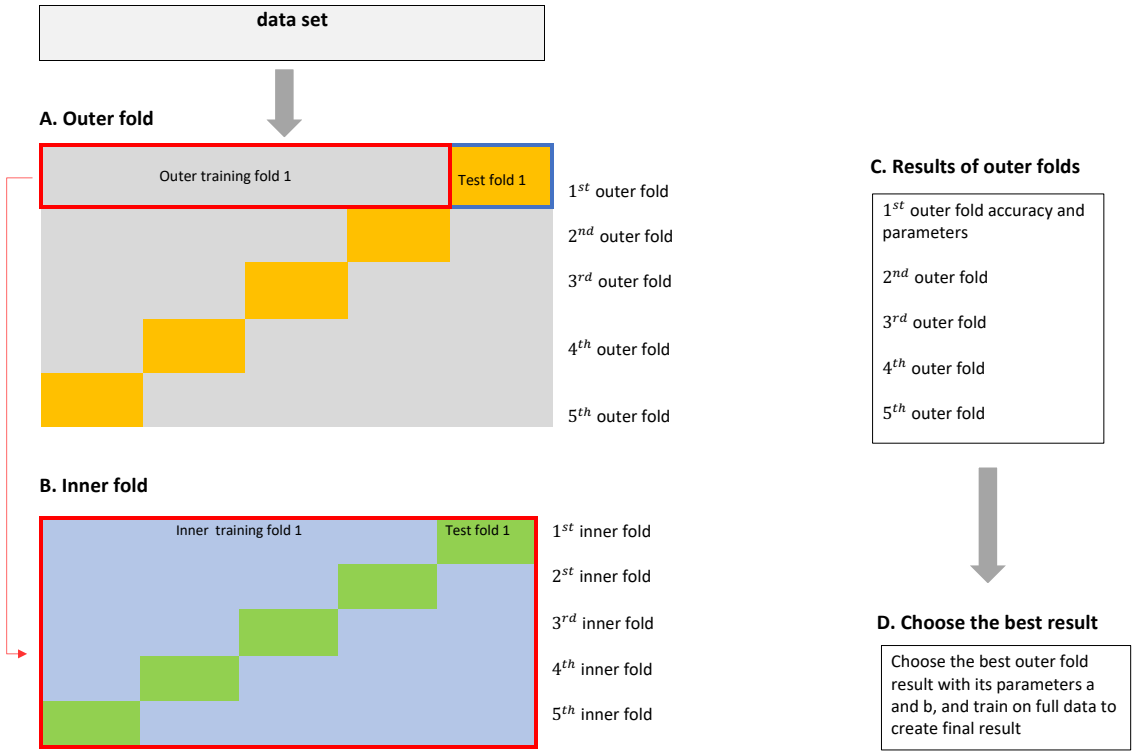


Figure 1: A diagram of a two-stage 5-fold cross-validation procedure to find the optimal values of the target proportions a and b.

[34], given in Section 2.1.1. The main difference is that we use the NPI method presented in Section 3.1 as a criterion for selecting the optimal threshold for each continuous-valued attribute with our proposed method of choosing the values of a and b presented in Section 2.1.1. Note that the NPI$_2$-Tree algorithm uses the information gain ratio as a split criterion to select the attribute variable at each node.

Suppose that we have a data set, $\mathcal{D}$, which has continuous-valued attributes $\{X_1, \ldots, X_f\}$, and a binary class variable, $C \in \{C_1, C_2\}$. As a first step, we divide the data set, $\mathcal{D}$, into two subsets: training data set, S, with n observations and testing data set, T, with m observations. Let n_1 represent the total number of observations that belong to class C_1, and n_2 represent the total number of observations that belong to class C_2. We set the number of future observations, m_1 and m_2, based on the T data distribution; that is, we choose the value of m_1 equal to the number of observations that belong to class C_1 in T and the value of m_2 equal to the number of observations that belong to class C_2 in T. As a starting point, we set the initial values of the a and b equal to the data proportion, meaning that we choose the value of a equal to the proportion of the training data S belonging to class C_1 and the value of b equal to the proportion of S belonging to class C_2. For the training data set S, we find the optimal threshold values for each of the continuous-valued attributes, X_i, for $i = 1, \ldots, f$, by maximising the NPI lower probability given in Equation (12). As shown in [6, 19], the optimal threshold value t can only be found in intervals in where the left-end point of the interval is an observation that belongs to class C_1 and the right-end point is an observation that belongs to class C_2 [6, 19]. The first and last intervals should also be considered. This property is useful when building classification trees because it speeds up the process of determining the optimal thresholds.

After selecting the optimal threshold t for X_i, we compute the IGR value for all attributes X_i, for $i = 1, \ldots, f$, to find the best attribute variable for the root node. Once the IGR values are computed for all attributes, the attribute variable with the highest IGR value is chosen as the best attribute for the root node. Then, based on the chosen attribute's threshold, we split the training data set S into two disjoint subsets, S_1 and S_2, where $S_1 \cup S_2 = S$ and $S_1 \cap S_2 = \emptyset$. For example, the IGR for attribute variable X_i can be defined as follows:

$$\mathrm{IGR}(S, X_i) = \frac{\mathrm{IG}(S, X_i)}{H(S, X_i)} \tag{20}$$

where

$$\mathrm{IG}(S, X_i) = H(S) - \left(\frac{|S_1|}{|S|} H(S_1) + \frac{|S_2|}{|S|} H(S_2) \right) \tag{21}$$

and

$$H(S, X_i) = -\left(\frac{|S_1|}{|S|} \log_2 \frac{|S_1|}{|S|} + \frac{|S_2|}{|S|} \log_2 \frac{|S_2|}{|S|} \right) \tag{22}$$

where S_1 is the subset of the training data set S with $X_i \leq t$, and S_2 is the subset of the data with $X_i > t$. It is important to say that during the process of building the classification trees by the NPI$_2$-Tree algorithm, we assume that small data values are more likely to be from C_1 and large data values from C_2, i.e. $X_i \leq t$ from class C_1 and $X_i > t$ from class C_2.

After selecting the best attribute variable at the root node and splitting the training data set into two subsets, we then again find the optimal thresholds and the IGR values for both subsets S_1 and S_2. The NPI$_2$-Tree algorithm continues recursively by selecting further splitting the data, and hence, the algorithm creates new subtrees for each branch. The tree branching is stopped when all observations in the subset belong to a single class, or if there is no attribute left, or when the number of observations per leaf node reaches the minimum split value. A minimum split number is a value that must exist before splitting the data. Algorithm 1, given in Appendix A, summarises the process of building binary classification trees using the NPI$_2$-Tree classification algorithm.

4. NPI-based classification trees with three ordered classes

This section extends the method for building binary classification trees, presented in Section 3, to build classification trees with three classes. Throughout this section, we assume that there is a natural ordering of the three classes, where observations from class C_1 tend to be smaller than observations from class C_2, which in turn tend to be smaller than observations from class C_3. Therefore, in order to determine the classes and split the data, there is a need to select two optimal thresholds, $t_1 < t_2$, for continuous-valued data. We first present the NPI method for selecting the optimal thresholds for data with three ordered classes, and we provide an example to illustrate this method. We then extend the procedure of choosing the target proportions for cases of two-class setting, presented in Section 3.2, to cases of three-class setting. Finally, we present a new classification algorithm, which we call the NPI$_3$-Tree algorithm, for building classification trees with three ordered classes.

