Direct Nonparametric Predictive Inference Classification Trees

Abdulmajeed Atiah H. Alharbi

A Thesis presented for the degree of
Doctor of Philosophy

Statistics and Probability
Department of Mathematical Sciences
University of Durham
England

February 2022

Dedicated

To my mother Aisha

for her unlimited love, sacrifices, prayers and support throughout all my life

To my father Atiah

for his prayers, support and belief in me

Direct Nonparametric Predictive Inference Classification Trees

Abdulmajeed Atiah H. Alharbi

Submitted for the degree of Doctor of Philosophy
February 2022

Abstract

Classification is the task of assigning a new instance to one of a set of predefined categories based on the attributes of the instance. A classification tree is one of the most commonly used techniques in the area of classification. In recent years, many statistical methodologies have been developed to make inferences using imprecise probability theory, one of which is nonparametric predictive inference (NPI). NPI has been developed for different types of data and has been successfully applied to several fields, including classification. Due to the predictive nature of NPI, it is well suited for classification, as the nature of classification is explicitly predictive as well.

In this thesis, we introduce a novel classification tree algorithm which we call the Direct Nonparametric Predictive Inference (D-NPI) classification algorithm. The D-NPI algorithm is completely based on the NPI approach, and it does not use any other assumptions. As a first step for developing the D-NPI classification method, we restrict our focus to binary and multinomial data types. The D-NPI algorithm uses a new split criterion called Correct Indication (CI), which is completely based on NPI and does not use any additional concepts such as entropy. The CI reflects how informative attribute variables are, hence if the attribute variable is very informative, it gives high NPI lower and upper probabilities for CI. In addition, the CI reports the strength of the evidence that the attribute variables will indicate regarding the possible class state for future instances, based on the data. The performance of the D-NPI classification algorithm is compared against several classification algo-

rithms from the literature, including some imprecise probability algorithms, using different evaluation measures. The experimental results indicate that the D-NPI classification algorithm performs well and tends to slightly outperform other classification algorithms.

Finally, a study of the D-NPI classification tree algorithm with noisy data is presented. Noisy data are data that contain incorrect values for the attribute variables or class variable. The performance of the D-NPI classification tree algorithm with noisy data is studied and compared to other classification tree algorithms when different levels of random noise are added to the class variable or to attribute variables. The results indicate that the D-NPI classification algorithm performs well with class noise and slightly outperforms other classification algorithms, while there is no single classification algorithm that acts as the best performing algorithm with attribute noise.

Declaration

The work in this thesis is based on research carried out at the Department of Mathematical Sciences, Durham University, England. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text.

Acknowledgements

First and foremost, I am truly grateful to Allah, the Most Gracious, the Most Merciful for the countless blessings he has bestowed on me generally in my life and particularly in accomplishing this thesis.

I would like to express my sincere gratitude and thanks to my supervisors, Prof. Frank Coolen and Dr. Tahani Coolen-Maturi for their continuous support, valuable advice, help and guidance throughout the period of my PhD study. Without their help, suggestions and guidance this work would not have been possible.

I am immensely grateful to my parents for their constant support, love, prayers and unwavering belief in me not only in my education but also in my journey through life. I would also like to thank my brothers and sisters for their confidence, love and prayers. My special thanks and appreciation go to my wife Nouf for her love, patience, encouragement and standing by me.

I gratefully acknowledge the financial support from Taibah University in Madinah and the Saudi Arabian Cultural Bureau in London for granting me the scholarship and offering me the opportunity to complete my studies abroad. Many thanks also to Durham University for the facilities that have enabled me to study smoothly.

Finally, many thanks go to my friends and to everyone who has assisted me, stood by me or contributed to my educational progress in any way.

Contents

Chapter 1

Introduction

1.1 Overview

Classification is one of the most common data mining techniques that is used for assigning a new instance to one of a set of predefined categories based on the attributes of the instance. The aim of classification is to successfully predict the unknown class state of an instance given the values for its attribute variables. Classification algorithms (or classifiers) are used to explain the underlying knowledge from training data in order to be applied to unseen data (test data) to predict their class states. There are many possible classification algorithms available in the literature that one might use to predict a categorical target variable, the classification tree (or decision tree) is one of the most commonly used because of its interpretational simplicity. There are a number of algorithms that are used to build classification trees, the ID3 algorithm [55] and its extension the C4.5 algorithm [56] are two of the most commonly used algorithms in order to generate classification trees.

In recent years, theories of imprecise probabilities have been widely developed for several areas of statistics. Many methods of statistical inference have been introduced based on imprecise probability theory, and it has been shown that they have some advantages over other methods based on the classical probability theory. One of the recently developed methodologies based on imprecise probability theory is Nonparametric Predictive Inference (NPI) [12, 24, 25, 26, 28]. NPI is a statistical

framework based on Hill's assumption $A_{(n)}$ [38], introduced in Section 2.4.1, with the use of lower and upper probabilities to make inferences about future instances. NPI has been introduced for several types of data sets and has many successful applications in statistics, operations research, risk and reliability [26]. In this thesis we use NPI for Bernoulli data [24] and NPI for multinomial data [28], with applications in classification trees.

Due to the predictive nature of NPI, it is well suited for classification, as the nature of classification is explicitly predictive as well. Therefore, several classification methods have been successfully developed based on the NPI approach [3, 14, 50, 51]. Different classification trees have been built based on NPI and using an extension of the information gain split criterion, which is a well-known classic split criterion used by the ID3 algorithm. These classification trees are built by replacing precise probabilities in the classical method with imprecise probabilities, which are obtained using the NPI approach. In this thesis, we build classification trees completely based on NPI and without adding any further assumptions. This is achieved by introducing a new split criterion, which is based on the NPI lower and upper probabilities and does not use any other added concepts from the literature.

In this thesis, we present a new algorithm to build classification trees using imprecise probabilities and based on the NPI approach, which we call the Direct Nonparametric Predictive Inference (D-NPI) classification tree algorithm. As a first step to develop the D-NPI classification method, we introduce the D-NPI classification algorithm for binary data, we then generalize it for multinomial data. The D-NPI classification method is a novel approach to generate classification trees by employing the NPI lower and upper probabilities for events with binary and multinomial data, without adding any further assumptions. In order to build classification trees using the D-NPI classification method, we introduce a new split criterion. The novel split criterion presented in this thesis is named Correct Indication (CI). The CI split criterion presented in this thesis is different to other split criteria in the way it does not use any additional concepts such as entropy. Based on the NPI methodology, the

CI split criterion reports the strength of the evidence that attribute variables will indicate the correct class state for new instances. Therefore, the CI split criterion reflects how informative attribute variables are with respect to predicting possible class states, and hence to select the best ones among them to place at each node when building a classification tree.

We conduct an experimental analysis on several data sets in order to assess the performance of classification trees built by the D-NPI algorithm and to compare its performance with the performance of other classical and imprecise classification tree algorithms. We also evaluate the performance of the D-NPI classification algorithm when different random noise levels are added to data sets, and then we compare its performance with other classification algorithms. Noisy data are data that contain incorrect values for the attribute variables or class variable. Real-world data sets are never perfect and often suffer from corruptions that may affect the performance of any classifier. We evaluate the performance of the classification algorithms using the classification accuracy (on testing sets), in-sample accuracy (on training sets) and average tree size. For this analysis, a 10-fold cross-validation scheme is applied on different data sets from the UCI Machine Learning Repository database.

1.2 Outline of thesis

This thesis is structured as follows. Chapter 2 introduces preliminary materials from the literature, relevant to this thesis. We begin with a brief overview of classification and classification trees. Then, a brief introduction to imprecise probabilities and NPI is given, particularly, we introduce NPI for Bernoulli data and NPI for multinomial data, where these two types are relevant to Chapter 3 and Chapter 4, respectively. Finally, we briefly summarise some previous research studies which consider classification trees from the NPI perspective.

In Chapter 3, we present a new classification method based on NPI, which we

call Direct NPI (D-NPI) classification. We illustrate how we can base classification on the NPI lower and upper probabilities for Bernoulli data without adding any further assumptions. Then, a novel split criterion (i.e. Correct Indication (CI)) is introduced to be used with binary data and for building classification trees. An experimental analysis is then conducted on five data sets in order to assess the performance of the D-NPI classification tree algorithm and to compare the results with other classification tree algorithms from the literature. Part of this chapter was presented at the 12th Workshop on Principles and Methods of Statistical Inference with Interval Probability (WPMSIIP 2019) at Durham University, UK in September 2019. Also, part of this chapter was presented at the 12th International Conference of the ERCIM WG on Computational and Methodological Statistics (CMStatistics 2019) and 13th International Conference on Computational and Financial Econometrics (CFE 2019) at University of London, UK in December 2019.

Chapter 4 generalizes the D-NPI classification method to multinomial data. We introduce the Direct NPI classification algorithm for multinomial data (D-NPI-M), which can be used with a known number of unordered categories. The D-NPI-M algorithm uses the CI split criterion, which is developed based on the NPI method for multinomial data. In this chapter, we explain how to derive the NPI lower and upper probabilities for CI, considering multinomial data. We carry out an experimental analysis on eight data sets in order to assess the performance of the D-NPI-M classification tree algorithm and to compare it with some other classification algorithms. This chapter was presented at the 12th International Symposium on Imprecise Probability: Theories and Applications (ISIPTA) at Granada, Spain in July 2021. Part of this chapter was also presented at a seminar at Durham University.

In Chapter 5, we explore the use of the D-NPI-M algorithm, introduced in Chapter 4, with noisy data sets. We begin this chapter with a brief overview of data noise and we highlight some previous research studies, which applied classification methods on noisy data sets. In order to test the performance of the D-NPI-M algorithm with noisy data sets, we add different levels of random noise to either class vari-

able or attribute variables. Then, we evaluate the performance of the D-NPI-M algorithm with the added noise levels, and we compare its performance with other classification algorithms. The results in this chapter were presented at a seminar at Durham University.

Finally, Chapter 6 presents the conclusions of this thesis. This chapter also suggests some interesting research topics to extend the research work presented in this thesis. Some of these future research topics are also discussed in the final sections of Chapters 3 to 5.

Chapter 2

Preliminaries

In this chapter we review the main topics from the literature to provide the relevant background for this thesis. An introduction to noisy data is presented at the start of Chapter 5. First, we introduce the concept of classification and classification trees with emphasis on classification tree construction and the most commonly used split criteria when building classification trees. Then, we introduce imprecise probability with highlighting the Imprecise Dirichlet Model (IDM) and some classification works done from an imprecise probability perspective. After that we present an overview of Nonparametric Predictive Inference (NPI), particularly, NPI for binary data, which is used in Chapter 3, and NPI for multinomial data, which is used in Chapter 4. Finally, we review some works on classification trees using the NPI approach.

2.1 Classification

Classification is one of the most common data mining techniques that is used to assign a new observation to one of a set of predefined categories or classes, based on the observed values of one or more attribute variables. Classification can be considered when the target variable, also called class variable, is categorical, whereas in the situation that the target variable is continuous, regression methods can be used. Throughout this thesis we will use the terms 'target variable' and 'class variable' interchangeably. The classification algorithms are built using a training set, where

classes are already known, and then their performances are evaluated on a separate testing set. There are many possible classification algorithms, or classifiers (we will use these two terms interchangeably), that have been used in the literature to predict a categorical target variable. For example, linear discriminant analysis, logistic regression, K-nearest neighbours, support vector machine, naive Bayes, classification tree and random forests. Some details and examples about these classification algorithms are given by Hastie et al. [37] and by James et al. [40].

Among the most used methods of classification, classification trees are the most widely used for classification. Classification trees are attractive because of their interpretational simplicity, and they can predict the possible class by a sequence of simple partitions. In order to build classification trees, a split criterion should be used to select the best splitting attribute variable at each node. Note that in this thesis we use the term attribute variables. They could be also called features, predictors, independent variables, or explanatory variables. More details about classification trees and split criteria are given in Section 2.2.

Classification tasks are fundamental in many situations. For example, classifiers can be used to assign individuals who are at low, medium or high risk of acquiring a particular disease; or to determine which customers can or cannot afford to buy a specific product in a supermarket, and many other examples in such situations. In this thesis we introduce a new method of classification which we call *Direct Nonparametric Predictive Inference classification*. In this method, we build classification trees using a novel split criterion which we call *Correct Indication*. Correct Indication is completely based on the lower and upper probabilities given by Nonparametric Predictive Inference (NPI).

2.2 Classification trees

A classification tree is a nonparametric technique which represents a hierarchical data structure. The idea of a classification tree is to classify a new observation into

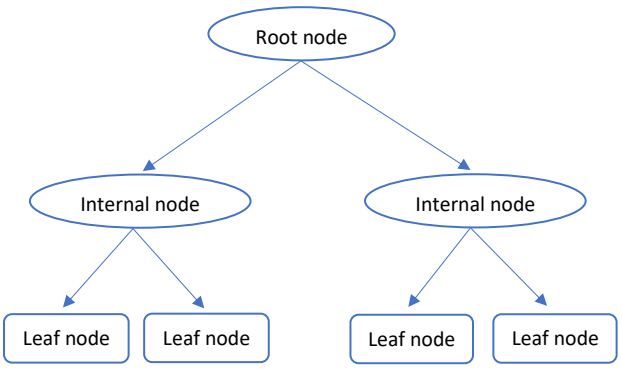

Figure 2.1: The structure of a classification tree.

one of a predefined set of classes based on its attribute values. So, classification trees are used to predict the class of a categorical target variable. Classification tree algorithms have been popular in machine learning and statistics for solving classification tasks [20, 56]. A classification tree has three types of nodes which are:

- A *root node* which is the topmost node in a tree and has no incoming edges.
- An *internal node* which has only one incoming edge and two or more outgoing edges.
- A *leaf node* which has no outgoing edges.

In a classification tree, each non-leaf node represents an attribute variable, each branch denotes the outcome of an attribute variable and each leaf node is assigned to one class label. Figure 2.1 shows the structure of a classification tree. Classifying a new observation is straightforward once a classification tree has been built. So, observations are classified by navigating them from the root node of the tree and going down to a leaf node according to the value of the attribute variables along the path [45]. Let us consider the following small example to further clarify the idea of classification trees, where the objective is to predict the possible class for a new instance based on a small number of attribute variables.

Example 2.2.1 Consider a data set that consists of a class variable with two states, A and B, and three attribute variables X, Y and Z. Where X has two possible

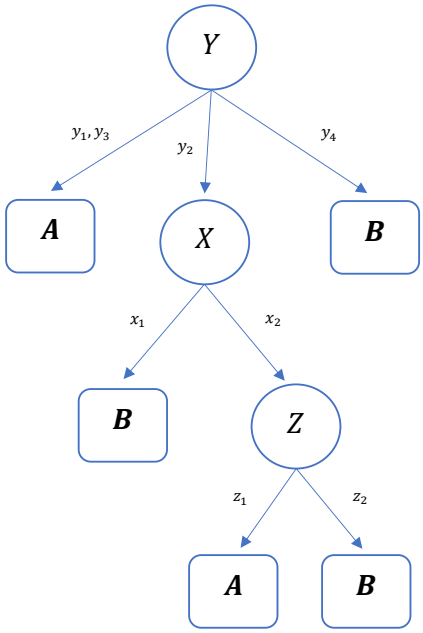

Figure 2.2: Classification tree for Example 2.2.1.

values x_1 and x_2, Y has four possible values y_1, y_2, y_3 and y_4 and Z has two possible values z_1 and z_2. Suppose that a classification algorithm leads to the tree in Figure 2.2, then a new instance with attribute values, $Y = y_2$, $X = x_2$ and $Z = z_1$ will lead to predicted class A.

□

2.2.1 Classification tree construction

To construct a classification tree, assume we have a data set which includes one or more attribute variables and a labelled class variable. First, the data set is divided into two disjoint subsets which are training and test sets. Then, the classifier is constructed using the training set and tested using the test set. One of the most commonly used methods in practice is a 10-fold cross validation [43]. In this method, the data set is split randomly into 10 subsets of approximately equal size, called folds. Each fold is used as a test set and the remaining 9 folds are combined together to use as a training set. This process is repeated 10 times so that each

fold is used once as a test set. Finally, the 10 classification accuracies achieved from these test sets are then averaged. Throughout all the experimental analysis in this thesis, we use 10-fold cross validation.

After dividing the data set into training and test sets, a classification tree algorithm is applied on the training set in order to choose the best splitting attribute variable. Different classification tree algorithms use different split criteria. After selecting the best attribute variable to be on the root node, we divide the training set into disjoint subsets based on the values of the chosen attribute variable. Note that this is the case when having a categorical attribute variable. For continuous attribute variables, a threshold is determined by the split criterion to convert it to a categorial variable. However, not all split criteria deal with continuous attribute variables. Hence, we have to decide to remove continuous variables or convert them to categorical ones. Then, we proceed in the same way for all internal nodes by choosing the best attribute variable (most informative) at each step, and dividing the training subsets based on its values. Note that each branch can only include each attribute variable once, but the same attribute variable can appear multiple times in the tree. Finally, if there is no further attribute variable available to use for splitting, we stop splitting and fix a leaf node with the most common state in the class variable. One may also stop splitting when there may still be attribute variables that could be used, but all are deemed uninformative.

There are a number of approaches available to use as a split method for building classification trees. In Section 2.2.2, we review the most commonly used classic split criteria, in addition to an imprecise split criterion that builds classification trees from imprecise probability perspective. Note that for our main contributions in this thesis in Chapter 3 and Chapter 4, we use a completely new split criterion based on the NPI approach.

2.2.2 Split criteria

A classification tree algorithm requires a split criterion which is used to select the best attribute variable to split on at each step of building the tree. Several classification tree algorithms have been developed using different split criteria. Split criteria are mainly used in order to reduce the impurity of a node. A node is 100% pure when all instances in it belong to the same class, and 100% impure when the instances are split evenly between classes. There are many split criteria that have been used for classification tree algorithms in the literature, we review three of the most commonly used classic split criteria which are Information Gain [55], Information Gain Ratio [56] and Gini Index [20]. These split criteria are used to implement the ID3, the C4.5 and the CART algorithms, respectively. In addition, we briefly review an imprecise split criterion called Imprecise Information Gain that has been used to construct credal classification trees [10]. These split criteria are introduced to compare some classification algorithms with our new algorithm, which uses a novel split criterion, which we call the *Correct Indication* split criterion. It is introduced in Chapter 3 for binary data and in Chapter 4 for multinomial data.

Information Gain: The *Information Gain* split criterion was introduced by Quinlan in 1986 [55] as a split criterion for the ID3 algorithm. Information Gain uses entropy as an impurity measure. Entropy, also called the Shannon Entropy [60], of a training set S is given by

$$H(S) = -\sum_{i=1}^{K} p_i \log_2(p_i), \tag{2.1}$$

where p_i is the proportion of S belonging to class i (for $i = 1, ..., K$), so K is the total number of classes, and $\log_2$ is used because the information is coded in bits [23]. Generally speaking, entropy represents a level of uncertainty or impurity in a set of instances. The Information Gain of an attribute B, relative to the training set S is given by

$$Gain(S, B) = H(S) - \sum_{u \in V(B)} \frac{|S_u|}{|S|} H(S_u), \tag{2.2}$$

where $V(B)$ denotes all values of attribute B, and S_u is the subset of S for which attribute B has value u, where $|S|$ denotes the cardinality of the set S. The Information Gain handles only categorical attributes.

The Information Gain split criterion is used by the ID3 algorithm in order to select the best splitting attribute variable at each node when building classification trees. The ID3 algorithm builds trees as follows:

- Given a training set with a set of instances, if all instances belong to the same class, then stop and assign this class as a leaf node.
- Consider all attribute variables that divide the training set into two or more subsets, then calculate the Information Gain (Formula 2.2) for each attribute variable.
- Select the attribute variable with the highest Information Gain, then assign it to the root node.
- Divide the training set into subsets based on the values of the selected attribute variable, then run this procedure recursively on each subset.

Gain Ratio: The *Gain Ratio* split criterion was introduced by Quinlan in 1993 [56] as an extension to the Information Gain split criterion. It is used as a split criterion for the C4.5 algorithm, hence, the C4.5 algorithm is an alternative version of the ID3 algorithm. Unlike the ID3, the C4.5 algorithm handles both categorical and continuous attributes. The Information Gain is biased toward attribute variables that have many states [55]. So, these attribute variables are more likely to be selected. To solve this problem, Quinlan [56] introduced the Gain Ratio split criterion, which normalizes the Information Gain as follows:

$$GR(S,B) = \frac{Gain(S,B)}{SI(S,B)} \ , \tag{2.3}$$

where $Gain(S,B)$ is given by Equation (2.2), and Split Information $SI(S,B)$ is given by:

$$SI(S, B) = -\sum_{j=1}^{n} \frac{|S_j|}{|S|} \log_2 \frac{|S_j|}{|S|}. \tag{2.4}$$

The $SI(S, B)$ represents the information generated by splitting the training data set S into n partitions corresponding to the values of the attribute variable B. The C4.5 algorithm builds classification trees in a similar way to the ID3 algorithm, but it uses the Gain Ratio split criterion (Formula 2.3) to select the splitting attribute variable at each node.

Gini Index: The *Gini Index* split criterion was introduced by Breiman et al. in 1984 [20] as a split criterion to implement the Classification and Regression Trees (CART) algorithm. The CART algorithm constructs only binary trees. Here, the term 'binary' means that each internal node in the classification tree can have only two outgoing edges (child nodes). The Gini Index measures the degree of impurity of an attribute with respect to the classes. It is defined as follows:

$$\text{Gini}(S) = 1 - \sum_{i=1}^{K} (p_i)^2, \tag{2.5}$$

where p_i is the relative frequency of class i in S, for $i = 1, ..., K$. The minimum value of the Gini Index is achieved when all observations in the sample belong to only one class, while the maximum value of the Gini Index is achieved when all classes have equal probability. After splitting S into two subsets S_1 and S_2 with sizes N_1 and N_2, the Gini Index of the split data is defined as

$$\text{Gini}_{\text{Split}}(S) = \frac{N_1}{N}\text{Gini}(S_1) + \frac{N_2}{N}\text{Gini}(S_2). \tag{2.6}$$

In this way, the split with the best Gini value is selected. Note that the CART classification algorithm will not be applied during the experimental analysis in this thesis, but we briefly introduce it as it one of the popular classification tree algorithm, besides the ID3 and the C4.5 algorithms.

Imprecise Information Gain: The *Imprecise Information Gain* (IIG) split criterion was introduced by Abellán and Moral in 2003 [10] to build classification trees

from an imprecise probability perspective. The IIG for an attribute variable X is defined as follows:

$$IIG(X, C) = S(K(C)) - \sum_i p(x_i) S(K(C|(X = x_i))), \tag{2.7}$$

where $S(K)$ is the maximum entropy of a credal set, and $K(C)$ and $K(C|(X = x_i))$ are credal sets for the class variable C and for C given the value x_i of the attribute variable X, respectively; and $i = 1, ..., n$ for a partition of the data set; and $p(x_i)$ is a probability distribution that belongs to the credal set $K(X)$. Credal sets are closed and convex sets of probability distributions [1]. The IIG is applied on credal sets using uncertainty measures of probability distributions [8]. For more details and extended explanations of the IIG see [8, 10, 11]. Different classification trees can be built using the IIG split criterion. For example, one can build a classification tree using the maximum entropy distributions from the credal set of distributions associated with the Imprecise Dirichlet Model (IDM) [1] or with the Nonparametric Predictive Inference for multinomial data (NPI-M) [2], which are introduced in Sections 2.3 and 2.4, respectively. In this thesis, we refer to a classification tree built with the IIG and the IDM by IDM *algorithm*; the IIG and NPI-M by NPI-M *algorithm*; and the IIG and the A-NPI-M by A-NPI-M *algorithm*, where A-NPI-M stands for Approximate NPI-M.

2.3 Imprecise probabilities

In the middle of the 19th century, the idea of imprecise probabilities was first proposed in Boole's book [17]. Since then, imprecise probability based methods have been developed for many areas of statistics. An overview of the main aspects of imprecise probabilities theory and applications has been presented by Augustin et al. [13], and by Walley [64].

In classical probability theory, for an event A, a precise probability $p(A) \in [0, 1]$ is used to quantify uncertainty about A, where p is a probability measure

satisfying Kolmogorov's axioms [13]. In real world data sets, precise probability calculated from the data is likely to be inaccurate, hence, having the possibility to use imprecise probability may give advantages over the use of precise probability. Imprecise probability uses lower and upper probabilities for the event, and hence reflects more uncertainty about the event. Unlike classical probability, in imprecise probability we assign an interval probability for an event A, such as $[\underline{P}(A), \overline{P}(A)]$, where $0 \leq \underline{P}(A) \leq \overline{P}(A) \leq 1$, and where $\underline{P}(A)$ denotes the lower probability and $\overline{P}(A)$ denotes the upper probability for event A. The classical probability is a special case in imprecise probability which occurs when $\underline{P}(A) = \overline{P}(A)$. Complete lack of information about an event A is represented by $\underline{P}(A) = 0$ and $\overline{P}(A) = 1$. Weichselberger [66] defined the structure, $\mathcal{M}$:

$$\mathcal{M} = \{p(.) : \underline{P}(A) \leq p(A) \leq \overline{P}(A), \forall A \in \mathcal{A}\}, \tag{2.8}$$

where $\mathcal{A}$ is a set of events, and $p(.)$ is a set-function defined on $\mathcal{A}$ satisfying Kolmogorov's axioms in classical probability theory. The lower and upper probabilities for an event A are:

$$\underline{P}(A) = \inf_{p(.) \in \mathcal{M}} p(A) \tag{2.9}$$

and

$$\overline{P}(A) = \sup_{p(.) \in \mathcal{M}} p(A). \tag{2.10}$$

Here some basic concepts of imprecise probability are briefly highlighted. For more details and a complete introduction to imprecise probability, we refer to Walley [64] and Augustin et al. [13].

In 1996, Walley introduced an imprecise probability model for inference from multinomial data [65], which is called the Imprecise Dirichlet Model (IDM). The IDM is one of the most popular imprecise probability models. Assume that we have a data set with N observations. Let X be a variable whose values, or categories belong to $\{x_1, ..., x_n\}$, and let n_{x_i} denote the total number of observations in x_i, for $i = 1, ..., n$. The IDM-based lower and upper probabilities for the event that the next future observation, X_{n+1} will be in x_i, are

$$\underline{P}_{IDM}(X_{n+1} \in x_i) = \frac{n_{x_i}}{N + \tilde{s}} \tag{2.11}$$

and

$$\overline{P}_{IDM}(X_{n+1} \in x_i) = \frac{n_{x_i} + \tilde{s}}{N + \tilde{s}}, \tag{2.12}$$

where $\tilde{s}$ is a parameter which is chosen independently of the data (this parameter is written as s by Walley [65] but because we use s later in the thesis for another parameter we denote a different symbol). The value of $\tilde{s}$ determines how quickly the lower and upper probabilities converge when the sample size increases [65]. Walley suggested to choose the value of the parameter $\tilde{s}$ equal to 1 or 2 [65]. As shown by Abellán [1], the IDM gives imprecise probabilities that lead to the following (closed and convex) credal set of probability distributions,

$$L = \left\{ p \mid p\left(x_i\right) \in \left[\frac{n_{x_i}}{N + \tilde{s}}, \frac{n_{x_i} + \tilde{s}}{N + \tilde{s}} \right], \quad i = 1, \ldots, n, \sum_{i=1}^{n} p(x_i) = 1 \right\}. \tag{2.13}$$

The IDM has been applied to many statistical problems in the literature. Some of these applications were reviewed by Bernard [15]. However, the use of the IDM has been criticised for some disadvantages [54]. Some of these disadvantages of the IDM were already discussed by Walley [65], and by other researchers, which motivate researchers to introduce alternative models for inference from multinomial data. The presence of these disadvantages motivates researchers to develop alternative inference models. Coolen and Augustin [27, 28] proposed a new model for inference from multinomial data, which is Nonparametric Predictive Inference for Multinomial data (NPI-M). This model is an alternative to the IDM and is based on the NPI method. The NPI-M is free from some disadvantages of the IDM such as that the NPI-M does not make any prior assumptions about the data.

In recent years, imprecise probability theory has been applied to classification trees. One of the first applications was introduced by Abellán and Moral [10], where they estimate the probabilities of the classes in each leaf node by using the Imprecise

Dirichlet Model. Following that paper, many papers have presented classification trees from imprecise probability perspective [9, 46, 47, 52]. Imprecise probability theory has also been used in classification by random forests as in [4, 6, 62]. Random forests are a combination of trees where each tree is built using a random vector sampled independently from the training set with the same distribution for all trees, and each tree votes for the most popular class [19]. Then, the majority vote is taken for classification. There are also several other works on classification trees from imprecise probability perspective, and based on Nonparametric Predictive Inference [3, 14, 35]. Abellán et al. [3] and Baker [14] show that applying the NPI-M to classification trees leads to slightly better classification accuracy than the IDM. In this thesis, we also introduce a new application of classification trees with imprecise probability, and based on the Nonparametric Predictive Inference approach. Our new classification method can base classification on the lower and upper probabilities given by NPI, without adding any further assumptions. We build classification trees using a new split criterion, which is also completely based on NPI. The Nonparametric Predictive Inference approach is introduced in Section 2.4. We also introduce some works on classification from the NPI perspective with more explanations in Section 2.5.

