Contributions to Nonparametric Predictive Inference: Classification and Performance Evaluation

GHONEM, HADEER,ABDOU,ABDELKADER

Contributions to Nonparametric Predictive Inference: Classification and Performance Evaluation

Hadeer A. Ghonem

A Thesis presented for the degree of
Doctor of Philosophy

Statistics Group
Department of Mathematical Sciences
University of Durham
England
September 2025

Dedicated to

My Beloved Family

Contributions to Nonparametric Predictive Inference: Classification and Performance Evaluation

Hadeer A. Ghonem

Submitted for the degree of Doctor of Philosophy
September 2025

Abstract

Nonparametric predictive inference (NPI) is a statistical methodology which uses imprecise probability to quantify uncertainty. In imprecise probability, lower and upper probabilities are assigned to events to represent uncertainty about the events. NPI provides lower and upper probabilities for future events based on observed data.

Since NPI uses imprecise probability, methods based on the NPI approach provide predictions which are, by nature, imprecise, making it challenging to compare their performance directly with methods based on classical probability. This highlights the importance of studying the performance of the methods based on the NPI approach. Examining their performance when predictions take the form of a single value, an interval or a more general set of values, is important. This thesis contributes to NPI-based methods by investigating their performance across different scenarios of predictive inference.

In classification, the Direct Nonparametric Predictive Inference (D-NPI) method is based on the NPI approach. This method uses a splitting criterion, Correct Indication (CI), which relies on NPI lower and upper probabilities. Imprecise classification and multi-label classification are two important problems within the area of classification. In imprecise classification, a classifier, which is known as imprecise classifier, predicts a set of class labels rather than a single class label. In multi-label classification, a single data point in a dataset, which is known as an instance, can be associated with multiple labels simultaneously. This thesis applies D-NPI to ensemble methods, imprecise classification and multi-label classification, and investigates its performance across different classification problems. Experimental studies are conducted to evaluate the performance of the proposed methods using several measures and statistical tests. Further studies are

carried out to compare the performance of the proposed methods to other methods from the literature.

The results obtained from the proposed ensemble methods, bagging and random forest, indicate their effectiveness compared to the D-NPI method. The D-NPI-based random forest method performs well compared to the well known random forest method from the literature. In imprecise classification, some of the proposed imprecise classifiers have strong performance compared to other methods. In multi-label classification problems, the results, according to various evaluation measures, show that the best performance is observed for methods which are based on the NPI approach for most of the datasets considered.

This thesis further contributes to the new use of performance evaluation measures for imprecise probability inferences focusing on an NPI-based method for bivariate data. Performance measures for imprecise probability methods are required to consider both the accuracy and the imprecision of predictions. These measures include loss functions as well as a new interval score measure, designed with three weights to evaluate the performance of prediction intervals. This thesis investigates the performance of an NPI method for bivariate data in different scenarios using the introduced performance measures, and compares its performance with an alternative method. The results show that the performance measures are effective in assessing the performance of the method, enabling investigation of the accuracy and the imprecision of intervals, and comparing different scenarios. Using the interval score measure enables evaluating the performance in terms of both accuracy and precision, selecting its weights depends on the requirements of the application.

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

Writing this thesis has been a journey of learning and personal growth. First and foremost, I would like to say "Alhamdulillah".

I would like to express my deep appreciation and gratitude to my supervisors, Prof. Tahani Coolen-Maturi and Prof. Frank Coolen, for their invaluable guidance, patience, and support. They have been more than supervisors; they have been like family throughout my PhD journey, making this process not just a scholarly pursuit but a truly enriching experience. I am deeply grateful for all that I have learned from them.

My special thanks and love go to my beloved family, Mum, Dad, sisters, and brother, for their unconditional love, prayers, and endless encouragement. Their support during the most challenging moments, has kept me going.

I would like to extend my gratitude to my friends and colleagues for the discussions and their encouragement when I needed it the most. Their shared experiences have made my journey not only bearable but truly memorable. They have influenced my growth in ways I can't fully express.

I would also like to express my gratitude to Prof. Thomas Augustin at the University of Munich for welcoming me during my research visit. I am grateful for interesting discussions with him and the group at the University of Munich, allowing me to expand my insights.

I would also like to express my appreciation to the Engineering and Physical Sciences Research Council (EPSRC) for the financial support, which made this research possible. A special thank you goes to the University of Durham for providing me with the resources and academic environment to carry out this research.

Contents

Chapter 1

Preliminaries

1.1 Introduction

In the field of statistics, the assessment of the performance of statistical methods is a crucial aspect of model evaluation, ensuring that models and methods are both reliable and valid. By systematically assessing the statistical methods, decision-makers can ensure that the derived insights are neither biased nor misleading. Performance evaluation provides a framework for recognising valid models and refining methodologies to improve predictive accuracy. Whether applied in predictive methods or inferential statistics, performance evaluation plays a pivotal role in reducing overconfidence in model predictions. For example, in classification, model performance evaluation provides an estimate of model accuracy and reliability, reflecting its true performance rather than overestimating its effectiveness due to overfitting, thereby its efficiency can be improved.

Imprecise probability-based methods frequently provide indeterminate predictions, which makes comparing with classical methods that are based on precise probability models complicated and performance evaluation challenging. In particular, Nonparametric Predictive Inference (NPI) is a frequentist statistical framework that relies on minimal modelling assumptions and uses imprecise probability to quantify uncertainty [15]. NPI provides lower and upper probabilities for events involving a future random quantity based on observed data. As a result, predictions from NPI-based methods are basically imprecise. Therefore, studying the performance of NPI methods is challenging.

When assessing predictions, the evaluation method used depends on the type of the prediction. Prediction types include single-value predictions, set-based predictions, and

interval predictions. A single-value prediction provides a precise value but it risks being entirely incorrect if the prediction value deviates from the true value, including traditional classification methods [24]. A set-based prediction offers a set of possible values. The prediction is considered correct if the true value falls within this set, such as in imprecise classification [4]. Although this approach increases the possibility of a correct prediction, the set might be too large. An interval prediction provides a range within which the true value is expected to fall [89]. Evaluation methods assess whether the interval contains the true value while also considering the width of the interval. For example, in classification, an instance might be associated with a single class label. A classical classifier predicts a single class label, while an imprecise classifier predicts a set of class labels.

In classification, various algorithms have been developed based on the NPI approach [7, 12, 17]. Abellán et al. [7] constructed classification trees using NPI Model for Multinomial data (NPI-M), with the maximum entropy measure used for selecting the best attribute to split the trees. Alharbi [12] presented a recent method for classification that is entirely based on the NPI approach. This method, named Direct NPI classification algorithm (D-NPI), introduces a split criterion based on the NPI lower and upper probabilities, which is called Correct Indication (CI).

In real-world applications, datasets used to build classification trees often contain uncertainty, noise, and incomplete information. Traditional classification methods assign a single class label to each instance, which can lead to overconfident and potentially misleading predictions. Assigning a set of class labels rather than a single class label to an instance offers a more robust and cautious approach, addressing the limitations of the traditional classification methods.

In Multi-label classification, each instance can be associated with multiple class labels, rather than being assigned to a single class label. Multi-label classification addresses scenario in which class labels overlap, which describe the situation in which an instance is simultaneously assigned multiple labels.

This study aims to investigate different types of predictions resulting from NPI-based methods. It introduces an application of the D-NPI method to bagging and random forest methods, providing deeper insights into its performance. The study also presents extensions of the D-NPI method to imprecise classification. Furthermore, this thesis contributes to multi-label classification based on the D-NPI method, investigating its

applicability and effectiveness in complex scenarios.

A further method in this thesis considers problems with bivariate data, which is based on the NPI approach. Coolen-Maturi et al. [29] introduced a semi-parametric predictive method to combine NPI with bivariate parametric copulas. They used NPI for the marginals, followed by estimating the dependence structure between random quantities using a bivariate parametric copula. They conducted a comprehensive study to evaluate the performance of the method by deriving lower and upper NPI probabilities for events involving a bivariate future observation. This thesis introduces a novel use of performance measures to evaluate the semi-parametric predictive method, enabling a more detailed analysis from a different perspective.

This chapter establishes the necessary foundations for Chapters 2, 3, and 4, while the introduction of the semi-parametric predictive method is presented in Chapter 5. Section 1.2 presents an overview of imprecise probability. An overview of the NPI approach for different data types is presented in Section 1.3. Section 1.4 provides the main concepts and methods of classification. Section 1.5 introduces the D-NPI method, for both binary data and multinomial data. Finally, the structure of the thesis is outlined in Section 1.6.

1.2 Imprecise probabilities

The idea of using imprecise probabilities dates back to the middle of the 19th century and was first proposed by Boole [19]. Since then, imprecise probabilities have been applied in a wide area of statistics, including predictive inference [45, 67], artificial intelligence [112], and robust Neyman-Pearson testing [14, 73]. Recently, there has been increasing activity in this area by researchers from various backgrounds, leading to special issues in academic journals [42, 44].

In classical probability, a precise probability is assigned to an event, while imprecise probability provides an interval of lower and upper probabilities to the event, representing uncertainty about the event. Let Ω be a sample space, and let $\mathcal{A}$ be a set of events. Imprecise probability involves assigning an interval probability, $[\underline{P}(A), \overline{P}(A)]$, to an event A, where $0 \leq \underline{P}(A) \leq \overline{P}(A) \leq 1$ for all $A \in \mathcal{A}$, to quantify uncertainty rather than using precise probability. Here, $\underline{P}(A)$ denotes the lower probability, while $\overline{P}(A)$ represents the upper probability. In classical probability theory, a probability $P(A) \in [0, 1]$ is

assigned to each event A, which describes uncertainty about A, where $P(A)$ satisfies Komogorov's axioms [15]. Generally speaking, classical probability is a special case of imprecise probability when $\underline{P}(A) = \overline{P}(A)$ for all $A \in \mathcal{A}$. Absence of information about event A can be represented by $\underline{P}(A) = 0$ and $\overline{P}(A) = 1$. Weichselberger [110] defined the structure $\mathcal{M}$:

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

where $p(.)$ is a set function on $\mathcal{A}$ in classical probability theory which satisfies Kolmogorov's axioms. The lower and upper probabilities of the event A are defined as follows:

$$\underline{P}(A) = \underset{p(.)\in M}{inf}\ p(A), \quad \forall A \in \mathcal{A},$$

$$\overline{P}(A) = \underset{p(.)\in M}{sup}\ p(A), \quad \forall A \in \mathcal{A}.$$

The lower and upper probabilities are conjugated, that is $\underline{P}(A) = 1 - \overline{P}(A^c)$, where A^c is the complement of the event A. The lower probability $\underline{P}(A)$ reflects the information in favour of A, while $1 - \overline{P}(A)$ reflects the information in favour of A^c. If A and B are disjoint, lower probability is superadditive, and upper probability is subadditive,

$$\underline{P}(A \cup B) \geq \underline{P}(A) + \underline{P}(B) \quad \forall A, B \in \mathcal{A}, \text{ with } A \cap B = \phi,$$

and

$$\overline{P}(A \cup B) \leq \overline{P}(A) + \overline{P}(B) \quad \forall A, B \in \mathcal{A}, \text{ with } A \cap B = \phi.$$

In the real world, the information about an event A is often not sufficient for using precise probability to quantify uncertainty. Using the lower and upper probabilities is more informative, as it can reflect the amount of information. Hence, imprecise probability can offer advantages compared to precise probability, as it can allow for a better representation of the uncertainty about A.

1.3 Nonparametric predictive inference (NPI)

Nonparametric predictive inference (NPI) is a frequentist inference framework based on Hill's assumption $A_{(n)}$ [69, 70], it uses only few modelling assumptions enabled by using

imprecise probabilities to quantify uncertainty [15].

The assumption $A_{(n)}$ was first proposed by Hill [69] for statistical inference on future observations, without requiring specific prior knowledge of the underlying distribution. Let $X_1, X_2, ..., X_{n+1}$ be real-valued and exchangeable random quantities, and let n observations $x_{(1)} < x_{(2)} < ... < x_{(n)}$ represent the order statistics of data x_i, where $i = 1, 2, ..., n$. Define $x_{(0)} = -\infty$, and $x_{(n+1)} = \infty$ for simplicity of notation. Assume there are no ties between data points, where $x_i \neq x_j$ for all $i \neq j$. The future observation, X_{n+1}, is equally likely to fall within each open interval $I_j = (x_{(j-1)}, x_{(j)})$, where $j = 1, 2, ..., n+1$ with equal probability. Therefore, $A_{(n)}$ is defined as [71]:

$$P(X_{n+1} \in I_j) = \frac{1}{n+1}, \quad \text{for} \quad j = 1, 2, ..., n+1.$$

So the data $x_1, x_2, ..., x_n$ split the real-line into $n+1$ intervals and the probability that the future observation falls within any interval $(x_{(j-1)}, x_{(j)})$ is $1/(n+1)$. The NPI lower and upper probabilities for the event $X_{n+1} \in B$, are defined as

$$\underline{P}(X_{n+1} \in B) = \frac{1}{n+1}|\{j : I_j \subseteq B\}|,$$

and

$$\overline{P}(X_{n+1} \in B) = \frac{1}{n+1}|\{j : I_j \cap B \neq \phi\}|.$$

The NPI lower probability is obtained by considering only the probability mass that has to be in B, corresponding to a probability mass of $\frac{1}{n+1}$ for each interval I_j, provided that the entire interval is fully contained within B. The NPI upper probability is achieved by including all probability mass that could possibly lie within B, which corresponds to a probability mass of $\frac{1}{n+1}$ for each interval I_j, provided that the intersection of I_j and B is non-empty. For $m \geq 1$ future observations, Hill's assumptions $A_{(n)}, A_{(n+1)}, ..., A_{(n+m-1)}$ can be applied subsequently [13]. The need to use NPI methods arises from the need to make inferences based on observed data, using few assumptions. NPI is suitable in situations where there is limited information about the random quantities of interest. NPI provides an approach not only for predicting a single future observation but also for events involving multiple future observations. NPI methods have been applied in various applications in statistics [31]. NPI methods have been used for different types of data

such as Bernoulli data [30], multinomial data [33, 34], real-valued data [35], and ordinal data [37, 52].

1.3.1 NPI for circular data

Circular data, or directional data, are measured on a cyclical scale where the beginning and end points coincide, unlike linear data, which are measured on a straight scale with fixed endpoints. The $A_{(n)}$ assumption is not suitable for circular data, as they are not represented on the real line. The Circular-$A_{(n)}$ assumption, denoted by $\text{Ⓐ}_{(n)}$, was proposed for prediction of a future random quantity X_{n+1} for the circular data [31]. Assume n ordered circular observations x_i, $i = 1, 2, ..., n$, so that n intervals on a circle are created by these data. According to the assumption $\text{Ⓐ}_{(n)}$, a future random quantity, X_{n+1}, falls within each open interval $C_j = (x_j, x_{j+1})$, where $j = 1, 2, ..., n$ and $C_n = (x_n, x_1)$, with equal probability, as [31],

$$P(X_{n+1} \in C_j) = \frac{1}{n}, \quad \text{for} \quad j = 1, 2, ..., n.$$

This assumption is related to the exchangeability of $n+1$ circular random quantities, in the same way that $A_{(n)}$ is related to the exchangeability of real-valued random quantities.

In [31], Coolen presented the NPI lower and upper probabilities for a future observation, X_{n+1}, based on $\text{Ⓐ}_{(n)}$ assumption. Suppose B is a segment of the circle on which the circular data are represented. The NPI lower probability for the event $X_{n+1} \in B$, $\underline{P}(X_{n+1} \in B)$, is derived by considering only the probability mass assigned to intervals C_j that fall completely within B. The NPI upper probability, $\overline{P}(X_{n+1} \in B)$, is derived by summing all probability masses assigned to intervals C_j that have a non-empty intersection with B.

1.3.2 NPI for Bernoulli data

Coolen [30] presented NPI for Bernoulli data, which provides NPI lower and upper probabilities for m future observations based on n observed values. The NPI for Bernoulli random quantities relies on a latent variable representation, in which Bernoulli data are represented by real-valued observations with unknown threshold value such that observations on one side are considered successes and those on the other side failures [30].

Consider a sequence of $n+m$ exchangeable Bernoulli trials, where each trial can result in either a 'success' or a 'failure', and the observed data consists of s successes out of the first n observations. Let X_1^n be the random number of successes in the first n trials, then, due to the assumed exchangeability of the trials, $X_1^n = s$ provides a sufficient representation of the data for the inferences. Let X_{n+1}^{n+m} be the random number of successes in trials $n+1$ to $n+m$. Let $R_t = \{r_1, r_2, \ldots, r_t\}$, with $1 \leq t \leq m+1$ and $0 \leq r_1 < r_2 < \cdots < r_t \leq m$, and let $\binom{s+r_0}{s} = 0$ for ease of notation. The NPI upper probability for the event $X_{n+1}^{n+m} \in R_t$, given that $X_1^n = s$, for $s \in \{0, 1, \ldots, n\}$, is given by [30]:

$$\overline{P}(X_{n+1}^{n+m} \in R_t | X_1^n = s) = \binom{n+m}{n}^{-1} \sum_{j=1}^{t} \left[\binom{s+r_j}{s} - \binom{s+r_{j-1}}{s} \right] \binom{n-s+m-r_j}{n-s}. \tag{1.1}$$

The corresponding NPI lower probability can be derived using the conjugacy property, $\underline{P}(A) = 1 - \overline{P}(A^c)$, where A^c is the complementary event of A,

$$\underline{P}(X_{n+1}^{n+m} \in R_t | X_1^n = s) = 1 - \overline{P}(X_{n+1}^{n+m} \in R_t^c | X_1^n = s), \tag{1.2}$$

where $R_t^c = \{0, 1, \ldots, m\} \setminus R_t$. These NPI lower and upper probabilities were derived by Coolen [30] using Hill's assumptions $A_{(n)}, A_{(n+1)}, ..., A_{(n+m-1)}$ and counting arguments. The $\binom{n+m}{n}$ different orderings of the observed data and future observations, based on Hill's assumptions, are equally likely. For each of these orderings, the $n+m$ values of latent variables partition the real line into $n+m+1$ intervals, and the success-failure threshold value can be in any of these intervals. The counting method for these NPI lower and upper probabilities is explained in details in Aboalkhair [9]. For illustrative purposes, consider the case when $m = 1$, the NPI lower and upper probabilities for the event $X_{n+1}^{n+1} = 1 | X_1^n = s$, are given by:

$$\underline{P}(X_{n+1}^{n+1} = 1 | X_1^n = s) = \frac{s}{n+1}, \quad \overline{P}(X_{n+1}^{n+1} = 1 | X_1^n = s) = \frac{s+1}{n+1}. \tag{1.3}$$

Correspondingly, for the event $X_{n+1}^{n+1} = 0 | X_1^n = s$, the lower and upper probabilities are:

$$\underline{P}(X_{n+1}^{n+1} = 0 | X_1^n = s) = \frac{n-s}{n+1}, \quad \overline{P}(X_{n+1}^{n+1} = 0 | X_1^n = s) = \frac{n-s+1}{n+1}. \tag{1.4}$$

The case when $m = 1$ is presented here due to its relevance in Section 1.5. Further

details about NPI for Bernoulli quantities can be found in [30].

1.3.3 NPI for multinomial data

Coolen and Augustin [33, 34] have introduced Nonparametric Predictive Inference for Multinomial data model (NPI-M) as an alternative model to Walley's Imprecise Dirichlet Model (IDM) [107]. The NPI-M is based on the $(A)_{(n)}$ assumption [31], which is presented in Section 1.3.1. Coolen and Augustin developed the NPI-M for both scenarios when the number of categories is known [34] and when it is unknown [33]. Throughout this study, primary attention is given to the scenario in which the number of categories is known and denoted by K. Assume that a known number of categories $K \geq 3$. When $K = 2$, the NPI for Bernoulli data is more appropriate [30] as it results in less imprecision; however, the NPI-M can then also be used. In this section, the NPI-M is summarised for the case when K is known.

The NPI-M is based on a probability wheel representation, where each categorical observation represented by a single segment of the wheel. A probability wheel consists of equally sized segments, each is an area between two lines from the centre of the wheel to its circumference. Let n be the total number observations, each of the n observations is represented by a line from the centre of the wheel, partitioning the wheel into n-equally sized slices. Using the circular-A_n assumption, presented in Section 1.3, each future observation has an equal probability of $\frac{1}{n}$ for the falls into each of these slices. The assumption that each observation category is represented by one segment on the probability wheel implies that observations within the same category are positioned next to each other [34].

Assume that the number of possible categories is known and denoted by $K \geq 3$. Let the categories be represented by $C_1, C_2, \ldots, C_K$ and let n_i denote the number of observations in category C_i, where $i = 1, 2, \ldots, K$. Assume $n_i \geq 1$ for a single category C_i. The NPI lower and upper probabilities for events $X_{n+1} \in C_i$ are

$$\underline{P}(X_{n+1} \in C_i) = \max\left(0, \frac{n_i - 1}{n}\right) \tag{1.5}$$

and

$$\overline{P}(X_{n+1} \in C_i) = \min\left(\frac{n_i + 1}{n}, 1\right). \tag{1.6}$$

If a category C_i has not yet been observed, then the NPI lower probability for events $X_{n+1} \in C_i$ is 0, and the NPI upper probability for the events $X_{n+1} \in C_i$ is $\frac{1}{n}$. Further details and examples of NPI for multinomial data are provided by Coolen and Augustin [34].

1.4 Classification trees

Classification is one of the most effective predictive tools in machine learning and data mining. In a classification problem, the dataset comprises a set of instances, where each instance refers to a single observation characterised by multiple input attributes and an associated class label from a predefined set of class labels. The input attributes may be numerical, categorical, or a mixture of both. The class label is a categorical variable, indicating the class or category to which the instance belongs.

The aim of classification is to construct a predictive model designed to assign instances to a class label from a predefined set of classes based on their attributes. This process involves learning the patterns within labeled dataset to enable prediction of class label for new, unseen observations. Several supervised learning classifiers have been used to construct the predictive model, including regression analysis [41], decision trees [24], K-nearest neighbours [40], neural networks [18], random forest [22], and support vector machine [39]. Throughout this thesis, the terms classifier, classification algorithm, and algorithm are used interchangeably.

Classification trees are used when the class variable is categorical, such as fraud detection; if it is fraud or not. They are widely used due to their ease of visualisation and interpretability, thereby enabling straightforward understanding [65]. Also, they capable of handling both numerical and categorical data.

The structure of the classification tree is represented in Figure 1.1. In a classification tree, each node represents an attribute except for the leaf nodes; a class label is represented in the leaf node. The structure of the classification tree begins with the root node, followed by branches growing towards the subsequent internal nodes, and ends with the leaf nodes. A classification tree is constructed by recursively splitting the data, starting from the root node and creating a branching structure that leads to leaf nodes where class labels predictions are made. This structure enables the classification of a new instance

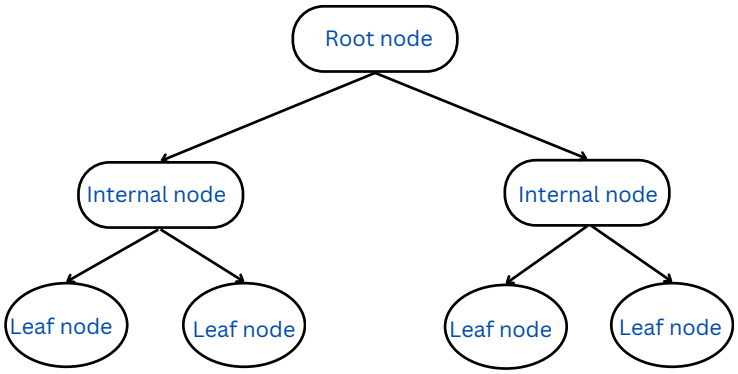


Figure 1.1: A structure of classification tree.

into a class label.

After constructing the classification tree, its performance is validated to ensure its predictive accuracy. To achieve this, the dataset is partitioned into two subsets: a training dataset, which is used to build the classification tree, and a test dataset, which is used to evaluate its performance. The training dataset enables the classifier to learn the pattern within the data, constructing the structure of the classification tree. While the test dataset is used for evaluating performance of the algorithm. .

Classification algorithms use various splitting criteria to divide the data based on the selected attribute variable. Splitting criteria refer to the measures used to determine the best attribute for splitting data at each node of a classification tree. These criteria aim to improve the purity of the resulting subsets. A node is considered pure when all instances it contains belong to the same class label. Among the various splitting criteria, those most commonly considered include criteria such as Gini impurity [24], information gain [93], gain ratio [94], imprecise information gain [5]. Alharbi [12] has recently introduced a splitting criterion called 'Correct Indication' (CI), which is based on the NPI approach and learns from data.

A variety of algorithms can be applied to construct the classification trees. Some of the most commonly algorithms include ID3, which uses the information gain criterion [94]; C4.5, an extension of ID3 that uses the gain ratio criterion [94]; and CART, which is based on the Gini impurity criterion [24]. Additionally, there are algorithms that use the imprecise information gain criterion and are based on imprecise probability models, such as Imprecise Dirichlet Model (IDM) [1] and Nonparametric Predictive Inference Model for multinomial data (NPI-M) [6].

During building classification trees using the C4.5 algorithm, the gain ratio splitting criterion is used to select the best attribute variable to split a node. The attribute variable with the highest gain ratio is selected and assigned to the node, then the dataset is divided into subsets according to the attribute values.

An essential concern in the construction of classification trees is to decide when to stop growing the trees. There are various methods used to stop building the trees, which are commonly grouped into two approaches: pre-pruning and post-pruning [53]. Pre-pruning approach stops tree construction during the growth phase by establishing stopping rules such as reaching a pure node, limiting the maximum depth of the tree, or having a minimum number of instances at a node. This approach aims to prevent the growth of the branches that seem to cause overfitting. Post-pruning allows the trees to grow fully, followed by reducing the size of the tree by removing less informative branches. Pre-pruning is commonly considered as a form of stopping criterion, as it stops the growth of the tree during its construction. While, post-pruning is often referred to simply as pruning. Selecting appropriate criteria to stop tree growth is important for balancing the tradeoff between tree complexity and predictive accuracy.

In order to establish the basis for the methods adopted in this study, the next section details the splitting and stopping criteria of a recent classification tree algorithm called Direct Nonparametric Predictive Inference (D-NPI) [12].

1.5 Classification trees using NPI

Abellán and Moral [5] have constructed classification trees based on imprecise probability using the IDM [1]. They used the maximum entropy measure to quantify uncertainty for building classification trees. In the IDM, prior assumptions about the data are assumed through a hyperparameter s [1]. The value of hyperparameter s determines how fast the lower and upper probabilities converge with increasing available data. For this reason, the NPI-M can provide an alternative to the IDM with using the maximum entropy measure to quantify uncertainty, as it does not assume any prior knowledge about the data. Abellán et al. [6] proposed two algorithms to derive the maximum entropy distribution using the NPI-M; the NPI-M algorithm and an approximation of the NPI-M, which is called A-NPI-M algorithm. Abellán et al. [7] have constructed classification trees using

the NPI-M, with using the maximum entropy measure to quantify uncertainty. It was shown that constructing classification trees with the IDM has a high dependence on the parameter used [7]. Further details and comparisons with NPI-M and A-NPI-M algorithms can be found in [7, 17].

A recent method for classification has been introduced by Alharbi [12], which is called D-NPI algorithm. This classification algorithm uses a new splitting criterion, correct indication (CI). This section presents the D-NPI classification algorithm and CI splitting criterion. The use of CI splitting criterion to construct classification trees is presented in this section. This thesis contributes to the application of the D-NPI algorithm to bagging and random forest in Chapter 2, to imprecise classification in Chapter 3, and to multi-label classification in Chapter 4. Section 1.5.1 provides an overview of the D-NPI algorithm for Bernoulli data, while Section 1.5.2 presents the method for multinomial data.

1.5.1 Direct-NPI for binary data

The D-NPI classification algorithm has recently been introduced by Alharbi [12] for both binary and multinomial data. For binary data, the author focused on scenarios where both the attribute variables and the class variable were binary. For the multinomial data setting, attribute variables or class variable can have multiple values. The D-NPI algorithm uses the splitting criterion CI, which is based on the NPI approach and does not use any additional concepts such as entropy. The CI reports the strength of evidence provided by each attribute variable, based on the data. The CI helps build more flexible and cautious models. The CI can be directly applied to categorical data or to continuous data after converting it to categorical data using a discretisation method. The tree construction process in the D-NPI algorithm follows a procedure similar to that of the C4.5 algorithm; but, it uses the CI splitting criterion to select the best attribute at each node. The performance of the D-NPI algorithm was evaluated through an experimental study and compared to other classification methods from the literature [12]. The results showed that the D-NPI algorithm performed slightly better than the other algorithms in terms of classification accuracy (the performance of the classification method on the testing dataset) and in-sample accuracy (the performance of the classification method on the training dataset). For the previous reasons, this criterion is interesting to investigate

in this thesis.

Consider a dataset with n instances that can take two values, 0 and 1, and with T binary attribute variables. Let x_i represent the attribute variables, where $i \in \{1, 2, \ldots, T\}$. The binary class variable is denoted by $C \in \{0, 1\}$. Let n^0 be the total number of instances classified as 0, and n^1 be the total number of instances classified as 1. The total number of instances of attribute x_i with the value 1 is denoted by $n(x_i = 1)$, while $n^1(x_i = 1)$ represents the total number of instances of attribute x_i with the value 1 and classified as 1. Similarly, $n^0(x_i = 1)$ denotes the total number of instances where the attribute x_i has the value 1 and $C = 0$. Let $n^0(x_i = 0)$ be the total number of instances which are classified as $C = 0$ and have the attribute value $x_i = 0$, and let $n(x_i = 0)$ be the total number of instances which have the attribute value $x_i = 0$.

Using NPI for Bernoulli data [30], the NPI lower and upper probabilities for the event that a future instance, is classified as $C_{n+1} = 1$ given that its attribute variable is $x_{n+1,i} = 1$, can be derived, where C_{n+1} is the unknown class label for the future instance and x_{n+1} is its attribute variable. The NPI lower and upper probabilities are

$$\underline{P}(C_{n+1} = 1 | x_{n+1,i} = 1) = \frac{n^1(x_i = 1)}{n(x_i = 1) + 1}, \tag{1.7}$$

and

$$\overline{P}(C_{n+1} = 1 | x_{n+1,i} = 1) = \frac{n^1(x_i = 1) + 1}{n(x_i = 1) + 1}. \tag{1.8}$$

By the conjugacy property, the NPI lower and upper probabilities for the event $C_{n+1} = 0 | x_{n+1,i} = 1$ are

$$\underline{P}(C_{n+1} = 0 | x_{n+1,i} = 1) = 1 - \overline{P}(C_{n+1} = 1 | x_{n+1,i} = 1) = \frac{n^0(x_i = 1)}{n(x_i = 1) + 1}, \tag{1.9}$$

and

$$\overline{P}(C_{n+1} = 0 | x_{n+1,i} = 1) = 1 - \underline{P}(C_{n+1} = 1 | x_{n+1,i} = 1) = \frac{n^0(x_i = 1) + 1}{n(x_i = 1) + 1}. \tag{1.10}$$

Similarly, the NPI lower and upper probabilities for the event that the future instance is classified as $C_{n+1} = 0$ given that the attribute value is $x_{n+1,i} = 0$ are

$$\underline{P}(C_{n+1} = 0 | x_{n+1,i} = 0) = \frac{n^0(x_i = 0)}{n(x_i = 0) + 1}, \tag{1.11}$$

and

$$\overline{P}(C_{n+1} = 0|x_{n+1,i} = 0) = \frac{n^0(x_i = 0) + 1}{n(x_i = 0) + 1}. \tag{1.12}$$

The NPI lower and upper probabilities for the event $C_{n+1} = 1|x_{n+1,i} = 0$ are derived using the conjugacy property as

$$\underline{P}(C_{n+1} = 1|x_{n+1,i} = 0) = 1 - \overline{P}(C_{n+1} = 0|x_{n+1,i} = 0) = \frac{n^1(x_i = 0)}{n(x_i = 0) + 1}, \tag{1.13}$$

and

$$\overline{P}(C_{n+1} = 1|x_{n+1,i} = 0) = 1 - \underline{P}(C_{n+1} = 0|x_{n+1,i} = 0) = \frac{n^1(x_i = 0) + 1}{n(x_i = 0) + 1}. \tag{1.14}$$

These NPI lower and upper probabilities report the strength of the evidence for the class label of the future instance, assuming to be exchangeable with the n instances in the dataset. Alharbi [12] derived the NPI lower and upper probabilities for the event that each attribute leads to the CI, aiming for the highest values for both the NPI lower and upper probabilities for CI.

Let p be the probability that the attribute x_i has the value 1, hence, $p = P(x_i = 1)$. Using NPI for Bernoulli data [30], presented in Section 1.3.2, then

$$p \in \left[\frac{n(x_i = 1)}{n + 1}, \frac{n(x_i = 1) + 1}{n + 1}\right], \tag{1.15}$$

where $n(x_i = 1)$ is the total number of instances of the attribute x_i which have the value 1, and n is the total number of instances in the dataset. Different attribute values occur with different frequencies in the dataset leads to using the theorem of total probability to calculate weighted average over all values of using p as weights. The total probability theorem states that if $B_1, B_2, \ldots, B_n$ is a partition of the sample space, then for any event A, the probability of A is expressed as $P(A) = \sum_{i=1}^{n} P(A \mid B_i)\, P(B_i)$. The NPI lower probability for the event that the attribute x_i leads to CI is derived by taking the lower probabilities for the events $C_{n+1} = 0 \mid x_{n+1,i} = 0$ and $C_{n+1} = 1 \mid x_{n+1,i} = 1$ to minimise the weighted average [12]:

$$\underline{P}_i(CI) = \min_p \left(\frac{n^0(x_i = 0)}{n(x_i = 0) + 1}\,(1 - p) + \frac{n^1(x_i = 1)}{n(x_i = 1) + 1}\,p\right). \tag{1.16}$$

The value of p which minimises $\underline{P}_i(CI)$ is:

$$p = \begin{cases} \dfrac{n(x_i=1)+1}{n+1} & \text{if } \dfrac{n^0(x_i=0)}{n(x_i=0)+1} \geq \dfrac{n^1(x_i=1)}{n(x_i=1)+1}, \\ \dfrac{n(x_i=1)}{n+1} & \text{otherwise.} \end{cases} \tag{1.17}$$

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

$$\overline{P}_i(CI) = \max_p \left(\frac{n^0(x_i=0)+1}{n(x_i=0)+1}\,(1-p) + \frac{n^1(x_i=1)+1}{n(x_i=1)+1}\,p \right), \tag{1.18}$$

which is achieved for

$$p = \begin{cases} \dfrac{n(x_i=1)+1}{n+1} & \text{if } \dfrac{n^0(x_i=0)+1}{n(x_i=0)+1} \leq \dfrac{n^1(x_i=1)+1}{n(x_i=1)+1}, \\ \dfrac{n(x_i=1)}{n+1} & \text{otherwise.} \end{cases} \tag{1.19}$$

The NPI lower and upper probabilities for the event that the attribute x_i leads to CI are calculated for each attribute variable. In [12], when constructing the classification tree, it is assumed that $\frac{n^1(x_i=1)}{n(x_i=1)} \geq \frac{n^1(x_i=0)}{n(x_i=0)}$, ensuring that the positive attribute value corresponds to the positive class label. Relabelling attribute values may be required during tree construction to maintain link between positive attribute values and the positive class label, as the dataset is partitioned across different tree stages.

After presenting the CI splitting criterion used to construct the classification tree, the D-NPI algorithm uses a stopping criterion introduced by Alharbi [12]. Let $\underline{P}(NA)$ and $\overline{P}(NA)$ be the NPI lower and upper probabilities for CI if no attribute variable used, respectively. These NPI lower and upper probabilities correspond to stating the most frequent value in the class variable. Using NPI for Bernoulli data, presented in Section 1.3.2, the NPI lower and upper probabilities for CI if no attribute variable used are [12]

$$[\underline{P}(NA), \overline{P}(NA)] = \left[\frac{n_{\max}}{n+1}, \frac{n_{\max}+1}{n+1} \right], \tag{1.20}$$

where $n_{\max}$ is the number of instances of the most frequent class and n is the total number of instances. When building the classification tree, at each node, for each attribute variable x_i, the NPI lower and upper probabilities for CI for that attribute are derived using Equations (1.16) and (1.18). After comparing the NPI lower and upper for CI for

all attributes, the attribute with the highest NPI lower and upper probabilities for CI is selected. The attribute variable x_i with the highest NPI lower and upper probabilities for CI $\underline{P}_i^*(CI)$ and $\overline{P}_i^*(CI)$ is selected to split the tree at a node if the following conditions are satisfied

$$\underline{P}_i^*(CI) > \underline{P}(NA) \quad \text{and} \quad \overline{P}_i^*(CI) > \overline{P}(NA). \tag{1.21}$$

If two or more attribute variables satisfy the conditions in (1.21) and have CI intervals which overlap, the attribute variable with the highest upper probability for CI is selected. In the case where multiple attribute variables that satisfy (1.21) and have the same NPI lower probabilities as well as the same NPI upper probabilities, any of these attribute variables can be selected. At the node, if there is no attribute that satisfies the conditions in (1.21), the algorithm stops building the tree. This thesis presents a new application of the D-NPI method for binary data to bagging and random forest, with an investigation of its performance in Chapter 2.

1.5.2 Direct-NPI for multinomial data

In many classification problems, such as finance, medical diagnosis, or image recognition, attribute variables can be categorical with multiple levels. For example, in medical diagnosis, patient data often include blood type or symptom categories (e.g., mild, moderate, severe). Alharbi [12] has recently constructed classification trees using the CI splitting criterion for multinomial data.

Suppose that there is a dataset with T attribute variables and n instances, and let x_i represent the attribute variables, where $i \in \{1, 2, \ldots, T\}$ and let v_j^i be the values in the attribute variable, x_i, where $j \in \{1, 2, \ldots, k_i\}$. The class variable is denoted by $C \in \{c_1, c_2, \ldots, c_R\}$, with all class labels being observed in the dataset.

Let n^{c_r} be the number of instances in the dataset classified as class c_r, for $r \in \{1, 2, \ldots, R\}$, then the total number of instances in the dataset is $n = n^{c_1} + n^{c_2} + \cdots + n^{c_R}$. Let $n(x_i = v_j^i)$ be the total number of instances where the attribute variable x_i has the value v_j^i, and let $n^{c_r}(x_i = v_j^i)$ be the total number of instances where the attribute variable x_i has the value v_j^i and is associated with class c_r. The NPI lower and upper probabilities for the event that a future instance is assigned to class c_r for $r \in \{1, 2, \ldots, R\}$, given that its attribute variable values are v_j^i, $x_{n+1,i} = v_j^i$, for $i \in \{1, 2, \ldots, T\}$, and $j \in \{1, 2, \ldots, k_i\}$,

are

$$\underline{P}(C_{n+1} = c_r | x_{n+1,i} = v_j^i) = \frac{n^{c_r}(x_i = v_j^i)}{n(x_i = v_j^i) + 1}, \tag{1.22}$$

and

$$\overline{P}(C_{n+1} = c_r | x_{n+1,i} = v_j^i) = \frac{n^{c_r}(x_i = v_j^i) + 1}{n(x_i = v_j^i) + 1}. \tag{1.23}$$

For each attribute variable, each value is linked with the class label most frequently associated with it [12]. To clarify, the value v_j^i is linked to the class label c_r that mostly occurs for this attribute value, where $(C = c_r | x_i = v_j^i)$ are the conditional events of interest according to the D-NPI algorithm [12].

Let $p_{i,j}$ be the probability that the attribute x_i has the observed values v_j^i, hence, $p_{i,j} = P(x_i = v_j^i)$, for $j \in \{1, 2, \ldots, k_i\}$, and $i \in \{1, 2, \ldots, T\}$, so $\sum_{j=1}^{k_i} p_{i,j} = 1$. Using NPI for multinomial data [34], presented in Section 1.3.3, then

$$p_{i,j} \in \left[\frac{n(x_i = v_j^i) - 1}{n}, \frac{n(x_i = v_j^i) + 1}{n} \right]. \tag{1.24}$$

Using the total probability theorem, the NPI lower probability for the event that the attribute x_i leads to CI is given by considering the lower probabilities for the events $C_{n+1} = c_1 \mid x_{n+1,i} = v_1^i, \ldots, C_{n+1} = c_R \mid x_{n+1,i} = v_{k_i}^i$ to minimise the weighted average [12]:

$$\underline{P}_i(CI) = \min_{p_{i,j} \in \mathcal{P}} \left(\sum_{j=1}^{k_i} \frac{n^{c_r}(x_i = v_j^i)}{n(x_i = v_j^i) + 1} \, p_{i,j} \right), \tag{1.25}$$

with

$$\mathcal{P}_i = \left\{ p | p_{i,j} \in \left[\frac{n(x_i = v_j^i) - 1}{n}, \frac{n(x_i = v_j^i) + 1}{n} \right] \forall j \in \{1, 2, ..., k_i\}, \sum_{j=1}^{k_i} p_{i,j} = 1 \right\}. \tag{1.26}$$

The corresponding NPI upper probability for the event that the attribute x_i leads to CI is

$$\overline{P}_i(CI) = \max_{p_{i,j} \in \mathcal{P}} \left(\sum_{j=1}^{k_i} \frac{n^{c_r}(x_i = v_j^i) + 1}{n(x_i = v_j^i) + 1} \, p_{i,j} \right). \tag{1.27}$$

The NPI lower and upper probabilities for CI are calculated for each attribute x_i, for $i \in \{1, 2, \ldots, T\}$. In [12], the NPI lower probability $\underline{P}_i(CI)$ for each attribute variable x_i has been obtained when k_i is even or odd, for $i \in \{1, 2, \ldots, T\}$ after ordering the attributes increasingly according to their fraction values $\frac{n^{c_r}(x_i = v_j^i)}{n(x_i = v_j^i) + 1}$. For k_i even, the NPI

lower probability, $\frac{n(x_i=v_j^i)-1}{n}$, is assigned to all attribute variable values, and there are k_i remaining probabilities each having a value of $\frac{1}{n}$. Then, a probability of $\frac{2}{n}$ is assigned to the attribute variable values v_1^i to $v_{\frac{k_i}{2}}^i$. When k_i is odd, after assigning the NPI lower probability $\frac{n(x_i=v_j^i)-1}{n}$ to all attribute variable values, a probability of $\frac{2}{n}$ is assigned to the values from v_1^i to $v_{\frac{k_i}{2}-\frac{1}{2}}^i$. Following that, the last remaining probability $\frac{1}{n}$ is assigned to the values $v_{\frac{k_i}{2}+\frac{1}{2}}^i$.

In order to derive the NPI upper probability $\overline{P}_i(CI)$ for each attribute x_i, for $i \in \{1, 2, \ldots, T\}$, Alharbi [12] assigned probabilities to attribute variable values considering k_i odd and even after ordering the attributes increasingly according to their fraction values $\frac{n^{cr}(x_i=v_j^i)+1}{n(x_i=v_j^i)+1}$. For k_i even, the NPI lower probability, $\frac{n(x_i=v_j^i)-1}{n}$, is first assigned to all the values. There will be remaining k_i probabilities, each with a value of $\frac{1}{n}$. A probability of $\frac{2}{n}$ is then assigned to $v_{\frac{k_i}{2}+1}^i$ to $v_{k_i}^i$. If k_i is odd, a probability of $\frac{n(x_i=v_j^i)-1}{n}$ is assigned to all the values. The probability $\frac{2}{n}$ is assigned to the values from $v_{\frac{k_i}{2}+\frac{3}{2}}^i$ to $v_{k_i}^i$, followed by assigning the last remaining probability $\frac{1}{n}$ to the value $v_{\frac{k_i}{2}+\frac{1}{2}}^i$.

Alharbi [12] constructed classification trees using the D-NPI algorithm following a similar process as in the C4.5 algorithm, but using the CI splitting criterion to select the best attribute at each node. During building a classification tree using the D-NPI algorithm, the NPI lower and upper probabilities for each attribute x_i for CI are derived, and a stopping criterion is used to determine when to stop building the tree. Alharbi [12] introduced a stopping criterion when using the CI splitting criterion for multinomial data. Let $\underline{P}(NA)$ and $\overline{P}(NA)$ be the NPI lower and upper probabilities for CI if no attribute variable is used. Using NPI for multinomial data, presented in Section 1.3.3, the NPI lower and upper probabilities are

$$\underline{P}(NA) = \max\left(0, \frac{n_{\max} - 1}{n}\right), \tag{1.28}$$

and

$$\overline{P}(NA) = \min\left(\frac{n_{\max} + 1}{n}, 1\right), \tag{1.29}$$

where $n_{\max}$ is the number of instances with the most frequent class label and n is the total number of instances. When building a classification tree, the attribute variable x_i with the highest NPI lower and upper probabilities $\underline{P}_i^*(CI)$ and $\overline{P}_i^*(CI)$ compared to the

other attributes, and satisfies the following conditions is selected:

$$\underline{P}_i^*(CI) > \underline{P}(NA) \quad \text{and} \quad \overline{P}_i^*(CI) > \overline{P}(NA). \tag{1.30}$$

As in CI for binary data, if two or more attribute variables meet the conditions (1.30) and have overlapping NPI lower and upper probabilities for CI, the attribute variable with the highest upper probability for CI is selected. In the case where multiple attribute variables that satisfy (1.30) and have the same NPI lower probabilities as well as the same NPI upper probabilities, any of these attribute variables can be selected. A node is not split further if there is no attribute satisfies the conditions in (1.30).

The pseudocode of the D-NPI algorithm based on the steps which introduced by Alharbi [12] is presented in Appendix A. Also, an illustrative example is presented in Appendix A to show how the D-NPI algorithm builds classification trees.

This thesis extends the use of the D-NPI algorithm for multinomial data across multiple classification problems: bagging and random forest in Chapter 2, imprecise classification in Chapter 3, and multi-label classification in Chapter 4.

1.6 Thesis outline

This thesis is organised as follows: Chapter 2 applies the D-NPI algorithm for both binary data and multinomial data to ensemble methods, specifically bagging and random forest, to assess how it performs when used in ensemble methods. The performance of the algorithms is also investigated and compared to other methods from the literature.

Chapter 3 extends the D-NPI classification method for multinomial data to imprecise classification, by developing new algorithms based on the D-NPI algorithm. The performance of the algorithms is evaluated, and compared to other classification algorithms from the literature. Some results of this chapter have been presented at a Statistics seminar at the University of Durham, United Kingdom, 26 June 2025.

In Chapter 4, the D-NPI algorithm is extended to multi-label classification problems, investigating its performance. An experimental study is conducted to evaluate the performance of the algorithm, comparing it to other algorithms from the literature.

Chapter 5 introduces a novel application of performance measures to evaluate impre-

cise probability inferences, including the NPI method for bivariate data. The focus of this chapter is to study the performance of the semi-parametric predictive method introduced by Coolen-Maturi et al. [29], using performance measures. A simulation study is conducted to evaluate the performance of the method under different scenarios, and a comparative analysis is carried out using the same measures to compare it with another method. Some results of Chapter 5 have been presented at the Research Students' Conference in Probability and Statistics at the University of Nottingham, United Kingdom, 05-08 September 2022. Also, part of this chapter has been presented at the 13th International Symposium on Imprecise Probability: Theories and Applications (ISIPTA'23) at Oviedo, Spain, 11-14 July 2023.

Finally, Chapter 6 provides the key findings and conclusions of the thesis. It also suggests interesting directions for future research.

The analysis and computations in this thesis have been implemented using the statistical software R [95]. Some of the computations have been carried out on different computers due to computation time. In order to implement the proposed methods presented in Chapters 2, 3, and 4, the D-NPI classification algorithm is implemented entirely in R without relying on existing packages. Implementing the D-NPI algorithm from scratch allows full control over the splitting and stopping criteria, ensuring that the extensions proposed in this thesis are straightforward to incorporate and adapt.

Chapter 2

Bagging and random forest with NPI

2.1 Introduction

Classification trees, while effective for a variety of predictive tasks, are inherently susceptive to high variance; meaning their predictive performance can be sensitive to small changes in the dataset used for training [60]. Their sensitivity arises from their tendency to capture details and closely align with the unique characteristics of the training dataset. For example, when a dataset has random noise or infrequent patterns, the classification tree may incorporate these into its structure. Consequently, small variations in the training dataset, such as slightly changing a few instances, can result in different classification trees.

The tradeoff between bias (the error caused by overly simplifying the model) and variance (the sensitivity of the model to small changes in the dataset) is a fundamental concept that affects the model's accuracy [68]. To address the limitations of classification trees, ensemble methods have proven to be effective tools by combining multiple classification trees to produce more accurate and robust models. These models have been shown to perform well on both noisy and noise-free datasets [48]. Among the most widely adopted ensemble methods are bagging (bootstrap aggregating) and random forest [21, 22].

In 1996, Breiman [21] introduced bagging as a technique to reduce variance by creating multiple bootstrap samples from the training dataset (i.e., random samples with replacement from the training dataset), training a separate model on each sample, and aggregating the predictions. The diversity of bootstrap samples introduces variation in individual classification trees, thereby reducing correlation between the predictions of

the individual trees and enhancing the ensemble's performance by reducing variance. Majority vote is the aggregation method used in classification ensemble methods.

Adding a further layer of randomness in attribute selection may offer improvements over the bagging method. Breiman [22] introduced random forest method as an extension of the bagging method, aiming to enhance predictive performance by reducing correlation among the predictions of the individual classification trees. This method adds an extra layer of randomness by selecting a random subset of attribute variables at each split within the classification trees. Selecting a subset of attributes at each split, instead of choosing a fixed subset prior to building the classification tree, increases the diversity of the individual classification trees in the ensemble, and helps to reduce correlation between the predictions. The random forest method helps in avoiding overfitting by ensuring that different attribute variables are considered during the tree construction process, enhancing the diversity of the individual classification trees in the ensemble, resulting in the method's ability to generalise.

Fernández-Delgado et al. [55] showed that, among the 179 classifiers evaluated across 121 datasets, the random forest versions ranked among the best-performing classifiers. The authors concluded that three out of the best five methods were random forest. The best classifier was parallel random forest (parRF_t), implemented using the '`randomForest`' package in R with '`caret`' for interface, tuning the parameter mtry (controlling the number of selected attributes at each split) to be 2:2:8 (from 2 to 8 in steps of 2). The other best results were achieved by classifiers rf_t and rforest_R, where the rf_t tuned the parameter mtry = 2:3:29 (from 2 to 29 in steps of 3), and the rforest_R was the random forest by Breiman [22] without tuning the parameter. The No Free Lunch (NFL) theorem for supervised classification suggests that no single classifier outperforms others across all datasets [111]. However, Fernández-Delgado et al. [55] showed that the performance of the random forest method was close to the best performance for almost all datasets, with the exception of three datasets out of a total of 121 datasets.

In this chapter, the application of the D-NPI method [12] to ensemble methods is introduced. New ensemble methods for bagging and random forest are proposed, which are called Nonparametric Predictive Inference for Bagging (NPI-Bag) and Nonparametric Predictive Inference for Random Forest (NPI-RF). These methods are introduced for both binary data and multinomial data using the D-NPI algorithm (introduced in Section 1.5)

as the base classifier. An experiment is conducted using different datasets, and the accuracy measure is used to assess the performance, comparing the results of models to the D-NPI algorithm.

This chapter is structured as follows: Section 2.2 provides an overview of bagging and random forest methods. Section 2.3 introduces the new methods for bagging and random forest. Section 2.4 details the experimental study, comprising a comparative study which compares the proposed ensemble methods with the base classifier (D-NPI). Section 2.4.2 presents a performance evaluation study to compare the performance of the proposed ensemble method against ensemble methods from the literature. Finally, Section 2.5 provides concluding remarks.

2.2 Bagging and random forest

In decision-making processes, it is crucial to combine multiple sources of information (or decisions) to improve the final outcome. In classification, ensemble approaches enhance the performance of classifiers by aggregating multiple trees [48]. Rather than relying on a single classification tree, ensemble methods combine a set of trees using the majority voting criterion, which is simple and intuitive.

Bagging was developed by Breiman [21] to generate multiple models using bootstrapped samples and to determine the final prediction through majority voting. Bootstrapped samples are datasets created by sampling with replacement from the original dataset, maintaining equal size, and thereby introducing variability into the training datasets [21]. Random forest was later introduced by Breiman [22] as an extension of bagging by adding attribute randomness, which reduces the correlation between the predictions of the individual trees. This ensemble method improves classifier accuracy and enhances robustness by randomly selecting a subset of attributes at each node. The bagging and random forest methods are illustrated in Figures 2.1 and 2.2, respectively.

Dietterich [47] presented an experimental comparison of three popular ensemble learning methods (randomizing, bagging, and boosting), aiming to compare their performance with the C4.5 algorithm [94]. His study showed that the results achieved by the boosting C4.5 method were the best in cases where there was no or little noise in the data, and the bagging C4.5 method achieved best performance when adding noise to data by taking

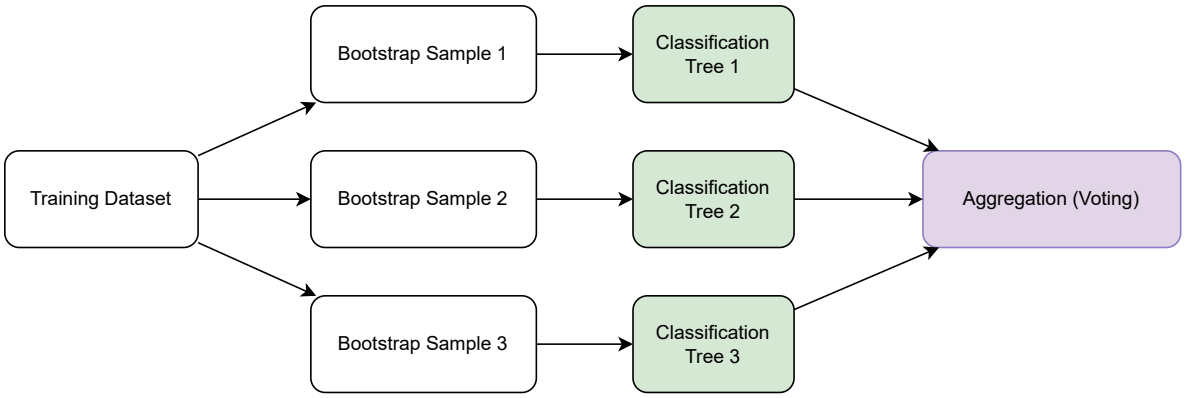


Figure 2.1: Bagging ensemble method.