4.1. NPI-based thresholds selection for three ordered classes

In Section 3.1, we have presented the results of the NPI method for selecting the optimal threshold for two classes. In this section, we present the results of the NPI method for selecting the optimal thresholds for three ordered classes, introduced in [6, 19], with an illustrative example. As explained in [6, 19], one could naively use the NPI method, presented in Section 3.1, twice, to find the optimal thresholds t_1 and t_2 for the three classes, i.e. one can find t_1 using C_1 and C_2 and then find t_2 using C_2 and C_3. However, because of the assumed ordering of the three classes, selecting the two thresholds in this way may not satisfy the condition that $t_1 < t_2$. Therefore, we will not do that; instead, we will use the NPI method for selecting the optimal thresholds for three ordered classes, which finds the optimal thresholds t_1 and t_2 simultaneously. First, we summarise the results of [6, 19] using the same notation as presented in Section 3.1, but with additional notation for class C_3.

Let n_3 denote the number of observations in class C_3, the ordered data from this class are denoted by $x^3_1 < x^3_2 < \cdots < x^3_{n_3}$. For ease of notation, we define $x^3_0 = -\infty$ and $x^3_{n_3+1} = \infty$. Again, the n_3 observations divide the real-line into $n_3 + 1$ intervals $I^3_l = (x^3_{l-1}, x^3_l)$, for $l = 1, 2, \ldots, n_3 + 1$. Let m_3 denote the number of future observations in class C_3, with random variable $X^3_{n_3+d}$, for $d = 1, \ldots, m_3$. Let the m_3 ordered future observations from class C_3 be denoted by $X^3_{(1)} < X^3_{(2)} < \cdots < X^3_{(m_3)}$. To classify observations into one of the classes, C_1, C_2 or C_3, we want to find the two optimal thresholds t_1 and t_2, where $t_1 < t_2$, such that

observations less than or equal to t_1 are classified as belonging to C_1, observations greater than t_1 and less than or equal to t_2 are classified as belonging to C_2 and observations greater than t_2 are classified as belonging to C_3. For particular values of t_1 and t_2, we denote the number of correctly classified future observations from class C_1, C_2 and C_3 by $L^1_{t_1}$, $L^2_{(t_1,t_2)}$ and $L^3_{t_2}$, respectively. Let denote the target proportions chosen to reflect the desired importance of the three classes by a, b and c, respectively. Selecting these values will depend on a person's beliefs of which class is more important to be correctly classified than others. There is no constraint on these values except to be in $(0, 1]$. One can choose a, b and c to be equal if one gives the same importance of correct classification to all three classes. In Section 4.2, we present a strategy to optimise these values through some automated algorithm. The general event of interest which we consider for the three classes C_1, C_2 and C_3 is that the number of correctly classified future observations from class C_1 is at least am_1, the number of correctly classified future observations from class C_2 is at least bm_2, and the number of correctly classified future observations from class C_3 is at least cm_3, that is $L^1_{t_1} \geq am_1$, $L^2_{(t_1,t_2)} \geq bm_2$ and $L^3_{t_2} \geq cm_3$.

Using the assumption of independence between the three ordered classes, the NPI lower probability for the event of interest is

$$\begin{aligned} &\underline{P}(L^1_{t_1} \geq am_1, L^2_{(t_1,t_2)} \geq bm_2, L^3_{t_2} \geq cm_3) = \\ &\underline{P}(L^1_{t_1} \geq am_1) \times \underline{P}(L^2_{(t_1,t_2)} \geq bm_2) \times \underline{P}(L^3_{t_2} \geq cm_3) \end{aligned} \tag{23}$$

and the corresponding NPI upper probability is

$$\begin{aligned} &\overline{P}(L^1_{t_1} \geq am_1, L^2_{(t_1,t_2)} \geq bm_2, L^3_{t_2} \geq cm_3) = \\ &\overline{P}(L^1_{t_1} \geq am_1) \times \overline{P}(L^2_{(t_1,t_2)} \geq bm_2) \times \overline{P}(L^3_{t_2} \geq cm_3) \end{aligned} \tag{24}$$

For $I^1_i = (x^1_{i-1}, x^1_i), i = 1, \ldots, n_1+1$, and $t_1 \in I^1_{i_{t_1}} = (x^1_{i_{t_1}-1}, x^1_{i_{t_1}})$, where $i_{t_1} \in \{2, 3, \ldots, n_1\}$ is defined as that interval $I^1_{i_{t_1}}$ which contains t_1, the NPI lower and upper probabilities for the event $L^1_{t_1} \geq am_1$ are [6, 19]

$$\underline{P}(L^1_{t_1} \geq am_1) = \underline{P}(X^1_{\lceil am_1 \rceil} \leq t_1) = \sum_{i=1}^{i_{t_1}-1} P(X^1_{\lceil am_1 \rceil} \in I^1_i) \tag{25}$$

$$\overline{P}(L^1_{t_1} \geq am_1) = \overline{P}(X^1_{\lceil am_1 \rceil} \leq t_1) = \sum_{i=1}^{i_{t_1}} P(X^1_{\lceil am_1 \rceil} \in I^1_i) \tag{26}$$

For $i_{t_1} = 1$, these NPI lower and upper probabilities are $\underline{P}(L^1_{t_1} \geq am_1) = 0$ and $\overline{P}(L^1_{t_1} \geq am_1) = P(X^1_{\lceil am_1 \rceil} \in I^1_1)$, and for $i_{t_1} = n_1 + 1$, they are $\underline{P}(L^1_{t_1} \geq am_1) = 1 - P(X^1_{\lceil am_1 \rceil} \in I^1_{n_1+1})$ and $\overline{P}(L^1_{t_1} \geq am_1) = 1$.

For $I^2_j = (x^2_{j-1}, x^2_j)$ with $j = 1, \ldots, n_2 + 1$ and $t_1 \in I^2_{j_{t_1}} = (x^2_{j_{t_1}-1}, x^2_{j_{t_1}})$, and $t_2 \in I^2_{j_{t_2}} = (x^2_{j_{t_2}-1}, x^2_{j_{t_2}})$, with $j_{t_1} \in \{1, 2, \ldots, n_1+1\}$ and $j_{t_2} \in \{1, 2, \ldots, n_2+1\}$, with $t_2 \geq t_1$ so $j_{t_2} \geq j_{t_1}$, the NPI lower and upper probabilities for the event $L^2_{(t_1,t_2)} \geq bm_2$ are [6, 19]

$$\underline{P}(L^2_{(t_1,t_2)} \geq bm_2) = P(L^2_{(x^2_{j_{t_1}}, x^2_{j_{t_2}-1})} \geq bm_2) \tag{27}$$

$$\overline{P}(L^2_{(t_1,t_2)} \geq bm_2) = P(L^2_{(x^2_{j_{t_1}-1}, x^2_{j_{t_2}})} \geq bm_2) \tag{28}$$

For $j_{t_1} = 1$ and $j_{t_2} = 2$, these NPI lower and upper probabiliites are $\underline{P}(L^2_{(t_1,t_2)} \geq bm_2) = 0$ and $\overline{P}(L^2_{(t_1,t_2)} \geq bm_2) = P(L^2_{(-\infty, x^2_{j_{t_2}})} \geq bm_2)$.