2.4 Nonparametric Predictive Inference (NPI)

Nonparametric Predictive Inference (NPI) is a statistical methodology which uses only a few modelling assumptions to learn from data in the absence of prior knowledge. NPI is based on Hill's assumption $A_{(n)}$ [38], explained in Section 2.4.1, and uses lower and upper probabilities, also known as imprecise probabilities, to quantify uncertainty [12]. NPI has been developed in recent years for different applications in statistics, operations research, risk and reliability [26]. NPI has also been presented for different types of data, such as Bernoulli data [24], real-valued data [12], right-censored data [29], ordinal data [33] and multinomial data [14, 27, 28]. In Section 2.4.1, we introduce Hill's assumption $A_{(n)}$. Then, we introduce NPI for Bernoulli

data in Section 2.4.2, and NPI for multinomial data in Section 2.4.3.

2.4.1 $A_{(n)}$ assumption

Hill [38] introduced the assumption $A_{(n)}$ for prediction about future observations when there is no strong prior knowledge about the form of the underlying distribution of a random quantity. Hill's assumption $A_{(n)}$ directly provides probabilities for one or more real-valued future random quantities, based on observed values of related random quantities. Let $X_1, ..., X_n, X_{n+1}$ be real-valued and exchangeable random quantities, where we assume that the probability of ties is zero. Let the ranked observed values of $X_1, ..., X_n$ be denoted by $x_1 < ... < x_n$, and let $x_0 = -\infty$ and $x_{n+1} = \infty$ for ease of notation. These ordered observations partition the real line into $n+1$ open intervals $I_j = (x_{j-1}, x_j)$ for $j = 1, ..., n+1$. The assumption $A_{(n)}$ states that the next future observation, represented by a random quantity X_{n+1}, falls equally likely in any interval I_j with probability $\frac{1}{n+1}$ for each $j = 1, ..., n+1$, i.e. $P(X_{n+1} \in I_j) = \frac{1}{n+1}$. Hill's assumption $A_{(n)}$ does not assume anything else, and it is clearly a post-data assumption related to exchangeability of $n+1$ values on the real-line [30].

2.4.2 NPI for Bernoulli data

This section summarises NPI for Bernoulli random quantities as introduced by Coolen [24]. Suppose that we have a sequence of $n+m$ exchangeable Bernoulli trials where the possible outcomes of each trial are either 'success' or 'failure', and the data consist of s successes in n trials, where m stands for future trials. Let Y_1^n and Y_{n+1}^{n+m} denote the random number of successes in trials 1 to n, and $n+1$ to $n+m$, respectively. Because of the assumed exchangeability of all trials, a sufficient representation of the data for the inference considered is $Y_1^n = s$. Now, let $R_t = \{r_1, ..., r_t\}$, with $1 \leq t \leq m+1$ and $0 \leq r_1 < r_2 < \ldots < r_t \leq m$, and for ease of notation, let $\binom{s+r_0}{s} = 0$. Then, the NPI upper probability for the conditional

event $Y_{n+1}^{n+m} \in R_t | Y_1^n = s$, for $s \in \{0, \ldots, n\}$, is

$$\overline{P}\left(Y_{n+1}^{n+m} \in R_t | Y_1^n = s\right) = \binom{n+m}{n}^{-1} \sum_{j=1}^{t} \left[\binom{s+r_j}{s} - \binom{s+r_{j-1}}{s} \right] \binom{n-s+m-r_j}{n-s}. \quad (2.14)$$

It is sufficient to determine the NPI upper probability only, as the corresponding NPI lower probability, can be derived by the conjugacy property, $\underline{P}(A) = 1 - \overline{P}(A^c)$, where A^c is the complementary event to A.

$$\underline{P}\left(Y_{n+1}^{n+m} \in R_t | Y_1^n = s\right) = 1 - \overline{P}\left(Y_{n+1}^{n+m} \in R_t^c | Y_1^n = s\right) \quad (2.15)$$

where $R_t^c = \{0, 1, \ldots, m\} \backslash R_t$. More details and examples about the NPI for Bernoulli quantities are give by Coolen [24].

In Chapter 3, we base classification on these NPI lower and upper probabilities, but considering only a single future observation, i.e. $m = 1$. For this case, the NPI lower and upper probabilities are

$$\underline{P}\left(Y_{n+1}^{n+1} = 1 | Y_1^n = s\right) = \frac{s}{n+1} \quad (2.16)$$

and

$$\overline{P}\left(Y_{n+1}^{n+1} = 1 | Y_1^n = s\right) = \frac{s+1}{n+1}. \quad (2.17)$$

Similarly, the NPI lower and upper probabilities for the event that $Y_{n+1}^{n+1} = 0 | Y_1^n = s$ are

$$\underline{P}\left(Y_{n+1}^{n+1} = 0 | Y_1^n = s\right) = \frac{n-s}{n+1} \quad (2.18)$$

and

$$\overline{P}\left(Y_{n+1}^{n+1} = 0 | Y_1^n = s\right) = \frac{n-s+1}{n+1}. \quad (2.19)$$

2.4.3 NPI for Multinomial data

Coolen and Augustin [12, 27, 28] have developed Nonparametric Predictive Inference for Multinomial data (NPI-M). The NPI-M is based on the *circular-*$A_{(n)}$ assumption, which is a variation of Hill's assumption $A_{(n)}$ [27]. Since multinomial data are

represented as observations on a probability wheel, and hence as circular data, we use the *circular-*$A_{(n)}$ assumption, which is denoted by $\text{Ⓐ}_{(n)}$ [25, 27]. Suppose that we have ordered circular data $y_1 < y_2 < ... < y_n$ which create n intervals on a circle, represented as $I_j = (y_j, y_{j+1})$ for $j = 1, ..., n-1$ and $I_n = (y_n, y_1)$. The assumption $\text{Ⓐ}_{(n)}$ states that the next future observation, represented by a random quantity Y_{n+1}, falls equally likely in any interval I_j for each $j = 1, ..., n$, i.e. $P(Y_{n+1} \in I_j) = \frac{1}{n}$. The $\text{Ⓐ}_{(n)}$ is a post-data assumption related to exchangeability for such circular data.

The NPI-M model represents multinomial data as observations on a probability wheel, where each of these n observations is represented by a line from the center of the wheel to its circumference. The wheel is divided into n equally-sized slices. Using the *circular-*$A_{(n)}$ assumption, the next future observation has probability mass $\frac{1}{n}$ of falling into any given slice. Hence, we have to decide which category each of these slices represents.

Coolen and Augustin [28] assume that each observed category is represented by one single segment of the probability wheel, where the segment is an area between two lines from the center to the circumference of the wheel. Combining this assumption with *circular-*$A_{(n)}$ implies that two or more lines representing observations in the same categories are positioned next to each other. Therefore, a slice that is bordered by two lines representing different categories is a separating slice, which could be assigned to any of these different categories or to unobserved category. It is also assumed that there is no ordering of the categories, and hence no ordering of the segments on the wheel.

Coolen and Augustin introduced the NPI-M for the case of a known number of categories [28] and for the case of an unknown number of categories [27]. We restrict our focus in this thesis, for Chapters 4 and 5, on the case where the number of possible categories, denoted by K, is known. We assume that $K \geq 3$. However, for the case when $K = 2$, the NPI-M can be used, but using NPI for Bernoulli data [24] is more appropriate as it leads to slightly less imprecision. In the following we

summarise the results of Coolen and Augustin [28] when K is known, using similar notation as in [28].

Suppose that there are $K \geq 3$ possible categories denoted by $C_1, ..., C_K$. We assume that $C_1, ..., C_k$ for $1 \leq k \leq K$ are observed categories, and $C_{k+1}, ..., C_K$ are unobserved categories. Let n_j represent the number of observations in category C_j for $j = 1, ..., k$, and let the total number of observations be $n = \sum_{j=1}^{k} n_j$. The general event of interest can be denoted by

$$Y_{n+1} \in \bigcup_{j \in J} C_j \tag{2.20}$$

where $J \subseteq \{1, ..., K\}$. Let $OJ = J \bigcap \{1, ..., k\}$ and $UJ = J \bigcap \{k+1, ..., K\}$ represent the index-set for the categories in the event of interest that have been observed, and the index-set for the categories in the event of interest that have not been observed, respectively. Let $r = |OJ|$ and $l = |UJ|$, hence $0 \leq r \leq k$ and $0 \leq l \leq K - k$. This leads to $k - r$ observed categories which are not included in the event of interest, and $K - k - l$ unobserved categories which are not included in the event of interest.

To find the NPI lower and upper probabilities for the event of interest, we have to consider all the possible configurations of the different segments on the probability wheel. The NPI lower probability for the event of interest is achieved by selecting the configuration that minimises the number of slices assigned to the event, and the NPI upper probability for the event of interest is achieved by selecting the configuration that maximises the number of slices assigned to the event.

The NPI-M lower probability for the event of interest (2.20), based on n observations, is

$$\underline{P}\left(Y_{n+1} \in \bigcup_{j \in J} C_j\right) = \frac{1}{n}\left(\sum_{j \in OJ} n_j - r + \max(2r + l - K, 0)\right) \tag{2.21}$$

and the NPI-M upper probability for the event of interest (2.20), based on n observations, is

$$\overline{P}\left(Y_{n+1} \in \bigcup_{j \in J} C_j\right) = \frac{1}{n}\left(\sum_{j \in OJ} n_j - r + \min(2r + l, k)\right). \tag{2.22}$$

The derivation of the NPI-M lower and upper probabilities (2.21) and (2.22) are explained with more details, examples and discussions in [28]. Coolen and Augustin [28] also presented some fundamental properties of NPI-M lower and upper probabilities.

For events $Y_{n+1} \in C_i$, so considering only a single category, the NPI-M lower and upper probabilities are

$$\underline{P}\left(Y_{n+1} \in C_i\right) = \max\left(0, \frac{n_i - 1}{n}\right) \tag{2.23}$$

and

$$\overline{P}\left(Y_{n+1} \in C_i\right) = \min\left(\frac{n_i + 1}{n}, 1\right) \tag{2.24}$$

where $i = 1, ..., K$ and n_i is the number of observations in category i. For classification problems, generally events with only observed categories are considered. Therefore, we consider the case that these single categories have been observed, i.e. $n_i > 0$.

For unobserved categories, the NPI-M lower and upper probabilities are

$$\underline{P}\left(Y_{n+1} \in C_i\right) = 0 \tag{2.25}$$

and

$$\overline{P}\left(Y_{n+1} \in C_i\right) = \frac{1}{n}. \tag{2.26}$$

In the following example we illustrate how we can calculate the NPI-M lower and upper probabilities for events with a single category, as these NPI-M lower and upper probabilities are used in our method of Direct NPI classification in Chapters 4 and 5.

Example 2.4.1 Suppose that there are five possible categories, C_1, C_2, C_3, C_4 and C_5, which have been observed 4, 2, 1, 0 and 0 times respectively. Suppose we

are interested in the event $Y_8 \in C_i$, for $i = 1, ..., 5$. The NPI-M lower and upper probabilities for the event $Y_8 \in C_1$ are as follows:

$$\underline{P}\,(Y_8 \in C_1) = \max\left(0, \frac{4-1}{7}\right) = \frac{3}{7}$$

and

$$\overline{P}\,(Y_8 \in C_1) = \min\left(\frac{4+1}{7}, 1\right) = \frac{5}{7}.$$

Similarly, the NPI-M lower and upper probabilities for the events $Y_8 \in C_2$ to $Y_8 \in C_5$ are

$$\begin{aligned}
\underline{P}\,(Y_8 \in C_2) &= \frac{1}{7} \quad \text{and} \quad \overline{P}\,(Y_8 \in C_2) = \frac{3}{7} \\
\underline{P}\,(Y_8 \in C_3) &= 0 \quad \text{and} \quad \overline{P}\,(Y_8 \in C_3) = \frac{2}{7} \\
\underline{P}\,(Y_8 \in C_4) &= 0 \quad \text{and} \quad \overline{P}\,(Y_8 \in C_4) = \frac{1}{7} \\
\underline{P}\,(Y_8 \in C_5) &= 0 \quad \text{and} \quad \overline{P}\,(Y_8 \in C_5) = \frac{1}{7}.
\end{aligned}$$

□

2.5 NPI classification

In imprecise probability applications, the maximum entropy measure is used to quantify uncertainty for building classification trees. Abellán and Moral [10] considered the use of maximum entropy distributions taken from the credal set associated with the Imprecise Dirichlet Model (IDM) for building classification trees. However, as the use of the IDM has been criticised due to some drawbacks, the NPI-M could be an appropriate alternative to the IDM for quantifying uncertainty, and hence replace the IDM in the maximum entropy measure. Unlike the IDM, the NPI-M does not assume any prior knowledge about the data. Due to the predictive nature of the NPI approach, it is well suited for classification, as the nature of classification is explicitly predictive as well.

The NPI-M has been applied successfully in the area of classification [2, 3, 14, 35, 50, 51]. In [2], two different algorithms were presented in order to obtain the maximum entropy distribution using the NPI approach. The first one is the NPI-M algorithm and the second one is an approximation of the NPI-M which is referred to as the A-NPI-M algorithm [2]. These two algorithms are also presented and explained in detail with some examples in Baker's thesis [14]. Note that due to some constraints of the NPI-M model, the set of probability distributions obtained from the NPI-M is not a credal set [2]. The NPI-M does not lead to a credal set because of the limitations which are caused by the constraints on the probability wheel. However, taking these constraints into account, Abellán et al. [2] presented an algorithm that obtain the maximum entropy distribution based on the NPI-M model. For ease of application, the A-NPI-M algorithm which leads to a credal set is used to obtain the maximum entropy distribution. The A-NPI-M is simpler than the NPI-M because it does not necessarily consider the constraints associated with the NPI-M model. These two NPI algorithms (NPI-M and A-NPI-M) can be used in order to build classification trees using the maximum entropy measure in a similar way to the IDM classification trees.

Abellán et al. [3] have presented an application of NPI-M in classification trees. They have conducted extensive experiments to assess the performance of classification trees built using the NPI-M model. They compare the performance of classification trees generated by the NPI-M with the performance of trees generated by the IDM and C4.5 algorithms. Their results show that the NPI-M slightly outperforms the IDM, besides, classification trees which are built using the NPI-M are smaller than the ones built using the C4.5 and IDM algorithms.

Another experiment has been conducted by Baker [14] in order to study the performance of the A-NPI-M algorithm when it is used to build classification trees. She compared the performance of the A-NPI-M algorithm to the IDM algorithm and three different classical algorithms. The findings indicate that the A-NPI-M algorithm significantly outperforms other classical methods such as the ID3 and the

C4.5 algorithms, but has very similar performance to the IDM algorithm. Baker [14] has carried out additional experiments in order to compare the A-NPI-M and NPI-M algorithms, and to determine which one of them is more successful when applied to classification trees. She also compared these two algorithms with the IDM algorithm. She found that all these algorithms have similar performance with the A-NPI-M and NPI-M algorithms slightly superior to the IDM algorithm. She also concludes that the A-NPI-M and NPI-M algorithms perform in a very similar way where in most data sets the difference in classification accuracy is negligible.

A recent application of the use of NPI-M in classification trees has been presented by Moral-García et al. [51]. They proposed a new adaptation of decision trees to Multi-Label Classification based on the NPI-M model. The Multi-Label Classification refers to the task of predicting the set of labels, which are associated with an instance. This method assumes that an instance could be associated with multiple labels. Their findings indicate that the Multi-Label Decision Trees based on NPI-M perform better than the Multi-Label Decision Trees based on precise probabilities, particularly when working with noisy data. Another application of NPI-M in classification is presented by Moral-García et al. [50]. They build an imprecise classifier based on the NPI-M and IDM algorithms, where the IDM algorithm is implemented with different values of the parameter $\tilde{s}$. In imprecise classification, trees may return a set of classes in leaf nodes rather than a single class. The results achieved by Moral et al. [50] indicate that the performance of the NPI-M is the same as the performance of the IDM with the best choice of the parameter $\tilde{s}$. As a result, the NPI-M is more suitable than the IDM to be used in imprecise classification since the NPI-M is parameter free.

In this thesis, we contribute to the theory of classification from the NPI perspective, by developing a new classification tree algorithm. The novel classification method presented in this thesis has been named *Direct Nonparametric Predictive Inference classification* (D-NPI). The D-NPI classification method commences by applying the NPI approach to a classification tree with the use of a new split crite-

rion named *Correct Indication* (CI) to choose the best splitting attribute variable at each node. The CI split criterion is completely based on the NPI lower and upper probabilities for events containing binary or multinomial data, and it does not use any additional concepts such as entropy. We first introduce the D-NPI classification method for binary data in Chapter 3, then we generalise that for multinomial data in Chapter 4.

Chapter 3

Direct NPI Classification for Binary Data

3.1 Introduction

In this chapter, we introduce a new method of classification using imprecise probabilities and based on the NPI approach, which we call Direct Nonparametric Predictive Inference (D-NPI) classification. We develop the D-NPI classification method for binary data in this chapter, while in Chapter 4 we generalize the D-NPI classification method to multinomial data. The D-NPI classification method can base classification on the NPI lower and upper probabilities for events containing binary data, without adding any further assumptions or information. We build D-NPI classification trees using a new split criterion, named *Correct Indication* (CI). The CI split criterion is completely based on the NPI lower and upper probabilities, and it does not use any additional concepts like entropy. Using this split criterion, we present a new classification tree algorithm, which is based on the D-NPI classification method.

We have carried out an experimental analysis in order to assess the performance of the D-NPI classification algorithm when building classification trees. We have also compared the performance of the D-NPI classification algorithm with other classical algorithm such as the C4.5 algorithm [56], and other imprecise algorithms based on the IDM or NPI-M such as the NPI-M algorithm [3, 14], the A-NPI-M

algorithm [3, 14] and the IDM algorithm [10] with two choices of the parameter $\tilde{s}$. These classification algorithms are introduced in Chapter 2. We then have evaluated the performance of the classification algorithms using the classification accuracy (on testing sets), in-sample accuracy (on training sets) and average tree size. A 10-fold cross validation scheme has been applied on different data sets from the UCI Machine Learning Repository database [32].

This chapter is organized as follows: Section 3.2 introduces the method of D-NPI classification, and illustrates how we can base classification on the NPI lower and upper probabilities for events with binary outcomes. In Section 3.3, we explain how the D-NPI classification method can be used to build classification trees. First, we introduce the novel split criterion, Correct Indication (CI), which is used to select the best splitting attribute when building classification trees. Then, the building process of a classification tree using our method is described with an illustrative example. In Section 3.4, we present results of an experimental analysis that is conducted to assess the performance of our method and compare it with some existing methods from the literature. Finally, some concluding remarks and related topics for future research are presented in Section 3.5.

3.2 Direct NPI classification

In this section, we introduce Direct NPI classification using NPI for Bernoulli data [24], introduced in Section 2.4.2. We illustrate how we can base classification on the NPI lower and upper probabilities for binary data without adding any further assumptions. Then, an example to illustrate the Direct NPI classification will be given. We first develop the method of Direct NPI classification for complete binary data, where both the class variable and the attribute variables are binary.

Assume that we have a data set of n instances which only have two values, 0 or 1. These instances can also be indicated as 'negative' or 'positive' cases. Suppose that there are $T \geq 1$ binary attribute variables. Let t_j indicate attribute variables,

for $j \in \{1, ..., T\}$. The value of each attribute is either 0 or 1, i.e. $t_j = 0$ or $t_j = 1$. Let D be a binary class variable, where $D = 0$ or $D = 1$. Let n^0 denote the total number of instances with $D = 0$, and let n^1 denote the total number of instances with $D = 1$, so $n = n^0 + n^1$.

Suppose that we want to see if attribute t_j is useful for indicating the possible class state for a future instance. The attribute t_j is useful if an instance with attribute value $t_j = 1$, has a high probability of being classified as $D = 1$, so that $t_j = 1$ is an indicator for $D = 1$, and an instance with attribute value $t_j = 0$, has a high probability of being classified as $D = 0$, so that $t_j = 0$ is an indicator for $D = 0$. Therefore, we are interested in the conditional events $D = 1|t_j = 1$ and $D = 0|t_j = 0$. Note here that this approach to indication allows the attribute values to be relabelled, possibly multiple times in the construction of a single tree. More clarification and example about this issue of relabelling are given in Section 3.3.2. Of course, the ideal situation would be that all instances with attribute value $t_j = 1$ are classified as $D = 1$, and all instances with attribute value $t_j = 0$ are classified as $D = 0$. These conditional events indicate that attribute value 1 ($t_j = 1$) is related to class state 1 ($D = 1$) in terms of the data set, and similarly for $D = 0|t_j = 0$. In other words, each attribute value in the data set indicates a particular class state. We consider such events for one future instance for which the attributes are available but we do not know its class states. This instance is assumed to be exchangeable with all other n instances in the data set. Let $n(t_j = 1)$ be the total number of instances in the data set which have value $t_j = 1$. Let $n^1(t_j = 1)$ denote the total number of instances which have value $t_j = 1$ and which are classified as $D = 1$, and let $n^0(t_j = 1)$ denote the total number of instances which have value $t_j = 1$ but are classified as $D = 0$. Thus, $n(t_j = 1) = n^1(t_j = 1) + n^0(t_j = 1)$. Note that these numbers are known from the data set.

It should be emphasized that the inference considers an instance which is not in the data, and hence, its class state is unknown. We denote the unknown class state of one future instance, which is not included in the data set, by D_{n+1}. Judgement on

correctness of the predictive inference on this instance is impossible at the time of such predictions, but the effectiveness of such judgments can be considered based on success of the attribute variables for the n available instances in the data set. Using NPI for Bernoulli data [24], introduced in Section 2.4.2, see Equations (2.16) and (2.17), we can derive the NPI lower and upper probabilities for $D_{n+1} = 1|t_{n+1,j} = 1$. Note here that $t_{n+1,j} = 1$ is the attribute value for this future instance. We can provide the NPI lower and upper probabilities for the event that a future instance, which is not included in the data, say 'instance $n+1$', will be classified as $D_{n+1} = 1$ given that its attribute value, $t_{n+1,j} = 1$. The NPI lower probability is

$$\underline{P}\left(D_{n+1} = 1|t_{n+1,j} = 1\right) = \frac{n^1\left(t_j = 1\right)}{n\left(t_j = 1\right) + 1}, \tag{3.1}$$

and the NPI upper probability is

$$\overline{P}\left(D_{n+1} = 1|t_{n+1,j} = 1\right) = \frac{n^1\left(t_j = 1\right) + 1}{n\left(t_j = 1\right) + 1}. \tag{3.2}$$

We can also derive the NPI lower and upper probabilities for $D_{n+1} = 0|t_{n+1,j} = 1$ via the conjugacy property,

$$\begin{aligned}\underline{P}\left(D_{n+1} = 0|t_{n+1,j} = 1\right) &= 1 - \overline{P}\left(D_{n+1} = 1|t_{n+1,j} = 1\right) \\ &= 1 - \frac{n^1\left(t_j = 1\right) + 1}{n\left(t_j = 1\right) + 1} = \frac{n^0\left(t_j = 1\right)}{n\left(t_j = 1\right) + 1},\end{aligned}$$

and

$$\begin{aligned}\overline{P}\left(D_{n+1} = 0|t_{n+1,j} = 1\right) &= 1 - \underline{P}\left(D_{n+1} = 1|t_{n+1,j} = 1\right) \\ &= 1 - \frac{n^1\left(t_j = 1\right)}{n\left(t_j = 1\right) + 1} = \frac{n^0\left(t_j = 1\right) + 1}{n\left(t_j = 1\right) + 1}.\end{aligned}$$

Similarly, the NPI lower and upper probabilities for $D_{n+1} = 0|t_{n+1,j} = 0$, i.e. the NPI lower and upper probabilities for the event that the class state is $D_{n+1} = 0$ for this future instance if the value of its attribute, $t_{n+1,j} = 0$, are given as follows:

$$\underline{P}\left(D_{n+1} = 0|t_{n+1,j} = 0\right) = \frac{n^0\left(t_j = 0\right)}{n\left(t_j = 0\right) + 1}, \tag{3.3}$$

and

$$\overline{P}\left(D_{n+1}=0|t_{n+1,j}=0\right)=\frac{n^{0}\left(t_{j}=0\right)+1}{n\left(t_{j}=0\right)+1}. \tag{3.4}$$

Also, by the conjugacy property, the NPI lower and upper probabilities for $D_{n+1}=1|t_{n+1,j}=0$ are

$$\underline{P}\left(D_{n+1}=1|t_{n+1,j}=0\right)=1-\overline{P}\left(D_{n+1}=0|t_{n+1,j}=0\right)=\frac{n^{1}\left(t_{j}=0\right)}{n\left(t_{j}=0\right)+1},$$

and

$$\overline{P}\left(D_{n+1}=1|t_{n+1,j}=0\right)=1-\underline{P}\left(D_{n+1}=0|t_{n+1,j}=0\right)=\frac{n^{1}\left(t_{j}=0\right)+1}{n\left(t_{j}=0\right)+1}.$$

It should be noticed that the values of the above NPI lower and upper probabilities will report the strength of the evidence for the class state of the future instance, which is not included in the data, but is assumed to be exchangeable with the n other instances in the data set. Of course, we can not conclude whether or not a prediction based on the value of attribute t_j is correct, but the above values report the predicted class state for future instances based on the n available instances in the data set. We can base NPI lower and upper probabilities for D_{n+1} on the values of each binary attribute, and then select the best attribute variable among them, which gives the largest lower and upper probabilities. This is actually the main aim of classification, hence, we can base classification on the NPI lower and upper probabilities. In Section 3.3, we explain how to build classification trees based on the D-NPI classification method. Example 3.2.1 illustrates the proposed D-NPI classification method.

Example 3.2.1 Suppose we have a data set of 100 people, where 35 people have a particular disease and 65 people do not have the disease. Consider the test t_1 (attribute variable) which is performed upon all people as shown in Table 3.1, where $t_1 = 1$ and $t_1 = 0$ represent a positive and negative test result, respectively. Let D represent disease status, where $D = 1$ denotes the presence of disease and $D = 0$

denotes the absence of disease. We use this data set to illustrate the above NPI lower and upper probabilities.

	$t_1 = 1$	$t_1 = 0$
$D = 1$	15	20
$D = 0$	5	60

Table 3.1: The result of test t_1.

First, just for the sake of clarifying some notations introduced earlier in this section, from Table 3.1, the total number of instances which have values $t_1 = 1$ is $n(t_1 = 1) = 20$, and the total number of instances which have values $t_1 = 1$ and which are classified as $D = 1$ is $n^1(t_1 = 1) = 15$, while the total number of instances which have values $t_1 = 1$ but which are classified as $D = 0$ is $n^0(t_1 = 1) = 5$. As a result, $n^1(t_1 = 1) + n^0(t_1 = 1) = n(t_1 = 1)$. Similarly, the values with regard to $t_1 = 0$ are $n(t_1 = 0) = 80$, $n^1(t_1 = 0) = 20$ and $n^0(t_1 = 0) = 60$.

Secondly, we consider the result of test t_1 for one future person who is not included in the data set, to predict his/her disease status. We calculate the NPI lower and upper probabilities for the event that a future person, say person 101, who is not included in the data set but is assumed to be exchangeable with the 100 people in the data set has the disease ($D_{101} = 1$) given that he/she has tested positive ($t_1 = 1$). The NPI lower probability, from Equation (3.1), is

$$\underline{P}\left(D_{101} = 1 | t_{101,1} = 1\right) = \frac{n^1\left(t_1 = 1\right)}{n\left(t_1 = 1\right) + 1} = \frac{15}{21} = 0.714$$

and the NPI upper probability, from Equation (3.2), is

$$\overline{P}\left(D_{101} = 1 | t_{101,1} = 1\right) = \frac{n^1\left(t_1 = 1\right) + 1}{n\left(t_1 = 1\right) + 1} = \frac{16}{21} = 0.762.$$

It is also possible to calculate these NPI lower and upper probabilities for the event that the person does not have the disease ($D_{101} = 0$), given that he/she has

tested positive ($t_1 = 1$), via the conjugacy property. The NPI lower and upper probabilities for this event are

$$\underline{P}\left(D_{101} = 0 | t_{101,1} = 1\right) = 1 - \overline{P}\left(D_{101} = 1 | t_{101,1} = 1\right) = \frac{n^0\left(t_1 = 1\right)}{n\left(t_1 = 1\right) + 1} = \frac{5}{21} = 0.238$$

$$\overline{P}\left(D_{101} = 0 | t_{101,1} = 1\right) = 1 - \underline{P}\left(D_{101} = 1 | t_{101,1} = 1\right) = \frac{n^0\left(t_1 = 1\right) + 1}{n\left(t_1 = 1\right) + 1} = \frac{6}{21} = 0.286.$$

Similarly we can calculate the NPI lower and upper probability for the event that this future person does not have the disease ($D_{101} = 0$) given that he/she has tested negative ($t_1 = 0$), these NPI lower and upper probabilities are

$$\underline{P}\left(D_{101} = 0 | t_{101,1} = 0\right) = 0.741$$

and

$$\overline{P}\left(D_{101} = 0 | t_{101,1} = 0\right) = 0.753.$$

By the conjugacy property, we can also calculate the NPI lower and upper probabilities for the event that this person has the disease given that he/she has tested negative, these NPI lower and upper probabilities are

$$\underline{P}\left(D_{101} = 1 | t_{101,1} = 0\right) = 0.247$$

and

$$\overline{P}\left(D_{101} = 1 | t_{101,1} = 0\right) = 0.259.$$

As mentioned earlier, these NPI lower and upper probabilities for such events directly report the strength of the evidence for the class state (disease status here) for future instances, based on the data. Thus, it is clear from the values of the NPI lower and upper probabilities that a future person with test result 1 has higher NPI lower and upper probabilities of being classified as diseased than a person with test result 0. Similarly, a person with test result 0 has higher NPI lower and upper

probabilities of being classified as non-diseased than a person with test result 1. This is also clear from the given data set that more people with test result 0 do not have the disease, and more people with test result 1 have the disease. Therefore, the correctness of the predictive inference on this future person can be considered based on success of the test results for the n people in the data set.