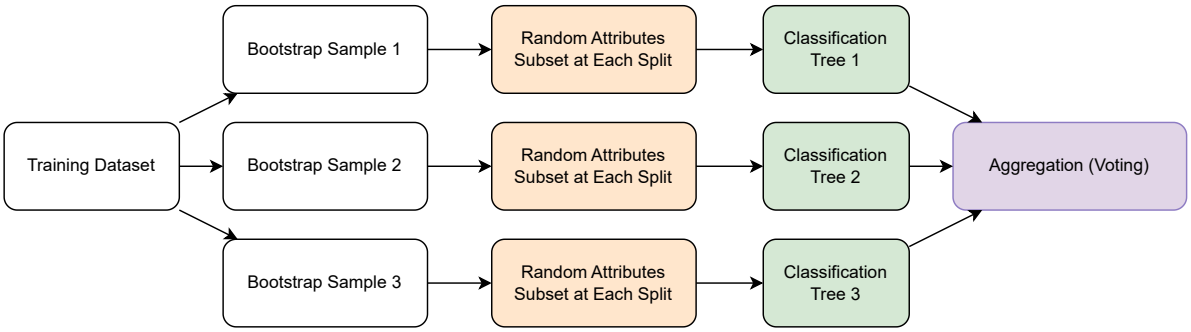


Figure 2.2: Random forest ensemble method.

advantage of the noise added to data to produce more diverse trees.

In the literature, bagging and random forest methods have been introduced using models based on imprecise probabilities. Abellán [2] introduced bagging using Nonparametric Predictive Inference Model for Multinomial data (NPI-M) to address classification problems with noise, and compared its performance to other classifiers, including bagging using the J48 classifier (the J48 classifier is an implementation of the C4.5 algorithm in Weka software), bagging with Credal decision trees (CDTs) using the Imprecise Dirichlet Model (IDM), and random forest classifier from the literature. The results showed that the bagging scheme with CDTs using the NPI-M outperformed other ensemble methods when the data contained a high level of noise.

Abellán et al. [8] modified the random forest base classifier to use a measure to quantify uncertainty based on imprecise probabilities instead of precise probability, which was called Credal Random Tree (CRT). It used Imprecise Info-Gain (IIG) measure [5]. Then, they introduced a random forest procedure using the CRT classifier as a base classifier. This method was called Credal Random Forest (CRF). An experimental study was carried

out to evaluate the performance of the CRF classifier in comparison with other classifiers when adding noise to datasets. The other classifiers were bagging using the C4.5 algorithm (BA-C4.5), bagging with CDTs (BA-CDT), random forest (RF), and CRF. The results showed that the CRF classifier outperformed other procedures and remained robust when noise was added to class variable.

The ability of bagging and random forest methods to achieve a tradeoff between bias and variance can give advantages over the base classifier. The D-NPI algorithm performed well in comparison with other algorithms from the literature [12], making it interesting to apply it to bagging and random forest methods. The next section presents the proposed algorithms designed for bagging and random forest.

2.3 NPI for bagging and random forest

This section presents the proposed algorithms for bagging and random forest using the Direct-NPI algorithm introduced by Alharbi [12] as the base classifier. The D-NPI algorithm employs the Correct Indication (CI) splitting criterion to build classification trees based on the NPI approach, as described in Section 1.5. This algorithm was originally developed for both binary and multinomial data. The proposed bagging and random forest algorithms are designed to improve the performance of the D-NPI classifier and to investigate its behaviour when used in ensemble methods. The bagging method is described first. Algorithm 2.1, Nonparametric Predictive Inference for Bagging (NPI-Bag), incorporates the D-NPI algorithm (presented in Appendix A) as the base classifier and adopts the CI splitting criterion. Algorithm 2.2, Nonparametric Predictive Inference for Random Forest (NPI-RF), also uses the D-NPI algorithm (presented in Appendix A) as the base classifier and applies the CI criterion for splitting, with random forests enabling additional attribute randomness to enhance the accuracy of the ensemble.

This study has two main goals. The first goal is to evaluate the performance of the ensemble methods NPI-Bag and NPI-RF in comparison to the base algorithm, D-NPI. This comparison aims to determine whether the ensemble methods improve the performance of the base classifier by combining multiple classifiers to reduce errors and enhance accuracy. The second goal is to compare the performance of the proposed ensemble methods with ensemble methods from the literature. Section 2.4 presents the experimental studies

Algorithm 2.1 Pseudocode of the NPI-Bag algorithm.

Input: Dataset ($\mathcal{D}$), Attributes (Att), Class Variable (C), Number of trees (N_{trees})
Procedure NPI-Bag ($\mathcal{D}, Att, C, N_{\text{trees}}$)
  for $t = 1$ to N_{trees}
    Generate bootstrap sample $\mathcal{D}_t$ from dataset $\mathcal{D}$
    Call D-NPI using $\mathcal{D}_t$ as the training dataset, and build Classification Tree $\mathcal{CT}_t$
  Return Bagging model
  for each new instance $\mathbf{x}$
    Aggregate predictions for instance $\mathbf{x}$ across all trees $\mathcal{CT}_1, \mathcal{CT}_2, \ldots, \mathcal{CT}_T$
    Determine final prediction by majority voting
Return The predicted class label of $\mathbf{x}$

Algorithm 2.2 Pseudocode of the NPI-RF algorithm.

Input: Dataset ($\mathcal{D}$), Attributes (Att), Class Variable (C), Number of trees (N_{trees}, Number of selected features (m))
Procedure NPI-RF ($\mathcal{D}, Att, C, N_{\text{trees}}$, m)
  for $t = 1$ to N_{trees}
    Generate bootstrap sample $\mathcal{D}_t$ from dataset $\mathcal{D}$
    Call D-NPI using $\mathcal{D}_t$ as the training dataset with m randomly selected attributes at each split, and build Classification Tree $\mathcal{CT}_t$
  Return Random Forest model
  for each new instance $\mathbf{x}$
    Aggregate predictions for instance $\mathbf{x}$ across all trees $\mathcal{CT}_1, \mathcal{CT}_2, \ldots, \mathcal{CT}_T$
    Determine final prediction by majority voting
Return The predicted class label of $\mathbf{x}$

conducted to evaluate the performance of the proposed ensemble methods.

2.4 Experimental setup

The experimental analysis is presented in two main sections. Section 2.4.1 provides a detailed analysis of the performance of the proposed ensemble methods, NPI-Bag and NPI-RF, compared to the base algorithm, D-NPI. This includes experiments conducted on several datasets. Section 2.4.2 presents a comparative study, in which the proposed methods are evaluated in comparison with other ensemble methods from the literature.

2.4.1 Performance comparison of proposed methods and base classifier

In this experiment, 20 datasets from the UCI *Machine Learning Repository* [75] are used. These datasets vary in size, number of attributes, number of class labels, and range of values for nominal attributes. They cover a variety of classification problems, including both binary and multinomial cases. Table 2.1 provides an overview of the datasets where the column "n" represents the size of the dataset, "Att" indicates the number of attributes, "Num" represents the number of numerical attributes, "Nom" displays the number of nominal attributes, "R" denotes the number of class labels, and "Range" specifies the range of values of nominal attributes.

The datasets are prepared for classification by replacing missing values with the mode for nominal attributes or the mean for continuous attributes. Discretisation is applied as a standard preprocessing step to prepare continuous attributes for classification. Note that discretisation is applied in this chapter only as a preprocessing step to transform continuous attributes into categorical attributes, thereby making the dataset compatible with the classification algorithms which require categorical attributes. The aim of this chapter is not to evaluate the discretisation methods, but rather to evaluate and compare the performance of the classification algorithms. For the Banknote Authentication and Breast Cancer Wisconsin dataset, the continuous attributes are discretised to binary attributes as in Alharbi [12], based on the optimal threshold determined using the Information Gain Ratio splitting criterion [94]. The remaining datasets, which contain continuous attributes are processed using the '`discretize`' function from the package '`FSelectorRcpp`' in R [114]. Continuous attributes are discretised using Fayyad and Irani's method [54]; if no valid cut points are found, the equal-width binning method is

Dataset	n	Att	Num	Nom	R	Range
Balance-Scale	625	4	4	-	3	-
Banknote Authentication	1372	4	4	-	2	-
Breast Cancer Wisconsin	699	9	9	-	2	-
Congressional Voting Records	435	16	-	16	2	2
CMC	1473	9	2	7	3	2-4
Dermatology	366	34	1	33	6	2-4
Glass	214	9	9	-	7	-
Hayes	160	4	-	4	3	3-4
Hypothyroid	3772	27	6	21	4	2-5
Iris	150	4	4	-	3	-
Mushroom	8124	22	-	22	2	2-12
Nursery	12960	8	-	8	5	2-5
Post-Operative Patient	90	8	-	8	3	2-4
Primary-Tumor	339	17	-	17	21	2-3
Qualitative-Bankruptcy	250	6	-	6	2	3
Soybean-Large	683	35	-	35	19	2-7
Sponge	76	44	-	44	3	2-9
Vehicle	846	18	18	-	4	-
Yeast	1484	8	8	-	10	-
Zoo	101	17	1	16	7	2

Table 2.1: Datasets [75]. Size "n", number of attributes "Att", number of numerical attributes "Num", number of nominal attributes "Nom", number of class labels "R", and range of values of nominal attributes "Range".

applied using the '`discretize`' function from the same package [114].

First, the D-NPI algorithm introduced by Alharbi [12] is used in this section to evaluate how the proposed ensemble methods improve performance compared to the base algorithm. The bagging and random forest methods, NPI-Bag and NPI-RF, along with D-NPI, are executed in R software. The proposed ensemble methods are implemented following the procedures in Algorithms 2.1 and 2.2.

The number of trees used in this study is 100, following the approach adopted in previous studies, including Abellán [2]. For the NPI-RF algorithm, the number of attributes selected at each split is approximately $\sqrt{\text{Att}}$, as suggested by Breiman [22]. To assess performance, the accuracy metric is applied, defined as the ratio of correct predictions to the total number of predictions. Performance is evaluated using K-fold cross-validation, where the dataset is divided into K folds. The model is trained on $K-1$ folds and tested on the remaining fold, repeating the process K times so that each fold serves once as the testing set [78].

Dataset	NPI-Bag	NPI-RF	D-NPI
Balance-Scale	**70.73**	70.26	68.17
Banknote Authentication	**89.50**	**89.50**	**89.50**
Breast Cancer Wisconsin	95.14	**95.71**	95.29
Congressional Voting Records	**95.61**	94.92	**95.61**
CMC	53.02	**53.70**	52.48
Dermatology	95.38	**95.64**	94.54
Glass	77.08	**78.40**	73.38
Hayes	76.25	**80.63**	71.25
Hypothyroid	99.20	92.52	**99.28**
Iris	**95.33**	94.67	**95.33**
Mushroom	**100.00**	**100.00**	99.96
Nursery	**91.98**	91.19	91.94
Post-Operative Patient	66.67	**71.11**	66.67
Primary-Tumor	39.80	**42.15**	40.09
Qualitative-Bankruptcy	99.60	**100.00**	99.60
Soybean-Large	85.35	**92.67**	85.36
Sponge	**92.32**	**92.32**	**92.32**
Vehicle	**73.15**	71.87	72.44
Yeast	**60.31**	60.24	59.30
Zoo	92.09	**94.18**	92.09

Table 2.2: Accuracy results: comparison with base classifier.

In the NPI-Bag and NPI-RF ensemble methods, for each iteration, one fold is hold out as the testing set, while the remaining $K-1$ folds are used to generate multiple bootstrap samples of the same size for training. The ensemble methods are then trained on these bootstrap samples. In this experiment, a 10-fold cross-validation procedure is applied across all datasets. The performance of all classifiers is assessed using the accuracy metric, and the final result is obtained by averaging the accuracy across all folds.

Table 2.2 presents the results for each algorithm, showing the accuracy for each dataset. The results demonstrate the effectiveness of the ensemble methods, NPI-Bag and NPI-RF, compared to the D-NPI algorithm.

The algorithms perform equally well for the datasets Banknote Authentication and Sponge. Having identical performance for all algorithms for these datasets can be due to the similarity of individual trees in the ensemble methods, leading to the class label being predicted as majority class same as in the D-NPI algorithm. For example, for the Sponge dataset, the class variable distribution is highly imbalanced, with three class labels containing 70, 3, and 3 instances. The minority class labels have very low frequencies.

As a result, the D-NPI algorithm tends to either stop very early or fails to split. In the ensemble methods, NPI-Bag and NPI-RF, bootstrap samples are likely to include the majority class repeatedly, while underrepresenting or even excluding the minority class labels. This results in many identical individual classification trees, even when attributes are sampled randomly in NPI-RF, leading to identical performance values across algorithms.

For binary datasets, such as Banknote Authentication, Breast Cancer Wisconsin and Congressional Voting Records, all algorithms tend to have similar performance. For the Congressional Voting Records dataset, the D-NPI and NPI-Bag algorithms show slightly higher accuracy values than the NPI-RF algorithm. This may be due to sampling subsets from a high number of attributes compared to the other datasets, which increases the possibility of missing informative attributes when building individual classification trees, unlike when using the full set of attributes in D-NPI and NPI-Bag.

The NPI-RF ensemble method tends to perform better for some datasets with a large number of attributes, as this increases the diversity in the ensemble, reducing correlation among predictions of individual trees. For example, for the datasets Dermatology, Primary-Tumor, Soybean-Large, and Zoo, the NPI-RF method benefits from the large number of attributes.

The Hypothyroid dataset has 27 attributes, the NPI-RF ensemble method has lower performance compared to the other algorithms. To explain this, all algorithms use a stopping criterion, but the NPI-RF algorithm produces shallower individual trees on average compared to D-NPI and NPI-Bag. This is primarily due to the attributes sampling in the NPI-RF algorithm, which limits the set of possible attributes at each split and leads to weaker splits. In contrast, the NPI-Bag and D-NPI algorithms use the full set of attributes, allowing the trees to grow deeper due to the availability of more informative splitting options. Further analysis of the NPI-RF method without the stopping criterion is presented later in Section 2.4.2.

For the datasets CMC, Mushroom, Nursery, Qualitative-Bankruptcy, Vehicle, and Yeast, the classification accuracies of the three algorithms show only minimal differences. However, the best accuracy values are observed among the ensemble methods, NPI-Bag and NPI-RF. All these datasets, except for the Yeast dataset, have low number of class labels (typically between 2 and 4), which may lead to less complex learning, and often

do not require deep individual trees. The Yeast dataset contains 10 class labels, but it has a highly imbalanced class distribution. When using bootstrap samples, the resulting samples can often contain the majority class repeated multiple times. This leads to individual classification trees in the ensemble methods with almost similar performance to that produced by the D-NPI algorithm. This explains the similar performance observed across all the algorithms for this dataset.

For datasets with small sizes, such as Glass, Hayes, and Iris datasets, differences in accuracy values among algorithms are expected. This is because their testing sets contain a small number of instances, meaning that a single incorrect prediction can affect the overall accuracy. However, for the Glass and Hayes datasets, the ensemble methods tend to perform better than the D-NPI algorithm, observing the highest accuracy values for the NPI-RF method. In the case of the Hayes dataset, the large difference in accuracy values between the NPI-RF algorithm and the D-NPI algorithms can be due to the informative nature of all attributes. Sampling attributes at random at each split allows different attributes to be used, which allows capturing different patterns. The D-NPI algorithm selects a single best attribute among the four attributes at the root node. However, due to the randomisation in the NPI-RF method, it allows all attributes to be selected across different individual trees. As a result, the variation among the individual trees can be beneficial, allowing better performance for the NPI-RF method compared to the single classification tree produced by the D-NPI algorithm.

Table 2.2 shows that among the three methods, the NPI-RF algorithm achieves the highest accuracy values for most of the datasets, followed by the NPI-Bag algorithm. The D-NPI algorithm has similar performance to either NPI-RF or NPI-Bag for a few number of datasets. These results show the strong performance of both NPI-Bag and NPI-RF compared to D-NPI. This experiment shows that the ensemble methods are capable of enhancing the performance of the D-NPI algorithm by applying it to ensemble methods.

Section 2.4.2 presents the performance of the new ensemble methods compared to two methods from the literature.

Dataset	NPI-Bag	NPI-RF	NPI-M-B	RF
Balance-Scale	**70.73**	70.26	69.29	69.78
Banknote Authentication	**89.50**	**89.50**	**89.50**	**89.50**
Breast Cancer Wisconsin	95.14	95.71	95.43	**96.43**
Congressional Voting Records	95.61	94.92	96.29	**96.53**
CMC	53.02	53.70	**54.38**	52.95
Dermatology	95.38	95.64	93.71	**97.84**
Glass	77.08	78.40	73.74	**79.42**
Hayes	76.25	80.63	73.75	**81.88**
Hypothyroid	99.20	92.52	98.99	**99.63**
Iris	**95.33**	94.67	**95.33**	94.00
Mushroom	**100.00**	**100.00**	**100.00**	**100.00**
Nursery	91.98	91.19	95.97	**99.57**
Post-Operative Patient	66.67	**71.11**	**71.11**	65.56
Primary-Tumor	39.80	42.15	**44.21**	41.56
Qualitative-Bankruptcy	99.60	**100.00**	99.60	99.60
Soybean-Large	85.35	92.67	92.09	**94.72**
Sponge	**92.32**	**92.32**	**92.32**	**92.32**
Vehicle	73.15	71.87	62.88	**74.58**
Yeast	**60.31**	60.24	58.68	60.24
Zoo	92.09	94.18	94.09	**96.00**

Table 2.3: Accuracy results: comparison with other ensemble methods.

2.4.2 Performance comparison of proposed methods and existing approaches

In this section, a comparative analysis between the proposed ensemble methods and two existing methods from the literature is presented. The first ensemble method considered is bagging, which uses a base classifier proposed by Abellán et al. [7], and is based on Nonparametric Predictive Inference Model for Multinomial data (NPI-M). It uses maximum entropy measure within the imprecise information gain splitting criterion to select the best attribute to split a node. This method is referred to as NPI-M-B throughout this chapter. Random forest, adopted as the second method for comparison, is based on the random forest algorithm proposed by Breiman [22]. This method is referred to as RF throughout this chapter.

The NPI-M-B method is implemented using the '`imptree`' package in R [56], and the RF method with the '`randomForest`' package in R [23], both with their default settings (with fully grown trees for RF). The same experimental procedure as in Section 2.4.1 is followed. Table 2.3 reports the accuracy for each dataset and the overall average.

Starting with the binary datasets, such as Banknote Authentication, when the number of attributes is low, all classifiers achieve the same performance. For the Breast Cancer Wisconsin, the RF ensemble method has the highest accuracy compared to the other algorithms. As mentioned in Section 2.4.1, the NPI-RF ensemble method uses a stopping criterion, while RF fully grows the individual trees, which may cause the observed decrease in the performance for the NPI-RF classifier. The NPI-RF ensemble method is implemented with fully grown individual trees later in this section to enable the comparison without using the stopping criterion for the individual trees. Regarding the NPI-Bag and NPI-M-B methods, they tend to perform similarly for the Breast Cancer Wisconsin dataset with only a slight difference.

For the Mushroom dataset, all classifiers achieve an accuracy of 100%, which can be attributed to the predictive nature of the attributes in this dataset, making all classifiers perform similarly well. All algorithms have identical performance for the Sponge dataset, for the reasons discussed in Section 2.4.1.

For further investigation, the RF algorithm achieves an accuracy of 99.51% for the Nursery dataset, subsequently outperforming the NPI-RF algorithm, which attains 91.33%. This performance gap is likely due to RF producing fully grown individual trees, which helps the learning of complex patterns in the dataset, particularly for this dataset with a large number of instances. Moreover, for the Hypothyroid dataset, when the bagging methods NPI-Bag and NPI-M-B are used, the observed accuracy values are high, as in the case when using the RF algorithm. However, for the NPI-RF algorithm the performance is comparatively lower than that of all classifiers. While the NPI-RF algorithm uses a stopping criterion, which may limit the individual classifiers' ability to learn more complex patterns compared to the RF algorithm. Tree Depth (TD) measure is additionally used. It refers to the maximum number of edges from the root node to any leaf node. When investigating further, the observed TD average value of the individual trees produced by the NPI-Bag method is higher than this observed for the NPI-RF method. This explains the low performance of the NPI-RF method, which can be improved by using fully grown individual trees.

The NPI-Bag and NPI-RF classifiers tends to perform well when the datasets contain relatively low Range (i.e., the number of values in each attribute is ≤ 4). These datasets include CMC, Glass, Hayes, Iris, Post-Operative Patient, Qualitative-Bankruptcy, and

Yeast. For the CMC, Iris, and Yeast datasets, the discretised attributes have a number of values ≤ 4. The NPI-RF classifier allows multiway splits, while the RF classifier allows binary splits. As a result, the RF classifier can produce deeper individual trees compared to the NPI-RF classifier. Adapting the D-NPI algorithm to work using binary splits presents an interesting direction for future research.

Across multiple datasets, the two algorithms NPI-Bag and NPI-M-B, show similar performance. The NPI-Bag algorithm achieves competitive results for certain datasets such as Dermatology, Glass, and Vehicle, and NPI-M-B achieves high values compared to the NPI-Bag classifier, particularly for the datasets Nursery, Post-Operative Patient, Primary-Tumor, and Soybean-Large. The reason for this observation remains unclear, indicating that further investigation may be required in the future.

The primary difference between the two random forest-based algorithms lies in their tree growth strategies: the RF algorithm allows fully grown individual trees and uses an ensemble of these trees to reduce overfitting through aggregating, which often helps capture complex patterns better. In contrast, the NPI-RF algorithm uses a stopping criterion, which may result in shallower individual trees that can affect their ability to capture complex structures, potentially limiting performance for more complex datasets. To investigate this further, the NPI-RF ensemble method is implemented without the stopping criterion, producing fully grown individual trees, and the results are presented in Table 2.4. In Table 2.4, the NPI-RF algorithm, when implemented with fully grown trees, is referred to as NPI-RF-FG. The results of both algorithms, NPI-RF and NPI-RF-FG, are compared to the RF algorithm.

In Table 2.4, it is observed that when constructing fully grown trees using the NPI-RF-FG classifier, the accuracy is higher than the that of the NPI-RF algorithm across most datasets. For the Hypothyroid datasets, as mentioned earlier in this section, when using the stopping criterion, as applied in the NPI-RF classifier, resulted in early termination of the trees construction, which produces shallow individual trees in the ensemble. When allowing fully grown trees using the NPI-RF-FG classifier, the performance is higher than that of the NPI-RF classifier.

For the Nursery dataset, the NPI-RF-FG classifier achieves better performance compared to the NPI-RF classifier, which is due to allowing deeper and more diverse individual trees in the ensemble. However, the accuracy of the NPI-RF-FG method is

Dataset	NPI-RF	NPI-RF-FG	RF
Balance-Scale	**70.26**	70.09	69.78
Banknote Authentication	**89.50**	**89.50**	**89.50**
Breast Cancer Wisconsin	95.71	**96.57**	96.43
Congressional Voting Records	94.92	95.84	**96.53**
CMC	**53.70**	52.34	52.95
Dermatology	95.64	96.73	**97.84**
Glass	78.40	**80.80**	79.42
Hayes	80.63	80.00	**81.88**
Hypothyroid	92.52	98.73	**99.63**
Iris	94.67	**95.33**	94.00
Mushroom	**100.00**	**100.00**	**100.00**
Nursery	91.19	96.44	**99.57**
Post-Operative Patient	**71.11**	65.56	65.56
Primary-Tumor	42.15	**44.51**	41.56
Qualitative-Bankruptcy	**100.00**	99.60	99.60
Soybean-Large	92.67	93.40	**94.72**
Sponge	**92.32**	**92.32**	**92.32**
Vehicle	71.87	**74.70**	74.58
Yeast	60.24	**60.31**	60.24
Zoo	94.18	95.00	**96.00**

Table 2.4: Accuracy results: comparison with base classifier (fully grown).

lower than that of the RF classifier, possibly because RF uses binary splits (based on the CART algorithm), which can produce deeper individual trees than those generated by the NPI-RF-FG classifier. This dataset contains a large number of instances, which may require deeper individual trees to be capable of capturing the underlying complexity of this dataset. This finding suggests that using ensemble methods with fully grown trees can be more effective when dealing with large datasets, as they enable better capturing of the complexity of the dataset.

Interestingly, applying the stopping criterion in the NPI-RF method led to slightly better performance compared to allowing the individual trees to grow fully, as in the NPI-RF-FG method, particularly for the datasets such as Balance-Scale, CMC, Dermatology, Post-Operative Patient, and Qualitative-Bankruptcy. For datasets like Post-Operative Patient, which has limited size and imbalance class labels, fully grown trees tend to make highly specific splits to fit rare patterns in the bootstrap samples. Using stopping criterion, as in the NPI-RF method, can lead to better performance.

In this analysis, the NPI-RF-FG method achieves a competitive performance compared to the RF method across a diverse range of datasets. It is also highlighted that

for some datasets, it is beneficial to allow stopping early, which can result in improved performance and computational efficiency.

To delve deeper into the performance of the introduced ensemble methods, an illustrative example showing the depth of the individual trees in the ensembles is provided in Appendix B. The TD measure is used in the example.

The results of this example show that the ensemble methods, NPI-Bag and NPI-RF, allow variation in TD among individual trees. For the NPI-RF classifier, the TD values of the individual trees tend to vary more than those of the NPI-Bag classifier, with many trees being deeper. This helps to enhance the performance of the NPI-RF method compared to the D-NPI and NPI-Bag algorithms in the example. When using bagging and random forest methods, deeper trees can be produced, which might overfit the bootstrap samples used for their training. However, using aggregation for the final prediction aims to reduce overfitting caused by the individual classifiers within the ensemble. Regarding the NPI-Bag method, the individual trees have limited complexity, observed in their TD values compared to the classification tree produced by the D-NPI classifier. The NPI-RF method injects more randomness by selecting a subset of attributes at each split [22], resulting diverse TD values of the individual trees compared to the NPI-Bag method, thereby enhancing the final predictions by aggregating the predictions of the individual classifiers.

2.5 Conclusions

In this chapter, two ensemble methods have been introduced for bagging and random forest, referred to as NPI-Bag and NPI-RF, respectively, each using D-NPI as the base classifier. These proposed ensemble methods were designed to use the strengths of the D-NPI algorithm while enhancing predictive accuracy through ensemble learning techniques. This study evaluated the performance of the proposed ensemble methods by comparing them to the D-NPI algorithm and other ensemble methods. The primary objective was to improve the performance of the D-NPI algorithm when applied to ensemble methods. The second goal was to investigate how the proposed ensemble methods perform in comparison with other ensemble methods.

Experimental studies were conducted to evaluate the performance of the proposed

methods. Two main scenarios were included in this study. The first focused on comparing the proposed ensemble methods to the base classifier, including experiments on multiple datasets. The second scenario involved comparing the proposed methods against two other ensemble methods, namely NPI-M-B for bagging and RF for random forest. This included an additional investigation in which the NPI-RF method was applied without a stopping criterion, allowing fully grown individual trees in the ensemble. This method, referred to as NPI-RF-FG method, was compared to the NPI-RF and RF methods.

In the first scenario, the proposed ensemble methods, NPI-Bag and NPI-RF, performed well compared to the D-NPI algorithm, with the NPI-RF algorithm showing best performance for most of the datasets.

In the example, the performance of the classification algorithms D-NPI, NPI-Bag, and NPI-RF was evaluated in terms of accuracy and tree depth across the Soybean-Large dataset (see Appendix B). Additionally, the distributions of the individual trees depths in the NPI-Bag and NPI-RF ensemble methods were analysed to assess model complexity. The results showed diversity in the tree depth values in the NPI-RF method, and how this diversity affected its overall high performance. The NPI-Bag method achieved the same accuracy as the D-NPI algorithm for this dataset, with the NPI-Bag method producing shallow and less diverse individual trees compared to the NPI-RF method. The reduced complexity of the NPI-Bag method compared to the NPI-RF method could affect its performance, making it lower than that of the NPI-RF method.

In the second scenario, the RF method was the only method to produce fully grown individual datasets, achieving the highest performance compared to the other methods. While the NPI-RF algorithm struggled to achieve the highest accuracy for several datasets, particularly for the large-sized datasets, due to its stopping criterion, limiting its ability to capture complex patterns. Meanwhile, the bagging methods, NPI-Bag and NPI-M-B, achieved similar results across most datasets, except for some datasets where performance varied without a clear reason, highlighting a potential area for future investigation.

For further study, the performance of the NPI-RF method was analysed without the stopping criterion to allow producing fully grown individual trees. This method was referred to as the NPI-RF-FG method. The results were compared to those of NPI-RF and RF. Producing individual trees using the NPI-RF-FG method resulted in higher per-

formance compared to the NPI-RF method across most of the datasets, especially when working with larger datasets such as the datasets Hypothyroid and Nursery. The NPI-RF-FG method had competitive performance compared to the RF method. While the NPI-RF-FG method shows promising results, for more complex datasets, future research could explore using binary splits rather than multiway splits, allowing the capture of more complex patterns. Based on these results, it is beneficial to allow deeper or fully grown individual trees in ensemble methods in order to capture complex patterns.

This study concluded the good performance of the ensemble methods, NPI-Bag and NPI-RF, using the D-NPI algorithm as a base classifier. A topic for future investigation is to extend this work to weighted cost sensitive classification [104], with the aim of addressing the challenge of imbalanced class distributions. Studying the robustness of the methods in classification noise problems is worth investigating in future work.

Chapter 3

Imprecise classification with NPI

3.1 Introduction

In classification problems, a single class label is assigned to each instance in traditional methods, which can potentially lead to misleading predictions due to the tendency of datasets to contain noise and limited information in real-life applications. Assigning a set of possible class labels to each instance offers a more cautious approach, providing an edge over the traditional methods. The classifiers used to predict a set of class labels are known as "*imprecise classifiers*" [4] or "*credal classifiers*" [113]. These classifiers are functions that map instances to a set of predicted class labels.

In traditional classification methods, a new instance is assigned the most frequent class label of the terminal node it reaches. In imprecise classification, by contrast, classifiers assign probability intervals using a model based on imprecise probabilities. A dominance criterion is then applied to obtain the set of predicted class labels, known as the set of non-dominated states. The term *set of non-dominated states* will be denoted as *set of non-dominated labels* in this chapter, to align with the terminology in this thesis.

In the literature, several methods use imprecise probabilities to build classification trees. These methods use Imprecise Dirichlet Model (IDM) [107] to derive credal sets or Nonparametric Predictive Inference Model for Multinomial data (NPI-M) [33] to obtain possible non-convex sets of probabilities. In 2002, Zaffalon [113] introduced the first adaptation of the Naive Bayes Classifier (NBC) to imprecise classification, using credal sets and the IDM, thereby establishing the Naive Credal Classifier (NCC). The NCC classifier was proposed as a generalisation of standard classification using a credal dominance cri-

terion to produce credal classification. The study developed procedures for classification, and for computing the corresponding lower and upper posterior probabilities. Corani and Zaffalon [38] extended the NCC classifier for treating of incomplete data, introducing a classifier which is named NCC2. In an empirical study, they compared the performance of the NBC and NCC2 classifiers. The results indicated that NBC produced overconfident and unreliable predictions on hard to classify instances, whereas the NCC2 classifier resulted in set-valued (cautious but informative) classifications when uncertainty was high. They showed that both classifiers performed similarly on the other instances (not hard instances).

Abellán and Masegosa [4] introduced an adaptation of the Credal Decision Tree (CDT) [5] to imprecise classification, using the IDM and upper entropy, which is called Imprecise Credal Decision Tree (ICDT). The ICDT classifier uses a dominance criterion to obtain a set of predicted class labels, known as the set of non-dominated states. They compared the ICDT classifier with the NCC classifier, revealing that the former is more precise and informative.

Moral et al. [84] presented an adaptation of the CDT to imprecise classification, based on Nonparametric Predictive Inference for Multinomial data (NPI-M) model [33], which is referred to as Imprecise Credal Decision Tree with NPI (ICDT-NPI). This classifier uses the NPI-M, unlike the ICDT classifier, which is based on the IDM. The main reason is that the IDM assumes prior knowledge about the data through the hyperparameter s, and its performance strongly depends on the choice of this parameter [81]. The study concluded that the ICDT-NPI classifier outperforms the ICDT classifier when the hyperparameter $s = 3$, while ICDT-NPI provides comparable results to ICDT when the hyperparameter $s = 1, 2$.

In this chapter, a new adaptation of the D-NPI algorithm (introduced in Section 1.5) to imprecise classification is proposed. Specifically, three new algorithms are introduced, which are based on the NPI approach and use imprecise probabilities without adding any additional assumptions. In order to assess the performance of the proposed algorithms, various evaluation metrics are applied. Additionally, the performance of the proposed algorithms is compared with four existing imprecise classifiers based on the NPI-M model and the IDM.

This chapter is organised as follows: Section 3.2 provides a brief overview of the

NPI-M and the dominance criterion. In Section 3.3, the new algorithms for imprecise classification are introduced. Section 3.4 presents the performance evaluation metrics for imprecise classification, along with the statistical tests for comparisons. Section 3.5 presents the experimental study, including comparisons, results, and discussions. Finally, Section 3.6 concludes the chapter and presents potential topics for future research.

3.2 NPI model for multinomial data

As introduced earlier in Section 1.3, the Nonparametric Predictive Inference Model for Multinomial data (NPI-M) has been developed by Coolen and Augustin [33, 34]. In this section, the notation used in their work is adapted to align with that used in this chapter.

Let C be a class variable with the values $c_1, c_2, \ldots, c_R$ and let n^{c_r} be the total number of instances in the category c_r, where $r = 1, 2, \ldots, R$. The total number of instances in $c_1, c_2, \ldots, c_R$ is n, hence $\sum_{r=1}^{R} n^{c_r} = n$. The NPI lower and upper probabilities for the event $C_{n+1} = c_r$, where C_{n+1} is the unknown class label for the future instance, are [34]

$$\underline{P}(C_{n+1} = c_r) = \max\left(0, \frac{n^{c_r} - 1}{n}\right), \tag{3.1}$$

$$\overline{P}(C_{n+1} = c_r) = \min\left(\frac{n^{c_r} + 1}{n}, 1\right). \tag{3.2}$$

Consequently, a set of probability intervals for individual singletons is defined as:

$$\mathcal{I} = \{[l_r, u_r], r \in \{1, 2, \ldots, R\}\}, \tag{3.3}$$

where $l_r = \max\left(0, \frac{n^{c_r}-1}{n}\right)$ and $u_r = \min\left(\frac{n^{c_r}+1}{n}, 1\right)$. This set of reachable probability intervals corresponds to the credal sets given below [43]

$$\mathcal{P}(\mathcal{I}) = \{p \in \mathcal{P}(C) \mid p(c_r) \in [l_r, u_r], \quad \forall r \in \{1, 2, \ldots, R\}\}, \tag{3.4}$$

where $\mathcal{P}(C)$ is the of set all probability distributions for the categorical class variable C.

For singleton events (e.g., observing a future outcome belonging to a specific category), the lower and upper probabilities associated with the NPI-M model are obtained to form a credal set, which is the set of all probability distributions consistent with these probability

intervals. However, not all the probability distributions in the credal set are compatible with the constraints of the NPI-M model due to the structure required by the assumed probability wheel [6]. In order to avoid the constraints associated with the NPI-M model, an approximation model is derived from the NPI-M model, this model is referred to as A-NPI-M model [6]. The A-NPI-M model considers the convex hull of the set of probabilities compatible with the NPI-M model. Abellán et al. [6] introduced two algorithms based on the NPI approach and the maximum entropy measure. The first algorithm is used to attain the maximum entropy probability distribution based on the NPI-M, and the second algorithm uses the A-NPI-M to obtain the maximum entropy probability distribution. These algorithms, NPI-M and A-NPI-M, can be used to construct classification trees with the maximum entropy probabilities in the imprecise information gain criterion to select the best attribute to split a node.

Dominance criterion is a concept in decision-making processes under uncertainty, used to get the most plausible class labels. A class label is considered non-dominated if no other class label is better, based on a dominance criterion. The set of predicted class labels is defined to be the set of non-dominated labels. A common dominance criterion used is "Strong dominance". Based on this criterion, a class label is said to dominate another if and only if its lower probability is greater than or equal to the upper probability for the other class label. Consider a class variable C with two possible values: c_1 and c_2 and let the probability intervals of c_1 and c_2 be represented by $[l_1, u_1]$ and $[l_2, u_2]$, respectively. In this case, c_1 is considered to have a strong dominance on c_2 if and only if $l_1 \geq u_2$. Another dominance criterion is the Credal dominance criterion [113]. The class label c_2 is said to be credal-dominanted by the class label c_1 if and only if $p(c_1) \geq p(c_2)$ for all probability distributions $p \in \mathcal{P}$. In this study, the strong dominance criterion is used to attain the set of non-dominated labels when building imprecise classification trees.

3.3 Direct-NPI for imprecise classification

This section introduces three new algorithms for imprecise classification based on the D-NPI algorithm as presented by Alharbi [12]. The D-NPI classification method is built on the NPI approach and applies the Correct Indication (CI) splitting criterion, which itself is derived from NPI, see Section 1.5 for more details. The main difference between

the vanilla D-NPI algorithm and the proposed algorithms lies in the criterion used when the tree reaches a leaf node during the classification of a new instance, denoted as $\mathbf{x}$. While the D-NPI algorithm assigns the most common class label to the leaf node, the proposed algorithms assign probability intervals to the class labels based on the NPI-M model, as discussed in Section 3.2. In the proposed algorithms, the strong dominance criterion is used to derive the non-dominated labels after assigning a probability interval for each class label. The main differences between them lie in the criterion used for selecting the best attribute to split the tree and in the stopping criterion. The selecting criterion and the stopping criterion in the D-NPI algorithm are recalled from Chapter 1 for their relevance in this chapter. In the process of constructing a classification tree, an attribute is selected to split the tree. In the D-NPI algorithm, the NPI lower and upper probabilities for the attribute variable x_i leads to CI which reports the strength of evidence for the class label of the future instance. After deriving the NPI lower and upper probabilities for each attribute x_i for CI, a stopping criterion is used determine when to stop building the tree. Let n be the total number of instances and $n_{\max}$ be the number of instances with the most frequent class label. Let $\underline{P}(NA)$ and $\overline{P}(NA)$ be the NPI lower and upper probabilities for CI if no attribute variable is used. Using NPI for multinomial data [34], the NPI lower and upper probabilities are

$$\underline{P}(NA) = \max\left(0, \frac{n_{\max} - 1}{n}\right), \tag{3.5}$$

and

$$\overline{P}(NA) = \min\left(\frac{n_{\max} + 1}{n}, 1\right). \tag{3.6}$$

Let $\underline{P}_i(CI)$ and $\overline{P}_i(CI)$ be the NPI lower and upper probabilities for the event that the attribute x_i leads to CI. During building the classification tree, the attribute variable x_i with the highest NPI lower and upper probabilities $\underline{P}_i^*(CI)$ and $\overline{P}_i^*(CI)$ compared to the other attributes is selected if

$$\underline{P}_i^*(CI) > \underline{P}(NA) \quad \text{and} \quad \overline{P}_i^*(CI) > \overline{P}(NA), \tag{3.7}$$

In [12], if two or more attributes satisfy the conditions in (3.7) and overlap in their lower and upper probabilities, the attribute variable with the highest upper probability

for CI is selected. This can result in an optimistic strategy by considering the maximum potential support for a class label of the future instance. As the lower probability reports the evidence in favour of the class label of the future instance, it is of interest to explore selecting the attribute with the maximum NPI lower probability for CI in case of multiple attributes overlap in their lower and upper probabilities and satisfy the conditions in (3.7). In the case of multiple attribute variables that satisfy the conditions (3.7) and have the same NPI lower probabilities and same NPI upper probabilities, any of these variables may be selected. If there is no attribute that satisfies the conditions (3.7), a node is not split further.

The aim of introducing the algorithms relies on the criterion of selecting the best attribute to split the tree and the stopping criterion according to the two conditions in (3.7). By satisfying both conditions in (3.7), the attribute selected to split the tree aims to achieve the highest strength of evidence for the class label of the future instance. Selecting an attribute which satisfies only the condition $\overline{P}_i^*(CI) > \overline{P}(NA)$ is likely to result in a higher level of imprecision in the NPI lower and upper probabilities for CI than an attribute which satisfies both conditions in (3.7). The purpose of varying the selection criterion, whether based on the NPI upper probability for CI, or both the NPI lower and upper probabilities for CI, is to investigate how different decision-making strategies affect the performance of the classifier and to identify the most effective approach in imprecise classification problems.

In this study, the new algorithms use the splitting criterion, CI, as presented in Section 1.5.2. The new algorithms are now introduced with adding the abbreviation IC to the algorithm names to reflect Imprecise Classification.

Algorithm 1: D-NPI-IC1

When building the classification tree, this algorithm stops building the tree at a node if no attribute satisfies the conditions in (3.7). If multiple attributes satisfy the conditions given in (3.7), the attribute with the highest NPI lower and upper probabilities for CI is selected. However, if no attribute has the highest NPI lower and upper probabilities among the attributes but multiple have overlapping CI intervals, the attribute with the highest NPI upper probability for CI is selected. This algorithm uses the same splitting and stopping criteria as in the D-NPI algorithm but it returns a set of predicted class labels rather than a single class label when reaching a leaf node.

Algorithm 2: D-NPI-IC2

In this algorithm, if there are multiple attributes that satisfy the conditions outlined in (3.7), and there is no single attribute that has both the maximum NPI upper and lower probabilities for CI, then the attribute with the highest NPI lower probability for CI is selected to split the tree. This algorithm stops splitting a node further if there is no attribute meeting the conditions (3.7).

Algorithm 3: D-NPI-IC3

This algorithm stops splitting a node further if there is no attribute meets the following condition

$$\overline{P}_i^*(CI) > \overline{P}(NA). \tag{3.8}$$

If two or more attributes satisfy the condition in (3.8), and overlap in their NPI lower and upper probabilities for CI, then the attribute with the highest NPI upper probability for CI is selected. In cases where multiple attributes have identical NPI upper probabilities for CI and identical NPI lower probabilities for CI and satisfy the condition in (3.8), any of these attributes can be selected.

In order to demonstrate how these algorithms select the best attribute, the following example is considered.

Example 3.1. Suppose the NPI lower and upper probabilities for attributes x_1 and x_2 are $[0.40, 0.60]$ and $[0.43, 0.57]$, respectively. Additionally, the NPI lower and upper probabilities if no attribute is used is $[0.33, 0.50]$. These two attributes overlap in their NPI lower and upper probabilities for CI (no single attribute has both maximum NPI lower and upper probabilities). In this case, both attributes x_1 and x_2 satisfy the conditions in (3.7). According to the D-NPI-IC1 algorithm, the attribute with the highest NPI upper probability for CI is selected to split the tree; in this case, it is x_1.

According to the D-NPI-IC2 algorithm, attribute x_2 is selected to split the tree as it achieves the highest NPI lower probability for CI (i.e., $0.43 > 0.40$). This algorithm may select the attribute variable with less imprecise CI interval compared to the D-NPI-IC1 algorithm.

If there are multiple attributes that satisfy the conditions in (3.7) and have the same NPI lower probabilities and the same NPI upper probabilities, then any of these attributes

Algorithm 3.1 Classification of a new instance.

1: **Input:** Classification Tree ($\mathcal{CT}$) and New Instance ($\mathbf{x}$)
2: **Procedure** ClassifyInstance ($\mathcal{T}, \mathbf{x}$)
3: Apply $\mathbf{x}$ in $\mathcal{CT}$ to reach a leaf node.
4: Obtain probability intervals for each class label in the terminal node using the NPI-M model: $\{[l_r, u_r], r = 1, \ldots, R\}$.
5: Apply the strong dominance criterion to the probability intervals from the previous step to get a set of non-dominated labels for $\mathbf{x}$.

can be selected according to algorithms D-NPI-IC1 and D-NPI-IC1. For instance, suppose that the attributes x_1 and x_2 have the same NPI lower probabilities and the same upper probabilities, $[0.43, 0.57]$. Then, either x_1 or x_2 can be selected to split the tree.

Based on algorithm D-NPI-IC3, Both attributes x_1 and x_2 satisfy the condition in (3.8). Without comparing the NPI lower probabilities for CI for the attributes, the attribute variable with the highest NPI upper probability for CI is selected. Therefore, the attribute x_1 is selected to split the tree.

Pseudocodes outlining these three algorithms are provided in Appendix C. After introducing the process of building classification trees using the three algorithms, the steps for classifying a new instance, $\mathbf{x}$, when reaching a leaf node are presented in Algorithm 3.1.

The next section presents various measures for evaluating the performance of imprecise classifiers as well as statistical tests, which will be used in Section 3.5.

3.4 Performance measures

This section outlines the appropriate measures to evaluate the performance of imprecise classifiers, along with the statistical tests used for comparative analysis.

3.4.1 Performance evaluation metrics

In the context of imprecise classification, several measures are used to evaluate the performance of imprecise classifiers. It is important to consider two main aspects: first, if the classifier correctly predicts the class label, i.e., if the correct class value falls within the set of predicted labels, and secondly, the informativeness of the set of predicted labels, quantified by its cardinality. Let n be the total number of instances in the dataset, and

let E_t represent the classifier's prediction for instance t, where E_t is the set of predicted labels assigned to instance t. The cardinality of a set E_t, denoted by $|E_t|$, is the number of labels contained in E_t. An instance t is classified precisely if $|E_t|= 1$, meaning the classifier returns only one label for that instance. The following metrics are defined by Corani and Zaffalon [38]:

- **Determinacy**: This is the proportion of instances with a precise classification. The Determinacy (DET) is defined as:

$$\text{DET} = \frac{1}{n}\sum_{t=1}^{n}\mathbb{I}(|E_t|= 1), \tag{3.9}$$

  where $\mathbb{I}(|E_t|= 1)$ is an indicator function that equals 1 if $|E_t|= 1$ (i.e., the instance is classified precisely), and 0 otherwise.

- **Single Accuracy**: This is the accuracy of the classifier when the predicted class is a single label. Let n_{p} denote the number of instances where the classifier predicts the label precisely, and let n_{cp} represent the number of instances among these n_{p} where the classifier correctly predicts the correct label. The Single Accuracy (SingleA) is

$$\text{SingleA} = \frac{n_{\text{cp}}}{n_{\text{p}}}. \tag{3.10}$$

- **Set Accuracy**: This is the proportion of the sets of predicted labels where the correct class label is among the corresponding set of the predicted labels. Let n_{s} denote the number of instances where the classifier predicts more than one state (i.e., there are multiple non-dominated labels). Let n_{cs} represent the number of instances among these n_{s} for which the actual class label is among the set of predicted labels. The Set Accuracy (SetA) can be expressed as:

$$\text{SetA} = \frac{n_{\text{cs}}}{n_{\text{s}}} \tag{3.11}$$

- **Indeterminacy Size**: This returns the average size of the set of predicted labels. This measure reflects the indeterminacy of the classifier when it outputs indetermi-

nate predictions. The Indeterminacy Size (IS) can be expressed as:

$$\text{IS} = \frac{1}{n_\text{s}} \sum_{t=1}^{n_\text{s}} |E_t|. \tag{3.12}$$

Each of these measures addresses one of the two main aspects of performance evaluation, namely whether the classifier predicts correctly or how informative the set of predicted labels is. For example, the Determinacy and Indeterminacy Size tell how informative the classifier is, while the Single Accuracy and Set Accuracy indicate if the classifier makes correct predictions or not. These metrics have been previously used by Corani and Zaffalon work [38], Abellán and Masegosa [4], and others in imprecise classification tasks.

It is also essential to assess the overall performance of the classifiers comprehensively, for which two additional metrics are used. The first metric is the Discounted Accuracy [38]. Let c_t be the correct class for instance t, and let E_t be the set of predicted labels for instance t. The Discounted Accuracy (DACC) is defined as:

$$\text{DACC} = \frac{1}{n} \sum_{t=1}^{n} \frac{\mathbb{I}(c_t \in E_t)}{|E_t|}, \tag{3.13}$$

where $\mathbb{I}(c_t \in E_t)$ denotes the indicator function, which equals 1 if the set of predicted labels contains the correct label, and 0 otherwise. This metric evaluates the performance of classification models based on the two key aspects of performance evaluation discussed earlier. It assigns a value of 0 if the classifier's prediction for a set of predicted labels is incorrect, meaning that the correct class value is not within the set of predicted labels. Conversely, when the classifier outputs a set of predicted labels that contains the correct class label, it adds a value to the metric, which is penalised by the size of the non-dominated labels. The optimal value of this metric is 1 when the classifier returns a set containing only one non-dominated labels for each instance in the dataset, and all predictions are correct. If the classifier's set of predicted labels includes all possible class labels (denoted by R), then the DACC value will be $\frac{1}{R}$. The value of DACC is 0 when the classifier misclassifies all instances in the testing dataset.

Another Measure for Imprecise Classification, denoted by MIC, has been introduced by Abellán and Masegosa [4]. This metric aims to deal with prediction errors by adding

a value when the classifier correctly predicts the labels among the set of predicted labels, depending on the value $\frac{|E_t|}{R}$. However, when the prediction is wrong, it adds a constant value depending on R. The MIC metric is defined as:

$$\text{MIC} = \frac{1}{n}\left(-\sum_{t:success} \log\frac{|E_t|}{R} + \frac{1}{R-1}\sum_{t:error} \log R\right). \tag{3.14}$$

The higher the value of MIC, the better the performance of the classifier, with the optimal value being $\log R$ when the classifier correctly and precisely predicts the set of labels for all instances in the testing dataset. The MIC value is 0 when the set of predicted labels is the same as the full set of class labels R, indicating that the classifier is not informative. The DACC metric assigns 0 to incorrect predictions, as it sums 0 when the classifier predicts a set of predicted labels that does not contain the correct class label. Unlike the DACC metric, each incorrect prediction is assigned a value of $\frac{\log R}{R-1}$ in the MIC. If the imprecise classifier always predicts the full set of class labels R, the value of DACC is $\frac{1}{R}$, and the value of MIC is 0, which is lower, indicating that the classifier is not informative. In [4], the MIC metric takes into account different degrees of importance of errors using a cost matrix of the errors. Since specifying these degrees of importance of the errors requires an expert in a specific domain, this study focuses on adding a constant value for all errors (i.e., assuming the same level of importance for all errors). Investigating the performance of imprecise classifiers in a particular area using the MIC metric with different levels of importance of errors is an interesting topic for future research.

3.4.2 Statistical tests

Statistical tests are used to evaluate the performance of classification algorithms, providing an approach for comparing their effectiveness over multiple datasets. Demšar [46] examined two tests for comparing the performance of multiple classifiers over multiple datasets, namely ANOVA test and Friedman test. Demšar [46] concluded that the Friedman test is safer than the ANOVA test, as it does not assume normality or homogeneity of variance. If the null hypothesis of the Friedman test is rejected, a post-hoc statistical test is used, which is named Nemenyi test. The Friedman test and Nemenyi test are defined as follows:

- **Friedman test**: A nonparametric statistical test used to compare the performance of different algorithms across multiple datasets [58]. The null hypothesis states that all classifiers have equivalent performance across multiple datasets. The test involves ranking the classifiers' performances within each dataset, assigning rank 1 to the best-performing classifier, rank 2 to the second-best classifier, and so on. The ranks are then analysed to determine if there are statistically significant differences between the classifiers. Let k be the number of classifiers and m be the number of datasets. Let S_{dl} be the rank of the l-th classifier on the d-th dataset based on a performance metric. Let $S_l = \sum_{d=1}^{m} S_{dl}$ denote the sum of ranks for classifier l across all m datasets. The test statistic (F) is calculated as follows:

$$F = \frac{12}{mk(k+1)} \sum_{l=1}^{k} S_l^2 - 3m(k+1) \tag{3.15}$$

  Under the null hypothesis, F approximately follows a chi-squared distribution with $k-1$ degrees of freedom: $F \sim \chi^2_{k-1}$. If significant differences are detected with the test statistic F, further post-hoc test can be performed to determine which classifiers differ in performance.

- **Nemenyi test**: A post-hoc statistical test used when the null hypothesis of Friedman test is rejected, indicating that at least one classifier performs differently from the others [88]. It compares all classifiers with each other to determine which pairs of classifiers reveal statistically significant differences in their performance. The critical difference (CD) is defined by

$$\mathrm{CD} = q_{\alpha,k} \sqrt{\frac{k(k+1)}{6m}} \tag{3.16}$$

  where $q_{\alpha,k}$ is the critical value from the Studentized range statistic (based on the desired significance level α and the number of classifiers k) divided by $\sqrt{2}$ [46]. If the difference in the average ranks of two classifiers is greater than the critical difference, CD, then the null hypothesis of the performance of the Nemenyi test is rejected.

3.5 Performance evaluation

In this section, the experiments evaluate the performance of the algorithms proposed in Section 3.3, with comparisons to existing methods. For this purpose, the classifiers NPI-M and A-NPI-M are included [6] (see Section 3.2), along with two IDM-based classifiers [1] using hyperparameter values $s = 1$ and $s = 2$. Section 3.5.1 describes the experimental setup, including datasets, classifiers, and evaluation metrics, while Section 3.5.2 presents and discusses the results in detail.