For $I^3_l = (x^3_{l-1}, x^3_l), l = 1, \ldots, n_3{+}1$, and $t_2 \in I^3_{l_{t_2}} = (x^3_{l_{t_2}-1}, x^3_{l_{t_2}})$ where $l_{t_2} \in \{1, 2, \ldots, n_3\}$, the NPI lower and upper probabilities for the event $L^3_{t_2} \geq cm_3$ are [6, 19]

$$\underline{P}(L^3_{t_2} \geq cm_3) = \underline{P}(X^3_{(m_3-\lceil cm_3 \rceil+1)} > t_2) = \sum_{l=l_{t_2}+1}^{n_3+1} P(X^3_{(m_3-\lceil cm_3 \rceil+1)} \in I^3_l) \tag{29}$$

$$\overline{P}(L^3_{t_2} \geq cm_3) = \overline{P}(X^3_{(m_3-\lceil cm_3 \rceil+1)} > t_2) = \sum_{l=l_{t_2}}^{n_3+1} P(X^3_{(m_3-\lceil cm_3 \rceil+1)} \in I^3_l) \tag{30}$$

where $\lceil cm_3 \rceil$ is the smallest integer greater than or equal cm_3. For $l_{t_2} = 1$, these NPI lower and upper probabilites are $\underline{P}(L^3_{t_2} \geq cm_3) = 1 - P(X^3_{(m_3-\lceil cm_3 \rceil+1)} \in I^3_1)$ and $\overline{P}(L^3_{t_2} \geq cm_3) = 1$, and for $l_{t_2} = n_3{+}1$, they are $\underline{P}\left(L^3_{t_2} \geq cm_3\right) = 0$ and $\overline{P}\left(L^3_{t_2} \geq cm_3\right) = P\left(X^3_{(m_3-\lceil cm_3 \rceil+1)} \in I^3_{n_3+1}\right)$.

We obtain the two optimal thresholds, t_1 and t_2, for the three ordered classes C_1, C_2 and C_3 by maximising either the NPI lower probability, Equation (23), or the NPI upper probability, Equation (24). One needs to search for the values t_1 and t_2 that maximise the lower or the upper probability within each of the $n_1 + n_2 + n_3 + 1$ intervals created by the data observations. In the following, we provide an example to illustrate the NPI method for selecting the optimal thresholds for three classes, as presented above. For further explanations, examples and discussions of this method, we refer to [6, 19].

4.1.1. Example

Assume that we have a real-valued data set from three ordered classes C_1, C_2 and C_3, with 22 observations, where $n_1 = 8$ and $n_2 = n_3 = 7$, consisting of the data {0.58, 0.59, 0.69, 0.80, 0.83, 1.22, 1.25, 2.29} for C_1, {0.92, 1.46, 2.11, 2.41, 2.43, 2.65, 3.03} for C_2 and {2.14,2.45, 2.52, 2.63, 3.04, 3.15, 3.32} for C_3. Table 3 presents the results of the optimal threshold values t_1 and t_2 obtained from the NPI method along with their corresponding lower and upper probabilities, for $m_1 = 8, m_2 = 7$ and $m_3 = 6$. To illustrate the NPI method, we have considered four different scenarios for the target proportions a, b and c. For the first scenario, we have chosen small values for the target proportions, $a = b = c = 0.30$, leading to optimal threshold values $t_1 = 0.83$ and $t_2 = 2.43$. As the required target proportions are quite easy to achieve, the corresponding NPI lower and upper probabilities are high. The second scenario has $a = b = c = 0.7$, which are quite high, so the NPI method attempts to make a balance between the three classes to meet the required target proportions and to find the optimal thresholds that maximise the NPI lower probability, given in Equation (23), and the NPI upper probability, given in Equation (24). The corresponding NPI lower and upper probabilities are smaller than for the first scenario, which is due to the fact that the required target proportions are larger. The third scenario has $a = b = 0.70, c = 0.20$, so

Scenario	Target proportions			NPI lower method			NPI upper method		
#	a	b	c	t_1	t_2	value	t_1	t_2	value
1	0.30	0.30	0.30	0.83	2.43	0.53	0.83	2.43	0.84
2	0.70	0.70	0.70	1.25	2.43	0.07	1.25	2.43	0.38
3	0.70	0.70	0.20	1.25	3.03	0.24	1.25	3.03	0.65
4	0.50	0.75	0.50	0.83	2.43	0.21	0.83	2.43	0.59

Table 3: Optimal thresholds (t_1, t_2) using NPI-based method, and corresponding values of the NPI lower and upper probabilities, for $m_1 = 8, m_2 = 7$ and $m_3 = 6$

Scenario	Target proportions			NPI lower mathod			NPI upper method		
#	a	b	c	t_1	t_2	value	t_1	t_2	value
1	0.30	0.30	0.30	0.83	2.43	0.89	0.83	2.43	0.98
2	0.70	0.70	0.70	0.83	2.43	0.06	0.83	2.43	0.41
3	0.70	0.70	0.20	1.25	3.03	0.31	1.25	3.03	0.79
4	0.50	0.75	0.50	0.83	3.03	0.65	0.83	3.03	0.93

Table 4: Optimal thresholds (t_1, t_2) using NPI-based method, and corresponding values of the NPI lower and upper probabilities, for $m_1 = m_2 = m_3 = 10$

there is an emphasis on the number of correctly classified future observations from class C_1 and class C_2 than from class C_3. This leads to t_2 being larger than in the other scenarios. The final scenario has $a = 0.50, b = 0.75, c = 0.50$, so it puts more emphasis on the number of correctly classified future observations from class C_2 than the numbers of correctly classified observations from classes C_1 and C_3. It is noticed that the optimal threshold values and the NPI lower and upper probabilities changed again in order to meet the target proportions.

To further illustrate the results of the NPI method for selecting the optimal threshold values, we present in Table 4 the optimal threshold values along with the corresponding NPI lower and upper probabilities for $m_1 = m_2 = m_3 = 10$, using the same target proportions presented in Table 3. The results are similar to those with $m_1 = 8, m_2 = 7$ and $m_3 = 6$ in Table 3. For $a = b = c = 0.30$ and $a = b = 0.70, c = 0.20$, the optimal thresholds are the same in both tables, but the corresponding NPI lower and upper probabilities are greater than those in Table 3. It should be clarified that the change in the NPI lower and upper probabilities here is due to the discrete nature of the event we consider, but it is not a direct effect of larger m, as larger m still needs the same proportion to be achieved. For example, for $m_1 = 8$, $m_1 = 10$ and $a = 0.3$, we need at least 3 good classifications in both cases, which is easier for $m_1 = 10$ than for $m_1 = 8$, so then for the latter the NPI lower and upper probabilities are probably smaller. For $a = b = c = 0.70$, the NPI method provides the same optimal threshold for t_2, but the optimal threshold for t_1 is different, while the values of the NPI lower and upper probabilities for both tables are quite similar. Finally, for $a = 0.50, b = 0.75$ and $c = 0.50$, the optimal threshold t_1 is the same in both tables, but the optimal threshold t_2 is different, and the corresponding NPI lower and upper probabilities in Table 4 are high compared with those for the same scenario in Table 3.