□

To build a classification tree, we are interested in the best splitting attribute variables to select at each node. Of course, we aim at the most informative attribute variables. In our Direct NPI classification, if the binary attribute variable is informative in both values, then it gives high NPI lower and upper probabilities for such events. Thus, we can use the above NPI lower and upper probabilities to decide on which attribute variable (test, in Example 3.2.1) we place at each node of a classification tree.

We can calculate the above NPI lower and upper probabilities for such events for all binary attribute variables and then determine which one is the best. In a classification tree, we would have to decide which attribute variables to place on the root node and at the child nodes. So, we have to use the above NPI lower and upper probabilities in order to make this decision. In Section 3.3.1, we explain in more detail our split criterion which is based on the NPI lower and upper probabilities for events with binary data. In classification trees, we must also decide whether or not including other attribute variables beyond the root node is useful. More details about these decisions are given in Section 3.3.2.

After this brief introduction to Direct NPI classification for binary data, the next step is to show how we build a classification tree based on these NPI lower and upper probabilities. One of the most important issues in classification trees is how to choose the best attribute variable to split on, i.e. the split criterion. We introduced some of the most commonly used split criteria in the literature in Section 2.2.2. In this thesis we propose a new split criterion to use with binary data, which we call *Correct Indication* (CI). In Section 3.3.1, we will introduce this split criterion for

use with binary data. However, we will generalise the *Correct Indication* (CI) split criterion to multinomial data in Chapter 4, where attribute variables may have more than two categories (see Section 4.3). In Section 3.3, we explain how we can use the the NPI lower and upper probabilities for binary data directly to build classification trees.

3.3 Direct NPI classification trees

In this section, we explain how the Direct NPI classification method can be used to build classification trees. A classification tree is a simple method that can be used to predict a class for a new instance based on its attribute values. The general classification tree structure and some other details about classification trees have been discussed in Section 2.2. In this section, we discuss building a classification tree from Direct NPI classification perspective. The most important step to build a classification tree is the split criterion. Several split criteria have been introduced in the literature, we discussed three of the most commonly used classic split criteria, and one recently introduced split criterion which is based on imprecise probability, in Section 2.2.2. To build a Direct NPI classification tree, we use a new split criterion, which we call Correct Indication (CI). The CI method is completely based on the NPI approach. In this chapter, we introduce the CI split criterion for use with binary data, while the CI split criterion is also introduced for use with multinomial data in Chapter 4.

This section is organised as follows: Section 3.3.1 introduces the split criterion CI for binary data. Section 3.3.2 describes the building process of the Direct NPI classification trees and in Section 3.3.3 an example is presented.

3.3.1 Correct Indication (CI)

In this section we introduce a novel split criterion to be used when building classification trees based on the Direct NPI classification method. The concept of CI is

used to decide on which attribute the data will be split. So, in order to select an attribute variable for each node of the classification tree, the NPI lower and upper probabilities for CI need to be calculated. After that we aim at the largest possible values for both the NPI lower and upper probabilities for CI. The CI reports the strength of the evidence that the attribute variables indicate, based on the data. In this section, we introduce the NPI lower and upper probabilities for CI corresponding to binary attribute variables.

Let $p = P(t_j = 1)$, hence, $P(t_j = 0) = 1 - p$. Using NPI for Bernoulli quantities [24], introduced in Section 2.4.2, we get

$$p \in \left[\frac{n(t_j = 1)}{n+1}, \frac{n(t_j = 1) + 1}{n+1} \right]. \tag{3.5}$$

Now we can determine the NPI lower probability for the event that attribute t_j leads to CI by taking the NPI lower probabilities for $(D_{n+1} = 0 | t_{n+1,j} = 0)$, and for $(D_{n+1} = 1 | t_{n+1,j} = 1)$, and p within the range given by (3.5) to minimise the weighted average,

$$\underline{P}_j(CI) = \min_p \left(\frac{n^0\left(t_j = 0\right)}{n\left(t_j = 0\right) + 1}(1 - p) + \frac{n^1\left(t_j = 1\right)}{n\left(t_j = 1\right) + 1} p \right). \tag{3.6}$$

This minimum is achieved for

$$p = \begin{cases} \dfrac{n(t_j = 1) + 1}{n+1} & \text{if} \quad \dfrac{n^0(t_j = 0)}{n(t_j = 0) + 1} \geq \dfrac{n^1(t_j = 1)}{n(t_j = 1) + 1}, \\ \dfrac{n(t_j = 1)}{n+1} & \text{otherwise.} \end{cases} \tag{3.7}$$

Similarly, the NPI upper probability for the event that attribute t_j leads to CI is given by

$$\overline{P}_j(CI) = \max_p \left(\frac{n^0\left(t_j = 0\right) + 1}{n\left(t_j = 0\right) + 1}(1 - p) + \frac{n^1\left(t_j = 1\right) + 1}{n\left(t_j = 1\right) + 1} p \right), \tag{3.8}$$

where p is such that

$$p = \begin{cases} \dfrac{n(t_j=1)+1}{n+1} & \text{if } \dfrac{n^0(t_j=0)+1}{n(t_j=0)+1} \leq \dfrac{n^1(t_j=1)+1}{n(t_j=1)+1}, \\ \dfrac{n(t_j=1)}{n+1} & \text{otherwise.} \end{cases} \tag{3.9}$$

These NPI lower and upper probabilities for CI should be calculated for each single attribute variable. We aim at the maximum probability for CI for both NPI lower and upper probabilities for a future instance. Generally, in a classification tree, the most informative attribute variable is desired. In CI for binary data, if the attribute variable is very informative in both cases, then it gives high NPI lower and upper probabilities for CI. For example, if all attribute values $t_j = 0$ indicate the class $D = 0$, and all attribute values $t_j = 1$ indicate the class $D = 1$, then this is an example which would give the highest possible NPI lower and upper probabilities for CI. Note that the CI given in this section is only applied to binary attribute variables. For the case of multinomial attribute variables, where attributes may have more than two categories, see Section 4.3. Throughout this thesis, we will use the term 'CI intervals' for the intervals which are created by the NPI lower and upper probabilities for CI for attribute variables, i.e. the NPI lower and upper probabilities for CI are considered as end points of the CI intervals. Example 3.3.1 illustrates the calculations of the CI split criterion, and how the NPI lower and upper probabilities for CI may indicate the best attribute variable to select when building a classification tree.

Example 3.3.1 Consider a data set with 100 instances, two binary attribute variables (t_1 and t_2) and a binary class variable (D). The value of each attribute variable is distributed between the two classes as illustrated by Table 3.2. The aim of this example is to show how the CI split criterion is calculated for different attribute variables, and how to identify the best attribute variable to use when building a classification tree.

Consider the attribute variable t_1. Using NPI for Bernoulli quantities [24], in-

D	t_1		t_2	
	1	0	1	0
1	65	2	30	25
0	5	28	20	25

Table 3.2: Data set description for Example 3.3.1.

troduced in Section 2.4.2, p is as follows

$$p \in \left[\frac{70}{101}, \frac{71}{101}\right].$$

The NPI lower and upper probabilities for CI for t_1 are

$$\underline{P}_{t_1}(CI) = \frac{28}{31}\left(1 - \frac{70}{101}\right) + \frac{65}{71}\left(\frac{70}{101}\right) = 0.9117$$

$$\overline{P}_{t_1}(CI) = \frac{29}{31}\left(1 - \frac{71}{101}\right) + \frac{66}{71}\left(\frac{71}{101}\right) = 0.9313.$$

Similarly, the NPI lower and upper probabilities for CI for t_2 are

$$\underline{P}_{t_2}(CI) = \frac{25}{51}\left(1 - \frac{50}{101}\right) + \frac{30}{51}\left(\frac{50}{101}\right) = 0.5387$$

$$\overline{P}_{t_2}(CI) = \frac{26}{51}\left(1 - \frac{51}{101}\right) + \frac{31}{51}\left(\frac{51}{101}\right) = 0.5593.$$

The values of the NPI lower and upper probabilities for CI for t_1 are greater than those values given by t_2. These values of the NPI lower and upper probabilities for CI reflect the strength of the evidence that each attribute variable indicates regarding the possible class states for future instances. Table 3.2 shows that instances with values $t_1 = 1$ have a higher proportion of being classified as $D = 1$ than being classified as $D = 0$, and instances with values $t_1 = 0$ have a higher proportion of being classified as $D = 0$ than being classified as $D = 1$. Therefore, this attribute variable has high NPI lower and upper probabilities for CI. In contrast, instances with values $t_2 = 0$ have the same proportion of being classified as $D = 1$ or $D = 0$,

and instances with values $t_2 = 1$ have similar proportions of being classified as $D = 1$ or $D = 0$. Thus, this attribute variable has not produced high NPI lower and upper probabilities for CI. Note that the highest possible lower and upper probabilities for CI are given by an attribute variable that perfectly classified the data. I.e. all instances with values 1 are classified as class 1 and all instances with values 0 are classified as class 0. The better indication about the possible class states for future instances of t_1 than of t_2 is reflected by larger values of the NPI lower and upper probabilities for CI for t_1 than those values for t_2. When building classification trees we aim for the most informative attribute variable to place at each node, and for this example one would select attribute variable t_1 as it gives higher NPI lower and upper probabilities for CI.

□

3.3.2 Building D-NPI classification trees

In this section, we describe how to build a Direct NPI classification tree using only binary data. The basic idea is simple and it is applied recursively to each of the nodes. The building process is similar to the well-known C4.5 algorithm (see Section 2.2), but we use the CI split criterion to select the best attribute variable at each node. Generally, the D-NPI classification tree recursively partitions the training data sets into smaller subsets, based on the most informative attribute variable which is selected by the CI split criterion. Note here that the most informative attribute variable is the one which has the largest NPI lower and upper probabilities for CI.

It is important to note that during the process of building D-NPI classification trees in this chapter, we assume that $\frac{n^1(t_j=1)}{n(t_j=1)} \geq \frac{n^1(t_j=0)}{n(t_j=0)}$, which means that attribute values are defined such that attribute value 1 (positive value) is related to the class state 1 (positive class state) in terms of the data set. This link between the attribute values and class states should be used at all stages of building the tree, hence, we may need to redefine the attribute values when working with subsets of the data, i.e. at different parts of the tree. Hence, further notation and attention

D	t_1		t_2	
	1	0	1	0
1	65	2	5	19
0	5	28	75	1

Table 3.3: Data set description for Example 3.3.2.

are required. Example 3.3.2 illustrates the case where an attribute variable needs to be re-labelled to satisfy the assumption $\frac{n^1(t_j=1)}{n(t_j=1)} \geq \frac{n^1(t_j=0)}{n(t_j=0)}$.

Example 3.3.2 Consider a data set with 100 instances, two binary attribute variables (t_1 and t_2) and a binary class variable (D). The value of each attribute variable is distributed between the two classes as illustrated by Table 3.3. The aim of this example is to illustrate when attribute variables need to be re-labelled when building the D-NPI classification trees.

For attribute variable t_1, the assumption $\frac{n^1(t_j=1)}{n(t_j=1)} \geq \frac{n^1(t_j=0)}{n(t_j=0)}$ is satisfied, while attribute variable t_2 does not satisfy this assumption. Hence, we need to re-label the attribute variable t_2 as follows: $t_2 = 1$ as $t_2 = 0$ and $t_2 = 0$ as $t_2 = 1$. Then, the attribute variable t_2 will satisfy the above assumption, and will be ready to use when building the D-NPI classification trees.

□

Suppose that we have a training data set, $\mathcal{D}$, which has a binary target variable and binary attribute variables, t_j, where $j = 1, ..., T$. The method starts with a tree with a root node, then we go through interior nodes finally arriving at the leaves which describe a possible class state. In our method, each non-leaf node will have an attribute variable with two possible values, say $t_j = 0$ or $t_j = 1$.

First, we calculate the *Correct Indication* (CI) intervals for the complete list of attribute variables t_j, using the NPI lower and upper probabilities for CI given in Equations (3.6) and (3.8). Then, we compare the values of the CI intervals for the full training set with the interval given by the NPI lower and upper probabilities for

CI if no attribute variable is used, which are defined in the following paragraph. In this thesis, we refer to the NPI lower and upper probabilities for CI if no attribute variable is used by the NPI lower and upper probabilities for NA, where NA is the event of CI in case no attribute variable is used.

Let $\underline{P}(NA)$ indicate the NPI lower probability for CI if no attribute variable is used, and $\overline{P}(NA)$ indicate the NPI upper probability for CI if no attribute variable is used. The NPI lower and upper probabilities for CI if no attribute variable is used, $\underline{P}(NA)$ and $\overline{P}(NA)$, correspond to simply stating the most common value in the target variable. Using the NPI method for Bernoulli quantities [24], introduced in Section 2.4.2, see also Equations (2.16) and (2.17), the NPI lower and upper probabilities for CI if no attribute variable is used are

$$[\underline{P}(NA), \overline{P}(NA)] = \left[\frac{s}{n+1}, \frac{s+1}{n+1}\right]$$

where s is the number of positive cases in n instances, or the larger value in the target variable. When the number of positive and negative cases is equal, we choose any of them to calculate the NPI lower and upper probabilities for CI if no attribute variable is used. For example, suppose we have a class variable with 70 instances in class state 0, and 30 instances in class state 1, where all instances are equal to 100, then the NPI lower and upper probabilities for CI if no attribute variable is used are

$$\underline{P}(NA) = \frac{s}{n+1} = \frac{70}{101}$$

and

$$\overline{P}(NA) = \frac{s+1}{n+1} = \frac{71}{101}.$$

The probability interval for CI if no attribute variable is used measures the weighted average of the most common value in the target variable at the first split step. Then it does the same with the following splits when building the classification tree, where these splits are based on the chosen attribute variable at the root node.

After that, for each attribute variable, t_j, we consider whether the NPI lower and upper probabilities for CI for this attribute are greater than the NPI lower and upper probabilities for NA, respectively. I.e. the NPI lower probability for CI is greater than the NPI lower probability for NA, and the same for the upper probabilities. If this is the case, then we choose the attribute variable with the highest CI interval as a root node. This means that we consider two conditions in order to split upon the attribute variable with the highest CI values, which are

$$\underline{P}(CI_{j^*}) > \underline{P}(NA) \quad \text{and} \quad \overline{P}(CI_{j^*}) > \overline{P}(NA) \tag{3.10}$$

where the $\underline{P}(CI_{j^*})$ and $\overline{P}(CI_{j^*})$ correspond to the attribute variable with the highest NPI lower and upper probabilities for CI, and $\underline{P}(NA)$ and $\overline{P}(NA)$ correspond to the highest class state of the target variable. So, j^* indicates the attribute variable that gives the maximum values for the NPI lower and upper probabilities for CI compared to the other attribute variables. If we have two or more attribute variables that all fulfil the two conditions in (3.10) but they overlap in their NPI lower and upper probabilities for CI, we choose the one with the highest NPI upper probability for CI. In the case of having two or more attribute variables that all have the same NPI upper probability for CI and the same NPI lower probability for CI giving that they all fulfil the two conditions in (3.10), we choose any of them to build the tree. If there is no attribute variable that fulfils the two conditions in inequalities (3.10), we do not split further and transform the node into a leaf with the most common class in the target variable.

The two conditions in (3.10) are used as a stop criterion when building the D-NPI classification trees. In order to split further when building the D-NPI classification trees, we need to choose attribute variables that produce more information with regard to predicting the possible class state. The NPI lower and upper probabilities for NA are achieved directly from the class variable, so information comes form the class variable only. Therefore, we look for an attribute variable that contains more information than the NPI lower and upper probabilities for NA. An attribute variable that has higher values for the NPI lower and upper probabilities for CI should have more information about predicting the possible class state than only consider-

ing the NPI lower and upper probabilities for NA. This stop criterion could prevent overfitting in the D-NPI classification trees, as it prevents us from building larger trees that may over fit the data and has less classification accuracy on the testing set.

However, studying this stop criterion in more detail remains as a future research topic. For example, using the imprecision (i.e. the difference between the NPI lower and upper probabilities for CI, and for NA) along with the two conditions in (3.10) particularly when there is overlap between two or more attribute variables that all fulfil these conditions or in other cases when one condition of (3.10) is met but not the another one. Such study and more investigations about this stop criterion might lead to a better conclusion about using them when building the D-NPI classification trees.

After selecting the best attribute variable t_{j^*} at the root node, based on CI, we split the training data set $\mathcal{D}$ into two subsets $\mathcal{D}_{j^*=0}$ and $\mathcal{D}_{j^*=1}$, where $\mathcal{D}_{j^*=0} \bigcup D_{j^*=1} = \mathcal{D}$ and $\mathcal{D}_{j^*=0} \bigcap \mathcal{D}_{j^*=1} = \emptyset$. After this stage, we calculate the CI intervals for each subset and then compare the values of each subset with the corresponding probability interval for NA following the first chosen attribute variable. That means comparing the intervals of CI for the first subset $\mathcal{D}_{j^*=0}$ with the probability interval for NA corresponding to $\mathcal{D}_{j^*=0}$, and comparing the intervals of CI for the second subset $\mathcal{D}_{j^*=1}$ with the probability interval for NA corresponding to $\mathcal{D}_{j^*=1}$. If there is no attribute variable with a greater CI interval than the CI interval for NA, i.e. no attribute variable that fulfils the two conditions in (3.10), then we do not split further and fix a leaf with the most common class in the target variable. Otherwise, we choose the attribute variable with the highest CI interval to split on. A node will be designated as a leaf node only if we do not have any attribute variable that fulfils the two conditions in (3.10). Finally, the above process can be represented as a classification tree. Algorithm 1 describes the D-NPI classification tree algorithm, named D-NPI algorithm. In Section 3.3.3, we apply the D-NPI classification method to a data set in order to illustrate the construction of the D-NPI classification tree.

Algorithm 1 Pseudocode of the D-NPI algorithm.

Input:
`TR`: Training data set
`Target`: Target variable
`Attr`: List of attribute variables
procedure D-NPI(`TR`, `Target`, `Attr`)
Create a Root node for the tree
    if `TR` have the same class C, **then**
        Return the single-node tree with class C
    if Attr is empty, **then**
        Return the single-node tree with the most common class C in `TR` (*)
    Otherwise
    for each attribute, t in `Attr` **do**
        Compute $\underline{P}_t(CI)$ and $\overline{P}_t(CI)$
        Choose attribute, t with highest $\underline{P}_t(CI)$ and $\overline{P}_t(CI)$
        if $\underline{P}_t(CI) > \underline{P}(NA)$ and $\overline{P}_t(CI) > \overline{P}(NA)$ **then**
            Choose the attribute, t (*)
        else
            Add a leaf node labelled with the most common class in `TR` (*)
    Set t the attribute for Root
    for each value of t, vi, **do**
        Add a branch below Root, corresponding to $t = v_i$
        Let $\texttt{TR}_{v_i}$ be the subset of `TR` that have $t = v_i$
        if $\texttt{TR}_{v_i}$ is empty, **then**
            Add a leaf node labelled with the most common class in `TR`
        else
            Add the subset generated by D-NPI($\texttt{TR}_{v_i}$, `Target`, `Attr`-$\{t\}$)
return Root
    (*): In case of ties, see descriptions in Section 3.3.2.

Attribute	Attribute Description	Attribute type
a1	Temperature of patient	continuous
a2	Occurrence of nausea	binary
a3	Lumbar pain	binary
a4	Urine pushing (continuous need for urination)	binary
a5	Micturition pains	binary
a6	Burning of urethra, itch, swelling of urethra outlet	binary
d1	Inflammation of urinary bladder	binary
d2	Nephritis of renal pelvis origin	binary

Table 3.4: Attribute information of Acute Inflammations data set.

3.3.3 Example of the D-NPI classification tree

In this section, we illustrate how to build a classification tree using the Direct NPI method. The data set used in this section is the Acute Inflammations data set, which is extracted from the UCI Machine Learning Repository [32]. This data set has been created by a medical expert as a data set to test an expert system, which will perform the presumptive diagnosis of two diseases of the urinary system. This data set consists of 120 instances, 6 attributes variables, and two target variables. There are no missing values in this data set. A brief description of this data set is given in Table 3.4.

All binary attribute variables have values of 'yes' or 'no' and the continuous variable (temperature) has values between 35.5 and 41.5. As we mentioned earlier, at this stage of developing the D-NPI classification method, we consider only binary attribute variables. Therefore, a threshold of 37.95 has been taken for the continuous variable to convert it into a binary variable ('Low' or 'High'). This threshold is the same as the threshold given by the Information Gain Ratio split criterion [56].

Note that this data set has two target variables, named d1 and d2, each denoting a particular disease. In this example, we build two different classification trees, each is based on one of the target variables. The tree building process is explained for one tree, while the another tree is built following the same procedures.

Attribute	$\underline{P}(CI)$	$\overline{P}(CI)$
a1	0.663	0.684
a2	0.591	0.612
a3	0.724	0.745
a4	0.804	0.828
a5	0.826	0.847
a6	0.571	0.592

Table 3.5: The intervals for CI for all attribute variables.

The Acute Inflammations data set has been divided into training and testing data sets, with the training set and testing set containing about 80% and about 20% of all data, respectively. The D-NPI classification trees for both target variables have been constructed on the same training data. Note that we apply this binary splitting for the training and testing data sets only here for the sake of illustrating the construction of the D-NPI classification tree. However, we apply the 10-fold cross validation scheme, introduced in Section 2.2.1, in order to assess the performance of the D-NPI classification trees and compare it with other classification methods (see Section 3.4). Thus, the 80%- 20% splitting method is only applied here to illustrate our method, and is not considered in the experimental analysis.

First, we consider the target variable d1. Using Equations (3.6) and (3.8), we calculate the NPI lower and upper probabilities for CI for all attribute variables. Table 3.5 shows the NPI lower and upper probabilities for CI for all attribute variables, considering d1 as the target variable. Now, we compare the intervals in Table 3.5 with the interval generated by the NPI lower and upper probabilities for NA. At this step, no further attribute means no attribute at all. We know from the training set that the most common value in the target variable is 'not having the disease' with 51 instances out of 96 instances. Therefore, the probability interval for NA corresponds to indicating that 'no disease' is most likely. Using the NPI for Bernoulli data [24], introduced in Section 2.4.2, the NPI lower and upper probabilities for NA are

$$\underline{P}(NA) = \frac{51}{97} = 0.526$$

Attribute	$\underline{P}(CI)$	$\overline{P}(CI)$
a1	0.648	0.699
a2	-	-
a3	0.765	0.815
a4	0.628	0.680
a6	0.495	0.548

Table 3.6: CI intervals for data set $\mathcal{D}_{\texttt{a5=0}}$.

Attribute	$\underline{P}(CI)$	$\overline{P}(CI)$
a1	0.639	0.694
a2	0.639	0.694
a3	0.639	0.694
a4	0.958	1
a6	0.618	0.673

Table 3.7: CI intervals for data set $\mathcal{D}_{\texttt{a5=1}}$.

and

$$\overline{P}(NA) = \frac{52}{97} = 0.536.$$

Now, we simply compare these NPI lower and upper probabilities, $[0.526, 0.536]$, with the NPI lower and upper probabilities for CI, given in Table 3.5, and then we choose the attribute variable with the highest CI interval, which is a5 (Micturition pains) to be assigned to the root node. Note that the two conditions in (3.10) have to be met. Thus, we split the data set according to the value of a5.

As the next step, the training data set $\mathcal{D}$ will be divided into two subsets $\mathcal{D}_{\texttt{a5=0}}$ and $\mathcal{D}_{\texttt{a5=1}}$, where $\mathcal{D}_{\texttt{a5=0}} \bigcup \mathcal{D}_{\texttt{a5=1}} = \mathcal{D}$ and $\mathcal{D}_{\texttt{a5=0}} \bigcap \mathcal{D}_{\texttt{a5=1}} = \emptyset$. $\mathcal{D}_{\texttt{a5=0}}$ contains 50 patients, 43 of them do not have the disease, and $\mathcal{D}_{\texttt{a5=1}}$ contains 46 patient, 8 of them have the disease. We now need to calculate the intervals for CI for each subset, these intervals are given in Table 3.6 and Table 3.7, respectively. Note that we do not calculate the CI interval for the attribute variable a2 with $\mathcal{D}_{\texttt{a5=0}}$ as all instances in a2 have the same value.

After that, we compare the CI intervals in Table 3.6 with the corresponding probability interval for NA following a5 with values 'no' ($\mathcal{D}_{\texttt{a5=0}}$), which is

$$[\underline{P}(NA), \overline{P}(NA)] = \left[\frac{43}{51}, \frac{44}{51}\right] = [0.843, 0.863].$$

Similarly, we compare the CI intervals in Table 3.7 with the corresponding probability interval for NA following a5 with values 'yes' ($\mathcal{D}_{\texttt{a5=1}}$), which is

$$[\underline{P}(NA), \overline{P}(NA)] = \left[\frac{38}{47}, \frac{39}{47}\right] = [0.809, 0.830]\,.$$

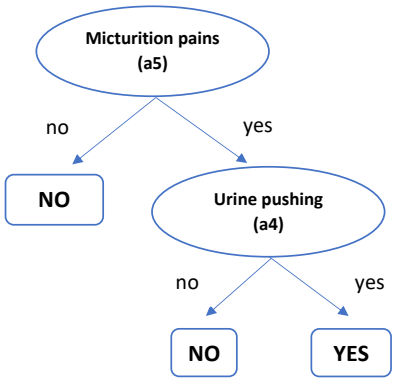

Figure 3.1: Classification Tree of Inflammation of urinary bladder.

Table 3.6 shows that there is no attribute variable that has higher CI interval than the corresponding NPI lower and upper probabilities for NA. Therefore, no splitting will be considered for this branch and we fix a leaf node with the most common value in the target variable, which is not having disease. For the set $\mathcal{D}_{\texttt{a5}=1}$, `a4` has the highest CI interval, which is greater than the corresponding NPI lower and upper probabilities for NA. Hence, we choose `a4` as a second splitting variable.

After choosing `a4` to split on for the data set $\mathcal{D}_{\texttt{a5}=1}$, the data set $\mathcal{D}_{\texttt{a5}=1}$ must be divided into two subsets, as follows: $\mathcal{D}_{\texttt{a5}=1,\texttt{a4}=0}$, for which patients have Micturition pains (`a5`) but do not have Urine pushing (`a4`). $\mathcal{D}_{\texttt{a5}=1,\texttt{a4}=1}$, for which patients have Micturition pains (`a5`) and also have Urine pushing (`a4`). $D_{\texttt{a5}=1,\texttt{a4}=0}$ contains 8 patients, all of them were found not to have the disease and $D_{\texttt{a5}=1,\texttt{a4}=1}$ contains 38 patients, all of them were found to have the disease. In this case, the intervals for CI are not needed, as both sets are completely accurate (pure sets). Thus, a leaf node is fixed for each branch with the corresponding class. Finally, the above information can be represented as a classification tree. Figure 3.1 shows this D-NPI classification tree, where the target variable is about predicting inflammation of the urinary bladder.

Secondly, by considering the second target variable `d2` (Nephritis of renal pelvis

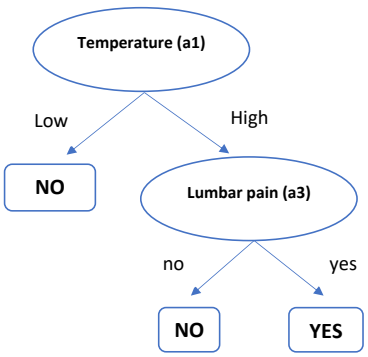

Figure 3.2: Classification Tree of Nephritis of renal pelvis origin.

origin) and following the same procedures, the D-NPI classification tree has been constructed, it is shown in Figure 3.2.

This example illustrates how to build D-NPI classification trees. Our aim in this example is only to illustrate the method of building the D-NPI classification tree. Further analysis of this data set and evaluating the performance of the D-NPI classification trees compared to other classification trees is presented in Section 3.4.