3.5.1 Experimental setup

In this experiment, three algorithms for imprecise classification are implemented, with the aim of evaluating their performance in comparison to existing methods. For comparison, two classifiers introduced by Abellán et al. [7] and two additional classifiers proposed by Abellán and Masegosa [4] are included. These classifiers are referred to as NPI-M, A-NPI-M, IDM1 (with hyperparameter $s = 1$), and IDM2 (with hyperparameter $s = 2$).

Twenty-one publicly available datasets were obtained from the UCI *Machine Learning Repository* [75]. These datasets vary in terms of size, number of attributes, number of class values, and the range of categories for nominal attributes. Table 3.1 summarises these datasets. The column "n" represents the size of the dataset, "Att" indicates the number of attributes, "Num" shows the number of numerical attributes, "Nom" displays the number of nominal attributes, "R" denotes the number of class values, and "Range" specifies the range of the categories for nominal attributes.

To prepare the datasets for classification, missing values in any dataset are replaced by the mode for nominal attributes and the mean for numerical attributes. Discretisation is applied to pre-process continuous attributes discretised using the '`discretize`' function from the '`FSelectorRcpp`' package in R [114], applying Fayyad and Irani's discretisation method, or the equal-width binning method when no valid cut points are found. For the Waveform dataset, the equal frequency method from the package '`arules`' in R is used via the '`discretize`' function [66], since some attributes in the Waveform dataset cannot be discretised using the '`discretize`' function from the '`FSelectorRcpp`' package.

The primary purpose of this experiment is to evaluate the classification performance of the proposed imprecise classifiers D-NPI-IC1, D-NPI-IC2, and D-NPI-IC3 compared to

other imprecise classifiers. The classifiers, namely NPI-M, A-NPI-M, IDM1, and IDM2, are implemented using the '`imptree`' package in R [56], and the strong dominance criterion is applied when reaching a leaf node for classifying a new instance. These classifiers are selected because they are based on imprecise probability.

For each dataset, a 10-fold cross-validation procedure is applied to mitigate bias and provide a robust evaluation of classifier performance [78].

The evaluation metrics used are Determinacy (DET), Single Accuracy (SingleA), Set Accuracy (SetA), Indeterminacy Size (IS), Tree Depth (TD), DACC, and MIC. Finally, statistical tests are used to compare the performance using the metrics DACC and MIC, across the different classifiers, as presented in Section 3.4. The Friedman test is applied, if the null hypothesis is rejected, the Nemenyi post-hoc test is applied with significance level $\alpha = 0.05$ (see Section 3.4 for more details about the tests).

The algorithms D-NPI-IC1, D-NPI-IC2, and D-NPI-IC3 are implemented by following the procedures outlined in Section 3.3 (Pseudocodes for these algorithms are presented in Appendix C). All experiments are conducted using the statistical software R.

3.5.2 Results and discussion

This section presents the results of evaluating the proposed algorithms against well-known algorithms using various performance metrics. To maintain clarity and conciseness, the algorithms are abbreviated in the tables as follows: the three new algorithms are referred to as 'IC_x' for $x = 1, 2, 3$, 'NPIM' corresponds to Algorithm NPI-M, and 'ANPIM' stands for Algorithm A-NPI-M. The abbreviations of the two algorithms, IDM1 and IDM2, are not changed.

Table 3.2 presents the results for each algorithm, averaged over all datasets, according to the metrics: Determinacy (DET), Single Accuracy (SingleA), Set Accuracy (SetA), Indeterminacy Size (IS), and Tree Depth (TD). The complete results for all metrics, for each dataset, are provided in Appendix C.

As seen in Table 3.2, the evaluation according to the DET metric, averaged across all datasets, reveals that algorithm IC_2 achieves the highest value. This algorithm differs from the IC_1 algorithm when multiple attributes have overlapping CI intervals. In such cases, the IC_2 algorithm tends to select the attribute with the less imprecise CI interval compared to the IC_1 algorithm. The IC_2 imprecise classifier tends to be quite confident

Dataset	n	Att	Num	Nom	R	Range
Balance-Scale	625	4	4	-	3	-
CMC	1473	9	2	7	3	2-4
Dermatology	366	34	1	33	6	2-4
Hayes	160	4	-	4	3	3-4
Hypothyroid	3772	27	6	21	4	2-5
Iris	150	4	4	-	3	-
Letter	20000	16	16	-	26	-
Lymphography	148	18	-	18	4	2-8
Nursery	12960	8	-	8	5	2-5
Page-Blocks	5473	10	10	-	5	-
Pendigits	10992	16	16	-	10	-
Primary-Tumor	339	17	-	17	21	2-3
Segment	2310	16	16	-	7	-
Soybean	683	35	-	35	19	2-7
Splice	3190	60	-	60	3	4-6
Sponge	76	44	-	44	3	2-9
Vehicle	846	18	18	-	4	-
Vowel	990	11	10	1	11	2
Waveform	5000	40	40	-	3	-
Yeast	1484	6	6	-	10	-
Zoo	101	17	1	16	7	2

Table 3.1: Datasets [75]. Size "n", number of attributes "Att", number of numerical attributes "Num", number of nominal attributes "Nom", number of class labels "R", and range of values of nominal attributes "Range".

in making single predictions (around 93%). The IC_1 algorithm follows closely with approximately 92%. In contrast, the IC_3 algorithm is less confident about producing single predictions, as it tends to build deeper trees than the other algorithms. The excessive splits in the IC_3 algorithm, compared to the IC_1 algorithm, can result in reaching leaf nodes with few instances, implying less confidence about producing single predictions. Algorithms NPIM, ANPIM, and IDM2 show to have lower determinacy compared to algorithms IC_1 and IC_2. The IDM1 algorithm tends to be more confident about predicting single predictions compared to IDM2, with a DET value approximately 3% higher than that of IDM2.

Regarding the accuracy of the classifier when the prediction is a single label (SingleA), algorithms with higher values of DET tend to produce slightly more incorrect predictions than those with lower DET. That is because, in some cases where certain algorithms are not confident in predicting a single label, others have higher confidence, which increases the risk of incorrect predictions. This can be observed in the IC_1, IC_2, and IC_3 algorithms,

Average	**IC_1**	**IC_2**	**IC_3**	**NPIM**	**ANPIM**	**IDM1**	**IDM2**
DET	0.9178	**0.9313**	0.8675	0.8670	0.8667	0.8949	0.8629
SingleA	0.8381	0.8306	**0.8501**	0.8081	0.8080	0.7997	0.8056
SetA	0.9359	0.9369	**0.9515**	0.9139	0.9154	0.9017	0.9046
IS	6.5446	6.7044	6.2187	4.7686	4.8087	**4.1197**	4.2833
TD	5.7048	5.9238	6.3333	5.1429	5.1286	5.2952	**4.9333**

Table 3.2: Summary results of measures: DET, SingleA, SetA, IS, and TD for all algorithms.

where the IC_2 algorithm which achieves the highest value of DET, also has the lowest SingleA value, and vice versa for the IC_3 algorithm. The SingleA values for the last four algorithms are relatively close.

A high value of DET means that the classifier returns a lower percentage of indeterminate predictions. In the case of the IC_3 algorithm, out of approximately 13% of indeterminate predictions, the accuracy is 95.15% (SetA), which is relatively high compared to the other algorithms. However, this high SetA value does not always indicate better performance of the classifier, as it might return a set of predicted labels with a large size. Therefore, using the DACC and MIC measures is advantageous, as they provide meaningful insights into the tradeoff between correct prediction and the size of the set of predicted labels. Algorithms NPIM and ANPIM perform similarly when producing indeterminate predictions, and the same applies to algorithms IDM1 and IDM2, although both have lower SetA values than those of algorithms NPIM and ANPIM.

Algorithms IDM1 and IDM2 have the lowest values of IS compared to the other algorithms, with the lowest value achieved by the IDM1 algorithm. The IC_1 algorithm achieves the highest value of IS; this may explain its high value of SetA. Similarly, the low IS values observed for the IDM1 and IDM2 algorithms correspond to their lower SetA values. This can be intuitively explained by the fact that, as the size of the set of predicted labels excessively increases, the classifiers' ability to include the correct class label within the set increases. Conversely, an excessive decrease in the size of the set of non-dominated labels has the opposite effect. The first three classifiers tend to produce prediction sets whose sizes correspond to approximately 86%, 89%, and 82%, respectively, of the average size of the full labels set, while classifiers NPIM and ANPIM include around 63% and 64%, respectively, of the full possible set of labels on average.

In terms of the TD measure, the values for all algorithms, except for the IC_3 algorithm,

Dataset	**IC$_1$**	**IC$_2$**	**IC$_3$**	**NPIM**	**ANPIM**	**IDM1**	**IDM2**
Balance-Scale	0.6833	0.6884	0.7193	**0.7241**	**0.7241**	0.7017	**0.7241**
CMC	0.5226	0.5451	0.4875	0.5442	0.5442	0.5416	**0.5455**
Dermatology	**0.9432**	0.9204	0.9359	0.8903	0.8903	0.8981	0.8835
Hayes	**0.7115**	0.6927	0.6833	0.6781	0.6781	0.6729	0.7021
Hypothyroid	**0.9930**	**0.9930**	0.9922	0.9881	0.9881	0.9897	0.9888
Iris	**0.9533**	0.9400	**0.9533**	0.9444	0.9444	**0.9533**	**0.9533**
Letter	0.7330	**0.7404**	0.7240	0.5016	0.5021	0.5081	0.5011
Lymphography	0.7552	0.7811	0.7338	0.7807	0.7807	**0.7932**	0.7742
Nursery	0.9182	0.9181	0.9262	0.9475	0.9465	**0.9624**	0.9376
Page-Blocks	**0.9652**	0.9651	0.9614	0.9572	0.9572	0.9601	0.9551
Pendigits	0.8665	**0.8707**	0.8537	0.8000	0.7998	0.8219	0.7980
Primary-Tumor	**0.3637**	0.3457	0.3180	0.3488	0.3466	0.3564	0.3623
Segment	0.9279	**0.9433**	0.9164	0.7759	0.7756	0.7864	0.7732
Soybean-Large	0.8192	0.8395	0.7975	0.9005	0.9005	**0.9033**	0.8376
Splice	**0.8847**	0.8812	0.8841	0.6969	0.6969	0.7016	0.6969
Sponge	**0.9232**	**0.9232**	0.8911	**0.9232**	**0.9232**	**0.9232**	**0.9232**
Vehicle	0.7058	**0.7112**	0.6899	0.6145	0.6145	0.6219	0.6047
Vowel	0.7351	**0.7441**	0.7299	0.5325	0.5306	0.5729	0.5262
Waveform	**0.7669**	0.7661	0.7488	0.7546	0.7546	0.7554	0.7483
Yeast	0.5848	**0.5865**	0.5667	0.5678	0.5676	0.5674	0.5610
Zoo	0.9038	0.8747	0.9038	0.8752	0.8752	**0.9352**	0.8574
Average	0.7933	**0.7938**	0.7818	0.7498	0.7496	0.7584	0.7454

Table 3.3: Performance of all algorithms on different datasets (Metric: DACC).

are close, with the IDM2 algorithm achieving the lowest value. The IC$_3$ algorithm has a TD value higher than that of the IC$_1$ algorithm, as the latter selects attribute that satisfies the two conditions in (3.7), while the former is less strict, selecting attribute that satisfies the only one condition in (3.8). It is worth mentioning that NPIM, ANPIM, IDM1, and IDM2 generally have lower TD values across most datasets.

In order to study the performance of the classifiers in terms of the balance between the correct prediction and the size of the set of predicted labels, the DACC and MIC metrics are used. The results for each dataset are presented to support further investigation.

In Table 3.3, the performance of the algorithms with respect to the DACC metric varies across the evaluated datasets. It is observed that the IC$_1$ and IC$_2$ algorithms achieve higher DACC values for most of the datasets. Algorithm IC$_2$ achieves the highest average DACC value, closely followed by algorithm IC$_1$, whereas algorithm IDM2 attains the lowest value.

For the Primray-Tumor dataset, the DACC values are notably low across all algo-

rithms when compared to the other datasets. This dataset is relatively small in size and is characterised by a high number of attributes and class labels, a presence of missing values, and an imbalanced class distribution. The classifiers demonstrate high indeterminacy for this dataset compared to others. Among the classifiers, algorithm IC_3 observed to be the most indeterminate classifier, returning approximately 50% of indeterminate predictions while achieving the lowest DACC value (see Appendix C). The IC_2 achieves the highest DET value, but the IC_1 has the highest DACC value. This is due to the IC_1 algorithm achieving a higher SingleA value compared to the IC_2 algorithm, with IC_2 having slightly higher SetA and IS values than the IC_1 algorithm. As a result, the IC_1 algorithm achieves the best balance between correct predictions and the sizes of the prediction sets according to the DACC measure. The IDM2 follows closely the IC_1 algorithm.

All the classifiers have high performance for the Iris and Page-Blocks datasets, which are defined by their relatively small number of continuous attributes, small number of class labels, and no missing values. For the Iris dataset, IC_1, IC_2, IC_3, IDM1, and IDM2 are 100% determinate. Notably, the IDM1 and IDM2 algorithm show identical behaviour and performance for the Iris dataset and same happens for both algorithms IC_1 and IC_3. The NPIM and ANPIM algorithms have lower DET values compared to the other algorithms, along with slightly higher TD values, which affects their overall lower DACC values.

For the Dermatology dataset, algorithm IC_1 achieves the most favourable tradeoff between correct predictions and the set of predicted labels, achieving higher value of DACC compared to the other algorithms. The first three algorithms have high determinacy and SingleA values compared to the remaining algorithms for this dataset, which contributes to their high values of DACC. Notably, the IC_1 achieves the highest DET and SingleA values.

For the Letter dataset, the gap between the DACC values for the first three algorithms and the last four algorithms is due to the low SingleA and SetA values observed for the latter algorithms compared to the first. The NPIM, ANPIM, IDM1, and IDM2 algorithms tend to stop building tree early (low TD values), producing prediction sets with smaller sizes compared to those of the algorithms IC_1, IC_2, and IC_3. For these reasons, the algorithms IC_1, IC_2, and IC_3 have better performance in terms of the DACC measure. Also, these algorithms have better performance for the Splice dataset, as they have high

SingleA and SetA values compared to the NPIM, ANPIM, IDM1, and IDM2 algorithms, which resulted in their higher accuracy values with respect to the DACC measure.

In the case of the Pendigits dataset, which contains a moderate number of instances, the first three algorithms have good performance, with the best DACC value achieved by algorithm IC_2. Although this dataset has a moderate number of class labels, it is characterised by a balanced class distribution. Algorithms IC_1, IC_2, and IC_3 have high values of SingleA and SetA compared to the NPIM, ANPIM, IDM1, and IDM2 algorithms, which explain their high values of DACC. The first three algorithms tend to produce deeper trees (i.e., higher values of TD), and when combined with the balanced class distribution, this can lead to identifying more meaningful splits, resulting in a higher percentage of correct predictions. Among the last four algorithms, algorithm IDM1 has a relatively high DACC value, which can be attributed to its low value of IS.

The NPIM, ANPIM, and IDM1 algorithms demonstrate strong performance in terms of their DACC values compared to other algorithms for the Nursery and Soybean datasets. For the Soybean dataset, it has a relatively high number of missing values, a moderate size, a high number of attributes, and imbalanced class distribution. The NPIM, ANPIM, and IDM1 algorithms tend to be highly confident in predicting single labels, often accompanied by high SingleA values. In contrast, the IC_1 and IC_2 algorithms have high determinacy but with low SingleA and SetA values and high IS values, resulting in their low DACC values compared to algorithms NPIM, ANPIM, and IDM1. Regarding the Nursery dataset, the IC_1 and IC_2 algorithms have low SingleA values, which impacts their low DACC values compared to the NPIM, ANPIM, and IDM1 algorithms.

The first three algorithms can have different structure of the trees due to their internal criteria for selecting the most informative attribute to split the trees, and their stopping criteria. In particular, selecting attributes based on the two conditions in (3.7) for the algorithms IC_1 and IC_2 results in better performance than selecting attributes based on the single condition given in (3.8).

For further investigation, performance is evaluated and presented using the MIC metric across all datasets. The results of the MIC in Table 3.4, on average, indicate that the IC_2 algorithm achieves the highest value, closely followed by the IC_1 algorithm, while the lowest value is obtained by the IDM2 algorithm. Notably, the average MIC metric values for Algorithms IC_3 and IDM1 show minimal variation, indicating comparable

Dataset	**IC$_1$**	**IC$_2$**	**IC$_3$**	**NPIM**	**ANPIM**	**IDM1**	**IDM2**
Balance-Scale	0.9233	**0.9244**	0.9085	0.8426	0.8426	0.8694	0.8426
CMC	0.7625	**0.7957**	0.6289	0.7385	0.7385	0.7693	0.7492
Dermatology	**1.6891**	1.6499	1.6755	1.6069	1.6069	1.6299	1.5972
Hayes	**0.8433**	0.8278	0.7304	0.7858	0.7858	0.8039	0.7970
Hypothyroid	**1.3794**	**1.3794**	1.3780	1.3709	1.3709	1.3742	1.3724
Iris	**1.0730**	1.0657	**1.0730**	1.0583	1.0583	**1.0730**	**1.0730**
Letter	2.3969	**2.4261**	2.3674	1.7496	1.7505	1.7691	1.7608
Lymphography	1.0740	1.1368	1.0264	1.0824	1.0824	**1.1372**	1.1008
Nursery	1.5094	1.5093	1.5101	1.5432	1.5417	**1.5629**	1.5340
Page-Blocks	1.5616	**1.5628**	1.5522	1.5525	1.5525	1.5580	1.5512
Pendigits	1.9978	**2.0097**	1.9617	1.8871	1.8865	1.9382	1.8834
Primary-Tumor	1.1499	1.1120	1.0141	1.1904	1.1825	1.2036	**1.2519**
Segment	1.8089	**1.8399**	1.7850	1.5682	1.5679	1.5962	1.5684
Soybean-Large	2.4354	2.4941	2.3803	2.6747	2.6747	**2.6949**	2.5708
Splice	**1.0175**	1.0142	0.9988	0.8136	0.8136	0.8417	0.8313
Sponge	**1.0564**	**1.0564**	0.9917	**1.0564**	**1.0564**	**1.0564**	**1.0564**
Vehicle	1.0641	**1.0752**	1.0336	0.9768	0.9768	1.0001	0.9666
Vowel	1.8109	**1.8311**	1.7990	1.4269	1.4199	1.5268	1.4095
Waveform	0.9561	**0.9575**	0.9202	0.9212	0.9212	0.9287	0.9336
Yeast	1.4425	**1.4457**	1.4074	1.4122	1.4115	1.4240	1.3997
Zoo	1.7834	1.7316	1.7834	1.6879	1.6879	**1.8241**	1.6540
Average	1.4160	**1.4212**	1.3774	1.3308	1.3300	1.3610	1.3288

Table 3.4: Performance of all algorithms on different datasets (Metric: MIC).

performance. Furthermore, the NPIM and ANPIM algorithms perform equally for most of the datasets with the NPIM algorithm showing slightly higher MIC values for some datasets. The findings per each dataset for the MIC metric align with those of when using the DACC metric with small variations for some datasets.

For the Balance-Scale dataset, interestingly, the first three algorithms have higher MIC values compared to the other algorithms. Upon further investigation, it is observed that the algorithms IC$_1$ and IC$_2$ are approximately 99% determinate with low SingleA and SetA values, while the NPIM, ANPIM, IDM1, and IDM2 algorithms are less determinate, having high values of SingleA and SetA. To explain, in terms of the DACC measure presented in Table 3.3, the performance of the IC$_1$ and IC$_2$ algorithms is affected by the the low values of SingleA and SetA, resulting in their lower DACC values. When applying the MIC measure, a constant value of $\frac{\log 3}{2} \simeq 0.55$ is added for each wrong prediction, and either $-\log \frac{2}{3} \simeq 0.41$ when the cardinality of the set of predicted labels is 2, or 0 when the cardinality is 3. As the IC$_1$ and IC$_2$ algorithms have incorrect predictions more than

the NPIM, ANPIM, IDM1, and IDM2 algorithms, as reflected in their values of SingleA and SetA, the value 0.55 is assigned to each wrong prediction more frequently, leading to increased MIC value. Similar reasoning explains the observed results for the CMC dataset.

In the case of the Primary-Tumor dataset, the IC_1 achieves the highest DACC value, while the IDM2 achieves the highest MIC value. As the IC_1 algorithm has IS approximately value 15, while IDM2 has a value of approximately 11, this difference affects the values that MIC adds for each correct imprecise prediction. In the case of IDM2, the MIC tends to assign higher values for correct imprecise predictions, which results in its high MIC value.

For further comparison, the performance of all algorithms is statistically tested using the Friedman rank test, followed by the Nemenyi test when significant differences are detected. Table 3.5 presents the results of the metrics DACC and MIC. The reason for selecting these two metrics, as discussed in Section 3.4, is that they capable of evaluating the performance of the classifiers while considering if the predictions are correct or not along with the size of the set of the predicted labels. These metrics have also been used in the literature for evaluating the performance of imprecise classifiers, including the work by Abellán and Masegosa [4]. In Table 3.5, the average values of the metrics, the Friedman rank, and the Nemenyi test results are presented for each algorithm.

For the DACC measure, the Friedman test results in a p-value of 0.0011, suggesting that there are significant differences between the algorithms at the 5% level. According to the results in Table 3.5, the algorithm IC_1 achieves the best average rank among the other classifiers, suggesting consistent performance across datasets, while the algorithm IC_2 achieves the highest average value of DACC. To further investigate these differences, the Nemenyi test is conducted, and both algorithms significantly outperform the IDM2 algorithm, while no significant differences are observed between the other algorithms. The algorithm IDM2 has the highest rank and the lowest average value of DACC, indicating overall weak performance compared to the other classifiers. This finding can be attributed to the observed behaviour of the IC_1 and IC_2 algorithms being more determinate compared to the other algorithms, along with achieving high SingleA and SetA values, resulting in their overall high accuracy.

According to the MIC measure, the Friedman test has a p-value of 1.142^{-7}, thereby

Algorithm	Average	Friedman Rank	Nemenyi
		DACC	
IC_1	0.7933	**2.8095**	**IDM2**
IC_2	**0.7938**	3.0476	**IDM2**
IC_3	0.7818	4.2857	-
NPIM	0.7498	4.4762	-
ANPIM	0.7496	4.7143	-
IDM1	0.7584	3.5238	-
IDM2	0.7454	5.1429	-
		MIC	
IC_1	1.4160	2.7143	**NPIM**, **ANPIM**, **IDM2**
IC_2	**1.4212**	**2.4286**	**IC_3, NPIM, ANPIM, IDM2**
IC_3	1.3774	4.6190	-
NPIM	1.3308	5.0238	-
ANPIM	1.3300	5.2619	-
IDM1	1.3610	3.0000	**NPIM**, **ANPIM**
IMD2	1.3288	4.9524	-

Table 3.5: Performance comparison of all algorithms with the metrics: DACC and MIC. Column "Nemenyi " shows the classifiers, in which the classifier in the row is significantly better based on the Nemenyi test.

indicating statistically significant differences among the algorithms. In Table 3.5, algorithm IC_2 achieves both the best rank and average values among the classifiers, followed closely by algorithm IC_1, then the IDM1 algorithm. After applying the Nemenyi test, the results show that IC_1 and IC_2 perform significantly better than NPIM, ANPIM, and IDM2. Moreover, IC_2 significantly outperforms IC_3. Additionally, IDM1 demonstrates better performance than NPIM and NPIM. The observed reason for the low performance of the NPIM, ANPIM, and IDM2 algorithms in terms of MIC is their lower DET values, which results in more imprecise predictions. As discussed previously, imprecise prediction can lead to a lower value of MIC as long as the cardinality is sufficiently low. This can be easily achieved if the number of classes in the dataset is low. These differences in MIC values are particularly noticeable for the datasets which have 3 class labels, as imprecise predictions can take only cardinality value between 2 and 3. The large differences between the algorithms according to this metric can be reduced for datasets with higher number of class labels.

3.6 Conclusions

In this chapter, three new algorithms were adapted to imprecise classification based on the NPI approach and the CI splitting criterion. These algorithms are D-NPI-IC1, D-NPI-IC2, and D-NPI-IC3. They were developed to investigate the performance of the D-NPI algorithm when adapted to imprecise classification, considering different criteria for selecting the best attribute and for stopping the tree-building process. In this study, after a tree is built using an algorithm, a probability interval from the NPI-M model is assigned to each class label at the leaf nodes. The strong dominance criterion is then applied to obtain the set of predicted labels.

Different measures for imprecise classification and statistical tests were presented in this study to investigate the performance of the classifiers. The measures DACC and MIC were important for assessing the overall performance of the algorithms. The DACC measure considers whether the set of predicted labels contains the correct class label, and the cardinality of the set of predicted labels. The MIC metric accounts for the same aspects as the DACC metric, while in addition assigning a constant value for incorrect prediction.

An experimental study was conducted to evaluate the performance of the proposed algorithms and compare them with the algorithms NPI-M, A-NPI-M, IDM1, and IDM2 from the literature. The NPI-M and A-NPI-M algorithms are based on the NPI approach, while the IDM1 and IDM2 are based on the IDM with the hyperparameter $s = 1$ and $s = 2$, respectively.

The results of this study revealed that the D-NPI-IC1 and D-NPI-IC2 algorithms tended to be less indeterminate compared to the other algorithms, with having high single accuracy and set accuracy values. The D-NPI-IC3 was less determinate but achieved the highest single and set accuracy values. The D-NPI-IC1, D-NPI-IC2, and D-NPI-IC3 tended to have deep trees. The NPI-M, ANPI-M, IDM1, and IDM2 algorithms achieved low indeterminacy size values compared to the proposed algorithms.

Regarding the DACC and MIC measures, the results indicated that the D-NPI-IC1 and D-NPI-IC2 algorithms demonstrated the best performance compared to the other algorithms. In terms of DACC, they achieved a balance between high accuracy values when the predictions were either precise or imprecise. They showed significantly better

performance than the IDM2 algorithm for this metric.

In case of the MIC measure, the D-NPI-IC1 and D-NPI-IC2 algorithms achieved the highest performance, showing significantly better performance than the NPI-M, A-NPI-M, and IDM2 algorithms. The observed reason for this was that the NPI-M, A-NPI-M, and IDM2 algorithms were less determinate compared to the D-NPI-IC1 and D-NPI-IC2 algorithms, which affected the resulting MIC values, particularly for datasets with low values of class labels.

The results showed that the proposed algorithms D-NPI-IC1 and D-NPI-IC2 achieved best performance compared to the other imprecise classifiers from the literature, indicating their potential as promising methods for imprecise classification problems.

In the D-NPI algorithm for multinomial data, Alharbi [12] derived the NPI lower and upper probabilities for the event that a future instance has a specific class based on NPI for Bernoulli data. However, he mentioned that using NPI for multinomial data instead might result in slightly larger imprecision. It is an interesting topic for future research to use NPI for multinomial data to derive these probabilities and investigate the performance when applied to imprecise classification tasks.

It will be interesting to extend the algorithms to weighted cost sensitive classification [104], aiming to improve the performance of the algorithms in scenarios where misclassification costs vary. It provides a more effective way to handle the issues related to imbalanced datasets than the traditional methods for classification using weights.

Adapting the D-NPI algorithm to imprecise classification by changing the criterion for selecting the best attribute based solely on the NPI lower probability is possible, but needed to be approached with caution. During splitting the tree, if only one of the possible values of an attribute is observed in the data, this can lead to high NPI lower probability for CI. If no appropriate settings are configured in implementation, such attribute can be selected to split the tree, but the splitting will be stopped later due to selecting uninformative attribute. This issue can be further investigated in future.

Chapter 4

Multi-Label classification with NPI

4.1 Introduction

Multi-label classification (MLC) is an area within machine learning where multiple labels can be assigned to each instance simultaneously, unlike traditional single-label classification. It has gained attention due to its applicability across various domains, including text categorization [74], biology [91], and bioinformatics [108], where multiple labels naturally appear. In text categorization [74], for instance, a single document can belong to multiple tags simultaneously, requiring a multi-label classification approach. Similarly, in biology [91], a single clinical report can describe more than one medical condition, further emphasising the necessity for multi-label classification models to assign multiple relevant diagnostic codes to each radiology report. In bioinformatics [108], multi-label classification is used to predict multiple functions for each protein based on its sequence.

The methods used for MLC problems are grouped into two main categories: algorithm adaptation methods and problem transformation methods. Algorithm adaptation methods, such as decision trees [27], neural networks [115], and support vector machines [51], extend traditional classification methods to handle multiple labels directly. Whereas problem transformation methods transform MLC problems into either one or two traditional single-label classification problems. These methods include Binary Relevance (BR) [20, 105], Label Powerset (LP) [105], and Classifier Chains (CC) [98].

Clare and King [27] made a foundational contribution to MLC by extending the well known C4.5 algorithm [94] to handle the problem of genes associated with multiple functions (the multi-label problem). Their study introduced the multi-label C4.5 (ML-C4.5)

algorithm by modifying the entropy splitting criterion for MLC and assigning multiple labels at the leaf nodes.

In 2007, Tsoumakas and Vlahavas [106] introduced an ensemble method for MLC which is called Random k-Labelsets algorithm (RAkEL). By constructing an ensemble of LP classifiers, the RAkEL method effectively considers label correlations avoiding the complications associated with the large number of labels. The results showed that the RAkEL classifier could achieve higher performance compared to the BR and LP classifiers.

Moral-García et al. [83] introduced a Multi-Label Decision Trees (ML-DT) algorithm, which is based on Nonparametric Predictive Inference Model for Multinomial data (NPI-M) [33], using imprecise probability, unlike the methods for decision trees which use precise probability in the splitting criterion. The experiments revealed that the ML-DT algorithm, based on NPI-M, outperformed the existing ML-DT algorithm, which uses precise probability, and proved to be more robust against data noise.

Moral-García and Abellán in [82] proposed two lazy Multi-Label Classification algorithms, Binary Relevance Credal K-Nearest Neighbor (BR-Credal-KNN) and Multi-Label Credal K-Nearest Neighbor (ML-Credal-KNN), which use NPI-M to improve robustness against label noise and address class imbalance, outperforming classical approaches like Multi-Label K-Nearest Neighbor (ML-KNN) and Binary Relevance K-Nearest Neighbor (BR-KNN) in Accuracy and F1 metrics. The experimental results indicated that the ML-Credal-KNN algorithm performed comparably to other algorithms in ranking-based metrics, whereas the BR-Credal-KNN algorithm was less effective in ranking problems. This finding suggested that the ML-Credal-KNN algorithm was better suited for MLC tasks, particularly those involving noise and class imbalance.

In this chapter, the Direct Nonparametric Predictive Inference (D-NPI) algorithm is first applied to multi-label classification (MLC) using the problem transformation approach, specifically the Label Powerset (LP) method, enabling it to handle multiple labels effectively. The LP method is simple and can capture dependencies between labels. However, using other transformation methods may be considered for future research. By converting an MLC problem into single-label classification problems, the D-NPI algorithm can be applied in this setting, thereby extending its applicability to multi-label tasks. The performance of the proposed algorithm is evaluated and compared to other algorithms from the literature across multiple datasets for MLC problems.

To address the limitations of classical MLC methods, which typically produce a deterministic prediction (a single subset of labels), this work also explores the use of imprecise classification algorithms for MLC problems. The aim is to reduce overconfidence by predicting a set of possible subsets of labels rather than a single subset of labels. Instead of forcing the classifier to return a single subset of labels when confidence is low, imprecise classification allows it to provide multiple candidate subsets. By considering a set of predicted subsets of labels, imprecise classification algorithms can enhance the ability to model the uncertainty of data.

This study implements the D-NPI-IC1 algorithm, introduced in Chapter 3 for imprecise classification, alongside the LP method for MLC, to investigate classifier performance in this context. In addition, several established imprecise classifiers from the literature are employed for comparison on MLC problems.

This chapter is structured as follows. Section 4.2 introduces the fundamental framework of the LP method and its application with classification algorithms in two scenarios. It outlines the procedures of the algorithms used in each case, with two detailed and illustrative examples. Section 4.3 describes the performance evaluation measures used to assess the multi-label classifiers. Section 4.4 explores the performance of the proposed methods through experimental studies by comparing them to other methods from the literature. Finally, Section 4.5 concludes the chapter and suggests directions for future research.

4.2 Label Powerset (LP) method for multi-label classification (MLC)

This section outlines the fundamental framework for applying the Label Powerset (LP) method in Multi-Label Classification (MLC). The framework is presented in two sections. Section 4.2.1 explains how multi-label datasets are transformed into single-label datasets, enabling the application of standard classification algorithms. Section 4.2.2 describes the framework for employing imprecise classification algorithms to address MLC tasks.

Algorithm 4.1 MLC using LP approach.

Input: Dataset $\mathcal{D} = \{(\mathbf{x}_t, \mathcal{Y}_t) \mid t = 1, 2, \ldots, n\}$
  foreach unique subset $\mathcal{Y}^r \subseteq \mathcal{Y}$ in $\mathcal{D}$, where $r \in \{1, 2, \ldots, R\}$
    Assign a unique class label $c_{\mathcal{Y}^r} = c_r$, $r \in \{1, 2, \ldots, R\}$
    Create a new dataset $\mathcal{D}' = \{(\mathbf{x}_t, c_t) \mid t = 1, 2, \ldots, n\}$, where $c_t \in \{c_1, c_2, \ldots, c_R\}$
    Train a single-label classifier, H, on the transformed dataset $\mathcal{D}'$
  foreach instance in the testing dataset $\mathbf{x}_j$ where $j = 1, 2, \ldots, n_{\text{new}}$
    Predict the class $\hat{c}_r$ using the trained model H
    Retrieve the corresponding subset $\mathcal{Y}^r$ associated with the predicted class $\hat{c}_r$
Return the predicted subset of labels $\mathcal{Y}^r$

4.2.1 MLC via LP transformation

In Multi-Label Classification (MLC), the problem is defined over T-dimensional attribute space: $\mathcal{X} \subseteq \mathbb{R}^T$, where $x_1, x_2, \ldots, x_T$ denote the T attribute variables. The set of Q labels is defined as: $\mathcal{Y} = \{y_1, y_2, \ldots, y_Q\}$, $Q > 1$. The objective is to learn a predictive model from a dataset: $\mathcal{D} = \{(\mathbf{x}_t, \mathcal{Y}_t) \mid 1 \leq t \leq n\}$, where $\mathbf{x}_t = (x_{t1}, x_{t2}, \ldots, x_{tT}) \in \mathcal{X}$ represents the T-dimensional attribute vector for the t-th instance, and $\mathcal{Y}_t \subseteq \mathcal{Y}$ represent the subset of labels associated with that instance. A label $y_q \in \mathcal{Y}$ is said to be relevant for an instance $\mathbf{x}_t$ if the label y_q is associated with the instance $\mathbf{x}_t$, that is $y_q \in \mathcal{Y}_t$, or else it is considered irrelevant.

Using the LP method, the dataset is first transformed into a higher-dimensional space, where each instance is associated with one of all possible subsets of labels from the set of labels $\mathcal{Y}$. This transformation captures the complexity of multi-label classification by treating each possible combination of labels as a distinct class label, thereby modelling the relationships between instances and their associated subsets of labels. This allows each unique subset of labels as a single class label. This transformation is defined by a mapping function that assigns each multi-label subset to a single class label.

Assume that H is a mapping which transforms a unique subset of labels in $\mathcal{Y}$ into a single class label. Then the mapping function H is defined as:

$$H : \mathcal{X} \to 2^{|\mathcal{Y}|}.$$

For a new instance $\mathbf{x}_{\text{new}}$, $H(\mathbf{x}_{\text{new}})$ is the subset of labels predicted as relevant for the new instance. The LP approach used in this study is outlined in Algorithm 4.1.

In this chapter, the LP method is employed to adapt algorithms originally designed

for single-label (standard) classification to multi-label problems. Five algorithms are considered. Three of these are based on the NPI approach: Direct-NPI (D-NPI) [12], introduced in Section 1.5, and the NPI-M and A-NPI-M algorithms [6], both derived from the NPI-M model. The remaining two algorithms are based on the IDM [1], using hyperparameter values $s = 1$ and $s = 2$, and are referred to as IDM1 and IDM2. These algorithms were also applied in Chapter 3. The following example presents the application of the D-NPI algorithm to MLC problems using the LP method.

Example 4.1. Suppose an artificial dataset consists of categorical attributes related to patients' lifestyle and demographics. Each instance corresponds to a patient and can be associated with multiple conditions. The goal is to illustrate the MLC setting, where more than one label can be assigned to an individual patient. Suppose five attribute variables: an age group, gender, physical activity level, diet type, and smoking status. These attributes are denoted by X_1, X_2 X_3, X_4, and X_5. Suppose the label space consists of four binary class variables: Diabetes, Heart Disease, Obesity, and Hypertension, where each label takes the value 1 if the condition is present and 0 otherwise. These variables are denoted by y_1, y_2, y_3, and y_4. Table 4.1 presents an artificial dataset consisting of 10 patient instances.

In order to apply the Label Powerset (LP) method for multi-label classification, each unique combination of class labels is treated as a single class label. This transformation converts the original MLC problem into a single-label classification problem, where each transformed class label represents a specific combination of the original binary health conditions (e.g., `Diabetes = 0`, `Obesity = 1`, `Heart Disease = 1`, `Hypertension = 0`). The combination of labels is called the subset of label in this chapter. The resulting dataset shows the original input attributes but associates each instance with a single label corresponding to its observed subset of labels. Table 4.1 presents the transformed dataset under this transformation method.

The transformed dataset presented in Table 4.2 is used to construct the classification tree. Using D-NPI classification method, the NPI lower and upper probabilities for each attribute leading to CI are derived, then the best attribute is selected to split the tree. After reaching a leaf node, the most frequent subset of labels is assigned.

In Figure 4.1, the corresponding classification tree is constructed using the transformed dataset in Table 4.2. In this method, after reaching a leaf node, the most frequent

X_1	X_2	X_3	X_4	X_5	y_1	y_2	y_3	y_4
x_2	x_2	x_2	x_2	x_2	0	1	1	0
x_1	x_1	x_1	x_1	x_1	0	0	0	0
x_4	x_2	x_2	x_4	x_2	1	0	0	1
x_3	x_1	x_3	x_5	x_1	0	0	0	1
x_2	x_1	x_2	x_2	x_1	0	0	1	0
x_4	x_1	x_2	x_1	x_2	0	1	0	1
x_1	x_2	x_1	x_3	x_1	0	0	0	0
x_3	x_2	x_3	x_4	x_2	1	0	0	0
x_2	x_1	x_1	x_5	x_1	0	0	0	0
x_4	x_2	x_2	x_2	x_2	1	1	1	0

Table 4.1: Dataset description for Example 4.1.

X_1	X_2	X_3	X_4	X_5	class
x_2	x_2	x_2	x_2	x_2	c_1
x_1	x_1	x_1	x_1	x_1	c_2
x_4	x_2	x_2	x_4	x_2	c_3
x_3	x_1	x_3	x_5	x_1	c_4
x_2	x_1	x_2	x_2	x_1	c_5
x_4	x_1	x_2	x_1	x_2	c_6
x_1	x_2	x_1	x_3	x_1	c_2
x_3	x_2	x_3	x_4	x_2	c_7
x_2	x_1	x_1	x_5	x_1	c_2
x_4	x_2	x_2	x_2	x_2	c_8

Table 4.2: Transformed dataset using LP method for Example 4.1.

class is assigned. The class is then reverse-transformed to its original subset of labels. For example, if a new instance with attribute variable $X_1 = x_3$ is considered, following the classification path from the root to a leaf node in Figure 4.1 leads to a leaf node where the predicted class label is c_4, corresponding to the original subset of labels $[0, 0, 0, 1]$. This indicates the presence of the last label (e.g. y_4) and the absence of the remaining labels.

4.2.2 Imprecise classification for MLC via LP transformation

Imprecise classifiers, when used for single-label classification problems, return a set of predicted states [4]. By integrating imprecise classification with MLC problems, the imprecise classifier predicts a set of subsets of labels rather than a single subset of labels. This approach is particularly valuable in contexts where the traditional classification method lacks confidence in predicting a single subset of labels due to insufficient infor-

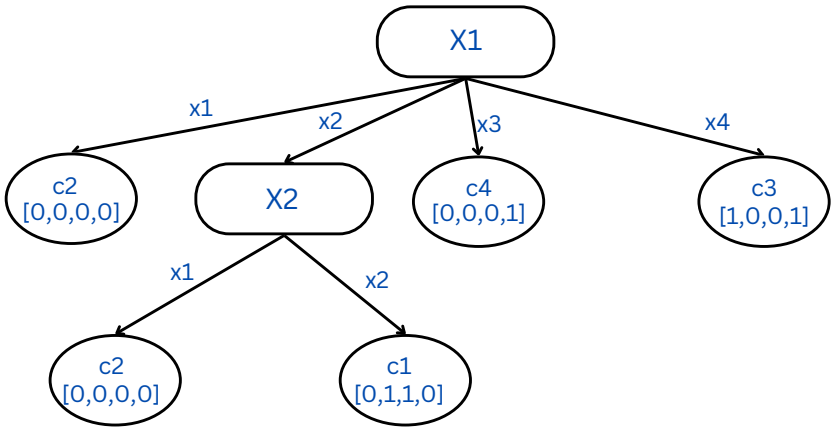


Figure 4.1: Classification tree using the LP method.

Algorithm 4.2 Imprecise classification for MLC using LP method.

Input: Dataset $\mathcal{D} = \{(\mathbf{x}_t, \mathcal{Y}_t) \mid t = 1, 2, \ldots, n\}$
  foreach unique subset $\mathcal{Y}^r \subseteq \mathcal{Y}$ in $\mathcal{D}$, where $r \in \{1, 2, \ldots, R\}$
    Assign a unique class label $c_{\mathcal{Y}^r} = c_r$, $r \in \{1, 2, \ldots, R\}$
    Create a new dataset $\mathcal{D}' = \{(\mathbf{x}_t, c_t) \mid t = 1, 2, \ldots, n\}$, where $c_t \in \{c_1, c_2, \ldots, c_R\}$
    Train a classifier, H, on the transformed dataset $\mathcal{D}'$
  foreach instance in the testing dataset $\mathbf{x}_j$ where $j = 1, 2, \ldots, n_{\text{new}}$
    Apply $\mathbf{x_j}$ in $\mathcal{CT}$ to reach a leaf node
    Get probability intervals for each class label in the terminal node using the NPI-M model: $[l_r, u_r], r \in \{1, 2, \ldots, R\}$
    Apply the strong dominance criterion to the probability intervals from the previous step to get a set of non-dominated labels for $\mathbf{x}$
    Retrieve the corresponding subset of labels $\mathcal{Y}^r$ for each subset in the set of non-dominated labels.
Return the predicted set of subsets of labels

mation.

In MLC problems, certain labels may be rare, making it challenging for a traditional classification method to accurately predict a single subset of labels. Instead, predicting a set of subsets of labels using imprecise classifiers can enable enhancement of the reliability of predictions. This approach provides multiple possible label combinations, enabling the classifier to be cautious in scenarios where predictions are difficult due to limited information.

In this classification method, the LP method is used with imprecise classification algorithms for MLC. These imprecise algorithms assign interval probabilities to each class label at the leaf nodes, rather than assigning the most frequent class label. The criterion used to determine the predicted set of labels is the dominance criterion, as presented in

X_1	X_2	X_3	X_4	X_5	y_1	y_2	y_3	y_4
x_2	x_2	x_2	x_2	x_2	1	0	1	1
x_1	x_1	x_1	x_1	x_1	0	0	0	0
x_4	x_2	x_2	x_4	x_2	1	1	1	1
x_3	x_1	x_3	x_5	x_1	0	0	0	0
x_2	x_1	x_2	x_2	x_1	1	1	1	1
x_4	x_1	x_2	x_1	x_2	0	0	0	0
x_1	x_2	x_1	x_3	x_1	0	0	0	0
x_3	x_2	x_3	x_4	x_2	1	1	1	1
x_2	x_1	x_1	x_5	x_1	0	0	0	0
x_4	x_2	x_2	x_2	x_2	1	1	1	1

Table 4.3: Dataset description for Example 4.2.

X_1	X_2	X_3	X_4	X_5	class
x_2	x_2	x_2	x_2	x_2	c_1
x_1	x_1	x_1	x_1	x_1	c_2
x_4	x_2	x_2	x_4	x_2	c_3
x_3	x_1	x_3	x_5	x_1	c_2
x_2	x_1	x_2	x_2	x_1	c_3
x_4	x_1	x_2	x_1	x_2	c_2
x_1	x_2	x_1	x_3	x_1	c_2
x_3	x_2	x_3	x_4	x_2	c_3
x_2	x_1	x_1	x_5	x_1	c_2
x_4	x_2	x_2	x_2	x_2	c_3

Table 4.4: Transformed dataset using LP method for Example 4.2.

Section 3.2. This method is presented in Algorithm 4.2.

To clarify how this approach works, the following example is provided.

Example 4.2. In this example, the same variables used in Example 4.1 are used. Table 4.3 presents an artificial dataset consisting of 10 patient instances. This dataset is then transformed using the LP method, and the resulting transformed dataset is shown in Table 4.4.

The procedure in Algorithm 4.2 is followed to construct the classification tree. The NPI lower and upper probabilities for CI for each attribute are derived as in the D-NPI classifier, and the best attribute is selected to split the tree. Unlike in Example 4.1, rather than assigning the most frequent subset of labels to the leaf node, according to this method a set of subsets of labels is assigned.

As shown in Figure 4.2, the corresponding classification tree is constructed using the transformed dataset in Table 4.4. In this method, after reaching a leaf node, the strong

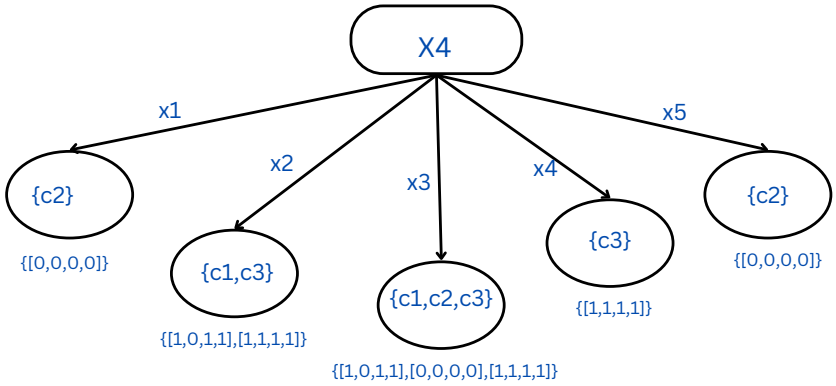


Figure 4.2: Imprecise classification tree using LP method.

dominance criterion is applied, and a set of non-dominated subsets of labels is assigned. This set is then reverse-transformed to the corresponding set of the original subsets of labels. For example, if a new instance has $X_4 = x_2$, the predicted set of labels is $\{c_1, c_3\}$, which corresponds to the original subsets of labels $\{[1, 0, 1, 1], [1, 1, 1, 1]\}$. Instead of assigning a single subset of labels, this approach allows for predicting a set of subsets of labels. This can be reliable in situations where the classifier lacks confidence in assigning a single subset of labels. For instance, this may occur when there is a lack of sufficient information.

4.3 Performance evaluation

In the context of multi-label classification, several measures are used to evaluate the performance of classifiers [116]. Evaluating the performance of MLC classifiers can be relatively difficult; the prediction is a subset of labels, and it can be fully correct, fully wrong, or partially correct [20]. In this chapter, example-based measures are used to assess MLC methods, focusing on the performance per instance (example) and averaged over all instances in testing dataset [61, 63, 99]. These measures include subset accuracy, accuracy, Hamming loss, Precision, Recall, and F1-score. Example-based classification measures provide a comprehensive evaluation by assessing the performance of the classifiers on a per-instance basis, allowing for an evaluation of how well the algorithms predict the subset of labels for each instance. However, future research may consider adopting label-based measures to enable a more detailed assessment of MLC methods across individual labels.

Let n be the total number of instances in the dataset, $\mathcal{Y}'_t$ represent the predicted subset of labels assigned to instance t, and $\mathcal{Y}_t$ be the subset of labels associated with instance t where $t = 1, \ldots, n$. The following measures [61, 63, 99], will be used in Section 4.4.

- **Subset Accuracy**: The proportion of instances for which the predicted subset of labels exactly matches the true subset of labels. It is defined as:

$$\text{Subset Accuracy} = \frac{1}{n}\sum_{t=1}^{n}\mathbb{I}(\mathcal{Y}'_t = \mathcal{Y}_t). \tag{4.1}$$

- **Accuracy**: The average Jaccard similarity coefficient [97] between the predicted subset of labels and the true subset of labels for each instance. For an instance $\mathbf{x}_t$, the Jaccard similarity between $\mathcal{Y}'_t$ and $\mathcal{Y}_t$ is given by:

$$\text{Accuracy} = \frac{1}{n}\sum_{t=1}^{n}\frac{|\mathcal{Y}'_t \cap \mathcal{Y}_t|}{|\mathcal{Y}'_t \cup \mathcal{Y}_t|}. \tag{4.2}$$

- **Hamming Loss**: The proportion of incorrect labels, on average, across all instances. It is defined as:

$$\text{Hamming Loss} = \frac{1}{n}\sum_{t=1}^{n}\frac{|\mathcal{Y}'_t \Delta \mathcal{Y}_t|}{Q}, \tag{4.3}$$

  where Δ denotes the symmetric difference between the predicted subset of labels and the true subset of labels, representing the labels that are in either of the subsets but not in their intersection.

- **Precision**: The average proportion of the relevant labels in the predicted set that are really relevant for the instances:

$$\text{Precision} = \frac{1}{n}\sum_{t=1}^{n}\frac{|\mathcal{Y}'_t \cap \mathcal{Y}_t|}{\mathcal{Y}'_t}. \tag{4.4}$$

- **Recall**: The average proportion of relevant labels for the instances which are predicted as relevant:

$$\text{Recall} = \frac{1}{n}\sum_{t=1}^{n}\frac{|\mathcal{Y}'_t \cap \mathcal{Y}_t|}{\mathcal{Y}_t}. \tag{4.5}$$

- **F1-score**: The average harmonic mean between Precision and Recall. It is given by:

$$\text{F1-score} = \frac{1}{n}\sum_{t=1}^{n}\frac{2\times|\mathcal{Y}'_t\cap\mathcal{Y}_t|}{|\mathcal{Y}'_t|+|\mathcal{Y}_t|}. \tag{4.6}$$

These measures evaluate how well the predicted subset of labels matches the correct subset of labels for each instance and have been widely used in MLC tasks (see, e.g. Madjarov et al. [80], Moral-García and Abellán [82], and Moral-García et al. [83]). In this chapter, they are applied to the testing dataset to assess and compare classifier performance.

For imprecise classification with MLC, the measures from Section 3.4 are used, including Determinacy (DET), Single Accuracy (SingleA), Set Accuracy (SetA), Indeterminacy Size (IS), DACC, and MIC.

4.4 Experiments

This section describes the experiments conducted to evaluate the performance of the algorithms for MLC using the LP method. These experiments are structured into two main sections based on the methods used for classification. The first section focuses on studying the performance of classifiers which predict a single subset labels, while the second section presents the experiment using imprecise classification algorithms for MLC.

In this section, six datasets from different applications, each with distinct characteristics, are used. These datasets can be accessed and downloaded from the '`mldr.datasets`' package in R [25]. The datasets vary in terms of the degree of multi-label nature. In order to measure the degree of multi-label nature, two metrics are used: label cardinality (l_c) and label density (l_d). The label cardinality metric is defined as the average number of labels associated with each instance in the dataset. The label density represents the average proportion of labels associated with each instance, divided by the total number of possible labels in the dataset. Table 4.5 summarises the datasets where "n" represents the size of the dataset, "Num" is the number of numerical attributes, "Nom" is the number of nominal attributes, "n_l" is the number of labels, "l_c" is the label cardinality, "l_d" is the label density.

Prior to the experiment, the datasets are subjected to a preprocessing phase. Un-

Dataset	n	Num	Nom	n_l	l_c	l_d	Domain
Bibtex	7395	-	1836	159	2.4	0.015	Text
Corel5k	5000	-	499	374	3.52	0.009	Multimedia
Enron	1702	-	1001	53	3.38	0.064	Text
Genbase	662	-	1186	27	1.252	0.046	Biology
Medical	978	-	1449	45	1.24	0.028	Text
Yeast	2417	103	-	14	4.24	0.303	Biology

Table 4.5: Datasets. Size "n", number of numerical attributes "Num", number of nominal attributes "Nom", number of labels "n_l", label cardinality "l_c", and label density "l_d".

labelled instances and attributes with constant values are removed, as they provide no useful information for learning. Additionally, continuous attributes in the Yeast dataset are discretised by applying the equal frequency method from the package '`arules`' in R by using the '`discretize`' function [66]. To reduce noise in the datasets from extremely rare labels, labels with fewer than 5 occurrences in the datasets are removed. In order to assess the performance of algorithms and compare their performances, a 5-fold cross-validation procedure is used due to the computational cost.

In this study, the primary aim is to evaluate how well the classifiers perform for MLC problems. The secondary aim is to compare the performance of the classifiers.

The algorithms used in these experiments, either for predicting a single subset of labels or a set of subsets of labels, are the D-NPI algorithm [12], the NPI-M algorithm [6], the A-NPI-M algorithm [6], the IDM1 algorithm with the hyperparameter $s = 1$ [1], and the IDM2 algorithm with $s = 2$ [1]. For the algorithms NPI-M, A-NPI-M, IDM1, and IDM2, the '`imptree`' package in R is used [56]. For ease of reference, multi-label classification using the LP method (as detailed in Section 4.2.1) is referred to as Method 1 throughout this section. Similarly, imprecise multi-label classification using the LP method (as explained in Section 4.2.2) is referred to as Method 2.