4.2. Selecting the target proportions for three ordered classes

Following the proposed method of choosing the values of a and b, presented in Section 3.2, we also focus here on choosing the values of a, b and c based on the data set used in

classification tasks, not setting them in advance. We will choose the values that improve classification performance. Consider the NPI method for selecting the optimal thresholds, t_1 and t_2, which is based on the NPI lower probability, Equation (23),

$$\begin{aligned} &\underline{P}(L^1_{t_1} \geq am_1, L^2_{(t_1,t_2)} \geq bm_2, L^3_{t_2} \geq cm_3) = \\ &\underline{P}(L^1_{t_1} \geq am_1) \times \underline{P}(L^2_{(t_1,t_2)} \geq bm_2) \times \underline{P}(L^3_{t_2} \geq cm_3) \end{aligned} \tag{31}$$

where $P(L^1_{t_1} \geq am_1)$, $P(L^2_{(t_1,t_2)} \geq bm_2)$ and $P(L^3_{t_2} \geq cm_3)$ are given by Equations (25), (27) and (29), respectively, and a, b, c are any values in $(0, 1]$. Note that we now consider a, b and c as parameters instead of achieved target proportions. As these parameters play an important role in the total classification accuracy, we propose to choose these parameters that maximise the classification accuracy on the testing sets. This raises the question of how one can choose the target proportions that maximise the classification accuracy and how to validate their performance in classification trees. Similar to the previous method, presented in Section 3.2, we suggest using a double cross-validation procedure in order to train our classification algorithm and tune the parameters a, b and c.

The process of choosing the values of a, b, and c using the two levels of the k-fold cross-validation procedure, where $k = 5$, is presented in Figure 2. As we see from the figure, the data set is divided into two levels of the 5-fold cross-validation procedure. In the first level, which is the outer level, we validate our method with optimal parameters of a, b and c, while in the second level, which is the inner fold, we use a search function, the Genetic Algorithm (GA) [25, 26], to tune the parameters a, b and c. In other words, the inner folds will return only the algorithm with the best values for a and b to the outer folds, while the outer folds will to validate the algorithm's quality. During the outer folds, we obtain five different performances with varying values of the parameters a, b, and c. However, since we aim to find the best values of a, b, and c for a given data set, we extract the optimal parameters from the outer folds. These optimal values are then used as target proportions for the data set, as in D. Further details on how the GA method works to fine-tune the values of a, b, and c in the inner level are provided in [8].

4.3. NPI$_3$-Tree algorithm

In this section, we present a new classification algorithm for building classification trees with data containing three ordered classes, which we call Nonparametric Predictive Inference for building classification trees with three ordered classes (NPI$_3$-Tree) algorithm. It is similar to the NPI$_2$-Tree algorithm, presented in Section 3.3, for building classification trees with data containing binary classes, but the main difference is that the NPI$_3$-Tree algorithm works with data that contains three classes. In the NPI$_3$-Tree algorithm, we use the NPI method for selecting the optimal thresholds, t_1 and t_2, presented in Section 4.1, to find the optimal threshold values, and the method of choosing the values of a, b and c, presented in Section 4.2. The procedure of building classification trees by the NPI$_3$-Tree algorithm is similar to the well-known C4.5 algorithm, but we use the NPI-based thresholds method, presented in Section 4.1, and the method of choosing the values of a, b and c, presented in Section 4.2. The procedure for building the NPI$_3$-Tree classification tree is described in Algorithm 2 (given in Appendix B).

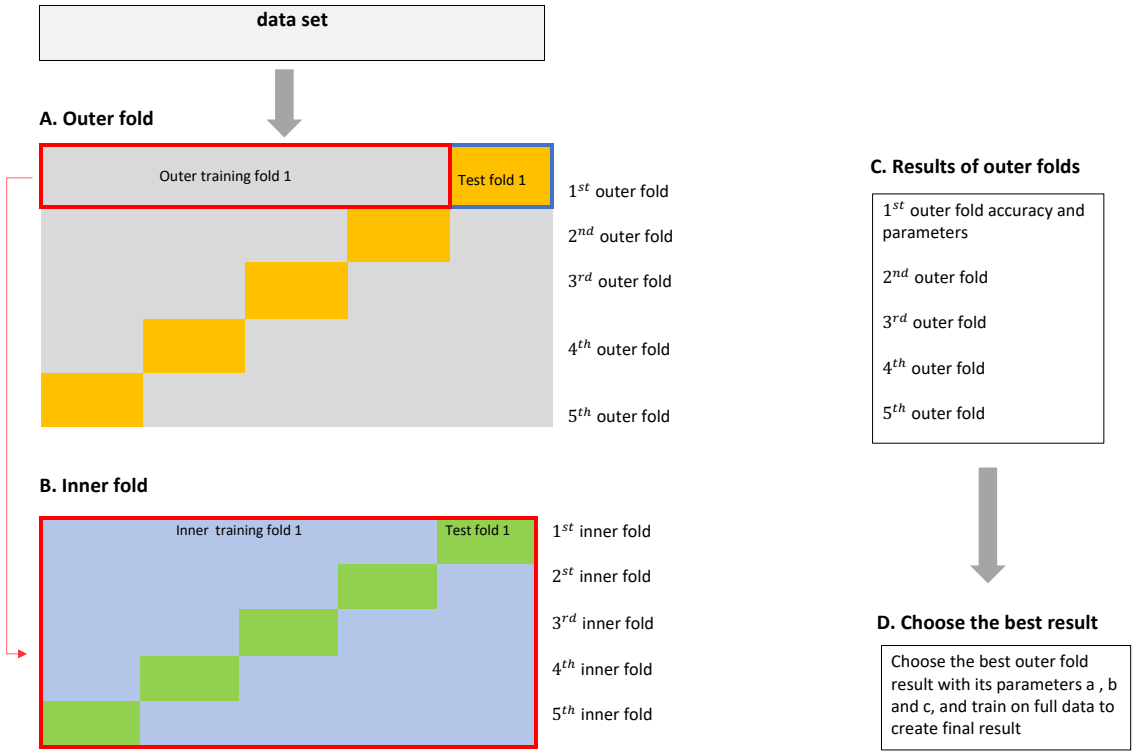


Figure 2: A diagram of two-levels 5-fold cross-validation procedure to find the optimal values of the target proportions a, b and c.

Data set	N	Attr	Pro of class 1	Pro of class 2
Breast Cancer	116	9	0.45	0.55
Blood Transfusion	748	3	0.76	0.24
Liver Patients	583	9	0.28	0.72
Haberman's Survival	306	3	0.73	0.27
Cryotherapy	90	5	0.53	0.47
QSAR Biodeg	1055	14	0.66	0.34

Table 5: A brief description of the data sets used with the NPI_2-Tree algorithm.

5. Experimental Analysis

In this section, we carry out an experimental analysis to examine the performance of the NPI_2-Tree and the NPI_3-Tree algorithm, and to compare their performance with other classification algorithms on several data sets obtained from the UCI Machine Learning Repository [21]. We compare their performance with the most commonly used classical algorithm, the C4.5 algorithm and the CART algorithm, and with some classification algorithms based on imprecise probabilities, namely the NPI-M and the IDM1 algorithms. More information about these algorithms has been given in Section 2.1.1. Note that as the IDM algorithm depends on the value of the parameter $\tilde{s}$, we use one recommended value of $\tilde{s}$, which is $\tilde{s} = 1$. We refer to the IDM with $\tilde{s} = 1$, by the IDM1 algorithm.

5.1. Experimental setup

We have used six data sets for the NPI_2-Tree algorithm and five data sets for the NPI_3-Tree algorithm. We have only used these data sets because the NPI_2-Tree and the NPI_3-Tree algorithms are suitable for continuous attribute variables with a two or three-class variable, and such data sets are uncommon in public databases. However, it would be interesting to fully automate the NPI_2-Tree algorithm to enable us to analyse more data sets, including categorical attributes. A brief description of the main properties of data sets used for the NPI_2-Tree and the NPI_2-Tree algorithm is given in Tables 5 and 6, respectively. Column 'N' gives the number of observations in the data set, column 'Attr' gives the number of attribute variables, column 'Pro of class 1' gives the data proportion in class 1, column 'Pro of class 2' gives the data proportion in class 2 and column 'Pro of class 3' gives the data proportion in class 3. Further details about these data sets can be found in [21]. It is important to note that, as we only work with continuous-valued attributes, the categorical attributes in the data sets were ignored. So, the number of attributes in the 'Attr' column is only the number of continuous-valued attributes. Therefore, we may not be surprised if the results found in the literature are different from our results. For example, the Liver Patients data set, given in Table 5, has ten attribute variables; nine are continuous, and one is categorical. The classification accuracy obtained in our experiment using the C4.5 algorithm is 77.25%, but in [36], the classification accuracy result using the C4.5 algorithm, is 84.86%, where the categorical attribute, which is the gender of the patient, was used.