3.4 Performance of the D-NPI method

In this section, we carry out experiments on five data sets in order to assess the performance of the D-NPI algorithm when building classification trees. We also compare its performance with classical and imprecise algorithms. We compare the D-NPI algorithm with the most commonly used classical algorithm to build classification trees, which is the C4.5 algorithm (see Section 2.2.2 for more details about this algorithm). We also compare the D-NPI algorithm with three imprecise algorithms which are the NPI-M, A-NPI-M and IDM algorithms (see Sections 2.2.2, 2.3 and 2.5 for more details about these algorithms). For the IDM algorithm, we use two recommended values for the parameter $\tilde{s}$, 1 and 2. We refer to the IDM algorithm with $\tilde{s} = 1$ by IDM1 and the IDM algorithm with $\tilde{s} = 2$ by IDM2.

Data set	N	Attr	# Class 1	# Class 2
Acute Inflammations 1	120	6	61	59
Acute Inflammations 2	120	6	70	50
Banknote authentication	1372	4	762	610
Breast Cancer Wisconsin	699	9	458	241
Congressional Voting Records	435	16	267	168

Table 3.8: Data sets description.

As a first step of developing the D-NPI algorithm, we apply it to only binary data, so we need data sets that have both attribute and target variables with only binary values. As such data sets are not commonly available, we have used only five data sets and we have converted some continuous attribute variables to binary ones. The five data sets used in this experimental analysis are obtained from the UCI Machine Learning Repository database [32]. Table 3.8 shows a summary of the main characteristics of these data sets, where 'N' is the number of instances in the data sets, 'Attr' is the number of attribute variables, '# Class 1' is the number of instances in the first class and '# Class 2' is the number of instances in the second class. All these data sets have only two classes for the target variable. Note that the Acute Inflammations data set has two target variables, based on each we construct a tree. Thus, we consider it as two separate data sets, each one with a different target variable. More details about these data sets can be found in [32].

We have applied the following pre-processing method: continuous attributes have been converted to binary attributes using the optimal threshold selected by Information Gain Ratio split criterion [56]. We then built all classifiers on the data sets with binary attribute variables. Further, missing values have been replaced with modal values.

For each data set, the D-NPI classification trees were built using the procedure explained in Section 3.3.2. Then, classical classification trees were built for each data set using the C4.5 algorithm. We used the RWeka package [39, 67] to build

	Class 1 (Predicted)	Class 2 (Predicted)
Class 1 (Actual)	TN	FP
Class 2 (Actual)	FN	TP

Table 3.9: A sample confusion matrix.

Data set	D-NPI	C4.5	NPI-M	A-NPI-M	IDM1	IDM2
Acute Inflammations 1	94.17	**100**	**100**	**100**	**100**	**100**
Acute Inflammations 2	**100**	**100**	**100**	**100**	**100**	**100**
Banknote authentication	**89.50**	89.49	89.49	89.49	89.49	89.49
Breast Cancer Wisconsin	**94.85**	94.27	93.19	93.19	93.19	93.19
Congressional Voting Records	**95.64**	95.58	95.58	95.58	95.58	95.35

Table 3.10: Accuracy results of all classification algorithms.

the C4.5 algorithm. We also build classification trees based on different imprecise algorithms, such as the NPI-M, A-NPI-M and IDM algorithms, using the imptree package [34]. A 10-fold cross validation scheme has been applied for each data set, and then the average results have been reported. More details about the 10-fold cross validation have been given in Section 2.2.1. The experiments are run using the statistical software R [57].

We used classification accuracy rates to measure and compare the performance of all classification algorithms. Classification accuracy is the most commonly used method to measure the performance of classification algorithms. It is calculated as the ratio of the total number of correctly classified instances on the testing set to the total number of instances. Given a sample confusion matrix as in Table 3.9, the classification accuracy is

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{N}},$$

where N = TN + TP + FN + FP, and TN, TP, FN and FP denote true negatives, true positives, false negatives and false positives, respectively.

Table 3.10 shows the average results of classification accuracy rates of all classification algorithms. It can be noticed that the D-NPI algorithm slightly outperforms

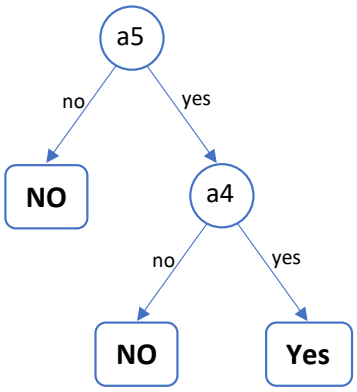

Figure 3.3: The resulting tree generated by the D-NPI algorithm for Acute Inflammations 1 data set.

other classification algorithms in four out of five data sets. However, for the Acute Inflammations 1 data set, the D-NPI algorithm does not achieve the full classification accuracy rate demonstrated by the other classification algorithms, but the D-NPI algorithm produces relatively smaller trees than the other algorithms for this data set. For more clarification, the D-NPI algorithm builds some trees for this data set as shown in Figure 3.3, while other classification algorithms such as the C4.5, NPI-M, A-NPI-M, and IDM algorithms builds relatively larger trees for this data set as shown in Figure 3.4. Note that this data set is created by a medical expert as a data set to test the expert system, which will perform the presumptive diagnosis of two diseases of the urinary system, hence it is not a strange situation to have a full classification accuracy rate by different classification algorithms. Some other researchers have also used this data set and have got a full classification accuracy rate, see Kadhem and Zeki [41] and Medjahed et al. [49], but Chandra and Bhaskar [22] and by Bertsimas and Dunn [16] have reported lower classification accuracy for this data set.

For the Banknote authentication, Breast Cancer Wisconsin and Congressional Voting Records data sets, the D-NPI classification algorithm has a slightly better performance in the classification accuracy rate than the other algorithms, although

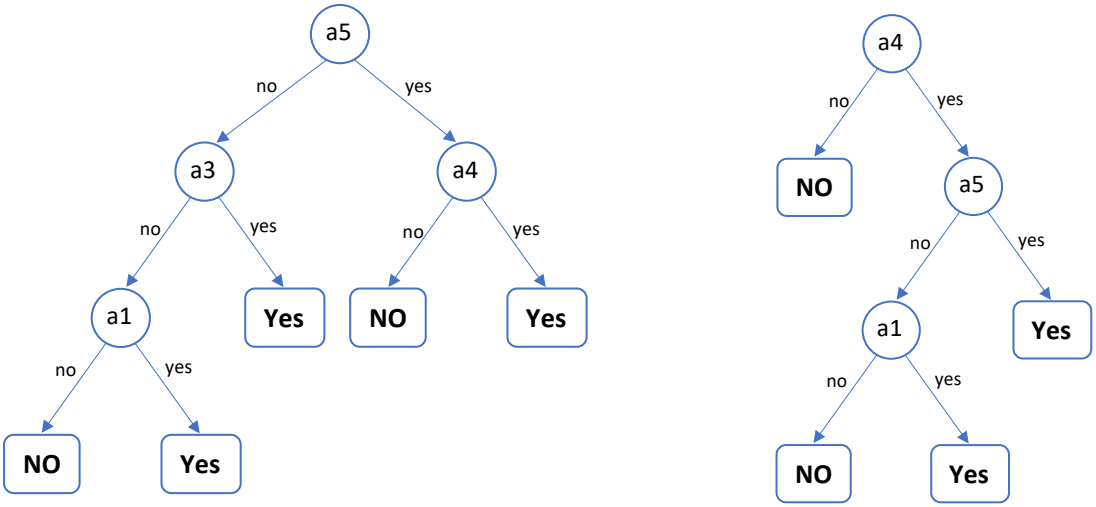

Figure 3.4: The resulting tree generated by the NPI-M and A-NPI-M algorithms (left) and the resulting tree generated by the C4.5 and IDM algorithms (right) both trees for Acute Inflammations 1 data set.

all classification algorithms have a very similar classification accuracy results. In Section 3.3.2, we have stated that there are two conditions (see the inequalities in (3.10)) which have to be fulfilled in order to split further when building the D-NPI classification trees. During the experimental analysis, we noticed that using these two conditions as a stop criterion leads to a better performance of the D-NPI classification algorithm.

To delve further into the analysis of the D-NPI algorithm and compare it with other classification algorithms, we calculate in-sample accuracy rates for all classification algorithms. In-sample accuracy measures the performance of the classification algorithms on the training data set [16, 53]. The in-sample accuracy measure is not commonly used to indicate classification accuracy, but it gives insight into how the algorithm performs on the training set. It is known that if the classification algorithm performs very well on the training set but not very well on the testing set, this is likely to indicate overfitting. Thus, the in-sample accuracy is reported to show the performance of classification algorithms on the training set, and hence, to

Data set	D-NPI	C4.5	NPI-M	A-NPI-M	IDM1	IDM2
Acute Inflammations 1	94.17	**100**	**100**	**100**	**100**	**100**
Acute Inflammations 2	**100**	**100**	**100**	**100**	**100**	**100**
Banknote authentication	**89.51**	**89.51**	**89.51**	**89.51**	**89.51**	**89.51**
Breast Cancer Wisconsin	**95.31**	94.20	93.67	93.67	93.84	93.67
Congressional Voting Records	95.63	**95.64**	**95.64**	**95.64**	**95.64**	**95.64**

Table 3.11: In-sample accuracy results of all classification algorithms.

Algorithm	D-NPI	C4.5	NPI-M	A-NPI-M	IDM1	IDM2
Average	3.2	**3**	4.6	4.6	4.5	4.2

Table 3.12: Average tree size for all classification algorithms.

check on overfitting as well. Table 3.11 illustrates the average in-sample accuracy of all classification algorithms. The results of in-sample accuracy are quite similar in all data sets except Acute Inflammations 1 where the D-NPI algorithm achieves the lowest in-sample accuracy, but with smaller trees generated for this data set.

Finally, in order to compare different trees generated by the classification algorithms, an average tree size of each algorithm is reported. Table 3.12 presents the average results of tree size (number of leaves) for each classification algorithm. Note that we refer to tree size as the total number of leaf nodes, as was done by Bertsimas and Dunn [16], and by Murthy and Salzberg [53]. However, the total number of all nodes can be also considered as a tree size. The C4.5 classification algorithm and the D-NPI classification algorithm generate the smallest trees with average tree size of 3 and 3.2, respectively. However, the D-NPI classification algorithm mostly builds smaller trees than the other algorithms, but in Breast Cancer Wisconsin, the C4.5 algorithm has the smallest tree size compared to the other algorithms, and hence, it gives the smallest average tree size. I.e. the D-NPI classification algorithm builds the same size of trees or even smaller than the C4.5 algorithm in the remaining four data sets. Compared to the NPI-M, A-NPI-M, IDM1 and IDM2 algorithms, our study shows that the D-NPI algorithm tends to generate smaller trees.

In this experimental analysis, we found that the D-NPI classification algorithm

performs well and slightly outperforms the other classification algorithms with respect to classification accuracy, in-sample accuracy and tree size. It is also noticed that the NPI-M algorithm and the A-NPI-M algorithm perform the same for all these data sets. The IDM1 algorithm slightly outperforms the IDM2 algorithm, but with relatively larger trees. This chapter is restricted to binary data, where all inputs for both attribute and class variable are binary. Thus, the conclusions drawn here are based on this type of data. In Chapter 4, we generalise the D-NPI algorithm to multinomial data sets, and hence, we analyse more data sets and draw a clearer conclusion about the performance of the D-NPI algorithm. We will also compare and evaluate the performance of the D-NPI classification algorithm compared to other classical and imprecise algorithms.

3.5 Concluding remarks

In this chapter we have presented a new method for building classification trees based on NPI, which is Direct NPI classification (D-NPI). It will be of interest to explore the use of the D-NPI classification method in random forests. The NPI approach could be well suited to be applied to other classification methods such as nonparametric classification methods, as it is a predictive method and the nature of classification is explicitly predictive as well.

As a first step for developing D-NPI classification trees, this chapter considered only binary data, where both attribute and class variables have only two values. Direct NPI classification trees have been built using a new split criterion, which we call Correct Indication (CI). The CI split criterion is completely based on the NPI lower and upper probabilities for binary data. It does not use any additional concepts or assumptions. We have carried out an experimental analysis in order to assess the performance of the D-NPI classification method, and to compare it with other classical and imprecise classification methods. The experimental results have suggested that the D-NPI classification algorithm slightly outperforms the other classification algorithms in terms of both classification accuracy and in-sample ac-

curacy. The results have also indicated that the D-NPI classification algorithm produces relatively smaller trees than the ones produced by imprecise algorithms, but slightly larger trees than the ones generated by the C4.5 algorithm. Applying the D-NPI classification algorithm to more data sets with more detailed analysis could lead to more conclusions about the performance of the D-NPI classification algorithm, and when it might be best used.

In Chapter 4, we generalize the D-NPI classification method to multinomial data. It would be of interest to extend this work further to involve real-valued data. Another interesting extension to this work is to develop the D-NPI classification tree with imprecise classification. In imprecise classification, trees might return a set of states rather than a single state in the class variable. It is also interesting to consider developing the D-NPI classification method with taking the cost of misclassification into account. In this consideration, we aim to minimize the total misclassification costs instead of trying to optimize the total accuracy rate [59]. Finally, exploring the use of imprecision as a stop criterion may lead to improvements on the D-NPI classification algorithm. For example, by considering the difference between the NPI lower and upper probabilities for CI, and for NA, especially, when there is overlap between two or more attribute variables that all fulfil the two conditions considered for splitting.

Chapter 4

Direct NPI Classification for Multinomial Data

4.1 Introduction

In Chapter 3, we have introduced the D-NPI classification method for binary data. In this chapter we generalise the D-NPI classification method to multinomial data. Throughout this chapter, we assume that there is a known number of categories, with no ordering of the categories. We also assume that there is at least one observation in each category. In this chapter, we restrict our focus to three or more different categories which are all observed. However, the classification algorithm in this chapter can be also used for two categories, and will lead to similar results to the classification algorithm introduced in Chapter 3. Note that the Direct NPI classification method for binary data, which has been introduced in Chapter 3 is not a special case of the method introduced in this chapter when we have two categories.

In this chapter we introduce the Direct NPI classification algorithm for multinomial data (D-NPI-M), which uses Correct Indication (CI) as a split criterion to build classification trees. The CI used in this chapter is based on the NPI lower and upper probabilities for multinomial data. We explain how to compute the NPI lower and upper probabilities for CI. We carry out experimental analysis on eight data sets in order to examine the performance of the D-NPI-M algorithm when

building classification trees. We also compare the D-NPI-M algorithm to a classical classification tree algorithm which is the C4.5 algorithm. We also compare the D-NPI-M algorithm to imprecise methods which are the NPI-M, A-NPI-M and IDM algorithms.

This chapter is organized as follows: Section 4.2 introduces the idea of Direct NPI classification for multinomial data with an illustrative example. In Section 4.3, we explain the split criterion, Correct Indication (CI) for multinomial data and we derive the NPI lower and upper probabilities for the CI. Section 4.4 presents a new algorithm to build classification trees with multinomial data, which we denote as the D-NPI-M algorithm. In Section 4.5, we assess the performance of the D-NPI-M algorithm and compare it with some other classification algorithms. Section 4.6 provides some more discussions about different classification trees generated by the classification algorithms, based on a particular data set. Finally, some concluding remarks and topics for future research are given in Section 4.7.

4.2 Direct nonparametric predictive classification

In this section, we illustrate how we can base classification on the NPI lower and upper probabilities for events with multinomial data, and without adding any further assumptions. We will use similar notations as introduced in Section 3.2. Assume that we have a data set of n instances. Suppose that there are $T \geq 1$ attribute variables. Let t_j for $j = \{1, ..., T\}$ indicate these attribute variables, where each attribute variable can have a different number of categories. Let c_i represent the categories in attribute t_j for $i = 1, ..., k_j$. So, k_j is the number of categories in attribute t_j. Suppose also that we have a target variable with a known number of classes. We assume that the target variable is represented by a class variable $D \in \{C_1, ..., C_R\}$. In our Direct NPI classification method we assume that all categories have been observed in the data. Throughout this chapter we refer to target classes by C_r, and we denote attribute categories by c_i.

Let n^{C_r} be the number of instances which are classified as class r, for $r = 1, ..., R$, hence, $n = n^{C_1} + n^{C_2} + ... + n^{C_R}$. Let $n(t_j = c_i)$ be the total number of instances in the data set which have $t_j = c_i$. Let $n^{C_1}(t_j = c_i)$ be the number of instances which have $t_j = c_i$ and which are classified as C_1, so $n(t_j = c_i) = \sum_{r=1}^{R} n^{C_r}(t_j = c_i)$ for $r = 1, ..., R$.

Now we will see if attribute t_j is useful or not. Clearly, t_j is useful if there is a high probability that an instance with attribute value $t_j = c_1$ indeed has class C_1, so that $t_j = c_1$ is an indicator for C_1, and an instance with attribute value $t_j = c_2$ indeed has class C_2, so that $t_j = c_2$ is an indicator for C_2, and so on. First, we assume that attribute category c_1 is linked with class state C_1, attribute category c_2 is linked with class state C_2 and attribute category c_3 is linked with class state C_3. It is important to note that this assumption is only considered here to illustrate the main idea of the D-NPI classification method, with Example 4.2.1 to illustrate the calculations of the NPI lower and upper probabilities based on this assumption. However, in the experimental analysis in this chapter, we link each attribute category with the class state which is most frequently associated with it. Clearly, the number of attribute categories c_i might not be the same as the number of possible states in the class variable C_r. This case is discussed in Example 4.2.2. In this chapter, we consider the class state with the highest frequency among possible states corresponding to each attribute category. In other words, each attribute category is linked with the class state that contains the largest number of instances. So it is possible for multiple attribute categories to indicate the same class state. With this consideration, we focus on the indication that is given by each category with regard to predicting the possible class state. So, we are interested in the conditional events $(D = C_r | t_j = c_i)$.

We consider such events for one future instance for which the attributes values are known but which class status is unknown. Let D_{n+1} denote the unknown class status for a single future instance which is not included in the data. Using NPI for Bernoulli data [24], introduced in Section 2.4, we can provide the NPI lower

and upper probabilities for the event that a future instance, which is not included in the data set has class r, C_r given that its attribute value is c_i, $t_{n+1,j} = c_i$, for $i = 1, ..., k_j$, i.e. $D_{n+1} = C_r | t_{n+1,j} = c_i$, for $i = 1, ..., k_j$ and $r = 1, ..., R$. The NPI lower probability is

$$\underline{P}\left(D_{n+1} = C_r | t_{n+1,j} = c_i\right) = \frac{n^{C_r}\left(t_j = c_i\right)}{n\left(t_j = c_i\right) + 1}, \tag{4.1}$$

and the NPI upper probability is

$$\overline{P}\left(D_{n+1} = C_r | t_{n+1,j} = c_i\right) = \frac{n^{C_r}\left(t_j = c_i\right) + 1}{n\left(t_j = c_i\right) + 1}. \tag{4.2}$$

We could also use NPI for multinomial data for these lower and upper probabilities, but that would lead to slightly larger imprecision.

The Direct NPI classification for the conditional event $D_{n+1} = C_r | t_{n+1,j} = c_i$ can be calculated via Formulas (4.1) and (4.2). It should be noticed that the values of these NPI lower and upper probabilities will directly reflect the strength of the evidence with regard to the possible class state for the single future instance, based on the data. Example 4.2.1 illustrates the calculations of the above NPI lower and upper probabilities. We also illustrate in this example how attribute variables may reflect the possible class state for a future instance.

Example 4.2.1 Consider a multinomial data set containing information about 10 people. Suppose that we have 3 tests (attribute variables) performed on all people, each has 3 possible values. These tests are denoted by t_1, t_2 and t_3. Also suppose that we have a target variable (disease status), denoted by D, with 3 classes. Table 4.1 shows the categories of these tests and corresponding disease status.

We can calculate the NPI lower and upper probabilities for the event that a future person who is not included in the data set, but assumed to be exchangeable with the n people in the data set, has class C_1 ($D_{11} = C_1$) given that his/her test result for t_1 is c_1 ($t_{11,1} = c_1$). Based on the given data set and using Formulas (4.1) and (4.2), the NPI lower and upper probabilities are

Person	t_1	t_2	t_3	D	Person	t_1	t_2	t_3	D
1	c_1	c_1	c_1	C_1	6	c_1	c_1	c_1	C_1
2	c_1	c_1	c_1	C_1	7	c_1	c_1	c_2	C_1
3	c_2	c_1	c_1	C_2	8	c_2	c_2	c_2	C_2
4	c_3	c_1	c_1	C_3	9	c_3	c_3	c_3	C_3
5	c_3	c_3	c_2	C_3	10	c_1	c_3	c_3	C_1

Table 4.1: Data set description for Example 4.2.1.

$$\underline{P}\left(D_{11} = C_1 | t_{11,1} = c_1\right) = \frac{n^{C_1}\left(t_1 = c_1\right)}{n\left(t_1 = c_1\right) + 1} = \frac{5}{6}$$

$$\overline{P}\left(D_{11} = C_1 | t_{11,1} = c_1\right) = \frac{n^{C_1}\left(t_1 = c_1\right) + 1}{n\left(t_1 = c_1\right) + 1} = \frac{6}{6}.$$

Formally, for this inference this person's disease status is assumed to be exchangeable with those of the people in the data set who also had $t_1 = c_{11}$. We can also calculate the NPI lower and upper probabilities for the event that this person has class C_2 given that his/her test result for t_1 is c_2, which are

$$\underline{P}\left(D_{11} = C_2 | t_{11,1} = c_2\right) = \frac{n^{C_2}\left(t_1 = c_2\right)}{n\left(t_1 = c_2\right) + 1} = \frac{2}{3}$$

$$\overline{P}\left(D_{11} = C_2 | t_{11,1} = c_2\right) = \frac{n^{C_2}\left(t_1 = c_2\right) + 1}{n\left(t_1 = c_2\right) + 1} = \frac{3}{3}.$$

Similarly, the NPI lower and upper probabilities for the event that this person has class C_3 given that his/her test result for t_1 is c_3, are

$$\underline{P}\left(D_{11} = C_3 | t_{11,1} = c_3\right) = \frac{n^{C_3}\left(t_1 = c_3\right)}{n\left(t_1 = c_3\right) + 1} = \frac{3}{4}$$

$$\overline{P}\left(D_{11} = C_3 | t_{11,1} = c_3\right) = \frac{n^{C_3}\left(t_1 = c_3\right) + 1}{n\left(t_1 = c_3\right) + 1} = \frac{4}{4}.$$

Similarly, the NPI lower and upper probabilities for the same event, but based on results for t_2, are as follows

$$[\underline{P}, \overline{P}](D_{11} = C_1 | t_{11,2} = c_1) = \left[\frac{4}{7}, \frac{5}{7}\right],$$

$$[\underline{P}, \overline{P}](D_{11} = C_2 | t_{11,2} = c_2) = \left[\frac{1}{2}, \frac{2}{2}\right],$$

$$[\underline{P}, \overline{P}](D_{11} = C_3 | t_{11,2} = c_3) = \left[\frac{2}{4}, \frac{3}{4}\right].$$

Finally, we can also get the NPI lower and upper probabilities for the same event, but based on results for t_3, which are as follows

$$[\underline{P}, \overline{P}](D_{11} = C_1 | t_{11,3} = c_1) = \left[\frac{3}{6}, \frac{4}{6}\right],$$

$$[\underline{P}, \overline{P}](D_{11} = C_2 | t_{11,3} = c_2) = \left[\frac{1}{4}, \frac{2}{4}\right],$$

$$[\underline{P}, \overline{P}](D_{11} = C_3 | t_{11,3} = c_3) = \left[\frac{1}{3}, \frac{2}{3}\right].$$

It can be noticed that attribute category c_2 in t_3 is equally associated with classes C_1, C_2 and C_3, and attribute category c_3 in t_3 is equally associated with classes C_1 and C_3. We link c_2 in t_3 with class C_2 while linking it with C_1 or C_3 will lead to the same result. This is also applied to c_3 in t_3. However, we have mentioned earlier in this section that we link attribute category c_1 with class C_1, c_2 with C_2 and c_3 with C_3 to illustrate the main idea of the D-NPI classification method. Further consideration of different senarios of linking attribute categories with classes is given in Example 4.2.2.

The above values of the NPI lower and upper probabilities reflect the strength of the evidence that comes from the attribute variables with regard to predicting the possible class state for the target variable. A perfect test would always give the correct class state, while a noninformative test would provide no way to predict the class state. For example, it is clear from the data set that t_1 is the best test with regard to indicating the class state, as each value from t_1 indicates a correct class state, and hence, it has not misclassified any category (no wrong diagnoses in the data set). Hence, t_1 has the highest NPI lower and upper probabilities for predicting possible class states for the new person. In other words, t_1 provides more information than the other tests, and hence it is more valuable for predicting the class state. We can also notice that t_2 is better than t_3 as it produces larger values for the NPI lower and upper probabilities for the events of interest.

Person	t_1	t_2	D	Person	t_1	t_2	D
1	c_1	c_1	C_1	6	c_2	c_1	C_1
2	c_1	c_2	C_1	7	c_2	c_3	C_3
3	c_1	c_3	C_2	8	c_2	c_4	C_3
4	c_1	c_4	C_1	9	c_2	c_2	C_2
5	c_1	c_1	C_3	10	c_2	c_4	C_3

Table 4.2: Data set description for Example 4.2.2.

□

In Example 4.2.2 we illustrate how to calculate the above NPI lower and upper probabilities but in which not all attribute variables have the same number of categories.

Example 4.2.2 Consider a multinomial data set containing information about 10 people. Suppose that we have 2 tests (attribute variables) performed on all people, where test 1 has two categories and test 2 has four categories. These tests are denoted by t_1 and t_2. Also suppose that we have a target variable (disease status), denoted by D, with 3 classes. Table 4.2 shows the categories of both tests and corresponding disease status.

As the number of attribute categories is not the same as the number of target classes, we label where possible each attribute category to match the target class that is most frequently associated with it. So, we can calculate the NPI lower and upper probabilities for the event that a future person who is not included in the data set, but assumed to be exchangeable with the n people in the data set, has class C_1 ($D_{11} = C_1$) given that his/her test result for t_1 is c_1 ($t_{11,1} = c_1$). Based on the given data set and using Formulas (4.1) and (4.2), the NPI lower and upper probabilities are

$$\underline{P}\left(D_{11} = C_1 | t_{11,1} = c_1\right) = \frac{n^{C_1}\left(t_1 = c_1\right)}{n\left(t_1 = c_1\right) + 1} = \frac{3}{6}$$

$$\overline{P}\left(D_{11} = C_1 | t_{11,1} = c_1\right) = \frac{n^{C_1}\left(t_1 = c_1\right) + 1}{n\left(t_1 = c_1\right) + 1} = \frac{4}{6}.$$

We can also calculate the NPI lower and upper probabilities for the event that this person has class C_3 given that his/her test result for t_1 is c_2, which are

$$\underline{P}\left(D_{11} = C_3 | t_{11,1} = c_2\right) = \frac{n^{C_3}\left(t_1 = c_2\right)}{n\left(t_1 = c_2\right) + 1} = \frac{3}{6}$$

$$\overline{P}\left(D_{11} = C_3 | t_{11,1} = c_2\right) = \frac{n^{C_3}\left(t_1 = c_2\right) + 1}{n\left(t_1 = c_2\right) + 1} = \frac{4}{6}.$$

Note here that we associate the attribute category c_2 with the class state C_3 as this class state is the most frequently class that is associated with c_2.

Similarly, the NPI lower and upper probabilities for the same event, but based on results for t_2, are as follows

$$[\underline{P}, \overline{P}](D_{11} = C_1 | t_{11,2} = c_1) = \left[\frac{2}{4}, \frac{3}{4}\right],$$

$$[\underline{P}, \overline{P}](D_{11} = C_2 | t_{11,2} = c_2) = \left[\frac{1}{3}, \frac{2}{3}\right],$$

$$[\underline{P}, \overline{P}](D_{11} = C_3 | t_{11,2} = c_3) = \left[\frac{1}{3}, \frac{2}{3}\right],$$

$$[\underline{P}, \overline{P}](D_{11} = C_3 | t_{11,2} = c_4) = \left[\frac{2}{4}, \frac{3}{4}\right].$$

Note that in the case of having two or more classes that are equally associated with an attribute category, we choose any of them as all will lead to the same NPI lower and upper probabilities. For example, the attribute category c_2 has two classes (C_1 and C_2) that are both equally associated with it, hence, we randomly choose C_2 while choosing C_1 will lead to the same result.

□

In summary, the main idea of the D-NPI classification is illustrated in Example 4.2.1 assuming that each attribute category indicate a different class state. Then, we illustrate the D-NPI classification method when the number of attribute categories is not the same as the number of class states. Finally, for the experimental analysis in this chapter, we link each attribute category with the class state that is most

frequently associated with it.

Generally speaking, higher values for the NPI lower and upper probabilities for an attribute variable indicate that it is more informative with regard to reflecting the possible class state. To build classification trees, we have to decide which attribute should be placed at each node, starting from the root node to the leaves in the tree. We can make such decisions by using these NPI lower and upper probabilities. In classification trees we also need to decide whether or not adding more attribute variables to the tree is useful. For such decisions, a stopping criterion should be used. In Section 4.3, we introduce our new split criterion, which is used to build the Direct NPI classification trees.