4.4.1 Predicting a single subset of labels (Method 1)

This section presents a study of the performance of the classifiers using the LP method for MLC, using the procedure in Algorithm 4.2 in Section 4.2. The performances of the classifiers are evaluated and compared using the metrics: subset accuracy, accuracy, Hamming loss, Precision, Recall and F1, which are introduced in Section 4.3. While performing 5-fold cross-validation, the metrics are computed for each fold, and the average

Dataset	D-NPI	NPI-M	A-NPI-M	IDM1	IDM2
Bibtex	0.1501	0.1562	**0.1571**	0.0719	0.0788
Corel5k	0.0206	0.0234	**0.0236**	0.0148	0.0144
Enron	0.1175	**0.1392**	0.1381	0.1281	0.1293
Genbase	0.9879	**0.9880**	**0.9880**	0.9864	0.9819
Medical	0.6869	**0.7004**	0.6973	0.6859	0.6818
Yeast	0.1192	0.1556	**0.1568**	0.1254	0.1332

Table 4.6: Subset accuracy for all classifiers.

values are reported to summarise the results.

The results of the five classification algorithms across the metrics are presented as follows: subset accuracy (Table 4.6), accuracy (Table 4.7), Hamming loss (Table 4.8), Precision (Table 4.9), Recall (Table 4.10) and F1-score (Table 4.11). In this section, the analysis of the experimental results is presented, showing the strengths and limitations of each algorithm within the context of the evaluated datasets.

The Bibtex dataset [74] contains data about Bibtex, where each instance represents a BibTeX item. The Bibtex dataset is characterised by high dimensionality, indicated by a large number of instances, attributes and labels, and exhibits label sparsity, as reflected by a low value of l_d. These characteristics lead to low values of subset accuracy across all algorithms. In Table 4.6, the best subset accuracy is achieved by the A-NPI-M algorithm. The IDM1 and IDM2 algorithms have low values compared to the first three algorithms, with the IDM1 algorithm achieving the lowest value of subset accuracy. The relatively low values for algorithms IDM1 and IDM2 reflect their lower ability to handle the dataset's complexity compared to the other algorithms.

The Corel5k dataset [50] contains 5000 Corel images, each annotated with 4-5 keywords. Due to the large number of labels and the sparsity of labels for this dataset, the subset accuracy values for all algorithms are the lowest among the other datasets, which struggle to return correct subset of labels for this dataset. The subset accuracy values for the NPI-M and A-NPI-M algorithms are the highest, with A-NPI-M being the best, followed by the D-NPI algorithm. Algorithms IDM1 and IDM2 achieve the lowest values of subset accuracy. The classifiers D-NPI, NPI-M, and A-NPI-M, which are based on the NPI approach tend to perform similarly for this dataset.

The Enron dataset [77] contains data on the emails of Enron seniors, featuring a large number of attributes, which makes it a challenging dataset for classification. The low

density of labels leads to high sparsity. The NPI-M and A-NPI-M algorithms achieve the highest subset accuracy values, followed closely by algorithms IDM2 and IDM1. The D-NPI algorithm is less effective compared to the other algorithms.

The Genbase dataset [49], which contains protein data, is particularly interesting for internal investigation, as all algorithms achieve the highest subset accuracy values for it compared to the other datasets. This dataset has high dimensional attribute with low labels density, however, it includes a large number of attributes with a constant value. During preprocessing, these attributes are removed because they are not informative, changing the number of attributes from 1186 to 112. This results in the dataset having a reduced dimensional attribute. The low total number of labels and low value of cardinality increases the chance of predicting the exact subset of labels for all algorithms. The D-NPI, NPI-M and A-NPI-M algorithms have similar performance, while IDM1 and IDM2 follow with values of 0.9864 and 0.9819, respectively.

The medical dataset [91] contains documents, each document has a patient symptom history. The subset accuracy values across classifiers show minimal variation for the this dataset, resulting in similar performance. The NPI-M algorithm achieves the highest value, whereas the IDM2 algorithm has the lowest value. The Yeast dataset [51] contains data about functions of genes. This dataset has a high cardinality of labels, which reduces the chance of accurately predicting the exact subset of labels, as demonstrated by the results in Table 4.6. Algorithms NPI-M and A-NPI-M achieve the highest subset accuracy values among all others, while both algorithms IDM1 and IDM2 achieve the lowest values.

Upon further investigations, the D-NPI algorithm tends to produce deep trees for many of the datasets, possibly learning complex patterns, but this can lead to overfitting, affecting its generalisation performance. In contrast, the IDM1 and IDM2 algorithms produce shallow trees for most of the datasets, possibly reducing overfitting. However, they may struggle to capture complex patterns, as observed in the case of the Bibtex dataset. Meanwhile, the NPI-M and A-NPI-M algorithms tend to produce classification trees with moderate depth compared to other algorithms across a variety of datasets, offering a good balance between learning meaningful patterns and avoiding overfitting. This can explain the best performance observed for the NPI-M and A-NPI-M algorithms across the subset accuracy measure.

Regarding the accuracy measure, since it is more forgiving of incorrect labels than

Dataset	D-NPI	NPI-M	A-NPI-M	IDM1	IDM2
Bibtex	**0.2632**	0.2608	0.2614	0.1391	0.1450
Corel5k	**0.0966**	0.0883	0.0929	0.0725	0.0714
Enron	0.3616	0.3774	**0.3787**	0.3640	0.3658
Genbase	0.9919	0.9922	0.9922	**0.9925**	0.9907
Medical	**0.7559**	0.7548	0.7532	0.7467	0.7450
Yeast	0.3945	**0.4559**	0.4564	0.4043	0.4173

Table 4.7: Accuracy for all classifiers.

the subset accuracy measure, its values are generally higher than those of the subset accuracy measure across the classifiers and the datasets. This is expected, as subset accuracy is more strict than accuracy measure in penalising incorrect labels. Table 4.7 shows that the D-NPI algorithm achieves the highest accuracy values for the Bibtex, Corel5k, and Medical datasets. Notably, these three datasets have the lowest label density values among the other datasets. For the Bibtex and Corel5k datasets, the combination of low label density values with a large number of labels leads to a large number of combinations when using the LP method. This, in turn, results in reduced performance across classifiers compared to the other datasets. For these datasets, the NPI-M and A-NPI-M algorithms achieve the next-highest performance, with A-NPI-M outperforming NPI-M for the Bibtex and Corel5k datasets. Meanwhile, IDM1 and IDM2 show the lowest values of accuracy for these datasets, with IDM2 outperforming IDM1 for the Bibtex and Medical datasets.

For the Enron dataset, the D-NPI classifier achieves an accuracy value that is three times higher than that of the subset accuracy measure. This highlights its ability to correctly identify relevant labels among its predicted subset of labels. The A-NPI-M algorithm has the highest accuracy value, followed by the NPI-M algorithm. The D-NPI, IDM1, and IDM2 perform similarly in terms of the accuracy measure for the Enron dataset. For the Genbase dataset, the classifiers tend to have similarly high performance, indicating that this dataset has informative attributes with simple pattern for classifiers to learn. The NPI-M classifier has the highest accuracy value for the Yeast dataset. However, the D-NPI, IDM1 and IDM2 classifiers have high accuracy values compared to their subset accuracy values, indicating their notable ability to predict relevant labels in their subset of labels.

In Table 4.8, according to the Hamming loss, all classifiers achieve their lowest values

Dataset	D-NPI	NPI-M	A-NPI-M	IDM1	IDM2
Bibtex	0.0202	0.0187	**0.0186**	0.0207	0.0202
Corel5k	0.0211	**0.0209**	0.0212	0.0220	0.0220
Enron	0.0735	0.0704	0.0702	0.0688	**0.0680**
Genbase	0.0012	0.0014	0.0014	**0.0011**	0.0014
Medical	**0.0214**	**0.0214**	0.0215	0.0227	0.0226
Yeast	0.2840	0.2485	0.2482	0.2722	**0.2666**

Table 4.8: Hamming Loss for all classifiers.

Dataset	D-NPI	NPI-M	A-NPI-M	IDM1	IDM2
Bibtex	0.3480	0.3551	**0.3572**	0.2004	0.2093
Corel5k	**0.1361**	0.1262	0.1315	0.1041	0.1024
Enron	0.4927	0.5113	**0.5118**	0.5064	0.5047
Genbase	**0.9950**	0.9945	0.9945	0.9940	0.9944
Medical	**0.7969**	0.7951	0.7941	0.7820	0.7846
Yeast	0.5304	0.5951	**0.5958**	0.5505	0.5605

Table 4.9: Precision for all classifiers.

for the Genbase dataset. As this dataset represents proteins, each protein can belong to multiple function label. These labels can co-occur frequently, leading to repeated patterns across the proteins. Using the LP method with these repeated patterns of labels results in a limited combinations of labels. For this dataset, using informative attributes along with the small number of label combinations leads to the high performance and low loss values.

All classifiers have low values of the Hamming loss measure for the Bibtex, Corel5k, and Medical datasets. As these datasets have the lowest values of label density, the possibility of incorrect labels in the predicted subset of labels is reduced. For example, if an instance is associated with only two out of 159 possible labels, and the predicted subset of labels includes three relevant labels, the loss remains minimal due to the large total number of labels. In contrast, the Yeast dataset has a moderate value of label density, which increases the possibility of incorrect labels in the subset of labels. This results in higher values of Hamming loss across classifiers compared to the other datasets.

As shown in Table 4.8, the A-NPI-M algorithm has the lowest misclassification rate among these algorithms for the Bibtex dataset. For the Enron dataset, the IDM2 algorithm has the lowest Hamming loss value. The D-NPI and NPI-M algorithms achieve an identical lowest Hamming loss for the Medical dataset.

Dataset	D-NPI	NPI-M	A-NPI-M	IDM1	IDM2
Bibtex	**0.3202**	0.2999	0.2991	0.1665	0.1691
Corel5k	**0.1366**	0.1194	0.1285	0.1047	0.1040
Enron	0.4712	0.4747	**0.4767**	0.4500	0.4510
Genbase	0.9925	0.9932	0.9932	**0.9940**	0.9918
Medical	**0.7839**	0.7699	0.7694	0.7728	0.7685
Yeast	0.5176	0.5621	**0.5626**	0.5184	0.5321

Table 4.10: Recall for all classifiers.

Dataset	D-NPI	NPI-M	A-NPI-M	IDM1	IDM2
Bibtex	**0.3132**	0.3067	0.3073	0.1712	0.1769
Corel5k	**0.1338**	0.1207	0.1278	0.1028	0.1015
Enron	0.4616	0.4753	**0.4765**	0.4599	0.4605
Genbase	0.9932	0.9933	0.9933	**0.9936**	0.9925
Medical	**0.7791**	0.7732	0.7722	0.7673	0.7661
Yeast	0.4980	0.5627	0.5632	0.5085	0.5223

Table 4.11: F1-score for all classifiers.

For the Bibtex dataset, Tables 4.9, 4.10, and 4.11 show that algorithm D-NPI has a reasonable Precision value, the highest Recall value, and the highest F1-score value. The results indicate that algorithm D-NPI includes more incorrect labels in the predicted subset of labels compared to the A-NPI-M and NPI-M algorithms, which achieve the highest Precision value. Algorithms IDM1 and IDM2 have the lowest values for these three measures for this dataset. The same pattern of results is observed for the medical dataset. For these two datasets, the IDM1 and IDM2 classifiers tend to produce shallow trees. In high-dimensional datasets, this may lead to a failure to capture complex relationships between attributes and labels, limiting their predictive performance.

The D-NPI algorithm has the highest values for Precision, Recall and F1-score for the Corel5k dataset, followed by algorithms A-NPI-M and NPI-M, respectively. Whereas, algorithms IDM1 and IDM2 have the lowest performance on these measures compared to the other algorithms. The Bibtex, Corel5k, and Medical dataset have lower label density value compared to the other datasets. The algorithms which are based on NPI approach tend to perform better than those based on the IDM for these datasets, highlighting the effectiveness of using NPI in such data characteristics.

While algorithm D-NPI has the best Precision value for the Genbase dataset, algorithm IDM1 has the highest Recall and F1-score values, achieving a good tradeoff

between Precision and Recall measures. Algorithms NPI-M and A-NPI-M follow closely behind, with algorithm D-NPI falling behind in terms of Recall and F1-score measures. All classifiers tend to perform similarly for this dataset across all performance measures.

Algorithms NPI-M and A-NPI-M perform similarly for the Enron dataset across the three measures presented in Tables 4.9, 4.10 and 4.11, achieving better results than the other algorithms. The D-NPI achieves the lowest Precision value, but offers a good tradeoff between Precision and Recall, as reflected by its F1-score.

The next section presents the use of imprecise classifiers for MLC problems, which results in a predicted set of subsets of labels rather than a single subset of labels.

4.4.2 Predicting a set of subsets of labels (Method 2)

This section presents a detailed comparative analysis of the imprecise classifiers over eight datasets using the metrics: Determinacy (DET), Single Accuracy (SingleA), Set Accuracy (SetA), Indeterminacy Size (IS), DACC, and MIC, as presented in Section 3.4. The imprecise classifier D-NPI-IC1, introduced in Chapter 3, is used in this experiment, following the steps outlined in Algorithm 4.2, in order to apply the LP method. The algorithms NPI-M, A-NPI-M, IDM1, and IDM2 are implemented after transforming the datasets using the LP method, by using the '`imptree`' package in R [56]. The strong dominance criterion is used at a leaf node when assigning a new instance. In this section, the algorithms are renamed to better align with the concept of imprecise classification and to ensure distinction from those used in the experiment in Section 4.4.1. The names incorporate the term Imprecise Classification (IC) to reflect their alignment with imprecise classification concepts. The algorithm names used in this experiment are D-NPI-IC, NPI-M-IC, A-NPI-M-IC, IDM1-IC, and IDM2-IC.

Tables 4.12 to 4.17 provide a summary of the performance of all algorithms across various datasets, evaluated using the eight different metrics.

Regarding the Bibtex dataset, A-NPI-M-IC has the highest determinacy value, but its SingleA value is the lowest among the first three algorithms. In contrast, D-NPI-IC, with a moderate value of DET, achieves the highest SingleA and SetA values among all algorithms. The primary reason that D-NPI-IC exhibits a notably higher SetA value is its comparatively high IS value, as observed in Table 4.15, relative to the other algorithms. This can be caused by its tendency to split further, resulting in deeper trees which

Dataset	D-NPI-IC	NPI-M-IC	A-NPI-M-IC	IDM1-IC	IDM2-IC
Bibtex	0.3992	0.4445	**0.4695**	0.2366	0.1953
Corel5k	0.1833	**0.4831**	0.2860	0.1109	0.0522
Enron	0.3478	**0.4900**	0.4571	0.3137	0.2720
Genbase	**1.0000**	0.9819	0.9819	0.9879	0.9894
Medical	**0.9053**	0.8785	0.8775	0.8836	0.8269
Yeast	0.4857	**0.8188**	0.8167	0.3947	0.3033

Table 4.12: Determinacy (DET) for all classifiers.

leads to more non-dominated subsets of labels, particularly with many class labels in the transformed dataset results after using the LP method. The first three algorithms have moderate confidence in predicting a single subset of labels. In contrast, the algorithms based on the IDM appear to be less deterministic. Consequently, they have the lowest values for both SingleA and SetA. For this dataset, the high number of labels with low value of label density leads to a large number of possible combinations when using the LP method. When the number of combinations increases co-occurrence is rare, the classifiers tend to lack confidence in returning a single subset of labels, often producing instead a set of subsets of labels. In terms of DACC and MIC measures in Tables 4.16 and 4.17, A-NPI-M-IC achieves the highest values for both, followed by NPI-M-IC and D-NPI-IC, respectively. The results indicate that it offers a good tradeoff between achieving correct predictions and maintaining an appropriate size of the predicted set, compared to the other algorithms. The low values of DACC and MIC for the IDM1-IC and IDM2-IC algorithms are due to their low values of SingleA and SetA compared to the other algorithms.

When analysing the results of the Corel5k dataset, all algorithms show variability in their DET values. For this dataset, the NPI-M-IC algorithm tends to terminate tree-building process earlier than the other classifiers. This early termination can result in higher confidence in predicting a single subset of labels compared to the other algorithms. However, terminating the tree construction early increases the possibility of having more observations in the class variables, which may cause one class label to dominate the others. The NPI-M-IC algorithm fails to achieve the highest SingleA value; rather, it records the lowest among all algorithms. However, it achieves the highest values for DACC and MIC. The results indicate that it offers a good balance in achieving correct predictions along with a good size of the predicted set compared to the other algorithms.

Dataset	D-NPI-IC	NPI-M-IC	A-NPI-M-IC	IDM1-IC	IDM2-IC
Bibtex	**0.3409**	0.3204	0.3058	0.2043	0.2500
Corel5k	**0.0776**	0.0427	0.0633	0.0475	0.0701
Enron	0.2908	0.2603	0.2750	0.3189	**0.3743**
Genbase	0.9879	**0.9923**	**0.9923**	0.9863	0.9893
Medical	0.7392	**0.7624**	0.7598	0.7482	0.7572
Yeast	0.2223	0.16721	0.1677	0.2373	**0.2896**

Table 4.13: Single Accuracy (SingleA) for all classifiers.

Dataset	D-NPI-IC	NPI-M-IC	A-NPI-M-IC	IDM1-IC	IDM2-IC
Bibtex	**0.9390**	0.8288	0.8566	0.5948	0.7497
Corel5k	**0.8895**	0.8214	0.8688	0.8030	0.8701
Enron	**0.9370**	0.9136	0.8962	0.6394	0.7740
Genbase	-	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Medical	**0.9241**	0.9075	0.9075	0.8688	0.7563
Yeast	**0.8812**	0.4589	0.4325	0.4530	0.6103

Table 4.14: Set Accuracy (SetA) for all classifiers.

The D-NPI-IC algorithm achieves low DET value and the highest SingleA and SetA values. Its high value of IS impacts the DACC and MIC values, leading to low values for both measures. Algorithm IDM2-IC has the weakest performance among all algorithms regarding the DACC and MIC measures. This is primarily due to its limited ability to produce determinant predictions compared to the other algorithms, which observed in its low value of DET. In terms of the DACC and MIC measures, the results of the first three algorithms compared to the IDM1-IC and IDM2-IC algorithms suggest that the algorithms based on the NPI approach have better performance compared to those based on the IDM for this dataset.

For the Enron dataset, the classifiers have the same pattern as for the Corel5k dataset in terms of their DET values. The NPI-M-IC and A-NPI-M-IC algorithms achieve the highest values for DACC and MIC. The D-NPI-IC algorithm has high values for SetA and IS which affect its low values for DACC and MIC compared to the other algorithms. Algorithms IDM1-IC and IDM2-IC achieve the lowest score for the SetA metric, which can be caused by their notable low values for IS.

Considering the Genbase dataset, the D-NPI-IC algorithm is 100% deterministic with a reasonably high value for SingleA and the highest DACC and MIC values compared to all algorithms. Note that, its SingleA value is the same as the subset accuracy value

Dataset	D-NPI-IC	NPI-M-IC	A-NPI-M-IC	IDM1-IC	IDM2-IC
Bibtex	2667	2276	2354	**1588**	2026
Corel5k	2738	2502	2662	**2445**	2661
Enron	693	668	652	**440**	540
Genbase	-	23	23	18	**4**
Medical	57	55	55	44	**30**
Yeast	173	58	54	77	102

Table 4.15: Indeterminacy Size (IS) for all classifiers.

Dataset	D-NPI-IC	NPI-M-IC	A-NPI-M-IC	IDM1-IC	IDM2-IC
Bibtex	0.1379	0.1482	**0.1495**	0.0565	0.0565
Corel5k	0.0160	**0.0209**	0.0197	0.0071	0.0057
Enron	0.1040	**0.1319**	0.1314	0.1087	0.1106
Genbase	**0.9879**	0.9751	0.9751	0.9762	0.9835
Medical	0.6747	**0.6776**	0.6745	0.6745	0.6526
Yeast	0.1123	**0.1507**	0.1505	0.1094	0.1079

Table 4.16: DACC for all classifiers.

when using the D-NPI algorithm in Section 4.4.1. The NPI-M-IC and A-NPI-M-IC algorithms have identical performance across all metrics for this dataset. The IDM2-IC algorithm has good performance across the metrics, achieving the second-highest values of DACC and MIC among all classifiers. Interestingly, the IS value for this classifier is much lower compared to the other algorithms, which could be due to its tendency to produce shallow trees compared to other algorithms. As more data becomes available, the classifier becomes more confident about the correct prediction, resulting in its low value of IS (small size of the predicted set). Due to the low value of IS, this algorithm achieves a good overall performance in terms of DACC and MIC. All classifiers have good performances for this dataset.

Based on the Medical dataset, the results indicate that NPI-M-IC and A-NPI-M-IC have similar performance, with slight variations across some measures. For the IDM2-IC algorithm, the IS value is very low, affecting its SetA low value, which results in the lowest values of both DACC and MIC for this dataset. The D-NPI-IC and IDM1-IC algorithms perform at a level close to that of the NPI-M-IC and A-NPI-M-IC algorithms in terms of DACC and MIC. The D-NPI-IC tends to be the most deterministic classifier among all classifiers for this dataset.

Finally, concerning the Yeast dataset, NPI-M-IC has the best performance regarding

Dataset	D-NPI-IC	NPI-M-IC	A-NPI-M-IC	IDM1-IC	IDM2-IC
Bibtex	1.1070	1.2494	**1.2579**	0.5817	0.5939
Corel5k	0.1384	**0.1941**	0.1833	0.1085	0.0854
Enron	0.7015	0.9028	**0.9079**	0.8432	0.8581
Genbase	**3.0993**	3.0560	3.0560	3.0643	3.0928
Medical	2.8691	2.8859	2.8730	**2.8989**	2.8537
Yeast	0.6018	**0.8830**	0.8813	0.6984	0.7499

Table 4.17: MIC for all classifiers.

the DACC and MIC measures. This dataset has considerable variation across algorithms. Algorithms NPI-M-IC and A-NPI-M-IC are almost twice as deterministic as the other classifiers with lower values for SingleA and SetA. However, they have the lowest values of IS, which helps maintain the balance observed in the DACC and MIC values. The D-NPI-IC classifier achieves a high DACC value but a relatively low MIC value compared to the IDM1-IC and IDM2-IC algorithms. This can be caused by producing approximately 49% of determinant predictions, with a relatively high value of SingleA and the highest value of SetA compared to the other classifiers. This can lead to a slight increase in the DACC value compared to the IDM1-IC and IDM2-IC. However, its high value of IS affects the decreased value of MIC. To explain, MIC assigns values to all predictions, correct and incorrect, depending on their correctness and informativeness. It assigns a lower value in case of the prediction is correct with a high value of set cardinality (less informative), which in some cases less than the value assigned to wrong prediction.

4.5 Conclusions

In this chapter, the D-NPI classification method was extended to multi-label classification (MLC) using the Label Powerset (LP) approach. In addition, four other algorithms were used in this study: the NPI-M and A-NPI-M algorithms, as well as the two algorithms based on the IDM with different hyperparameters values (i.e., $s = 1, 2$).

This study considered two methods for addressing MLC problems using the LP approach. The first method focused on predicting a single subset of labels, employing algorithms originally developed for classical classification problems. The second method applied imprecise classifiers to obtain a predicted set of subsets of labels.

Experimental studies were conducted to evaluate the performance of the classifiers for

the two methods. Different evaluation measures were used to assess the performance of the algorithms. For the first method, example-based measures were used to evaluate the performance of the classifiers on a per-instance bases. For the second method, imprecise classification measures were used. This study included six datasets, each characterised by distinct properties, including size, attribute types, label cardinality, and label density. The use of label-based classification measures and ranking-based measures for the first method is a topic for future investigation.

The results of the first method varied across the classifiers according to the characteristics of the datasets. All classifiers tend to perform well for the Genbase and Medical datasets compared to the others. For the Genbase dataset, labels appeared to be correlated and frequently co-occurred, which enabled the classifiers to perform more effectively. Although all classifiers performed well for this dataset, the IDM2 algorithm had slightly lower performance. For the Medical dataset, classifiers based on the NPI approach slightly performed better than those based on the IDM.

The datasets from the text domain, such as Bibtex, Enron, and Medical, the D-NPI, NPI-M, and A-NPI-M classifiers generally performed better than the IDM1 and IDM2 classifiers across most evaluation measures. The Corel5k dataset, which contains a high number of labels and low label density value, caused challenges for all algorithms, resulting in poor performance. Applying the LP method under these conditions likely led to a high number of label combinations, making it difficult for classifiers to produce meaningful patterns. However, classifiers based on the NPI approach achieved slightly better performance than those based on the IDM; specifically, the D-NPI algorithm achieved the best results across most evaluation measures. This could be due to its tendency to produce deep trees which can capture complex patterns.

The NPI-M and A-NPI-M algorithms performed similarly across various datasets, and the same was observed for the IDM1 and IDM2 for some datasets. The D-NPI algorithm showed good performance with datasets that have low label density values.

Transforming the datasets from a multi-label classification problem to a single-label classification problem using the LP approach introduces challenges due to the large number of class labels in the transformed datasets and the potential for high imbalance. Exploring alternative methods for MLC problem, including adaptation of D-NPI to handle multiple labels directly, remains an important topic for future research.

In the second method, all the algorithms are used for imprecise classification after transforming the datasets. For prediction, after reaching a leaf node, a strong dominance criterion is applied and instead of assigning a single subset of labels to a new instance, a set of subsets of labels is assigned to it. The imprecise algorithms were called D-NPI-IC, NPI-M-IC, A-NPI-M-IC, IDM1-IC and IDM2-IC in this chapter.

The performance of the algorithms varied across the datasets and the evaluation metrics. The results showed that Determinacy (DET) and single accuracy (SingleA) were generally inconsistent, indicating that the classifiers' ability to confidently predict a single subset of labels depended on the dataset. For the Genbase and Medical datasets, as discussed for the first method, appeared to provide favourable characteristics for the classifiers to perform effectively. Similarly, in this method, the classifiers tended to be more deterministic than for the other datasets, achieving high accuracy among predictions of a single subset of labels.

Set accuracy was often high, leading to the classifiers' capability to include the correct class label within the predicted set of subsets of labels. The D-NPI-IC algorithm tended to produce the highest SetA values due to its high values of Indeterminacy size (IS). This observation led to decreasing the informativeness of this classifier compared to the others for certain datasets. While DACC and MIC were efficient for imprecise classification problems, providing tradeoff between imprecision and accuracy. The NPI-M-IC and A-NPI-M-IC algorithms tended to achieve the highest DACC and MIC values across most datasets. This study revealed the importance of dataset characteristics in determining classifier suitability. Certain datasets, such as the Genbase dataset, had scenarios where the algorithms performed exceptionally well, with near-perfect determinacy and accuracy. Conversely, the Corel5k dataset is challenging due to its label density value being low, combined with a high number of labels.

Further investigation is required to determine the efficiency of imprecise classifiers when being used for MLC problems. In some cases, due to the large number of labels obtained after transforming the data using the LP method, the size of the indeterminate predicted set of subsets of labels is large, which directly impacts the DACC and MIC values. Exploring this point in future research can provide valuable findings. Investigating a criterion for selecting the most relevant labels from among the predicted set of subsets of labels presents an interesting direction for future research.

Overall, classifiers based on the NPI approach outperformed those based on the IDM across most datasets for both methods. Specifically, the classifiers based on the NPI-M and A-NPI-M. The D-NPI classifier tended to produce deep trees, which can be beneficial for some datasets, helping in capturing their complex patterns. However, in the second method, after transforming the datasets, producing deep trees with a large number of class labels in the transformed dataset can result in less informative indeterminate predictions. Implementing an early stopping criterion or applying a post-pruning approach for MLC problems remains a topic for future investigation.

Chapter 5

Performance evaluation of NPI with bivariate copula

5.1 Introduction

Evaluating the performance of statistical inference methods, including prediction, parameter estimation, or hypothesis tests, is important to understand their effectiveness. After introducing a new method, assessing the performance using evaluation measures is required to measure the efficiency of the method. In statistics, several methods are used for performance evaluation, either for evaluating a new method without any comparisons to other methods or to compare its performance to different methods.

Investigating the performance of Nonparametric Predictive Inference (NPI) methods is important. NPI methods have been widely applied in various areas of statistics, making it important to assess their performance both individually and in comparison with other methods. The nature of NPI results differs from that of standard statistical methods, as they primarily consist of imprecise probabilities. Consequently, evaluating these methods while taking imprecision into account can be challenging.

The aim of this chapter is to introduce the use of performance evaluation measures to assess the performance of NPI-based methods in terms of both accuracy and precision. In this context, accuracy refers to the ability of a method to correctly or approximately include the target value within its predictions, while precision refers to the narrowness of the prediction intervals.

Modelling dependences between random quantities is important in statistics. Across

various application areas, it is important to understand relationships between variables by studying the dependence structure. Copulas are widely used in finance [26, 79], in hydrology [59], and in reliability analysis [103]. Sklar's Theorem [102] is a theoretical foundation for copula-based modelling. It establishes a connection between multivariate cumulative distribution functions and their marginal distributions functions through a copula function. It states that for any multivariate cumulative distribution function (CDF) $G(y_1, y_2, \ldots, y_d)$, with marginal CDFs $F_1(y_1), F_2(y_2), \ldots, F_d(y_d)$, a copula function $C : [0,1]^d \to [0,1]$ exists such that

$$G(y_1, y_2, ..., y_d) = C(F_1(y_1), F_2(y_2), \ldots, F_d(y_d)). \tag{5.1}$$

This theorem allows modelling the dependence structure separately from the marginals.

There are several parametric copula families, each differing in terms of dependance structure, characteristics and applications, including Gaussian (Normal) [87], Frank [57], Clayton [28], and Gumbel [64]. An overview of various families of bivariate copulas has been provided by Nelson [87]. In the following, some of the famous families of bivariate copulas are presented for the aim to be used later in this chapter.

One of the best-known copulas is the Normal copula, which can be used to model linear dependencies, particularly when the dependencies are symmetric. It is defined as

$$C_\theta(y_1, y_2) = \Phi_\theta(\Phi^{-1}(y_1), \Phi^{-1}(y_2)), \tag{5.2}$$

where Φ is the CDF of the standard normal distribution, and Φ_θ is the CDF of the standard bivariate normal with correlation parameter θ. It captures the dependence between variables, but it is asymptotically independent in both the lower and upper tails, so it does not show tail dependence [87].

Another important family of copulas is the Frank copula, which can be used to model symmetric dependence between variables [57]. It is characterised by a parameter θ, which controls the strength of the dependence. The Frank copula is defined as:

$$C_\theta(y_1, y_2) = -\frac{1}{\theta} \ln \left(1 + \frac{(e^{-\theta y_1} - 1)(e^{-\theta y_2} - 1)}{e^{-\theta} - 1} \right), \tag{5.3}$$

where $\theta \in (-\infty, \infty) \setminus \{0\}$ is the copula parameter.

A further family of copulas is the Clayton copula, particularly useful for modelling lower tail dependence [28]. This means that it is suitable for cases where extreme values are more likely to occur together in the lower range of the variables. It is defined as:

$$C_\theta(y_1, y_2) = \left[\max\left({y_1}^{-\theta} + {y_2}^{-\theta} - 1, 0\right)\right]^{-1/\theta}, \tag{5.4}$$

where $\theta \in (-1, \infty) \setminus \{0\}$ is the copula parameter which controls the strength of dependence, particularly in the lower tail.

The copula's parameter θ has a relationship with the measure of association that is Kendall's tau, τ [87]. The relationship between the copula's parameter θ and τ has a distinct formula for each of the previously discussed parametric copula families, see [29]. For the Normal copula, $\tau = \frac{2}{\pi}\arcsin(\theta)$; for the Frank copula, $\tau = 1 - \frac{4}{\theta}[1 - \frac{1}{\theta}\int_0^\theta \frac{x}{e^x - 1}dx]$; and for the Clayton copula, $\tau = \frac{\theta}{\theta+2}$.

Coolen-Maturi et al. [29] introduced a semi-parametric predictive method by combining NPI with bivariate parametric copulas. This method uses NPI for the marginals, while the dependence structure between random quantities is estimated using a bivariate parametric copula. The performance of this method was previously assessed by using a simulation study [29, 85]. This method was introduced for prediction of a future bivariate random quantity, providing NPI lower and upper probabilities for an event of interest. The method was assessed via a simulation study using frequentist comparisons. This evaluation process resulted in an interval of lower and upper bounds of proportions, which corresponds to a quantile level. The findings of that study raise an important question regarding the performance evaluation: is it always indicative of poor performance when the quantile level lies outside the interval? And if the interval does not contain the quantile level, how can the performance be evaluated in that case? This chapter presents an investigation into these questions by introducing a new use of performance evaluation measures, and is organised as follows: Section 5.2 presents an outline of the semi-parametric predictive method by Coolen-Maturi et al. [29]. Section 5.3 describes a new use of measures for evaluating a method's performance. Section 5.4 presents the results of this performance evaluation study. A comparative study with an alternative method is conducted in Section 5.5. Finally, concluding remarks are provided in Section 5.6.

5.2 Preliminaries

This section provides an overview of the semi-parametric predictive method using copulas, denoted by NPI-C throughout this thesis, presented by Coolen-Maturi et al. [29]. The NPI-C method uses NPI for the marginals together with a parametric copula to take the dependence structure between the random quantities into account. Assume n bivariate observations, denoted as (x_i, y_i), $i = 1, 2, ..., n$, and let x_i and y_j represent the ordered observations related to the marginals, where $i = 1, 2, ..., n$ and $j = 1, 2, ..., n$. In order to predict one future bivariate observation, (X_{n+1}, Y_{n+1}), Hill's assumption $A_{(n)}$ is used to derive a predictive probability distribution for X_{n+1} given the observations $x_1, x_2, ..., x_n$, and a predictive probability distribution for Y_{n+1} given the observations $y_1, y_2, ..., y_n$, as follows:

$$P(X_{n+1} \in (x_{i-1}, x_i)) = \frac{1}{n+1} \text{ and } P(Y_{n+1} \in (y_{j-1}, x_j)) = \frac{1}{n+1}, \; i, j = 1, 2, ..., n+1,$$

where $x_0, y_0 = -\infty$ and $x_{n+1}, y_{n+1} = \infty$. The bivariate random quantity (X_{n+1}, Y_{n+1}) is then transformed from the $[-\infty, \infty]^2$ plane to the $[0, 1]^2$ plane, where the plane $[0, 1]^2$ is partitioned into $(n + 1)^2$ squares of equal size based on the observed n bivariate observations. The resulting transformed random quantities are denoted by $(\tilde{X}_{n+1}, \tilde{Y}_{n+1})$, where

$$\tilde{X}_{n+1} \in \Big(\frac{i-1}{n+1}, \frac{i}{n+1}\Big) \text{ and } \tilde{Y}_{n+1} \in \Big(\frac{j-1}{n+1}, \frac{j}{n+1}\Big),$$

where, $i, j = 1, 2, ..., n + 1$. Then based on $A_{(n)}$ assumption for the marginals,

$$P\Big(\tilde{X}_{n+1} \in \Big(\frac{i-1}{n+1}, \frac{i}{n+1}\Big)\Big) = \frac{1}{n+1} \text{ and } P\Big(\tilde{Y}_{n+1} \in \Big(\frac{j-1}{n+1}, \frac{j}{n+1}\Big)\Big) = \frac{1}{n+1}.$$

After the transformation for the marginals, the corresponding uniform marginal distributions are discretised on $[0, 1]$, which aligns with copulas, since copulas have uniform marginals. In the next step, a bivariate copula is assumed, and the parameter is estimated. To align with the marginal transformations and to allow for the estimation of the copula parameter, denoted by $\hat{\theta}$, independently of the marginals, the transformed data pairs are used in the estimation step. Instead of using the original observed pairs (x_i, y_i) for $i = 1, 2, ..., n$, the ranks $(\frac{r_i^x}{n+1}, \frac{r_i^y}{n+1})$ are used, where r_i^x is the rank of x_i among the

n pairs of x observations, and r_i^y is the rank of y_i among the n pairs of y values. By combining NPI for marginals with the estimated copula parameter $\hat{\theta}$, the probability for the event that the transformed pair $(\tilde{X}_{n+1}, \tilde{Y}_{n+1})$ lies within any of the $(n+1)^2$ squares of the partitioned plane $[0,1]^2$ is:

$$h_{ij}(\hat{\theta}) = P_C(\tilde{X}_{n+1} \in \left(\frac{i-1}{n+1}, \frac{i}{n+1}\right), \tilde{Y}_{n+1} \in \left(\frac{j-1}{n+1}, \frac{j}{n+1}\right) |\hat{\theta}), \tag{5.5}$$

where $i, j = 1, 2, ..., n+1$, and $P_C(.|\hat{\theta})$ is the copula probability with the estimated parameter $\hat{\theta}$. Suppose $T_{n+1} = X_{n+1} + Y_{n+1}$ and the event $T_{n+1} > t$ is the event of interest, then the corresponding NPI lower and upper probabilities for this event are

$$\underline{P}(T_{n+1} > t) = \sum_{(i,j) \in L_t} h_{ij}(\hat{\theta}), \tag{5.6}$$

where $L_t = \{(i,j) : x_{i-1} + y_{j-1} > t, i, j \in \{1, 2, ..., n+1\}\}$, and

$$\overline{P}(T_{n+1} > t) = \sum_{(i,j) \in U_t} h_{ij}(\hat{\theta}), \tag{5.7}$$

where $U_t = \{(i,j) : x_i + y_j > t, i, j \in \{1, 2, ..., n+1\}\}$.

In order to evaluate the performance of the NPI-C method, a simulation study was conducted by Coolen-Maturi et al. [29] and Muhammad [85], for this event of interest. At each simulation step $j = 1, 2, ..., N$, $n+1$ bivariate data pairs are generated from an assumed parametric copula family. The first n pairs are used as the observed data for the NPI-C method, while the remaining pair is used to evaluate the performance of the method. Let (x_i^j, y_i^j) denote the i-th pair in the j-th sample, where each sample consists of n pairs, and let (x_f^j, y_f^j) be the last pair for the j-th simulation step. Let t_f^j be the sum of the last remaining pair in the j-th sample, then $t_f^j = x_f^j + y_f^j$. Using Equations (5.6) and (5.7), and for a given quantile level $q \in (0,1)$, the inverse values for the NPI lower and upper probabilities for the event $T_{n+1} > t$ can be derived and denoted by $\underline{t}_q^j$ and $\overline{t}_q^j$ for each simulation step j. The key to this evaluation is to satisfy the inequality $l \leq q \leq u$, where l and u are defined as:

$$l = \frac{1}{N} \sum_{j=1}^{N} \mathbb{I}(t_f^j \geq \overline{t}_q^j), \tag{5.8}$$

and

$$u = \frac{1}{N} \sum_{j=1}^{N} \mathbb{I}(t_f^j \geq \underline{t}_q^j), \tag{5.9}$$

where $j = 1, 2, \ldots, N$.

The simulation study conducted by Coolen-Maturi et al. [29] and Muhammad [85] focused on two scenarios. The first scenario assumed that the copula used for generating the data was the same as the copula used for the NPI-C method. The second scenario involved two different copulas, one for the NPI-C method and another for simulating the data. The first scenario is important for assessing how well the method performs and for studying the level of imprecision expressed by $u - l$. Although the first scenario is important, the second scenario may be more relevant in practice, as in real-life applications, the copula family assumed for the data is unknown and prone to potential misspecification. Two values, l and u, are calculated for each quantile level q, as $\underline{t}_q^j \leq \overline{t}_q^j$ holds, the performance is evaluated by checking whether q lies within the interval $[l, u]$. As the sample size n increases, the imprecision $u - l$, decreases, resulting in a higher number of q values not satisfying the inequality $l \leq q \leq u$. This chapter investigates the performance of the NPI-C method using different performance evaluation measures, aiming to include studying cases when q lies outside the interval $[l, u]$. In the following section, the performance measures used for the evaluation process in this chapter are presented.

5.3 Performance evaluation measures

The aim of this chapter is to examine performance measures for imprecise probability inferences, with a focus on NPI with copulas for bivariate data, in order to evaluate the performance of the NPI-C method presented by Coolen-Maturi et al. [29] and Muhammad [85]. The performance measures considered in this chapter address two aspects of evaluation: accuracy and precision, as explained in Section 5.1. Accuracy refers to whether an interval contains a target value, while precision concerns the width of the interval. In what follows, several performance measures are presented to capture these two aspects.

Prediction Interval Coverage Probability (PICP) is a measure used to assess the efficiency of predictive methods. It measures the proportion of target values (actual observations) which fall within the prediction intervals. Khosravi et al. [76] constructed

prediction intervals instead of point prediction produced neural network metamodels, aiming to determine best structures of metamodels. The PICP measure has been widely used in the literature to evaluate the performance of prediction intervals, and many authors have applied it across various applications, including Alcántara et al. [11], Pang et al. [90], and Shrestha and Solomatine [100]. Given N_c experimental runs, let the prediction interval for the k-th run be $[l_k, u_k]$ and the target value for the k-th run be q_k. The Prediction Interval Coverage Probability (PICP) is defined by:

$$\text{PICP} = \frac{1}{N_c} \sum_{k=1}^{N_c} \mathbb{I}(l_k \leq q_k \leq u_k). \tag{5.10}$$

A high value of PICP indicates good performance; however, if it is too high, the intervals may be excessively wide, suggesting high indeterminacy. Conversely, if the value is too low, the intervals may be too narrow, indicating low uncertainty. While PICP assesses how well the intervals capture the target values, it does not provide insight into the imprecision level of these intervals.

Mean Prediction Interval Width (MPIW) is a measure defined to consider the width of the prediction interval. The metric is used to measure the average width of the prediction intervals in order to assess how narrow or wide the intervals are [100]. For N_c experimental runs, the mean prediction interval width (MIPW) is defined as:

$$\text{MPIW} = \frac{1}{N_c} \sum_{k=1}^{N_c} (u_k - l_k), \tag{5.11}$$

where $u_k - l_k$ is the width of the interval k-th. Lower values of MPIW indicate that the intervals is narrow (i.e., less imprecision), whereas higher values suggest that the intervals are likely to be wide (i.e., high imprecision).

Two additional measures aim to separate the intervals that include the target values from those that do not, thereby creating two groups. The average of each group is then calculated separately. These measures are constructed using the two previously defined measures in Equations (5.10) and (5.11). For clarity and conciseness, the first group, where the target values fall within the intervals, is referred to as Case 1. The second group, where the intervals do not contain the target values, is referred to as Case 2. The MPIW for Case 1 (MPIW-C1) and for Case 2 (MPIW-C2) are introduced below.

Specifically, MPIW-C1 is used to assess the level of imprecision (width) for intervals that include the target values, while MPIW-C2 measures the imprecision (width) of intervals that do not contain the target values.

Mean Prediction Interval Width for Case 1 (MPIW-C1) is constructed using the previous measures in Equations (5.10) and (5.11). It is defined by

$$\text{MPIW-C1} = \frac{\sum_{k=1}^{N_c}(u_k - l_k) \times \mathbb{I}(l_k \leq q_k \leq u_k)}{\sum_{k=1}^{N_c}\mathbb{I}(l_k \leq q_k \leq u_k)}. \tag{5.12}$$

This measure aims to assess the level of imprecision (width) for the prediction interval that include the target values. To measure the imprecision (width) of the intervals that do not contain the target values, the following measure is used. **Mean Prediction Interval Width (MPIW) for Case 2 (MPIW-C2)** is defined by

$$\text{MPIW-C2} = \frac{\sum_{k=1}^{N_c}(u_k - l_k) \times (1 - \mathbb{I}(l_k \leq q_k \leq u_k))}{\sum_{k=1}^{N_c}(1 - \mathbb{I}(l_k \leq q_k \leq u_k))}. \tag{5.13}$$

By applying these measures, prediction intervals results from a predictive method can be assessed in terms of both accuracy and precision. Although these measures are useful, it is important to further study the distance between the target value and the corresponding interval, whether the target value falls within its corresponding interval or not. For instance, if there are two prediction intervals, $[0.8, 0.9]$ and $[0.5, 1]$, and the target value is 0.7, then one may consider the second interval is the best choice as it contains the target value 0.7, but in fact, in some applications, the first interval can be more informative than the second, even if it does not contain the value 0.7. This can be explained by two factors. The first factor is the width of the intervals, where the width of the first interval is much less than the width of the second interval; 0.1 and 0.5, respectively. The second reason is that the first interval, $[0.8, 0.9]$, is reasonably close to the true value 0.7 although it does not contain it.

In statistics, a loss function is often used to measure the distance between the predicted values and the target values. Several loss functions are commonly used, including the quadratic loss function [10, 68], the absolute loss function [68, 92], the Huber loss function [68, 72], and the Hinge loss function [109]. Studying the performance of predictive methods using loss functions is interesting. This section also presents loss functions to evaluate the performance of predictive methods, taking into account the distance between

the target value and the interval. The discussion focuses on two types: the quadratic and absolute loss functions, which are defined below. Other loss functions, though also of interest, are left for future research.

The quadratic loss function (L_2) is defined as the squared distance between the predicted value (p) and the target value (q):

$$L_2 = (p - q)^2. \tag{5.14}$$

The L_2 loss function penalises larger errors heavily, so it is sensitive to outliers.

The absolute loss function (L_1) measures the absolute value of the distance between the predicted value (p) and the target value (q):

$$L_1 = |p - q|. \tag{5.15}$$

This loss function is less sensitive to outliers than the L_2 loss function.

In certain applications, strict adherence to either the lower or upper bound of the interval is enforced, depending on the context and decision criteria. If the target value falls outside the interval, a metric is used to assign a score to the interval, assigning penalties accordingly. The Interval score measure is detailed below.

Here, explanation of how the loss functions are used in this study is provided. Regarding quadratic loss L_2 and absolute loss L_1, two types of intervals are considered to derive the minimum loss and the maximum loss. First, when the interval contains the target value, the minimum loss of the interval is 0, and the maximum loss is defined as the maximum distance between the target value and either the lower or upper bound of the interval, depending on the loss function used. Secondly, if the interval does not contain the target value, the minimum loss is the minimum distance between the target value and either the lower or upper bound of the interval, and the maximum loss is obtained as in the first type.

Interval Score ($\mathbf{IS}_\alpha$) is a metric defined as a score for the $(1 - \alpha) \times 100\%$ central prediction interval [62]. It is a linear function which remains constant between the lower and upper bounds and has slope of $\pm\frac{2}{\alpha}$ outside the interval. This metric comprises three terms, each providing a penalisation while evaluating the intervals. Let $[l, u]$ represent the $(1 - \alpha) \times 100\%$ central prediction interval, and q denote the target value. The IS_α is

defined as follows:

$$\mathrm{IS}_\alpha = (u - l) + \frac{2}{\alpha}\max\{0, l - q\} + \frac{2}{\alpha}\max\{0, q - u\}. \tag{5.16}$$

This metric assigns a value for each prediction interval, with a lower score indicating better performance. The value of the IS_α is 0 when $l = q = u$, which is the optimal score. In this study, this metric is refined by replacing the original coefficients 1, $\frac{2}{\alpha}$, and $\frac{2}{\alpha}$ with new weights c_1, c_2, c_3, respectively, where $\sum_{i=1}^{3} c_i = 1$ and $c_i \geq 0$. As a result, this allows for a reformulation of the IS_α using the new weights, which adds flexibility in a given context. Let $[l, u]$ be an interval, the new interval score $\mathrm{IS}_{(c_1,c_2,c_3)}$ is defined as follows:

$$\mathrm{IS}_{(c_1,c_2,c_3)} = c_1(u - l) + c_2\max\{0, l - q\} + c_3\max\{0, q - u\}. \tag{5.17}$$

The weights c_2 and c_3 are used to penalise the distances between the target value and either the lower bound or the upper bound, represented by l and u, respectively, depending on the application requirements. The weight c_1 is used to penalise the width of the interval. If l and u are in $[0, 1]$, then the minimum and maximum scores will be 0 and 1, respectively. A lower score reflects better performance, with 0 being the optimal score. This measure can assess the performance of the method in terms of both accuracy and precision.

Let us consider two prediction intervals: $[1, 5]$ and $[3, 5]$ and let the target value be $q = 2$. The $\mathrm{IS}_{(c_1,c_2,c_3)}$ for these intervals using (5.17) can be determined as follows. For the first interval $[1, 5]$:

$$\mathrm{IS}_{(c_1,c_2,c_3)} = c_1(5 - 1) + c_2\max\{0, 1 - 2\} + c_3\max\{0, 2 - 5\} = 4c_1,$$

and for the second interval $[3, 5]$:

$$\mathrm{IS}_{(c_1,c_2,c_3)} = c_1(5 - 3) + c_2\max\{0, 3 - 2\} + c_3\max\{0, 2 - 5\} = 2c_1 + c_2.$$

Regarding the first interval, the score depends only on c_1 since the interval $[1, 5]$ includes the true value $q = 2$. Therefore, the penalisation will be determined based on the width of the interval according to the application or the decision criteria. On the other hand, the $\mathrm{IS}_{(c_1,c_2,c_3)}$ for the second interval is influenced by the weights c_2 and c_3, because $q = 2$

does not fall within the interval $[3, 5]$, specifically lying to the left of its lower bound 3.

This chapter investigates the use of these performance measures for evaluating the NPI-C method. In the following section, a simulation study is conducted, enabling the use of the measures to assess the performance of the NPI-C method in terms of two evaluation aspects: accuracy and precision.

5.4 Performance evaluation via simulation study

This section presents the simulation study conducted to evaluate the performance of the NPI-C method, introduced by Coolen-Maturi et al. [29]. In this simulation study, the main steps as presented by Coolen-Maturi et al. [29] and Muhammad [85] are followed with slight modifications. First, the entire simulation steps are repeated a number of times, say $N_c = 100$, to obtain 100 intervals instead of one interval. Additionally, due to the computation cost, the number of repeating internal simulation steps, N, is reduced from 1000 to 500 during the calculation of the probabilities using Equations (5.8) and (5.9). The steps used to implement this simulation study are presented in Algorithm 5.1. Two scenarios are considered: the first scenario uses the same parametric copula for both generating the samples and the NPI-C method. The second scenario assumes different parametric copulas, one for simulating the samples and the other for the NPI-C method.

In the first scenario, a Normal copula is used both to generate the samples and for the NPI-C method. In the second scenario, a Normal copula generates the data, while Frank and Clayton copulas are used for the NPI-C method. Three different values for each τ and q are employed: $\tau = 0.25, 0.5, 0.75$, and $q = 0.25, 0.5, 0.75$. When generating the samples, the parameter of the assumed parametric copula is set to the value corresponding to the selected value of τ. The estimation method used for estimating the copula parameter is pseudo maximum likelihood, which is available in the R package '`VineCopula`' [86]. For Clayton copula, the generated data should have a valid copula parameter to ensure the NPI-C method could be properly applied; this step should not affect the process, as parameter estimation methods are not the focus of this chapter. Two sample sizes are considered in this study: $n = 20$ and $n = 50$.

The main difference between this simulation study and the simulation study implemented by Coolen-Maturi et al. [29] and Muhammad [85] is that this simulation study

Algorithm 5.1 Calculating l_k and u_k, where $k = 1, \ldots, N_c$.

For $k = 1$ **to** N_c
  For $r = 1$ **to** N
    Generate $n + 1$ pairs sample from a parametric copula
    Use the first n pairs for the NPI-C method and the last pair for assessing performance
    Estimate the copula parameter $\hat{\theta}$ using the first n pairs sample
    Compute the probabilities $h_{ij}(\hat{\theta})$ using Equation (5.5)
    Compute $t^r_{n+1} = x^r_{n+1} + y^r_{n+1}$, $\underline{t}^r_q$, and $\overline{t}^r_q$
    Compute l and u using Equations (5.8) and (5.9)
  Return l and u
Return l_k and u_k, where $k = 1, \ldots, N_c$

results in N_c intervals for each q value, while the study by Coolen-Maturi et al. [29] and Muhammad [85] results in one interval corresponding to each q value. Obtaining a number of intervals allows for investigating the performance of the method in terms of applying the measures presented in Section 5.3. The intervals of proportions from this simulation study, $[l_k, u_k]$, for $k = 1, 2, \ldots, 100$, are derived from the NPI lower and upper probabilities following the steps in Algorithm 5.1, reflecting varying levels of imprecision. This chapter introduces measures, as described in Section 5.3, to evaluate the NPI-C method while accounting for imprecision, particularly for intervals that do not include the value q. For clarity, the value q is referred to as the "target value" throughout this study.

After obtaining the N_c intervals $[l_k, u_k]$, the first four measures in Section 5.3 are applied using Equations (5.10), (5.11), (5.12) and (5.13). The results are presented in Tables 5.1, 5.2 and 5.3. Table 5.1 displays the results for the first scenario, in which a Normal copula is used for both simulation and inference. Tables 5.2 and 5.3 show the results for the second scenario, where the data are generated using a Normal copula, and inference is based on assumed Frank and Clayton copulas.

The results in Table 5.2 indicate that for $n = 20$, the values of PICP, MPIW, MPIW-C1 and MPIW-C2 are greater than those observed for $n = 50$. To explain, as the sample size increases, the inverse values of the NPI lower and upper survival functions $\underline{t}^j_q$ and $\overline{t}^j_q$ tend to converge toward the quantile value t^j_f for $j = 1, 2, \ldots, N$. As a result, the probabilities l_k and u_k tighten around the value q, leading to a narrower width of the interval $[l_k, u_k]$. Reducing the width of the interval $[l_k, u_k]$ affects the number of intervals that contain the value q, resulting in lower values of PICP when $n = 50$.