As a first step, we use the NPI_2-Tree and the NPI_3-Tree algorithms to build a classification tree for each data set. This was done using the tree-building process presented in Section 3.3 for the NPI_2-Tree algorithm and in Section 4.3 for the NPI_3-Tree algorithm, with the

Data set	N	Attr	Pro of class 1	Pro of class 2	Pro of class 3
Iris	150	4	0.33	0.33	0.33
Seeds	210	7	0.33	0.33	0.33
Wine	178	3	0.34	0.39	0.27
CMC	1473	2	0.42	0.24	0.34
Fitness	8020	10	0.36	0.41	0.23

Table 6: A brief description of the data sets used with the NPI_3-Tree algorithm.

proposed method of choosing the values of a and b and the values of a, b and c as presented in Sections 3.2 and 4.2, respectively. We compare the performance of our classification algorithms with the most commonly used classical classification trees, which are the C4.5 and the CART algorithms, and with two imprecise algorithms, which are the NPI-M and the IDM algorithms. More details about these algorithms have been presented in Section 2.1.1.

The R software [35] has been used to carry out this experiment. We used the `RWeka` package [29, 41] to implement the C4.5 algorithm, the `rpart` package [40] to implement the CART algorithm and the `imptree` package [24] to implement both the NPI-M and IDM1 algorithms. Some pre-processing steps in our data sets have been carried out. The missing values for continuous attributes were replaced with mean values using the missing value filter in R. In addition, some of the data sets contain tied observations; we dealt with these by adding a small amount to the tied observations. We also tested our method without breaking the tied observations and observed that the results were close. Furthermore, for the NPI-M and the IDM1 algorithms, as they can only handle categorical attributes, therefore, we discretised the continuous variables presented in Tables 5 and 6 using the `mdlp` package in R and the 'discretisation' function. This function converts a continuous variable into a categorical variable using the Fayyad and Irani method [22, 23], which finds a threshold value using only the average class entropy to evaluate the partitions created by each candidate threshold. These pre-processing steps are essential, and they were carried out at the beginning of the analysis for all algorithms to ensure a fair comparison for all algorithms. After that, we applied all classification algorithms to all data sets and the results were compared in several ways. The following measures were used to evaluate the performance of the classification algorithms: classification accuracy, in-sample accuracy and tree size. The classification accuracy is the ratio of the number of correctly classified observations to the total number of observations in the test data set, while the in-sample accuracy is the classification accuracy for the training set. The tree size is the number of leaf nodes in the tree [12]. All results given in this experiment were obtained using the average of a 10-fold cross-validation scheme, as given in Section 3.2. With respect to the NPI_2-Tree and the NPI_3-Tree algorithms, we first found the optimal values of target proportions using the proposed method, and then we used the 10-fold cross-validation scheme to report the final result. Below, we provide the outcomes of the experiment concerning our algorithms, presented separately.

5.2. Performance of the NPI_2-Tree algorithm

Table 7 presents the results of classification accuracies of the NPI_2-Tree algorithm and other algorithms for each data set, including the optimal values of a and b that correspond

Data set	a	b	NPI_2-Tree	C4.5	CART	NPI-M	IDM1
Breast Cancer	0.57	0.73	87.10	86.89	86.90	87.65	**87.86**
Blood Transfusion	0.79	0.64	**89.48**	75.43	74.76	79.56	79.56
Liver Patients	0.32	0.65	**80.70**	77.25	76.43	80.28	80.28
Haberman's Survival	0.84	0.42	75.19	75.62	73.18	**76.39**	**76.39**
Cryotherapy	0.56	0.50	79.38	80.11	**83.48**	81.45	80.18
QSAR Biodeg	0.65	0.82	**91.22**	73.13	77.86	82.16	82.16
Average	-	-	**83.84**	78.07	78.72	81.24	81.07

Table 7: Average result of classification accuracy of different classification algorithms and the optimal values of a and b for the NPI_2-Tree algorithm.

Data set	a	b	NPI_2-Tree	C4.5	CART	NPI-M	IDM1
Breast Cancer	0.57	0.73	88.12	**88.22**	**88.22**	**88.22**	**88.22**
Blood Transfusion	0.79	0.64	82.78	**84.58**	**84.58**	84.40	84.40
Liver Patients	0.32	0.65	89.80	**83.93**	83.20	81.96	81.62
Haberman's Survival	0.84	0.42	75.40	**76.99**	**76.99**	76.23	76.23
Cryotherapy	0.56	0.50	**82.87**	82.64	80.34	82.16	82.28
QSAR Biodeg	0.65	0.82	87.39	87.50	**88.86**	85.96	85.96
Average	-	-	82.72	**83.97**	83.71	83.16	83.11

Table 8: Average result of the in-sample accuracy of the different classification algorithms and the optimal values of a and b for the NPI_2-Tree algorithm.

to the NPI_2-Tree algorithm. Table 8 shows the results of in-sample accuracy for all classification algorithms. The tree sizes of classification algorithms are presented in Table 9. In all tables, the best results are presented in bold font.

As shown in Table 7, the classification accuracies indicate that the NPI_2-Tree algorithm outperforms other classification algorithms for most data sets, and it has the highest average classification accuracy, followed by the NPI-M, the IDM1, the CART and the C4.5. For the Breast Cancer data set, all classification algorithms obtained similar results, with the NPI-M algorithm performing slightly better than the other algorithms. For the Blood Transfusion and QSAR Biodeg data sets, there is a noticeable difference in classification accuracies between the classification algorithms, with the NPI_2-Tree algorithm clearly outperforming the other classification algorithms. We have investigated the characteristics of these data sets in order to gain insight into reasons that may cause this clear difference in the performance of the NPI_2-Tree algorithm, and we found that these data sets are large compared to other data sets and have less overlap between their data classes, which could be a reason why the NPI_2-Tree algorithm is superior to the other algorithms on these data sets. The worst-performing algorithms are the CART and C4.5 algorithms. The NPI-M and IDM1 algorithms obtained very similar classification accuracies. For the Liver Patients data set, we can see that the NPI_2-Tree, the NPI-M and the IDM1 algorithms perform better than the classical algorithms. For Haberman's Survival data set, the NPI-M and the IDM1 perform slightly better than the other algorithms, obtaining the same classification accuracy of 76.39%. Finally, for the Cryotherapy data set, the NPI_2-Tree algorithm obtained the worst result, which is 79.38%. For this data set, the NPI_2-Tree algorithm built smaller trees than the ones built by the other algorithms, which might be the reason why it performs a little less than the other algorithms.

Algorithm	NPI_2-Tree	C4.5	CART	NPI-M	IDM1
Average	**4.20**	4.72	4.46	4.33	4.39

Table 9: Average result of the tree size of the different classification algorithms.