4.3 Correct Indication with multinomial data

In this section, we generalise our split criterion, *Correct Indication* (CI), to multinomial data. The CI is introduced in Section 3.3.1 for binary data, where each attribute variable has only two possible values. Generally speaking, the most informative attribute variable is selected in classification trees for each split. For the CI split criterion, if the attribute is very informative in all cases, then it gives high NPI lower and upper probabilities for CI. Thus, we aim at the maximum values for both NPI lower and upper probabilities for CI. In this section, we generalise the CI formulas to attributes with a known number of categories, $k_j \geq 3$. Recall that k_j is the number of observed categories, labelled $c_1, ..., c_{k_j}$. In this chapter we assume that all possible categories have been observed as generally in classification unobserved categories are never more likely than observed categories. It may also possible to adapt our method to consider unobserved categories in future work.

Let $p_{j,i} = P(t_j = c_i)$ for $i = 1, ..., k_j$ and $j = 1, ..., T$, where for each attribute variable t_j, $\sum_{i=1}^{k_j} p_{j,i} = 1$. Using NPI for multinomial data [28], introduced in Section

2.4, we get

$$p_{j,i} \in \left[\frac{n_{ji}-1}{n}, \frac{n_{ji}+1}{n}\right] \tag{4.3}$$

where n_{ji} denotes the number of times we have observed category c_i for t_j.

We can determine the NPI lower probability for the event that attribute t_j leads to CI, by taking the NPI lower probabilities for the events $D_{n+1} = C_1|t_{n+1,j} = c_1, ..., D_{n+1} = C_R|t_{n+1,j} = c_{k_j}$, and $p_{j,i}$ within the range given by (4.3) to minimise the weighted average,

$$\underline{P}_j(CI) = \min_{p_{j,i}\in\mathcal{P}} \sum_{i=1}^{k_j} \frac{n^{C_r}\left(t_j = c_i\right)}{n\left(t_j = c_i\right)+1}\, p_{j,i} \tag{4.4}$$

where $\mathcal{P}$ is the set of probability distributions over the categories which correspond to the NPI lower and upper probabilities, and which is defined as follows:

$$\mathcal{P} = \left\{ p \mid \frac{n_{ji}-1}{n} \leq p_{j,i} \leq \frac{n_{ji}+1}{n}, \forall i = 1, ..., k_j, \sum_{i=1}^{k_j} p_{j,i} = 1 \right\}. \tag{4.5}$$

Similarly, the NPI upper probability for the event that attribute t_j leads to CI is

$$\overline{P}_j(CI) = \max_{p_{j,i}\in\mathcal{P}} \sum_{i=1}^{k_j} \frac{n^{C_r}\left(t_j = c_i\right)+1}{n\left(t_j = c_i\right)+1}\, p_{j,i}. \tag{4.6}$$

The above NPI lower and upper probabilities for CI should be calculated for each single attribute variable, then we choose the most informative attribute at each stage of building the classification tree based on these NPI lower and upper probabilities.

To compute the NPI lower and upper probabilities for CI, which are given by Equations (4.4) and (4.6), we consider all possible configurations δ on the probability wheel (see Section 2.4), applying the *circular-*$A_{(n)}$ assumption to each δ to get corresponding NPI lower and upper probabilities for CI ($\underline{P}_\delta(CI)$ and $\overline{P}_\delta(CI)$), and

then we take the NPI lower and upper probabilities with respect to the set $\mathcal{S}$ of all configurations δ such that

$$\underline{P}(CI) = \min_{\delta \in \mathcal{S}} \underline{P}_{\delta}(CI) \tag{4.7}$$

and

$$\overline{P}(CI) = \max_{\delta \in \mathcal{S}} \overline{P}_{\delta}(CI). \tag{4.8}$$

Next, we derive optimal configurations which lead to the NPI lower and upper probabilities for CI. We first consider the NPI lower probability for CI, then the NPI upper probability for CI is considered.

4.3.1 Lower probability

To find the NPI lower probability for CI for attribute t_j, $\underline{P}_j(CI)$, we need to minimise over all possible configurations of the probability wheel, then to choose the configuration that gives the smallest possible value.

Let $f_{j,i} = \dfrac{n^{C_r}(t_j = c_i)}{n(t_j = c_i) + 1}$, for $i = 1, ..., k_j$ and $j = 1, ..., T$, so we rewrite Equation (4.4) in the following way

$$\underline{P}_j(CI) = \min_{p_{j,i} \in \mathcal{P}} (f_{j,1}p_{j,1} + f_{j,2}p_{j,2} + ... + f_{j,k_j}p_{j,k_j}). \tag{4.9}$$

Suppose that the fractions $f_{j,i}$ are reordered in an increasing way and relabelled such that $\acute{f}_{j,1} \leq \acute{f}_{j,2} \leq ... \leq \acute{f}_{j,k_j}$, with corresponding $\acute{p}_{j,i}$. For example, considering attribute variable t_1, if the smallest $f_{1,i}$ is $f_{1,2}$, then $\acute{f}_{1,1} = f_{1,2}$ and $\acute{p}_{1,1} = p_{1,2}$. Thus, the NPI lower probability for CI is

$$\underline{P}_j(CI) = \min_{\acute{p}_{j,i} \in \mathcal{P}} (\acute{f}_{j,1}\acute{p}_{j,1} + \acute{f}_{j,2}\acute{p}_{j,2} + ... + \acute{f}_{j,k_j}\acute{p}_{j,k_j}). \tag{4.10}$$

We separate categories corresponding to the largest $\acute{f}_{j,i}$ as much as possible to ensure that we can assign probability masses of slices 'in between' to its neighbour with smaller value of $\acute{f}_{j,i}$, for minimisation. Therefore, the configuration of the wheel which gives the most flexibility to do so is the arrangement where categories $c_1, ..., c_{k_j}$ corresponding to $\acute{p}_{j,1}, ..., \acute{p}_{j,k_j}$ are permuted in the following way,

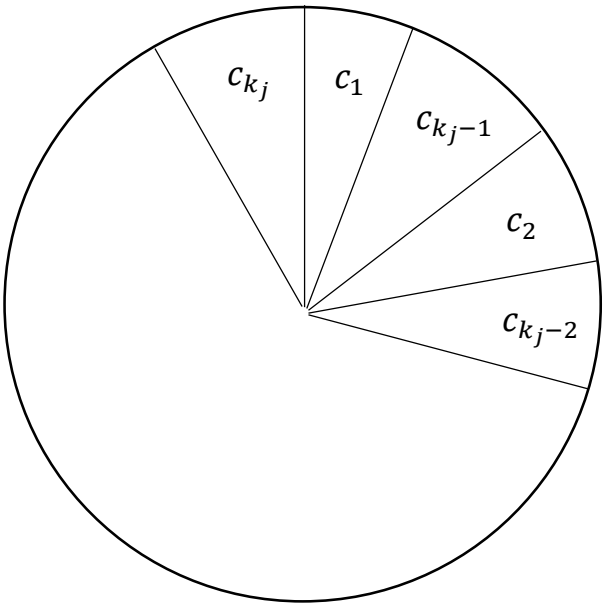

Figure 4.1: Optimal configuration.

$c_1, c_{k_j-1}, c_2, c_{k_j-2}, c_3, c_{k_j-3}, ..., c_{k_j}$. This arrangement can be represented on the probability wheel as in Figure 4.1.

Generally speaking, to find the NPI lower probability for CI for each attribute variable t_j, each category is assigned its lower probability, $\frac{n_{ji}-1}{n}$ and the remaining probability mass is then shared between the categories with the smallest $f_{j,i}^{'}$ in such a way to derive the smallest value of the NPI lower probability for CI. However, the way in which this can be shared must not violate the constraints on the probability wheel for each category. Hence, only $\frac{1}{n}$ can be assigned from either side to a category, which implies at most $\frac{2}{n}$ in total to any category. We now consider two cases: First, when k_j is even. Then, when k_j is odd.

Case 1: k_j is even

To find the NPI lower probability for CI for each attribute variable t_j, we initially assign probability mass $\frac{n_{ji}-1}{n}$ to each category. Once these probability assignments are made, there are k_j separating slices remaining. These separating slices must

then be shared equally between the categories with the smallest fractions $f_{j,i}^{'}$, provided that the resulting probabilities are no larger than their upper limits $\frac{n_{ji}+1}{n}$. Therefore, the k_j remaining probability masses, each $\frac{1}{n}$, must be assigned to c_1 to $c_{\frac{k_j}{2}}$, hence, we assign additional probability mass of $\frac{2}{n}$ to each of c_1 to $c_{\frac{k_j}{2}}$.

Example 4.3.1 Suppose we have 6 possible categories, c_1, c_2, c_3, c_4, c_5 and c_6, and data where $(n_1, n_2, n_3, n_4, n_5, n_6) = (1, 2, 3, 4, 5, 6)$, so $n = 21$. For simplicity, suppose also that we reorder their corresponding fractions $f_{j,i}$, for $i = 1, ..., 6$ in an increasing order such that $f_{j,1}^{'} \leq ... \leq f_{j,6}^{'}$. Then the NPI lower probability for CI for attribute t_j is

$$\underline{P}_j(CI) = \min_{\acute{p}_{j,i} \in \mathcal{P}} (f_{j,1}^{'}\acute{p}_{j,1} + f_{j,2}^{'}\acute{p}_{j,2} + ... + f_{j,6}^{'}\acute{p}_{j,6}) \tag{4.11}$$

Using NPI for multinomial data, each category is assigned its lower probability $\frac{n_{ji}-1}{n}$. Thus, we assign $\left(0, \frac{1}{21}, \frac{2}{21}, \frac{3}{21}, \frac{4}{21}, \frac{5}{21}\right)$ to $(c_1, c_2, c_3, c_4, c_5, c_6)$, respectively. After that, there is a total remaining probability mass of $\frac{6}{21}$, which can be assigned to c_1, c_2 and c_3, with the constraint that not more than $\frac{2}{21}$ can be assigned to a single category. Figure 4.2 shows these categories in the optimal configuration of the probability wheel, which leads to the NPI lower probability for CI. Note that this and similar below configurations (for min and max) are not unique optima, as one could exchange some of the segments with the same results. The slices separating c_1 from c_5 and c_6 are both assigned to c_1, the slices separating c_2 from c_4 and c_5 are both assigned to c_2, and the slices separating c_3 from c_4 and c_6 are both assigned to c_3. Hence, these assignments are $\left(\frac{2}{21}, \frac{2}{21}, \frac{2}{21}\right)$ to (c_1, c_2, c_3). Therefore, the final assignments are $\left(\frac{2}{21}, \frac{3}{21}, \frac{4}{21}, \frac{3}{21}, \frac{4}{21}, \frac{5}{21}\right)$ to $(c_1, c_2, c_3, c_4, c_5, c_6)$, respectively. Following these assignments we get $\underline{P}_j(CI)$.

□

Case 2: k_j is odd

As for even-valued k_j, we initially assign probability mass $\frac{n_{ji}-1}{n}$ to each category, c_i, for $i = 1, ..., k_j$. Then, as we aim to assign maximal probability masses to the

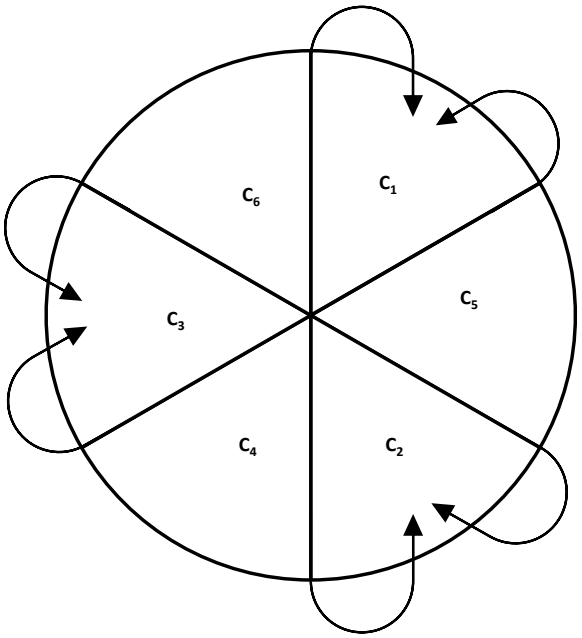

Figure 4.2: Optimal configuration of probability wheel for Example 4.3.1

categories with the smallest $f_{j,i}^{'}$, we assign probability mass of $\frac{2}{n}$ to each of c_1 to $c_{\frac{k_j}{2}-\frac{1}{2}}$. After that we assign the last remaining probability mass $\frac{1}{n}$ to category $c_{\frac{k_j}{2}+\frac{1}{2}}$.

Example 4.3.2 Consider the same data described in Example 4.3.1, excluding the last category c_6 and its corresponding fraction $f_{j,6}$. We then have 5 possible categories where $n = 15$. Recall that their corresponding fractions $f_{j,i}$ are reordered, for $i = 1, ..., 5$ in an increasing order. Figure 4.3 shows the optimal configuration of the probability wheel, which leads to the NPI lower probability for CI. In this example, first, each category is assigned its lower probability $\frac{n_{ji}-1}{n}$. Thus, we assign $\left(0, \frac{1}{15}, \frac{2}{15}, \frac{3}{15}, \frac{4}{15}\right)$ to $(c_1, c_2, c_3, c_4, c_5)$, respectively. Then, we assign the slices separating c_1 from c_4 and c_5 to c_1, and we assign the slices separating c_2 from c_3 and c_4 to c_2. The slice between c_3 and c_5 are assigned to c_3. This means that we assign $\left(\frac{2}{15}, \frac{2}{15}\right)$ to (c_1, c_2), respectively, and we assign the last remaining probability mass $\frac{1}{15}$ to c_3. Therefore, the final assignment is $\left(\frac{2}{15}, \frac{3}{15}, \frac{3}{15}, \frac{3}{15}, \frac{4}{15}\right)$ to $(c_1, c_2, c_3, c_4, c_5)$,

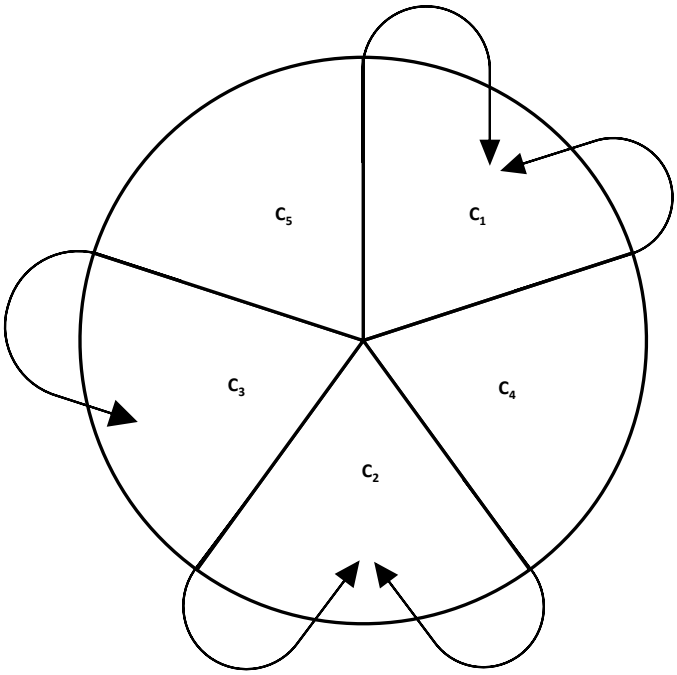

Figure 4.3: Optimal configuration of probability wheel for Example 4.3.2

respectively, which leads to $\underline{P}_j(CI)$.

□

Following the above method in Case 1 and Case 2 of assigning probability masses to the possible categories, we get the NPI lower probability for CI for attribute t_j, $\underline{P}_j(CI)$.

4.3.2 Upper probability

To find the NPI upper probability for CI for attribute t_j, we need to maximise over all possible configurations on the probability wheel, and then choose a configuration that gives the highest possible value of $\overline{P}_j(CI)$.

Let $f_{j,i} = \dfrac{n^{C_r}(t_j = c_i) + 1}{n(t_j = c_i) + 1}$, for $i = 1, ..., k_j$ and $j = 1, ..., T$, so we rewrite the

Formula (4.6) in the following way

$$\overline{P}_j(CI) = \max_{p_{j,i} \in \mathcal{P}} (f_{j,1}p_{j,1} + f_{j,2}p_{j,2} + ... + f_{j,k_j}p_{j,k_j}) \tag{4.12}$$

and then we rearrange the $f_{j,i}$ in an increasing order such that $\acute{f}_{j,1} \leq \acute{f}_{j,2} \leq ... \leq \acute{f}_{j,k_j}$. Thus, the NPI upper probability for CI for attribute t_j is

$$\overline{P}_j(CI) = \max_{\acute{p}_{j,i} \in \mathcal{P}} (\acute{f}_{j,1}\acute{p}_{j,1} + \acute{f}_{j,2}\acute{p}_{j,2} + ... + \acute{f}_{j,k_j}\acute{p}_{j,k_j}). \tag{4.13}$$

To maximise these NPI upper probabilities for CI, we separate the largest categories as much as possible to ensure that we can assign probability masses of slices 'in between' to their neighbours with larger value of $\acute{f}_{j,i}$. Therefore, we consider the same configuration of the probability wheel (see Figure 4.1) that is used to find the NPI lower probability, but we want to assign as much probability mass as possible to the categories with the largest $\acute{f}_{j,i}$. Of course, each category is assigned its lower probability $\dfrac{n_{ji} - 1}{n}$, and the remaining probability masses are shared between other categories in such a way to get the largest possible value for the probability for CI. We consider the following two cases to explain how this maximisation can be done.

Case 1: k_j is even

As a first step, we assign probability mass $\dfrac{n_{ji} - 1}{n}$ to each category. As a second step, the remaining probability masses are then shared equally between the categories with the largest possible $\acute{f}_{j,i}$. Hence, we assign probability mass of $\dfrac{2}{n}$ to the categories $c_{\frac{k_j}{2}+1}$ to c_{k_j}. This method leads to the maximum possible value of the NPI upper probability for CI, since we assign the remaining probability masses between the largest categories. Note that we can not assign more than probability mass of $\dfrac{2}{n}$ to any category.

Example 4.3.3 Consider the same data described in Example 4.3.1, where $k = 6$. Recall that we reorder their corresponding fractions $f_{j,i}$, for $i = 1, ..., 6$ in an increasing order. Figure 4.4 shows these categories in the optimal configuration of

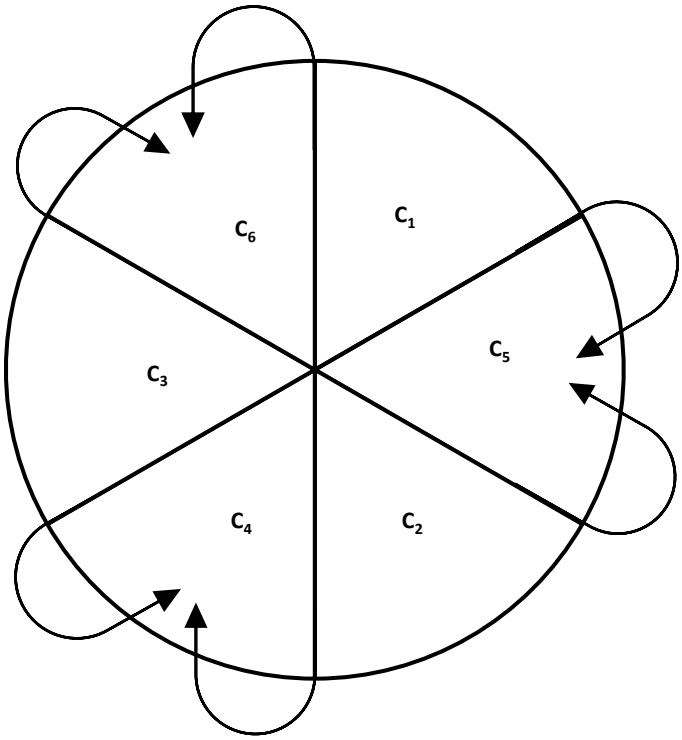

Figure 4.4: Optimal configuration of probability wheel for Example 4.3.3

probability wheel, which leads to the NPI upper probability for CI. First, we assign $\left(0, \frac{1}{21}, \frac{2}{21}, \frac{3}{21}, \frac{4}{21}, \frac{5}{21}\right)$ to $(c_1, c_2, c_3, c_4, c_5, c_6)$, respectively. Then, to derive the NPI upper probability for CI, the slices separating c_6 from c_1 and c_3 are both assigned to c_6, the slices separating c_5 from c_1 and c_2 are both assigned to c_5, and the slices separating c_4 from c_2 and c_3 are both assigned to c_4. These assignments are $\left(\frac{2}{21}, \frac{2}{21}, \frac{2}{21}\right)$ to (c_4, c_5, c_6). Therefore, the final assignments are $\left(0, \frac{1}{21}, \frac{2}{21}, \frac{5}{21}, \frac{6}{21}, \frac{7}{21}\right)$ to $(c_1, c_2, c_3, c_4, c_5, c_6)$, respectively. Following these assignments we get $\overline{P}_j(CI)$.

□

Case 2: k_j is odd

We initially assign probability mass $\frac{n_{ji} - 1}{n}$ to each category, c_i, for $i = 1, ..., k_j$. Then, the best way to distribute the remaining probability masses is to assign probability mass of $\frac{2}{n}$ to the categories $c_{\frac{k_j}{2}+\frac{3}{2}}$ to c_{k_j}. After that, to assign the last

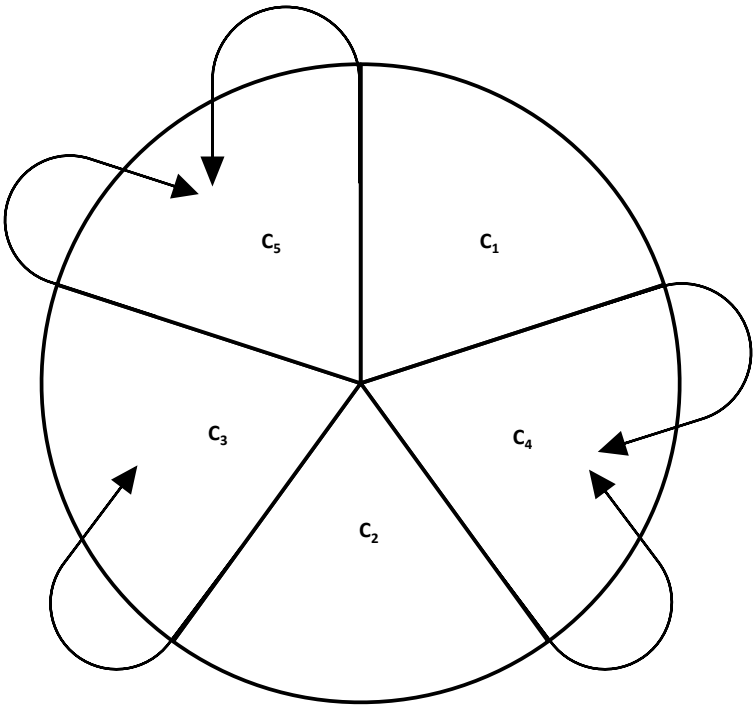

Figure 4.5: Optimal configuration of probability wheel for Example 4.3.4

probability mass $\frac{1}{n}$ to $c_{\frac{k_j}{2}+\frac{1}{2}}$. This distribution of the probability masses leads to the maximum possible value of the probability for CI.

Example 4.3.4 Consider the same data in Example 4.3.2, where $k = 5$. Assume also that we reorder their corresponding fractions $f_{j,i}$, for $i = 1, ..., 5$ in an increasing order. Figure 4.5 shows the optimal configuration of the probability wheel, which leads to the NPI upper probability for CI. To derive the NPI upper probability for CI, we first assign $\left(0, \frac{1}{15}, \frac{2}{15}, \frac{3}{15}, \frac{4}{15}\right)$ to $(c_1, c_2, c_3, c_4, c_5)$, respectively. Then, we assign the slices separating c_5 from c_1 and c_3 to c_5, and we assign the slices separating c_4 from c_1 and c_2 to c_4. The slice between c_2 and c_3 is assigned to c_3. This means that we further assign $\left(\frac{2}{15}, \frac{2}{15}\right)$ to (c_4, c_5), respectively, and $\frac{1}{15}$ to c_3. Therefore, the final assignments are $\left(0, \frac{1}{15}, \frac{3}{15}, \frac{5}{15}, \frac{6}{15}\right)$ to $(c_1, c_2, c_3, c_4, c_5)$, respectively, which leads to $\overline{P}_j(CI)$.

□

Following the above method of assigning probability masses to the possible categories, we get the NPI upper probability for CI for attribute t_j, $\overline{P}_j(CI)$. Finally, we can use these NPI lower and upper probabilities for CI to build classification trees using the D-NPI-M algorithm, as explained in Section 4.4.

4.4 The D-NPI-M algorithm

In this section we present a new algorithm for classification trees with multinomial data, which we call *Direct Nonparametric Predictive Inference classification for Multinomial data* (D-NPI-M). It is similar to the D-NPI classification tree algorithm introduced in Section 3.3.2 for binary data, but the main difference is that multinomial data are considered here. In the D-NPI-M algorithm, we use the CI split criterion for multinomial data, introduced in Section 4.3, in order to choose the best splitting attribute variable at each node when building a classification tree. The building process is similar to the well-known C4.5 algorithm (see Section 2.2), but using the CI, introduced in Section 4.3, as a splitting criterion. We use the same stop criterion introduced in Section 3.3.2.

Suppose that we have a training data set, $\mathcal{D}$, which has categorical attribute variables, t_j, with $c_1, ..., c_{k_j}$ as their possible values. Let C be the target variable with $C_1, ..., C_R$ as its possible classes. Note that we refer to target classes by C_r, and we use c_i to indicate attribute categories, for $i = 1, ..., k_j$. For the training data set, $\mathcal{D}$, we first calculate the CI intervals, i.e. the NPI lower and upper probabilities for CI, for the complete list of attribute variables t_j, using the NPI lower and upper probabilities for CI as given by Equations (4.4) and (4.6). Then, we compare these NPI lower and upper probabilities for CI with the NPI lower and upper probabilities for NA given by the NPI-M lower and upper probabilities as shown in Section 2.4.3 (see Equations (2.23) and (2.24)), which correspond to the largest class state. Recall that we indicate the NPI lower and upper probabilities for NA by $\underline{P}(NA)$ and $\overline{P}(NA)$, respectively, as shown in Section 3.3.2. Recall also that

these NPI lower and upper probabilities are the NPI lower and upper probabilities for CI if no attribute variable is used. The NPI lower and upper probabilities for NA correspond to simply stating the most common class of the target variable in the data set, given in Equations (2.23) and (2.24), that correspond to the largest class of the target variable. If there are two or more classes with the same number of observations, we use any of them as all will give the same NPI lower and upper probabilities for NA. Of course, one would be looking at the subset of the data set at the specific stage of the classification tree.

Similarly as done in Section 3.3.2, we choose the attribute variable with the highest value of the NPI lower and upper probabilities for CI, and which satisfy the two conditions in (3.10) to place at the root node. Recall that the two conditions in (3.10) are used as a stop criterion in Section 3.3.2. The same stop criterion is also used in this section.

After selecting the best attribute variable t_{j^*} for the root node, we split the training data set, $\mathcal{D}$, into disjoint subsets $\mathcal{D}_{j^*=c_i}$, where $\mathcal{D}_{j^*=c_i}$ includes all instances with value $t_{j^*} = c_i$, for $i = 1, ..., k_{j^*}$ for the selected attribute variable. For example, if we choose an attribute variable to split on with three categories, red, blue and green, then we will have three subsets of the training data set, $\mathcal{D}_{j^*=red}$, $\mathcal{D}_{j^*=blue}$ and $\mathcal{D}_{j^*=green}$. Then we calculate the NPI lower and upper probabilities for CI for each subset and compare the CI intervals of each subset with the corresponding NPI lower and upper probabilities for NA.

The D-NPI-M algorithm continues recursively by splitting further and hence, constructs a new subtree for each branch not yet ending in a node. The algorithm terminates when the two conditions in (3.10) are not fulfilled or when the observations in the subset all belong to the same class, in which case this class is used as a label to that corresponding leaf node. This process can be represented as a classification tree. The procedure to classify a new instance in the D-NPI-M algorithm can be summarized in Algorithm 2, where this pseudocode of the D-NPI-M algorithm is

similar to the algorithm used in Chapter 3 to build classification trees with binary data. The main difference between the two algorithms is that here we apply the CI split criterion to multinomial data. In Example 4.4.1, we illustrate the procedure to build a classification tree using the D-NPI-M algorithm.

Algorithm 2 Pseudocode of the D-NPI-M algorithm.

Input:
`TR`: Training data set
`Target`: Target variable
`Attr`: List of attribute variables
procedure D-NPI-M(`TR`, `Target`, `Attr`)
Create a Root node for the tree
    if `TR` have the same class C, **then**
        Return the single-node tree with class C
    if Attr is empty, **then**
        Return the single-node tree with the most common class C in `TR`
    Otherwise
    for each attribute, t in `Attr` **do**
        Compute $\underline{P}_t(CI)$ and $\overline{P}_t(CI)$
        Choose attribute, t with highest $\underline{P}_t(CI)$ and $\overline{P}_t(CI)$
        if $\underline{P}_t(CI) > \underline{P}(NA)$ and $\overline{P}_t(CI) > \overline{P}(NA)$ **then**
            Choose the attribute, t
        else
            Add a leaf node labelled with the most common class in `TR`
    Set t the attribute for Root
    for each value of t, vi, **do**
        Add a branch below Root, corresponding to $t = v_i$
        Let $\texttt{TR}_{v_i}$ be the subset of `TR` that have $t = v_i$
        if $\texttt{TR}_{v_i}$ is empty, **then**
            Add a leaf node labelled with the most common class in `TR`
        else
            Add the subset generated by D-NPI-M($\texttt{TR}_{v_i}$, `Target`, `Attr`-$\{t\}$)
return Root

Attribute	$\underline{P}(CI)$	$\overline{P}(CI)$
a1	0.5463	0.6759
a2	0.5737	0.6571
a3	0.5737	0.6571
a4	0.6314	0.7532

Table 4.3: The CI intervals for all attribute variables.

Example 4.4.1 In this example we illustrate the process of building the D-NPI-M classification tree, using the Lenses data set, obtained from the UCI machine learning repository [32]. The Lenses data set consists of 24 instances, for each there are instances of 4 attribute variables and a class variable. All attribute variables are categorical with 2 or 3 categories. The class variable has three states which are 'the patient should be fitted with hard contact lenses', denoted here as 'Hard Lenses', 'the patient should be fitted with soft contact lenses', denoted by 'Soft Lenses' and 'the patient should not be fitted with contact lenses', denoted by 'No Lenses'. The class variable is distributed as follows: 4 instances in state 1, 5 instances in state 2 and 15 instances in state 3. Note that the aim of this example is only to illustrate the building process of the D-NPI-M classification tree, measuring its performance and comparing it to other trees is discussed in Section 4.5.

First, we calculate the NPI lower and upper probabilities for CI (CI intervals) for all attribute variables using Formulas (4.4) and (4.6). These NPI lower and upper probabilities for CI are shown in Table 4.3. Then, we compare the CI intervals in Table 4.3 with the NPI lower and upper probabilities for NA, which are given by the NPI-M lower and upper probabilities for the largest state of the class variable. These NPI lower and upper probabilities for NA are

$$\underline{P}(NA) = \max\left(0, \frac{n_i - 1}{n}\right) = \frac{14}{24} = 0.5833$$

and

$$\overline{P}(NA) = \min\left(\frac{n_i + 1}{n}, 1\right) = \frac{16}{24} = 0.6667$$

where n_i is the total number of instances in the largest class state.

Attribute	$\underline{P}(CI)$	$\overline{P}(CI)$
a1	0.4000	0.6000
a2	0.4286	0.5714
a3	0.6310	0.7976

Table 4.4: The CI intervals for a4 = 2.

In order to find the best splitting attribute, we choose the attribute with the highest value of the NPI lower and upper probabilities for CI which satisfy the two conditions given in (3.10). We can see that attribute 4 (a4) has the highest CI interval and satisfies the two conditions in (3.10). Therefore, we assign a4 to the root node. Then, we split the data set according to the value of a4. The attribute variable a4 has two values coded as 1 and 2. Note that a4 is no longer considered for splitting in the next stages of building the D-NPI-M classification tree. As a4 has only 2 values, we split the data set into two subsets and again calculate the NPI lower and upper probabilities for CI for each subset. Then, we compare the values of CI intervals in each subset with the corresponding NPI lower and upper probabilities for NA. Table 4.4 shows the CI intervals when a4 has value 2. In the subset of the data set with a4 value 1, we get a pure subset in which all instances have class 'No Lenses'.

As the next step in the process of building the classification tree, we compare the CI intervals in Table 4.4 with the corresponding NPI lower and upper probabilities for NA which are $[0.3, 0.5]$. Therefore, we choose a3 as a second split below the branch of a4 = 2. Similarly, we select a2 as a further split below the branch of a3 = 2. The tree terminates when the two conditions in (3.10) are not met or when instances in a subtree all have the same class. So, we stop here and present the resulting full tree as in Figure 4.6. More analysis of the D-NPI-M classification tree and comparisons with other classification methods for building classification trees are presented in Section 4.5.

□

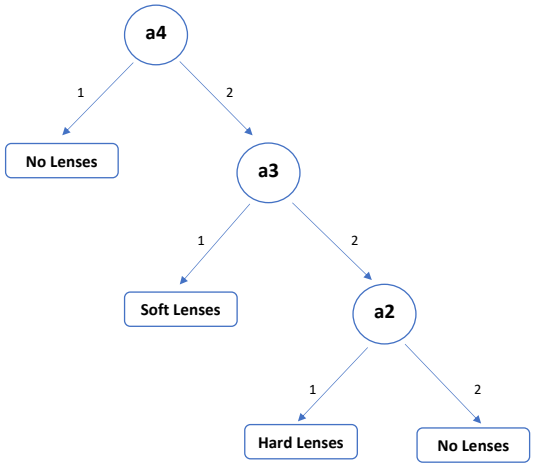

Figure 4.6: Classification Tree of Lenses Data set.

4.5 Performance of the D-NPI-M algorithm

In this section, we examine the performance of the D-NPI-M algorithm on eight data sets from the UCI Machine Learning Repository database [32]. The aim of this experimental analysis is to assess the performance of the D-NPI-M algorithm, and to compare it with the most commonly used classical algorithm, the C4.5 algorithm, and with some algorithms based on imprecise probabilities, namely the NPI-M, A-NPI-M and IDM algorithms (the latter with two choices of the parameter $\tilde{s}$). More details about these algorithms have been presented in Section 2.2.2. The data sets used in this experimental analysis are diverse in terms of their size, the number of classes and the range of categories of the attribute variables. A summary of the main characteristics of each data set is given in Table 4.5, where column 'N' indicates the number of instances in a data set, column 'Att' indicates the number of attribute variables, column 'Range of Att' indicates the range of the number of states of the categorical variables, and 'Classes' indicates the number of states of the class variable. Further information and more details about these data sets can be found in [32]. Two of these data sets have only two classes, but we use them in this chapter as they have more than two categories for the attribute variables. Note that in Chapter 3 we consider data sets where both the class variable and attribute variables are binary.

Data set	N	Att	Range of Att	Classes
CMC	1473	9	2-4	3
Hayes-Roth	160	5	3-4	3
Lenses	24	4	2-3	3
Modified Iris	150	4	3	3
Monk's Problems-1	124	7	2-4	2
Nursery	12960	8	2-5	5
Post-Operative Patient	90	8	2-4	3
Qualitative-Bankruptcy	250	6	3	2

Table 4.5: Data sets description.

The experiments were conducted using the statistical software R [57]. Six classification algorithms have been used in this analysis. To build the C4.5 algorithm, we used the RWeka package [39, 67]. More details about the C4.5 algorithm have been presented in Section 2.2.2. We used the imptree package for building the NPI-M, A-NPI-M and IDM algorithms [34]. The D-NPI-M classification algorithm has been built using the procedure introduced in Section 4.4. A 10-fold cross-validation procedure, as described in Section 2.2.1, has been used for each data set. In this analysis, we used only eight data sets because we have not fully automated the D-NPI-M algorithm for constructing the classification trees, which makes the application to each data set time-consuming.

The Nursery data set is large, so to reduce the amount of computation required for this data set, we fix a minimum split number of 100 observations in order to split any node further or otherwise we terminate the tree and fix a leaf node with the most common class in that node. So, this is an added stop criterion when building classification trees. A minimum split number is a value that needs to be at least attained to qualify for splitting. A minimum split value is sometimes fixed for a specific value to reduce the required computation, in the same way as done by Bertsimas and Dunn [16]. This minimum split value is also used for all other algorithms applied to the Nursery data set, hence the comparison will be fair although the accuracy could be lower than what is expected for this data set when the minimum split value is not fixed.

In the modified Iris data set, we convert four continuous attributes to categorical attributes with three categories coded as, 'L', 'M' and 'H' using equal frequency from the `'arules'` package in R and using the `'discretize'` function. This function converts a continuous variable into a categorical variable using different strategies. The default one is used which is equal frequency. Note that the frequencies may not be exactly equal because of ties in the data. However, we aimed to convert all attributes to categorical ones, then to apply all classification algorithms on the converted data, because the D-NPI-M algorithm presented in this thesis only functions for categorical attributes. It will be of interest to generalise the D-NPI-M algorithm to deal with continuous attribute variables, as a topic for future research. All other data sets only have categorical attribute variables. All missing values were replaced by modal values. Finally, all classification algorithms have been applied to all these data sets with the same pre-analysis steps in order to compare them fairly.

In order to measure the performance of the D-NPI-M algorithm and all the other algorithms, three metrics have been used. First, we used the classification accuracy rate, which is the most commonly used method to measure the performance of classification algorithms. It is calculated as the ratio of the total number of correctly classified instances on the testing set to the total number of instances in the testing set. More details about the classification accuracy rate have been introduced in Section 3.4. Secondly, we used in-sample accuracy, which is the classification accuracy rate on the training set (see Section 3.4 for more details). Thirdly, an average tree size for each algorithm is reported. Note that we refer to tree size as the total number of leaf nodes, as was done by Bertsimas and Dunn [16] and by Murthy and Salzberg [53]. All these measures have been calculated from 10 runs using 10-fold cross validation, then the average results of these runs are reported as a final result.

First, the performance of the D-NPI-M classification algorithm in terms of its classification accuracy has been evaluated against five other classification algorithms. Table 4.6 presents the classification accuracies of the D-NPI-M classification algo-

Data set	D-NPI-M	C4.5	NPI-M	A-NPI-M	IDM1	IDM2
CMC	**45.49**	45.31	42.93	42.93	42.93	42.93
Hayes-Roth	66.76	66.92	64.62	64.62	63.08	**67.69**
Lenses	**81.67**	70.00	80.00	80.00	75.00	80.00
Modified Iris	**95.33**	92.67	90.00	90.00	92.67	92.67
Monk's Problems-1	**73.33**	69.17	69.17	69.17	69.17	69.17
Nursery	**90.37**	89.21	89.20	89.20	89.20	89.20
Post-Operative Patient	67.78	68.89	**71.11**	**71.11**	**71.11**	**71.11**
Qualitative-Bankruptcy	**99.60**	98.00	98.40	98.40	98.40	98.40
Average	**77.54**	74.94	75.68	75.68	75.20	76.40

Table 4.6: Accuracy results of all classification algorithms.

rithm and all the other algorithms for each data set. The results in Table 4.6 indicate that the D-NPI-M classification algorithm slightly outperforms other classification algorithms in six data sets. In Section 4.4, we have argued that we should use the two conditions in (3.10) in order to select any attribute variable for splitting when building classification trees by the D-NPI-M classification algorithm. It is noticed that during the experimental analysis, using these two conditions leads to some improvements in the classification accuracy for the D-NPI-M algorithm.

For the Lenses data set, there is a clear difference in the classification accuracies among classification algorithms, where the D-NPI-M classification algorithm clearly outperforms the C4.5 and IDM1 algorithms, and is slightly superior to the NPI-M, A-NPI-M and IDM2 algorithms. The reduction in the accuracy for the C4.5 and IDM1 algorithms might be because they sometimes stop earlier than other algorithms when building classification trees. However, the clear difference in the classification accuracy between different classification algorithms could be also because it is a small data set with only 24 observations. This could be expected particularly for smaller data sets, where may be larger differences in the classification accuracy.

For the Monk's Problem-1 data set, the D-NPI-M classification algorithm has the highest classification accuracy of 73.33%, where all other algorithms have the same accuracy of 69.17% but with different trees generated by these algorithms.

For this data set, the D-NPI-M classification algorithm returns relatively larger trees than the other algorithms which might be the reason for its superiority. On some occasions when analysing this data set, the CI split criterion, which is used for the D-NPI-M classification algorithm, suggests more attribute variables to split on when considering subsets for which other classification algorithms choose not to split. This may indicate that our split criterion identified a further useful variable, leading to better overall accuracy.

For the Post-Operative Patient data set, the NPI-M, A-NPI-M and IDM classification algorithms have the highest classification accuracy rates. For the Hayes-Roth data set, the IDM2 algorithm is slightly superior to all other algorithms with classification accuracy of 67.69%. For the CMC, Modified Iris, Nursery and Qualitative-Bankruptcy data sets, the D-NPI-M classification algorithm is superior to all the other algorithms. For the Modified Iris data set, the D-NPI-M classification algorithm sometimes selects a different attribute variable on the root node than the other classification algorithms, which improves its classification accuracy slightly. In fact, its split criterion CI returns two different attribute variables with the same NPI lower and upper probabilities for CI, which gives us the decision to start with one of them. So, we randomly choose one of them. For the Qualitative-Bankruptcy data set, the D-NPI-M classification algorithm splits one further step in some trees, while the other classification algorithms stopped splitting. So here the split criterion CI also identifies an attribute variable with some useful information, which may indicate the possible class state of the target variable. As a result, the D-NPI-M algorithm gives slightly better classification accuracy than the other algorithms. Overall, according to the average classification accuracy rate, we can say that all classification algorithms are performing similarly, but with a slightly better performance by the D-NPI-M algorithm.

Secondly, following [16, 53], we have used the in-sample accuracy rate to measure the performance of the D-NPI-M classification algorithm on the training set, and to compare it with the other classification algorithms. The in-sample accuracy mea-

Data set	D-NPI-M	C4.5	NPI-M	A-NPI-M	IDM1	IDM2
CMC	**48.84**	47.22	45.17	45.17	45.17	45.17
Hayes-Roth	**83.67**	81.51	79.33	79.33	81.51	82.35
Lenses	**87.49**	84.55	85.91	85.91	85.00	85.91
Modified Iris	95.33	**95.41**	92.07	92.07	94.07	94.07
Monk's Problems-1	**84.24**	74.82	73.66	73.66	73.66	73.66
Nursery	**90.37**	89.21	89.26	89.26	89.26	89.26
Post-Operative Patient	71.23	**71.36**	71.11	71.11	71.11	71.11
Qualitative-Bankruptcy	**99.60**	99.24	98.40	98.40	98.40	98.40
Average	**82.60**	80.42	79.36	79.36	79.77	79.99

Table 4.7: In-sample accuracy results of all classification algorithms.

sure is not commonly used to indicate classification accuracy, but it gives insight into how the algorithm performs on the training set. It is known that if the classification algorithm performs very well on the training set but not very well on the testing set, this indicates likely overfitting. Thus, the in-sample accuracy is reported to show the performance of classification algorithms on both training and testing sets.

Table 4.7 shows the in-sample accuracy results of all classification algorithms. The D-NPI-M classification algorithm slightly outperforms the other classification algorithms in several data sets, followed by the C4.5 classification algorithm which is also superior to other algorithms in some data sets. It is noticed that the IDM2 algorithm slightly outperforms the IDM1 algorithm with regard to both average classification accuracy and average in-sample accuracy rates. In this experimental analysis, the NPI-M and A-NPI-M algorithms are equivalent in all performance measures. These two algorithms do not always lead to the same result as shown by Baker [14]. Finally, the D-NPI-M classification algorithm has the highest average result of in-sample accuracy compared to the other classification algorithms. It should be clarified that the D-NPI-M algorithm has good results on in-sample accuracy and classification accuracy as well, which may indicate that it does not suffer from overfitting. For example, the D-NPI-M classification algorithm has a largest in-sample accuracy rate in the Monk's Problems-1 data set compared to other classification algorithms, but it also has the largest classification accuracy rate on testing set (see

Algorithm	D-NPI-M	C4.5	NPI-M	A-NPI-M	IDM1	IDM2
Average	8.53	**7.85**	8.79	8.79	9.58	8.64

Table 4.8: Average tree size for all classification algorithms.

Table 4.6).

Thirdly, in order to compare different trees generated by the classification algorithms, the average tree size for each algorithm is reported. Table 4.8 shows the average tree size for each classification algorithm. Note that we refer to tree size as the total number of leaf nodes as was done by Bertsimas and Dunn [16], and Murthy and Salzberg [53]. However, other researchers may consider the total number of all nodes. Of course both methods can be used to refer to the tree size. It can be observed from Table 4.8 that the average tree size of all algorithms is nearly equivalent with some smaller trees generated by the C4.5 algorithm followed by the D-NPI-M algorithm. With regard to the IDM1 and IDM2 algorithms, we notice that increasing the value of the parameter $\tilde{s}$ could lead to generating smaller trees. This result for the size of trees generated by the IDM algorithms has also been reported by Abellán et al. [3] in an extensive experiment to assess the performance of different classification tree algorithms.

To summarise, from the classification accuracy given in Table 4.6, we draw the following conclusions about the performance of the D-NPI-M algorithm. The D-NPI-M algorithm is performing well and slightly outperforms other algorithms. The NPI-M and A-NPI-M algorithms are performing the same on all these data sets. The IDM2 algorithm outperforms the IDM1 algorithm with regard to this measure. With regard to the in-sample accuracy, the D-NPI-M algorithm is slightly superior to other algorithms followed by the C4.5 algorithm. As the D-NPI-M algorithm performs well with regard to both the classification accuracy and in-sample accuracy, this could be an indication it does not overfit the data sets. Finally, the C4.5 algorithm has the smallest average tree size, while the IDM1 algorithm has the largest average tree size.

Algorithm	D-NPI-M	C4.5	NPI-M	A-NPI-M	IDM1	IDM2
Accuracy	**75.00**	50.00	66.67	66.67	58.33	58.33

Table 4.9: Single classification accuracies for Monk's Problems-1 data set.

4.6 Monk's Problems-1 data set example

In this section, we analyse the Monk's Problems-1 data set in more detail, in order to see the differences between the classification trees which are generated by different classification algorithms. This data set is considered in this analysis as there is a clear difference in the overall classification accuracy between the D-NPI-M algorithm and the other classification algorithms. This data set contains a total of 124 instances equally divided between two classes, but we consider this data set here as its attribute variables have more than two categories. Note that in Chapter 3 we use data sets where both the class variable and attribute variables are binary. There are seven attribute variables, where one of these attribute variable refers to an ID for each instance, hence, this ID attribute variable has been removed before the analysis. The remaining six attribute variables are named as **a1**, **a2**, **a3**, **a4**, **a5**, and **a6**. These attribute variables consists of 2 to 4 possible categories as follows: **a1**, **a2** and **a4** have three categories, **a3** and **a6** have two categories and **a5** has four categories. There are no missing values in this data set.

In order to evaluate the performance of the D-NPI-M algorithm, and to compare it with other classification algorithms, we have applied a 10-fold cross-validation scheme on this data set, where the average results are shown in Table 4.6. In this section, we randomly take one case from these 10 cases, to illustrate the differences between the classification algorithms considered in this thesis. Table 4.9 illustrates the single classification accuracy for each algorithm with this data set. Note that all classification algorithms have been trained on the same training set and evaluated on the same test set.

Attribute	$\underline{P}(CI)$	$\overline{P}(CI)$
a1	0.6072	0.6363
a2	0.5383	0.5659
a3	0.5438	0.5614
a4	0.5644	0.5920
a5	0.6943	0.7362
a6	0.5088	0.5263

Table 4.10: The CI intervals for all attribute variables.

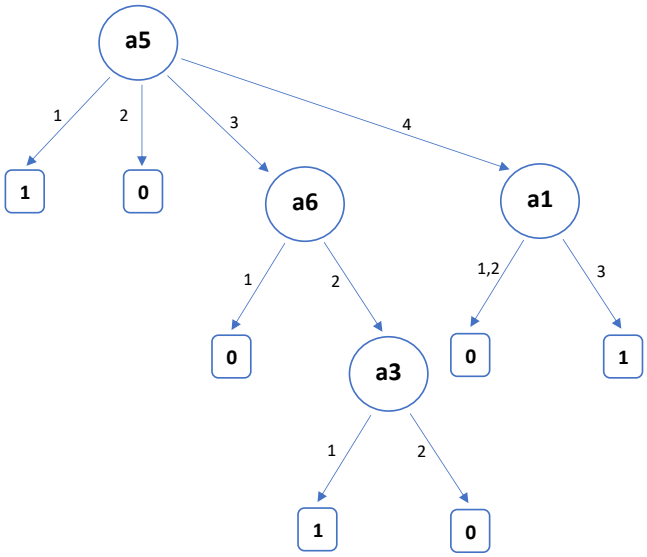

Figure 4.7: The resulting tree for the D-NPI-M algorithm.

The classification tree built for the D-NPI-M algorithm (see Figure 4.7) shows that **a5** is the most informative attribute variable, hence, it is used for splitting at the root node. Table 4.10 shows the NPI lower and upper probabilities for Correct Indication (CI) for all attribute variables, where it is clear that **a5** has the highest NPI lower and upper probabilities for CI, which are also greater than the NPI lower and upper probabilities for NA, given by, $\left[\underline{P}, \overline{P}\right](NA) = [0.4911, 0.5089]$. The C4.5 algorithm also selected **a5** for the root node in its classification tree (see Figure 4.8).

For more clarification, after the second split on **a5= 3** and **a6= 2**, the NPI lower and upper probabilities for NA are $\left[\underline{P}, \overline{P}\right](NA) = [0.5385, 0.6923]$, these val-

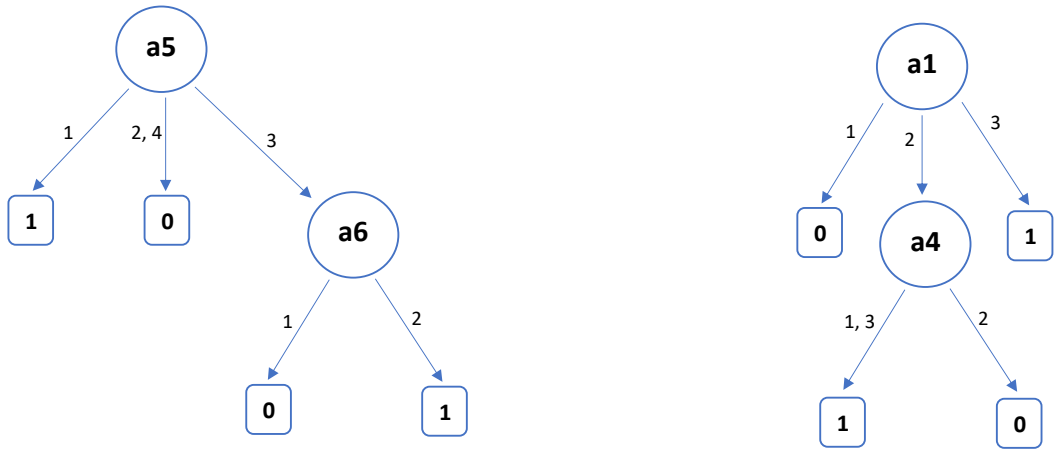

Figure 4.8: The resulting tree for the C4.5 algorithm (left) and the resulting tree for the IDM2 algorithm (right).

ues reflect the amount of information regarding the possible predicted class that come from the class variable at this subtree. At this step, the C4.5 algorithm stopped splitting and fixed a leave node with class **1**. However, the D-NPI-M algorithm found that **a3** has the NPI lower and upper probabilities for CI equals to $\left[\underline{P}, \overline{P}\right](CI) = [0.6410, 0.8205]$, which means that **a3** has more information regarding the possible predicted class than the information given by the NPI lower and upper probabilities for NA. Thus, the D-NPI-M split one more step in this subtree based on **a3**. Here, it can be argued that the CI split criterion identified a useful attribute variable which could lead to better accuracy result on the test set. Although the D-NPI-M algorithm has produced a larger tree than the one generated by the C4.5 algorithm, it has performed clearly better than the C4.5 algorithm on the test set. The D-NPI-M algorithm gives accuracy result of 75%, while the accuracy result for the C4.5 algorithm is only 50%. The fact that the D-NPI-M algorithm has split two more steps in the tree compared to the C4.5 algorithm is clearly the reason for the better performance of the D-NPI-M on this data set.

The NPI-M, A-NPI-M and IDM2 algorithms have built their trees by choosing another attribute variable to place at the root node, which is **a1**. According to the

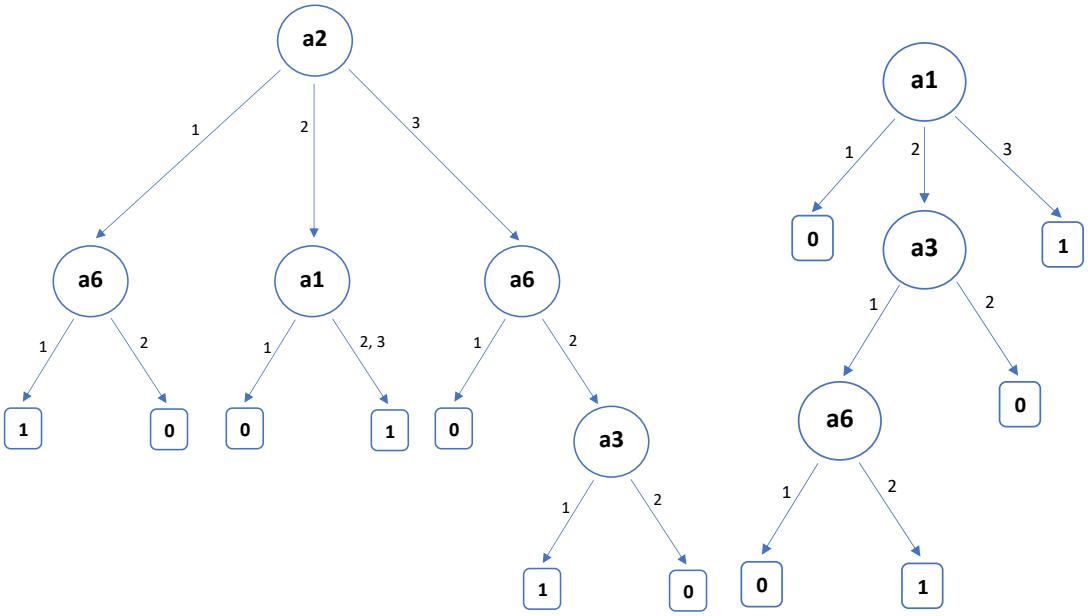

Figure 4.9: The resulting tree for the IDM1 algorithm (left) and the resulting tree for the NPI-M and A-NPI-M algorithms (right).

CI split criterion, **a1** is the second-most informative attribute variable after **a5**, as it has the second largest CI values (see Table 4.10). Figure 4.8 shows the resulting tree built by the IDM2 algorithm, and Figure 4.9 shows the resulting tree built by the NPI-M and A-NPI-M algorithms. The trees which are built by the NPI-M, A-NPI-M and IDM2 algorithms are smaller than the one generated by the D-NPI-M algorithm, but they perform less well than the D-NPI-M algorithm on the test set.

Finally, the tree generated by the IDM1 algorithm is the largest tree compared to all other trees generated by the other algorithms used in this thesis. Interestingly, the IDM2 algorithm built the smallest tree and the IDM1 algorithm built the largest tree, while both algorithms give the same classification accuracy on the test set. However, the D-NPI-M algorithm built a tree which is relatively larger than some of the other trees, but which correctly classified most of unseen instances in the test set. Hence, this tree led to substantial improvement of the classification accuracy result on the test set. Of course, generating large trees which are not performing well on the test set is undesired, this is the case here for the IDM1 algorithm

(see Figure 4.9). This is clearly undesirable because the tree generated by the IDM1 algorithm has accuracy of 58.33% on the test set, while the smallest tree generated by the IDM2 algorithm (see Figure 4.8) gives the same accuracy result on the test set. This means that in this case, the IDM1 produces a larger tree without improving the classification accuracy result. However, relatively larger trees could be desirable if they improve the classification accuracy result on the test set. For example, the tree generated by the D-NPI-M algorithm is relatively large, but it gives the highest classification accuracy result of 75% on the test set compared to other classification algorithms. This may indicate that the CI split criterion identifies some few useful attribute variables which may be missed by the other algorithms, and which lead to some improvements on the overall classification accuracy.

This section has provided additional discussions about different trees generated by the classification algorithms used in this thesis. However, further analysis of this data set, the other data sets considered here, and additional data sets could lead us to more insight into the performance of the D-NPI-M algorithm compared to other algorithms for classification trees and when it might perform well. Such investigations could also enable us to decide when it is best to use the D-NPI-M algorithm or other classification algorithms based on characteristics of the data set. This detailed investigation is not considered here, but it is an interesting topic for future research.

4.7 Concluding remarks

In this chapter we have presented a new algorithm to build classification trees for multinomial data, from the NPI perspective. This classification algorithm is called *Direct Nonparametric Predictive Inference classification for Multinomial data* (D-NPI-M). The D-NPI-M classification algorithm uses *Correct Indication* (CI) as a split criterion. We have discussed the idea of Direct NPI classification with multinomial data. We have then generalised the CI split criterion introduced in Chapter 3

to multinomial data with examples to illustrate the computations of the NPI lower and upper probabilities for CI. Finally, the D-NPI-M algorithm has been introduced with an illustrative example explaining how to build D-NPI-M classification trees.

An experimental analysis has been carried out in order to assess the performance of the D-NPI-M classification algorithm, and to compare it with the C4.5, NPI-M, A-NPI-M, IDM1 and IDM2 classification algorithms. These classification algorithms have been used to build classification trees for several data sets extracted from the UCI Machine Learning Repository database. The performance of all classification algorithms has been then measured using the classification accuracy, in-sample accuracy and tree size.

The results achieved in this chapter have shown that the D-NPI-M classification algorithm slightly outperforms other classification algorithms in some but not all data sets, with regard to classification accuracy rate and in-sample accuracy rate. It has been noticed that, for some data sets, the CI split criterion used in the D-NPI-M algorithm leads to some improvements towards the overall classification accuracy by suggesting splitting on different attribute variables than other algorithms or by concluding a greater number of attribute variables are useful when building classification trees. The results have also indicated that the D-NPI-M classification algorithm tends to build smaller trees than the NPI-M, A-NPI-M, IDM1 and IDM2 classification algorithms, but larger trees than the C4.5 classification algorithm.

As topics for future research, the idea of the Direct NPI classification could be applied to other classification methods such as random forests. We have applied the Direct NPI classification to binary data and multinomial data, however, it would be interesting to generalize the Direct NPI classification to involve other types of data such as real-valued data, ordinal data or data that may contain any other known structure or relationship between the categories. Another possible extension to this work would be the application of the D-NPI-M classification algorithm to imprecise

classification, where it is allowed to return a set of classes rather than a single class. A further topic for future research is the comparison of the D-NPI-M classification algorithm with other classification methods from the literature which do not utilize classification trees. It is clear that the inferences provided by the D-NPI-M classification algorithm are different from the inferences provided by other classification methods, but such comparisons could still lead to some useful information and conclusions. Another topic to investigate, which is in line with the one we discussed at the end of Chapter 3, is to consider developing the D-NPI-M classification algorithm by taking the cost of misclassification into account. In this chapter we have considered only observed categories, but once cost of misclassification is considered, one cannot just neglect unobserved categories.

Finally, it would be of interest to investigate more fully the use of CI as a split criterion for the D-NPI-M classification algorithm. For example, by considering the imprecision, which might be defined as the difference between the NPI lower and upper probabilities for CI, as a stopping criterion. Such considerations may lead to improvement of the performance of the D-NPI-M classification algorithm. Also, further investigations of the performance of the D-NPI-M classification algorithm by considering more data sets with detailed analysis may be beneficial and lead to more insights into aspects of data sets that may cause the D-NPI-M algorithm to perform better than the other methods.

Chapter 5

Direct NPI Classification with Noisy Data

5.1 Introduction

Classification tree algorithms are built using real-world data sets, where the values of these data sets are the inputs for the classification algorithms. However, real-world data sets are never perfect and could suffer from several problems, one of these problems is the presence of noise. Data noise can be classified into class noise and attribute noise. Class noise occurs when class states are incorrectly labelled, while attribute noise refers to erroneous values in the attribute variables. Many studies have been conducted to evaluate the performance of classification algorithms when they are applied to noisy class or attribute variables, where only the former has been widely addressed. Class noise tends to have more effect on the performance of classification algorithms than attribute noise, while dealing with attribute noise is more difficult than dealing with class noise [69]. The performance of the classification tree algorithms based on such noisy data will definitely be affected, but some classification algorithms may be more robust against noise than other algorithms.

Erroneous entries in data sets are unavoidable in many data mining applications [68]. Erroneous or inaccurate attribute values are considered as an example of data noise, which may cause a negative impact on the performance of any classification

algorithm. In this chapter, we consider an application of the Direct NPI classification tree algorithm for Multinomial data (D-NPI-M), as presented in Chapter 4, with noisy data. We also assess the performance of the D-NPI-M classification algorithm in terms of its classification accuracy when different levels of random noise are added to the class or to attribute variables. We then compare its performance with the C4.5, NPI-M, A-NPI-M and IDM classification algorithms, introduced in Section 2.2.2, when different noise levels are present.

The rest of this chapter is organized as follows. In Section 5.2, an overview and examples of data noise are given, and some methods for adding noise to data sets are introduced. Section 5.3 presents a brief introduction to the literature on classification methods for noisy data. In Section 5.4, we present the results of the experimentation conducted to assess and compare the performance of the D-NPI-M classification algorithm with other classification algorithms, on data sets with different levels of added random noise. Finally, Section 5.5 presents some concluding remarks and possible future research topics.

5.2 Data noise

One of the most common data problems is data noise, with incorrect values being recorded for some attribute variables or for the class variable. The presence of noise is a common problem in real-world data sets that can suffer from corruptions which may affect the performance of classifiers constructed on the basis of such noisy data [58]. The noise might appear in data sets for several reasons. For examples, the inputs may be incorrectly measured, experts may wrongly describe the input values, damaged measurement devices might unintentionally be used, or data might get lost when transmitting and sorting the information [63]. It is expected that the classification algorithms based on noisy data sets will be less accurate compared to others based on noise-free data sets [61].

Data noise can be categorized into two types which are class noise and attribute noise. Class noise occurs when a class of an instance is incorrectly labelled, while attribute noise occurs when the values of one or more attribute variables of an instance are corrupted [21, 58]. For example, attribute noise could be because of some errors in attribute values, such as erroneous attribute values, missing values or incomplete values [69]. Examples of class noise sources include the same instance appearing twice or more but are labelled with different classes, or some instances being labelled with wrong classes [69]. The latter is the more common type in situations of class noise. In this chapter we refer to class noise as misclassifications, whereas attribute noise is referred to as erroneous attribute values. These types of noise examples are chosen to refer to class and attribute noise because these types occur most commonly in real-world data sets [69]. These two types have been also considered by Sáez et al. [58] for their extensive study tackling the problem of classification with noisy data. Sáez et al. [58] have applied different methods in order to add noise into data sets, we introduce these methods in this section.

Note that our aim in this chapter is neither to detect noise in training data, nor to identify and correct attribute or class noise from the training data. We mainly focus on checking how the performances of the classification algorithms are affected when they are applied to noisy data sets. In this chapter, we study the performance of the D-NPI-M algorithm when it is based on noisy data sets. We also compare it to other classical and imprecise algorithms that are applied to the same noisy data sets. In order to examine the degree of robustness of the classification algorithms with noisy data sets, we compare the performance of the algorithms built on the original (noise-free) data set with the performance of the algorithms built on the same data set with some added noise. If the classification accuracy results for noisy data are close to those for the data set without added noise, then the algorithm is considered to be robust. The robustness of classification tree algorithms is about its capability to generate classification trees that are not sensitive to corrupted data sets. The classification tree algorithm is more robust than the other algorithms if it generates trees that are less influenced by added noise to data sets. This way

of analysing the degree of robustness of classification algorithms in the presence of noise has been also used by Sáez et al. [58].

In order to test the performance and robustness of classification algorithms with noisy data we need a method to add noise to a data set. However, many available data sets may not contain much noise or we may not know which instances are noisy. Different methods for adding noise into data sets have been introduced in the literature. Adding noise to data sets enables us to check the effect of noisy data on the performance of classification algorithms and hence to identify which algorithms are robust with regard to noisy data, and also to search for possible solutions to improve the classification algorithms' performance on such noisy data sets. In this section we review some methods which are used in the literature to add noise to data sets, not only for the sake of introducing them, but also to justify our choice of the noise introduction method.

Zhu and Wu [70] applied two corruption mechanisms for adding noise to a class variable, which are *total random corruption* and *proportional random corruption*. For *total random corruption*, they randomly introduce noise to all classes, given a specific corruption level that is determined by users. Thus, based on this determined level, classes of instances are mislabelled. While for *proportional random corruption*, the class distribution remains constant during adding noise. In this method, given M classes, the class distribution is denoted as $P_1, P_2, ..., P_M$, where P_1 and P_M are the percentage of the most and least common classes, respectively, and $P_i \geq P_{i+1}$. To corrupt a particular noise level $x.100\%$, random noise is proportionally introduced to the different classes, where an instance labelled as i has $\frac{P_1}{P_i}.x.100\%$ chance to be mislabelled. With this method, the actual noise level could be less than the intended corruption level. For more details and explanations about these methods of adding noise to data sets, see Zhu and Wu [70].

Another example of a method for adding noise to class and attribute variables was introduced by Zhu and Wu [69]. To add noise to the class variable, given a

noise percentage $x\%$ and a pair of classes, an instance with the first class has an $x\%$ chance to be changed to the second class, so dose an instance of the second class. For adding noise to attribute variables, given a noise percentage $x\%$, the value of an attribute is randomly changed (approximately $x\%$ of the time) to other possible values, where each possible value is equally likely to be chosen. For numerical attribute variables, a random value between the minimum and maximum possible values is chosen. For more details about this method, we refer to Zhu and Wu [69]. Note that for adding noise in our experimental analysis in this chapter, we exclude the original value of the attribute or class variable from the assignments, hence, the actual percentage will be exactly the same as the theoretical percentage. This means that exactly $x\%$ of instances will have different values as there is no chance for their original values to be selected. More details about adding noise to class and attribute variables in our experimental analysis are given in Section 5.4.

Sáez et al. [58] have used four different methods in order to add a noise level $x\%$ to each data set. For class noise, they use a *uniform class noise scheme*, which corrupts the class labels of the instances by randomly replacing a class by another one from the available classes, and a *pairwise class noise scheme*, which labels instances of the largest class as belonging to the second largest class. For attribute noise, they use a *uniform attribute noise scheme* and the *Gaussian attribute noise scheme*. For the uniform attribute noise scheme, to corrupt an attribute with a particular noise level $x\%$, $x\%$ of the instances are selected and their attribute values are replaced by other values from the domain of the attribute. In this method, a uniform distribution is used to select the replacement value for both numerical and nominal attribute variables. The Gaussian attribute noise scheme is similar to the uniform attribute noise scheme, but it uses a Gaussian distribution. For more details about these methods see Sáez et al. [58].

A commonly used method for adding noise to data sets is presented by Mantas and Abellán [46, 47], Abellán and Masegosa [7, 9], Mantas et al. [48] and Abellán et al. [5]. In this method, they add a particular percentage of random noise to

the class variable or to attribute variables in the training set only, so the test set is left unmodified. The procedure to add noise into the class variable or attribute variables is as follows: first, they choose randomly a given percentage of instances in the training set, then, the values for these selected instances are randomly replaced by other possible values. Gray and Fan [36] have also used the same method when they test the performance of different classification methods with noisy data. They add random noise to the class variable by choosing a given percentage of instances, then their classes are changed to other classes. In this chapter we use the same method for adding noise to class and attribute variables. We randomly select a given percentage of instances and then we replace the values of these selected instances by other possible values. More details about applying the random noise to data sets for class or attribute variables in our work are given in Section 5.4.

5.3 Data noise impact on classification algorithms

In this section we briefly discuss some previous studies which have investigated the impact of data noise on classification algorithms. Many researchers have considered class noise or attribute noise in their studies, we briefly describe some of these studies and the concluded results. In the literature, class noise has been widely addressed, while attribute noise has received less attention.

Zhu and Wu [69] presented a systematic evaluation of the effect of noise in machine learning, by differentiating noise into two types, class noise and attribute noise. They investigated how the class and attribute noise could affect the classification accuracy of different classification algorithms, including the C4.5 algorithm. They put more focus on attribute noise, and investigated the relationship between attribute noise and its effects on the classification accuracy. They demonstrated that data noise severely affects the classification algorithms in many circumstances, where handling attribute noise is more difficult than class noise. The relationship between attribute noise and the classification accuracy is not clear as the impact

of attribute noise highly depends on the dependence between the class variable and attribute variables [69]. However, in real-world data sets, the class variable is much cleaner than attribute variables i.e. has less noise than attribute variables [69].

Mantas and Abellán [46] showed that classification trees based on imprecise probabilities give better results compared to other classic classification trees when they are applied to data sets with added random noise. They tested the performance of classification trees by adding different levels of random noise, up to 30%, to both class and attribute variables. Other researchers have also studied the effects of attribute noise or class noise on the classification accuracy of different classification algorithms [42, 58, 61]. However, in the literature, more attention has been paid to noise in the class variable. In this chapter, we investigate the performance of our newly presented classification method in case of noisy attribute and class variables.

Many researchers have studied the performance of classification algorithms when they are applied to data with noisy class variable [5, 7, 9, 21, 47, 48, 70]. Some recent studies have demonstrated that class noise has more impact on the performance of classification algorithms than attribute noise [21, 69]. Zhu and Wu [70] have studied the impact of class noise on classification algorithms for cost-sensitive classification. Cost-sensitive classification aims to minimize the misclassification cost instead of maximizing the classification accuracy as generally done in classification. They concluded that class noise may seriously affect the performance of cost-sensitive classification algorithms, particularly when incorrectly predicting some classes becomes extremely expensive.

Mantas and Abellán [47] have conducted several experimental studies to compare the performance of the Credal-C4.5 classification algorithm, which is based on imprecise probability, with other classical algorithms, the C4.5 and ID3 algorithms, when building classification trees with class noise. The comparison has been carried out by adding 10% and 30% of random noise to the class variable only. Their results concluded that without added noise, all classification algorithms have similar per-

formance, while the Credal-C4.5 classification algorithm obtains better performance than other algorithms when noise is added. It has also been found that classification tree algorithms based on imprecise probability, such as the Credal-C4.5 algorithm, tend to perform better than classic classification methods such as the C4.5 algorithm, when they are applied to data sets with added random noise to the class variable [5, 48].

Abellán and Masegosa [7, 9] have presented an application of bagging credal classification trees, which is based on imprecise probability using data sets with added random class noise. A bagging classifier is a method for generating multiple versions of a classifier, then to use these classifiers to get an aggregated classifier [18]. In [7, 9], Abellán and Masegosa have compared their method with similar schemes using classification trees built on the classic C4.5 algorithm, and they concluded that their method outperforms other Bagging methods on data sets with a noisy class variable. It would be of interest to compare the D-NPI-M classification algorithm with bagging methods, but such comparisons are left as topics for future research.

5.4 Experimental analysis

In this section we study the performance of the D-NPI-M classification tree algorithm when it is applied to noisy data sets. We also compare its performance with other classic and imprecise classification tree algorithms, the C4.5, NPI-M, A-NPI-M and IDM algorithms applied to the same noisy data sets. These algorithms have been introduced in Sections 2.2.2 and were also used for the comparisons in Chapter 4. We aim to assess the performance of these classification algorithms with different noise levels. In this experimental analysis, we have implemented both the NPI-M and A-NPI-M algorithms, but we exclude the A-NPI-M algorithm from the result sections as both give always the same results with the used data sets. These algorithms could give different results when they are applied to more data sets, as shown by Baker [14]. In this section, we first present the way that the experiments have been

Data set	N	Att.	Range of Att.	Classes
Lenses	24	4	2-3	3
Modified Iris	150	4	3	3
Monk's Problems-1	124	7	2-4	2
Nursery	12960	8	2-5	5
Post-Operative Patient	90	8	2-4	3
Qualitative-Bankruptcy	250	6	3	2

Table 5.1: Data sets description.

conducted and a brief description of the used data sets is given. We then clarify how the noise is added to either the class variable or to attribute variables. After that, we present and discuss the results of the performance of the classification tree algorithms with noisy class and attribute variables.

5.4.1 Experimental setup

In our experiments, we have used six data sets from the UCI Machine Learning Repository database [32]. The characteristics of these data sets are summarized in Table 5.1. These data sets have been also used in Section 4.5, hence, some more details about these data sets are given in Section 4.5. We used only six data sets because we have not fully automated the D-NPI-M algorithm for constructing the classification trees, which makes the application to each data set time-consuming. Different levels of random noise have been added to each data set, then six classification tree algorithms have been applied to each data set. We use the statistical software R for our experimentations [57]. All R packages used to assess the performance of these algorithms are introduced earlier in Section 4.5.

For these data sets, as in most of the real-world data sets, we do not know how much noise they contain, if any, or which instances may be noisy. Thus, we do not assume any particular level of noise in these data sets, hence, we consider these data sets as noise-free. Therefore, we implement a random corruption method in order to introduce some noise into these data sets. We add the following random noise levels to the attribute and class variables: 5%, 15% and 30%. These random

levels are selected following several researchers in the literature. It is reasonable to add noise up to 30% as in most cases data sets may not contain more noise. Many researchers in the literature have also added noise levels in their experiments up to 30% to either class or attribute variables [5, 7, 9, 44, 46, 47, 48].

To corrupt a class variable or attribute variables, i.e. adding noise into them, $x\%$ of the instances are selected, where x refers to the noise level we want to add. For adding noise to the class variable, $x\%$ of the instances in the training set are randomly selected, then their class labels are replaced by another class from the available classes, excluding the original class label. For adding noise to attribute variables, we randomly select $x\%$ of instances, then for each categorical attribute variable, we change the value of it to another possible value from the domain of the attribute, excluding the original attribute value. The noise levels are added to the training sets only, while the test sets are left unchanged. Adding noise to only training sets enables us to check the effects of different noise levels of the training set on the performance of the classification algorithms which are based on the data with the noise level, but which are tested on a test data set without noise. This way of adding noise allows direct comparison between the performance of the classification algorithms on equivalent test sets, for increased levels of noise in the training sets. Note that the D-NPI-M classification algorithm handles only categorical variables, hence, any continuous variable is converted to categorical variable before applying the classification algorithms on the training set. Unlike [61, 69], we exclude the original value from the random assignments for both class and attribute variables in order to ensure that $x\%$ of the training set will be changed.

In this experimental analysis, we have used 10-fold cross-validation scheme, introduced in Section 2.2.1. In this chapter, we only use the classification accuracy rate to assess the performance of the classification algorithms with noisy data sets. The classification accuracy rate is the most commonly used metric for assessing the performance of classification algorithms. We use only accuracy rate as we want to have a general insight about the performance of the D-NPI-M classification algo-

rithm and other algorithms with noisy data sets. However, using more metrics to assess the performance of these classification algorithms might lead to more conclusions.

The performance of the classification algorithms built on the original (noise-free) training set acts as a reference that could be directly compared with the performance of the classification algorithms obtained with different noisy levels of training data. In other word, in order to check the degree of robustness of the classification algorithms with noisy data sets, we compare the accuracy results of the classification algorithms from the original data sets with the performance (accuracy results) of classification algorithms from data sets with different levels of noise. Thus, the most robust classification algorithm is the one obtained the most similar classification accuracy results with noisy data sets, compared to its accuracy results with noise-free data sets. This method of comparing and analysing the degree of robustness has also been used by Sáez et al. [58]. We apply our experiments in two phases. First, we add the random noise only into the class variable, thereafter we add random noise to the attribute variables.

5.4.2 Experimental results for class noise

In this section, we present the results of the performance evaluation of the D-NPI-M algorithm with noisy data. We also compare the performance of the D-NPI-M algorithm to the performances of the C4.5, NPI-M and IDM algorithms, similarly to the approach taken in Chapter 4. In order to concentrate on the effects of noise on these classification algorithms, we study and present the results of this experimental analysis in two main categories. In this section, we present the results for the case of random noise added to the class variable. Secondly, the case of random noise added to the attribute variables is presented in Section 5.4.3.

To evaluate the impact of class noise, we have conducted experiments on six data sets which were also used in Section 4.5, where different levels of class noise

Data set	D-NPI-M	C4.5	NPI-M	IDM1	IDM2
Lenses	**81.67**	70	80.00	75.00	80.00
Modified Iris	**95.33**	92.67	90.00	92.67	92.67
Monk's Problems-1	**73.33**	69.17	69.17	69.17	69.17
Nursery	**90.37**	89.21	89.20	89.20	89.20
Post-Operative Patient	67.78	68.89	**71.11**	**71.11**	**71.11**
Qualitative-Bankruptcy	**99.60**	98	98.40	98.40	98.40

Table 5.2: Accuracy of classification trees on the original data sets.

are added to training sets only in the 10-fold set-up. We apply the different classification algorithms to these noisy data sets and evaluate the impact of class noise on the above classification algorithms on noise-free testing sets.

Table 5.2 shows the classification accuracy rate for all classification algorithms based on the original data sets, and Table 5.3 presents the classification accuracy rate for all classification algorithms based on noisy data sets with percentages of random noise equal to 5%, 15% and 30%, added to the class variable in the training sets. For the Lenses data set, there are clear differences in the classification accuracies among the classification algorithms on different noise levels, which is likely due to the fact that it is a small data set with only 24 instances. It is also noticed that the classification accuracies for all algorithms in this data set have increased with 5% noise level, but decreased after that with 15% and 30% noise levels. Such increase was also noticed in some papers from the literature when they add some noise levels to data sets [9, 46, 48]. This illustrates that it may actually happen that added noise leads to improved performance of a classification algorithm, which would just be due to randomness and is more likely to happen for small data sets and small noise levels. With low noise levels, some classification algorithms might slightly perform better than their performance on noise-free data sets, but looking at the performance of these algorithms on different noise levels enable us to check the robustness of these classification algorithms with noisy data sets. For all the noise levels considered, as well as without noise (as also seen in Chapter 4), the D-NPI-M algorithm tends to perform better than the other algorithms, but with

Data set	Noise	D-NPI-M	C4.5	NPI-M	IDM1	IDM2
Lenses	5%	**86.67**	85	85	80	85
Modified Iris	5%	**94**	**94**	**94**	**94**	**94**
Monk's Problems-1	5%	69.49	**71.67**	68.33	69.17	68.33
Nursery	5%	**90.37**	89.33	89.30	89.30	89.30
Post-Operative Patient	5%	67.78	**71.11**	70	**71.11**	**71.11**
Qualitative-Bankruptcy	5%	**99.2**	98.4	98.4	98.4	98.4
Lenses	15%	**76.67**	70	75	75	70
Modified Iris	15%	**93.33**	92.67	91.33	92.67	92.67
Monk's Problems-1	15%	**71.92**	67.5	67.5	69.17	67.5
Nursery	15%	**90.39**	89.17	89.17	89.17	89.14
Post-Operative Patient	15%	**73.33**	72.22	68.89	68.89	70
Qualitative-Bankruptcy	15%	**98.4**	98	**98.4**	**98.4**	**98.4**
Lenses	30%	**65**	55	60	55	**65**
Modified Iris	30%	**94.67**	**94.67**	93.33	93.33	**94.67**
Monk's Problems-1	30%	**67.37**	64.17	55	55.83	53.33
Nursery	30%	**90.29**	89.14	88.97	88.97	89.07
Post-Operative Patient	30%	**70**	62.22	62.22	62.22	63.33
Qualitative-Bankruptcy	30%	**94.8**	94.4	84.8	87.2	91.6

Table 5.3: Accuracy of classification trees on data sets with class noise.

equal accuracy, for this data set, to the IDM2 when 30% noise level is present.

Table 5.3 shows that with noise level 5%, the D-NPI-M algorithm obtains the best classification accuracy on three data sets and performs the same as the other classification algorithms for the Modified Iris data set. For the Monk's Problem-1 data set, the C4.5 algorithm slightly outperforms all other algorithms, and for the Post-Operative Patient data set, the C4.5 and IDM algorithms slightly outperform other algorithms at this noise level. However, for the Post-Operative Patient data set, the D-NPI-M algorithm obtains the best classification accuracy when the noise level increases to 15% and 30%.

The results in Table 5.3 show that the D-NPI-M algorithm performs well in the case of a noisy class variable, and mostly better than the other algorithms considered

here. In Table 5.3, for 15% noise level, the D-NPI-M algorithm achieves the highest classification accuracy in five out of six data sets with equal accuracy results to imprecise algorithms in the Qualitative-Bankruptcy data set. In Table 5.3, for the 30% noise level, the D-NPI-M mostly performs better than the other classification algorithms on all data sets with equal results of accuracy to the C4.5 algorithm and the IDM2 algorithm for the Modified Iris data set, and with equal accuracy results to the IDM2 algorithm for the Lenses data set.

To study the effects of adding noise to the class variable in more detail, Figure 5.1 shows the classification accuracy results of each classifier for each data set with the different noise levels. Note that these classification results are the average accuracy of applying 10-fold cross validation to each data set. Figure 5.1 shows that the D-NPI-M algorithm slightly outperforms the other algorithms in most cases. For the Post-Operative Patient data set, the D-NPI-M performs less well than the other algorithms with the original data sets and 5% noise levels, but it achieves the highest classification accuracy results with 15% and 30% noise levels. This is due to the randomness in the data sets, with and without noise, overall one expects more noise to lead to poorer results, but this can vary in specific applications. This data set is relatively small with 90 instances, hence, such difference in the accuracy results might happens. For this data set, we also notice that the performance of the C4.5 algorithm has increased with 5% and 15% noise levels. This behaviour of unexpected increasing in accuracy results with adding some noise was also noticed in [46], when the performance of the C4.5 algorithm has increased with noise level of 5%, besides giving higher accuracy results with 20% and 30% noise levels than the accuracy result with 10%. For the Nursery data set, all algorithms have a very similar performance with different noise levels, while the D-NPI-M algorithm slightly outperforms the other algorithms for all noise levels. For the Nursery data set, the classification accuracies are stable across all noise levels which could be because it is a large data set with 12960 observations, hence, the added noise may not clearly affect the classification accuracy.

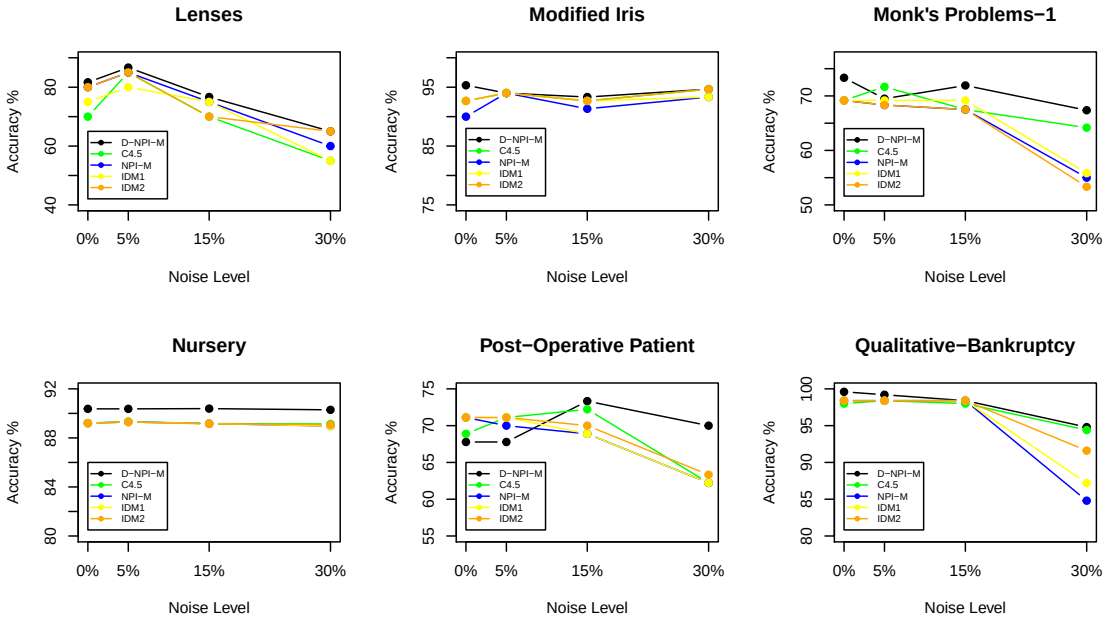

Figure 5.1: Classification accuracy of all classifiers applied to each data set with different noise levels added to the class variable.

The Monk's Problems-1, Post-Operative Patient and Qualitative-Bankruptcy data sets lead to big differences in the accuracy of the different algorithms in case of 30% class noise. However, for all of these three data sets, the D-NPI-M algorithm obtains the highest classification accuracy. This mostly better performance of the D-NPI-M algorithm in these three data sets indicates that it is more robust to higher levels of noise in the class variable than the other algorithms considered here. We conclude from this study that the D-NPI-M algorithm tends to be most robust with regard to class noise among the algorithms considered in this study.

It is also interesting to look at the average accuracy over all data sets. Figure 5.2 depicts the average results of classification accuracy of each classification algorithm when it is applied to data sets with percentages of random class noise equal to 0% (the original data sets), 5%, 15% and 30%. From these average results, the D-NPI-M classifier has the highest classification accuracy results when it is applied to the original data sets, 15% and 30% noise levels. However, for 5% class noise, the C4.5 algorithm performs slightly better than the other algorithms, but with very close

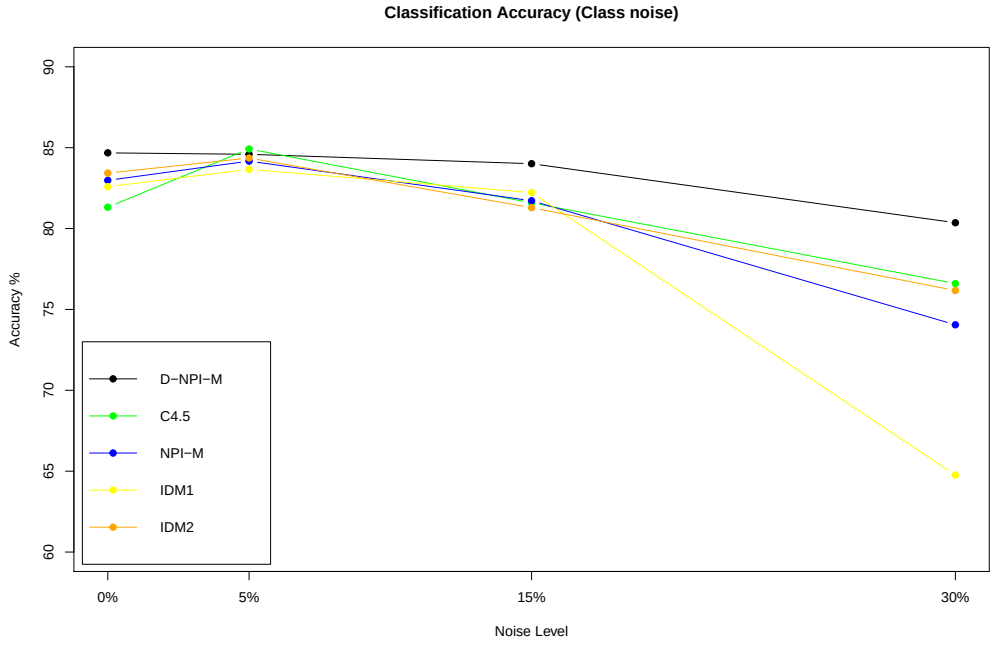

Figure 5.2: Average classification accuracy for all classifiers with different levels of random noise added to the class variable.

performance to the D-NPI-M algorithm. For 30% class noise, the differences in the average classification accuracy results are much bigger among the algorithms, with the D-NPI-M algorithm having best accuracy and the IDM1 algorithm the worst.

One reasonable method to analyse the degree of robustness of classification algorithms when they are applied to noisy data sets is to compare their performance with original data sets to their performance for different levels of added noise. If the accuracy is quite similar with and without added noise, then the algorithm is robust to noise in the data. Figure 5.2 shows that the D-NPI-M algorithm has a very similar average accuracy for the original data sets and the different noise levels considered, in this study it was more robust than the other algorithms. The C4.5, NPI-M and IDM2 algorithms also perform quite well and hence also show quite good robustness to noise in the class variable, but the IDM1 algorithm is clearly less robust at the 30% noise level.

5.4.3 Experimental results for attribute noise

In this section, we present the results of the performance evaluation of each classification algorithm with different added noise levels to attribute variables. Table 5.4 shows the classification accuracy rates of each algorithm when it is applied to data sets with 5%, 15% and 30% random noise added to attribute variables. Note that the noise is added to each attribute variable. The classification accuracy of each algorithm with noise-free data sets is shown in Table 5.2. Generally speaking, the results achieved from applying random noise to attribute variables are quite mixed, there is no single classification algorithm that seems to have the best performance with noisy attribute variables. Attribute variables could have some correlations between each other and may depend on the class variable. Thus, this correlation and dependence on the class variable may vary from an attribute to another, where the effects of adding noise to each attribute variable could have different impact on the performance of the classification algorithm. Therefore, analysing the performance of classification algorithms with added attribute noise is more difficult and complex than class noise. Zhu and Wu [69] also stated that the relationship between the added noise to each attribute variable and the performance of classification algorithm is unclear. They also did not reach to a clear conclusion about whether or not there are particular types of attribute variables that are sensitive to added noise or more sensitive than others.

Table 5.4 shows that, for 5% noise level, the D-NPI-M algorithm achieves the highest performance in all data sets with equal performance to the other algorithms in the Post-Operative Patient data set. With 15% and 30% noise levels, the results are quite mixed without a clearly best algorithm. With 15% noise level, the D-NPI-M algorithm performs slightly better than the other algorithms. The C4.5 algorithm performs the same as the D-NPI-M algorithm in four data sets. At 15% noise level, the D-NPI-M and C4.5 algorithms perform slightly better than the other algorithms. However, with 30% noise level, the NPI-M and IDM algorithms perform better than the C4.5 and D-NPI-M algorithms. At this noise level, for the Lenses data set, all algorithms have the same performance, while the NPI-M algorithm performs slightly

Data set	Noise	D-NPI-M	C4.5	NPI-M	IDM1	IDM2
Lenses	5%	**73.33**	65	68.33	68.33	68.33
Modified Iris	5%	**94**	92.67	90	92.67	92.67
Monk's Problems-1	5%	**72.95**	70.32	66.28	67.82	65.51
Nursery	5%	**90.21**	89.14	89.09	89.04	89.11
Post-Operative Patient	5%	**71.11**	**71.11**	**71.11**	**71.11**	**71.11**
Qualitative-Bankruptcy	5%	**99.6**	98.4	98.4	98.4	98.4
Lenses	15%	**83.34**	**83.34**	**83.34**	**83.34**	80.01
Modified Iris	15%	**94**	**94**	82.67	92.67	92.67
Monk's Problems-1	15%	74.04	**74.23**	70.19	72.63	69.42
Nursery	15%	**90.17**	89.24	89.21	89.21	89.21
Post-Operative Patient	15%	**70**	**70**	**70**	**70**	**70**
Qualitative-Bankruptcy	15%	**99.6**	**99.6**	98.4	98.4	98.4
Lenses	30%	**63.34**	**63.34**	**63.34**	**63.34**	**63.34**
Modified Iris	30%	92	92.67	**93.33**	91.33	91.33
Monk's Problems-1	30%	61.35	66.15	66.86	**68.59**	66.09
Nursery	30%	89.01	88.85	89.07	**89.14**	89.07
Post-Operative Patient	30%	70	70	**71.11**	**71.11**	**71.11**
Qualitative-Bankruptcy	30%	98	**98.8**	97.6	97.6	97.6

Table 5.4: Accuracy of classification trees on data sets with attribute noise.

better than the other algorithms for the Modified Iris and Post-Operative Patient data sets, with equal performance to the IDM algorithms for the Post-Operative Patient data set. Finally, the IDM1 algorithm performs well in four data sets with this noise level.

Figure 5.3 depicts the classification accuracy rates of each algorithm applied to each data set with different levels of noise added to each attribute variable. Unexpectedly, in the Lenses data set, all algorithms have higher classification accuracy with 15% noise level than their obtained accuracy results with the original data sets and 5% noise levels, but all classification algorithms have then obtained similar smaller accuracy results with 30% noise level. This increase, i.e. with 15% noise level, might be because of the small number of all instances in this data set and the random assignments as well. However, as seen in some previous researches with sim-

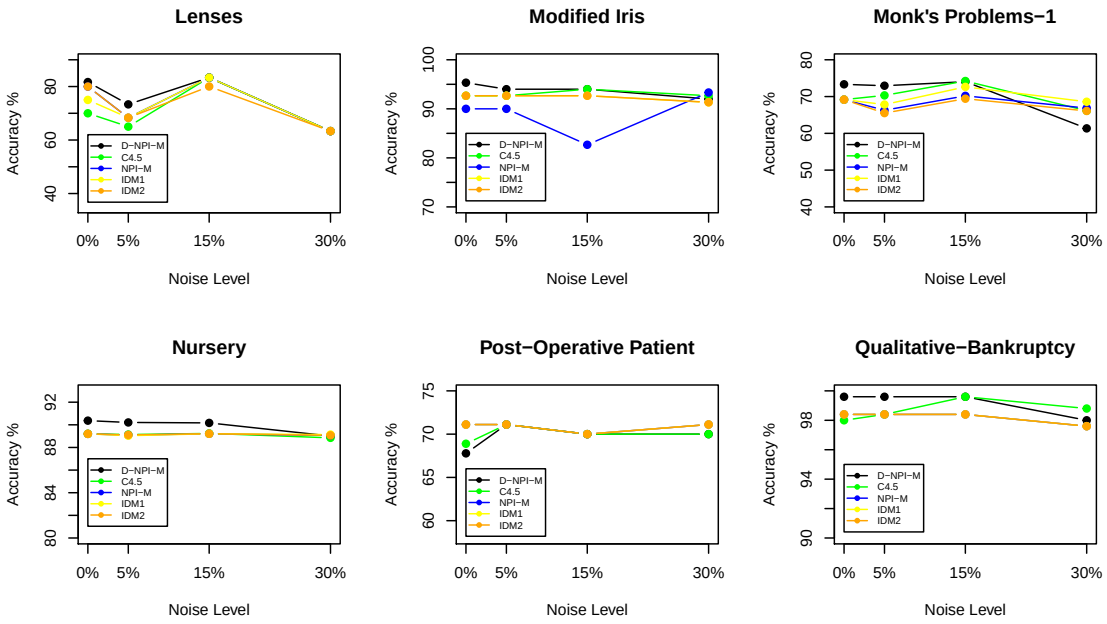

Figure 5.3: Classification accuracy of all classifiers applied to each data set with different noise levels added to attribute variables.

ilar analysis, such increasing in the classification accuracy with adding noise could occur. For the Nursery data set, the D-NPI-M algorithm attains the best average classification accuracy result for noise levels of 0% (the original data sets), 5% and 15%, while for 30% noise level all classification algorithms attain similar average accuracy results. For the remaining data sets, there is no clear behaviour for the algorithms in terms of their accuracy results with different noise levels. Although, the D-NPI-M algorithm seems to perform well with noisy attribute variables, we can not draw a clear conclusion about the results obtained from adding noise to attribute variables. However, the relationship between attribute noise and the classification accuracy is unclear [69]. We also find that in some situations, it might be hard to interpret the results achieved by adding noise to attribute variables.

According to the average results of classification accuracy shown in Figure 5.4, there is no clear superiority of any algorithm over the other algorithms. However, the D-NPI-M algorithm has the highest average classification accuracy with the original data sets and 5% noise level. With 15% noise level, the D-NPI-M algorithm has

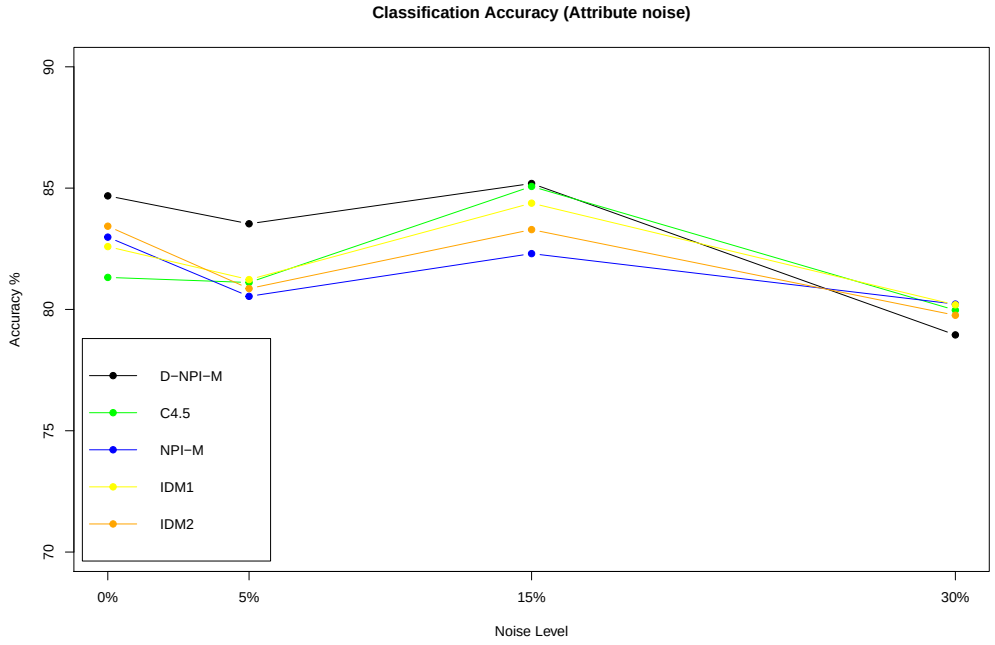

Figure 5.4: Average classification accuracy for all classifiers with different levels of random noise added to attribute variables.

the best average accuracy result with similar performance to the C4.5 algorithm. However, the D-NPI-M algorithm performs a bit worse than the other algorithms in terms of the average classification accuracy at the 30% noise level. Unexpectedly, the average classification accuracy results for all algorithms have slightly increased with 15% noise level compared to the average accuracy results with the original data sets and 5% noise levels. This increase could be because of the unexpected higher classification accuracy results achieved by all classification algorithms for Lenses data set with 15% noise level. However, for some data sets such increase might happen with adding noise levels.

To sum up, the performance results of the classification algorithms with different levels of random noise added to each attribute variable are quite mixed and vary among the algorithms and the noise levels. Although the D-NPI-M algorithm tends to perform quite well and attains the best accuracy results compared to other algorithms at the 5% and 15% noise levels, it is not the best performing algorithm at the 30% noise level.

The impact of adding noise to attribute variables could be unclear in terms of affecting the performance of the classification algorithms. This unclarity should be studied and investigated in much more detail. For example, the attribute variables could be correlated between each other and with the class variable. Of course, the correlation levels between each attribute variable and the class variable are not the same. Consequently, the impact of the added noise to the attribute variables differs from an attribute to another. For example, if we add a particular noise level to the attribute variables, this may cause more effects in terms of the classification accuracy on the attribute variable that are more related to the class variable, while the attribute variables, which are less related to the class variable may not clearly suffer from this noise level. A possible solution for this issue, which is left as a topic of future research, is to investigate separately each attribute variable with the class variable, by adding the noise levels to only one attribute variable and then to observe and analyse the performance results of the algorithm, and so on for the remaining attribute variables, i.e. to study the effect of noise per attribute variable.

In this section we have given an insight into the effects of adding random noise to attribute variables on the D-NPI-M algorithm, and compared its performance to other classification algorithms with added noise. It is still unclear how the impact of attribute noise may affect the performance of classification algorithms, although we have obtained some insight about this impact. Analysing more data sets and observing different types of attribute variables in a larger experiment could lead to better conclusions about the effects of attribute noise on the performance of classification algorithms. Zhu and Wu [69] studied the impact of attribute noise on the classification accuracy in an extensive experiment. However, they also found that cases with attribute noise are much more complicated than class noise.

5.5 Concluding remarks

This chapter presented an application of the D-NPI-M classification tree algorithm to data sets with noisy instances, the noise is presented either in the class variable or attribute variables. Data noise is often a problem in real-world data sets, which may affect the performance of classification algorithms based on such data sets. We briefly highlighted some studies that considered the problem of data noise on the performance of classification algorithms.

An experimental analysis has been carried out on data sets with different levels of random noise in order to asses the performance of the D-NPI-M classification tree algorithm with noisy data, and to compare its performance with some existing classification tree algorithms. We used the classification accuracy rate, i.e. percentages of correct classification instances, to measure and compare the performance of different classification algorithms. It will be of interest to consider more methods of the performance measures in the future. The results obtained in this chapter have shown that the D-NPI-M classification algorithm performs well on data sets with class noise, with its performance at different levels of random noise being similar to its performance on noise-free data sets. This indicates that it is robust with regard to class noise in the data. It performed slightly better than the other classification algorithms with different noise levels added to the class variable. For attribute noise, none of the classification tree algorithms considered performed noticeably better overall than the others. However, the D-NPI-M algorithm has the highest average accuracy rate with 5% and 15% of noise levels, but it did not perform the best with 30% noise.

This chapter gives an insight into the performance of the D-NPI-M algorithm with noisy data sets, but some further interesting topics remain to be investigated. In this chapter we check the performance of classification algorithms on noisy data sets by adding some noise levels to data sets. However, it is also interesting to detect and identify noisy instances in the training data sets, before trying to correct them before training the classification algorithms [21]. This could be conducted in

extensive experimentations which is beyond our scope in this thesis. In this chapter we have added random noise to training data sets, while considering other kinds of noise could also be possible and may return different results. It might also be possible to study and compare different methods of adding noise to the training data set and investigate their effects on the overall performance results. It will be also interesting to study the effect of adding noise to both class and attribute variables at the same time.

Another possible future work is to investigate further the impact of attribute noise on the performance of classification algorithms, as this type of noise has achieved less attention in the literature than class noise, and also our study did not reveal much insight. However, dealing with attribute noise is more complex than class noise due to a variety of reasons. For attribute noise, it might be better to consider separately adding noise to each attribute variable and then to assess the overall performance of the classification algorithm, and so on for the rest of the attribute variables. Finally, a more detailed experimental analysis with a wider range of data sets and more levels of noise for these data sets could lead to more conclusions about the performance of the D-NPI-M classification algorithm with noisy data sets. Conducting such experimentations with more focus on the characteristics of the data sets on which application of the D-NPI-M algorithm performs well, or less well, could lead us to identify characteristics of data sets for which the D-NPI-M algorithm will be appropriate.

Chapter 6

Conclusions and Future Research

In this chapter we first present a summary of this thesis, highlighting the main results presented. We then discuss some potential future research topics.

6.1 Conclusions

In this thesis we have introduced a new method to build classification trees, based on Nonparametric Predictive Inference (NPI). To start introducing this method to the area of classification, we developed this method only focusing on binary data, where both class variable and attribute variables are binary. Then, we have extended our method further to multinomial data, where class or attribute variables can have more than two categories. The novel classification approach presented in this thesis, which is completely based on NPI and does not use any other assumptions, is named Direct Nonparametric Predictive Inference classification (D-NPI).

To build classification trees based on the D-NPI classification method, we have introduced a new split criterion, which is called Correct Indication (CI). The CI split criterion is completely based on the lower and upper probabilities given by NPI for binary or multinomial data and it does not use any other assumptions or any added concepts such as entropy. The CI split criterion reports the strength of the evidence that each attribute variable will indicate the correct class state for new

instances, hence, it can be used to determine which attribute variable it is best to place at the root node when building classification trees, and so on for other internal nodes. The NPI lower and upper probabilities for CI are also used as a stopping criterion, by comparing them with the NPI lower and upper probabilities for CI in case no further attribute variable is used.

To assess the performance of classification trees built using the D-NPI classification method, we have carried out an experimental analysis on several data sets, and we have evaluated the performance using the classification accuracy rate, in-sample accuracy and tree size. We have compared the performance of our method with other classification tree methods from the literature. The results from our experimental analysis have shown that the D-NPI classification method performs well, slightly better than the other methods considered. We have presented an initial study of the use of our classification method when data noise is present. We have analysed the performance of our method when different random noise levels are added to either the class variable or to the attribute variables, and we compared its performance with other classification methods. Initial results suggest that our method performs well in case of noise in the class variable, compared to other classification methods, while there is little difference in the performance of all considered methods in case of noise in the attribute values.

6.2 Topics for future research

The work presented in this thesis leads to many possible future research directions. As a first step to develop the D-NPI classification method, we started with only binary data, then we have generalized it to multinomial data. Now, the D-NPI classification method can be extended further to involve other types of data such as real-valued data or ordinal data. In our work in Chapter 4 we assume that the categories are not ordered, however, it would be interesting to consider the work where the categories are ordered or if there is any other known structure or relationship

between the categories. Another idea for future research is to explore the use of the D-NPI classification method in random forests.

One important topic for future research is to develop the D-NPI classification trees with imprecise classification. In imprecise classification, trees might return a set of possible classes in their leaves rather than a single class, as is the case in classical classification trees and also in the new method presented in this thesis. It will also be interesting to develop the D-NPI classification method taking the cost of misclassification into account. Generally, the aim in classification is to maximize the total classification accuracy rate, however, in many practical scenarios the aim is to minimize misclassification costs. In particular in applications where the consequences of misclassifications vary considerably, including the costs in the analysis is crucial.

Further investigations of the D-NPI classification method on the used data sets with a more detailed analysis, and considering a larger number of available data sets, could lead to more insights about the performance of the D-NPI classification method, as well as aspects of data sets which may indicate that the method can be used successfully. Another useful research topic would be to evaluate and compare the performance of the D-NPI classification method with other classification methods using more evaluation measures, such as nonparametric tests for performance comparisons [31].

Finally, it will be interesting to further investigate the use of Correct Indication (CI) as a split criterion for the D-NPI classification method. In this thesis, we have stated two conditions for the CI split criterion in order to select any attribute variable for splitting. These two conditions are as follows: the NPI lower probability for CI has to be greater than the NPI lower probability for CI if no attribute variable is used, and the NPI upper probability for CI has to be greater than the NPI upper probability for CI if no attribute variable is used. However, we may consider the use of imprecision, i.e. the difference between the NPI lower and upper probabilities

for CI as a stopping rule, or only use one of the two conditions, or even combine them or consider different conditions. This could improve the performance of the D-NPI classification algorithm and lead to useful insights on when it is best to stop splitting in the D-NPI classification trees.

Bibliography

[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., Baker R.M. 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.

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

[4] Abellán J., Mantas C.J. and Castellano J.G. (2017). A random forest approach using imprecise probabilities. *Knowledge-Based Systems*, 134, 72–84.

[5] Abellán J., Mantas C.J. and Castellano J.G. (2018). AdaptativeCC4.5: Credal C4.5 with a rough class noise estimator. *Expert Systems with Applications*, 92, 363–379.

[6] Abellán J., Mantas C.J., Castellano J.G. and Moral-Garcia S. (2018). Increasing diversity in random forest learning algorithm via imprecise probabilities. *Expert Systems with Applications*, 97, 228–243.

[7] Abellán J. and Masegosa A.R. (2010). Bagging decision trees on data sets with classification noise. In *International Symposium on Foundations of Information and Knowledge Systems*, pp. 248–265. Springer.

[8] Abellán J. and Masegosa A.R. (2010). An ensemble method using credal decision trees. *European Journal of Operational Research*, 205, 218–226.

[9] Abellán J. and Masegosa A.R. (2012). Bagging schemes on the presence of class noise in classification. *Expert Systems with Applications*, 39, 6827–6837.

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

[11] Abellán J. and Moral S. (2005). Upper entropy of credal sets. Applications to credal classification. *International Journal of Approximate Reasoning*, 39, 235–255.

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

[13] Augustin T., Coolen F.P.A., De Cooman G. and Troffaes M.C. (2014). *Introduction to Imprecise Probabilities*. John Wiley and Sons, Chichester.

[14] Baker R.M. (2010). *Multinomial Nonparametric Predictive Inference: Selection, Classification and Subcategory Data*. Ph.D. thesis, Durham University.

[15] Bernard J.M. (2005). An introduction to the imprecise Dirichlet model for multinomial data. *International Journal of Approximate Reasoning*, 39, 123–150.

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

[17] Boole G. (1854). *An Investigation of the Laws of Thought on which are founded the Mathematical Theories of Logic and Probabilities*. Walton and Maberly, London.

[18] Breiman L. (1996). Bagging predictors. *Machine Learning*, 24, 123–140.

[19] Breiman L. (2001). Random forests. *Machine Learning*, 45, 5–32.

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

[21] Catal C., Alan O. and Balkan K. (2011). Class noise detection based on software metrics and ROC curves. *Information Sciences*, 181, 4867–4877.

[22] Chandra B. and Bhaskar S. (2011). A new approach for classification of patterns having categorical attributes. In *2011 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 960–964.

[23] Changala R., Gummadi A., Yedukondalu G. and Raju U. (2012). Classification by decision tree induction algorithm to learn decision trees from the class-labeled training tuples. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2, 427–434.

[24] Coolen F.P.A. (1998). Low structure imprecise predictive inference for Bayes' problem. *Statistics and Probability Letters*, 36, 349–357.

[25] Coolen F.P.A. (2006). On nonparametric predictive inference and objective Bayesianism. *Journal of Logic, Language and Information*, 15, 21–47.

[26] Coolen F.P.A. (2011). Nonparametric predictive inference. *International Encyclopedia of Statistical Science*, pp. 968–970.

[27] Coolen F.P.A. and Augustin T. (2005). Learning from multinomial data: a nonparametric predictive alternative to the Imprecise Dirichlet Model. In *International Symposium on Imprecise Probability: Theories and Applications*, volume 5, pp. 125–134.

[28] Coolen F.P.A. and Augustin T. (2009). A nonparametric predictive alternative to the Imprecise Dirichlet Model: the case of a known number of categories. *International Journal of Approximate Reasoning*, 50, 217–230.

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

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

[31] Demšar J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7, 1–30.

[32] Dua D. and Graff C. (2019). *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences. Http://archive.ics.uci.edu/ml.

[33] Elkhafifi F.F. and Coolen F.P.A. (2012). Nonparametric predictive inference for accuracy of ordinal diagnostic tests. *Journal of Statistical Theory and Practice*, 6, 681–697.

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

[35] Fink P. and Crossman R.J. (2013). Entropy based classification trees. In *International Symposium on Imprecise Probability: Theories and Applications*, volume 13, pp. 139–147.

[36] Gray J.B. and Fan G. (2008). Classification tree analysis using TARGET. *Computational Statistics and Data Analysis*, 52, 1362–1372.

[37] Hastie T., Tibshirani R. and Friedman J. (2009). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer, New York.

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

[39] Hornik K., Buchta C. and Zeileis A. (2009). Open-Source Machine Learning: R Meets Weka. *Computational Statistics*, 24, 225–232.

[40] James G., Witten D., Hastie T. and Tibshirani R. (2013). *An Introduction to Statistical Learning: with Applications in R*. Springer, New York.

[41] Kadhem M.H. and Zeki A.M. (2014). Prediction of urinary system disease diagnosis: A comparative study of three decision tree algorithms. In *2014 IEEE International Conference on Computer Assisted System in Health*, pp. 58–61.

[42] Khoshgoftaar T.M. and Van Hulse J. (2009). Empirical case studies in attribute noise detection. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 39, 379–388.

[43] Kohavi R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, volume 2, pp. 1137–1143.

[44] Li P., Wu X., Hu X., Liang Q. and Gao Y. (2010). A random decision tree ensemble for mining concept drifts from noisy data streams. *Applied Artificial Intelligence*, 24, 680–710.

[45] Maimon O.Z. and Rokach L. (2014). *Data Mining with Decision Trees: Theory and Applications*. World Scientific, Singapore.

[46] Mantas C.J. and Abellán J. (2014). Analysis and extension of decision trees based on imprecise probabilities: Application on noisy data. *Expert Systems with Applications*, 41, 2514–2525.

[47] Mantas C.J. and Abellán J. (2014). Credal-C4.5: Decision tree based on imprecise probabilities to classify noisy data. *Expert Systems with Applications*, 41, 4625–4637.

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

[49] Medjahed S.A., Saadi T.A. and Benyettou A. (2015). Urinary system diseases diagnosis using machine learning techniques. *International Journal of Intelligent Systems and Applications*, 5, 1–7.

[50] Moral S., Mantas C.J., Castellano J.G. and Abellán J. (2020). Imprecise Classification with Non-parametric Predictive Inference. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pp. 53–66. Springer.

[51] Moral-García S., Mantas C.J., Castellano J.G. and Abellán J. (2020). Non-parametric predictive inference for solving multi-label classification. *Applied Soft Computing*, 88, 106011.

[52] Moral-García S., Mantas C.J., Castellano J.G., Benítez M.D. and Abellán J. (2020). Bagging of credal decision trees for imprecise classification. *Expert Systems with Applications*, 141, 112944.

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

[54] Piatti A., Zaffalon M. and Trojani F. (2005). Limits of Learning from Imperfect Observations under Prior Ignorance: the Case of the Imprecise Dirichlet Model. In *International Symposium on Imprecise Probability: Theories and Applications*, volume 5, pp. 276–286.

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

[56] Quinlan J.R. (1993). C4.5: programs for machine learning. *Morgan Kuafmann*.

[57] R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

[58] Sáez J.A., Galar M., Luengo J. and Herrera F. (2013). Tackling the problem of classification with noisy data using multiple classifier systems: analysis of the performance and robustness. *Information Sciences*, 247, 1–20.

[59] Sahin Y., Bulkan S. and Duman E. (2013). A cost-sensitive decision tree approach for fraud detection. *Expert Systems with Applications*, 40, 5916–5923.

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

[61] Teng C.M. (1999). Correcting Noisy Data. In *Machine Learning*, pp. 239–248.

[62] Utkin L.V., Kovalev M.S. and Coolen F.P.A. (2020). Imprecise weighted extensions of random forests for classification and regression. *Applied Soft Computing*, 92, 106324.

[63] Vagin V. and Fomina M. (2011). Problem of knowledge discovery in noisy databases. *International Journal of Machine Learning and Cybernetics*, 2, 135–145.

[64] Walley P. (1991). *Statistical Reasoning with Imprecise Probabilities*. Chapman and Hall, London.

[65] Walley P. (1996). Inferences from multinomial data: learning about a bag of marbles. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58, 3–34.

[66] Weichselberger K. (2000). The theory of interval-probability as a unifying concept for uncertainty. *International Journal of Approximate Reasoning*, 24, 149–170.

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

[68] Zhang Y. and Wu X. (2010). Integrating induction and deduction for noisy data mining. *Information Sciences*, 180, 2663–2673.

[69] Zhu X. and Wu X. (2004). Class noise vs. attribute noise: A quantitative study. *Artificial Intelligence Review*, 22, 177–210.

[70] Zhu X. and Wu X. (2004). Cost-guided class noise handling for effective cost-sensitive learning. In *Fourth IEEE International Conference on Data Mining*, pp. 297–304. IEEE.