$\tau = 0.25$					
q	n	PICP	MPIW	MPIW-C1	MPIW-C2
0.25	20	0.78	0.0556	0.0579	0.0475
	50	0.48	0.0230	0.0253	0.0210
0.5	20	0.87	0.0668	0.0682	0.0577
	50	0.50	0.0260	0.0271	0.0250
0.75	20	0.88	0.0569	0.0574	0.0533
	50	0.35	0.0238	0.0254	0.0229
$\tau = 0.5$					
q	n	PICP	MPIW	MPIW-C1	MPIW-C2
0.25	20	0.83	0.0523	0.0527	0.0505
	50	0.38	0.0208	0.0212	0.0205
0.5	20	0.83	0.0564	0.0573	0.0519
	50	0.46	0.0220	0.0225	0.0216
0.75	20	0.87	0.0522	0.0530	0.0463
	50	0.46	0.0219	0.0239	0.0202
$\tau = 0.75$					
q	n	PICP	MPIW	MPIW-C1	MPIW-C2
0.25	20	0.75	0.0500	0.0504	0.0486
	50	0.42	0.0200	0.0208	0.0194
0.5	20	0.75	0.0478	0.0486	0.0456
	50	0.42	0.0206	0.0209	0.0203
0.75	20	0.78	0.0488	0.0508	0.0415
	50	0.38	0.0209	0.0227	0.0197

Table 5.1: Simulation from Normal; Normal copula assumed for inference.

For small sample sizes, such as $n = 20$, the interval $[l_k, u_k]$ tends to be wider, allowing more q values to satisfy the inequality $l_k \leq q \leq u_k$. Notably, the results show that MPIW-C1 $>$ MPIW $>$ MPIW-C2, indicating that intervals which do not contain the value q are slightly narrower than those that contain the value q.

In Table 5.2, when using a Frank copula for inference, the pattern of the results is generally similar to those in Table 5.1 with only slight differences. However, in Table 5.3, the results differ from those observed in Table 5.1 and Table 5.2 when $q = 0.5$. When generating the samples using a Normal copula, they are characterised by having a symmetric dependence structure and no tail dependence. Conversely, using a Clayton copula for inference, it models the data as having asymmetric dependence structure and lower tail dependence. This tends to affect the results for the NPI-C method. Since the Clayton copula tends to capture strong lower tail dependence, the corresponding lower

$\tau = 0.25$					
q	n	PICP	MPIW	MPIW-C1	MPIW-C2
0.25	20	0.78	0.0560	0.0581	0.0485
	50	0.47	0.0232	0.0261	0.0206
0.5	20	0.85	0.0666	0.0680	0.0584
	50	0.50	0.0264	0.0271	0.0257
0.75	20	0.89	0.0569	0.0574	0.0524
	50	0.44	0.0249	0.0267	0.0235
$\tau = 0.5$					
q	n	PICP	MPIW	MPIW-C1	MPIW-C2
0.25	20	0.76	0.0512	0.0524	0.0473
	50	0.39	0.0205	0.0225	0.0192
0.5	20	0.81	0.0550	0.0552	0.0543
	50	0.42	0.0239	0.0257	0.0226
0.75	20	0.80	0.0497	0.0509	0.0450
	50	0.43	0.0213	0.0219	0.0208
$\tau = 0.75$					
q	n	PICP	MPIW	MPIW-C1	MPIW-C2
0.25	20	0.75	0.0498	0.0516	0.0444
	50	0.46	0.0198	0.0206	0.0192
0.5	20	0.77	0.0484	0.0489	0.0467
	50	0.46	0.0212	0.0218	0.0207
0.75	20	0.81	0.0497	0.0509	0.0442
	50	0.46	0.0205	0.0223	0.0189

Table 5.2: Simulation from Normal; Frank copula assumed for inference.

and upper inverse quantiles $\underline{t}_q^j$ and $\overline{t}_q^j$ may be sensitive to the dependence near the centre, often resulting in lower values of both l_k and u_k. Consequently, many values of q lie above the interval $[l_k, u_k]$. The observed intervals for $q = 0.5$ show that the value q frequently falls outside the interval $[l_k, u_k]$ above the upper bound, that is, when $u_k < q$, as shown in Figures D.7 to D.9 in Appendix D.

For illustration, the intervals $[l_k, u_k]$ for $k = 1, ..., 100$ are plotted in Figures D.1 to D.9 in Appendix D. Figures D.1, D.2 and D.3, when using a Normal copula for inference, show that although many intervals do not contain the value q when $n = 50$, the intervals are very close to the value q. Figures D.4, D.5 and D.6 display comparable trends to those observed in the first three figures.

Studying the performance of the NPI-C method using loss functions is considered next. First, the maximum and minimum losses are derived using the quadratic loss function L_2

$\tau = 0.25$					
q	n	PICP	MPIW	MPIW-C1	MPIW-C2
0.25	20	0.82	0.0568	0.0587	0.0486
	50	0.53	0.0223	0.0239	0.0206
0.5	20	0.67	0.0672	0.0687	0.0642
	50	0.31	0.0269	0.0281	0.0264
0.75	20	0.90	0.0569	0.0576	0.0498
	50	0.38	0.0239	0.0251	0.0231
$\tau = 0.5$					
q	n	PICP	MPIW	MPIW-C1	MPIW-C2
0.25	20	0.87	0.0536	0.0541	0.0498
	50	0.45	0.0217	0.0232	0.0205
0.5	20	0.60	0.0553	0.0565	0.0535
	50	0.27	0.0237	0.0255	0.0230
0.75	20	0.80	0.0492	0.0510	0.0421
	50	0.34	0.0220	0.0229	0.0215
$\tau = 0.75$					
q	n	PICP	MPIW	MPIW-C1	MPIW-C2
0.25	20	0.77	0.0492	0.0504	0.0452
	50	0.45	0.0200	0.0212	0.0189
0.5	20	0.67	0.0487	0.0491	0.0479
	50	0.28	0.0203	0.0222	0.0195
0.75	20	0.78	0.0482	0.0501	0.0415
	50	0.37	0.0205	0.0225	0.0193

Table 5.3: Simulation from Normal; Clayton copula assumed for inference.

and the absolute loss function L_1, based on Equations (5.14) and (5.15), as described in Section 5.3. Next, the average maximum and minimum losses are presented, with results reported separately for cases where q falls within the interval and where it does not. The results when applying the quadratic loss function L_2 are presented in Figures D.10 to D.14 in Appendix D, and additional results for the absolute loss function L_1 are provided in Appendix D.

Regarding the average values of the maximum loss using L_2 in Figure D.10, the results are quite similar when using either Normal or Frank copulas for inference. When $\tau = 0.75$ (lower tail), the results using a Clayton copula for inference remain consistent with those obtained using the other two copulas for inference. Using a Clayton copula, the averages of the maximum losses when $q = 0.5$, with both values of $\tau = 0.25$ and $\tau = 0.5$, are relatively high. This is due to the observation that, as the value of τ increases,

the resulting intervals $[l_k, u_k]$ become narrower, reducing the level of imprecision and, consequently, the maximum loss values. This decrease in the level of imprecision can be seen in the corresponding values of the measures MPIW, MPIW-C1, and MPIW-C2 in Table 5.3. In Figures D.10, D.11 and D.12, the average values of the maximum losses for $q = 0.5$ are greater when $\tau = 0.25$ than when $\tau = 0.5$ or $\tau = 0.75$. Notably, the differences between the averages when $n = 20$ and $n = 50$, for $\tau = 0.25$ are larger than the differences observed for $\tau = 0.5$ and $\tau = 0.75$. Similar patterns are observed in Figure D.12, suggesting that when $q \in [l_k, u_k]$, any assumed parametric copula for inference leads to comparable results.

In terms of the average values of the minimum loss obtained using the L_2, Figure D.13 shows that when using a Clayton copula for inference, the highest values are achieved when $q = 0.5$ holds across all values of τ, with the highest average occurring for $\tau = 0.5$. This is due to the large number of intervals that do not include the value q, which is most frequent when $\tau = 0.5$ compared to the other cases.

When the averages of the minimum loss values are reported, as shown in Figure D.14, the overall averages when $n = 50$ are greater than those when $n = 20$. This is because when $n = 50$, more intervals do not contain the value q compared to $n = 20$. However, the average values remain relatively low, as a result of the intervals being near the value q. An exception to this occurs when using a Normal copula with $\tau = 0.25$ and $q = 0.25$, where the average value when $n = 20$ is slightly higher than when $n = 50$. This is caused by the presence of some intervals that are far from the target value q when $n = 50$.

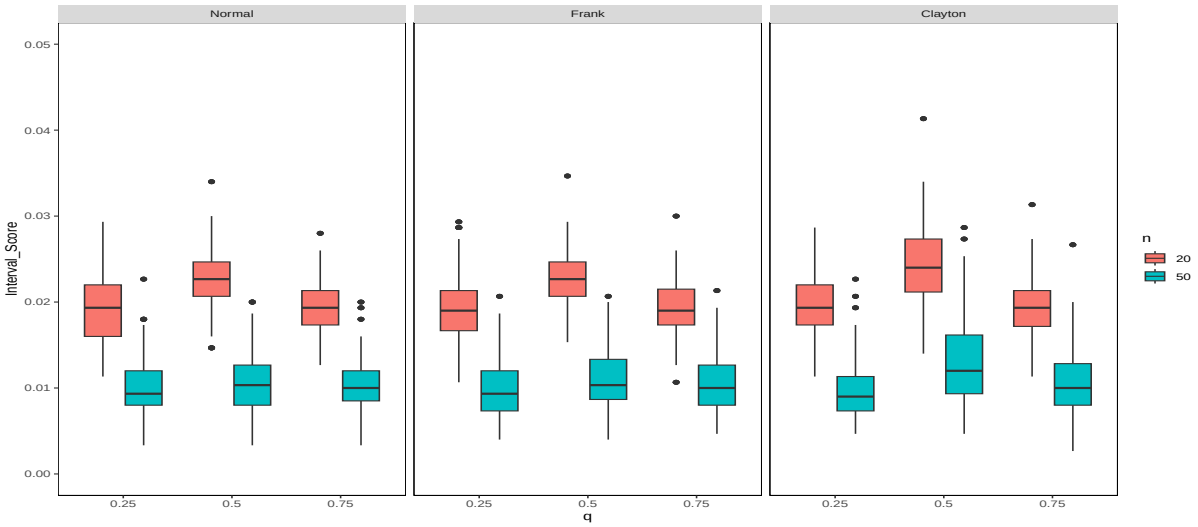
In general, the average values of the maximum loss using $L2$ tend to decrease as τ increases. This can be attributed to the strong dependence structure, which enhances the ability of the copula to model the data more accurately, resulting in more accurate intervals. The results also demonstrate that using loss functions to evaluate the performance of the NPI-C method is effective. For example, in the presence of a strong dependence structure, any copula used for inference produces almost identical results for both $q = 0.25$ and $q = 0.75$. If the goal is to achieve more precise intervals while relaxing the strict inclusion criterion, a larger sample size (e.g., $n = 50$) can yield satisfactory results.

Finally, the interval score measure $\text{IS}_{(c_1,c_2,c_3)}$, as presented in Section 5.3, is used in this simulation study, assuming different values for the weights c_1, c_2, and c_3. It should be

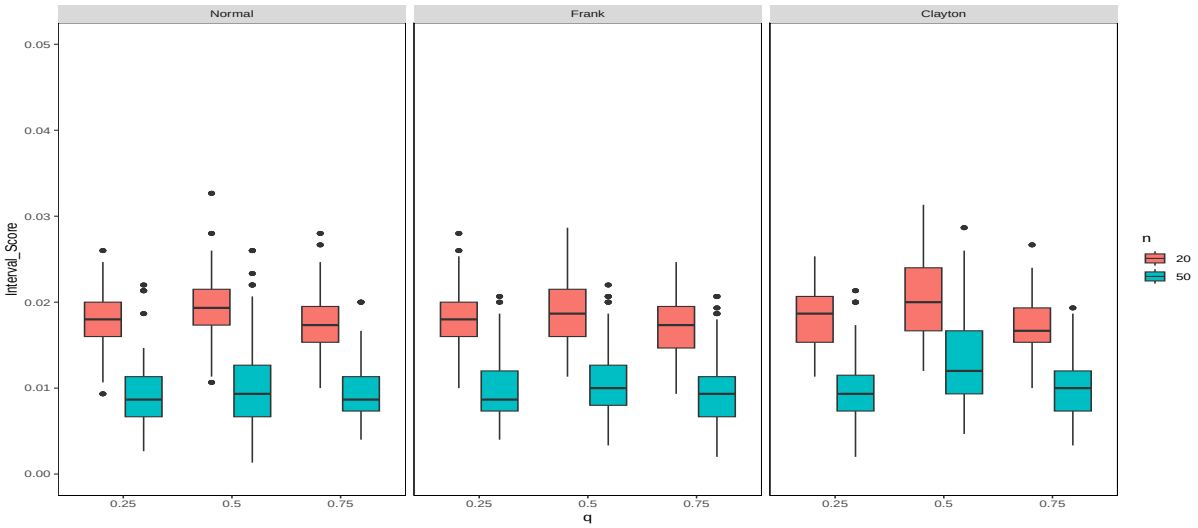
noted that determining appropriate values for these weights depends on the context and specific application requirements. However, the selection of the weights in this chapter is primarily intended to observe the performance of the method across different cases and to examine how changes in the weights affect the overall performance. Two cases are considered in this chapter: first, when $c_1 = c_2 = c_3 = \frac{1}{3}$, and second, when $c_1 = 0.2$ and $c_2 = c_3 = 0.4$. Additional cases are included in Appendix D. The scores are represented as box plots in Figures 5.1 and 5.2.

In Figure 5.1, when using equal weights, the results show that when $n = 50$, the intervals often achieve lower scores than when $n = 20$. As explained earlier in this section, when $n = 50$ the intervals $[l_k, u_k]$ tend to be close to the q value even if they do not capture the q value, resulting in lower scores compared to cases when $n = 20$. When penalising intervals that do not include the q value more strictly, as shown in Figure 5.2 ($\text{IS}_{(0.2,0.4,0.4)}$), and assigning a lower penalty to the width of the interval, the results indicate that the scores of the intervals are lower than those of $\text{IS}_{(\frac{1}{3},\frac{1}{3},\frac{1}{3})}$. This difference is affected by being less strict about the degree of imprecision of the intervals, which reduces the gap between the cases $n = 20$ and $n = 50$. When $n = 20$, more intervals satisfy the condition $l_k \leq q \leq u_k$, although they are often wide compared to those when $n = 50$. However, when $\text{IS}_{(0.2,0.4,0.4)}$, the results show more outliers, particularly when $n = 50$, due to the presence of intervals with $u < q$ or $l > q$ that are far from the q value.

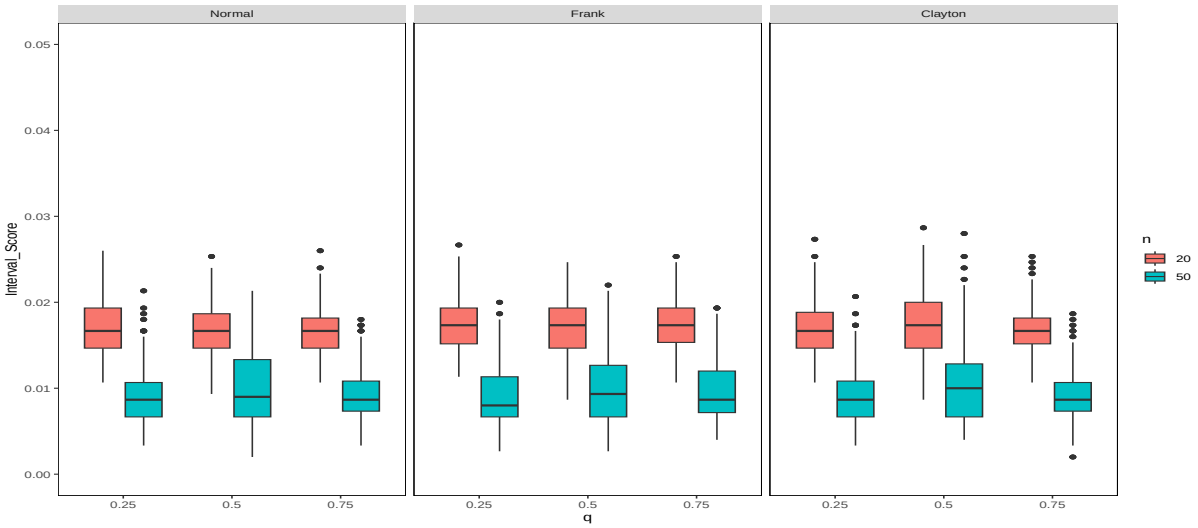
Overall, most of the values of interval score when $n = 50$ tend to be lower than those observed when $n = 20$, indicating that when $n = 50$, even if the intervals do not satisfy the inequality $l_k \leq q \leq u_k$, they are likely to be close to the q value. Selecting the weights c_1, c_2, and c_3 primarily depends on the application requirements. For example, if an application requires the predictive intervals to strictly include the target value or to be outside but near the lower bound, then a higher value should be assigned to the weight c_2, and the method that produces the lowest interval score will be considered the best.



(a) $\tau = 0.25$

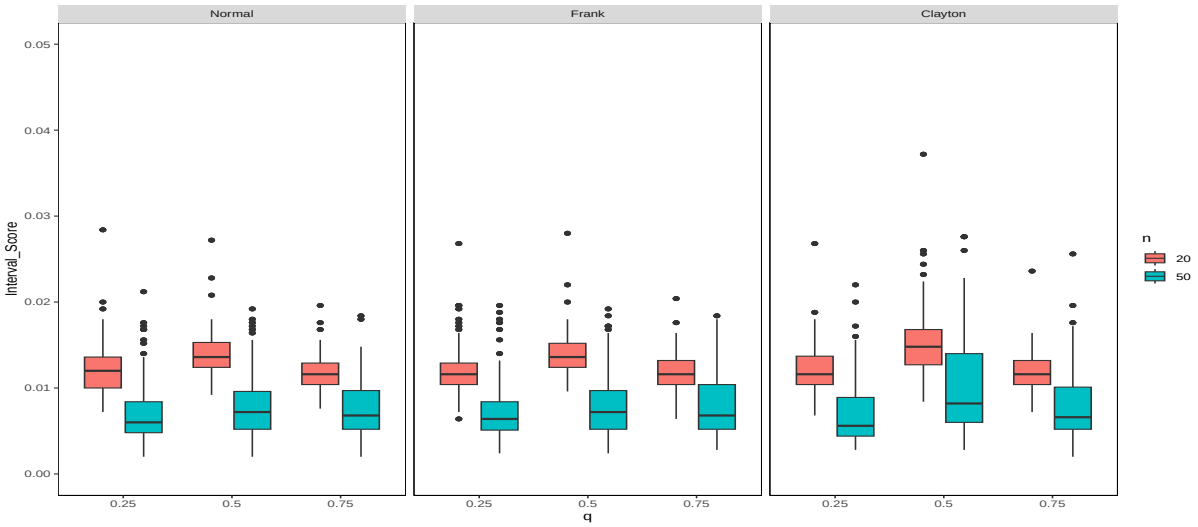


(b) $\tau = 0.5$

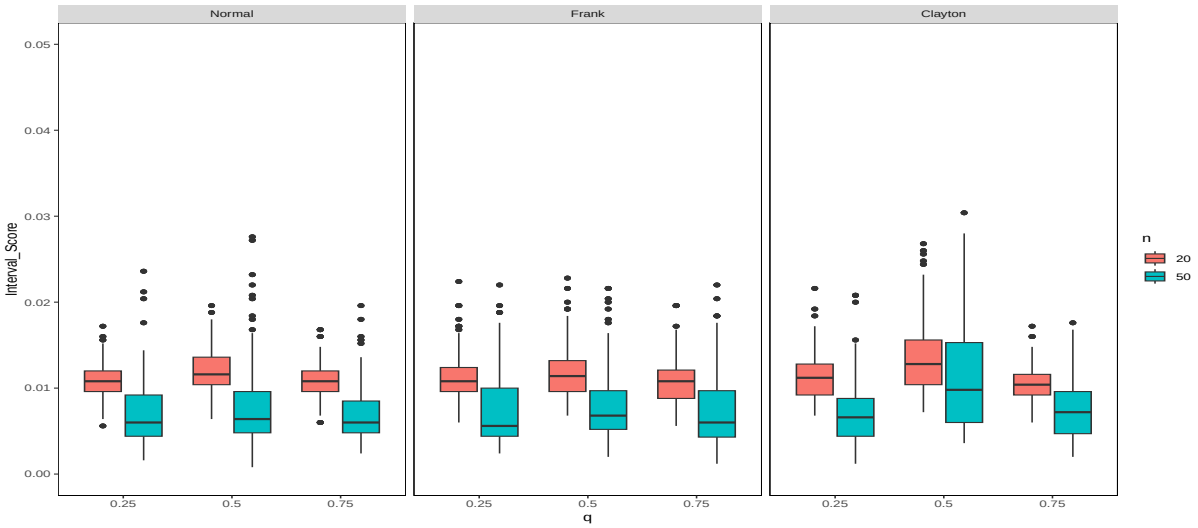


(c) $\tau = 0.75$

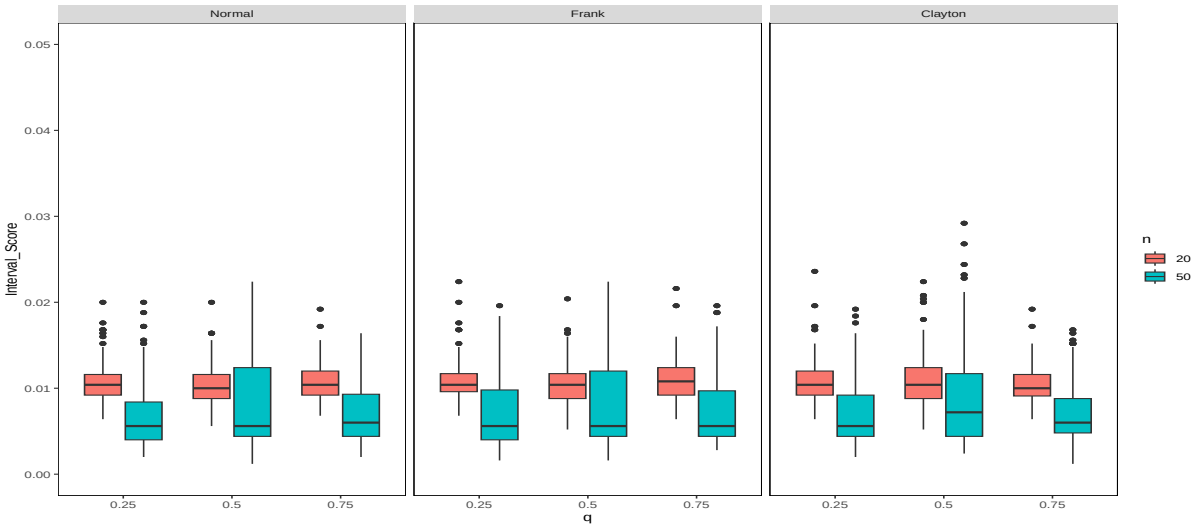
Figure 5.1: Results of $\text{IS}_{(\frac{1}{3},\frac{1}{3},\frac{1}{3})}$; simulation from Normal copula; Normal, Frank, and Clayton copulas are for inference.



(a) $\tau = 0.25$



(b) $\tau = 0.5$



(c) $\tau = 0.75$

Figure 5.2: Results of $\text{IS}_{(0.2,0.4,0.4)}$; simulation from Normal copula; Normal, Frank, and Clayton copulas are for inference.

5.5 Comparison with existing method

In this section, the performance evaluation measures presented in Section 5.3 are used to compare the NPI-C method with another method introduced by Muhammad [85]. This method employs a nonparametric copula with NPI for bivariate data, unlike the NPI-C method, which uses a parametric copula. It is referred to as NPI-NC throughout this chapter. In the NPI-NC method, NPI is applied to the marginals and combined with a nonparametric copula. Only kernel-based methods are used in NPI-NC, providing greater flexibility than the parametric copulas used in the NPI-C method [85]. For more details on the NPI-NC method, see [85]. The aim of this section is to compare the performance of the NPI-C and NPI-NC methods, following the same simulation study steps presented in Section 5.4.

A Normal copula is used to simulate the data, while a nonparametric copula is used for inference. Since the primary aim of this chapter is to propose a new use of performance measures for NPI-based methods rather than introduce a new predictive method, a single nonparametric copula estimation method, the normal reference rule-of-thumb for bandwidth selection in kernel-based nonparametric estimation [101] is applied, and the results are compared to those in Section 5.4. The normal reference rule-of-thumb is implemented using the '`npudistbw`' function in the '`np`' package in R [96].

In this comparison study, the value of τ is set to 0.25, and three values of q are used, $q = 0.25, 0.5, 0.75$. Two sample sizes are employed in this section: $n = 20$ and $n = 50$. As explained in Section 5.4, before applying the measures presented in Section 5.3, the main steps in Algorithm 5.1 are followed to obtain 100 intervals $[l_k, u_k]$, but using a nonparametric copula instead of a parametric copula. After obtaining 100 intervals, the first four measures described in Section 5.3, namely Equations (5.10), (5.11), (5.12) and (5.13), are applied.

The results obtained using the nonparametric copula are presented and compared to those obtained using Normal, Frank, and Clayton copulas in Section 5.4, as shown in Tables 5.4, 5.5 and 5.6. The normal reference rule-of-thumb method for fixed bandwidth is indicated as fixed bandwidth in the tables.

Table 5.4 presents a comparison between the different scenarios in terms of the accuracy of the predictions, represented by the PICP measure, and in terms of precision,

presented by the measures: MPIW, MPIW-C1 and MPIW-C2.

When the sample size is small, $n = 20$, all copulas used for inference results in reasonably high values of PICP except the Clayton copula when $q = 0.5$. When using a Normal copula for inference, the intervals often tend to have small width, regardless of whether they include the q values or not, compared to using other copulas. As Frank copula is symmetric as Normal copula, it tends to perform similarly to Normal copula in most cases, except when $q = 0.75$ and $n = 50$, where using a Frank copula results in more intervals having the value q than when using a Normal copula. However, on average, the resulting intervals when using a Normal copula are less imprecise than when using a Frank copula. When using a nonparametric copula, the method performs well, but the widths of the intervals are slightly wide compared to using other copulas, except when $q = 0.75$ and $n = 50$, where it achieves the narrowest intervals, on average. The results indicate that when a method yields intervals with high accuracy (e.g., high PICP), it often tends to be less imprecise. Specifying the correct copula family before applying the method helps ensure a balanced tradeoff between the accuracy and the precision. If the copula family is known beforehand, it leads to more accurate probabilities without a high level of imprecision $(u - l)$. On the other hand, if the data comes from unknown copula family and it is difficult to identify it, it may be safer to use a nonparametric copula, especially when the sample size is large.

In order to investigate the performance of the methods in terms of the loss functions, the average values of the maximum and minimum losses for intervals are calculated and presented in Table 5.5. The average value of maximum losses when using $L2$ is referred to as MaxL_2, and MaxL_1 when applying $L1$. Similarly, the average value of minimum losses is referred to as MinL_2 when applying $L2$, and MinL_1 when applying $L1$.

The results in Table 5.5 show that when using L_2 loss function the average values of losses are nearly similar for most cases. However, when using the Clayton copula and $q = 0.5$ for both values of n, the values of MaxL_2 and MaxL_1 are slightly higher than in other copulas. Additionally, when using the nonparametric copula for $q = 0.75$ and $n = 20$, the value of the MaxL_2 is greater than that obtained from other copulas. The high value of MaxL_2 results from the high level of imprecision observed when using a nonparametric copula, compared to other copulas. This is reflected in the highest values of the measures MPIW, MPIW-C1 and MPIW-C2 among other copulas.

Using L_1 highlights the differences between the various scenarios more effectively than L_2, as shown in Table 5.5. When using the nonparametric copula, the results indicate that when $q = 0.75$ and $n = 50$, the value of MaxL_1 is the lowest among the other scenarios. This is again, caused by the low values of the measures MPIW, MPIW-C1 and MPIW-C2, indicating that when the intervals include the q value, the average of widths is low, but when they do not include it, the average of widths tends to be lower, with the value of q being outside but close to the bounds of the intervals.

Table 5.6 shows the average values of the interval score measure, which reflects both the accuracy and precision of the method, using different weights. Using $\text{IS}_{(\frac{1}{3},\frac{1}{3},\frac{1}{3})}$, the $c_1 = \frac{1}{3}$ places greater importance on the interval width compared to the weight $c_1 = 0.2$ using $\text{IS}_{(0.2,0.4,0.4)}$.

Using the $\text{IS}_{(\frac{1}{3},\frac{1}{3},\frac{1}{3})}$ places greater importance on the interval width compared to using $\text{IS}_{(0.2,0.4,0.4)}$. Consequently, when using any copula, the resulting values when $n = 20$ are higher than the values when $n = 50$. This is because, when $n = 20$, more of the intervals include the q value compared to when $n = 50$. However, the distance between the value q and the bounds of the intervals is shorter when $n = 50$ (e.g., MaxL_2 and MaxL_1 values are lower when $n = 50$), resulting in lower values of $\text{IS}_{(\frac{1}{3},\frac{1}{3},\frac{1}{3})}$ when $n = 50$. The lowest values are observed when $n = 50$ for $\text{IS}_{(0.2,0.4,0.4)}$, due to balancing between having narrow intervals along with q lying in or outside but close to the bounds of the intervals.

In this comparative study, the analysis presents the results of the performance measures used to evaluate the method when employing a parametric or a nonparametric copula for inference. Misspecification of the copula used for inference, compared to the one used to generate the data, leads to different results across the copulas. The performance measures allow comparison of the methods in terms of two key aspects: accuracy and precision.

Normal copula					
q	n	PICP	MPIW	MPIW-C1	MPIW-C2
0.25	20	0.78	0.0556	0.0579	0.0475
	50	0.48	0.0230	0.0253	0.0210
0.5	20	0.87	0.0668	0.0682	0.0577
	50	0.50	0.0260	0.0271	0.0250
0.75	20	0.88	0.0569	0.0574	0.0533
	50	0.35	0.0238	0.0254	0.0229
Frank copula					
q	n	PICP	MPIW	MPIW-C1	MPIW-C2
0.25	20	0.78	0.0560	0.0581	0.0485
	50	0.47	0.0232	0.0261	0.0206
0.5	20	0.85	0.0666	0.0680	0.0584
	50	0.50	0.0264	0.0271	0.0257
0.75	20	0.89	0.0569	0.0574	0.0524
	50	0.44	0.0249	0.0267	0.0235
Clayton copula					
q	n	PICP	MPIW	MPIW-C1	MPIW-C2
0.25	20	0.82	0.0568	0.0587	0.0486
	50	0.53	0.0223	0.0239	0.0206
0.5	20	0.67	0.0672	0.0687	0.0642
	50	0.31	0.0269	0.0281	0.0264
0.75	20	0.90	0.0569	0.0576	0.0498
	50	0.38	0.0239	0.0251	0.0231
Fixed bandwidth					
q	n	PICP	MPIW	MPIW-C1	MPIW-C2
0.25	20	0.81	0.0568	0.0583	0.0504
	50	0.51	0.0245	0.0251	0.0239
0.5	20	0.84	0.0669	0.0682	0.0596
	50	0.43	0.0275	0.0284	0.0268
0.75	20	0.80	0.0578	0.0576	0.0587
	50	0.41	0.0229	0.0241	0.0221

Table 5.4: Simulation from Normal; PICP, MPIW, MPIW-C1 and MPIW-C2.

Normal copula					
q	n	MaxL_2	MinL_2	MaxL_1	MinL_1
0.25	20	0.0022	0.0001	0.0450	0.0026
	50	0.0009	0.0001	0.0268	0.0068
0.5	20	0.0030	0.0000	0.0525	0.0015
	50	0.0010	0.0001	0.0295	0.0067
0.75	20	0.0020	0.0000	0.0430	0.0011
	50	0.0010	0.0001	0.0293	0.0073
Frank copula					
q	n	MaxL_2	MinL_2	MaxL_1	MinL_1
0.25	20	0.0022	0.0001	0.0454	0.0024
	50	0.0009	0.0001	0.0272	0.0070
0.5	20	0.0030	0.0000	0.0526	0.0015
	50	0.0010	0.0001	0.0295	0.0065
0.75	20	0.0020	0.0000	0.0429	0.0011
	50	0.0010	0.0001	0.0298	0.0074
Clayton copula					
q	n	MaxL_2	MinL_2	MaxL_1	MinL_1
0.25	20	0.0021	0.0001	0.0442	0.0023
	50	0.0008	0.0001	0.0264	0.0071
0.5	20	0.0039	0.0001	0.0590	0.0050
	50	0.0017	0.0004	0.0372	0.0127
0.75	20	0.0019	0.0000	0.0425	0.0009
	50	0.0010	0.0002	0.0292	0.0080
Fixed bandwidth					
q	n	MaxL_2	MinL_2	MaxL_1	MinL_1
0.25	20	0.0021	0.0000	0.0441	0.0020
	50	0.0010	0.0001	0.0282	0.0070
0.5	20	0.0031	0.0000	0.0531	0.0021
	50	0.0013	0.0002	0.0339	0.0087
0.75	20	0.0025	0.0000	0.0471	0.0024
	50	0.0010	0.0002	0.0284	0.0080

Table 5.5: Simulation from Normal; loss functions.

Normal copula			
q	n	$\text{IS}_{(\frac{1}{3},\frac{1}{3},\frac{1}{3})}$	$\text{IS}_{(0.2,0.4,0.4)}$
0.25	20	0.0122	0.0194
	50	0.0100	0.0073
0.5	20	0.0228	0.0140
	50	0.0109	0.0079
0.75	20	0.0193	0.0118
	50	0.0104	0.0077
Frank copula			
q	n	$\text{IS}_{(\frac{1}{3},\frac{1}{3},\frac{1}{3})}$	$\text{IS}_{(0.2,0.4,0.4)}$
0.25	20	0.0122	0.0195
	50	0.0100	0.0074
0.5	20	0.0227	0.0139
	50	0.0110	0.0079
0.75	20	0.0193	0.0118
	50	0.0108	0.0080
Clayton copula			
q	n	$\text{IS}_{(\frac{1}{3},\frac{1}{3},\frac{1}{3})}$	$\text{IS}_{(0.2,0.4,0.4)}$
0.25	20	0.0197	0.0123
	50	0.0098	0.0073
0.5	20	0.0241	0.0154
	50	0.0132	0.0105
0.75	20	0.0193	0.0117
	50	0.0106	0.0080
Fixed bandwidth			
q	n	$\text{IS}_{(\frac{1}{3},\frac{1}{3},\frac{1}{3})}$	$\text{IS}_{(0.2,0.4,0.4)}$
0.25	20	0.0196	0.0122
	50	0.0105	0.0077
0.5	20	0.0230	0.0142
	50	0.0121	0.0090
0.75	20	0.0193	0.0118
	50	0.0103	0.0078

Table 5.6: Simulation from Normal; interval score.

5.6 Conclusions

This chapter presented a novel use of performance evaluation measures for imprecise probability inferences focusing on NPI-based method for bivariate data. This method is the semi-parametric predictive method (NPI-C), which was introduced by Coolen-Maturi et al. [29]. When assessing the performance of the NPI-C method in [29, 85], it resulted in intervals of proportions $[l, u]$ corresponding to the quantile level q using NPI lower and upper probabilities. The aim of this study is to investigate if the interval $[l, u]$ is still informative when it does not contain the value q, taking into account the imprecision $u - l$ using different performance measures. These measures focus on the accuracy and precision of the predictions.

This study introduced different measures to evaluate the performance of the NPI-C method when dealing with prediction intervals. The first four measures, PICP, MPIW, MPIW-C1, and MPIW-C2, were introduced to measure the coverage (accuracy) and the width (imprecision) of the intervals. Moreover, loss functions were used to measure the maximum and minimum distances between $[l, u]$ and the target value q. Two loss functions were used: quadratic loss function (L_2) and absolute loss function (L_1). Finally, a modified interval score ($\text{IS}_{(c_1,c_2,c_3)}$) was introduced with three weights, c_1, c_2, and c_3, where c_1 penalises wider intervals, while c_2 and c_3 imposes penalties if $l > q$ and if $u < q$, respectively. The choice of the weights of this metric depends on the context and the application requirements.

A simulation study was conducted to evaluate the performance of the NPI-C method using the introduced measures. The study included two scenarios: one where the same parametric copula was used for both generating the data and inference, and another where different copulas were applied. Subsequently another study was carried out to apply the performance measures in order to compare the performance of the NPI-C method with a method introduced by Muhammad [85], which uses a nonparametric copula for inference. This method is referred to as NPI-NC throughout this chapter.

The results of this study showed that the intervals, when $n = 20$, tended to be more accurate than when $n = 50$. However, when $n = 50$, the intervals were more precise compared to the case when $n = 20$. Using the same copula can help achieve a tradeoff between the accuracy of the predictions and the precision of intervals. However, when

using a different copula for inference (parametric or nonparametric), it can still achieve good results, but with the risk of reducing performance in specific cases. For example, when a Normal copula for generating the data and a Clayton copula for inference, the Clayton copula may fail to capture the dependence in the centre, leading to lower performance compared to the other copulas when $q = 0.5$. Using a nonparametric copula can be a safer choice if the underling copula for the data is unknown or uncertain. For larger sample sizes, both method NPI-C and method NPI-NC can produce good results if the goal is to get more precise intervals, and the target value is accepted to be outside but close the bounds of the interval.

Using the loss functions revealed that when the inequality $l \leq q \leq u$ was not satisfied, the intervals tended to be near the value q. The average values of the maximum loss tended to decrease with an increase in the value of τ. This could be attributed to the ability of copula to model the data more accurately when the dependence was strong, which resulted in more precise intervals.

Using the interval score measure $\text{IS}_{(c_1,c_2,c_3)}$ proved effective in evaluating the performance of the methods in terms of both the accuracy and the precision. However, selecting the weights c_1, c_2, and c_3 is important and depends on the specific application. This motivates the development of a method to find optimal values for weights in this metric.

While this study aimed to apply different measures to evaluate the performance of the NPI method for bivariate data, it would be interesting to extend these measures to assess other methods and applications based on the NPI approach.

Chapter 6

Conclusions and future work

This thesis has contributed to nonparametric predictive classification methods and the performance evaluation of NPI-based methods. The choice of performance evaluation methods depends on the nature of the prediction, and this thesis involved various types of predictions, necessitating different evaluation measures. Key contributions include applying the D-NPI classification method introduced by Alharbi [12] to ensemble methods, imprecise classification, and multi-label classification, with an investigation of the performance of these new algorithms.

In classical classification methods, classifiers predict a single class for an instance, which is associated with a correct class label; this framework was used when evaluating the ensemble methods. In imprecise classification, classifiers predict a set of possible labels for an instance, which has a correct class label. In multi-label classification, classifiers predict a subset of labels for an instance, which is associated with a correct subset of labels. This work investigated the performance of the D-NPI algorithm, based on the NPI approach, across these classification problems. A further scenario of performance evaluation examined in this thesis involved providing an interval and assessing whether a single target value falls within it, as in the work by Coolen-Maturi et al. [29]. This chapter presents the main findings of the thesis and highlights several interesting directions for future research.

Chapter 2 introduced two ensemble methods, including bagging and random forest, using the D-NPI algorithm. These methods were named NPI-B for bagging and NPI-RF for random forest. These ensemble methods were designed to adopt the strengths of the D-NPI algorithm with the aim of investigating its performance when applied to ensemble

techniques. The performance of the ensemble methods was evaluated by comparing them to the D-NPI algorithm and other ensemble methods. Two main scenarios were included in this study. The first scenarios focused on comparing the proposed ensemble methods against the base classifier, while the second aimed to compare the proposed methods with a bagging method based on the Nonparametric Predictive Inference for Multinomial data model, which was referred to as NPI-M-B, and the random forest method referred to as RF. In the first scenario, the NPI-B and NPI-RF algorithms performed well compared to the D-NPI algorithm, with the best performance observed for the NPI-RF algorithm. An example was illustrated in-depth to further investigate the performance of the algorithms. The results of this example showed the ability of the NPI-RF algorithm to produce individual classification trees with diverse depth values, resulting in its high performance. In the second scenario, the RF method achieved good performance but it was the only method to produce fully grown individual trees. The NPI-RF algorithm struggled to achieve high accuracy across several datasets, due to its stopping criterion which prevents individual trees from learning complex patterns, particularly for large-sized datasets. The bagging methods NPI-B and NPI-M-B achieved similar performance for most datasets. For further investigation, the performance of the NPI-RF method was evaluated without the stopping criterion, allowing fully grown individual trees. This method was called NPI-RF-FG. The NPI-RF-FG method achieved higher performance compared to the NPI-RF algorithm, having competitive performance compared to the RF method. According to the results, allowing fully grown individual trees in the ensemble methods helps in capturing complex pattern. Building on our findings, subsequent research could extend the NPI-B and NPI-RF to weighted cost sensitive classification, aiming to address the problem arising from imbalanced class distribution.

In Chapter 3, three new algorithms for imprecise classification, referred to as D-NPI-IC1, D-NPI-IC2, and D-NPI-IC3, were proposed based on the D-NPI algorithm. The D-NPI algorithm used NPI lower and upper probabilities; therefore, this study aimed to develop new algorithms by considering different criteria for selecting the best attribute to stop building the classification tree. The selection process depended on the NPI lower and upper probabilities for CI for each attribute. During the process of building a classification tree, upon reaching a leaf node, a probability interval is assigned to each class label. Then, a dominance criterion was used to determine the set of non-dominated labels, which was

called strong dominance. An experimental study was conducted to evaluate and compare their performance with four existing imprecise classifiers from the literature, called NPI-M, A-NPI-M, IDM1 (hyperparameter $s = 1$), and IDM2 (hyperparameter $s = 2$) [1, 7]. Various metrics were used to evaluate the performance, along with using statistical tests for a comparative analysis. Among the metrics, DACC and MIC were used to report how informative the classifiers were. The D-NPI-IC1 and D-NPI-IC2 algorithms achieved the highest performance among the other algorithms. Extending these algorithms to ensemble methods would be an intriguing topic for future research. Additionally, it will be of interest to adapt the algorithms to weighted cost sensitive classification.

Chapter 4 introduced the D-NPI classification method and the D-NPI-IC1 imprecise classifier to multi-label classification using the Label Powerset (LP) approach. This study included two methods: the first method used classical classification methods for multi-label classification tasks to predict a single subset of labels. The second method used imprecise classification methods for MLC problems to predict a set of subsets of labels. The study included additional four algorithms from the literature, including NPI-M, A-NPI-M, IDM1, and IDM2 [1, 7]. When applying the first method, example-based measures were used to evaluate the performance of the classifiers on a per-instance bases. Further performance measures are necessary to be explored in future research, including label-based classification measures and ranking-based measures. Performance evaluation measures for imprecise classification were used when applying the second method, as used in Chapter 3. An experiment study was conducted, and the results varied across the classifiers according to the characteristics of the datasets. The results suggested that no single classifier consistently outperformed the others across all datasets and metrics for both methods. Classifiers based on the NPI approach had high performance compared to the classifiers based on the IDM across most datasets in both methods, particularly, the NPI-M and A-NPI-M classifiers. Expanding this research to include algorithm adaptation methods for multi-label classification is a topic for future research. An interesting direction for further research would be studying in-depth the scenario when an instance associated with a subset of class label and the classifier predicts a set of subset of class labels.

Chapter 5 presented a new use of performance evaluation measures for NPI-based methods for bivariate data. This method is called the semi-parametric predictive method

(NPI-C) and introduced by Coolen-Maturi et al. [29]. In [29, 85], a simulations study was conducted to evaluate the performance of the NPI-C method, where intervals of proportions, $[l, q]$, using NPI lower and upper probabilities were produced corresponding to a quantile level q. The aim of this chapter was to evaluate the performance of the NPI-C method, investigating if the interval $[l, u]$ was still informative when it did not contain the value q, taking into account the level of imprecision $u - l$. The measures are introduced to focus on the accuracy and precision of the predictions. In this chapter, a simulation study was conducted to evaluate the performance of NPI-C under different scenarios, varying the copulas used for data generation and inference—whether they were the same or different—using the measures introduced earlier. Further study was presented to use the measures for comparative evaluation with a method introduced by Muhammad [85]. The results of this chapter revealed that the intervals obtained with smaller sample sizes tended to be more accurate than those obtained with larger sample sizes. However, when a larger sample size was used, the intervals were more precise. The measures in this study included loss functions and interval scores. When applying loss functions, the results showed that when the value q was observed outside the interval, it is tended to be near the interval. When using the interval score, the predictions were evaluated in terms of both the accuracy and precision, however, selecting the weights for the measure was important according to the specific application. Future investigations could focus on studying the performance of the other NPI-based methods using these measures.

Further topics will be interesting for future research. For example, in the D-NPI method introduced for multinomial data, the NPI lower and upper probabilities for the event that a future instance has a specific class were derived based on NPI for Bernoulli data. An interesting direction for further research would be using NPI for multinomial data instead to provide insights when used for imprecise classification.

When assigning a precise probability or probability interval to each class label, applying a choice function to obtain either a single prediction or a predicted set of class labels is a further topic left for future research. These choice functions could include Γ-maximax, Threshold-Based Decision Rule, Interval dominance, and E-admissibility (see Augustin et al. [16]).

Furthermore, exploring classifiers in classical classification, imprecise classification, and multi-label classification problems while taking into account ordinal class variable is

another possible direction. For example, if a class variable has values $\{1, 2, 3\}$ and an instance with a class label 1. If a classical classifier predicts the class label to be 2, how can this prediction be assessed? This topic is also left for future research.

Appendix A

Extra material for Chapter 1

A.1 Pseudocode of the D-NPI algorithm

In this section, the pseudocode of the D-NPI algorithm according to the steps introduced by Alharbi [12] is presented in A.1. The notations used in Alharbi's work [12] are adapted to be consistent with that used in this thesis. For binary data, the D-NPI for binary data is used for calculating the NPI lower and upper probabilities as presented in Section 1.5.1. For multinomial data, the NPI lower and upper probabilities are derived based on the D-NPI for multinomial data as described in Section 1.5.2.

Algorithm A.1 Pseudocode of the D-NPI algorithm.

Input: Dataset ($\mathcal{D}$), Attributes (Att), Class Variable (C)
Create a Root node for the tree
  if $\mathcal{D}$ have the same class label **then**
  Return a single-node tree with the class label
  if Att is empty **then**
  Return the single-node tree with the most common class label in $\mathcal{D}$
  Otherwise
  foreach attribute, x in Att **do**
    Calculate $\underline{P}_x(CI)$ and $\overline{P}_x(CI)$
    Choose the attribute with the highest NPI lower and upper probabilities for CI, $\underline{P}_x(CI)$ and $\overline{P}_x(CI)$
    if $\underline{P}_x(CI) > \underline{P}(NA)$ and $\overline{P}_x(CI) > \overline{P}(NA)$ **then**
      Choose the attribute, x for Root $(*)$
    else
      Return a leaf node labelled with the most common class label in $\mathcal{D}$
    Create Root with attribute x
    foreach value of x, v_j, **do**
      Create a branch below Root, with $x = v_j$
      Create a subset of $\mathcal{D}$ based on $x = v_j$, referred as $\mathcal{D}_{x=v_j}$
      if $\mathcal{D}_{x=v_j}$ is empty **then**
        Add a leaf node labelled with the most common class in $\mathcal{D}$
      else
        Add the subset via D-NPI using the $\mathcal{D}_{x=v_j}$ as the training dataset
Return Root
$(*)$ In case of ties, see Section 1.5.

A.2 Example

In this section, an illustrative example is presented in order to show how the D-NPI algorithm works.

Example A.1. In this illustrative example, the process of building a classification tree using the D-NPI for multinomial data is presented. Suppose a small artificial dataset consists of 12 instances, as presented in Table A.1. This dataset includes four attribute variables and a class variable, which contains three class labels. The first step in constructing the tree is to verify whether the dataset contains the same class label; if so, a leaf node is returned with the class label. This step is repeated throughout the branching process. Subsequently, the NPI lower and upper probabilities for each attribute for CI are derived using Equations (1.25) and (1.27) in Section 1.5.1.

The resulting NPI lower and upper probabilities for each attribute for CI are presented

x_1	x_2	x_3	x_4	Class
3	2	1	2	3
3	2	1	1	2
3	2	1	2	3
2	3	1	2	1
2	1	2	2	3
2	3	2	1	2
1	1	2	1	1
3	3	1	2	3
3	1	2	2	3
2	3	2	2	1
3	3	2	1	3
1	2	1	1	1

Table A.1: Dataset description for Example A.1.

x_i	$\underline{P}_i(CI)$	$\overline{P}_i(CI)$
x_1	0.5754	0.8286
x_2	0.3833	0.6167
x_3	0.4286	0.5714
x_4	0.4792	0.6667
NA	0.4167	0.5833

Table A.2: CI intervals for all attributes.

in Table A.2. To decide which attribute to select for splitting, the NPI lower and upper probabilities for NA are calculated using the Equations (1.28) and (1.29) in Section 1.5.1. Then, the following values are obtained: $\underline{P}(NA) = \max(0, \frac{6-1}{12}) = 0.4167$ and $\overline{P}(NA) = \min(\frac{6+1}{n}, 1) = 0.5833$. By comparing the lower and upper probabilities for each attribute for CI with the probabilities [0.4167, 0.5833], as shown in Table A.2, and according to the conditions defined in (1.30) in Section 1.5.1, attributes x_1 and x_4 satisfy the conditions. Next, by comparing CI intervals of x_1 and x_4, attribute x_1 achieves the highest CI interval. Therefore, attribute x_1 is selected to split the tree and assigned to the root node. Subsequently, the dataset is partitioned into three subsets according to the values of attribute x_1.

After partitioning the dataset, three branches are obtained $x_1 = 1$, $x_1 = 2$, and $x_1 = 3$. For the branch $x_1 = 1$, the resulting subset contains all instances with the same class label 1. In this case, a leaf node is created, and the class label 1 is assigned to it. For the branch $x_1 = 2$, the NPI lower and upper probabilities for each attribute (excluding the previously used splitting attribute x_1) are calculated, and the NPI lower and upper probabilities for

x_i	$\underline{P}_i(CI)$	$\overline{P}_i(CI)$
x_2	0.5000	0.8750
x_3	0.2500	0.7500
x_4	0.5000	0.8750
NA	0.2500	0.7500

Table A.3: CI intervals for all attributes; the branch $x_1 = 2$.

x_i	$\underline{P}_i(CI)$	$\overline{P}_i(CI)$
x_2	0.5278	0.9170
x_3	0.6111	0.9000
x_4	0.5667	0.9444
NA	0.6667	1.0000

Table A.4: CI intervals for all attributes; the branch $x_1 = 3$.

NA is derived. The results are presented in Table A.3. After comparing the calculated probabilities, attributes x_2 and x_4 satisfy the conditions in (1.30) in Section 1.5.1. Since both attributes have identical lower probabilities and identical upper probabilities, either one can be selected to split the tree. For the branch $x_1 = 3$, the resulting probabilities are presented in Table A.4. Upon comparison, no attribute satisfies both conditions in (1.30) in Section 1.5.1. Therefore, a leaf node is created and assigned the most frequent class label, which is 3.

Below the branch $x_1 = 2$ a node is created with the selected attribute x_2, and three branches are created with the attribute values $x_2 = 1$, $x_2 = 2$, and $x_2 = 3$. There is no further splits below these branches. For the first branch, the subset contains only one instance, and the corresponding class label 3 is assigned to a leaf node. In the second branch, the resulting subset contains no instances, so a leaf node is created and assigned the most frequent class label from the dataset on which the split was performed, which is 1. For the third branch, no attribute satisfies the conditions in (1.30) in Section 1.5.1, and a leaf node is created with the most frequent label 1. The classification tree is illustrated in Figure A.1.

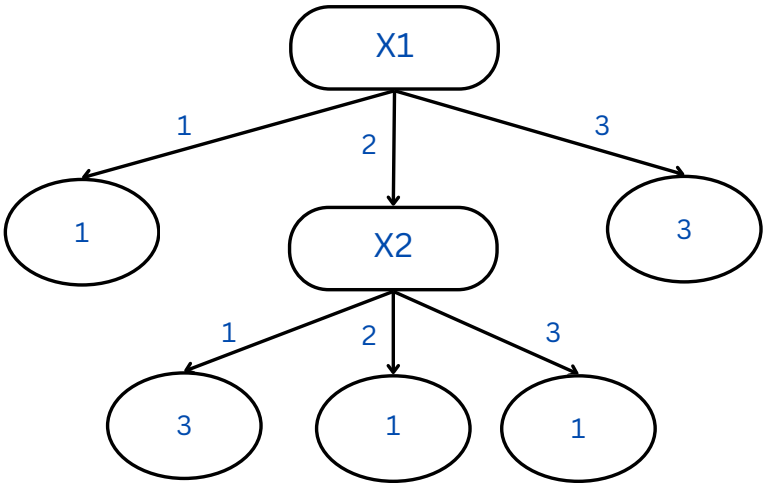


Figure A.1: A structure of classification tree Example A.1.

Appendix B

Extra material for Chapter 2

This appendix presents an additional example for Chapter 2. In this example, a detailed analysis of the performance of the three algorithms is conducted for the Soybean-Large dataset. The Soybean-Large dataset comprises 638 instances and includes 35 nominal attributes and 19 class labels. The missing values in this dataset are replaced by the modal values of the corresponding attribute. This dataset is selected due to the difference in classification accuracy between the NPI-RF method and the other classifiers, as well as its nominal attributes, which eliminates the need for discretisation.

In the experiment in Section 2.4, a 10-fold cross validation procedure was used to evaluate the performance of the classifiers for this dataset. In this example, one case from these folds is randomly selected to illustrate the differences between the classifiers. To ensure a consistent comparison among the classifiers, the same fold is used for all classifiers.

In order to study the complexity of the individual classification trees, Tree Depth (TD) measure is used. Table B.1 presents the classification accuracy for each algorithm for the Soybean-Large dataset. Additionally, Table B.1 shows the TD value for the D-NPI algorithm (e.g., single classification tree) and the average TD values for the individual trees when ensemble methods NPI-Bag and NPI-RF are used. Furthermore, the distribution of trees depths in the NPI-Bag and NPI-RF ensemble methods is analysed to provide insights into the variability and complexity of the individual classification trees in the ensemble methods. Figure B.1 illustrates the resulting TD values for the NPI-Bag and NPI-RF classifiers.

As shown in Table B.1, the accuracy value for the D-NPI classifier coincides with

Algorithm	D-NPI	NPI-Bag	NPI-RF
Accuracy	85.51	84.06	**95.65**
TD	7	6.84	8.73

Table B.1: Classification accuracies and tree depth (TD) for Soybean-Large dataset.

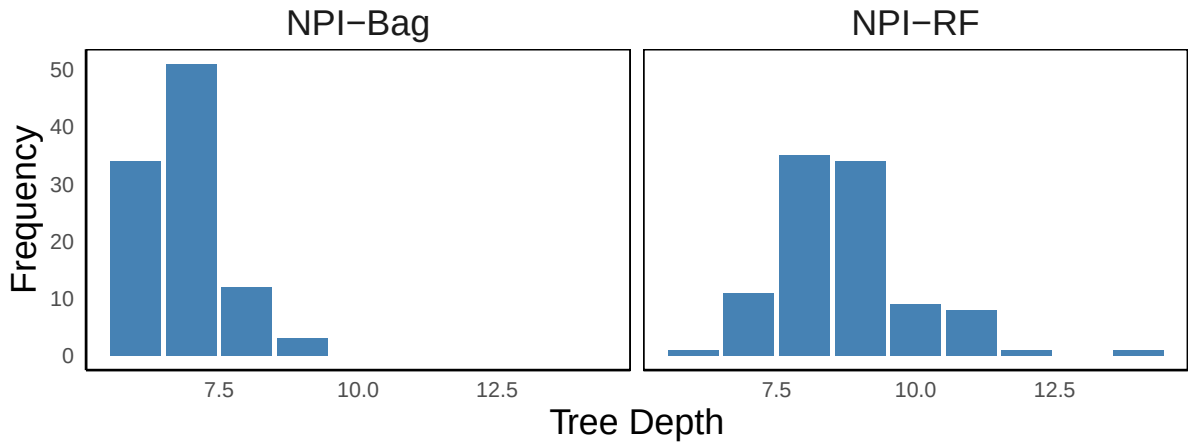


Figure B.1: Bagging ensemble method for Soybean-Large dataset.

the classification tree having low TD value. The NPI-Bag classifier has lower accuracy value than this of the D-NPI classifier. As illustrated in Figure B.1, the NPI-Bag method has predominality shallow individual trees in the ensemble, with almost half of the trees having a TD value of 7, which is the same as the TD value of the D-NPI algorithm. Moreover, more than 30 of the trees have TD value 6. The limited variations in TD values among the individual classification trees produced by the NPI-Bag algorithm, as well as many shallow trees, result in no improvement in capturing complex patterns compared to the D-NPI classifier for this dataset which has relatively a moderate number of attributes.

As illustrated in Figure B.1, compared to the NPI-Bag classifier, the NPI-RF classifier increases the diversity among the individual trees, which can be observed in variation in TD values of the individual trees. This is due to the randomness in the NPI-RF introduced through attributes selection at each split, unlike in the NPI-Bag classifier, which uses all available attributes at each node. Moreover, the NPI-RF classifier produces deeper individual trees, with TD values varying from 6 to 14, which helps in capturing complex patterns in the dataset. This results in the highest accuracy achieved by the NPI-RF classifier compared to the other classifiers, as shown in Table B.1.

Appendix C

Extra material for Chapter 3

C.1 Pseudocodes for the imprecise classifiers

In this section, the pseudocodes outlining the steps of the D-NPI-IC1, D-NPI-IC2, and D-NPI-IC3 algorithms are presented in C.1 to C.3.

Algorithm C.1 Pseudocode of the D-NPI-IC1 algorithm.

Input: Dataset ($\mathcal{D}$), Attributes (Att), Class Variable (C)
Procedure D-NPI-IC1 ($\mathcal{D}, Att, C$)
Create a Root node for the tree
  if all Data have the same class label **then**
  Return a leaf node
  else
  foreach attribute, x_i in Att **do**
    Calculate $\underline{P}_i(CI)$ and $\overline{P}_i(CI)$
    Choose the attribute with the highest NPI lower and upper probabilities for CI, $\underline{P}_i^*(CI)$ and $\overline{P}_i^*(CI)$
    if $\underline{P}_i^*(CI) > \underline{P}(NA)$ and $\overline{P}_i^*(CI) > \overline{P}(NA)$ **then**
      Choose the attribute, x_i $(*)$
    else
    Return a leaf node
    Create a node with attribute x_i
    foreach value of x_i, v_j^i, **do**
    Create a branch for $x_i = v_j^i$
    Create a subset of data based on $x_i = v_j^i$, referred as $\mathcal{D}_{subset}$
    if subset is not empty **then**
    Call D-NPI-IC1 ($\mathcal{D}_{subset}, Att, C$) recursively
    else
    Return a leaf node
Return Root

$(*)$ In case of ties, see Section 3.3.

Algorithm C.2 Pseudocode of the D-NPI-IC2 algorithm.

Input: Dataset ($\mathcal{D}$), Attributes (Att), Class Variable (C)
Procedure D-NPI-IC2 ($\mathcal{D}, Att, C$)
Create a Root node for the tree
  if all Data have the same class label **then**
  Return a leaf node
  else
  foreach attribute, x_i in Att **do**
    Calculate $\underline{P}_i(CI)$ and $\overline{P}_i(CI)$
    Choose the attribute with the highest NPI lower and upper probabilities for CI, $\underline{P}_i^*(CI)$ and $\overline{P}_i^*(CI)$
    if $\underline{P}_i^*(CI) > \underline{P}(NA)$ and $\overline{P}_i^*(CI) > \overline{P}(NA)$ **then**
      Choose the attribute, x_i $(*)$
    else
    Return a leaf node
    Create a node with attribute x_i
    foreach value of x_i, v_j^i, **do**
    Create a branch for $x_i = v_j^i$
    Create a subset of data based on $x_i = v_j^i$, referred as $\mathcal{D}_{subset}$
    if subset is not empty **then**
    Call D-NPI-IC2 ($\mathcal{D}_{subset}, Att, C$) recursively
    else
    Return a leaf node
Return Root

$(*)$ In case of ties, see Section 3.3.

Algorithm C.3 Pseudocode of the D-NPI-IC3 algorithm.

Input: Dataset ($\mathcal{D}$), Attributes (Att), Class Variable (C)
Procedure D-NPI-IC3 ($\mathcal{D}, Att, C$)
Create a Root node for the tree
  if all Data have the same class label **then**
  Return a leaf node
  else
  foreach attribute, x_i in Att **do**
    Calculate $\underline{P}_i(CI)$ and $\overline{P}_i(CI)$
    Choose the attribute with the highest NPI lower and upper probabilities for CI, $\underline{P}_i^*(CI)$ and $\overline{P}_i^*(CI)$
    if $\overline{P}_i^*(CI) > \overline{P}(NA)$ **then**
      Choose the attribute, x_i $(*)$
    else
    Return a leaf node
    Create a node with attribute x_i
    foreach value of x_i, v_j^i, **do**
    Create a branch for $x_i = v_j^i$
    Create a subset of data based on $x_i = v_j^i$, referred as $\mathcal{D}_{subset}$
    if subset is not empty **then**
    Call D-NPI-IC3 ($\mathcal{D}_{subset}, Att, C$) recursively
    else
    Return a leaf node
Return Root

$(*)$ In case of ties, see Section 3.3.

C.2 Additional results

This section presents the results for each imprecise classifier presented in Chapter 3. Tables C.1 to C.7 present the results for each algorithm, showing their performance across all datasets according to the metrics: Determinacy (DET), Single Accuracy (SingleA), Set Accuracy (SetA), Indeterminacy Size (IS), DACC, MIC, and Tree Depth (TD).

Datasets	**DET**	**SingleA**	**SetA**	**IS**	**DACC**	**MIC**	**TD**
Balance-Scale	0.9937	0.6860	0.5000	2.0000	0.6833	0.9233	3.6
CMC	0.8684	0.5482	0.8912	2.6384	0.5226	0.7625	7.9
Dermatology	0.9645	0.9719	1.0000	6.0000	0.9432	1.6891	6.0
Hayes	0.8438	0.7733	1.0000	2.7283	0.7115	0.8433	3.3
Hypothyroid	0.9995	0.9934	1.0000	4.0000	0.9930	1.3794	4.9
Iris	1.0000	0.9533	-	-	0.9533	1.0730	1.0
Letter	0.7945	0.9019	0.9587	22.6301	0.7330	2.3969	6.9
Lymphography	0.8714	0.8258	1.0000	3.8889	0.7552	1.0740	3.2
Nursery	0.9976	0.9198	1.0000	4.3214	0.9182	1.5094	7.0
Page-Blocks	0.9889	0.9732	0.9833	4.4143	0.9652	1.5616	4.4
Pendigits	0.9140	0.9351	0.9693	8.9428	0.8665	1.9978	5.6
Primary-Tumor	0.6022	0.5549	0.7786	15.1531	0.3637	1.1499	12.2
Segment	0.9554	0.9618	1.0000	6.2471	0.9279	1.8089	5.4
Soybean-Large	0.9210	0.8778	0.9183	14.6674	0.8192	2.4354	7.4
Splice	0.9721	0.8994	0.9917	2.7304	0.8847	1.0175	7.0
Sponge	1.0000	0.9232	-	-	0.9232	1.0564	1.8
Vehicle	0.8901	0.7492	0.9632	3.0353	0.7058	1.0641	6.6
Vowel	0.8535	0.8203	0.9498	7.1826	0.7351	1.8109	6.7
Waveform	0.9766	0.7764	0.9676	2.6887	0.7669	0.9561	8.8
Yeast	0.9266	0.6139	0.9096	6.5777	0.5848	1.4425	7.0
Zoo	0.9400	0.9409	1.0000	4.5000	0.9038	1.7834	3.1

Table C.1: Results of different measures for the imprecise classifier D-NPI-IC1.

Datasets	**DET**	**SingleA**	**SetA**	**IS**	**DACC**	**MIC**	**TD**
Balance-Scale	0.9920	0.6922	0.5000	2.3333	0.6884	0.9244	3.7
CMC	0.9002	0.5682	0.8601	2.6359	0.5451	0.7957	8.0
Dermatology	0.9535	0.9576	1.0000	6.0000	0.9204	1.6499	6.0
Hayes	0.8312	0.7611	1.0000	2.6810	0.6927	0.8278	3.6
Hypothyroid	0.9995	0.9934	1.0000	4.0000	0.9930	1.3794	4.9
Iris	1.0000	0.9400	-	-	0.9400	1.0657	1.0
Letter	0.8202	0.8843	0.9412	21.9834	0.7404	2.4261	7.4
Lymphography	0.9186	0.8274	1.0000	3.6000	0.7811	1.1368	4.0
Nursery	0.9976	0.9197	1.0000	4.3214	0.9181	1.5093	7.0
Page-Blocks	0.9923	0.9709	0.9667	4.6250	0.9651	1.5628	4.8
Pendigits	0.9251	0.9306	0.9674	9.0430	0.8707	2.0097	5.7
Primary-Tumor	0.6668	0.4830	0.7952	15.5294	0.3457	1.1120	12.2
Segment	0.9732	0.9648	1.0000	6.7857	0.9433	1.8399	5.7
Soybean-Large	0.9459	0.8793	0.9630	14.2778	0.8395	2.4941	7.9
Splice	0.9715	0.8964	1.0000	2.8196	0.8812	1.0142	7.1
Sponge	1.0000	0.9232	-	-	0.9232	1.0564	1.8
Vehicle	0.9150	0.7468	1.0000	3.2863	0.7112	1.0752	6.9
Vowel	0.8667	0.8216	0.9705	7.2399	0.7441	1.8311	6.4
Waveform	0.9800	0.7745	0.9642	2.7336	0.7661	0.9575	8.8
Yeast	0.9286	0.6162	0.8737	6.4892	0.5865	1.4457	7.7
Zoo	0.9800	0.8918	1.0000	7.0000	0.8747	1.7316	3.8

Table C.2: Results of different measures for the imprecise classifier D-NPI-IC2.

Datasets	**DET**	**SingleA**	**SetA**	**IS**	**DACC**	**MIC**	**TD**
Balance-Scale	0.8974	0.7616	0.7677	2.1324	0.7193	0.9085	4.0
CMC	0.6599	0.5602	0.8880	2.6131	0.4875	0.6289	8.5
Dermatology	0.9591	0.9689	1.0000	6.0000	0.9359	1.6755	6.9
Hayes	0.6750	0.8278	0.9900	2.6392	0.6833	0.7304	4.0
Hypothyroid	0.9984	0.9934	1.0000	4.0000	0.9922	1.3780	5.0
Iris	1.0000	0.9533	-	-	0.9533	1.0730	1.0
Letter	0.7745	0.9113	0.9584	22.5464	0.7240	2.3674	6.9
Lymphography	0.8176	0.8351	1.0000	3.7500	0.7338	1.0264	3.5
Nursery	0.9811	0.9391	1.0000	4.4461	0.9262	1.5101	7.0
Page-Blocks	0.9777	0.9770	0.9747	4.1290	0.9614	1.5522	5.6
Pendigits	0.8856	0.9459	0.9754	8.9186	0.8537	1.9617	6.0
Primary-Tumor	0.5048	0.5633	0.7749	15.0844	0.3180	1.0141	14.7
Segment	0.9411	0.9607	0.9938	5.9710	0.9164	1.7850	6.9
Soybean-Large	0.8843	0.8794	0.9348	13.4594	0.7975	2.3803	8.1
Splice	0.9461	0.9139	0.9958	2.8043	0.8841	0.9988	8.0
Sponge	0.9357	0.9321	1.0000	3.0000	0.8911	0.9917	1.8
Vehicle	0.8499	0.7497	0.9384	2.9911	0.6899	1.0336	8.1
Vowel	0.8232	0.8335	0.95696	6.9969	0.7299	1.7990	6.9
Waveform	0.9338	0.7743	0.9845	2.6433	0.7488	0.9202	9.3
Yeast	0.8323	0.6301	0.8957	5.7492	0.5667	1.4074	7.7
Zoo	0.9400	0.94091	1.0000	4.5000	0.9038	1.7834	3.1

Table C.3: Results of different measures for the imprecise classifier D-NPI-IC3.

Datasets	**DET**	**SingleA**	**SetA**	**IS**	**DACC**	**MIC**	**TD**
Balance-Scale	0.7200	0.8573	0.8315	2.0786	0.7241	0.8426	4.0
CMC	0.7800	0.5974	0.8689	2.4537	0.5442	0.7385	6.8
Dermatology	0.9482	0.9287	1.0000	5.7407	0.8903	1.6069	4.6
Hayes	0.6875	0.8055	0.8785	2.2802	0.6781	0.7858	2.9
Hypothyroid	0.9928	0.9931	1.0000	3.6963	0.9881	1.3709	5.7
Iris	0.9867	0.9523	1.0000	3.0000	0.9444	1.0583	1.2
Letter	0.8794	0.5393	0.6627	6.0256	0.5016	1.7496	4.9
Lymphography	0.8495	0.8738	1.0000	3.9375	0.7807	1.0824	5.0
Nursery	0.9931	0.9518	1.0000	3.8006	0.9475	1.5432	7.0
Page-Blocks	0.9832	0.9676	0.9486	3.2233	0.9572	1.5525	3.0
Pendigits	0.9209	0.8472	0.9270	6.0096	0.8000	1.8871	4.0
Primary-Tumor	0.5545	0.5324	0.8053	12.3073	0.3488	1.1904	10.3
Segment	0.9139	0.8196	0.9338	4.1707	0.7759	1.5682	4.0
Soybean-Large	0.9429	0.9407	0.9619	11.3005	0.9005	2.6747	6.9
Splice	0.8082	0.7763	0.9433	2.7192	0.6969	0.8136	6.7
Sponge	1.0000	0.9232	-	-	0.9232	1.0564	1.9
Vehicle	0.8167	0.6745	0.8629	2.5347	0.6145	0.9768	5.0
Vowel	0.6586	0.6599	0.8265	4.4175	0.5325	1.4269	6.1
Waveform	0.9286	0.7819	0.9951	2.6190	0.7546	0.9212	8.3
Yeast	0.9420	0.5899	0.8329	6.0561	0.5678	1.4122	5.9
Zoo	0.9000	0.9584	1.0000	7.0000	0.8752	1.6879	3.8

Table C.4: Results of different measures for the imprecise classifier NPIM.

Datasets	**DET**	**SingleA**	**SetA**	**IS**	**DACC**	**MIC**	**TD**
Balance-Scale	0.7200	0.8573	0.8315	2.0786	0.7241	0.8426	4.0
CMC	0.7800	0.5974	0.8689	2.4537	0.5442	0.7385	6.8
Dermatology	0.9482	0.9287	1.0000	5.7407	0.8903	1.6069	4.6
Hayes	0.6875	0.8055	0.8785	2.2802	0.6781	0.7858	2.9
Hypothyroid	0.9928	0.9931	1.0000	3.6963	0.9881	1.3709	5.7
Iris	0.9867	0.9523	1.0000	3.0000	0.9444	1.0583	1.2
Letter	0.8807	0.5392	0.6664	6.0816	0.5021	1.7505	5.0
Lymphography	0.8495	0.8738	1.0000	3.9375	0.7807	1.0824	5.0
Nursery	0.9934	0.9508	1.0000	4.0087	0.9465	1.5417	7.0
Page-Blocks	0.9832	0.9676	0.9486	3.2233	0.9572	1.5525	3.0
Pendigits	0.9204	0.8473	0.9287	6.0384	0.7998	1.8865	4.0
Primary-Tumor	0.5426	0.5361	0.8283	12.6554	0.3466	1.1825	10
Segment	0.9147	0.8188	0.9338	4.1788	0.7756	1.5679	4.0
Soybean-Large	0.9429	0.9407	0.9619	11.3005	0.9005	2.6747	6.9
Splice	0.8082	0.7763	0.9433	2.7192	0.6969	0.8136	6.7
Sponge	1.0000	0.9232	-	-	0.9232	1.0564	1.9
Vehicle	0.8167	0.6745	0.8629	2.5347	0.6145	0.9768	5.0
Vowel	0.6636	0.6558	0.8366	4.5717	0.5306	1.4199	6.0
Waveform	0.9286	0.7819	0.9951	2.6190	0.7546	0.9212	8.3
Yeast	0.9420	0.5899	0.8238	6.0561	0.5676	1.4115	5.9
Zoo	0.9000	0.9584	1.0000	7.0000	0.8752	1.6879	3.8

Table C.5: Results of different measures for the imprecise classifier ANPIM.

Datasets	**DET**	**SingleA**	**SetA**	**IS**	**DACC**	**MIC**	**TD**
Balance-Scale	0.8128	0.7829	0.8057	2.1230	0.7017	0.8694	4.0
CMC	0.8391	0.5781	0.8550	2.4492	0.5416	0.7693	7.6
Dermatology	0.9510	0.9289	1.0000	4.1481	0.8981	1.6299	4.9
Hayes	0.7250	0.7730	0.8735	2.2984	0.6729	0.8039	3.0
Hypothyroid	0.9952	0.9928	1.0000	3.0750	0.9897	1.3742	6.6
Iris	1.0000	0.9533	-	-	0.9533	1.0730	1.1
Letter	0.8847	0.5455	0.6069	4.9179	0.5081	1.7691	5.0
Lymphography	0.9119	0.8469	1.0000	3.9048	0.7932	1.1372	4.8
Nursery	0.9950	0.9651	1.0000	2.5081	0.9624	1.5629	7.0
Page-Blocks	0.9876	0.9674	0.9300	3.1159	0.9601	1.5580	3.0
Pendigits	0.9450	0.8537	0.8902	5.0277	0.8219	1.9382	4.0
Primary-Tumor	0.5956	0.5187	0.7580	12.2524	0.3564	1.2036	10.4
Segment	0.9416	0.8124	0.9206	3.3288	0.7864	1.5962	4.0
Soybean-Large	0.9531	0.9297	1.0000	6.6278	0.9033	2.6949	6.0
Splice	0.8511	0.7596	0.9506	2.7274	0.7016	0.8417	6.9
Sponge	1.0000	0.9232	-	-	0.9232	1.0564	1.9
Vehicle	0.8653	0.6586	0.9030	2.3765	0.6219	1.0001	5.9
Vowel	0.7404	0.6600	0.8068	3.3853	0.5729	1.5268	6.0
Waveform	0.9430	0.7776	0.9955	2.6853	0.7554	0.9287	9.1
Yeast	0.8855	0.6069	0.8360	4.3227	0.5674	1.4240	6.1
Zoo	0.9700	0.9587	1.0000	7.0000	0.9352	1.8241	3.9

Table C.6: Results of different measures for the imprecise classifier IDM1.

Datasets	**DET**	**SingleA**	**SetA**	**IS**	**DACC**	**MIC**	**TD**
Balance-Scale	0.7200	0.8573	0.8315	2.0786	0.7241	0.8426	4.0
CMC	0.8072	0.5939	0.8590	2.5362	0.5455	0.7492	6.9
Dermatology	0.9510	0.9203	1.0000	5.7778	0.8835	1.5972	4.2
Hayes	0.6938	0.8387	0.8948	2.2643	0.7021	0.7970	3.1
Hypothyroid	0.9936	0.9931	1.0000	3.4375	0.9888	1.3724	6.7
Iris	1.0000	0.9533	-	-	0.9533	1.0730	1.1
Letter	0.8485	0.5510	0.6810	6.5084	0.5011	1.7608	5.0
Lymphography	0.8171	0.8746	1.0000	2.9750	0.7742	1.1008	3.9
Nursery	0.9994	0.9378	1.0000	2.0000	0.9376	1.5340	6.0
Page-Blocks	0.9845	0.9642	0.9183	3.0258	0.9551	1.5512	3.0
Pendigits	0.9284	0.8405	0.9249	6.0259	0.7980	1.8834	4.0
Primary-Tumor	0.5102	0.5721	0.7464	10.8792	0.3623	1.2519	9.7
Segment	0.9152	0.8131	0.9374	3.7702	0.7732	1.5684	4.0
Soybean-Large	0.8375	0.9243	0.9755	5.3039	0.8376	2.5708	5.4
Splice	0.8458	0.7608	0.9539	2.8124	0.6969	0.8313	6.8
Sponge	1.0000	0.9232	-	-	0.9232	1.0564	1.8
Vehicle	0.8085	0.6657	0.8582	2.5092	0.6047	0.9666	5.8
Vowel	0.6879	0.6341	0.8342	4.5133	0.5262	1.4095	5.5
Waveform	0.9544	0.7650	0.9854	2.6436	0.7483	0.9336	7.7
Yeast	0.9279	0.5876	0.7866	5.3211	0.5610	1.3997	5.1
Zoo	0.8909	0.9475	1.0000	7.0000	0.8574	1.6540	3.9

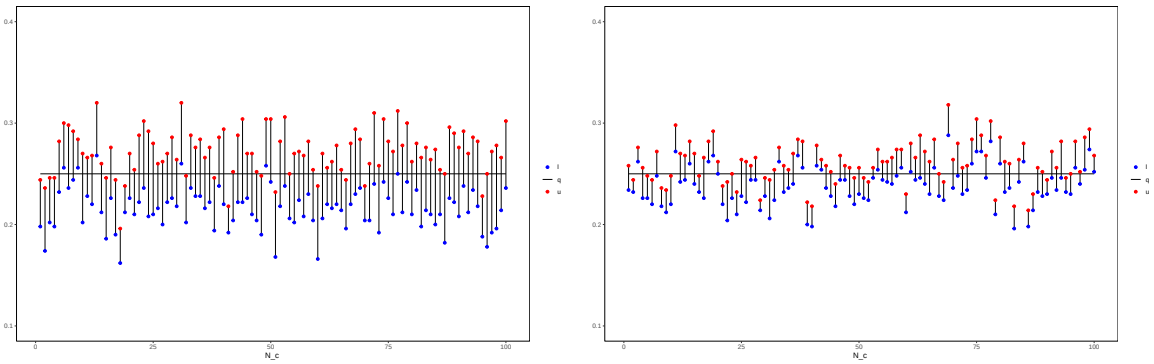
Table C.7: Results of different measures for the imprecise classifier IDM2.

Appendix D

Extra material for Chapter 5

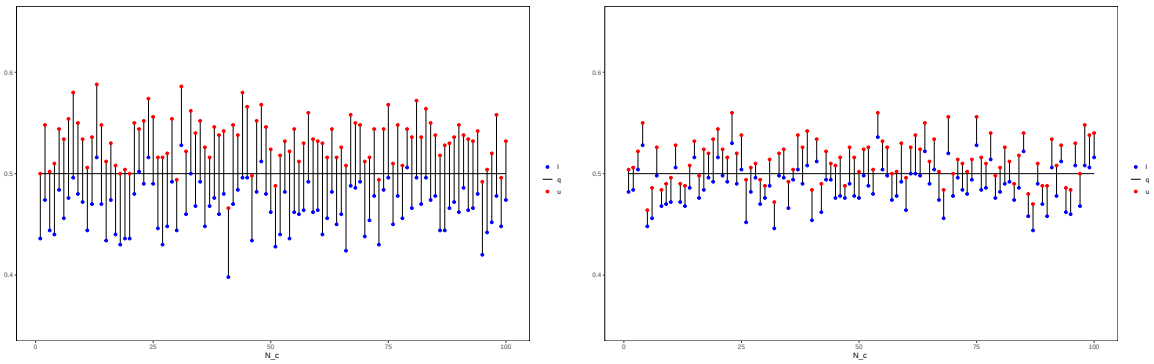
D.1 Illustrations of the intervals $[l_k, u_k]$

This section includes the intervals $[l_k, u_k]$ for $k = 1, ..., 100$ obtained from the simulation study. The results are presented in Figures D.1 to D.9.



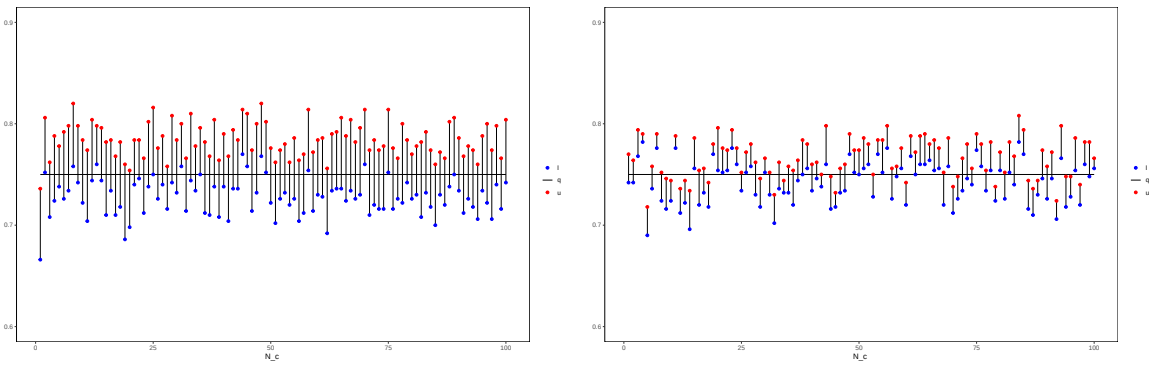
(a) $q = 0.25$, $n = 20$

(b) $q = 0.25$, $n = 50$



(c) $q = 0.5$, $n = 20$

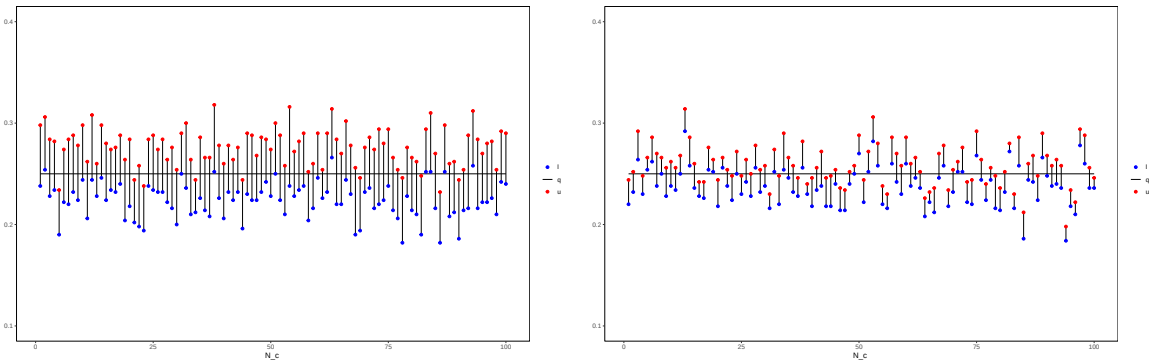
(d) $q = 0.5$, $n = 50$



(e) $q = 0.75$, $n = 20$

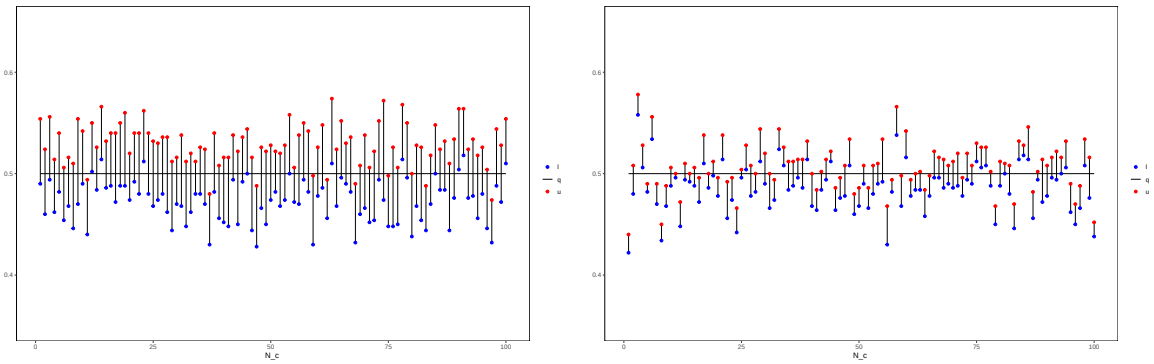
(f) $q = 0.75$, $n = 50$

Figure D.1: Intervals $[l_k, u_k]$ where $k = 1, ..., 100$; Normal copula is for both simulation and inference, $\tau = 0.25$.



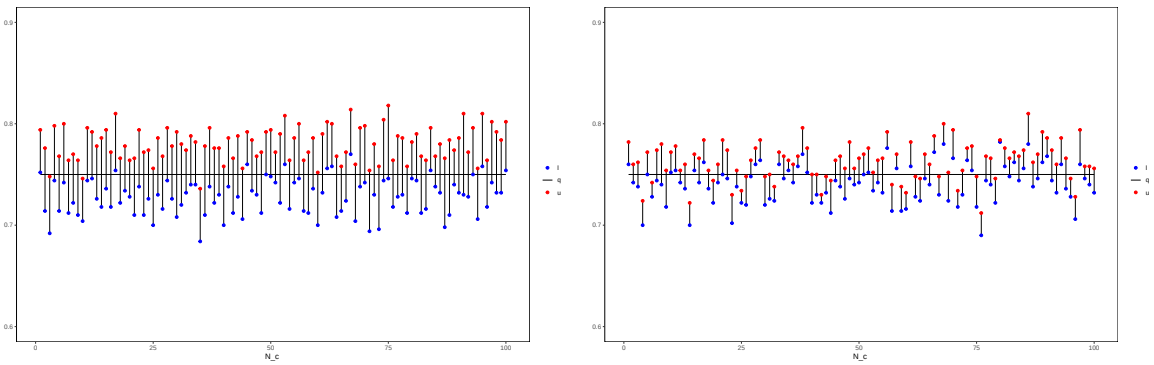
(a) $q = 0.25$, $n = 20$

(b) $q = 0.25$, $n = 50$



(c) $q = 0.5$, $n = 20$

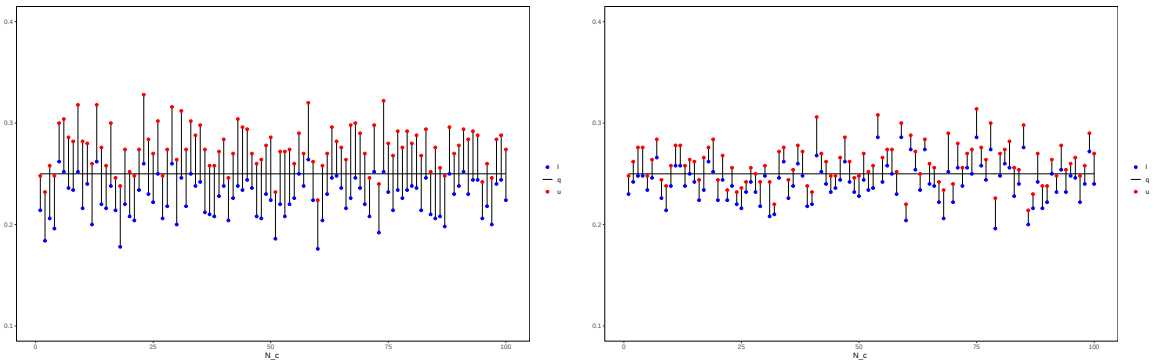
(d) $q = 0.5$, $n = 50$



(e) $q = 0.75$, $n = 20$

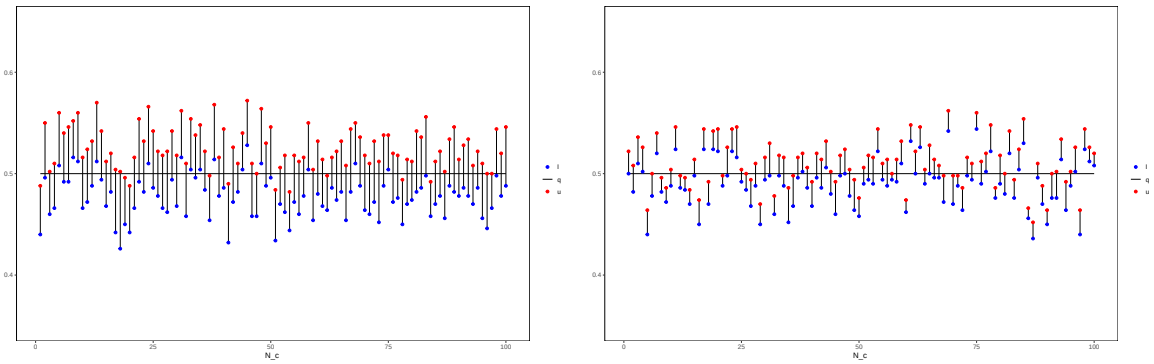
(f) $q = 0.75$, $n = 50$

Figure D.2: Intervals $[l_k, u_k]$ where $k = 1, ..., 100$; Normal copula is for both simulation and inference, $\tau = 0.5$.



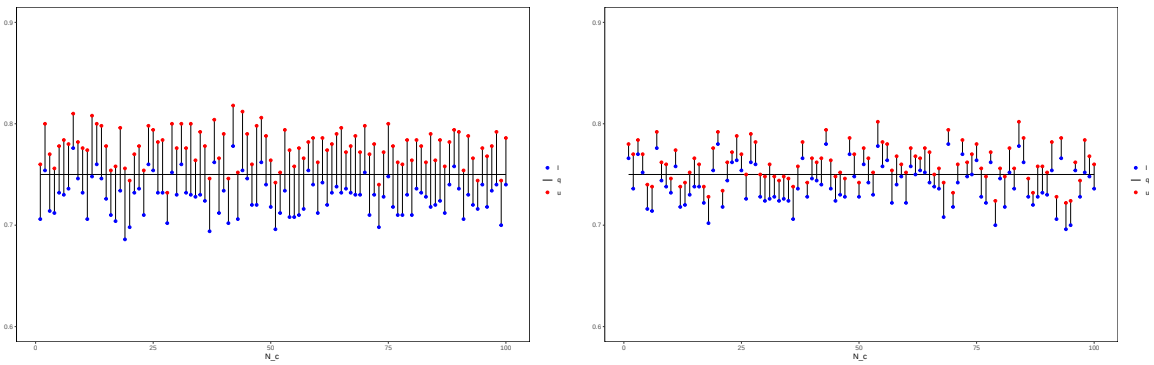
(a) $q = 0.25$, $n = 20$

(b) $q = 0.25$, $n = 50$



(c) $q = 0.5$, $n = 20$

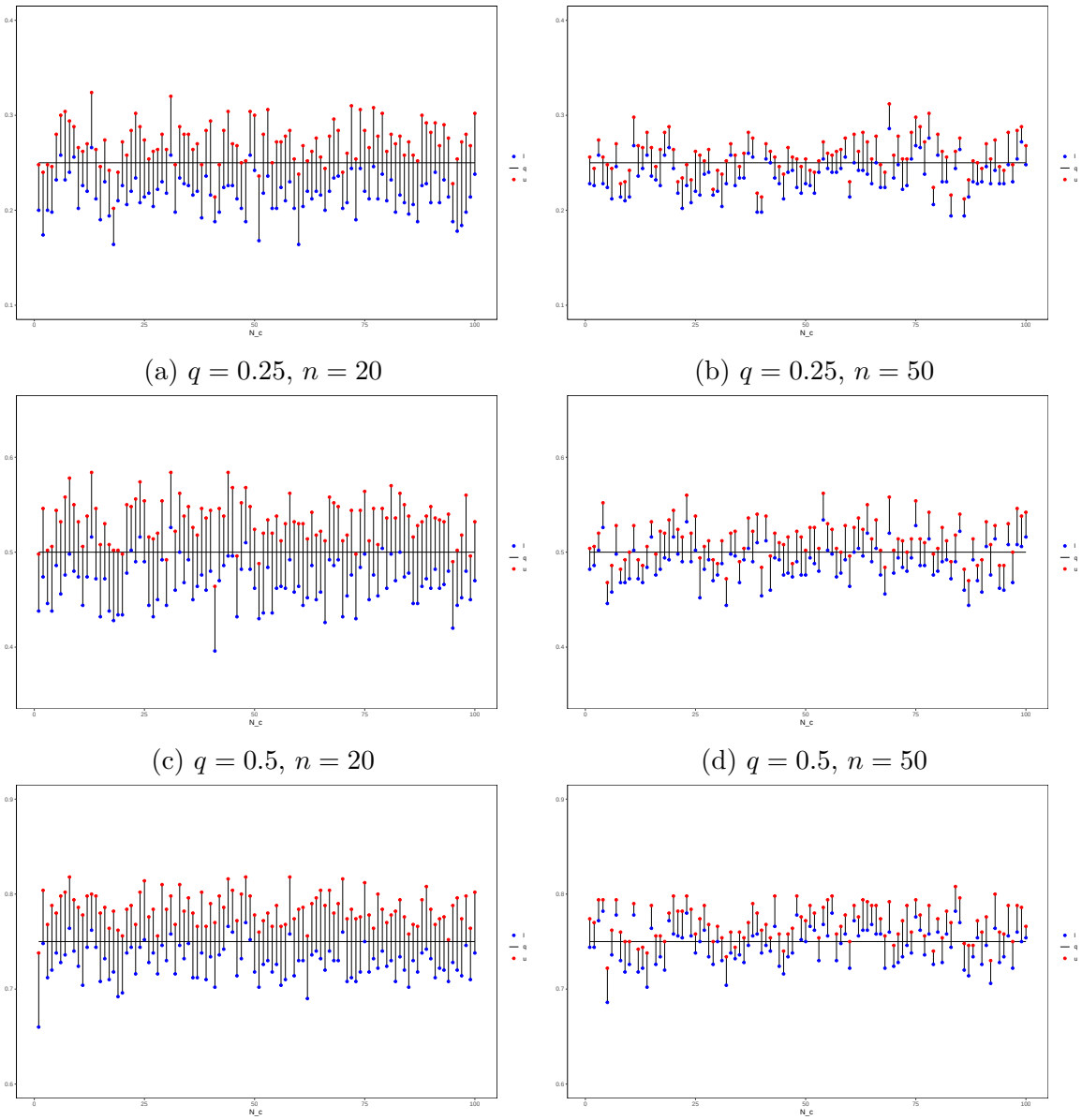
(d) $q = 0.5$, $n = 50$



(e) $q = 0.75$, $n = 20$

(f) $q = 0.75$, $n = 50$

Figure D.3: Intervals $[l_k, u_k]$ where $k = 1, ..., 100$; Normal copula is for both simulation and inference, $\tau = 0.75$.



(a) $q = 0.25$, $n = 20$

(b) $q = 0.25$, $n = 50$

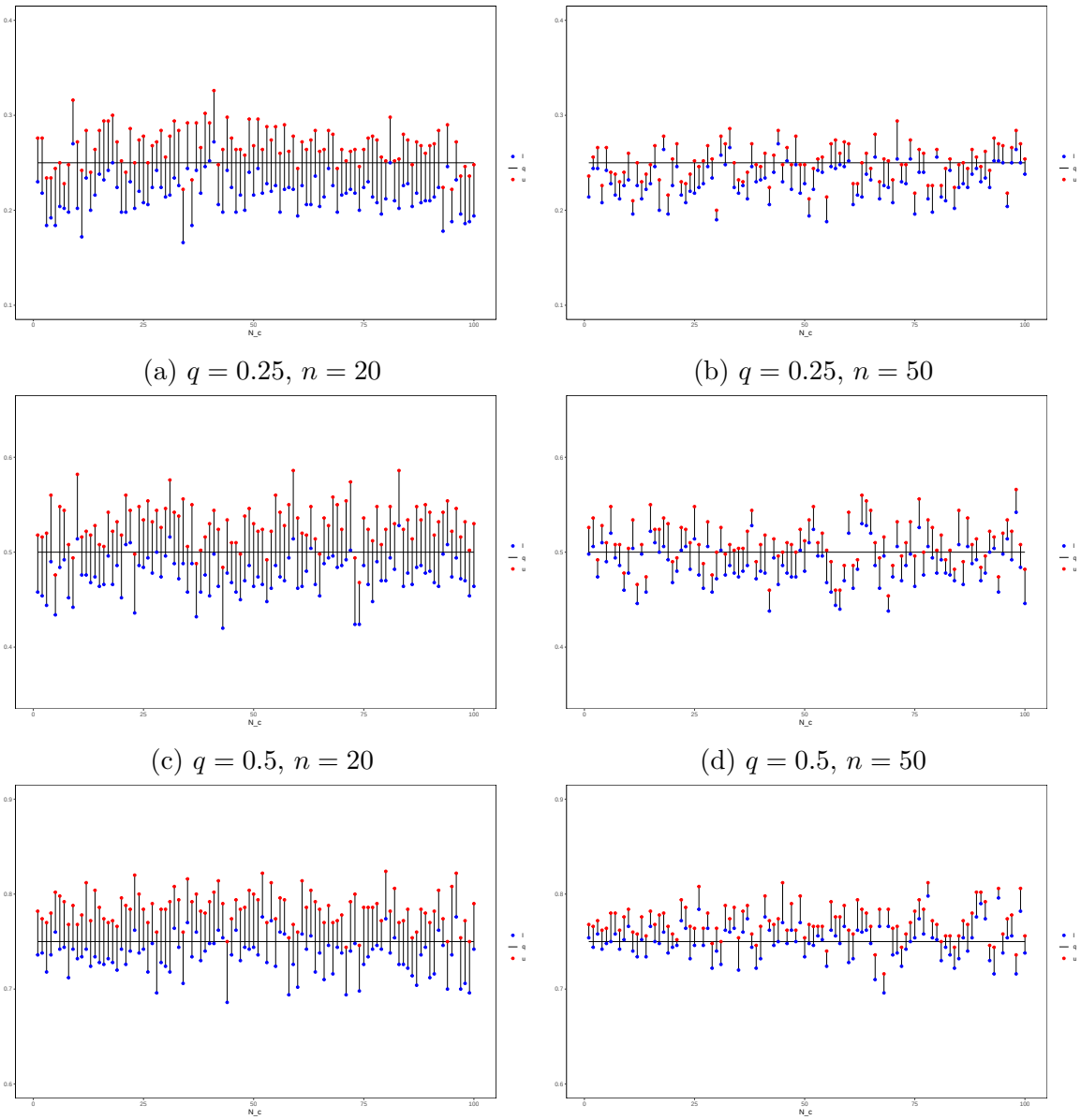
(c) $q = 0.5$, $n = 20$

(d) $q = 0.5$, $n = 50$

(e) $q = 0.75$, $n = 20$

(f) $q = 0.75$, $n = 50$

Figure D.4: Intervals $[l_k, u_k]$ where $k = 1, ..., 100$; Normal copula is for simulation and Frank copula is for inference, $\tau = 0.25$.



(a) $q = 0.25$, $n = 20$

(b) $q = 0.25$, $n = 50$

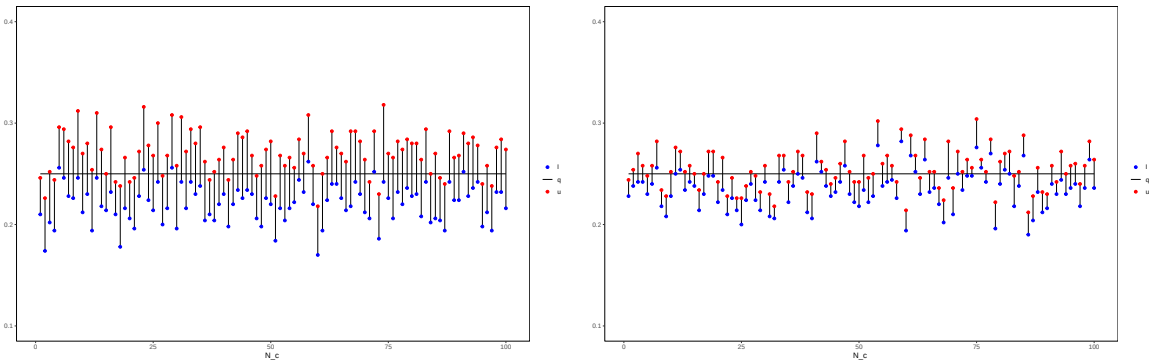
(c) $q = 0.5$, $n = 20$

(d) $q = 0.5$, $n = 50$

(e) $q = 0.75$, $n = 20$

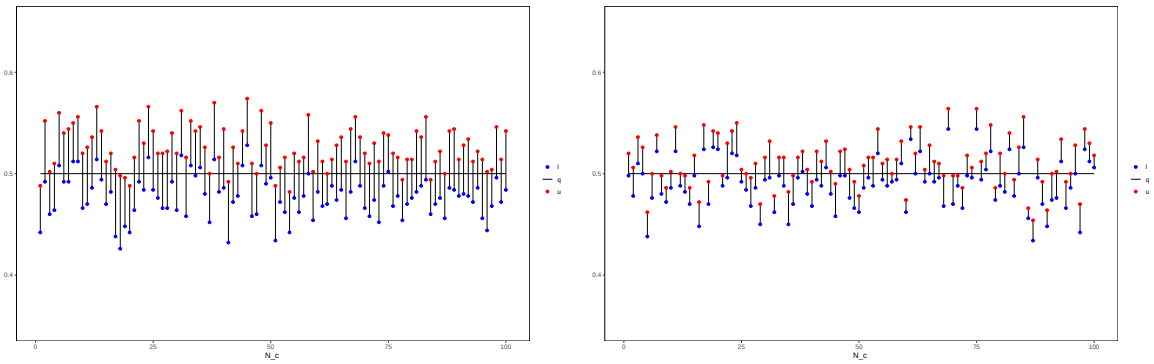
(f) $q = 0.75$, $n = 50$

Figure D.5: Intervals $[l_k, u_k]$ where $k = 1, ..., 100$; Normal copula is for simulation and Frank copula is for inference, $\tau = 0.5$.



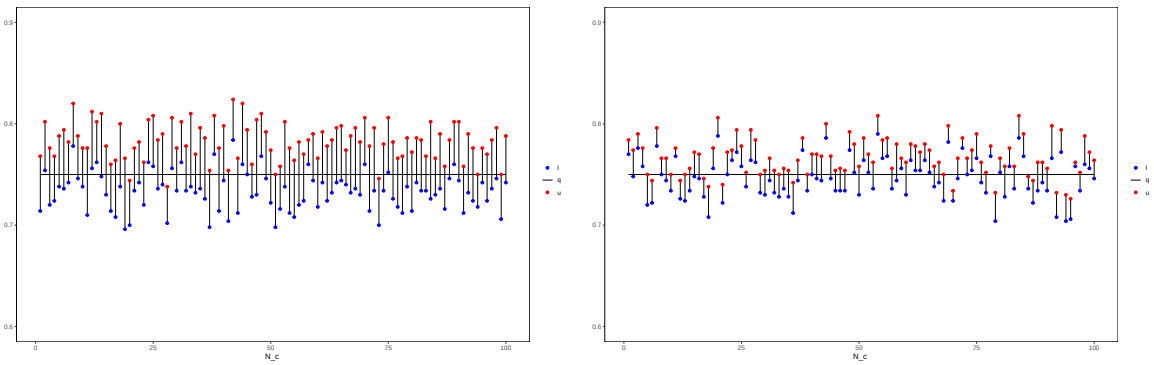
(a) $q = 0.25$, $n = 20$

(b) $q = 0.25$, $n = 50$



(c) $q = 0.5$, $n = 20$

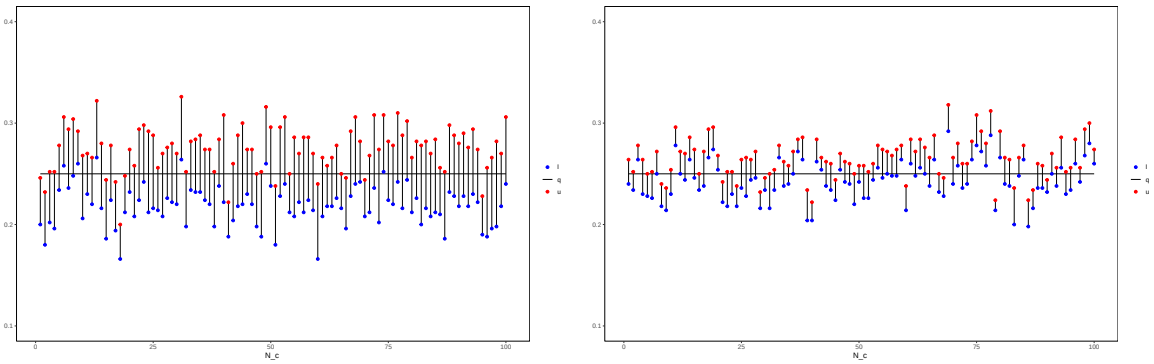
(d) $q = 0.5$, $n = 50$



(e) $q = 0.75$, $n = 20$

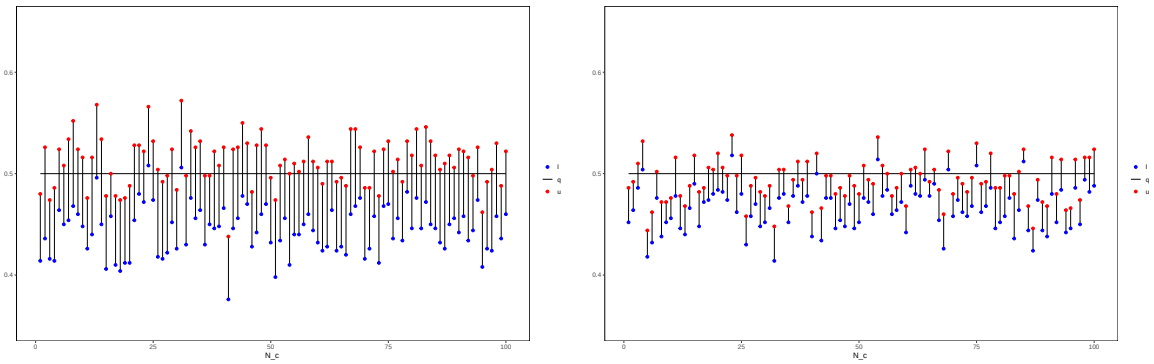
(f) $q = 0.75$, $n = 50$

Figure D.6: Intervals $[l_k, u_k]$ where $k = 1, ..., 100$; Normal copula is for simulation and Frank copula is for inference, $\tau = 0.75$.



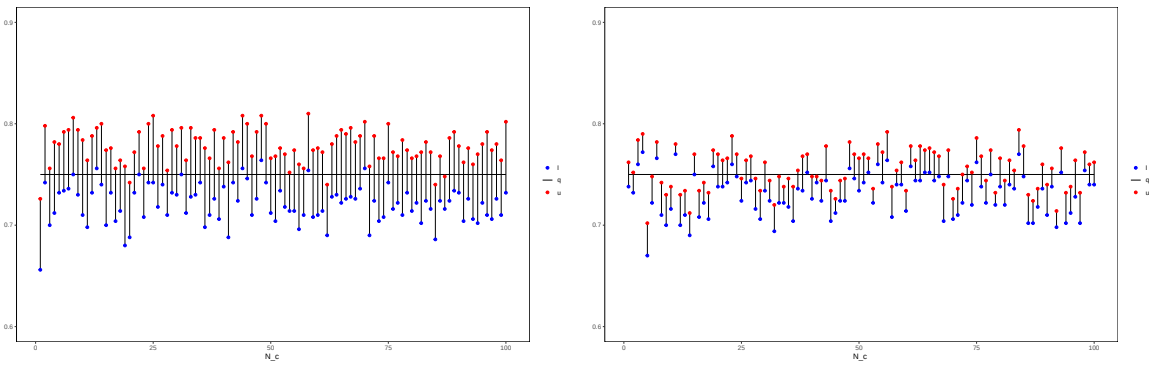
(a) $q = 0.25$, $n = 20$

(b) $q = 0.25$, $n = 50$



(c) $q = 0.5$, $n = 20$

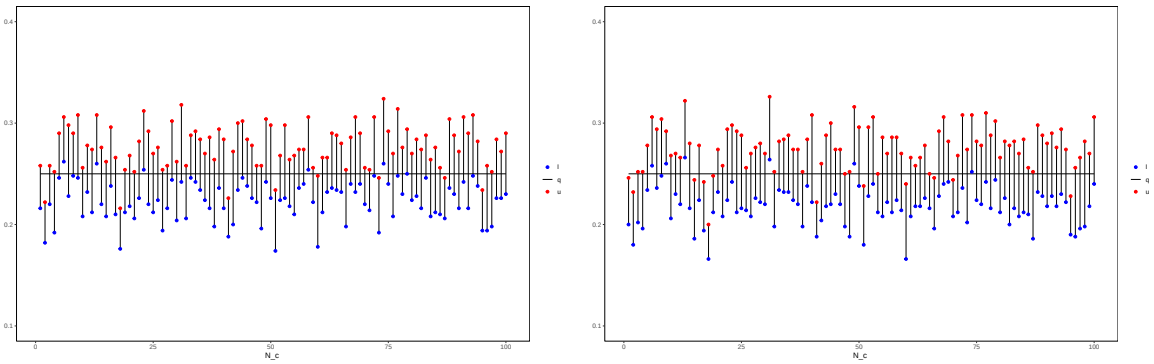
(d) $q = 0.5$, $n = 50$



(e) $q = 0.75$, $n = 20$

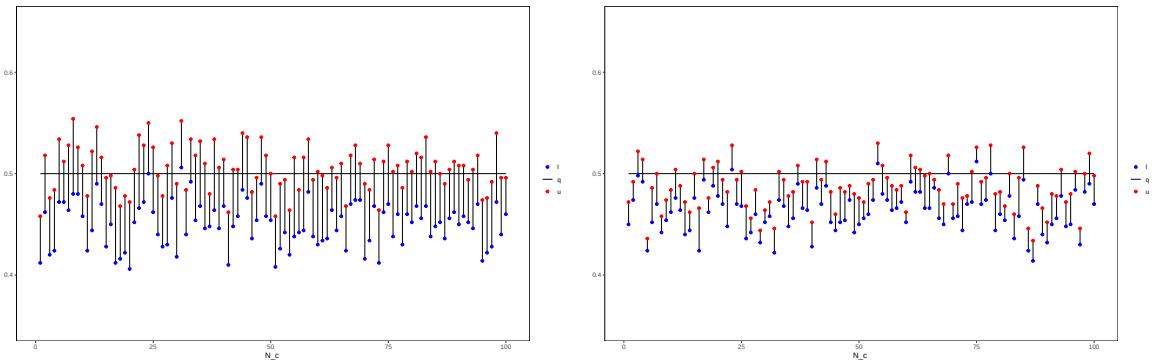
(f) $q = 0.75$, $n = 50$

Figure D.7: Intervals $[l_k, u_k]$ where $k = 1, ..., 100$; Normal copula is for simulation and Clayton copula is for inference, $\tau = 0.25$.



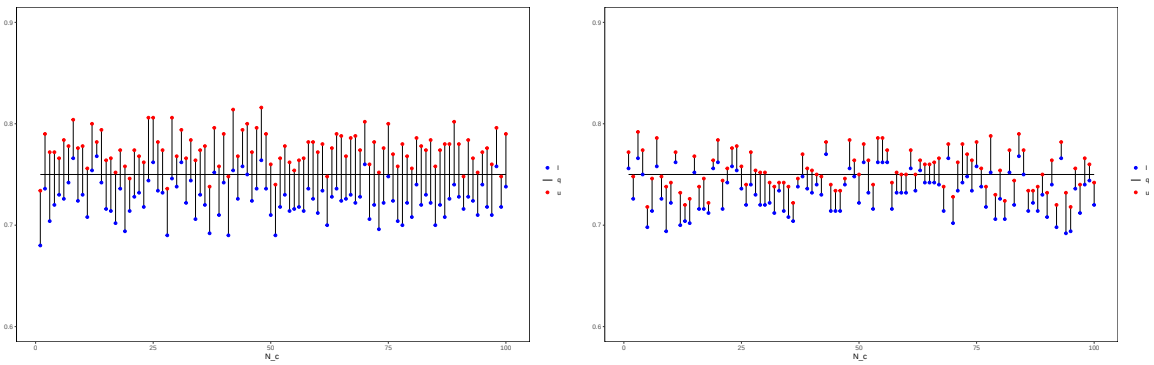
(a) $q = 0.25$, $n = 20$

(b) $q = 0.25$, $n = 50$



(c) $q = 0.5$, $n = 20$

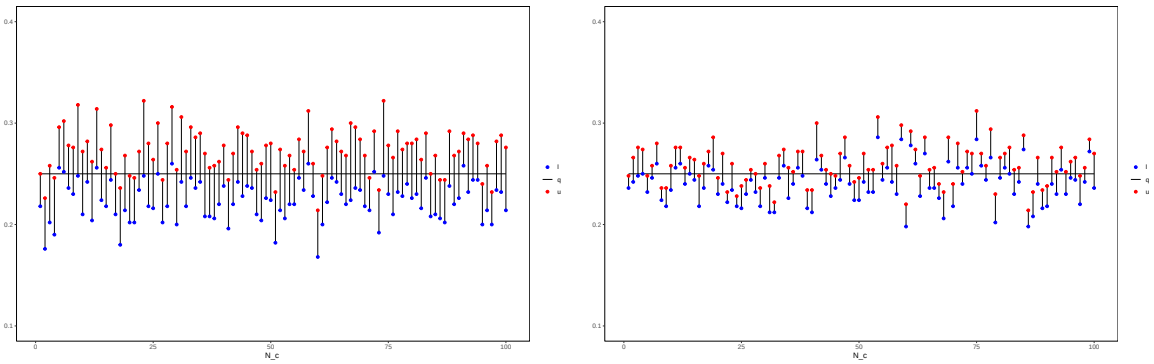
(d) $q = 0.5$, $n = 50$



(e) $q = 0.75$, $n = 20$

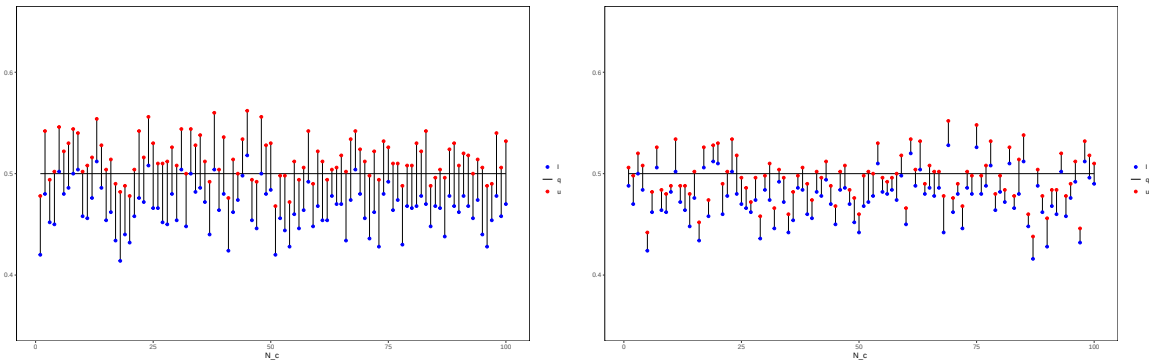
(f) $q = 0.75$, $n = 50$

Figure D.8: Intervals $[l_k, u_k]$ where $k = 1, ..., 100$; Normal copula is for simulation and Clayton copula is for inference, $\tau = 0.5$.



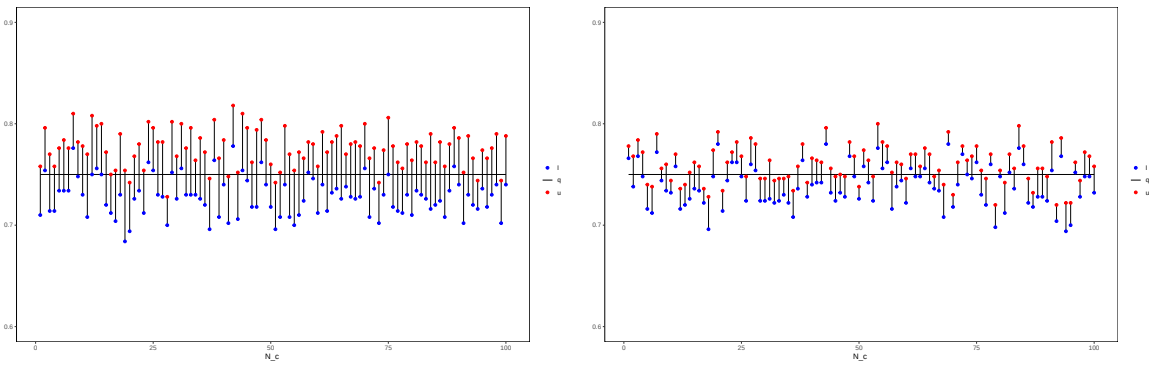
(a) $q = 0.25$, $n = 20$

(b) $q = 0.25$, $n = 50$



(c) $q = 0.5$, $n = 20$

(d) $q = 0.5$, $n = 50$



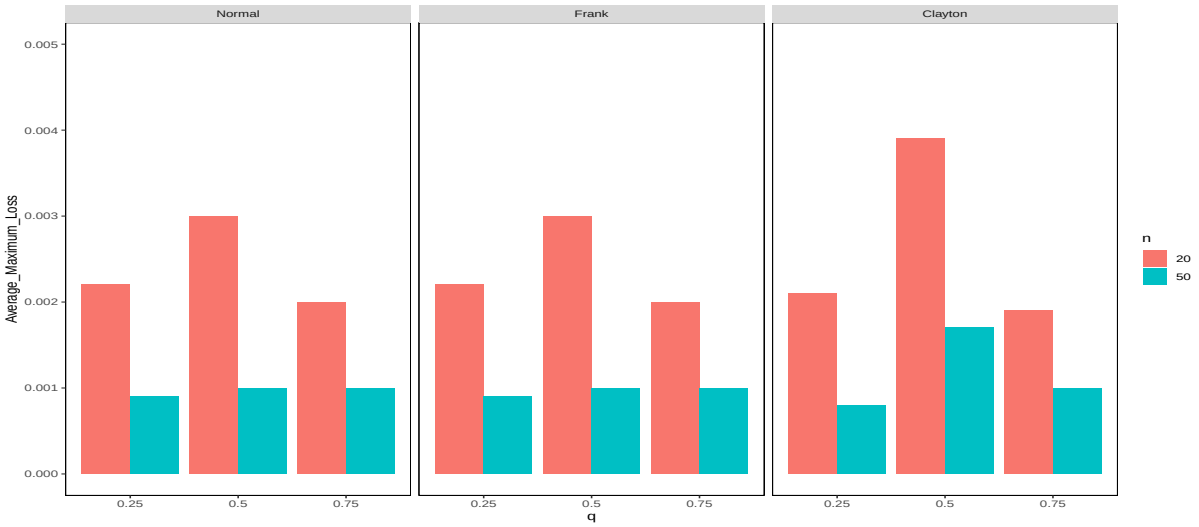
(e) $q = 0.75$, $n = 20$

(f) $q = 0.75$, $n = 50$

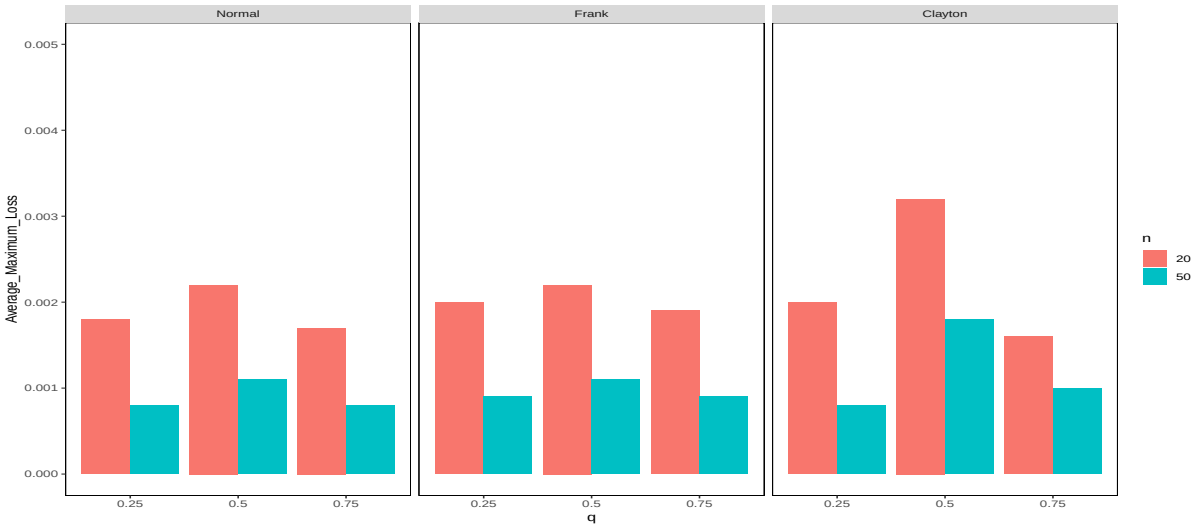
Figure D.9: Intervals $[l_k, u_k]$ where $k = 1, ..., 100$; Normal copula is for simulation and Clayton copula is for inference when $\tau = 0.75$.

D.2 Extended results for loss function

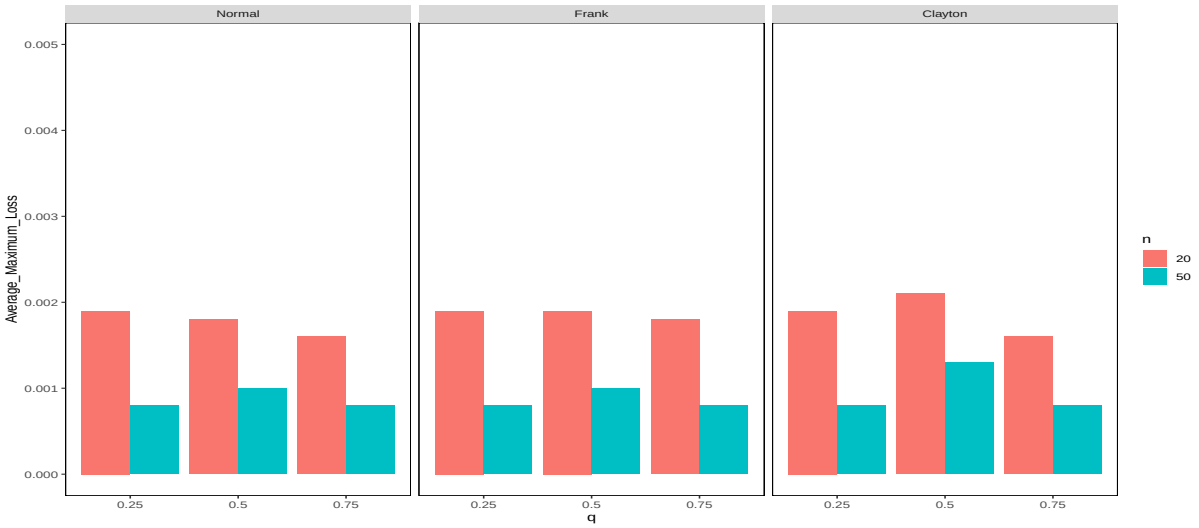
This section includes additional results from the simulation study when using absolute loss function (L_1) using Equation (5.15) in Chapter 5. Then the average values of maximum and minimum loss are reported in Figures D.18 and D.19. The results show almost the same pattern as when using quadratic loss L_2. Figure D.19 shows that when $\tau = q = 0.25$ and Normal copula is used for inference, unlike when using L_2, the average value of minimum loss using L_1 when $n = 50$ is higher than the value when $n = 20$. This can be explained as L_2 penalises outliers heavily unlike L_1 which is less sensitive to them.



(a) $\tau = 0.25$

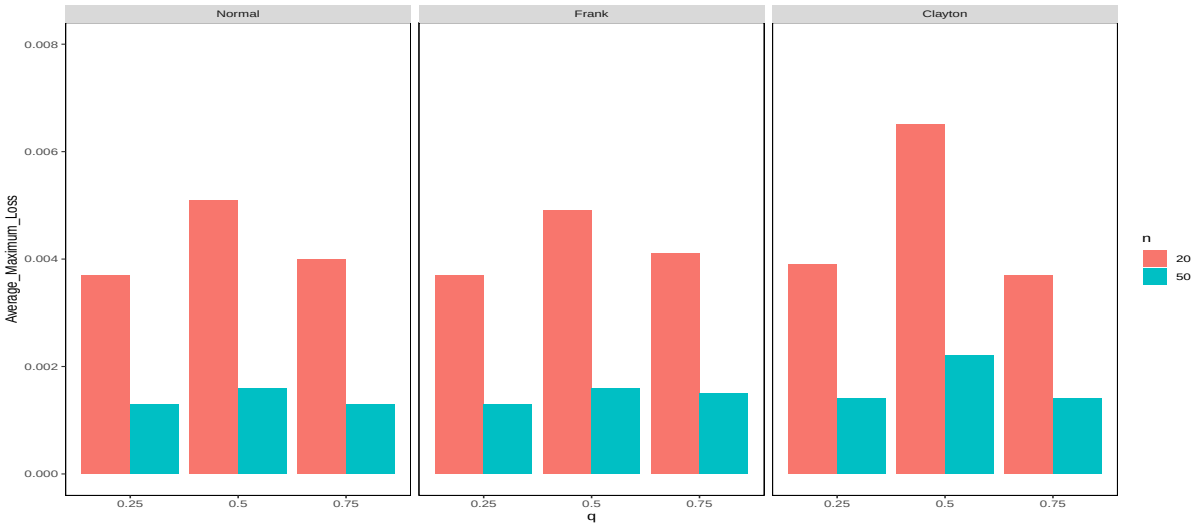


(b) $\tau = 0.5$

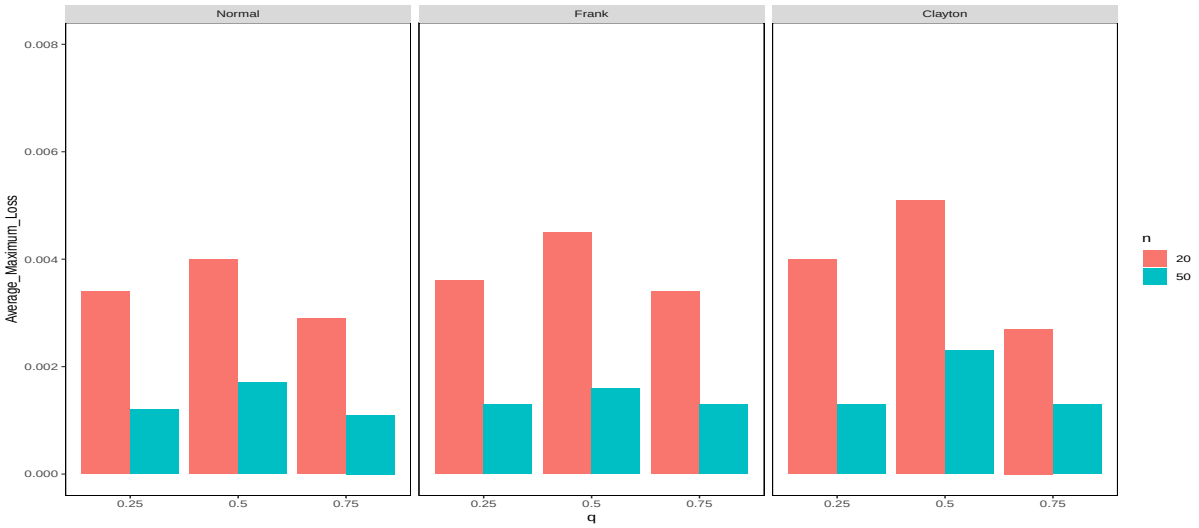


(c) $\tau = 0.75$

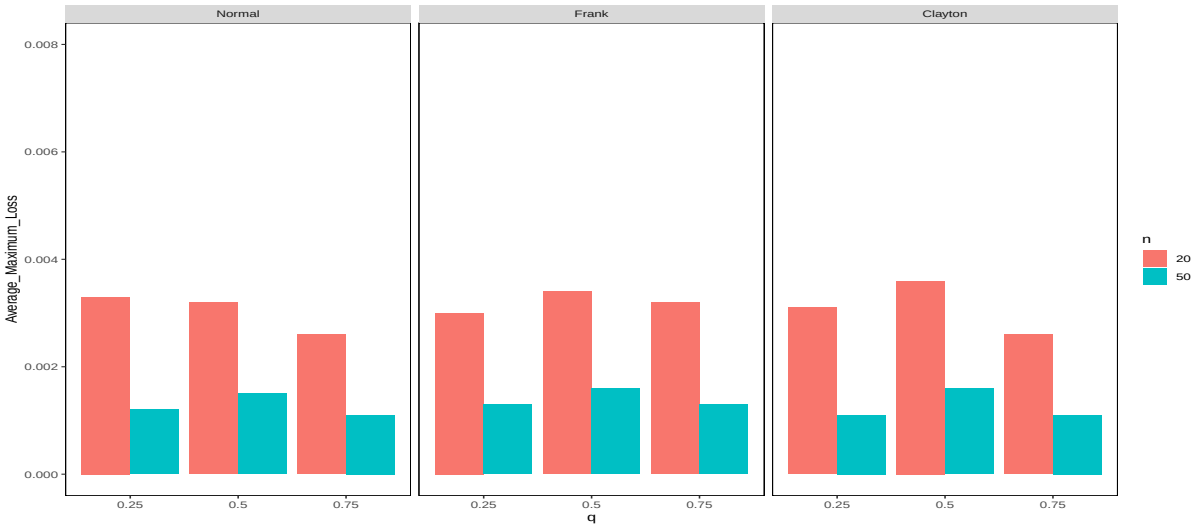
Figure D.10: Average of maximum loss using L_2; simulation from Normal copula; Normal, Frank, and Clayton copulas are for inference.



(a) $\tau = 0.25$

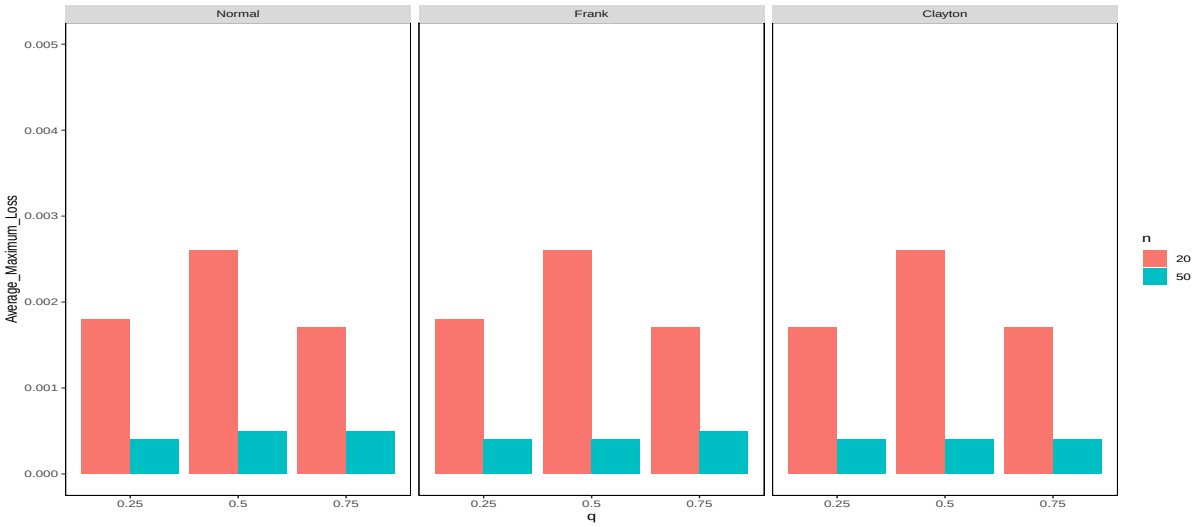


(b) $\tau = 0.5$

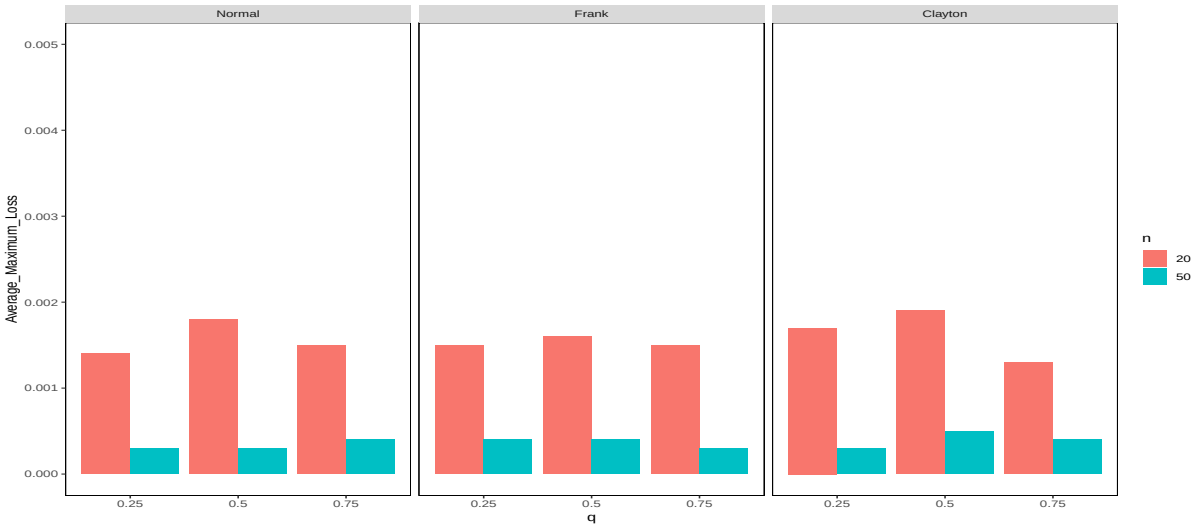


(c) $\tau = 0.75$

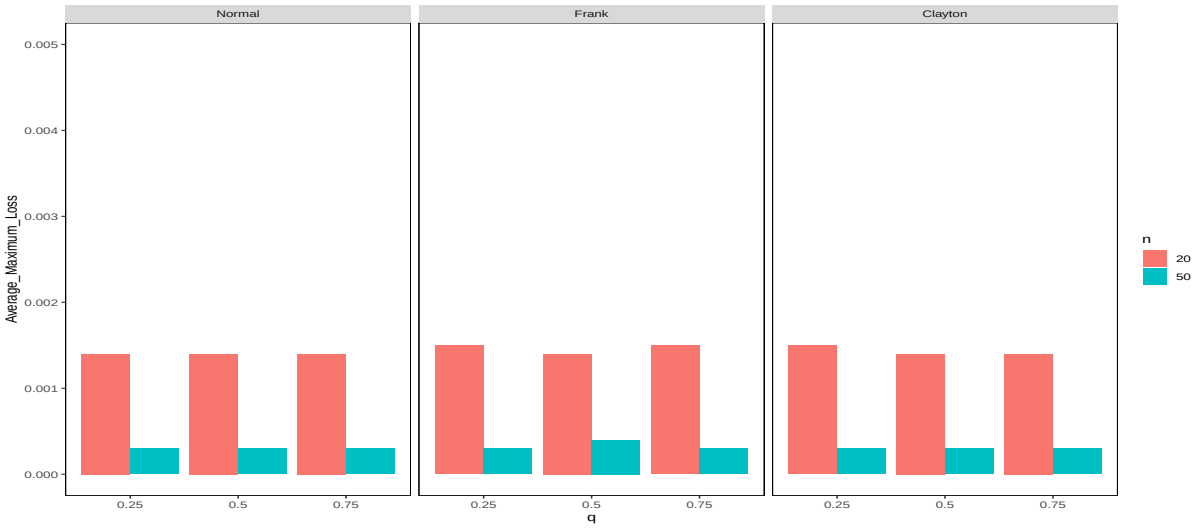
Figure D.11: Average of maximum loss using L_2, when $q \notin [l_k, u_k]$; simulation from Normal copula; Normal, Frank, and Clayton copulas are for inference.



(a) $\tau = 0.25$

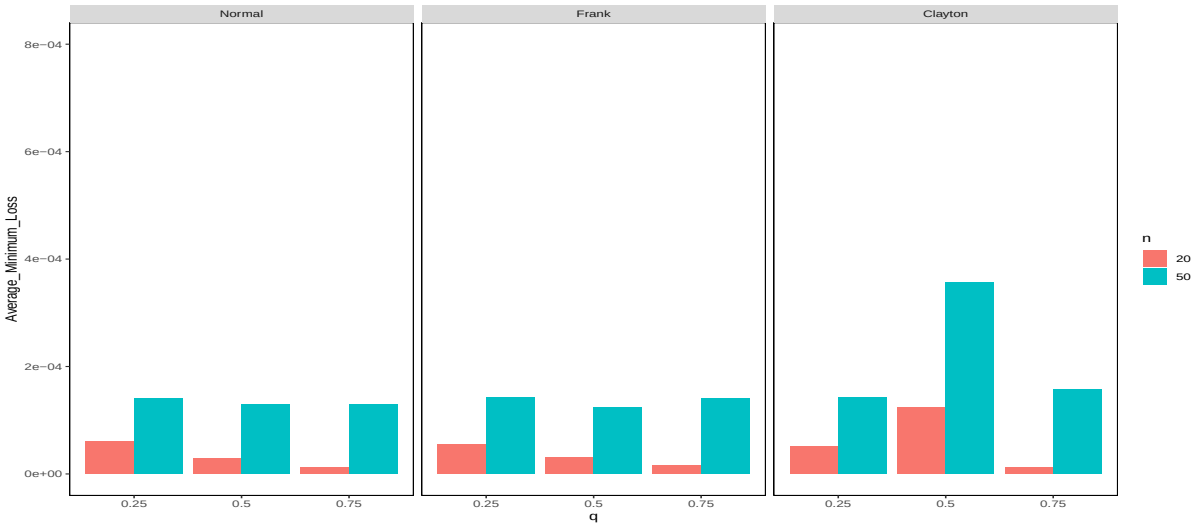


(b) $\tau = 0.5$

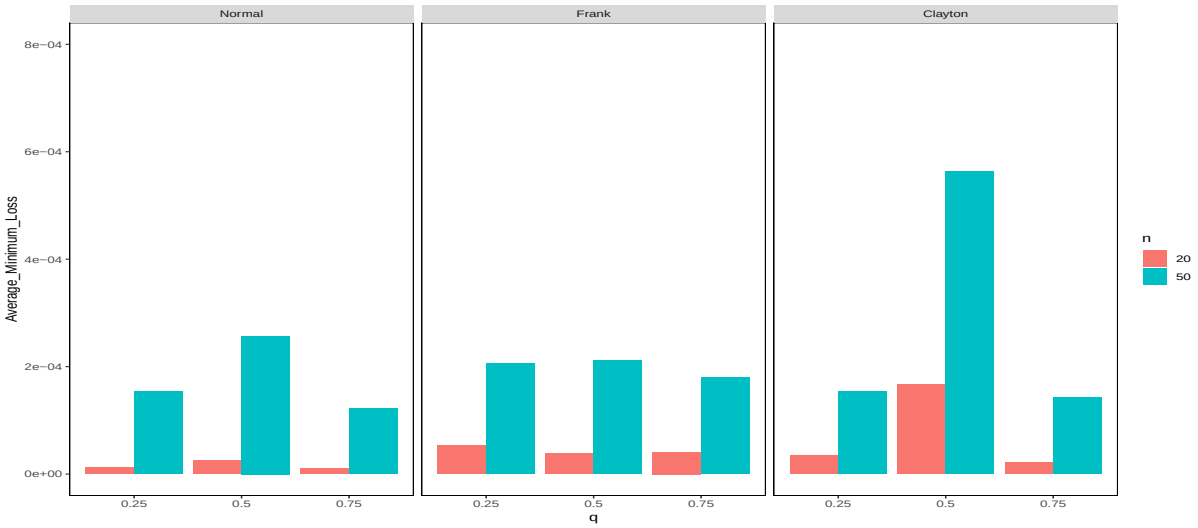


(c) $\tau = 0.75$

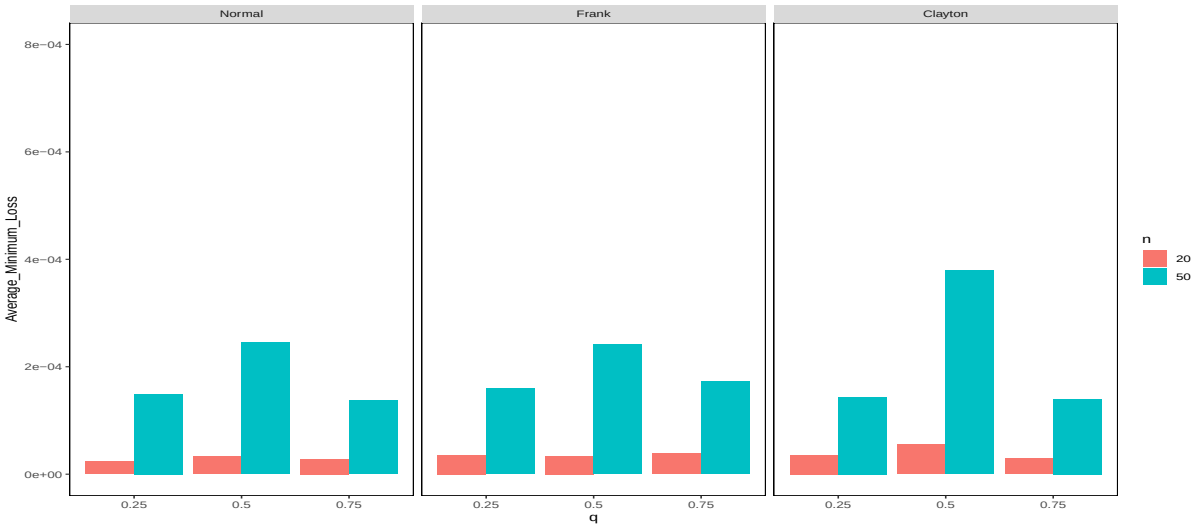
Figure D.12: Average of maximum loss using L_2, when $q \in [l_k, u_k]$; simulation from Normal copula; Normal, Frank, and Clayton copulas are for inference.



(a) $\tau = 0.25$

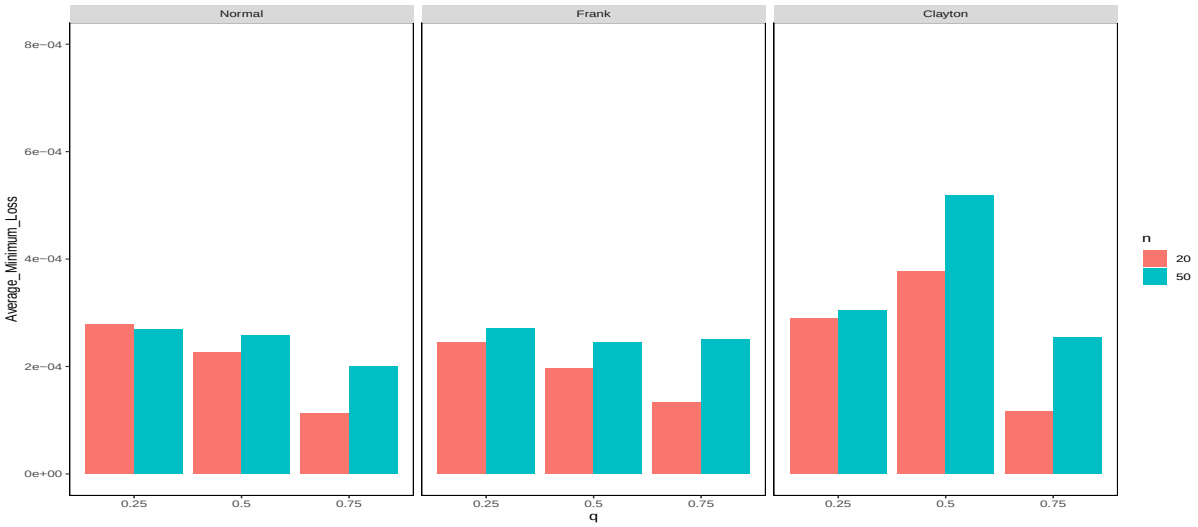


(b) $\tau = 0.5$

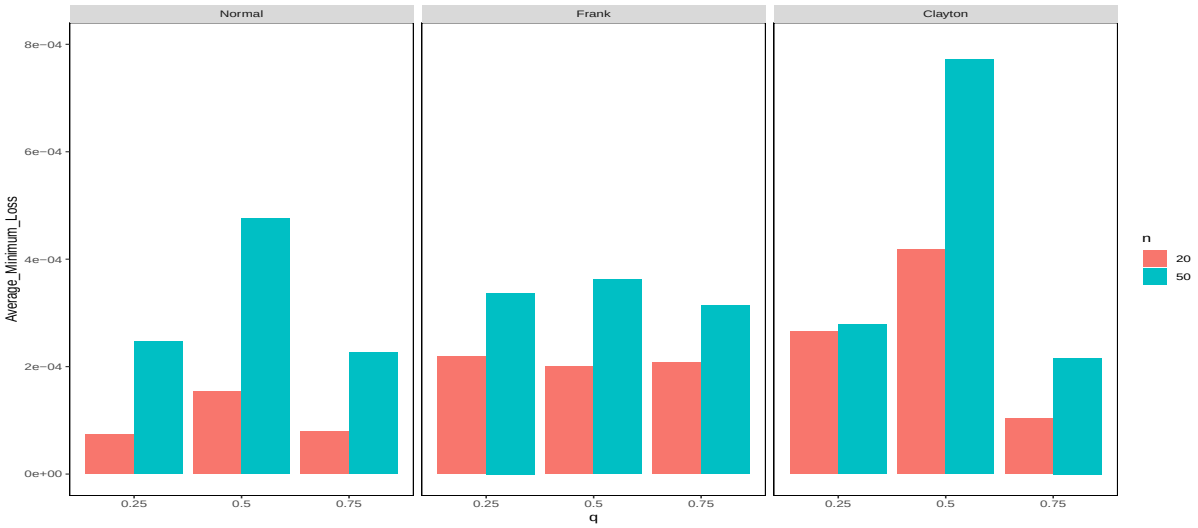


(c) $\tau = 0.75$

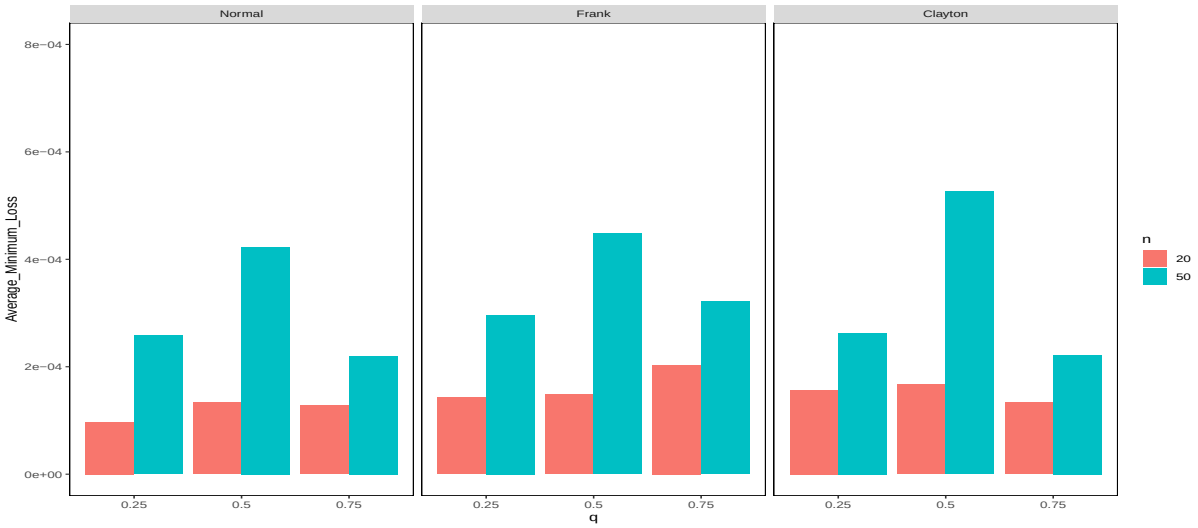
Figure D.13: Average of minimum loss using L_2; simulation from Normal copula; Normal, Frank, and Clayton copulas are for inference.



(a) $\tau = 0.25$

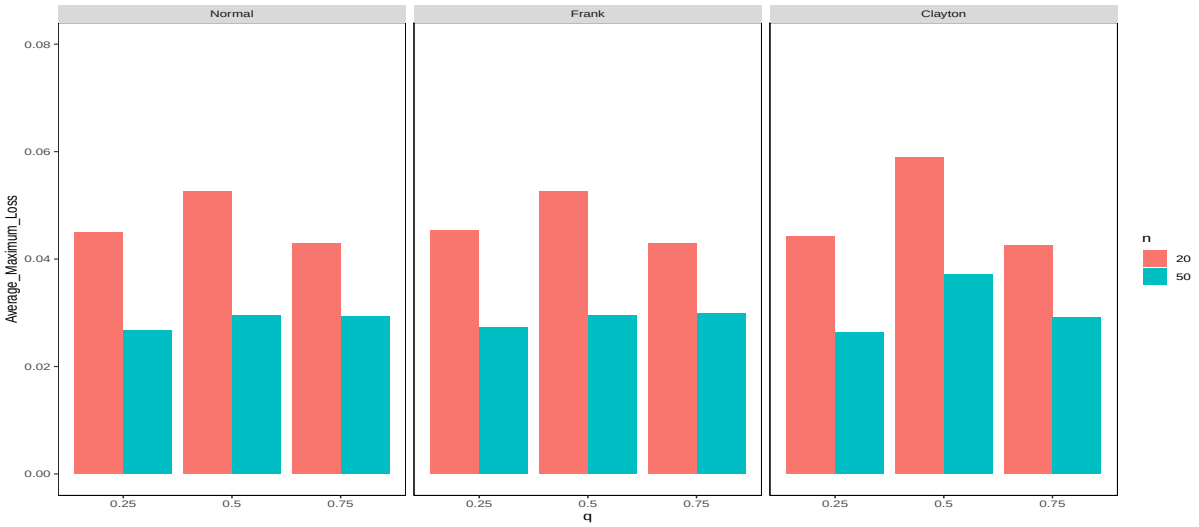


(b) $\tau = 0.5$

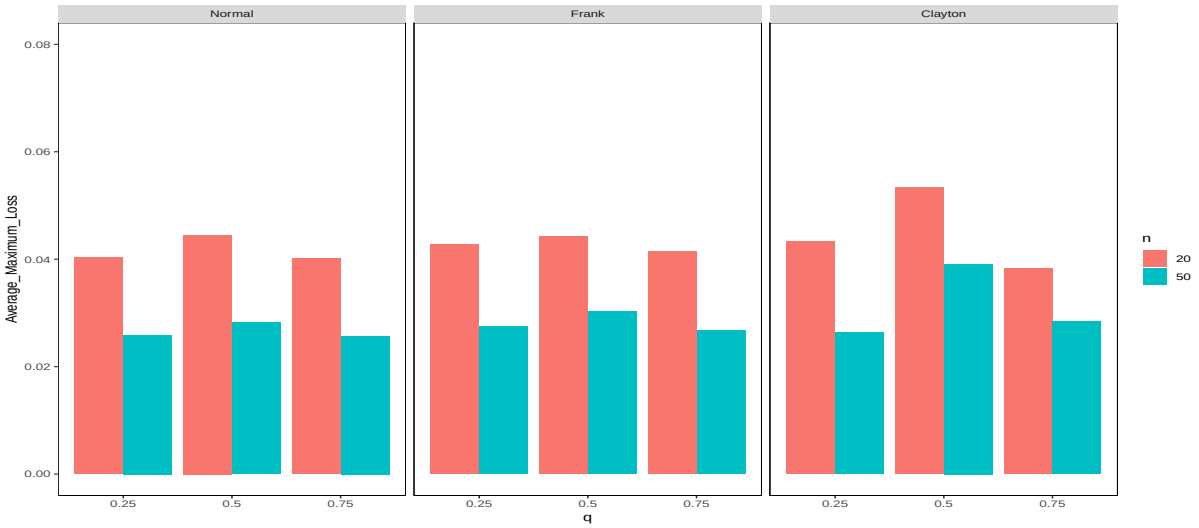


(c) $\tau = 0.75$

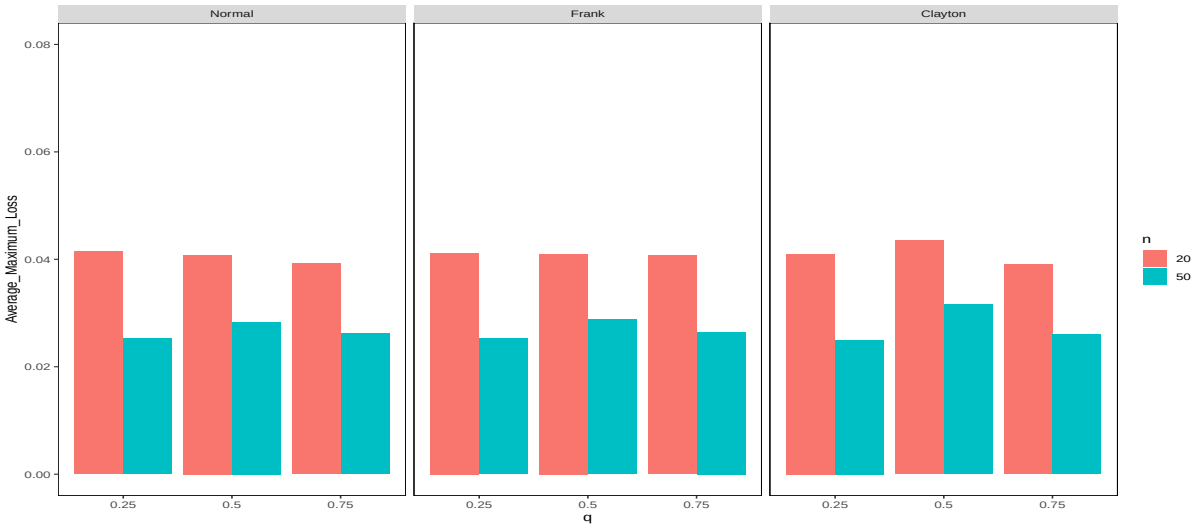
Figure D.14: Average of minimum loss using L_2 when $q \notin [l_k, u_k]$; simulation from Normal copula; Normal, Frank, and Clayton copulas are for inference.



(a) $\tau = 0.25$

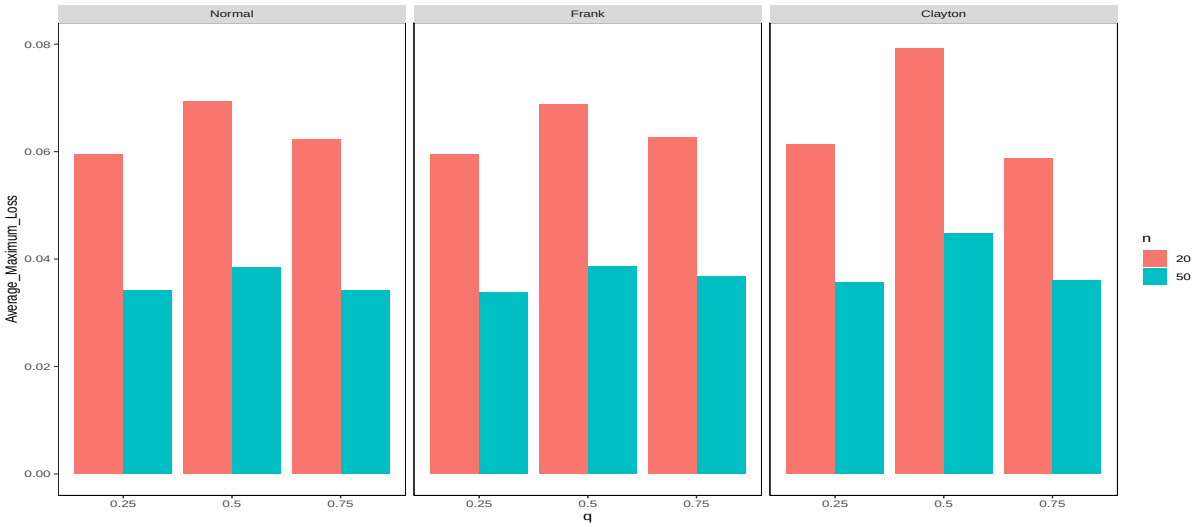


(b) $\tau = 0.5$

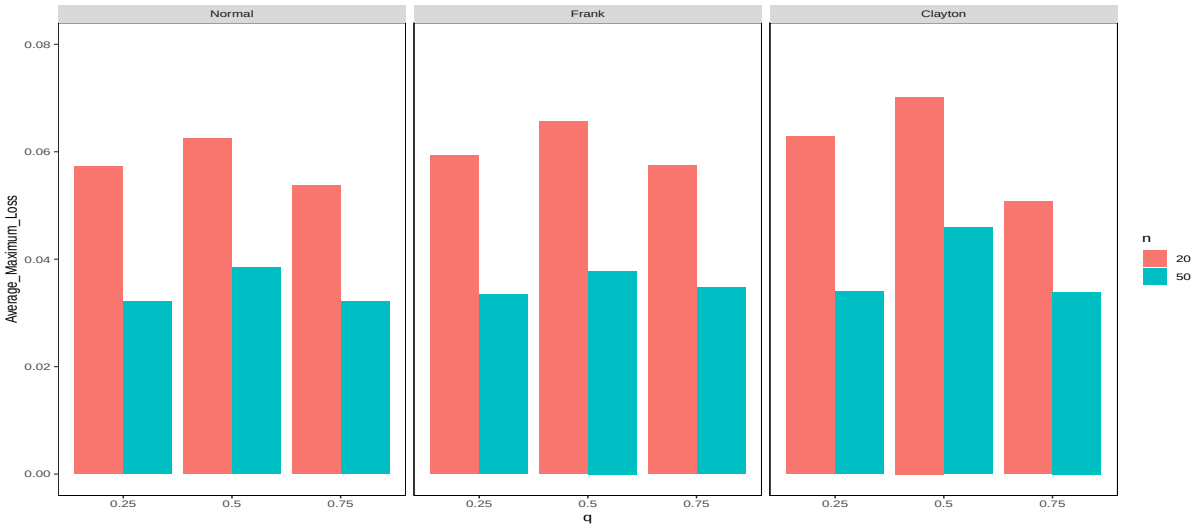


(c) $\tau = 0.75$

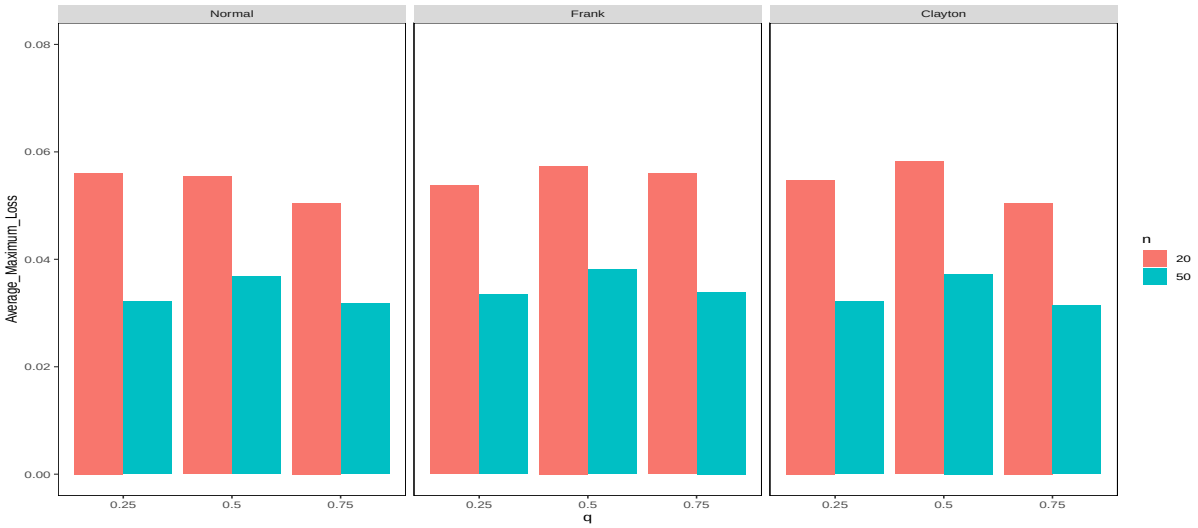
Figure D.15: Average of maximum loss using L_1; simulation from Normal copula; Normal, Frank, and Clayton copulas are for inference.



(a) $\tau = 0.25$

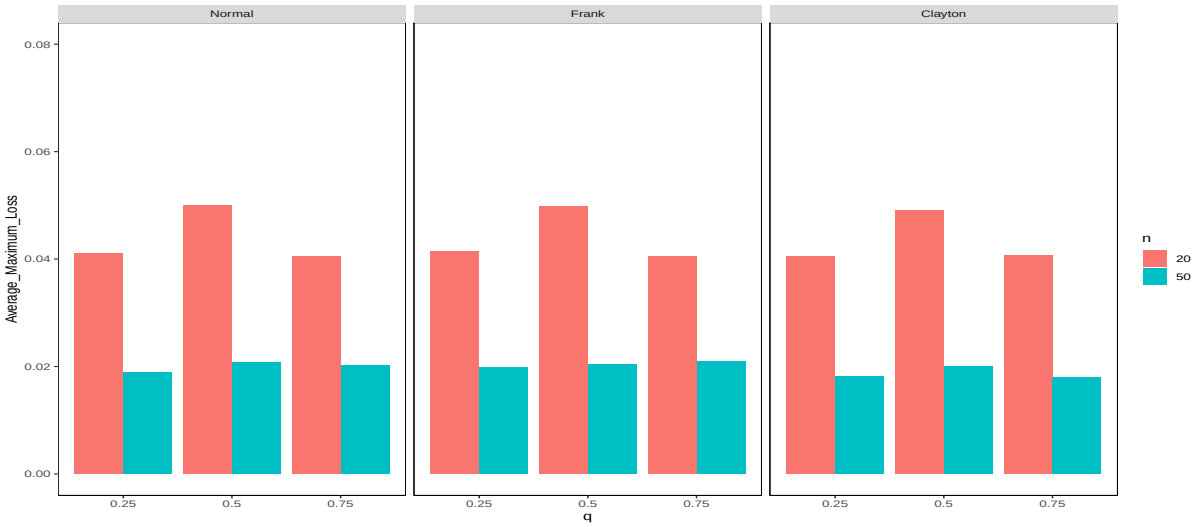


(b) $\tau = 0.5$

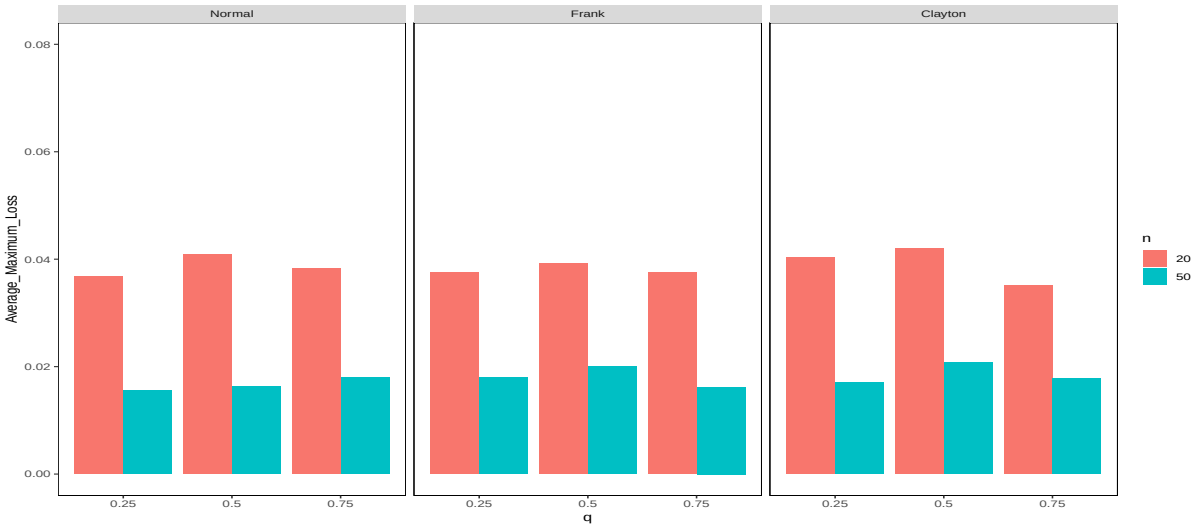


(c) $\tau = 0.75$

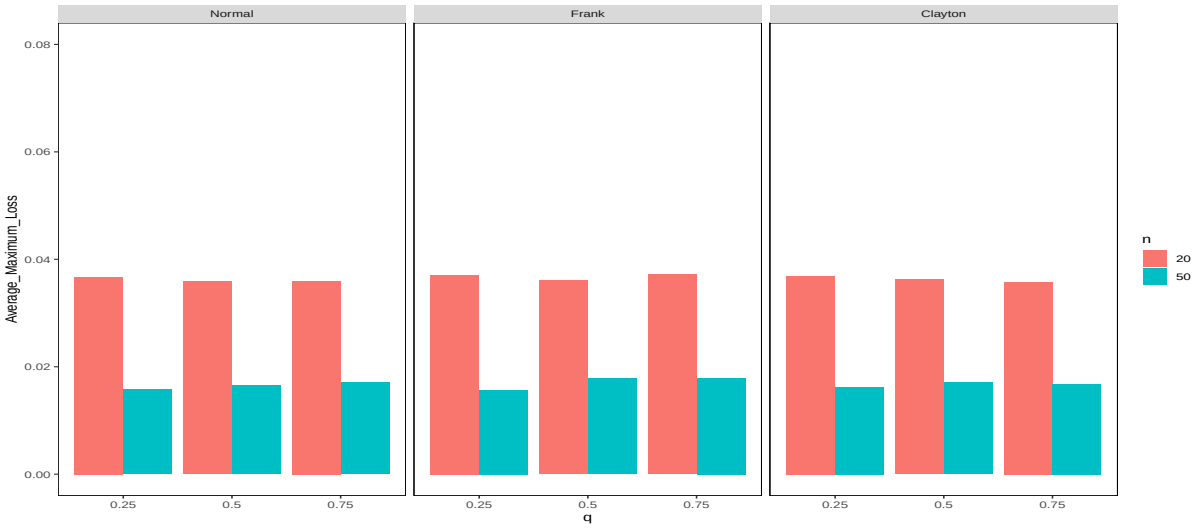
Figure D.16: Average of maximum loss using L_1 when $q \notin [l_k, u_k]$; simulation from Normal copula; Normal, Frank, and Clayton copulas are for inference.



(a) $\tau = 0.25$

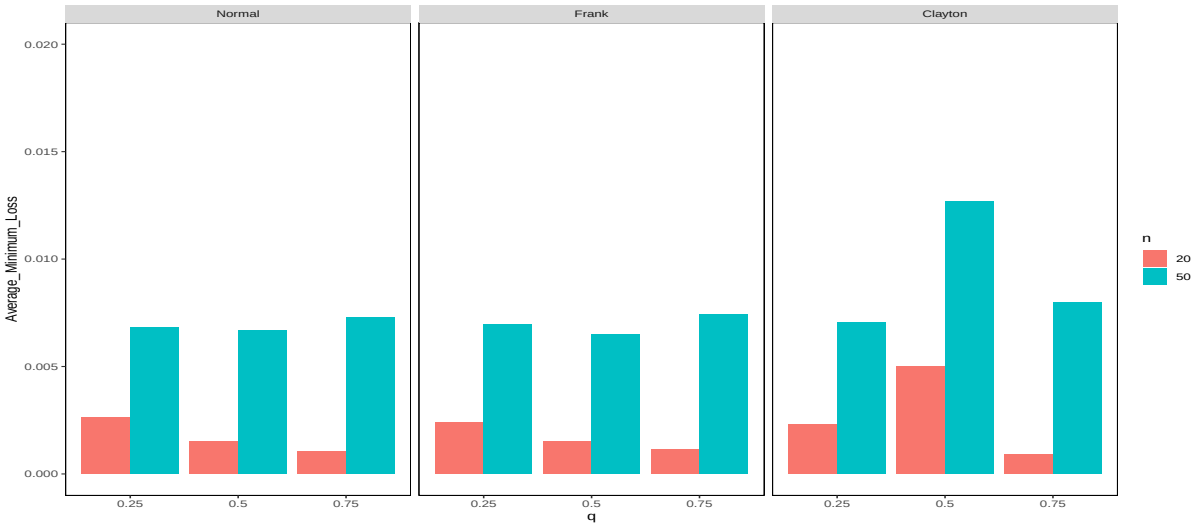


(b) $\tau = 0.5$

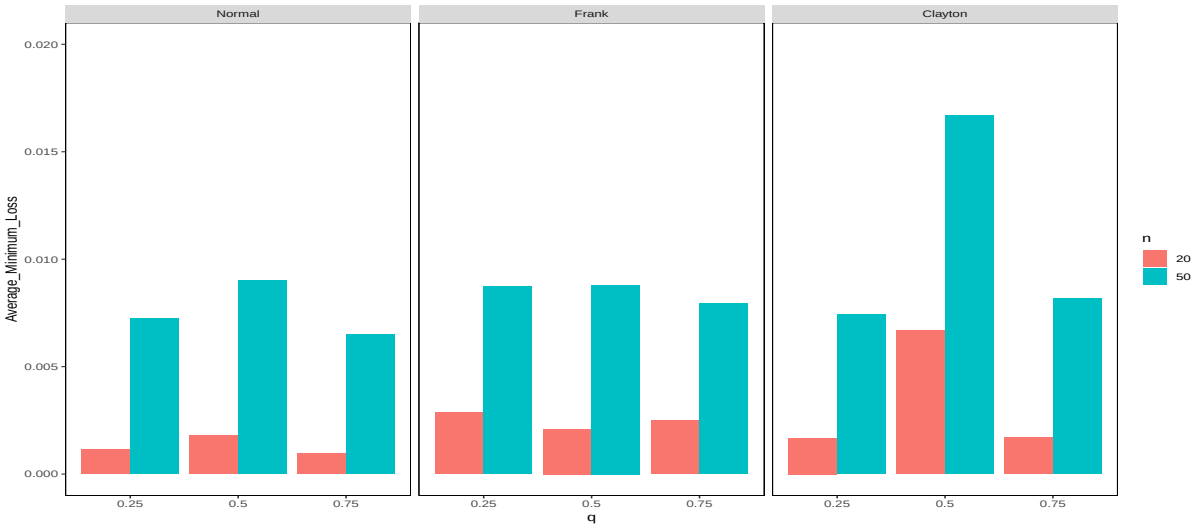


(c) $\tau = 0.75$

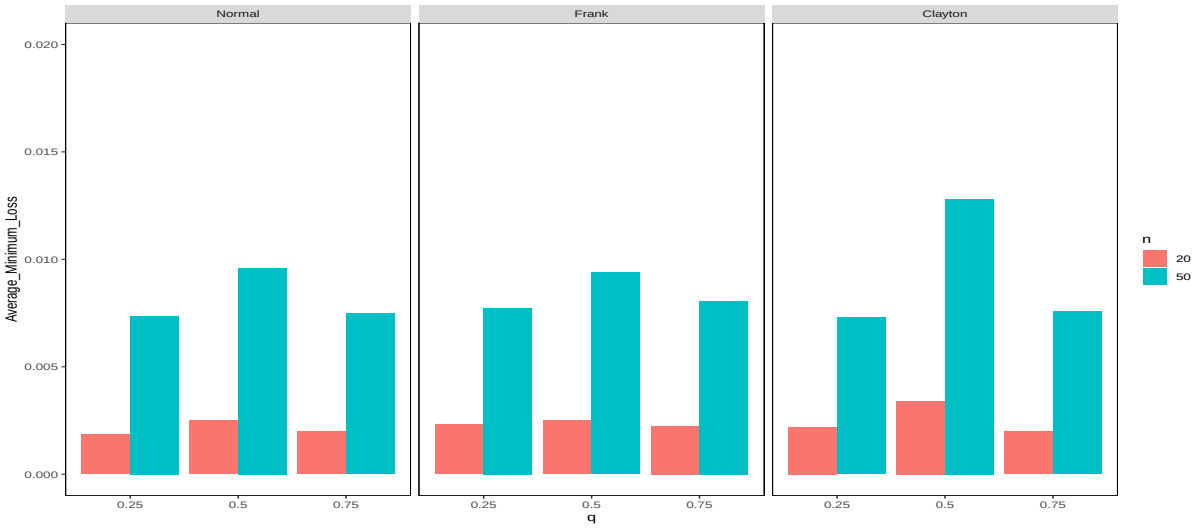
Figure D.17: Average of maximum loss using L_1 when $q \in [l_k, u_k]$; simulation from Normal copula; Normal, Frank, and Clayton copulas are for inference.



(a) $\tau = 0.25$

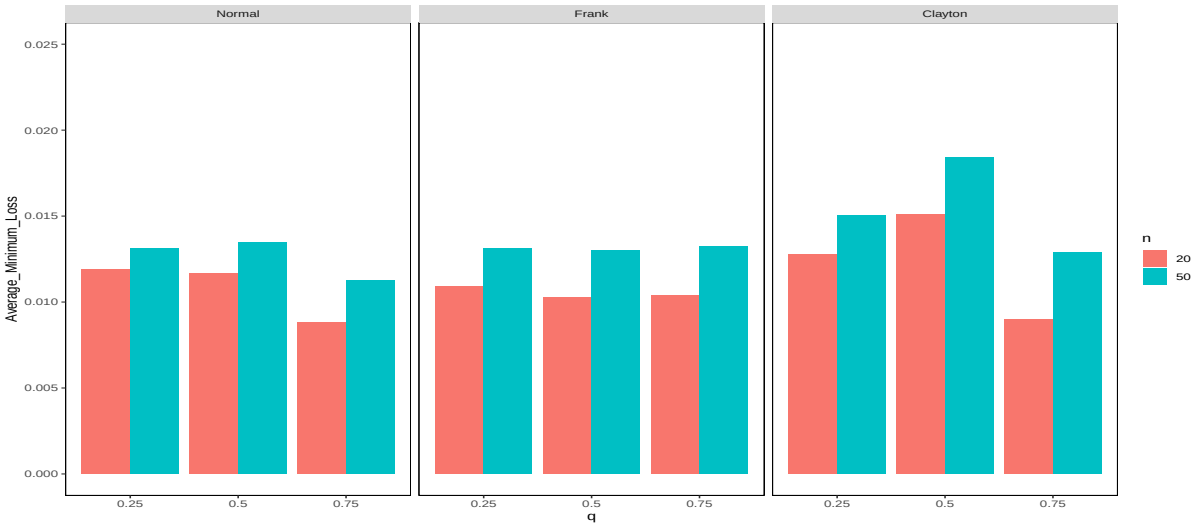


(b) $\tau = 0.5$

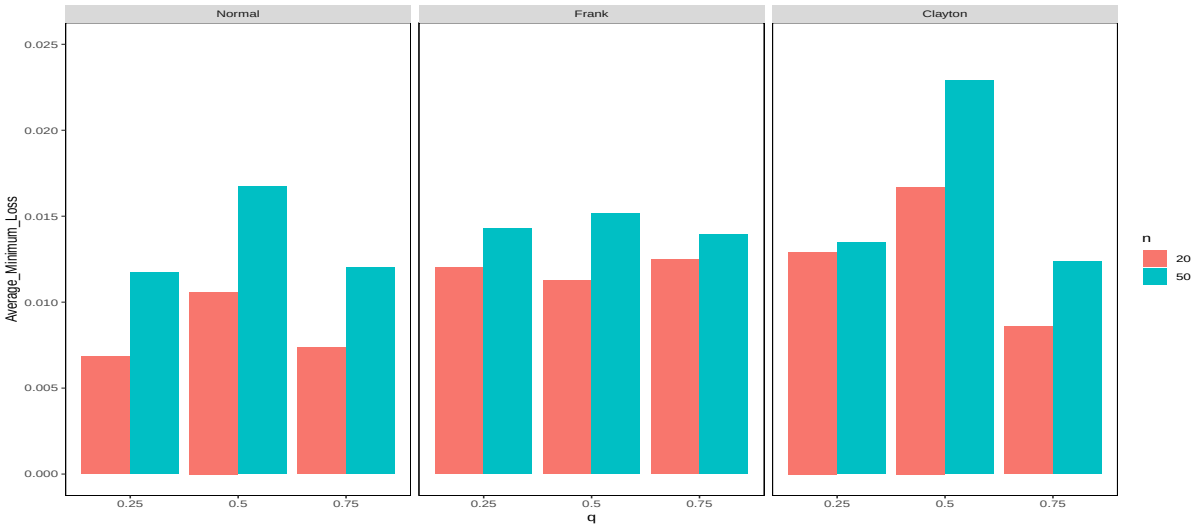


(c) $\tau = 0.75$

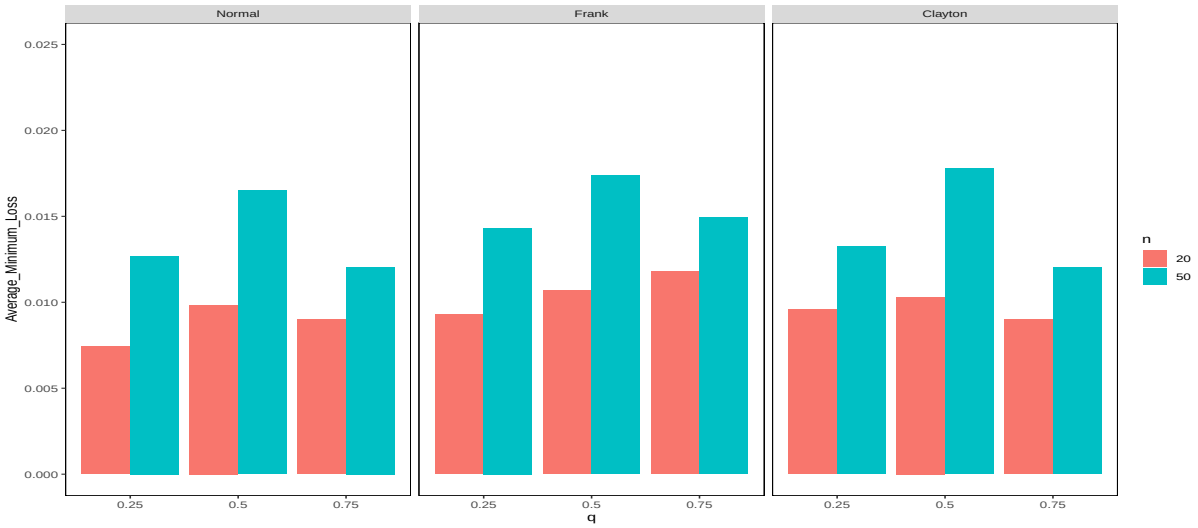
Figure D.18: Average of minimum loss using L_1; simulation from Normal copula; Normal, Frank, and Clayton copulas are for inference.



(a) $\tau = 0.25$



(b) $\tau = 0.5$

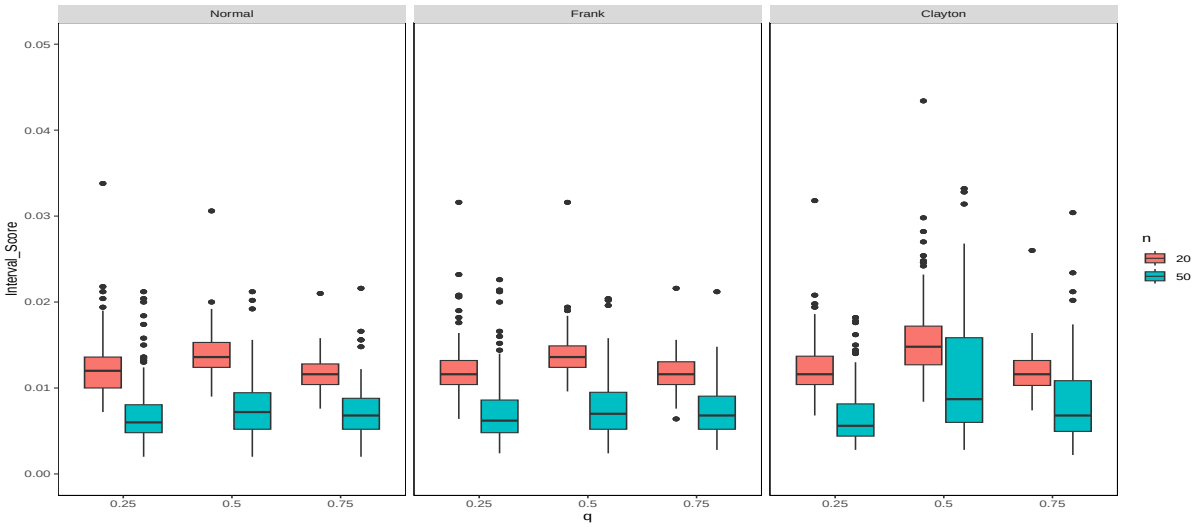


(c) $\tau = 0.75$

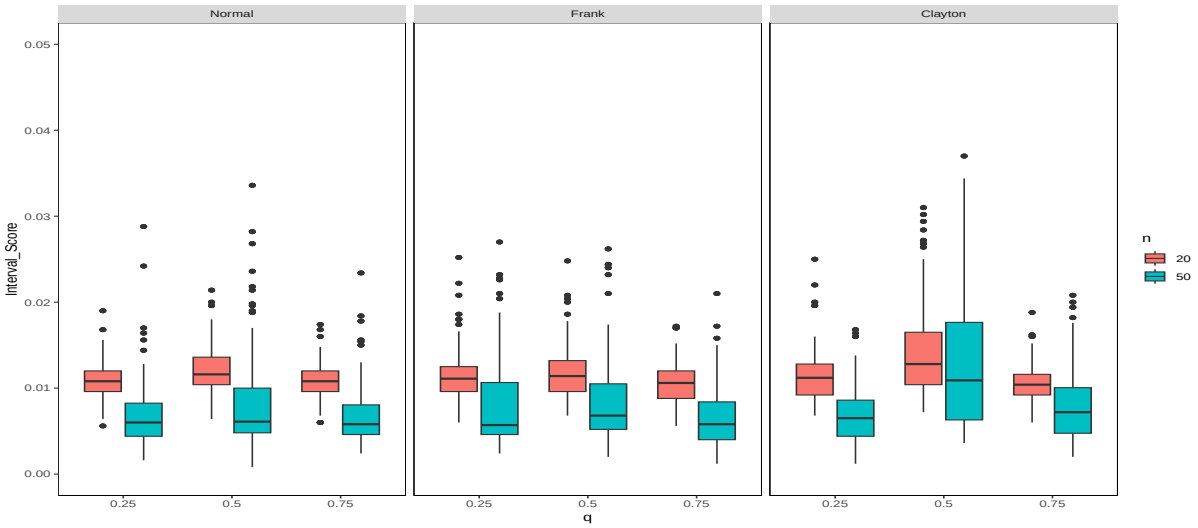
Figure D.19: Average of minimum loss using L_1 when $q \notin [l_k, u_k]$; simulation from Normal copula; Normal, Frank, and Clayton copulas are for inference.

D.3 Extended results for interval score

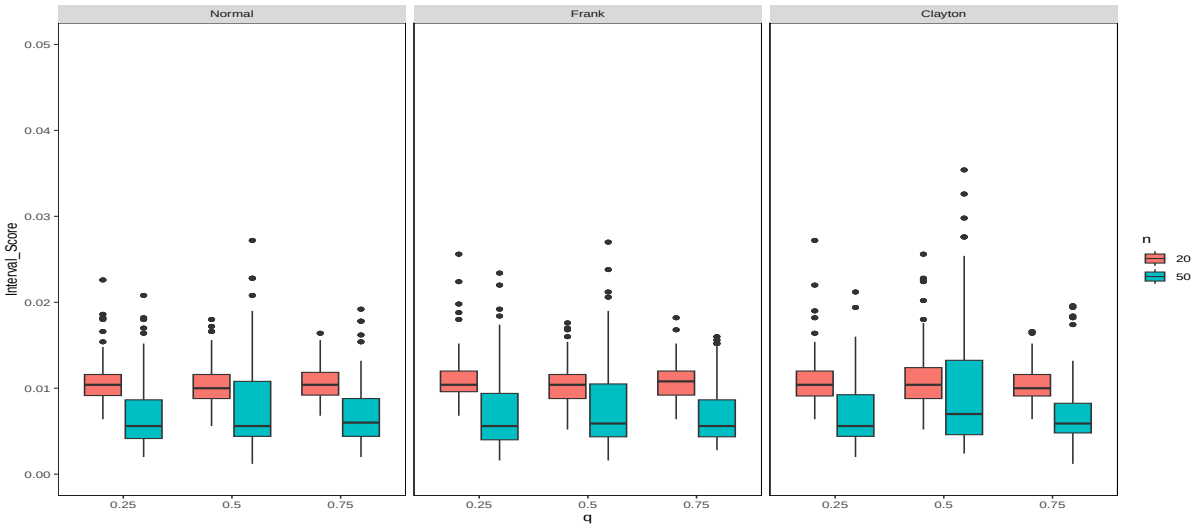
Interval score ($\text{IS}_{(c_1,c_2,c_3)}$) is used in this section, assuming additional different values for the weights c_1, c_2, and c_3. Two cases are considered: first, when $c_1 = 0.2$, $c_2 = 0.3$, and $c_3 = 0.5$, and second, when $c_1 = 0.2$, $c_2 = 0.5$, and $c_3 = 0.3$. The results are presented in Figures D.20 and D.21. In Figure D.20, when the penalisation for $u < q$ is greater than that for $l > q$ and $u - l$, the results show almost the same pattern as in Figure 5.2 in Chapter 5. In some cases, interquartile ranges are lower than those in Figure 5.2, with more outliers. Increasing the weight c_3 leads to considering more intervals "far away" from the value q. In contrast, when using Clayton copula for inference, and $q = 0.5$ and $n = 50$, the interquartile ranges are higher than those in Figure 5.2. This it is likely because when the inequality $l < q < u$ is not satisfied, more intervals tend to follow $u < q$. This trend is also observed in Figures D.7, D.8, and D.9 in Chapter 5. When comparing those results to those of the box plots. In Figure D.21, when $\text{IS}_{(0.2,0.5,0.3)}$, the interquartiles ranges are lower due to the positions of the intervals being near the value q when $l > q$ is applied.



(a) $\tau = 0.25$

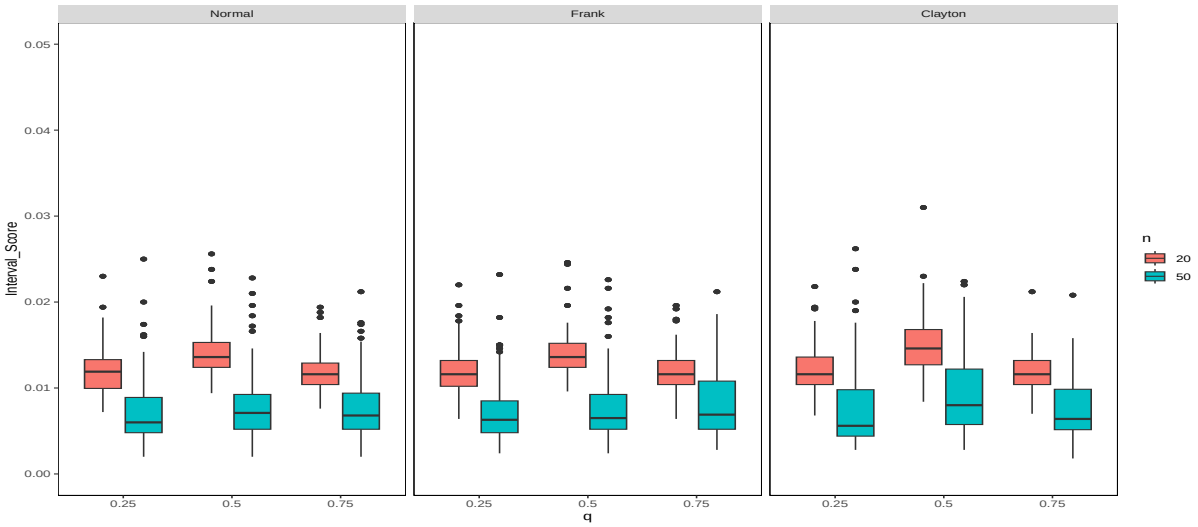


(b) $\tau = 0.5$

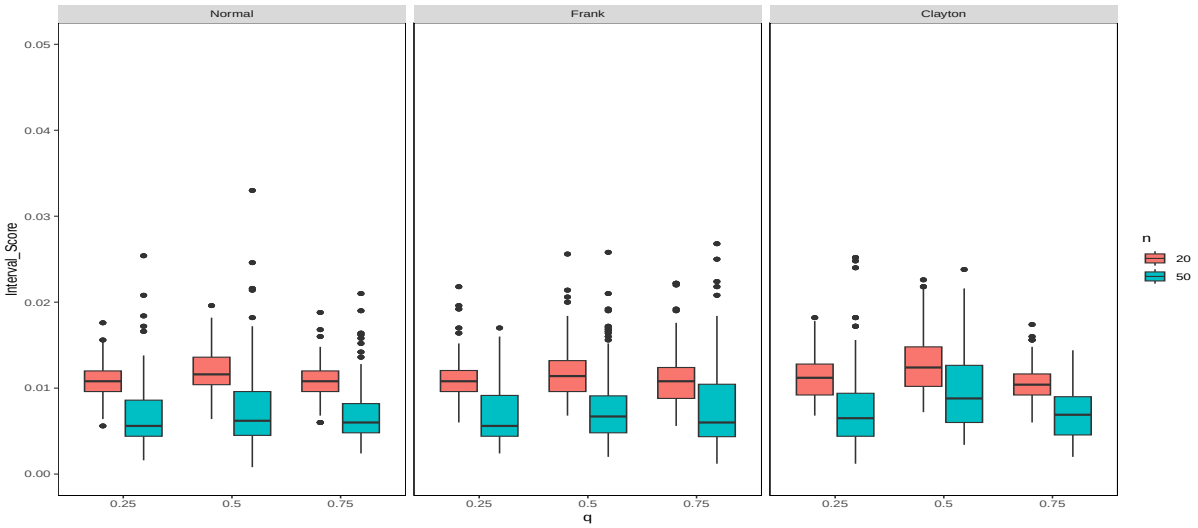


(c) $\tau = 0.75$

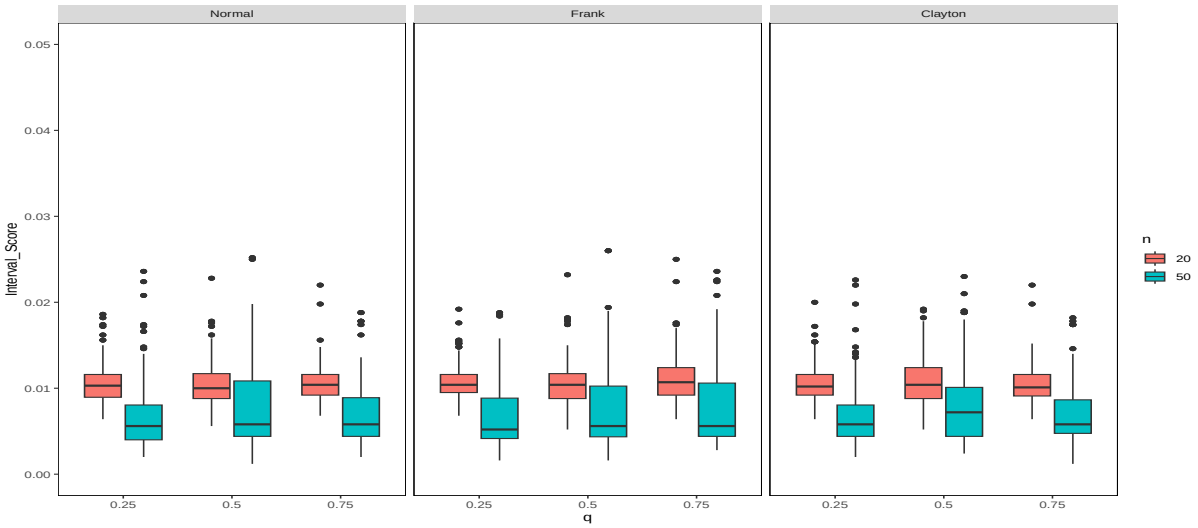
Figure D.20: Results of $\text{IS}_{(0.2,0.3,0.5)}$; simulation from Normal copula; Normal, Frank, and Clayton copulas are for inference.



(a) $\tau = 0.25$



(b) $\tau = 0.5$



(c) $\tau = 0.75$

Figure D.21: Results of $\text{IS}_{(0.2,0.5,0.3)}$; simulation from Normal copula; Normal, Frank, and Clayton copulas are for inference.

Bibliography

[1] Abellán, J. (2006). Uncertainty measures on probability intervals from the imprecise Dirichlet model. *International Journal of General Systems,* 35(5), 509–528. doi: 10.1080/03081070600687643.

[2] Abellán, J. (2013). An application of non-parametric predictive inference on multiclass classification high-level-noise problems. *Expert Systems with Applications,* 40(11), 4585–4592. doi: 10.1016/j.eswa.2013.01.066.

[3] Abellán, J. and Masegosa, A.R. (2012a). Bagging schemes on the presence of class noise in classification. *Expert Systems with Applications,*, 39(8), 6827–6837. doi: 10.1016/j.eswa.2012.01.013.

[4] Abellán, J. and Masegosa, A.R. (2012b). Imprecise classification with credal decision trees. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 20(5), 763–787. doi: 10.1142/S0218488512500353.

[5] Abellán, J. and Moral, S. (2003), Building classification trees using the total uncertainty criterion. *International Journal of Intelligent Systems*, 18(12), 1215–1225. doi: 10.1002/int.10143.

[6] 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(1), 112–122. doi: 10.1016/j.ejor.2011.01.020.

[7] 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. doi: 10.1016/j.csda.2013.02.009.

[8] Abellán, J., Mantas, C.J. and Castellano, J.G. (2017). A Random Forest approach using imprecise probabilities. *Knowledge-Based Systems,* 134, 72–84. doi: 10.1016/j.knosys.2017.07.019.

[9] Aboalkhair, A.M.A. (2012). *Nonparametric Predictive Inference for System Reliability.* PhD thesis, Durham University, United Kingdom. https://etheses.dur.ac.uk/3918/.

[10] Aitken, A.C. (1936). On least squares and linear combination of observations. *Proceedings of the Royal Society of Edinburgh,* 55, 42–48. doi: 10.1017/S0370164600014346.

[11] Alcántara, A., Galván, I.M. and Aler, R. (2022). Direct estimation of prediction intervals for solar and wind regional energy forecasting with deep neural networks. *Engineering Applications of Artificial Intelligence,* 114, 105128. doi: 10.1016/j.engappai.2022.105128.

[12] Alharbi, A.A. (2022). *Direct Nonparametric Predictive Inference Classification Trees.* PhD thesis, Durham University, United Kingdom. http://etheses.dur.ac.uk/14473/.

[13] Arts, G.R.J., Coolen, F.P.A. and Laan, P.van der. (2004). Nonparametric predictive inference in statistical process control. *Quality Technology & Quantitative Management,* 1(2), 201–216. doi: 10.1080/16843703.2004.11673073.

[14] Augustin, T. (2002). Neyman-Pearson testing under interval probability by globally least favorable pairs; Reviewing Huber-Strassen theory and extending it to general interval probability. *Journal of Statistical Planning and Inference*, 105(1), 149–173. doi: 10.1016/S0378-3758(01)00208-7.

[15] Augustin, T. and Coolen, F.P.A. (2004). Nonparametric predictive inference and interval probability. *Journal of Statistical Planning and Inference*, 124(2), 251–272. doi: 10.1016/j.jspi.2003.07.003.

[16] Augustin, T., Coolen, F.P.A., de Cooman, G. and Troffaes, M.C.M. (2014). *Introduction to Imprecise Probabilities.* John Wiley & Sons, New York.

[17] Baker, R.M. (2010). *Multinomial Nonparametric Predictive Inference: Selection, Classification and Subcategory data*. PhD thesis, Durham University, United Kingdom. http://etheses.dur.ac.uk/257/.

[18] Bishop, C.M. (1995). *Neural Networks for Pattern Recognition.* Oxford university press, Oxford.

[19] Boole, G. (1854). *An Investigation of the Laws of Thoughts on Which Are Founded the Mathematical Theories of Logic and Probabilities.* Walton and Maberly, London. doi: 10.1017/CBO9780511693090.

[20] Boutell, R.M., Luo, J., Shen, X. and Brown, M.C. (2004). Learning multi-label scene classification. *Pattern Recognition,* 37(9), 1757–1771. doi: 10.1016/j.patcog.2004.03.009.

[21] Breiman, L. (1996). Bagging predictors. *Machine Learning,* 24(2), 123–140. doi: 10.1007/BF00058655.

[22] Breiman, L. (2001). Random forests. *Machine Learning,* 45, 5–32. doi: 10.1023/A:1010933404324.

[23] Breiman, L., Cutler, A., Liaw, A. and Wiener, M. (2022). *randomForest: Breiman and Cutlers Random Forests for Classification and Regression.* R package version 4.7-1.1 https://CRAN.R-project.org/package=randomForest.

[24] Breiman, L., Friedman, J., Olshen, R.A. and Stone, C.J. (1984). *Classification and Regression Trees.* Chapman & Hall, New York. doi: 10.1201/9781315139470.

[25] Charte, F. and Rivera, A.J. (2019). *mldr.datasets: R Ultimate Multilabel Dataset Repository.* R package version 0.4.2 https://CRAN.R-project.org/package=mldr.datasets.

[26] Cherubini, U., Luciano, E. and Vecchiato, W. (2004). *Copula Methods in Finance.* John Wiley & Sons, Chichester. doi: 10.1002/9781118673331.

[27] Clare, A. and King, R.D. (2001). Knowledge discovery in multi-label phenotype data. In *Principles of Data Mining and Knowledge Discovery,* vol 2168. Springer, Berlin, Heidelberg, pp. 42–53. doi: 10.1007/3-540-44794-6˙4.

[28] Clayton, D.G. (1978). A model for association in bivariate life tables and Its application in epidemiological studies of familial tendency in chronic disease incidence. *Biometrika,* 65(1), 141–151. doi: 10.2307/2335289.

[29] Coolen-Maturi, T., Coolen, F.P.A. and Muhammad, N. (2016). Predictive inference for bivariate data: combining nonparametric predictive inference for marginals with an estimated copula. *Journal of Statistical Theory and Practice*, 10(3), 515–538. doi: 10.1080/15598608.2016.1184112.

[30] Coolen, F.P.A. (1998). Low structure imprecise predictive inference for Bayes' problem. *Statistics and Probability Letters*, 36(4), 349–357. doi: 10.1016/S0167-7152(97)00081-3.

[31] Coolen, F.P.A. (2006). On nonparametric predictive inference and objective Bayesianism. *Journal of Logic, Language and Information*, 15, 21–47. doi: 10.1007/s10849-005-9005-7.

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

[33] Coolen, F.P.A. and Augustin, T. (2005). Learning from multinomial data: a nonparametric predictive alternative to the Imprecise Dirichlet Model. In *ISIPTA'05: Proceedings of the Fourth International Symposium on Imprecise Probabilities and Their Applications,* vol 5, pp. 125–134.

[34] 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(2), 217–230. doi: 10.1016/j.ijar.2008.03.011.

[35] Coolen, F.P.A. and van der Laan, P. (2001). Imprecise predictive selection based on low structure assumptions. *Journal of Statistical Planning and Inference,* 98(1–2), 259–277. doi: 10.1016/S0378-3758(00)00313-X.

[36] Coolen, F.P.A. and Yan, K.J. (2004). Nonparametric predictive inference with right-censored data. *Journal of Statistical Planning and Inference*, 126, 25–54. doi: 10.1016/j.jspi.2003.07.004.

[37] Coolen, F.P.A., Coolen-Schrijner, P., Coolen-Maturi, T. and Elkhafifi, F.F. (2013). Nonparametric predictive inference for ordinal data. *Communications in Statistics - Theory and Methods*, 42(19), 3478–3496. doi: 10.1080/03610926.2011.632104.

[38] Corani, G. and Zaffalon, M. (2008). Learning reliable classifiers from small or incomplete data sets: the naive credal classifier 2. *Journal of Machine Learning Research,* 9, 581–621.

[39] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning,* 20, 273–297. doi: 10.1007/BF00994018.

[40] Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. In *IEEE Transactions on Information Theory*, 13(1), 21–27. doi: 10.1109/TIT.1967.1053964.

[41] Cox, D.R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, 20(2), 215–242.

[42] Cozman, F. and Moral, S. (2000). Reasoning with imprecise probabilities. *International Journal of Approximate Reasoning*, 24(2–3), 121–299. doi: 10.1016/S0888-613X(00)00030-X.

[43] de Campos, L.M., Huete, J.F. and Moral, S. (1994). Probability intervals: a tool for uncertain reasoning. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 2(2), 167–196. doi: 10.1142/S0218488594000146.

[44] de Cooman, G. (2000). Imprecise probabilities. *Risk, Decision and Policy*, 5(2), 107–181. doi: 10.1017/S135753090000017X.

[45] Dempster, A.P. (1967). Upper and lower probabilities included by a multi-valued mapping. *The Annals of Mathematical Statistics*, 38(2), 325–339. doi: 10.1214/aoms/1177698950.

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

[47] Dietterich, T.G. (2000a). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, Boosting, and Randomization. *Machine Learning*, 40, 139–157. doi: 10.1023/A:1007607513941.

[48] Dietterich, T.G. (2000b). Ensemble methods in machine learning. In *Multiple Classifier Systems,* vol 1857. Springer, Berlin, Heidelberg. doi: 10.1007/3-540-45014-9˙1.

[49] Diplaris, S., Tsoumakas, G., Mitkas, P.A. and Vlahavas, I. (2005). Protein classification with multiple algorithms. In *Advances in Informatics,* vol 3746. Springer, Berlin, Heidelberg, pp. 448–456. doi: 10.1007/11573036˙42.

[50] Duygulu, P., Barnard, K., de Freitas, J.F.G. and Forsyth, D.A. (2002). Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *Computer Vision,* vol 2353. Springer, Berlin, Heidelberg, pp. 97–112. doi: 10.1007/3-540-47979-1˙7.

[51] Elisseeff, A. and Weston, J. (2001). A Kernel method for multi-labelled classification. In *Proceedings of the Annual ACM Conference on Research and Development in Information Retrieval*, pp. 681–687.

[52] 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. doi: 10.1080/15598608.2012.719802.

[53] Esposito, F., Malerba, D. and Semeraro, G. (1997). A comparative analysis of methods for pruning decision trees. In *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 19(5), pp. 476-491. doi: 10.1109/34.589207.

[54] Fayyad, U. and Irani, K. (1993). Multi-valued interval discretization of continuous-valued attributes for classification learning. In *Joint Conference on Artificial Intelligence*, pp. 1022–1027.

[55] Fernández-Delgado, M., Cernadas, E., Barro, S. and Amorim, D. (2014). Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? *Journal of Machine Learning Research,* 15, 3133–3181.

[56] Fink, P. (2018). *imptree: Classification Trees with Imprecise Probabilities.* R package version 0.5.1. https://CRAN.R-project.org/package=imptree.

[57] Frank, M.J. (1979). On the simultaneous associativity of $F(x, y)$ and $x+y-F(x, y)$. *Aequationes Mathematicae,* 19, 194–226. doi: 10.1007/BF02189866.

[58] Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics,* 11(1), 86–92. doi: 10.1214/aoms/1177731944.

[59] Genest, C. and Favre, A.C. (2007). Everything you always wanted to know about copula modeling but were afraid to ask. *Journal of Hydrologic Engineering,* 12(4), 347–368. doi: 10.1061/(ASCE)1084-0699(2007)12:4(347).

[60] Geurts, P., Ernst, D. and Wehenkel. L. (2006). Extremely randomized trees. *Machine learning,* 63, 3–42. doi: 10.1007/s10994-006-6226-1.

[61] Ghamrawi, N. and McCallum, A. (2005). Collective multi-label classification. In *International Conference on Information and Knowledge Management,* pp. 195–200. doi: 10.1145/1099554.1099591.

[62] Gneiting T. and Raftery, A.E. (2007) Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477), 359–378. doi: 10.1198/016214506000001437.

[63] Godbole, S. and Sarawagi, S. (2004). Discriminative methods for multi-labeled classification. In *Advances in Knowledge Discovery and Data Mining*, vol 3056. Springer, Berlin, Heidelberg, pp. 22–30. doi: 10.1007/978-3-540-24775-3˙5.

[64] Gumbel, E.J. (1960). Distributions de valeurs extrêmes en plusieurs dimensions. *Publications de l'Institute de Statistique de l'Université de Paris*, 9, 171–173.

[65] Gupta, B., Rawat, A., Jain, A., Arora, A. and Dhami, N. (2017) . Analysis of various decision tree algorithms for classification in data mining. *International Journal of Computer Applications,* 163(8), 15–19. doi: 10.5120/ijca2017913660.

[66] Hahsler, M., Buchta, C., Gruen, B. and Hornik, K. (2023). *arules: Mining Association Rules and Frequent Itemsets.* R package version 1.7-6 https://CRAN.R-project.org/package=arules.

[67] Hampel, F. (1997). What can the foundation discussion contribute to data analysis? And what may be some of the future directions in robust methods and data analysis? *Journal of Statistical Planning and Inference*, 57(1), 7–19. doi: 10.1016/S0378-3758(96)00031-6.

[68] Hastie, T., Tibshirani, R. and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Second Edition, Springer Series in Statistics.

[69] Hill, B.M. (1968). Posterior distribution of percentiles: Bayes' theorem for sampling from a population. *Journal of the American Statistical Association*, 63(322), 677–691. doi: 10.2307/2284038.

[70] Hill, B.M. (1988). De Finetti's Theorem, Induction, and $A_{(n)}$ or Bayesian nonparametric predictive inference (with discussion). *Bayesian Statistics*, vol 3. Oxford University Press. pp. 211–241.

[71] Hill, B.M. (1993). Parametric models for A_n: Splitting processes and mixtures. *Journal of the Royal Statistical Society: Series B (Methodological)*, 55(2), 423–433. doi: 10.1111/j.2517-6161.1993.tb01912.x.

[72] Huber, P.J. (1964). Robust estimation of a location parameter. *The Annals of Mathematical Statistics,* 35(1), 73–101. doi: 10.1214/aoms/1177703732.

[73] Huber, P.J. and Strassen, V. (1973). Minimax tests and the Neyman-Pearson lemma for capacities. *The Annals of Statistics*, (1)2, 251–263. doi: 10.1214/aos/1176342363.

[74] Katakis, I., Tsoumakas, G. and Vlahavas, I. (2008). Multilabel text classification for automated tag suggestion. In *Proceedings of the ECML/PKDD Discovery Challenge,* pp. 75–83.

[75] Kelly, M., Longjohn, R. and Nottingham, K. (2025). *The UCI Machine Learning Repository.* Available: https://archive.ics.uci.edu.

[76] Khosravi, A., Nahavandi, S. and Creighton, D. (2010). A prediction interval-based approach to determine optimal structures of neural network metamodels. *Expert Systems with Applications*, 37(3), 2377–2387. doi: 10.1016/j.eswa.2009.07.059.

[77] Klimt, B., Yang, Y. (2004). The Enron corpus: A new dataset for email classification research. *Machine Learning,* vol 3201. Springer, Berlin, Heidelberg, Berlin. pp. 217–226. doi: 10.1007/978-3-540-30115-8˙22.

[78] 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,* vol 2, pp. 1137–1143.

[79] Kole, E., Koedijk, K. and Verbeek, M. (2007). Selecting copulas for risk management. *Journal of Banking & Finance*, 31(8), 2405–2423. doi: 10.1016/j.jbankfin.2006.09.010.

[80] Madjarov, G., Kocev, D., Gjorgjevikj, D. and Džeroski, S. (2012). An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition,* 45(9), 3084–3104. doi: 10.1016/j.patcog.2012.03.004.

[81] Mantas, C.J., Abellán, J. (2014). Credal-C4.5: Decision tree based on imprecise probabilities to classify noisy data. *Expert Systems with Applications,* 41(10), 4625–4637. doi: 10.1016/j.eswa.2014.01.017.

[82] Moral-García, S. and Abellán, J. (2024). Lazy multi-label classification algorithms based on non-parametric predictive inference. *Expert Systems with Applications,* 256(5), 124921. doi: 10.1016/j.eswa.2024.124921.

[83] Moral-García, S., Mantas, J.C., Castellano, G.J. and Abellán, J. (2020). Non-parametric predictive inference for solving multi-label classification. *Applied Soft Computing,* 88, 106011. doi: 10.1016/j.asoc.2019.106011.

[84] Moral, S., Mantas, C.J., Castellano, J.G. and Abellán, J. (2020). Imprecise classification with non-parametric predictive inference. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems. IPMU 2020. Communications in Computer and Information Science,* vol 1238. Springer, Cham. doi: 10.1007/978-3-030-50143-3˙5.

[85] Muhammad, N. (2016). *Predictive Inference with Copulas for Bivariate Data.* PhD thesis, Durham University, United Kingdom. https://etheses.dur.ac.uk/11597/.

[86] Nagler, T., Schepsmeier, U., Stoeber, J., and Brechmann, E.C., Graeler, B. and Erhardt, T. (2023). *VineCopula: Statistical Inference of Vine Copulas.* R package version 2.5.0 https://CRAN.R-project.org/package=VineCopula.

[87] Nelsen, R.B. (2006). *An Introduction to Copulas.* Springer Series in Statistics, New York.

[88] Nemenyi, P. (1963). *Distribution-Free Multiple Comparisons.* Doctoral dissertation, Princeton University, New Jersey, USA.

[89] Olive, D.J. (2007). Prediction intervals for regression models. *Computational Statistics & Data Analysis*, 51(6), 3115–3122. doi: 10.1016/j.csda.2006.02.006.

[90] Pang, J., Liu, D., Peng, Y. and Peng, X. (2018). Optimize the coverage probability of prediction interval for anomaly detection of sensor-based monitoring series. *Sensors,* 18(4), 967. doi: 10.3390/s18040967.

[91] Pestian, J.P., Brew, C., Matykiewicz, P., Hovermale, D.J., Johnson, N., Cohen, B. and Duch, W. (2007). A shared task involving multi-label classification of clinical free text. In *Biological, translational, and clinical language processing, Association for Computational Linguistics,* pp. 97–104.

[92] Powell. J.L. (1984). Least absolute deviations estimation for the censored regression model. *Journal of Econometrics,* 25(3), 303–325. doi: 10.1016/0304-4076(84)90004-6.

[93] Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning*, 1, 81–106. doi: 10.1007/BF00116251.

[94] Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, California.

[95] R Core Team (2023). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/.

[96] Racine, J.S. and Hayfield, T. (2024). *np: Nonparametric Kernel Smoothing Methods for Mixed Data Types.* R package version 0.60.18 https://CRAN.R-project.org/package=np.

[97] Rajaraman, A. and Ullman, J.D. (2011). *Mining of Massive Datasets.* Cambridge University Press, Cambridge.

[98] Read, J., Pfahringer, B., Holmes, G. and Frank, E. (2009). Classifier chains for multi-label classification. In *Machine Learning and Knowledge Discovery in Databases,* vol 5782. Springer, Berlin, Heidelberg, pp. 254–269. doi: 10.1007/978-3-642-04174-7˙17.

[99] Schapire, R.E. and Singer, Y. (2000). Boostexter: A boosting-based system for text categorization. *Machine Learning,* 39, 135–168. doi: 10.1023/A:1007649029923.

[100] Shrestha, D.L. and Solomatine, D.P. (2006). Machine learning approaches for estimation of prediction interval for the model output. *Neural Networks*, 19(2), 225–235. doi: 10.1016/j.neunet.2006.01.012.

[101] Silverman, B.W. (1986). *Density Estimation for Statistics and Data Analysis.* Chapman & Hall, London.

[102] Sklar, A. (1959). Fonctions de répartition à n-dimensions et leurs marges. *Publications de l'Institut de Statistique de l'Université de Paris*, 8, 229–231.

[103] Tang X.-S., Li, D.-Q. Zhou, C.-B. and Zhang L.-M. (2013). Bivariate distribution models using copulas for reliability analysis. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability,* 227(5), 499–512. doi: 10.1177/1748006X13481928.

[104] Ting, K.M. (2002). An instance-weighting method to induce cost-sensitive trees. In *IEEE Transactions on Knowledge and Data Engineering*, 14(3), 659–665. doi: 10.1109/TKDE.2002.1000348.

[105] Tsoumakas, G. and Katakis, I. (2007). Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3), 1–13. doi: 10.4018/jdwm.2007070101.

[106] Tsoumakas, G. and Vlahavas, I. (2007). Random k-Labelsets: An ensemble method for multilabel classification. In *Machine Learning*, vol 4701. Springer, Berlin, Heidelberg, pp. 406–417. doi: 10.1007/978-3-540-74958-5˙38.

[107] Walley, P. (1996). Inferences from multinomial data: learning about a bag of marbles. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1), 3–57.

[108] Wang, H., Yan, L., Huang, H. and Ding, C. (2017). From protein sequence to protein function via multi-label linear discriminant analysis. In *IEEE/ACM Transactions on Computational Biology and Bioinformatics,* 14(3), 503–513. doi: 10.1109/TCBB.2016.2591529.

[109] Wang, Q., Ma, Y., Zhao, K. and Tian, Y. (2022). A comprehensive survey of loss functions in machine learning. *Annals of Data Science,* 9, 187–212. doi: 10.1007/s40745-020-00253-5.

[110] Weichselberger, K. (2000). The theory of interval-probability as a unifying concept for uncertainty. *International Journal of Approximate Reasoning*, 24(2–3), 149–170. doi: 10.1016/S0888-613X(00)00032-3.

[111] Wolpert, D.H. (2002). The supervised learning no-free-lunch theorems. In *Soft Computing and Industry*, Springer, London. pp. 25–42. doi: 10.1007/978-1-4471-0123-9˙3.

[112] Yager, R.R., Kacprzyk. J. and Fedrizzi, M. (1994). *Advances in the Dempster-Shafer Theory of Evidence.* John Wiley & Sons, USA.

[113] Zaffalon, M. (2002). The naive credal classifier. *Journal of Statistical Planning and Inference*, 105(1), 5–21. doi: 10.1016/S0378-3758(01)00201-4.

[114] Zawadzki, Z. and Kosinski, M. (2023). *FSelectorRcpp: 'Rcpp' implementation of 'FSelector' entropy-based feature selection algorithms with a sparse matrix support.* R package version 0.3.11 https://CRAN.R-project.org/package=FSelectorRcpp.

[115] Zhang, M.-L. and Zhou, Z.-H. (2006). Multilabel neural networks with applications to functional genomics and text categorization. In *Transactions on Knowledge and Data Engineering,* 18 (10), 1338–1351. doi: 10.1109/TKDE.2006.162.

[116] Zhang, M.-L. and Zhou, Z.-H. (2014). A review on multi-label learning algorithms. In *IEEE Transactions on Knowledge and Data Engineering,* 26(8), 1819–1837. doi: 10.1109/TKDE.2013.39.