Table 8 presents the results of the in-sample accuracy of all classification algorithms. The in-sample accuracy, which is the classification accuracy on the training set, is not widely used to evaluate classification algorithms, but it gives insight into how the classification algorithm works on the training set. It is well known that if the classification algorithm works very well on the training set but does not work very well on the testing set, it likely indicates overfitting. Therefore, it is useful to show the classification algorithms' performance on both training and testing sets and, hence, to check the overfitting as well, as was done in [12, 32]. As we can see from Table 8, the C4.5 algorithm performs slightly better than the other classification algorithms, followed by the CART, the NPI-M, the IDM1 and the NPI_2-Tree. Note that the C4.5, the CART, the NPI-M and the IDM1 algorithms have better in-sample accuracy than classification accuracy, but the performance of the NPI_2-Tree algorithm on the training sets is slightly worse than its performance on the testing sets. It is possible that these results are due to the fact that the NPI_2-Tree algorithm selects the optimal thresholds by focusing on prediction, unlike the other algorithms, which focus on maximising the correct classification on the training sets. We see from the results that the C4.5 and the CART algorithms clearly perform better on the training sets than on the testing sets, which indicates that these algorithms may be suffering from overfitting. According to the average results of classification accuracy and in-sample accuracy, we can state that the NPI_2-Tree algorithm works well on both data sets, training and testing sets, which might indicate that the NPI_2-Tree algorithm does not suffer from overfitting.

Finally, Table 9 shows the average results of the tree sizes for all the classification algorithms. Note that it is referred to the tree size as the total number of leaf nodes in the tree, as was done by Bertsimas and Dunn [12], and by Murthy and Salzberg [32]. However, some researchers may refer to the tree size as the total number of all nodes in the tree. Of course, one can use any method to refer to the size of the tree. Table 9 shows that the NPI_2-Tree algorithm creates slightly smaller trees than the other algorithms, followed by the NPI-M algorithm. The C4.5 algorithm creates the largest trees with an average tree size of 4.72, which could be the reason for its good performance on the training sets. The CART algorithm mostly creates similar trees to the C4.5 algorithm, but in Liver Patients and Cryotherapy data sets, the CART algorithm has the smallest tree size compared to the C4.5 algorithm, and hence, it gives a smaller average tree size than the C4.5 algorithm. This study shows that the NPI_2-Tree algorithm tends to create smaller trees than the C4.5, the CART, the NPI-M and the IDM1 algorithms.

5.3. Performance of the NPI_3-Tree algorithm

Tables 10 present the results of the classification accuracies of the NPI_3-Tree algorithm and all the other classification algorithms for each data set. It also presents the target proportions a, b and c that correspond to the NPI_3-Tree algorithms. As shown in Table

Data set	a	b	c	NPI$_3$-Tree	C4.5	CART	NPI-M	IDM1
Iris	0.77	0.75	0.80	94.61	94.22	94.38	**94.69**	94.52
Seeds	0.81	0.79	0.78	**93.43**	89.72	90.42	92.63	92.38
Wine	0.94	0.68	0.87	**96.54**	93.12	91.14	95.19	94.64
CMC	0.43	0.36	0.52	49.96	**50.10**	48.40	49.81	49.81
Fitness	0.53	0.64	0.49	**79.81**	77.31	72.19	77.60	77.82
Average	-	-	-	**82.87**	80.89	79.30	81.98	81.83

Table 10: Average result of the classification accuracy of all algorithms and the optimal values of a, b and c for the NPI$_3$-Tree algorithm.

10, the NPI$_3$-Tree algorithm performs better than the other algorithms for most data sets, and it achieves the highest average classification accuracy, followed by NPI-M, IDM1, C4.5 and CART, respectively. For the Iris data set, all classification algorithms, including the NPI$_3$-Tree, have similar results. This similarity between all algorithms' performance could be because the Iris data set has 150 observations distributed equally across the three classes, and there is less overlap between their data classes. For this data set, the values of a, b and c are also similar. For the Seeds and Wine data sets, the NPI$_3$-Tree algorithm clearly outperforms the classical algorithms, the C4.5 and the CART algorithms, and slightly performs better than the NPI-M and the IDM1 algorithms. For these data sets, the NPI$_3$-Tree classification algorithm built larger trees than the other classification algorithms, which might be the reason why the NPI$_3$-Tree algorithm performs better than the other algorithms. For the CMC data set, all the classification algorithms, including the NPI$_3$-Tree algorithm, have not achieved good results, with the C4.5 algorithm slightly performing better than the other algorithms, which obtained classification accuracy of 50.10%. For this data set, the values of a, b and c are also low. We have analysed this data in-depth in order to give an insight into the characteristics of this data set that could be causing these results. The CMC data set has 1473 observations and two continuous attribute variables. However, this data set has overlaps between their data observations' classes of more than half of the data. This could be the reason for these results, which also match those observed in earlier studies, e.g. [4, 31]. Finally, for the Fitness data set, the NPI$_3$-Tree algorithm obtained the highest classification accuracy of 79.81%, whereas the C4.5, the NPI-M and the IDM1 algorithms have similar results. On the other hand, the CART obtained the worst result. Overall, according to the average classification accuracy, we can say that the NPI$_3$-Tree algorithm tends to perform better than the C4.5 and the CART algorithms, and it tends to perform slightly better than the NPI-M and the IDM1 algorithms.

Following [32, 12], we have evaluated the performance of the NPI$_3$-Tree algorithm on the training data set and compared it with the other classification algorithms. The in-sample accuracy measure, which is the performance of algorithms on the training set, is not commonly used to indicate classification accuracy, but it would be useful to give insight into how the classification algorithm works on the training set. It is well known that if the classification algorithm works very well on the training set but does not work very well on the testing set, it likely indicates overfitting. Therefore, it is useful to show the classification algorithms' performance on both training and testing sets and, hence, to check the overfitting as well. Table 11 presents the results of the in-sample accuracy of the NPI$_3$-Tree algorithm and the

Data set	a	b	c	NPI$_3$-Tree	C4.5	CART	NPI-M	IDM1
Iris	0.77	0.75	0.80	**94.82**	**94.82**	**94.82**	**94.82**	**94.82**
Seeds	0.81	0.79	0.78	90.66	**92.53**	92.11	91.16	91.98
Wine	0.94	0.68	0.87	93.54	94.56	**94.93**	92.40	92.40
CMC	0.43	0.36	0.52	51.68	52.39	51.68	**52.78**	**52.78**
Fitness	0.53	0.64	0.49	79.96	**80.77**	79.61	80.22	80.28
Average	-	-	-	82.13	**83.04**	82.63	82.27	82.45

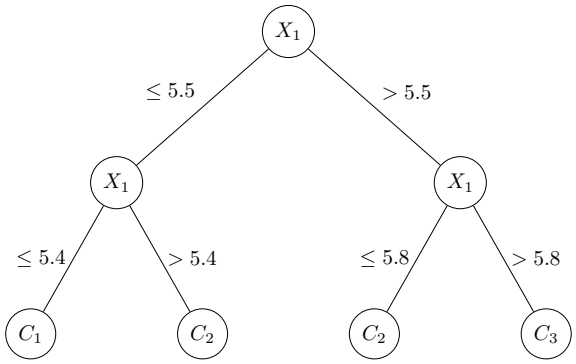
Table 11: Average result of the in-sample accuracy for all classificaton algorithms and the optimal values for a, b and c to the NPI$_3$-Tree algorithm.

Algorithm	NPI$_3$-Tree	C4.5	CART	NPI-M	IDM1
Average	**3.66**	5.80	4.48	5.92	5.94

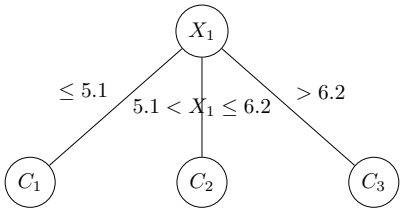
Table 12: Average result of the tree size of all classification algorithms.

other classification algorithms. All classification algorithms perform better on the training data sets than on the testing data sets, except the NPI$_3$-Tree algorithm, whose performance on the training sets is slightly less than its performance on the test sets. This is due to the fact that the NPI$_3$-Tree algorithm selects the optimal thresholds by focusing on prediction, unlike the other classification algorithms, which focus on maximising the correct classification of the training sets. The C4.5 algorithm has the highest average result of in-sample accuracy, followed by the CART, the IDM1, the NPI-M and the NPI$_3$-Tree algorithms. For the Iris data set, although all classification algorithms do not give the same trees, they have obtained the same result, which is 94.82%. In this experimental analysis, it is noticed that the classical algorithm, the C4.5 and the CART, clearly perform better on the training data sets compared to their performance on the testing sets, which could be an indication they suffer from overfitting. From the results presented in Table 11, we can explain that the NPI$_3$-Tree has good results on both the training and testing data sets, which may indicate that the NPI$_3$-Tree algorithm does not overfit the data sets.

Finally, the average tree size for all the classification algorithms is given in Table 12. As we can see from the table, the NPI$_3$-Tree algorithm creates the smallest trees, which has obtained the smallest result of the average tree size, followed by the CART, the C4.5, the NPI-M and the IDM1 algorithms. The use of a multiple split on continuous attributes, which is used by the NPI$_3$-Tree algorithm, could be one of the main reasons for creating the smallest trees by the NPI$_3$-Tree algorithm. The use of a multiple split on an attribute variable usually makes a classification tree smaller, easier to understand and faster to build. Unlike the binary split, which makes the trees larger because it allows an attribute variable to appear many times in the paths from the root of the tree to its leaf. For more clarification, we illustrate in Figure 3 two classification trees created by the C4.5 and the NPI$_3$-Tree algorithms for one attribute variable, X_1, of the Iris data. Figure 3(a) presents the classification tree created by the C4.5, which uses a binary split, e.g. two branches, while Figure 3(b) presents the classification tree created by the NPI$_3$-Tree algorithm, which uses a multiple split, e.g. three branches. As we can see from the two trees, the classification tree built by the NPI$_3$-Tree algorithm is smaller than the classification tree built by the C4.5 algorithm for the same data set.



(a) Tree with a binary split



(b) Tree with a multiple split

Figure 3: Classification trees with binary and multiple split.

6. Conclusions and related future research

In this paper, we introduced a new method to build classification trees based on Nonparametric Predictive Inference (NPI). We built classification trees using the NPI approach for selecting optimal thresholds for data sets that involve continuous-valued attributes. This approach selects the optimal threshold values using predictive inference that considers specific numbers of future observations and the target proportions. As a first step, we presented our method for binary classification trees, where the attribute variables are continuous, and the class variable is binary. A new classification algorithm, which we called the NPI_2-Tree algorithm, was presented for building binary classification trees. We then extended our method to classification trees with three classes, where the attribute variables are continuous, and a class variable has three ordered classes. A new classification algorithm for building classification trees with three classes was presented, which we called the NPI_3-Tree algorithm. We introduced a new procedure for selecting the target proportions by choosing to maximise classification performance on the testing datasets. We carried out an experimental analysis on several data sets to evaluate the performance of the NPI_2-Tree and NPI_3-Tree classification algorithms and compare their performance with other classification algorithms from the literature. Classification accuracy, in-sample accuracy, and tree size were used to measure the performance of all the classification algorithms. The results of the experimental analysis have indicated that the NPI_2-Tree and NPI_3-Tree classification methods perform well and better than the other classification algorithms.

The work presented in this paper provides many possible topics for future research. As a first step in building classification trees based on our methods, we started with two classes; we then extended our method to build classification trees with three classes. Now, it would be of interest to extend this method further to involve more than three classes. This can be achieved by first developing the NPI method for selecting the optimal threshold to include more than three classes. In this work, we have restricted attention to using only continuous-valued attributes; it is of interest to investigate the performance of the NPI_2-Tree and the NPI_3-Tree algorithms on data sets that include categorical attributes. Finally, one important topic for future work is to develop the NPI_2-Tree and the NPI_3-Tree algorithms with consideration for the misclassification cost. In many practical applications, classification aims to minimise misclassification costs instead of maximising the total classification accuracy. In this paper, we have chosen the values of target proportions that maximise the total classification accuracy. However, it would be useful to develop the process of choosing these target proportions while also considering the misclassification cost.

Acknowledgements

Masad Alrasheedi would like to thank Taibah University and the Saudi government for their invaluable support, which has made it possible for me to complete my PhD studies.

Appendix A. NPI$_2$-Tree algorithm

Algorithm 1 Pseudocode NPI$_2$-Tree algorithm

1. Input:($\mathcal{D}$, C, Ω)
   $\mathcal{D}$: Data set
   C: Binary class variable $C = \{C_1, C_2\}$
   Ω: Set of continuous attributes $\Omega = \{X_1, \ldots, X_f\}$
2. Procedure NPI$_2$-Tree($\mathcal{D}$, C, Ω)
3. Create a Root node for the tree
4. If all observations in $\mathcal{D}$ have the same class C, then
5. Return the single-node tree with class C
6. If Ω is empty (i.e. there are no attributes available), then
7. Return the single-node tree with most common class C in $\mathcal{D}$
8. Otherwise
9. The data set $\mathcal{D}$ is divided into two subsets:
   S: training set
   T: testing set
10. Select the values of a, b and m_i for $i = 1, 2$
    Make the initial values of a and b equal to the class proportion in S,
    i.e. make $a = \frac{n_1}{n}$ and $b = \frac{n_2}{n}$
    Make the values of m_i equal to the number of observations in class C_i in T
11. For each attribute, X_i in Ω do
    Find the optimal threshold values that maximise the NPI lower probability, given in Equation (12)
    Compute the IGR value, given in Equation (20)
12. Choose attribute X from Ω, with the highest IGR value
13. Assign the attribute X for the Root node
14. Add a branch below the Root node, corresponding to $X \leq t$ and $X > t$
15. Let S_i, for $i = 1, 2$ be the subset of S that has $X \leq t$ and $X > t$, respectively
16. If S_i, for $i = 1, 2$ is empty (one of them), then
17. Add a leaf node below the branch with the most common class in S
18. Check the stopping criteria mentioned above
19. Else
20. Add the subset created by NPI$_2$-Tree (S_i, C, $\Omega - \{X\}$)
21. Return Root

Appendix B. NPI$_3$-Tree algorithm

Algorithm 2 Pseudocode NPI$_3$-Tree algorithm

1. Input:($\mathcal{S}$, C, Ω)
2. $\mathcal{S}$: Training data set
3. C: A class variable $C = \{C_1, C_2, C_3\}$
4. Ω: Set of continuous attributes $\Omega = \{X_1, \ldots, X_f\}$
5. Procedure NPI$_3$-Tree($\mathcal{S}$, C, Ω)
6. Create a Root node for the tree
7. if all observations in $\mathcal{S}$ have the same class C, **then**
8. Return the single-node tree with class C
9. if Ω is empty (i.e. there are no attributes available), then
10. Return the single-node tree with most common class C in $\mathcal{S}$
11. Otherwise
12. Select the values of a, b, c and m_i for $i = 1, 2, 3$
13. Make the initial values of a, b and c equal to the class proportion in S,
14. i.e. make $a = \frac{n_1}{n}$, $b = \frac{n_2}{n}$ and $c = \frac{n_3}{n}$
15. Make the values of m_i equal to the number of observations in class C_i in S,
16. i.e. make $m_1 = n_1,\ m_2 = n_2$ and $m_3 = n_3$
17. for each attribute, X_i in Ω, **do**
18. Find the threshold values t_1 and t_2 that maximise the NPI lower probability, given in Equation (23)
19. Compute the IGR value using Equation (4)
20. Choose attribute variable X from Ω, with the highest IGR value
21. Assign the attribute X for the Root node
22. Add a branch below Root, corresponding to $X \leq t_1$, $t_1 < X \leq t_2$ and $X > t_2$,
23. Let S_i, for $i = 1, 2, 3$, be the subset of S that has $X \leq t_1$, $t_1 < X \leq t_2$ and $X > t_2$, respectively
24. if S_i is not empty, **then**
25. Add the subset created by NPI$_3$-Tree (S_i, C, $\Omega - \{X\}$)
26. return Root

References

[1] Abellán J. (2006). Uncertainty measures on probability intervals from the imprecise Dirichlet model. *International Journal of General Systems*, 35, 509–528.

[2] Abellán J. and Moral S. (2003). Building classification trees using the total uncertainty criterion. *International Journal of Intelligent Systems*, 18, 1215–1225.

[3] Abellán, J., Baker, R. and Coolen, F.P.A. (2011). Maximising entropy on the nonparametric predictive inference model for multinomial data. *European Journal of Operational Research*, 212, 112–122.

[4] Abellán, J., Baker, R., Coolen, F.P.A., Crossman, R. and Masegosa, R. (2014). Classification with decision trees from a nonparametric predictive inference perspective. *Computational Statistics and Data Analysis*, 71, 789–802.

[5] Abellán J. (2013). An application of Non-Parametric Predictive Inference on multiclass classification high-level-noise problems. *Expert Systems with Applications*, 40(11), 4585–4592.

[6] Alabdulhadi, M. (2018). *Nonparametric predictive inference for diagnostic test thresholds*. Ph.D. thesis, Durham University.

[7] Alqifari, H. (2017). *Nonparametric predictive inference for future order statistics*. Ph.D. thesis, Durham University.

[8] Alrasheedi, M. (2023). *Optimal Thresholds for Classification Trees using Nonparametric Predictive Inference*. Ph.D. thesis, Durham University.

[9] Augustin T. and Coolen F.P.A. (2004). Nonparametric predictive inference and interval probability. *Journal of Statistical Planning and Inference*, 124, 251–272.

[10] Baker, R. (2010). *Multinomial nonparametric predictive inference: selection, classification and subcategory data*. Ph.D. thesis, Durham University.

[11] Battineni, G., Sagaro, G., Nalini, C., Amenta, F. and Tayebati, S. (2019). Comparative machine-learning approach: a follow-up study on type 2 diabetes predictions by cross-validation methods. *Machines*, 7, 74.

[12] Bertsimas, D. and Dunn, J. (2017). Optimal classification trees. *Machine Learning*, 106, 1039–1082.

[13] Breiman, L., Friedman, J., Olshen, R. and Stone, C. (1984). *Classification and Regression Trees*. Wadsworth International Group, Belmont.

[14] Coolen F.P.A., Coolen-Maturi T. and Alqifari H. (2018). Nonparametric predictive inference for future order statistics. *Communications in Statistics: Theory and Methods*, 47, 2527–2548.

[15] Coolen F.P.A. and Maturi T. (2010). Nonparametric predictive inference for order statistics of future observations. *In: Combining Soft Computing and Statistical Methods in Data Analysis*, pp. 97–104.

[16] Coolen, F.P.A. (2011). *Nonparametric predictive inference*. Springer, Berlin.

[17] Coolen, F.P.A. and Yan, K. (2004). Nonparametric predictive inference with right-censored data. *Journal of Statistical Planning and Inference*, 126, 25–54.

[18] Coolen, F.P.A., Coolen-Schrijner, P. and Yan, K. (2002). Nonparametric predictive inference in reliability. *Reliability Engineering & System Safety*, 78, 185–193.

[19] Coolen-Maturi T., Coolen F.P.A. and Alabdulhadi M. (2020). Nonparametric predictive inference for diagnostic test thresholds. *Communications in Statistics-Theory and Methods*, 49, 697–725.

[20] De Finetti, B. (1974). *Theory of Probability*. Wiley, London.

[21] Dua, D. and Graff, C. (2019). UCI machine learning repository. *University of California, Irvine, School of Information and Computer Science*. Http://archive.ics.uci.edu/ml.

[22] Fayyad, U. and Irani, K. (1992). On the handling in decision tree of continuous-valued attributes generation. *Machine Learning*, 8, 87–102.

[23] Fayyad, U. and Irani, K. (1993). Multi-interval discretization of continuous-valued attributes for classification learing. *In: Proceeding of the 13th International Joint Conference on Artificial Inteligence*, pp. 1022–1027.

[24] Fink P. (2018). *imptree: Classification Trees with Imprecise Probabilities*. R package version 0.5.1.

[25] Genlin, J. (2004). Survey on genetic algorithm. *Computer Applications and Software*, 2, 69–73.

[26] Haldurai L., Madhubala T. and Rajalakshmi R. (2016). A study on genetic algorithm and its applications. *International Journal of Computer Sciences and Engineering*, 4, 139.

[27] Hill B. (1988). De Finetti's theorem, induction, and $A_{(n)}$ or Bayesian nonparametric predictive inference (with discussion). *Bayesian Statistics*, 3, 211–241.

[28] Hill, M. (1968). Posterior distribution of percentiles: Bayes' theorem for sampling from a population. *Journal of the American Statistical Association*, 63, 677–691.

[29] Hornik, K. and Buchta, C. and Zeileis, A. (2009). Open-source machine learning: R meets Weka. *Computational Statistics*, 24, 225–232.

[30] Kotsiantis, S., Zaharakis, I. and Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. *Emerging Artificial Intelligence Applications in Computer Engineering*, 160, 3–24.

[31] Mantas, C., Abellán, J. and Castellano, J. (2016). Analysis of Credal-C4. 5 for classification in noisy domains. *Expert Systems with Applications*, 61, 314–326.

[32] Murthy, S. and Salzberg, S. (1995). Decision Tree Induction: How Effective Is the Greedy Heuristic? *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pp. 222–227.

[33] Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, 1, 81–106.

[34] Quinlan, J. (1993). *C4.5: Program for machine learning*. Morgan Kaufmann.

[35] R Core Team (2013). *R: A Language and Environment for Statistical Computing*.

[36] Rabbi, M., Hasan, S., Champa, A., AsifZaman, M. and Hasan, Md. (2020). Prediction of liver disorders using machine learning algorithms: a comparative study. In *2020 2nd International Conference on Advanced Information and Communication Technology*, pp. 111–116. IEEE, Dhaka (Bangladesh).

[37] Rokach, L. and Maimon, O. (2008). *Data Mining with Decision Trees: Theory and Applications*. World Scientific.

[38] Shannon, C. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27, 379–423.

[39] Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B*, 36, 111–133.

[40] Therneau T., Atkinson B. and Ripley B. (2015). *rpart: Recursive Partitioning and Regression Trees*. R package version 4.1.16.

[41] Witten, I. and Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco.