University of Exeter
Department of Computer Science

Diversity and generalisation error in classification ensembles

Carina Ivaşcu

June, 2023

Supervised by Professor Jonathan Fieldsend
and Professor Richard Everson

Submitted by Carina Ivaşcu, to the University of Exeter as a thesis for the degree of Doctor of Philosophy in Computer Science , June, 2023.

I certify that all material in this thesis which is not my own work has been identified and that no material has previously been submitted and approved for the award of a degree by this or any other University.

(signature) .....

Abstract

Ensembles are important tools in machine learning because they are often more accurate than single predictors. Although it has been shown that an accurate ensemble would benefit from having both accurate and diverse predictors, some studies in the literature could not support the influence that diversity has on the overall accuracy of an ensemble. In this thesis we are investigating the influence that diversity has on improving accuracy or equivalently reducing the generalisation error.

There have been many diversity measures introduced in the literature, however as outlined in [1] the only one that had a strong negative correlation with generalisation error, was a diversity measure called ambiguity. The ambiguity measure was obtained by using the bias-variance decomposition of classifiers along with the 0-1 loss. As a result, our first set of experiments focuses on this type of diversity measure. We analyse the effect that the ambiguity measure has on decreasing the generalisation error of forests created by bootstrapping. We compare the effect of the ambiguity by having bootstrapping with or without replacement, by varying the number of trees, by varying the patterns or features used in building each tree. Our results show that bootstrapping without replacement yields lower test errors. A similar effect has been seen on bigger ensembles or by providing more data to the classifiers. We propose pruning approaches that involve ambiguity and compare their effect on the generalisation error versus a pruning method that promotes randomness. Our results show that there is no significant difference between the two types of approaches.

Next, we define two new ambiguity measures derived from the cross entropy and hinge loss. We analyse their properties and find that out of the three ambiguity measures defined for classifiers (including the 0-1 loss introduced earlier), the only one that achieves all the desired properties of a diversity measure is the one obtained from the cross entropy (being always positive, and zero if and only if all the classifiers agree). We build ensembles by using bagging and by varying the sampling rates, we find that there is a negative correlation between generalisation error and diversity at high sampling rates; conversely generalisation error is positively correlated with diversity when the sampling rate is low and the diversity high. We use an evolutionary algorithm in order to maximise ambiguity and we find that the evolved ensemble in general has lower generalisation error than the initial ensemble. We define the term “ambiguous ensembles” as ensembles with high values of ambiguity. Additionally, we investigate the effect of pruning on larger ensembles and propose several pruning methods that prioritize ambiguity, as well as others that promote

less ambiguous ensembles. Our results show that the approaches the prefer ambiguous ensembles reduce the generalisation error. Hence, our overall results support the influence that the diversity has on minimising generalisation error.

Finally, we define diverse forests by building trees with different impurities. We choose families of impurities which are characterized by different parameters and we analyse the effect of choosing different parameters has on the generalisation performance. By tuning the parameters we can define symmetric or asymmetric impurities. In the case of imbalanced datasets the use of asymmetric impurities has been proven beneficial in predicting the minority class which usually is of big interest. We contrast the behaviour of the forests by using symmetric, asymmetric impurities with forests of trees built with different impurities (different parameters). Our results do not show a significant difference in performance.

To my mother

Acknowledgements

I am very grateful to many people I have encountered throughout my PhD journey. Without their help, support and enthusiasm the PhD journey would have been much more difficult.

First, I would like to express my gratitude to my supervisors, profs Jonathan Fieldsend and Richard Everson, for their continuous help, support and guidance. It was a amazing opportunity for me to be guided by such wonderful academics.

I have been blessed with many friends who have supported me throughout this period. I would like to thank Aylin, Cristina, Canan and Cynthia for the lovely memories we spent together throughout our PhD journeys. Such bonds cannot be forgotten and will be life-long friendships.

I had the pleasure of sharing the office with wonderful colleagues. I would like to thank Tim, John, Amjad, Aziz, Ola for being such helpful, kind and amazing colleagues.

I am extremely grateful to my mother and grandmother, to which I owe everything in my life and without their sacrifices, I wouldn't have been writing this thesis now.

Last but not least, I would like to thank God and all the saints that helped me in my life and eased my way to achieving this career goal.

Contents

List of Figures

List of Tables

The list of the symbols used is the following:

Symbol	Meaning
M	number of classifiers
N	number of patterns
X	datapoints
x_n	n^{th} pattern
x'_n	n^{th} feature
t	target vector
$t_n \in \{0, 1\}$	n^{th} target
c_i, $i \in \{1..M\}$	weights of the classifiers
$y_{mn} \equiv y_m(x_n)$	prediction of the m^{th} classifier of the n^{th} pattern
y_m	the m^{th} classifier
$Y_n \equiv Y(\mathbf{x}_n) = \sum_{i=1}^{M} c_i y_{in}$	ensemble prediction of the n^{th} pattern
$\mathcal{Y}_n = \{y_{in} = y_i(\mathbf{x}_n)\}_{i=1}^{M}$	outputs of the ensemble members
$\mathcal{Y}$	ensemble
$L_{01}(Y_n \cdot t_n) = \begin{cases} 0 & \text{if } Y_n \cdot t_n \geq 0 \\ 1 & \text{if } Y_n \cdot t_n < 0. \end{cases}$	ensemble 0-1 loss for the n^{th} target
$\text{amb}_{01}(\mathcal{Y}_n) = \frac{1}{2} \sum_{i=1}^{M} (\frac{1}{M} Y_n - c_i y_{in}) t_n$	ensemble ambiguity derived from the 0-1 loss
$L_{log}(Y_n, t_n) = -[t_n \log(Y_n) + (1 - t_n) \log(1 - Y_n)]$	ensemble log loss
$\text{amb}_{CE}(\mathcal{Y}_n) = \sum_{i=1}^{M} c_i L_{log}(y_{in}, t_n) - L_{log}(Y_n, t_n)$	ensemble ambiguity derived from the log loss
$L_H(Y_n, t_n) = \max(0, 1 - t_n Y_n)$	ensemble hinge loss
C^n	class of functions with continuous nth derivative

Publications

The materials presented in chapters 3, 4 have been published in:

Carina Ivaşcu, Richard M Everson, and Jonathan E Fieldsend. Optimising diversity in classifier ensembles of classification trees. In Applications of Evolutionary Computation: 24th International Conference, EvoApplications 2021, Held as Part of EvoStar 2021, Virtual Event, April 7–9, 2021, Proceedings 24, pages 634–648. Springer, 2021. 128

The materials presented in chapter 5 have been published in:

Carina Ivaşcu, Richard M Everson, and Jonathan E Fieldsend. Optimising diversity in classifier ensembles. SN Computer Science, 3(3):191, 2022.

Chapter 1

Introduction

1.1 Supervised learning

Machine learning is a popular field, widely used in our daily lives. From applications ranging from voice or facial recognition to image processing or medical diagnosis, machine learning is becoming a useful tool in our lives.

Supervised learning is a branch of machine learning that aims to find a mapping between some input and output data [2]. According to the type of output data, supervised learning is divided into to two sub-categories: classification and regression. Supervised learning is used to answer questions such as: "How the weather is going to be tomorrow?", "Is the patient likely to have a specific illness?", "Are the property prices going to increase next year, taking into account the economic fluctuation?", etc. Classification problems are concerned with assigning the input vectors to a finite number of discrete categories or classes. On the other hand, in regression problems the output contains one or more continuous variables [3].

Machine learning techniques involve training a model to "learn" the patterns of the data, in order to be able to recognise similar patterns on new, unseen data (called test data), action called generalisation. A model is assessed according to its predictive capability, which is quantified by the generalisation ability from the training patterns to the test data.

The unseen data is supposed to have the same distribution as the training data, so that the patterns learned during the training stage can be identified in the test data. If the test data does not follow the same distribution, the model might fail to recognize the data patterns correctly and may not generalise well [3],[4], [5], [6]. One way of ensuring that training and test data have the same distribution, is by generating them from the same data by using stratified k-fold cross validation [5],[7]. Formally in the case of supervised, we can define a model in the following way:

Let f be a model, defined as a function $f : X \to Y$, where X represents the data, as a sequence of examples and Y the true classes of the examples (in the classification case), or the targets (regression case). There can be many mappings that associate the data X with

the targets Y, let us denote the family of models as $\mathcal{F}$. The functions f belonging to the family are defined on a set of random independent identically distributed examples drawn from a probability distribution $P(x, y)$, which is unknown, where $x \in X$ and $y \in Y$. In the classification case, Y is a discrete set, whereas in the regression case it is a continuous set. In this thesis we will focus on classification.

Given all these possible models, the question arises, which model to choose? In order to answer this question, we will evaluate each of these candidates models by using a loss function, L. A loss (error) function quantifies the amount of incorrectly classified examples or patterns. The best candidate model will be the one with the lowest average loss. The average loss of a classifier can be defined in the following way:

$$R[f] = \int_{X,Y} L(y, f(x))P(x, y)dxdy \tag{1.1}$$

where R is called the expected risk. Obviously, the best classifier will be the one that minimises this quantity:

$$f^* = \arg\min_{f \in \mathcal{F}} R[f] \tag{1.2}$$

Since the distribution P is unknown, another metric in order to quantify the performance of a model was defined, called the empirical risk minimization. The empirical risk minimization method was firstly introduced in [8] and evaluates the loss on the training data:

$$R_{erm}[f] = \frac{1}{N} \sum_{i=1}^{N} L(y, f(x_n)) \tag{1.3}$$

The empirical risk is an approximation to R because the data samples are presumed to be drawn from $P(x, y)$. The empirical risk minimization method is the closest metric we have for assessing the model's performance from the training process, since the test data is unknown and the distributions of the two types of data are known.

1.2 Ensembles of classifiers

Obviously, the most important feature of a model is its prediction capabilities. Another technique that has been shown to produce successful predictions is called ensemble aggregation. An ensemble is a collection of classifiers, which combines their predictions into one. It has been shown that a collection of classifiers performs better than a single one [9]. Ensemble predictions are obtained by combining the predictions of individual classifiers. The most popular methods used in classification are majority voting or weighted voting.

Majority voting implies that each predictor assigns a class to a specific pattern and the class that gets the most votes, will be the ensemble's prediction for that particular example. Mathematically, majority voting can be defined as:

$$Y_n = argmax_{class} \sum_{i=1}^{M} I(y_{in} = class) \tag{1.4}$$

where Y_n is the ensemble prediction and y_{in} is the prediction of the ith classifier, both for the nth pattern, M is the number of classifiers and I is the identity function.

Weighted voting requires assigning different weights to the predictions of individual base models before combining them. The ensemble prediction via weighted voting can be defined in the following way:

$$Y_n = \sum_{i=1}^{M} c_i y_{in} \tag{1.5}$$

where c_i are the non-negative weights, that sum up to 1. The aggregating of the classifiers' predictions is illustrated in Figure 1.1.

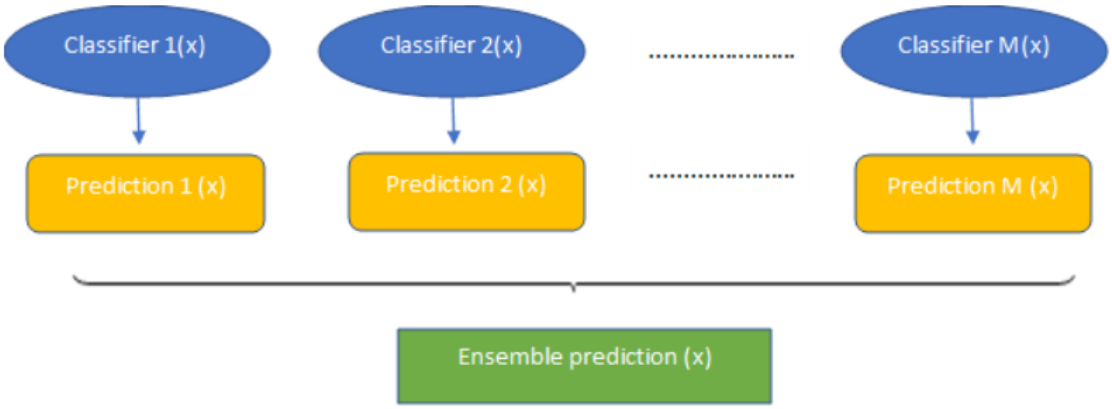

Figure 1.1. Schematic representation of ensemble predictions

Ensembles were introduced by Hansen and Salamon in [9], who have shown the improvement in generalisation performance in an ensemble of neural networks. Many successful ensemble techniques have been suggested in literature: bagging [10], boosting [11], random forests [12] and negative correlation learning [13]. Bagging involves sampling different parts of the data and feeding it to the classifiers in order to produce different models. Afterwards their predictions are averaged. Data can be divided into rows (patterns) or columns (features). In the case of bagging, all features are considered, whereas the patterns are randomly selected. Boosting gradually builds ensembles by training the models in multiple iterations. At each iteration the examples that are misclassified at an earlier stage are given higher priority. Random forests on the other hand, train the trees with random subsets of features. Each of these ensemble methods have been used in real-world applications. For example, random forests and boosting techniques have been used intensively in classification and regression problems, with applications in the field of bioinformatics [14], medical research [15], [16], forecasting stock prices [17], [18] etc. Neural-networks have also been used successfully in deep learning problems, such as: object detection [19], handwritten digit recognition [20], etc. Even though all these ensemble methods can be suitable for machine learning tasks, the characteristics of the data, the computational resources might lead to some of these ensemble techniques to be preferred.

1.3 Accuracy and diversity

In the previous section, we emphasized the importance of ensembles. However, one question that arises is how to select the component classifiers? It has been shown that an ensemble with good generalisation performance should contain members which differ in their predictions [21],[22]. Therefore, accuracy and diversity are main features to take into account when building an ensemble.

It has also been shown that ensembles containing a large number of classifiers achieve a good generalisation performance. However the computational time necessary to produce the prediction is influenced by the number of models in the ensemble, therefore having a large number of predictors is not always beneficial.

In this thesis, we will explore the effect that different diversity measures have on the generalisation error of small ensembles. Our experiments show that the benefit of having a larger ensemble is not significantly higher in terms of generalisation error, compared to the one of a small ensemble. We have analysed diversity measures defined on the input or output space. Diversity measures defined on the input space quantify differences amongst the data selected to build the predictors. Whereas diversity measure defined on the output space measure the differences between the models' predictions.

There is no universal formula for diversity, therefore many types of diversities have been introduced in the literature. Some studies could not support the influence that diversity has on generalisation error, e.g. [23]. This aspect was explained in [1] which found that different diversity measures have a different correlation level with the test error. The authors found that there is a high correlation between diversity and generalisation error when diversity is small, however the correlation decreases after diversity surpasses a certain threshold.

A new diversity measure based on the ambiguity decomposition of regression ensembles and the bias-variance decomposition was introduced in [21]. Based on this idea, we will introduce new diversity measures, also called ambiguities. We will explore their theoretical characteristics and evaluate their use empirically.

1.4 Research Questions

This thesis focuses on the effect that diversity has on reducing the generalisation error. We have analysed the correlation of different diversity measures with the test error and have introduced two new diversity measures based on the bias-variance decomposition, by using the cross-entropy or hinge losses. We investigate the effect that the ambiguity obtained from the cross-entropy (amb_{CE}) has on reducing the generalisation error, with the aid of an evolutionary algorithm. We have conducted our analysis in the case of random forests, due to their high generalisation performance. Next, we suggest pruning methods that involve the ambiguity based on the cross-entropy measure and study the effects on the generalisation error. Finally, we inject diversity in a random forest by building trees with different asymmetric impurities and investigate the impact that it has on the generalisation error. Therefore, the research questions that the thesis focuses on are:

1. What is the correlation between various diversity measures and generalisation error?

2. Which diversity measure has the most impact on generalisation error reduction?

3. Does pruning according to diversity measures have a positive impact on generalisation error?

4. Does injecting diversity in a random forest by building trees with different impurities, contribute to error reduction?

1.5 Contributions of the Thesis

This thesis includes the following contributions to the field of evolutionary optimisation and machine learning.

1. The derivation of two new ambiguity measures, based on the bias-variance decomposition by using the cross-entropy error or hinge loss

2. An analysis of the properties of the above mentioned ambiguities. Out of the three ambiguity measures, only the one derived from the cross-entropy, amb_{CE}, satisfies all the desired properties (is always positive and zero if and only if the predictions of all the constituent classifiers agree). We prove that the ambiguity obtained from the hinge loss, amb_{HL} does not satisfy the property if it is equal to 0, then all classifiers predict the same.

3. An evolutionary algorithm is developed which favours ambiguous ensembles, which empirically is shown to decrease the generalisation error for small ensembles over a range of different classification problems

4. A series of ensemble pruning techniques that incorporate the amb_{CE} diversity measure are introduced, which are shown to be efficient in reducing the generalisation error.

5. An analysis of the effects of building trees with asymmetric impurities, as a method of injecting diversity is provided. Impurity functions are used in building decision trees, by determining the optimal split of a node. Asymmetric impurities bias the predictions of a tree to a specific class and are used in the case of imbalanced datasets.

1.6 Outline of the thesis

This chapter briefly presents some basic knowledge for the subsequent chapters. We have also presented the research questions that this thesis focuses on and the main contributions of the thesis.

Chapter 2 reviews basic concepts from the literature. Section 2.1 describes the concepts of supervised learning. The bias-variance decomposition is described in section 2.2 along with some methods of reducing each of these components. In section 2.3 ensembles are defined, whereas section 2.4 reviews some popular ensemble techniques such as bagging and boosting. Section 2.4.4 presents a popular ensemble method, also used in our experiments, random forests. The main factors in building successful ensembles, accuracy and diversity are presented in section 2.5. Finally, since diversity and accuracy are key features in designing ensembles, a multi-objective approach can be used. Techniques for multi-objective optimisation are presented in section 2.6. Some state of the art algorithms

which involve multi-objective optimisation are summarized in Section 2.7. Another important aspect to consider when building ensembles, is computational costs. While in some cases having a large ensemble would be beneficial in terms of accuracy, it would increase the computational time and memory used. Ensemble reduction techniques, which aim to reduce ensemble size without compromising too much on ensemble accuracy, are presented in section 2.8.

Chapter 3 investigates the correlation between different diversity measures and the generalisation error. We considered the following diversity measures: coincidence failure diversity (CFD) [24], disagreement (DIS) [25], Kohavi-Wolpert (KW) [26], ambiguity [1]. We also introduce a new diversity measure, called coherence, which measures the angle between the predictions of each classifier and the ensemble prediction. Ensembles obtained by using bagging are employed in this investigation. We analyse the relationship between these diversity measures and the generalisation error in ensembles obtained by varying features or patterns, in Section 3.3. Our results empirically demonstrate the superiority of the ambiguity measure defined in [1], which has a negative correlation with the test error. In Section 3.6 we study the effect that different ambiguity based pruning techniques have on the generalisation error. Pruning methods involve reducing the size of an model, by discarding non-essential components which do not have a high impact on the model's performance. Our results show that pruning according to diversity does not yield better generalisation errors than random ensembles.

In Chapter 4 we continue to investigate the correlation between ensemble generalisation error and diversity. We introduce two new ambiguity measures derived from the log loss or hinge loss. We analyse their properties. We characterize the trade off between ensemble training ambiguity and generalisation error on ensembles of decision trees, by using different sampling rates. The sampling rates represent the percentage of the training data that is randomly selected in order to build models. We find that generalisation error is negatively correlated with low diversity at high sampling rates; conversely generalisation error is positively correlated with high diversity and low sampling rates. We performed this experiment on multiple datasets and for each of them there was a different sampling rate for which the lowest test error was achieved. Also, our experiments showed that it is impossible to predict from the training data, the sampling rate that would yield the lowest test error. As a result, we designed an evolutionary algorithm which maximises ambiguity to obtain this rate. Our results show that the generalisation error decreases at the end of the optimisation, which demonstrates empirically the positive effect that ambiguity has on reducing ensemble error.

In the previous chapter our experiments were based on small ensembles, however in some cases a higher number of predictors might make the ensemble more stable and robust to variations in the training data or to noisy data. If individual members make errors on specific patterns due to noise, ensembles having more predictors might overcome the issue. A drawback of having big ensembles, is that a big number of predictors would increase linearly the execution time. As a result, determining the right number of predictors is not straightforward, therefore Chapter 5 focuses on pruning techniques, in order to determine the optimal number of trees. We define pruning methods that favour ambiguity

(the one derived from the log loss, amb_{CE}). We have developed two methods, one that starts with a forest of trees and at each iteration a tree is discarded according to different criteria and one that has a fixed number of trees, m and tries to add and discard a tree at each iteration. We have also designed a new evolutionary algorithm that randomly selects m trees and keeps always the more ambiguous forest. We have compared all these approaches and concluded that the techniques that favour ambiguous ensembles, along with the evolutionary algorithms yielded the best generalisation errors.

Chapter 6 explores a different path of ensuring diversity in a forest. We investigate the effect that building trees with the different impurities has on the ensemble error. We analyse in the case of imbalanced datasets, where the minority class is of more interest. It has been proven that asymmetric impurities that bias the prediction in the direction of the minority class, are more beneficial in the case of imbalanced classes. We compare the behaviour of trees built with asymmetric impurities against forests built with symmetric impurities. Our results show that there is no significant difference between the two approaches.

Chapter 2

Background and related work

This chapter summarizes the literature related to this thesis. Section 2.1 describes a sub-class of machine learning, called supervised learning. Section 2.2 presents the bias-variance trade off and its importance in the prediction process. Section 2.3 describes into more detail the concept of ensembles, while Section 2.4 presents popular ensemble methods. Section 2.5 reviews two key aspects in building successful ensembles, accuracy and diversity. These two factors are successfully combined in multi-objective algorithms, presented in Sections 2.6 and 2.7. Section 2.8 reviews ensemble reduction methods from the literature.

2.1 Supervised learning

One of the aims of machine learning is to develop computer algorithms and techniques "*that are able to learn i.e. to improve automatically through experience*" [2]. Any machine learning technique is formed of two stages: it first selects a candidate model and then in the second stage it estimates the parameters of the model with the aid of available data and a learning algorithm [2]. A widely used type of machine learning is supervised learning. The algorithms belonging to this class select a model that approximates the mapping between the input and output data [2]. Supervised learning is divided into two sub-categories: classification and regression. Both types of supervised learning can be described mathematically as being able to train a machine $\mathcal{F}$, such that:

$$\mathcal{F} : x \rightarrow t$$

$$t = \mathcal{F}(x, w)$$

where x is the input vector, t is the target vector and w are the parameters of the model that can be adjusted to control the behaviour of $\mathcal{F}$ [27]. However, this is not always possible, and F is just an approximation of the required mapping.

Classification problems are concerned with assigning the input vectors to a finite number of discrete categories. On the other hand, in regression problems the output contains one or more continuous variables [3]. Another difference between classification and regression is the fact that they have different error functions. The regression error function is often the

sum of squared errors [27], which is used in conjunction with the assumption of Gaussian distributed noise:

$$L_{MSE}(y) = \frac{1}{2\sigma^2}\sum_{n=1}^{N}(t_n - y(x_n))^2 \tag{2.1}$$

where t_n are the targets, y the model, σ the standard deviation of the samples, x_n the n^{th} pattern.

In the classification case where the targets are discrete, a suitable error function would have to minimize the difference between the predicted values and the true classes. Therefore a good candidate would be the cross entropy error (log loss). In this thesis we only consider binary classification, in which case the output of the supervised machine, $y(x_n)$ is the estimated probability of belonging to one class. The log loss in a binary classification case would be defined as [27]:

$$L_{log}(y) = -\sum_{n=1}^{N}[t_n \log\, y(x_n) + (1 - t_n)\log(1 - y(x_n))] \tag{2.2}$$

In this thesis we will focus on supervised learning, using classification.

Another goal of machine learning techniques is to ensure generalisation. Generalisation is the ability of a learner to perform well when predicting on unseen data [28]. Learning the exact mapping that would fit training data and recognize an unseen pattern is difficult in practice, due to the noise present in the training data.

Since a learning algorithm receives only a subset of training data, the learning goal is to create a statistical model that will fit as much as possible the training data, but at the same time given another related data set (generated by the same process as the training data), the error of the model will not become significantly different [28]. This statement describes two important terms in machine learning: bias and variance, which are the causes of prediction error.

Understanding these two sources of error is important, because by taking measures to minimize them as much as possible, we would obtain more accurate models. However, by trying to reduce one of them, it will increase the other one and vice-versa, also known as the bias and variance trade-off. In the next section we will present these two sources of errors into more detail and also suggest techniques to balance them.

2.2 Bias and variance trade off

The bias and variance decomposition was firstly introduced in [29]. These two quantities were defined in the regression case, by using the mean squared error (see Eq 2.1). Let y

be the approximated function and t the targets. From Equation 2.1 we will obtain:

$$\begin{aligned} MSE(y) &= \mathbb{E}[(y-t)^2] = \mathbb{E}[(y-\mathbb{E}(y)+\mathbb{E}(y)-t)^2] \\ &= \mathbb{E}[(y-\mathbb{E}(y))^2 + 2(y-\mathbb{E}(y))(\mathbb{E}(y)-t) + (\mathbb{E}(y)-t)^2] \\ &= \mathbb{E}[(y-\mathbb{E}(y))^2] + \mathbb{E}[2(y-\mathbb{E}(y))(\mathbb{E}(y)-t)] + \mathbb{E}[(\mathbb{E}(y)-t)^2] \\ &= \mathbb{E}[(y-\mathbb{E}(y))^2] + 2(\mathbb{E}(y)-t)\mathbb{E}(y-\mathbb{E}(y)) + (\mathbb{E}(y)-t)^2 \\ &= \mathbb{E}[(y-\mathbb{E}(y))^2] + 2(\mathbb{E}(y)-t)(\mathbb{E}(y)-\mathbb{E}(y)) + (\mathbb{E}(y)-t)^2 \\ &= \mathbb{E}[(y-\mathbb{E}(y))^2] + (\mathbb{E}(y)-t)^2 \end{aligned}$$

where $\mathbb{E}$ is the expectation over all possible training sets. The above equality proves that the mean squared error is can be expressed as the sum of two terms: the bias and the variance.

The bias is the defined as:

$$Bias^2 = (\mathbb{E}(y)-t)^2 \tag{2.3}$$

The formula for variance is:

$$\mathbb{E}[(y-\mathbb{E}(y))^2] \tag{2.4}$$

The error due to bias is based on the difference between the model suggested by the algorithm (on an average over all data sets and initial conditions) and the correct mapping [28]. The simplifying assumptions made by a model to make the desired mapping easier to learn are a source of bias [30].
If we consider a simple model $\mathcal{F}$, which is independent of the data and differs from t, the bias term will be very high. On the other hand, complex models have a low bias [28].

The variance error refers to the variability of the model for a given pattern, given all possible weight initialization and choice of input parameters. The target function is determined from the training data by a machine learning algorithm, therefore the algorithm it is expected to have some variance. Ideally, the target function should not change too much when switching from one dataset to another, meaning that the algorithm is good at determining the hidden underlying mapping between the inputs and the output variables. Machine learning algorithms with high variance are strongly influenced by the characteristics of the training data. Therefore, the number and types of parameters used to characterize the mapping function are influenced by the specifics of the training data [30]. For a simple model f, the variance will be zero (f being independent of the data), whereas for a complex model the variance will be high [28].
As a result, algorithms that suggest small changes to the estimate of the target function with changes to the training dataset, have low variance. Conversely, algorithms that suggest large changes to the estimate of the target function with changes to the training dataset, have high variance [30].
Figure 2.1 illustrates the relationship between the complexity of the algorithm and the error due to bias and variance.

Similar decompositions have been suggested in the classification case [31], [32], [33] how-

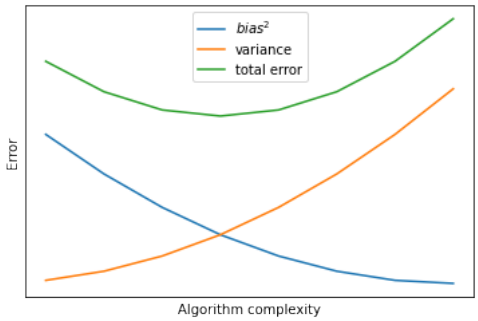

Figure 2.1. Bias and variance relationship

ever none of them provided decompositions similar to the regression case, with the same obvious interpretations (i.e $loss = bias + variance$) [1]. In our work we have also suggested a series of decompositions by using different loss functions.

Methods of reducing bias and variance

Bias and variance can be reduced by several ensemble methods. For example, both of them can be reduced by using boosting algorithms [34], whereas variance can be reduced by using bagging or ensembles [35]. Next we will present the ensemble concept.

2.3 Ensembles

Ensemble methods are collections of predictors, that obtain better prediction performance than a single predictor, by reducing the variance. The models are trained on the same data and their predictions are combined for the final prediction of the ensemble. Figure 2.2 describes how an ensemble is built in the case of classification .

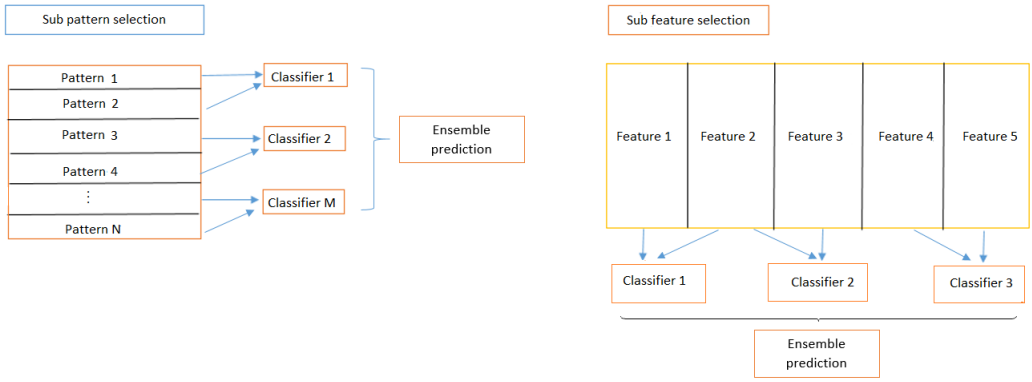

Figure 2.2. The figure on the left illustrates the ensemble prediction process, when subpatterns of the data are selected when creating individual models. The figure on the right denotes the same process when subfeatures are used.

In the next section we will present popular ensemble learning techniques, such as bagging and boosting. We will consider M to be the number of base classifiers and N the total number of patterns.

2.4 Ensemble learning techniques

2.4.1 Boosting

Boosting is an ensemble meta-algorithm that converts weak learners to strong ones. A weak learner is considered a classifier that can label data better than random guessing

[34].

Boosting algorithms contain weak classifiers that learn iteratively and these classifiers are combined to form a strong classifier. After being added, they are weighted according to their accuracy. After the addition of a weak learner, the data are given different weights: the misclassified examples gain weight, whereas correctly classified examples lose weight. As a result, the examples that were misclassified by previous learners will be used the most by future weak learners [34]. In this way the ensemble corrects the errors made by earlier models, reducing bias. Additionally by combining multiple weak learners into a strong model, boosting can also help reduce variance.

A frequently used boosting algorithm is called ADABOOST. The pseudocode for ADABOOST is presented in Algorithm 1, where $I(true) = 1$ and $I(false) = 0$ and $y_j(x_i)$ is the prediction of the j^{th} classifier of the i^{th} pattern, x_i [36]. ADABOOST starts by assigning equal weights to all examples in line 1. At each iteration, a new model is being built according to the current weights. The total error of the current model is being calculated in line 5. If the error is worse than random guessing, then the weights are reinitialised from a uniform distribution and the process starts again (lines 7-9). If the error is less than random guessing, then the amounts of say/importance of the predictor, α_j is being calculated, defined in line 10. After this step the new weights are being calculated (line 11). If a pattern was correctly classified by the model, then the new weights given to this particular datapoint will be $w_{j+1}(i) = w_j(i)e^{\alpha_j}$ and $w_{j+1}(i) = w_j(i)e^{-\alpha_j}$ if it was misclassified. In this way, patterns that were correctly classified are given less importance in the prediction process and the new model will focus more on more difficult to predict examples. Finally, the model with the highest importance will be returned. The test data will be passed to trained ensemble and the final prediction will be made according to majority voting.

Algorithm 1 ADABOOST

Input: $X = \{\mathbf{x}_i\}_{i=1}^{N}$ ▷ training data
Input: $t = \{t_i\}_{i=1}^{N}$ ▷ targets
Input: $NoTrials$
Input: $WeakLearner$ ▷ a weak learner
Output: y_f ▷ ADABOOST classifier

1: $w_1(i) = \frac{1}{N} \forall i$ ▷ initially a uniform distribution
2: **for** $j = 1 \rightarrow NoTrials$ **do**
3: $p_j(i) = \frac{w_j(i)}{\sum_i w_j(i)}$
4: $y_j = WeakLearner(p_j)$
5: $\epsilon_j = \sum_i p_j(i) I(y_j(x_i) \neq t_i)$ ▷ total error
6: **if** $\epsilon_j > 0.5$ **then**
7: restart with uniform weights
8: $w_j(i) = \frac{1}{N}$
9: go to line 3
10: $\alpha_j = \frac{1}{2} \log\left(\frac{1-\epsilon_j}{\epsilon_j}\right)$ ▷ amount of say/importance
11: $w_{j+1}(i) = w_j(i) \begin{cases} e^{\alpha_j}, & \text{if pattern } i \text{ is correctly classified} \\ e^{-\alpha_j}, & \text{otherwise} \end{cases}$ ▷ reinitialize weights
12: **return** $y_f(x) = argmax_{y \in t} \sum_{j=1}^{NoTrials} \alpha_j I(y_j(X) = t)$

2.4.2 Bagging (Bootstrap aggregating)

Bagging is an ensemble meta-algorithm used in statistical classification and regression.

Bagging as a technique splits the training set X of size N, into m training sets X_i of size n by sampling from X uniformly with replacement. As a result, some observations can appear several times in X_i. This type of sampling is called bootstrap sampling. Then m models are generated by fitting the m bootstrap samples and their predictions are combined by averaging the output (in the regression case) or by voting (in the classification case) [35].

This technique improves the accuracy and stability of algorithms and it is used to prevent overfitting and to reduce variance [35]. It reduces variance by making the model less sensitive to fluctuations in the training data. This is achieved by training multiple diverse models from bootstrap samples and averaging their predictions.

2.4.3 Negative correlation learning algorithm

A different type of ensemble learning algorithm, called negative correlation learning was introduced in [13], [37]. This approach focuses on creating biased learners, whose errors are negatively correlated and therefore tend to cancel when averaged in the ensemble. It was developed for neural networks, the error of each neural network being defined as:

$$e_i = \sum_{i=1}^{N} (y_{in} - t_n)^2 + \lambda p_i \tag{2.5}$$

where y_{in} is the prediction of ith network of the nth pattern, t_n the corresponding target, λ the weighting parameter of the correlation penalty term, given by:

$$p_i = -\sum_{i=1}^{N}(y_{in} - Y_n)^2 \tag{2.6}$$

where Y_n is the ensemble prediction for the nth pattern.

Rather than training individually all the ensemble members, which would produce uncorrelated errors, these learners are trained simultaneously, through the correlation penalty term. The λ term controls the degree of correlation between the members, a value of $\lambda = 0$ would imply that the networks are trained individually and uncorrelated.

Brown et al [38] offered a statistical interpretation of the efficiency of this method and also found an upper bound for the penalty term, by using the properties of the Hessian matrix. In order for the errors of the networks to converge to a local minima and therefore contribute to a potential decrease in ensemble error, the authors have shown that the penalty term should satisfy the following inequality:

$$\lambda < \frac{M}{M-1}$$

where M is the number of predictors

2.4.4 Random forests

Another well-known example of ensembles is random forests. A random forest is an ensemble of randomly trained decision trees, used in classification, regression and other tasks [39], [40]. A decision tree is a binary tree, where each non-leaf node represents a binary partition of the feature space.

A decision tree consists of a sequence of nodes, starting from the first node, called the root. Nodes can be split into one or two branches, called the left or the right child or cannot be split anymore, in which case they are called leaves.

A split consists of an attribute of the data and a value or threshold that determines the split. For each pattern that will be evaluated by the tree, the value of its corresponding attribute will be compared to the threshold, if it is lower, then the pattern will go down the left branch and otherwise on the right.

In order to determine the best split, an impurity function is used.

The splitting process is repeated until a leaf is reached. In every leaf a final prediction is stored. In the classification case each leaf contains the empirical distribution over the classes associated with the training data that reached the leaf. The leaf prediction model for the t^{th} tree from the forest is: $p_t(c|X)$, where $c \in c_k$ represents the class. All trees are trained independently in a forest. In the testing phase each pattern is pushed simultaneously through all trees starting from root until it reaches the leaves [40]. The

ensemble prediction will be the average over all leaf predictions:

$$p(c|X) = \frac{1}{M} \sum_{t=1}^{M} p_t(c|X)$$

for a forest formed of M trees. By averaging the predictions the effect of noise is reduced [40].

The "randomness" refers to the fact that the trees from the forest are randomly different from each other. This ensures de-correlation between tree predictions, leading to improved generalisation. Randomness is ensured during the training phase and two of the most popular methods are:

- bagging
- randomized node optimisation

Bagging (short form for bootstrapping aggregation learning) is an ensemble technique, which involves providing each predictor a different training data, obtained by sampling uniformly with replacement from the original data.

Randomized node optimisation for a specific node j is based on selecting a subset of parameters, $\mathcal{T}_j$, from the set of all possible parameters θ which we denote by $\mathcal{T}$, and train that specific node with those parameters. The optimisation is done by finding for each split node j the parameters that maximise the information gain:

$$\theta_j^* = \arg\max_{\theta_j \in \mathcal{T}_j} \mathcal{I}_j$$

where $\mathcal{I}_j$ is the information gain. The information gain [41] is a metric to establish which would be the best split for a decision tree in terms of class separation and predictability performance. Other metrics can be used like Gini index [42] and different impurity functions. Next we will present the concept of impurity functions, as the experiments in Chapter 6 are based on these concepts.

2.4.5 Impurity functions

Definition 2.4.1. *An impurity function, $f : [0, 1] \to \mathbb{R}$, is a concave function, satisfying the following properties [43]:*

1. *f is continuous on [0, 1] and C^3 on $(0, 1)$;*
2. *$f'' < 0$ on $(0, 1)$*
3. *$f(0) = f(1)$*

where C^3 means that the 3rd derivative of the function is continuous.

An impurity function can be used on all the attributes and for all possible thresholds, and the pair that would yield the lowest value of the impurity will be chosen as the optimal split. f maps the re-scaled data on the chosen attribute/feature to impurity.

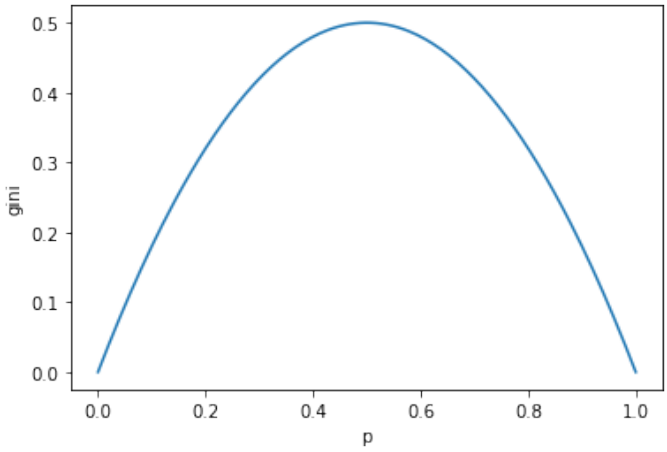

Figure 2.3. The values of the Gini index are plotted on the y-axis, whereas the corresponding probabilities are shown on the x-axis.

The impurity functions can be divided into two categories: symmetric and asymmetric. Next, we will present the two types of impurities into more detail.

2.4.6 Symmetric impurities

Symmetric impurities are given by symmetric functions on [0, 1] . A function f is symmetric if:

$$f(p) = f(1 - p) \tag{2.7}$$

One of the most used impurity functions in building decision trees is the Gini index, defined as:

$$g(p) = 2p(1 - p),\ p \in [0, 1] \tag{2.8}$$

The Gini impurity evaluates how mixed the classes are from each of the groups, determined by the split. A perfect separation would yield a score of 0, whereas in the worst case the classes would be distributed equally in both groups, yielding a score of 0.5 [44].

The Gini function is plotted in Figure 2.3.

2.4.7 Asymmetric impurities

Asymmetric impurities as opposed to symmetric impurities do not satisfy Equation (2.7). They are used in the case of imbalanced classes, where one class occurs much more frequently than the other, the minority class usually being the one of interest. Asymmetric impurities bias the predictions towards one class, which can be the minority class in the case of imbalanced classes.

A family of asymmetric impurity functions was defined in [45], defined by the following equation:

$$h(p) = \frac{p(1 - p)}{(-2w + 1)p + w^2} \tag{2.9}$$

where p is the probability of the minority class from the leaves and w is a parameter specified by the user. The plot for this impurity can be found in Figure 2.4. Standard decision trees would reach maximum uncertainty when the probability of a class is 0.5. However, in the case of imbalanced datasets this scenario is rarely encountered, due to

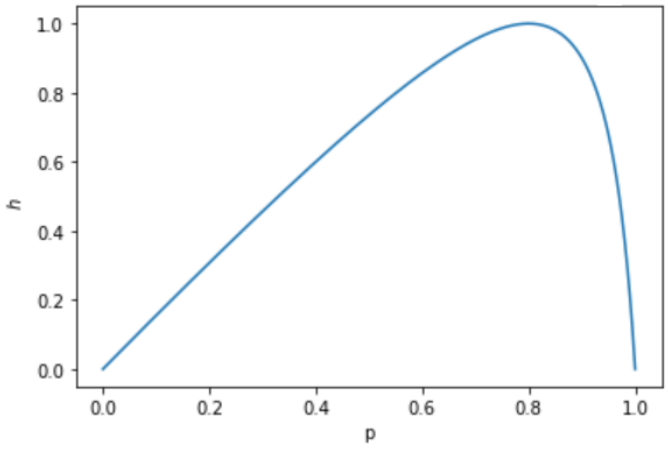

Figure 2.4. The values of the h impurity function from Equation 2.9 are plotted on the y-axis, whereas the corresponding probabilities are shown on the x-axis. This plot was obtained for $w = 0.8$

the fact that the minority class appears in fewer samples. In such cases, since the focus is on correctly predicting the minority class, the level of maximum uncertainty should be decreased as much as possible. Marcellin et al [45] have defined parameter w as being the value for which the maximum uncertainty would be achieved for the predictions. The effect that these asymmetric functions would have in terms of recall and precision in comparison with a symmetric impurity was analysed, by using different classifiers (C45 [46] and Random forests). The experiments showed that the most effective classifier was the random forest with an asymmetric impurity. A similar analysis was done in [47], where the performance of asymmetric versus symmetric impurities was compared by using ROC curves, also in the case of imbalanced classes.

In [43] the authors characterize different impurity functions formally, providing different theorems that aid the comparison between different impurity functions. First of all, they introduce the notion of a preimpurity function, which is a function $f : [0, 1] \to R$, satisfying the following conditions:

1. f is continuous on [0,1] and C^3 on $(0, 1)$
2. $f'' < 0$ on (0,1)

If $f(0) = f(1) = 0$, then f is an impurity function. The authors use this notion of preimpurity functions in order to have more flexibility in fixing the values of the end points.
They define the notion of positive prevalence of a node, as being the weighted proportion of the positive class in that node, denoted by c, respectively a and b for the left and right child.
The authors provide the following mathematical framework in order to characterize impurity functions.

Proposition 1. *Given a node n, with positive prevalence c and total weight W, with left and right positive prevalences a and b and weights W_l and W_r, then:*

$$W_l = W\frac{b-c}{b-a}, W_r = W\frac{c-a}{b-a}$$

and the total impurity of the split given the preimpurity function f is:

$$W\left(\frac{b-c}{b-a}f(a)+\frac{c-a}{b-a}\right)f(b)$$

Definition 2.4.2. *Let f, g be two preimpurity functions. f is equivalent to g if for every node n and every set of possible splits, the optimal split with respect to f is the same as the one for g. Mathematically, this condition can be written as:*

$$f \Leftrightarrow g \text{ if } \forall c \in (0,1)\ \forall \textit{finite subsets } S \subseteq ([0,c)\times(c,1])\cup(c,c)$$

we have

$$\underset{(a,b)\in S}{arg\ min}\left(\frac{b-c}{b-a}f(a)+\frac{c-a}{b-a}f(b)\right)=\underset{(a,b)\in S}{arg\ min}\left(\frac{b-c}{b-a}g(a)+\frac{c-a}{b-a}g(b)\right)$$

Definition 2.4.3. *Let f, g be two preimpurity functions. f splits more positively than g if for every node n and every set of possible splits, there exist an optimal split with respect to f that produces a rights child with positive prevalence greater than or equal to the positive prevalence of every node produced by every optimal splits of n with respect to g. Mathematically, this condition can be written as:*

$$if\ \forall c \in (0,1)\ \forall \textit{finite subsets } S \subseteq ([0,c)\times(c,1])\cup(c,c)$$

we have

$$max\left\{\underset{(a,b)\in S}{arg\ min}\left(\frac{b-c}{b-a}f(a)+\frac{c-a}{b-a}f(b)\right)\right\}\geq max\left\{\underset{(a,b)\in S}{arg\ min}\left(\frac{b-c}{b-a}g(a)+\frac{c-a}{b-a}g(b)\right)\right\}$$

Equivalently, g splits more negatively purely than f if :

$$min\left\{\underset{(a,b)\in S}{arg\ min}\left(\frac{b-c}{b-a}g(a)+\frac{c-a}{b-a}g(b)\right)\right\}\leq min\left\{\underset{(a,b)\in S}{arg\ min}\left(\frac{b-c}{b-a}f(a)+\frac{c-a}{b-a}f(b)\right)\right\}$$

In [43] the authors have used class weighting as a technique to bias a tree's prediction towards one class. They show that the optimal splits given by weighting a class is equivalent to applying a transformation to an impurity function.

Definition 2.4.4. *Given $w > 0$, define the following function $\phi_w : [0,1] \to [0,1]$ by:*

$$\phi_w(p)=\frac{wp}{1+(w-1)p}$$

Definition 2.4.5. *Define the transformation T_w for $w > 0$ on a set of functions f defined on [0,1] by:*

$$(T_w f)(p) = (1+(w-1)p)(f\circ\phi_w)(p)$$

where $\circ$ denotes the function composition operator

Definition 2.4.6. *Given f a preimpurity function, f respects class weighting if for all $w_1, w_2 > 0$, we have:*

$$w_1 \leq w_2 \rightarrow T_{w_1} f \quad \textit{splits more positively purely than } T_{w_2} f$$

Impurity functions like Gini and the entropy respect class weighting. Another example of impurity function that respects the class weighting is $f(p) = p - p^\alpha, \alpha > 1$ and $f(p) = p^\alpha - p$, $0 < \alpha < 1$.

The authors define also the notion of cost-insensitive function as being a function insensitive to class weighting, meaning that class weighting does not affect the optimal splits. More formally: f is a cost insensitive if f is equivalent to $T_w f, \forall w > 0$.
Another interesting result about a family of impurity functions is the following:

Theorem 1. *Let f be an impurity function, which $f \in C^4$ on (0,1). Then f is cost-insensitive if and only if f is equal to a positive multiple one of the functions from the family f_α given by:*

$$f_\alpha(p) = p^\alpha (1-p)^{1-\alpha}, \alpha \in (0,1)$$

Also, f_α splits more positively purely than f_β if and only if $\alpha \geq \beta$.

Next we will present examples from the literature when asymmetric impurities were used in conjunction with imbalanced datasets.

2.4.8 Applications of impurities

A frequent situation in real world problems is when the classes have different misclassification costs. For example in the case of cancer datasets, a false negative might result in the death of a patient and would have much higher consequences than a false positive. One method of dealing with these issues is the use of asymmetric impurity functions when determining the splits, in the case of decision trees. These asymmetric impurities bias the predictions to a specific class, as opposed to the symmetric impurities for which the predictions are random.
Real world applications often exhibit class imbalance, when one class occurs much more frequently than the other. In some cases the minority class is of great interest, in which case the accuracy of a learner might not suffice in calculating its performance. In such situations, the best approach would be to evaluate the true positive rate and the false positive rate and maximise the AUC (area under the receiver operating characteristic curve). The positive class is considered to be the minority class. Class imbalance and asymmetric misclassification costs are connected to one another. Imbalance could be tackled by increasing the cost of misclassifying the minority class, whereas one way to force an algorithm to be cost sensitive could be done by deliberately imbalancing the classes during training [106]. In order to counteract imbalanced classes, approaches like under-sampling the majority class (reducing its appearance) or over-sampling the minority class have been developed. One way of doing this could be done by class weighting (the weights of the chosen class for all examples is scaled by a factor) [43].

In [48] the authors tackle the imbalanced datasets problem by combining *external* approaches with *internal* approaches and comparing their performances. External methods of dealing with imbalanced classes imply under-sampling or oversampling. Numerous methods to reduce the imbalance have been developed, for example for the oversampling strategies, algorithms like SMOTE [49] and Borderline2-SMOTE [50] have been developed, which involve selecting randomly a minority class instance and its k nearest minority class neighbours. In the case of under-sampling clusters including the majority classes instances could be built, and each cluster could be represented by its centroid. Internal methods refer to learning methods that incorporate some bias towards the minority class. Also some hybrid methods that combine the two approaches (internal and external) have been developed. In the case of the decision trees, internal methods of dealing with imbalanced class include:

1. Adapting the split criteria
2. Using a particular pruning scheme
3. The assignment rule of a class to each example

Guermazi et al considered adapting the split criteria by using three asymmetric impurity functions:

1. The off-centered entropy, defined by: $\eta(p) = -\pi \log_2(\pi) - (1-\pi)\log_2(1-\pi)$
   where $\pi = \begin{cases} \frac{p}{2w}, & \text{if } 0 \leq p \leq w \\ \frac{p+1-2w}{2(1-w)}, & w \leq p \leq 1 \end{cases}$ and w is a parameter defined by the user, which represents as before the maximum level of uncertainty
2. Weighted information entropy (IEW), $IEW(p_w) = -p_w \log_2 p_w - (1-p_w)\log_2(1-p_w)$
   where $p_w(D_i) = \frac{q_i|D_i|}{\sum_{i=1}^{|D|} q_i|D_i|}$ and q_i is the weighted coefficient of the class D_i, which is defined as $q_i = \frac{|U|}{|U_{D_i}|}$, where U is the set of objects and U_{D_i} contains the elements from the D_i class
3. The asymmetric entropy defined in equation 2.9

Chaabane et al [48] also studied the particularities of the minority instances, by classifying them into safe and unsafe minority patterns (which are also divided into borderline, rare and outlier examples). Unsafe patterns represent the ones that are more likely to be misclassified [51]. Borderline examples represent the instances close to the decision boundary. Outlier examples are considered to be single minority examples which are inside the majority class region, whereas rare examples are isolated few minority instances also located in the majority class region [48]. The authors combined the three asymmetric impurities with the over-sampling, under-sampling techniques mentioned above and with hybrid method SMOTE-ENN. SMOTE-ENN uses the techniques from the SMOTE method, but after that performs a cleansing of the data, by using the ENN method. The ENN technique implies removing from the dataset any training example that is misclassified by its three nearest neighbours [52]. In order to asses the performance of each method, the true

positive rate, true negative rate and IBA (Index of Balanced Accuracy) metric were used with the aid of statistical tests. The IBA metric is defined as:

$$IBA = (1 + 0.1 \times (TPR - TNR)) \times TPR \times TNR$$

where TPR stands for true positive rate and TNR for true negative rate.
The authors' findings include that the undersampling methods are more efficient when dealing with unsafe data, however when IBA is used as a metric, this method is more performant in the case of outlier data. In this case, there was not a significant improvement in the case of oversampling techniques of hybrid methods. In the case of safe data sets, non-sampling seemed to be the most effective.
After introducing the properties of some state-of-the-art ensemble methods, next we will present two key factors in building successful ensembles.

2.5 Accuracy and diversity

When constructing ensembles two essential features should be taken into account, which are accuracy and diversity. The more accurate the classifiers are, the more similar their predictions are likely to be. Since ensembles are needed for better generalisation, similar accurate ensembles would exclude the need for ensembles, therefore diversity between members is needed. The more diverse the classifiers are, the more spread their predictions will be around the desired output, ensuring that the mean value of their predictions will be close to the target value. Diversity has also been shown to be effective in reducing the variance of ensembles. In order for the outputs of the members to be around the target value, accuracy is required. Therefore, in order for an ensemble to be performant in generalising, the members have to be accurate and diverse. This aspect describes the accuracy and diversity trade off [53]. Next we will present these two concepts.

2.5.1 Accuracy

Accuracy calculates the rate of correctly classified patterns. In order to calculate the accuracy of an ensemble formed of M predictors, we first combined the predictions of the base classifiers by majority voting :

$$Y_n = \begin{cases} 0 & \frac{1}{M}\sum_{j=1}^{M} y_{jn} \leq 0.5 \\ 1 & \text{otherwise} \end{cases} \tag{2.10}$$

where y_{jn} is the prediction of the n^{th} pattern by the j^{th} base classifier
Then the training accuracy of the ensemble is calculated by using the following formula:

$$acc(\mathcal{Y}) = \sum_{i=1}^{N} \mathcal{Y}_i(+)/N, \tag{2.11}$$

where $\mathcal{Y}_i(+)$ denotes if the i^{th} pattern is correctly classified, $\mathcal{Y}$ the ensemble, N is the total number of patterns [54].

2.5.2 Diversity

In [55] there are presented four methods for creating diverse ensembles. Diversity can be achieved by: using different learning algorithms, initializing learning models with different structures or weights, supplying different training data [54].
Since a unique definition for diversity does not exist, many diversity measures have been proposed. Next we will present a list of them.

1. Coincident failure diversity (CFD)
   The coincident failure diversity is the property of different learners missclasifying different patterns. This diversity measure uses the value p_n which is the probability that n classifiers predict incorrectly k_n patterns. Therefore, we have:

   $$p_n = k_n/N,$$

   The formula of the CFD measure is :

   $$div_{CFD} = \begin{cases} 0 & \text{if } p_0 = 1 \\ \frac{1}{1-p_0} \sum_{i=1}^{M} \frac{M-i}{M-1} p_i & \text{if } p_0 < 1 \end{cases} \tag{2.12}$$

   This diversity measure ranges from 0 to 1. 0 corresponds to the least diverse ensemble (when they simultaneously classify correctly or incorrectly the same patterns) and 1 to the most diverse (when the learners missclasify different patterns). The CFD is a non-pairwise diversity measure on the objective space [54].

2. Disagreement measure (DIS)
   DIS uses the elements of an oracle matrix, which is defined as:

   $$o_{i,j} = \begin{cases} + & \text{if } x_i \text{ is correctly classified by } h_j, \\ - & \text{otherwise.} \end{cases}$$

   Lets consider two base classifiers h_i and h_j, then let $n_{i,j(a,b)}$ be the number of patterns on which the oracle output of h_i and h_j is a and b.
   The DIS measure is defined as:

   $$div_{DIS} = \frac{2}{NM(M-1)} \sum_{i=1}^{M} \sum_{j=i+1}^{M} n_{i,j}(+,-) + n_{i,j}(-,+) \tag{2.13}$$

   Basically this diversity measure calculates for each two classifiers how many times their predictions differ and then averages over all pairs. It is a pairwise diversity measure on the objective space. Its values also range from 0 to 1, 0 being the least diverse ensemble and 1 corresponding to the maximum diversity.

3. Hamming distance diversity measure (HD)
   The HD diversity measure is defined on the input space and it calculates how different

every two features or pattern vectors are from each other. The formula is:

$$div_{HD} = \frac{2}{NM(M-1)} \sum_{i=1}^{M} \sum_{j=i+1}^{M} \sum_{k=1}^{N} (x'_i(k) \oplus x'_j(k)) \tag{2.14}$$

where $x'_i(k)$ is the k^{th} element of the feature vector x'_i and $\oplus$ is the logical exclusive operator or [54]. Its values range also from 0 to 1.

4. Double fault measure

   This measure, as the DIS measure, uses the term $n_i(a, b)$ and the idea behind it is that two classifiers in order to be diverse, should perform differently. It was proposed by Giacinto and Roli [56] and their idea was that two different classifiers will have few coincident misclassifications between them. The formula is [57]:

$$DF = \frac{2}{NM(M-1)} \sum_{j=1}^{M} \sum_{k=j+1}^{M} n_{j,k}(-1, -1) \tag{2.15}$$

5. Kohavi-Wolpert variance

   This measure is based on the bias-variance decomposition of the error of a classifier. The variability of the predicted class y_i for a pattern x_i is:

$$variance_{x_i} = \frac{1}{2} \left(1 - \sum_{j=1}^{C} P(y = \omega_j | x_i)^2 \right)$$

   where C is the number of classes
   Kuncheva and Whitaker elaborated a modified version of the above formula in [58]:

$$KW = \frac{1}{NM^2} \sum_{i=1}^{N} l_i(M - l_i)$$

   where l_i is the product between the M classifiers and the sum of the weights of the classifiers that misclassify the pattern x_i. Formally, l_i can be written as:

$$l_i = M \sum_{o_{i,j}=-1} w_j$$

   where the $o_{i,j}$ is the oracle matrix that also the Disagreement diversity measure uses and w_j are the weights of the learners that misclassify pattern x_i.

6. Generalised diversity

   The assumption behind this diversity measure is that given two classifiers the maximum diversity is achieved when a misclassification of one classifier is followed by a correct classification of the other classifier. Conversely, minimum diversity is achieved when two classifiers fail simultaneously [57].
   Let x_i be a sample drawn randomly from the training set, T_j be the probability that

$l_i = j$ (l_i has the same meaning as for the Kohavi-Wolpert diversity measure), then the formula for the generalised diversity is [57]:

$$GD = 1 - \frac{\sum_{j=1}^{M} \frac{j(j-1)}{M(M-1)} T_j}{\sum_{j=1}^{M} T_j \frac{j}{M}} \tag{2.16}$$

7. Measurement of inter-rater agreement

The idea behind this diversity measure is that in order to have a diverse ensemble, the set of classifiers should disagree with each other. The formula is:

$$K = 1 - \frac{\sum_{i=1}^{N} (M - l_i) l_i}{NM(M-1)P(1-P)} \tag{2.17}$$

where P denotes the average classification accuracy of the base classifiers on the training data and l_i again is the product between the M classifiers and the sum of the weights of the classifiers that misclassify the pattern x_i [57].

8. Kappa statistics

Given two classifiers, y_p and y_q from a range of M classifiers and a dataset formed of N patterns, a contingency matrix, T can be formed, where T_{ij} represents the number of examples x, for which $y_p(x) = i$ and $y_q(x) = j$. Obviously, T_{ii} will be the number of examples for which the two classifiers agree.

Let Θ_1 be:

$$\Theta_1 = \frac{\sum_{i=1}^{M} T_{ii}}{N}$$

Θ_1 can be seen as a probability of how much two classifiers agree [36]. A disadvantage of this measure would appear in the case of imbalanced classes, where the two classifiers will tend to agree on the dominant class.

In order to correct this issue, let Θ_2 "be the probability that two classifies agree by chance" [36]:

$$\Theta_2 = \sum_{i=1}^{M} \left(\sum_{j=1}^{M} \frac{T_{ij}}{N} \cdot \sum_{j=1}^{M} \frac{T_{ji}}{N} \right)$$

Hence, the k statistic can be defined as:

$$k = \frac{\Theta_1 - \Theta_2}{1 - \Theta_2} \tag{2.18}$$

k ranges between 0 and 1, a low values of k indicates low agreement, whereas a high value represents high agreement.

9. Q statistics

Let's define two classifiers y_j and y_q and $z_{i,j} = 1$ if y_j correctly classifies pattern i

and 0 otherwise. Let N^{qp} be the number of patterns for which $z_{i,j} = q$ and $z_{i,q} = p$. Then the Q-statistics between the classifiers y_j and y_q is defined as:

$$Q_{j,k} = \frac{N_{11}N_{00} - N_{01}N_{10}}{N_{11}N_{00} + N_{01}N_{10}} \tag{2.19}$$

The Q statistics measure ranges between -1 and 1. For classifiers that agree on many examples, the value of the Q statistics will be positive, and negative otherwise.

10. Another type of diversity could be considered the penalty term from the NCL [13], [37]:

$$p_i = -\sum_{i=1}^{N}(y_{in} - Y_n)^2 = \sum_{i=1}^{N}(y_{ik} - Y_i)\Big[\sum_{j\neq k, j=1}^{N}(y_{ij} - Y_i)\Big] \tag{2.20}$$

11. In [59] a new diversity formula was defined for neural networks. The diversity of the i^{th} network is calculated as:

$$D_i = \sum_{x}[\sigma_i(x) - \hat{\sigma}(x)]^2 \tag{2.21}$$

where $\hat{\sigma} = \sum_{i\in N} c_i \cdot \sigma_i, \sum_{i\in N} c_i = 1$ and σ_i are the predicted outputs of the network . This diversity measure was obtained as a results of the bias-variance decomposition in the regression case, introduced by [21].

12. Ambiguity
    In [1] the authors have defined another diversity measure, called ambiguity, by using the bias-variance decomposition from [21] and the 0-1 loss. Chen's ambiguity measure, amb_{01} is defined as:

$$\text{amb}_{01}(\mathcal{Y}_n) = \frac{1}{2}\sum_{i=1}^{M}\left(\frac{1}{M}Y_n - c_i y_{in}\right)t_n \tag{2.22}$$

where Y_n is the ensemble prediction of the n^{th} pattern and $\mathcal{Y}_n$ is the set of outputs of the ensemble members.

In this section we have presented a range of diversity measures. Some are operating on the feature space, like the Hamming distance or on the prediction space like the rest. Diversity measures have been used in real-world applications such as cryptography (the Hamming distance [60], [61]), social sciences and psychology (Kappa statistics [62], Q statistics [63], Inter-rater agreement [64], medicine (Inter-rater agreement and Kappa statistics [65]), DNA classification (Hamming distance [66], [67]). The coincidence failure measure can be used in the field of reliability engineering and system safety analysis. The concept of simultaneous failures has been discussed in the following papers [68],[69]. The CFD, DIS, GD, Double fault measure, KW and Generalized diversity have also been employed in studies to check their impact on the generalisation error. A comparison of their correlation with the ensemble error has been conducted in [58] and [70]. In [58] the authors could not empirically demonstrate a strict correlation between these diversity measures and generalisation error, however the amb_{01} measure has been shown to have a positive impact

on the generalisation error [70]. We will continue this analysis in Chapter 3.
Since an ensemble has to be accurate and diverse, we are faced with a multi-objective optimisation problem. We will now introduce the basic concepts related to multi-objective optimisation.

2.6 Multi-objective optimisation

As the name suggests, multi-objective optimisation deals with finding solutions that optimise more than one objective. Mathematically a multi-objective optimisation problem can be formulated without loss of generality as [71]:

$$min\, F(z),\ F = \{f_1(z), f_2(z), \dots f_n(z)\} \tag{2.23}$$

where $z = (z_1 \dots z_p)$ is the parameter vector in the decision space.
One method of dealing with multi-objective optimisation is by combining the objectives into a scalar function. For example, in a 2-objective optimisation problem, we would have:

$$F = f_1 + \lambda f_2,$$

where $\lambda > 0$ takes a value that has to be defined by the user.

This approach has been used in machine learning in regularizing neural networks, creating interpretable fuzzy rules and generating negatively correlated ensemble members [2]. However this method has some drawbacks. Firstly, it is not easy to determine the hyperparameter λ. Secondly, a single solution is obtained, from which little insight can be gained [2]. Combining the objectives into a weighted sum, can lead to a loss of detailed information about specific objectives. In some cases, the objectives can be conflicting and by aggregating them into a sum, might oversimplify the trade-off and the complex relationship between them.

An alternative approach to deal with multi-objective optimisation is to use the Pareto approach. In this method the objective function is a vector. As an example, in the case of a multi-objective optimisation minimisation problem (see Equation2.23), a solution z dominates another solution y if $f_j(z) \leq f_j(y)\ \forall j = 1 \dots n$ and $\exists i \in 1 \dots n$ such that $f_i(z) < f_j(y)$, which is denoted by $z \prec y$.
A solution z is called Pareto-optimal if it is not dominated by any other feasible solution. All Pareto-optimal solutions in decision space form the Pareto set. The set containing the image of Pareto-optimal solutions in the objective space is called the Pareto front [2].

Multi-objective optimisation can be done with the aid of a genetic algorithm. In genetic algorithms, a population of candidate solutions (called individuals) for an optimisation problem is evolved to find better solutions. Each candidate solution is represented by a set of properties (chromosomes) [72].

There are a number of operators used in genetic algorithms to help the algorithms to obtain a solution. These operators are: mutation, crossover and selection. These operators are usually used together, forming the so-called evolution strategies (ES) [73]. Let's denote by μ the number of parents and by λ the number of children involved in evolution, where

$\mu < \lambda$. We can distinguish two types of evolution strategies (ES):

- $(\mu + \lambda)$-ES, where the parents for the next population can be selected from both the parents and children population, also known as "plus" selection
- (μ, λ)-ES, where the parents for the next population can be chosen only from the children population

Better solutions are given preference by the selection operator, allowing them to pass their genes to the next generation. The solutions are ranked using a fitness function.

The crossover operator combines the genes of the parents to generate offsprings. By combining parts of good solutions, it is more likely to generate a better solution [73].

The mutation operator changes the chromosomes encouraging genetic diversity amongst solutions. By mutating the genes of a parent the new solution may be completely different from the parent. Genes can be represented a string of 0s and 1s. There are different types of mutations: bit mutation (modifies random bits in a binary string with low probability) or more complex mutation methods which may use random values chosen from a uniform or Gaussian distribution to replace the genes in the parent [73].

Evolutionary computation techniques are used to find the optimal or near optimal parameters which have unknown or complex mappings from parameters to objectives [74].

2.7 Multi-objective optimisation within ensembles

As mentioned in the previous sections, effective ensembles should be both accurate and diverse, suggesting the need of using multi-objective optimisation. Regardless of the method used for ensuring diversity, the methods for constructing ensembles require two steps: first multiple classifiers are trained and then a subset of them are selected for building the ensemble.

Diversity should be evaluated for all ensemble members, which is difficult to achieve if the ensemble members are selected one by one [54]. In order to overcome this issue, in [54] the authors suggest a method that combines the generation and selection steps, ensuring that all members are generated simultaneously with the aid of a MOEA(Multi-Objective Evolutionary Algorithm). By using this method, the algorithm optimises the accuracies of the base classifiers and finds subsets of them with maximum diversity. In this way, the diversity of the final solution can be evaluated accurately and the trade-off between diversity and accuracy can be taken into account in the ensemble generation stage [54].

The authors ensured diversity by selecting a subset of features or patterns and used the CFD (Eq. 2.12), DIS (Eq. 2.13) and HD (Eq. 2.14) diversity measures.
The accuracy was calculated using majority voting and the formulae used are (2.10) and (2.11). The learners used in this method are SVMs (support vector machines) [8] and the evolutionary algorithm used is NSGA-II [75]. The authors have compared the testing performance of the generated ensembles versus the generalisation error of a single classifier. In most of the cases, the ensembles outperformed the single classifiers, however the approach was only tested for one run/one split of the data. In order to provide a more

general conclusion, the algorithm should have been tested on multiple splits of the data and statistical tests should have been performed in order to validate the comparisons.

Another algorithm for creating diverse and accurate ensembles was presented in [53]. This algorithm called DIVACE (Diverse and Accurate Ensemble learning algorithm) it is based on the memetic Pareto ANN (MPANN) algorithm, but it uses the Pareto differential evolution and the negative correlation algorithm (NCL). MPANN is used for the evolutionary process and diversity is minimised by using the negative correlation penalty function of NCL (see Equation 2.20).

The reason for choosing this diversity measure is due to its link to mutual information. Mutual information is a measure of correlation between two random variables. Therefore, by minimizing the mutual information between the outputs of two ensemble members, we can obtain diversity. It has also been shown that NCL can minimize mutual information between ensemble members (due to the use of this penalty function) [53]. The algorithm was implemented using neural networks, but it can be adapted to any learner. The authors have compared empirically the performance of the algorithm against the MPANN algorithm, their results have shown an improvement in performance. However, the algorithms were only tested on two small datasets from the UCI Machine learning repository, Australian and Diabetes [76], therefore in order to properly assess its superiority more datasets and algorithms should be compared.

Another method called ADDEMUP (Accurate and Diverse Ensemble-Maker giving United Predictions) was suggested in [59] and it is implemented using neural networks. This algorithm creates an initial population which is evolved using genetic operators (crossover and mutation), creating in this way new networks. The main idea of this method is that during the training step, a set of networks that are accurate are kept as long as they disagree with each other as much as possible [77].

The fitness of each member is calculated in the following manner:

$$Fitness_i = acc_i + \lambda \cdot div_i$$

where acc_i is the network i's validation set accuracy and the diversity of the i^{th} network is defined in Equation 2.21. At each iteration new networks are generated and the first M are kept according to their fitness. If the ensemble error is not increasing and the diversity is decreasing, then the λ value will be increased. Conversely, if the ensemble is increasing, but the diversity is not, then λ will be decreased. Even though, the authors have demonstrated empirically the algorithm's superiority compared to other state-of-the art algorithms, the fact that λ has to be defined beforehand and keeps changing, is a disadvantage.

In [78] a new method was suggested, called MRNCL (multi-objective regularized negative correlation learning). This algorithm is a multi-objective regularizer NCL algorithm that contains an additional regularization term for the ensemble [78]. As in the previous two algorithms the learners used were neural networks. In this algorithm a correlation penalty term to the cost function of each network is added. Then each neural network minimizes

the mean square error together with the correlation [78]. The algorithm has reduced the overfitting of the ensemble, as opposed to other state-of-the-art algorithms, which is a promising result.

In [79] another multi-objective problem is discussed which has as objectives to be optimised, the reduction on the training set and the performance obtained after this reduction with a given set of SVM hyper-parameters. By training an SVM a constrained quadratic programming optimisation should be solved, which has a complexity of $\mathcal{O}(|N|^3)$, where $|N|$ is the number of instances in the training set. As a result, when dealing with large databases the computational cost might be an issue. This drawback induces the need of finding ways of reducing the training size that would not affect the effectiveness of the SVM. The effectiveness of the SVM is controlled by a set of hyper-parameters and the choice of the kernel function. The authors have used as genetic algorithm the MOEA/D [80] algorithm and as methods of reducing the training size, they used a filter or a wrapper. They constructed ensembles from the solutions of the Pareto front in 5 different ways:

1. Global Pareto Ensemble (GPE)
2. Incremental Error Reduction Ensemble (IERE)
3. Complementary Incremental Ensemble (CIE)
4. Margin Distance-Based Ensemble (MDE)
5. Boosting

The GPE approach supposes combining all the solution from the Pareto front into one ensemble.

The IERE approach builds the ensemble in an incremental manner. In the first step, the solution with the lowest training error is included in the ensemble. In the next step, the solution that minimizes the error of the partial ensemble is added to the ensemble and so on.

In CIE, at each iteration the classifier with the performance the most complementary to the partial ensemble is added. The first classifier included is the one with the lowest training error, the next classifiers selected are the ones that have the lowest error rate on the samples missclassified by the partial ensemble.

The MDE approach defines a signature vector $c_i^{(k)}$ of a classifier k, which is equal to 1 if the classifier predicts correctly the ith instance or -1 otherwise. The average signature vector is defined as:

$$\bar{c_i} = \frac{1}{M} \sum_{k=1}^{M} c_i^{(k)}, \forall i \in M$$

The condition for a pattern to be correctly classified by the ensemble is $\bar{c_i} > 0$. The aim of this approach is to minimize the distance of the average signature vector to a positive reference point.

This results show that in terms of classification accuracy both the wrapper and filter were similar. Also the type of ensemble that achieved the highest performance in both objectives was the one obtained via boosting. The authors have compared the ensemble generalisation errors against other state-of-the-art algorithms, such as random forests. Statistical tests have shown the superiority of these approach, which is a promising result.

Another important aspect when building ensembles is computational costs. In some cases it has been shown that the more predictors in an ensemble, the better its performance will be [81]. However the amount of time needed for training such an ensemble, increases linearly with the number of models. In [81] the authors have shown that the ensemble size can be reduced substantially and still obtain a similar performance. In the next section we will present a number of state of the art ensemble reduction methods, which are also called ensemble pruning methods.

2.8 Ensemble pruning

The process of ensemble pruning refers to reducing the number of predictors, by discarding those with little contribution and which will determine an accurate ensemble. Next we will summarise a number of papers from the literature, in which the pruning techniques used take into account diversity and have presented evidence to generate accurate ensembles.

Margineantu and Dietrich [36] suggest ensemble pruning as a method of reducing classification time with the effect of obtaining similar performance to the unpruned ensemble, i.e. a more parsimonious ensemble with equivalent performance. In their paper they suggest 5 pruning methods for ensembles produced by the ADABOOST algorithm (see Algorithm 1). The goal is to obtain an ensemble of M members, by using the following techniques:

1. Early stopping: In this method the first M classifiers generated by ADABOOST are kept. The disadvantage of this approach is that the classifiers produced later could be more useful and they will not be considered

2. KL-divergence pruning: This approach focuses on the diversity of the classifiers, quantified by the KL (Kullback-Leibler) distance of the probability distributions obtained from ADABOOST (see Algorithm 1, line 3). The KL-divergence is defined as:

$$D_{KL}(R||S) = \sum_{x \in X} R(x) \log \left(\frac{R(x)}{S(x)} \right)$$

where R, S are probability distributions and X is the probability space.

A greedy algorithm is used in order to find the sub-ensemble (of M members) that would maximise the summed pairwise KL divergence, given by:

$$J(\mathcal{Y}) = \sum_{i,j}^{M} D_{KL}(p_i||q_j) \tag{2.24}$$

where p_i and q_j are the probability distributions of members y_i, y_j. The ensemble contains at the beginning the first classifier generated by ADABOOST and at each

iteration adds the member that would maximise the quantity from equation (2.24) until M classifiers are found.

3. Kappa pruning: The Kappa pruning method orders the pairs of classifiers in increasing order of their k value, where k is obtained by using the Kappa statistics (see Equation (2.18)). The pairs are added until an ensemble of M classifiers is formed. Kappa statistics measures the level of agreement of different classifiers and can be considered a measure of diversity.

4. Kappa-error convex hull pruning: In this method the k value of each pair of classifiers is plotted against the average error of the pair and the convex hull of the points is considered as the pruned ensemble. A drawback to this method is that not always the desired ensemble size can be kept (M). However, by using this method, the convex hull will include the most accurate and most diverse classifiers.

5. Reduce-error pruning with backfitting: In this method, the data is divided into a training set and a pruning set which will be used to select the ensemble that would minimise the error. The algorithm starts by selecting the classifier with the lowest error on the pruning set, y_1. After that, the classifier y_2 that along with y_1 would form the ensemble with the lowest error, is selected. In the next steps, each of the previously chosen classifiers will be replaced by other classifiers that would minimize the ensemble error on the pruning set. The algorithm will stop when the classifiers will not be changed anymore or when a number of iterations, Q has been reached.

These methods were tested on 10 datasets, by using 10-fold cross validation. The results showed that ensembles can be pruned up to 60-80% and achieve similar performance as the unpruned ensemble. The best pruning approaches were Kappa pruning and Reduce-error pruning with backfitting [81].

In [82] a pruning algorithm that takes into account both accuracy and diversity of classifiers is considered (the Q statistics measure is used, see equation (2.19)). The algorithm, called Accuracy-Diversity Pruning (dubbed ADP) at each iteration increases the ensemble size by adding a pair of diverse classifiers, as long as the accuracy of the new formed ensemble is higher than the previous one. In order to assess the performance of the algorithm, the authors count how many times the ADP algorithm generates the optimal ensemble. Their results show that in most cases the pruned ensembles are optimal and whenever this is not the case, the loss of accuracy is not significant. This conclusion highlights the importance that diversity has on improving the accuracy of an ensemble.

Another pruning algorithm which uses diversity was defined in [83] (called Forward Ensemble Selection, dubbed FES). The authors defined a diversity measure which focuses on the strength of the decision of the current ensemble and takes into account the predictions of individual members. The following quantities for the pair (x_i, y_i), where x_i is the i^{th} pattern, y_i the i^{th} target, y_{ki} the prediction of the k^{th} classifier of the i^{th} pattern and Y_i the ensemble's prediction of the i^{th} pattern, have been defined:

1. NT_i-the proportion of ensemble members that correctly classify the i^{th} pattern.

2. NF_i-the proportion of ensemble members that misclassify the i^{th} pattern.

3. For the k^{th} classifier the following quantities are defined:

   a) e_{tfi}: $y_i = y_{ki}$ and $y_i \neq Y_i$

   b) e_{fti}: $y_i \neq y_{ki}$ and $y_i = Y_i$

   c) e_{tti}: $y_i = y_{ki}$ and $y_i = Y_i$

   d) e_{ffi}: $y_i \neq y_{ki}$ and $y_i \neq Y_i$

The algorithm starts with an ensemble formed of all the possible members and gradually removes the member that would minimise the following diversity measure:

$$f_{es}(h_t) = \sum_{i=1}^{N}(NT_i \cdot I(e_{tfi}) - NF_i \cdot I(e_{fti}) + NF_i \cdot I(e_{tti}) - NT_i \cdot I(e_{ffi}))$$

where $I(true) = 1$ and $I(false) = 0$

If an example is misclassified by most of the ensemble members, then it could be a hard example and it should not influence the ensemble selection. On the other hand, if the pattern is misclassiffied by about half of the ensemble components, meaning that the result of the ensemble is close to the decision boundary which might lead to a misclassification, then it should strongly influence ensemble selection. For example, in the case of the event e_{tfi}, if the number of classifiers that correctly classify example x_i, NT_i is small, then it means that x_i is a hard example. Since NT_i is small, the contribution $NT_i \cdot I(e_{tfi})$ will be small, therefore it will increase the chances of this classifier to be discarded. On the other hand, if many predictors already classify this example correctly (NT_i is large) and the ensemble prediction is close to the decision boundary, then this potential member might change the ensemble prediction into making the correct classification. As a result, its contribution should be rewarded ($NT_i \cdot I(e_{tfi})$ will be larger) and decrease the chances of this classifier being discarded.

The authors have chosen to use heterogeneous ensembles, formed of different models (SVMs, decision trees, kNNs Naïve Bayes or decision trees) and have compared the performance of their algorithm against other state of the art algorithms. Their results show that the FES can reach competitive performances by reducing the ensemble size from 200 to 15 on average.

In [84] the authors consider bagging as an ensemble method and suggest 3 pruning methods.

1. Reduce error pruning, similar to [36] but without backfitting.

2. The complementariness method which gradually adds a model to an ensemble, in order to improve the ensemble's prediction. This is done by choosing the models that correctly predict a pattern, when the ensemble is wrong. Mathematically it can

be expressed as [85]:

$$COM(y_k, \mathcal{Y}) = \sum_{i=1}^{N} I(e_{tfi}). \tag{2.25}$$

The model with the lowest error is considered at the first iteration. In the next iterations the model that maximises the quantity from equation (2.25) will be added.

3. Margin Distance Minimization. In this method each model is assigned a signature vector, defined as following:

$$(c_t)_i = y_i y_{ti}, i \in 1 \cdots N.$$

In a binary classification, for classes ± 1, the element on the i^{th} position of the signature vector will be 1, if the i^{th} example is correctly classified by model y_t and -1 otherwise.

The method also requires defining a desired vector, o which would correspond to the ideal case when all the examples are correctly classified.

The classifier that will be chosen at each iteration, would be the one that minimises the following distance, called the margin:

$$MARD(y_k; S) = d\left(o; \frac{1}{k}(c_k + \sum_{i=1}^{k-1} c_i)\right)$$

where d is the Euclidian distance.

The authors generated bagging ensembles of 200 CART trees and pruned them according to the above 3 methods. They compared these approaches and their results showed that the best method was the Margin Distance Minimization, followed by Complementariness. The last two methods could be considered as diversity measures, highlighting once more the influence that diversity has on pruning ensembles and enhancing ensemble accuracy.

2.9 Conclusion

In this chapter we revise the main background information on which this thesis is based. The main goal of machine learning is to build models which generalise well on unseen data, meaning that their generalisation error is as low as possible. Our research is done in the case of supervised learning which is presented in Section 2.1. The main causes of error, bias and variance are discussed in Section 2.2. One of the most popular machine learning techniques are ensembles. Ensembles are collection of predictors whose performance has been shown to surpass the ones of individual models, the ensemble concepts are presented in Section 2.3. We present ensemble models in Section 2.4. It has been shown that ensemble error or the opposite of error, accuracy, is positively influence by the diversity of the ensemble's constituent classifiers. We present the notions of accuracy and different diversity measures in Sections 2.5.1 and 2.5.2 respectively. One method of building ensembles which are accurate and diverse at the same time is to optimise these two factors at once. A general method of optimising multiple objectives at ones, is done

via evolutionary algorithms which are presented in Section 2.6. We revise a number of popular multi objective optimisation algorithms from the literature in Section 2.7. Along with ensemble accuracy, another important aspect to consider when building ensembles is computational costs. In section 2.8 we present ensemble pruning methods, which reduce the number of ensemble models without compromising ensemble accuracy too much.
Optimising diversity and accuracy together has been shown to be beneficial for the ensemble's overall performance. However, since many authors have suggested different diversity measures, it shows the difficulty of finding the right diversity measure. In the next chapter we will investigate the impact that different diversity measures have on the ensemble's accuracy.

Chapter 3

Correlation between test error and different diversity measures

3.1 Introduction

A principal concern of supervised machine learning is to ensure a predictor demonstrates good *generalisation*. A predictor is considered to have the ability to generalise, if it has a good performance in predicting on unseen data drawn from the same process that it was trained on [3, 6].

Ensembles are collections of predictors, each of which is trained on a different subset of patterns or features. Some ensemble methods such as bagging [10] or boosting [11] have been seen to be very successful in pattern classification tasks [22], due to their efficiency in reducing ensemble variance and bias and consequently improving the generalisation error. Random forests are one of the main applications of bagging, each of the component trees are being trained on bootstrap samples of the data and their predictions are being aggregated. Boosting has been successfully used in predictive analysis in medical research [16], or in forecasting stock prices [18]. Ensembles have been shown in general to predict better than a single predictor [86, 87].

In this chapter we consider classification of patterns $\mathbf{x}_n$, $n = 1, \ldots, N$ into two classes, the positive and the negative class. Each of the M members of the ensemble yields a score $y_{in} \equiv y_i(\mathbf{x}_n)$, $i = 1, \ldots, M$ indicating how likely it is that $\mathbf{x}_n$ belongs to the positive class, and the ensemble score $Y_n \equiv Y(\mathbf{x}_n)$, which may be converted to a decision by thresholding is, in general, the weighted average of the constituent predictor scores [88]:

$$Y_n \equiv Y(\mathbf{x}_n) = \sum_{i=1}^{M} c_i y_{in} \tag{3.1}$$

where c_i are the non-negative weights assigned to the constituent ensemble members, $\sum_i^M c_i = 1$. Here we assume throughout that the ensemble members carry equal weight so that $c_i = 1/M$ for all i. When the constituent classifiers produce a hard decision and the weights are equal this amounts to the often used majority voting.

Clearly, an accurate ensemble requires accurate members. However, Krogh and Vedelsby [21] have shown that an ensemble with good generalisation performance consists of members which disagree in their predictions [89]. As a result, diversity and accuracy are key factors in building successful ensembles.

Although the role of diversity has long been recognised, many ways of quantifying the diversity of an ensemble have been proposed. Kuncheva and Whitaker [23] empirically compared different diversity measures in order to assess the impact that diversity has on an ensemble's generalisation performance. However, their results could not support the influence of diversity on the overall performance of the ensembles. This aspect was partially explained in [1], which showed that different diversity measures have different degrees of correlation with generalisation error. It was also shown that there tends only to be high (negative) correlation between diversity and generalisation error when diversity is low and generalisation error is high; as diversity increases the correlation with generalisation error decreases [1]. We explore this aspect in more detail below.

In [21] Krogh and Vedelsby introduced a new diversity measure based on the ambiguity decomposition of regression ensembles and the bias-variance decomposition. The ambiguity term is obtained by subtracting the ensemble error from the average error of the predictors. The average error of the individual models has been proven to be greater than the ensemble error [90],[21], as a result from the ambiguity decomposition we can conclude that the ambiguity measure is always positive. The ambiguity measures how much the predictions of the ensemble members differ from the ensemble prediction and as a result can be considered a type of diversity. Chen [1] defined another ambiguity measure in a similar fashion as to [21], but for classifiers and using the 0-1 loss. Chen's ambiguity measure, amb_{01} is defined as:

$$\text{amb}_{01}(\mathcal{Y}_n) = \frac{1}{2}\sum_{i=1}^{M}\left(\frac{1}{M}Y_n - c_i y_{in}\right)t_n \tag{3.2}$$

where Y_n is the ensemble prediction of the n^{th} pattern and $\mathcal{Y}_n$ is the set of outputs of the ensemble members. In his work, Chen demonstrated that out of all the diversity measures tested (Q-statistics (Equation 2.19), Kappa statistics (Equation 2.18), Correlation coefficient ([91]), Disagreement (Equation 2.13), Entropy [92], Kohavi-Wolpert variance (Equation 5), the measure of difficulty [93], generalised diversity (Equation 2.16), coincident failure diversity (Equation 2.12), the ambiguity measure had the highest correlation with the generalisation error [1]. We will use the term *ambiguity* to refer to a measure of ensemble diversity.

In this chapter we will investigate the effect that different diversity measures have on the test error. The diversity measures considered are the Coincidence Failure, Kohavi-Wolpert, Disagreement, Generalised Diversity and the amb_{01} measure introduced in [1]. We also present a new diversity measure, called coherence in Section 3.2 which measures the minimum angle between the predictions of one classifier and those of the rest of the classifiers. The coherence measure is also included in our analysis of the effect that diversity measures have on the test error, in Section 3.3. In Section 3.5 we write the bias-variance

decomposition of the amb_{01} measure in a different manner, which leads to a series of pruning methods based on this type of ambiguity. Section 3.7 presents the conclusions of this chapter.

The main contributions of this chapter are the following:

1. The definition of a new diversity measure, called coherence, which quantifies the minimum angle between the predictions of one classifier and those of the rest of the classifiers

2. The analysis of the correlation between diversity and generalisation error across the following cases: when varying the features of the trees, when setting the bootstrapping to false or when varying the ensemble or data size

3. The analysis of the correlation between diversity and the area under the curve

4. The derivation of different pruning techniques which involve ambiguity or coherence

3.2 Coherence diversity measure

Since there is no universal definition for diversity, many ways of quantifying diversity have been introduced. One possible way of quantifying diversity in an ensemble, is to calculate how different the vectors of predictions of each classifier are. We can quantify this difference as the angle between the vector of predictions of each classifiers. We would like these angles to be as big as possible (the prediction vectors to be far away/different from each other). The prediction vectors would be in a N dimensional space, hence these angles could be represented in a N-sphere. However, for illustration purposes, we present an example of "diverse predictions" on a sphere, as seen in Figure 3.1. As such, we can start by measuring the minimum angle between the prediction of a classifier and the rest of the predictions:

$$div(y_i) = \min_j \theta_{i,j} = \alpha \tag{3.3}$$

where $\theta_{i,j}$ is the angle between the vectors y_i and y_j of the predictions of the ith and jth classifiers on N test cases.

One way of determining the angle between vectors would be by using dot products, obtaining:

$$\cos\theta_{i,j} = \frac{y_i^T y_j}{\|y_i\|\|y_j\|} = \frac{y_i^T y_j}{N} \tag{3.4}$$

where N is the number of patterns. In the case of binary classification, we will consider the targets to be ± 1 , therefore we will have $\|y_i\|^2 = N$.

In order for the rest of the vectors to be spread out, the minimum angles of the rest of the predictions should be greater than α, as such we are aiming to maximise those angles. Next, lets define the coherence function as being:

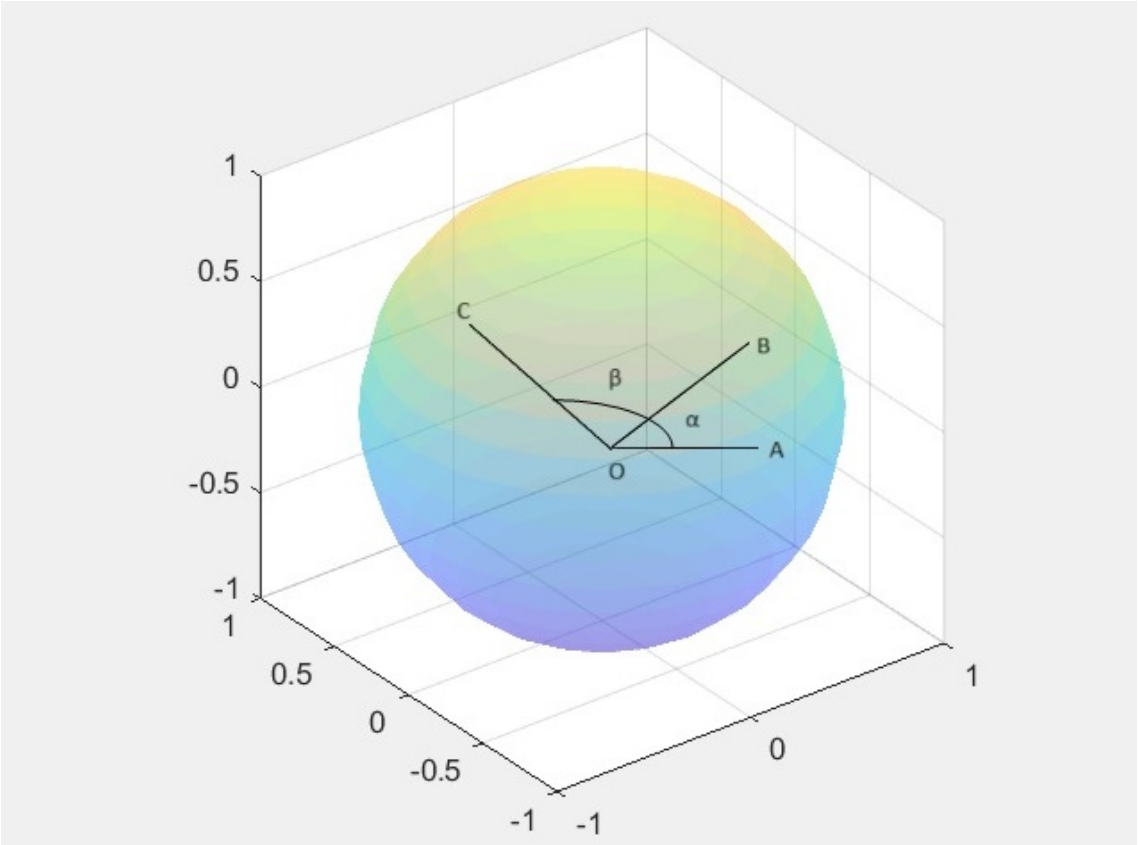

Figure 3.1. Example of diverse predictions in a 3-dimensional space

$$\Gamma(y_i) = \max_j y_i^T y_j \tag{3.5}$$

By applying equations (3.4), (3.5) we obtain

$$\frac{\Gamma(y_i)}{N} = max(cos(\theta_{i,j})) = cos(min(\theta_{i,j})) = cos(div(y_i)) \tag{3.6}$$

As a result the coherence is the inverse of diversity. Since the coherence decreases as the ensemble members become more diverse, in a multi-objective setting, we will need to minimise the coherence and maximise accuracy for all the classifiers from the ensemble. One equivalent way to do this is to minimise both coherence and the error of each classifier. However, it is impossible to obtain a Pareto front of classifiers using this definition, because there will always be two classifiers with the same minimum angle (being the angle from one to another). As a result, we add the angle between a classifier's prediction and the prediction of the ensemble containing all the classifiers. Therefore the definition of coherence becomes :

Definition 3.2.1. *Given an ensemble of M classifiers, where Y represents the ensemble outputs and y_i, $i = 1..M$, denote the individual classifier predictions. Then, the coherence of the ith classifier can be defined as:*

$$COH(y_i) = max(y_i^T y_j) + \frac{y_i^T Y}{N} \tag{3.7}$$

Datasets	Patterns	Features	Feature Type	Positive samples
Australian	690	14	Categorical, Integer, Real	44%
Cancer	569	30	Real	35%
Heart	270	13	Categorical, Real	44%
Sonar	208	60	Real	53%
Ionosphere	351	34	Integer, Real	64%

Table 3.1. Dataset characteristics

Hence, we can define the coherence of the ensemble as being:

$$COH(\mathcal{Y}) = \frac{1}{M}\sum_{i=1}^{M} COH(y_i) = \frac{1}{M}\sum_{i=1}^{M} max(y_i^T y_j) + \frac{y_i^T Y}{N} \tag{3.8}$$

We would like to note that the coherence diversity measure is not sensitive to data scaling. Even though we considered the case, $y_i \in \{\pm 1\}$, the coherence measure maximizes the scalar product between vectors, which could be of any magnitude or range.

3.3 Correlation between diversity and test error

Inspired by Chen's work [1], we investigate the correlation of different diversity measures with the test error. In addition, we will also assess the correlation of these diversity measures with the area under the curve.

We generate ensembles of decision trees, by using bagging with a sampling rate ranging between $[0.1, 1]$ with the interval 0.05. For each sampling rate a forest of 100 trees is generated. We use 5 fold cross validation and the training set for fitting the models and the test set is used to asses the error and the diversity of the ensembles. We generate 100 ensembles and the average test error and test diversity over these runs were used to measure the correlation. The experiment was conducted twice, once for bootstrapping with replacement and once without (Chen [1] used bootstrapping without replacement). The datasets used in these experiments were Australian credit card [94], Wisconsin Breast Cancer [95], Heart disease [96], Sonar [97] and Ionosphere [98] from the UCI Machine learning repository [76]. The characteristis of these datasets can be found in Table 3.1.

The decision trees can be formed by varying the number of patterns selected (in such a manner that the sampling rate is kept) and at the same time considering all the features, as in [1]. Equivalently, another way of building trees could be, by varying the number of features.

We ran the experiment described above in both cases, when considering all features or just a random number of them. We will present the results of these cases in the following two sections.

3.3.1 All features considered

The first case analysed is the one when all the features are considered and subsets of patterns are selected for each tree. We plot for each diversity measure the test error versus the test diversity per dataset. Since we usually do not have access to the testing data, we tried to assess the correlation between the training diversity and the test error, to see if we can predict from the training data how the test error would be, see Figures 3.2,

3.3. The first row of panels corresponds to the results obtained by having bootstrapping with replacement, whereas the bootstrapping without replacement plots are on the second row. In the first column the test error versus test diversity is plotted, whereas in the second column the test error versus training diversity is shown.

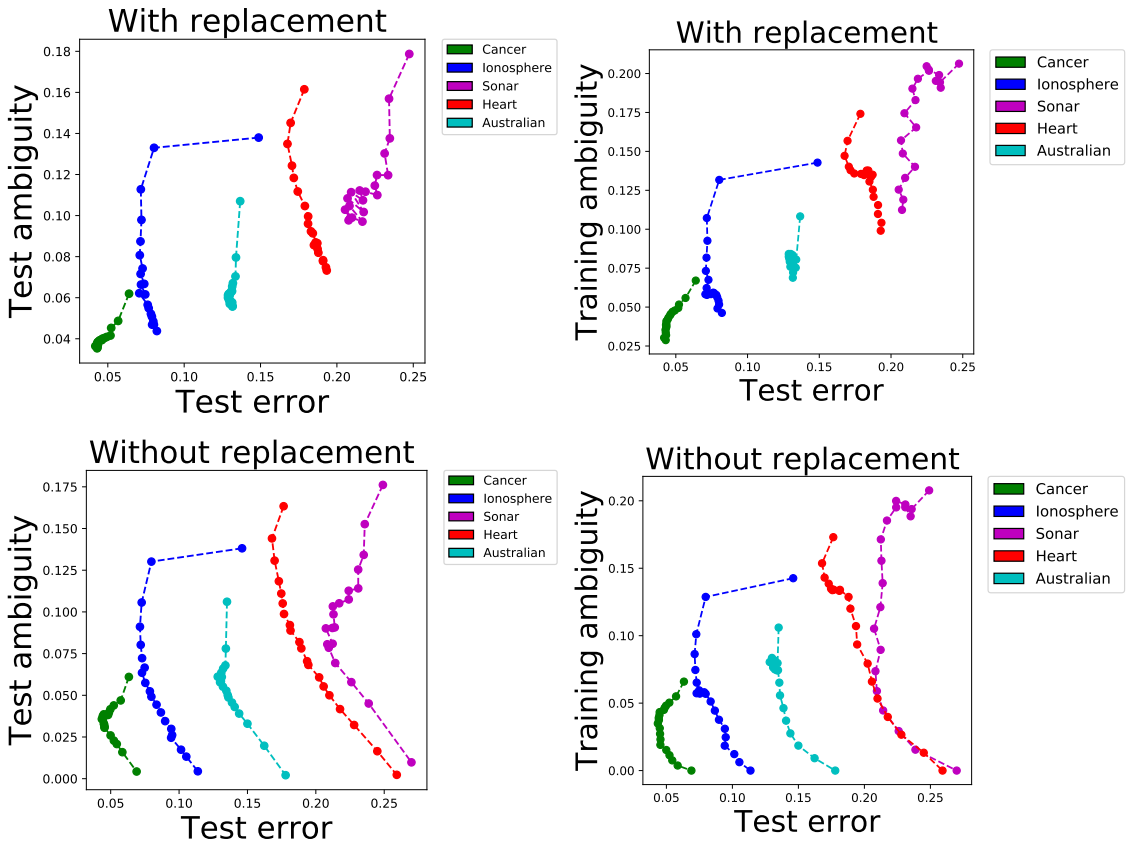

Figure 3.2. Correlation of the training or test amb_{01} with the test error, evaluated on different datasets. The first row from the panel corresponds to the values obtained by having bootstrapping with replacement, whereas bootstrapping without replacement in the second row respectively. The relationship between test amb_{01} and test error is shown in the first column, whereas training amb_{01} and test error in the second column. The points on the curves represent the sampling rates used for selecting portions of data for training the decision trees.

For each dataset we tried to determine the threshold values up to which diversity was negatively correlated with the test error. However, these thresholds varied per dataset. Therefore it is difficult to determine the threshold after which the correlation ceases to exist. However, for all these datasets considered, the curves generated displayed an approximate correlation of the test error with the diversity measure, emphasized by the knee of the curve. The variation in actual test errors across datasets can be justified by the fact that each dataset may have a different Bayes error. The Bayes error is the lowest possible error that any classifier can achieve for a given classification problem. These plots show that in general, at low diversity there is negative correlation between diversity and test error. The same behaviour can be noticed also from the plots of the training diversity versus the test error.

We have also compared the performance of bootstrapping with or without replacement on the test error in Figure 3.4. We can observe from these plots that in the case of the training error, the bootstrapping with no replacement achieves the lowest error. When the

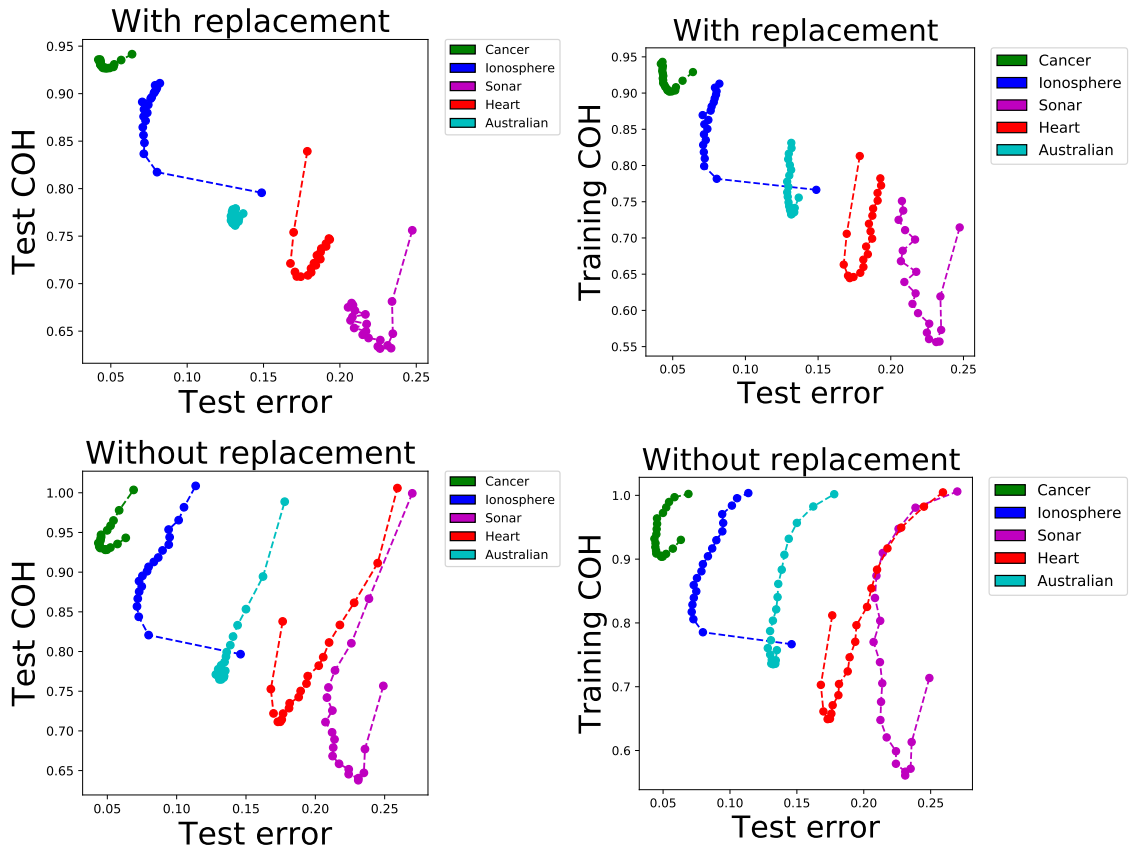

Figure 3.3. Correlation of the training or test *COH* with the test error, evaluated on different datasets. The first row from the panel corresponds to the values obtained by having bootstrapping with replacement, whereas bootstrapping without replacement in the second row respectively. The relationship between test *COH* and test error is shown in the first column, whereas training *COH* and test error in the second column. The points on the curves represent the sampling rates used for selecting portions of data for training the decision trees.

sampling rate is the highest, $r = 1$, therefore the whole data is used in fitting the trees, the training error is 0, which makes sense. However, in the case of the test error, the two types of bootstrapping have similar behaviour for most of the sampling rates, except the high ones.

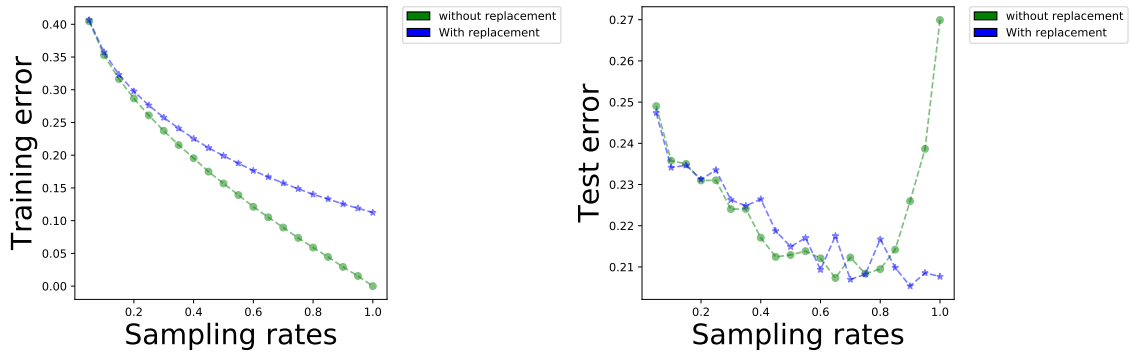

Figure 3.4. Comparison between bootstrapping with and without replacement. The left plot in the panel shows the training error versus the sampling rates for the two types of bootstrapping. The right panel presents the relationship between the sampling rates and the test error. The results for the bootstrapping without replacement are presented in green and blue for bootstrapping with replacement, respectively. These results were obtained for the Sonar dataset.

In order to check the previous results, we calculated the correlation levels of these diversity

measures with the test error. We used non-parametric Spearman's rank correlation [99] in order to assess the correlation. Spearman's correlation is used to assess the monotonicity between two pairs of observations, X, Y. These observations are ranked from lowest to highest and in the case there are no ties in the ranking, the formula for the Spearman's rank correlation is the following:

$$\rho = 1 - \frac{6 \sum_{i=1}^{N} d_i^2}{n(n^2 - 1)} \tag{3.9}$$

where $d_i = R(X_i) - R(Y_i)$ and $R(X_i)$ is the rank of the i^{th} value of observation X.

In the case there are ties in the ranking, the following formula is being used:

$$\rho = \frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum (X_i - \bar{X})^2 (Y_i - \bar{Y})^2}} \tag{3.10}$$

where $\bar{X}$ is the mean value of all the observations from X. The rank $r_s \in [-1, 1]$, positive values showing a positive correlation, negative a negative one and 0 no correlation. The correlation between test diversity and test error was calculated on all the datasets from Table 3.1. The values were then averaged across all datasets and the results are presented in Table 3.2. The same experiments were done in the case of training diversity and test error; the results are presented in Table 3.3.

Test diversity	Ensemble test error	
	Without replacement	With replacement
amb_{01}	-0.52	-0.03
CFD	-0.65	-0.08
DIS	-0.51	-0.001
KW	-0.51	-0.001
GD	-0.55	-0.07
COH	0.55	0.04

Table 3.2. Mean correlation between different diversity measures and test error, across all datasets from Table 3.1. The diversity was measured on the test data. The correlation was evaluated using the non-parametric Spearman's rank method. The first column displays the results for bootstrapping without replacement, whereas the second for bootstrapping with replacement.

Training diversity	Ensemble test error	
	Without replacement	With replacement
amb_{01}	-0.47	0.065
CFD	0.23	0.0003
DIS	-0.51	-0.0003
KW	-0.51	-0.0003
GD	0.44	0.0008
COH	0.55	0.0017

Table 3.3. Mean correlation between different diversity measures and test error, across all datasets from Table 3.1. The diversity was measured on the training data. The correlation was evaluated using the non-parametric Spearman's rank method. The first column displays the results for bootstrapping without replacement, whereas the second for bootstrapping with replacement.

By analysing these two tables, we can observe that the amb_{01} has a one of the strongest correlations with the test error, in the case of bootstrapping without replacement. It is

not well understood why this diversity measure has the strongest correlation with the generalisation error, one possible empirical justification could be found in [1]. The authors have generated a vector of 181 sampling rates, generated ensembles and for each of them, calculated the ensemble diversity and test error. The diversity measures considered were the same ones as in Table 3.2, except for the COH measure. They have projected these 181 element vectors in a two dimensional plane, by using the multidimensional scaling technique [100]. In general the amb_{01} measure was the closest in the plane to the generalisation error, than the other diversity measures, showing a stronger correlation.

There is a strong negative correlation between the diversity measures and the test error for the bootstrapping without replacement. However this correlation decreases significantly for the bootstrapping with replacement. This phenomenon could be explained by the fact that bootstrapping with replacement can allow some data points to be selected multiple times, leading to a decrease in diversity and an increase in the bias of the model. In contrast, bootstrapping without replacement enhances diversity in the ensemble and reduces the bias, contributing to a better generalisation performance. It has lower variance being more similar to the original dataset than bootstrapping with replacement and ensures greater stability by giving robust models.

Next we will present the results obtained by selecting a random subset of features.

3.3.2 Varying the subfeatures

More diverse decision trees can be created by selecting a random subsets of features. Therefore, we repeated the experiment in the case of a random number of features. However, the results were similar, as shown in Figures 3.5, 3.6.

In our experiments the number M of trees was fixed, therefore a natural question could be "is $M = 100$ a representative number for this analysis?". In the next section, we varied the number of trees in the forest and analysed the effect on the correlation between diversity and test error.

3.3.3 Varying the size of the forest

We repeated the experiments described in Section 3.3.1, by varying the number of classifiers. We ran the experiment for $M \in \{10, 100, 1000\}$, results are presented in Figure 3.7.

The first row of panels shows the relationship between test diversity and test error for all the datasets considered before. The second row illustrates the relation between training diversity and test error across all datasets. The first column presents the results for $M = 10$ trees, the middle for $M = 100$, whereas the last for $M = 1000$.

The results showed that there was a stronger correlation between diversities and test error for ensembles of 100 or 1000 trees and also the lowest ensemble test errors were obtained for the forests of 100 or 1000 trees. Increasing the number of trees and combining their predictions can contribute to reducing overfitting and variance, which will make the ensemble's predictions more robust and reduce generalisation error. At the same time, increasing the number of trees in the forest might lead to a better coverage of the feature space. The ensemble can capture the relationship between different features and patterns

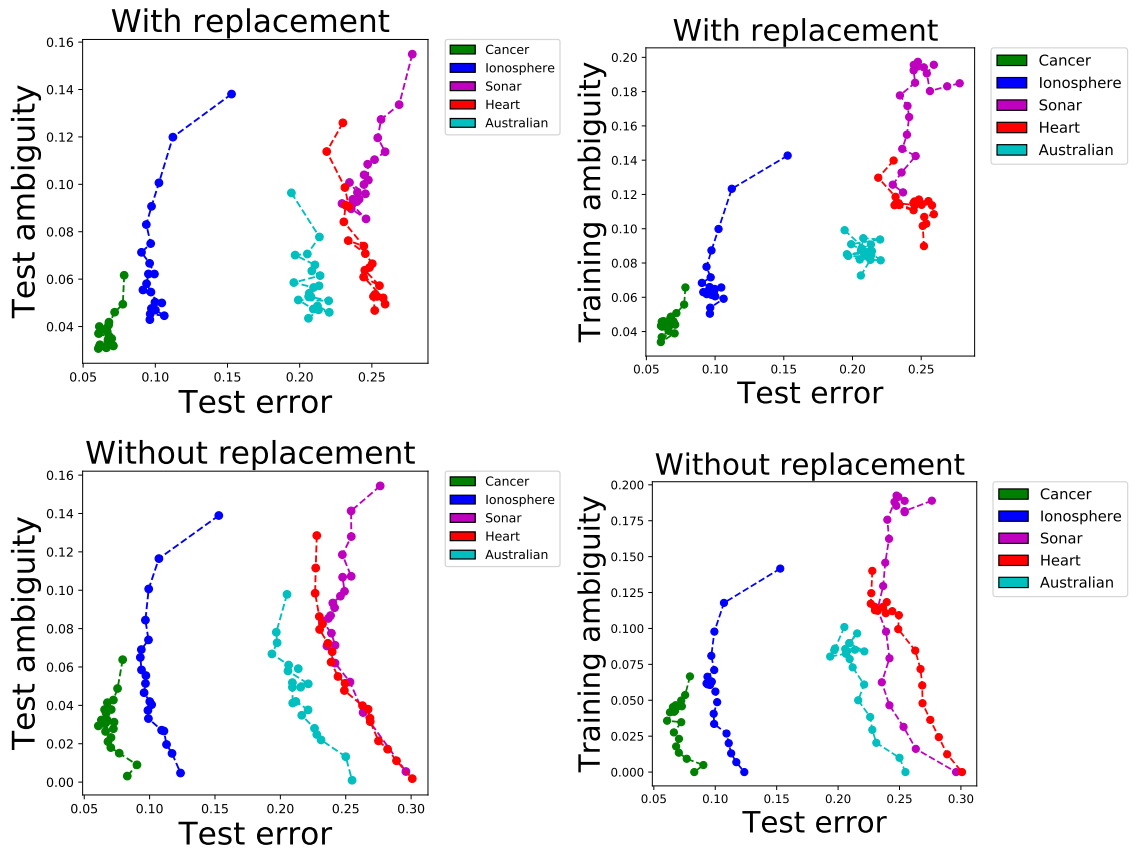

Figure 3.5. Correlation of the training or test amb_{01} with the test error, evaluated on different datasets. The forests were obtained from varying the subset of features. The first row from the panel corresponds to the values obtained by having bootstrapping with replacement, whereas bootstrapping without replacement in the second row respectively. The relationship between test amb_{01} and test error is shown in the first column, whereas training amb_{01} and test error in the second column. The points on the curves represent the sampling rates used for selecting portions of data for training the decision trees.

combinations, making it more sensitive to capturing the diversity of the data, which can lead to a stronger correlation between diversity and predictions.

Our results showed that diversity was correlated with the test error only up to a threshold as in [1]. An alternative metric of performance for a classifier could be the AUC (area under the Receiver Operating Characteristic curve, which is a measure of how well a classifier is able to separate the classes over the full range of misclassification costs [101]), therefore in the next section we analysed the relationship between diversity and area under the curve, to check if there was a different behaviour.

3.3.4 Diversity versus AUC

The receiver operating characteristic curve (ROC) connects the points of the true positive rates versus the false positive rate of a classifier, obtained for a range of thresholds. The AUC measures the area under the ROC curve. A classifier that has a good classification rate will have the AUC close to 1, whereas an AUC= 0.5 denotes a behaviour similar to random guessing. An example of ROC curve with its corresponding AUC is displayed in Figure 3.8. The ROC curve obtained for a forest fitted on the GMM5 dataset is denoted in blue, whereas random guessing is displayed in red with an AUC of 0.5.

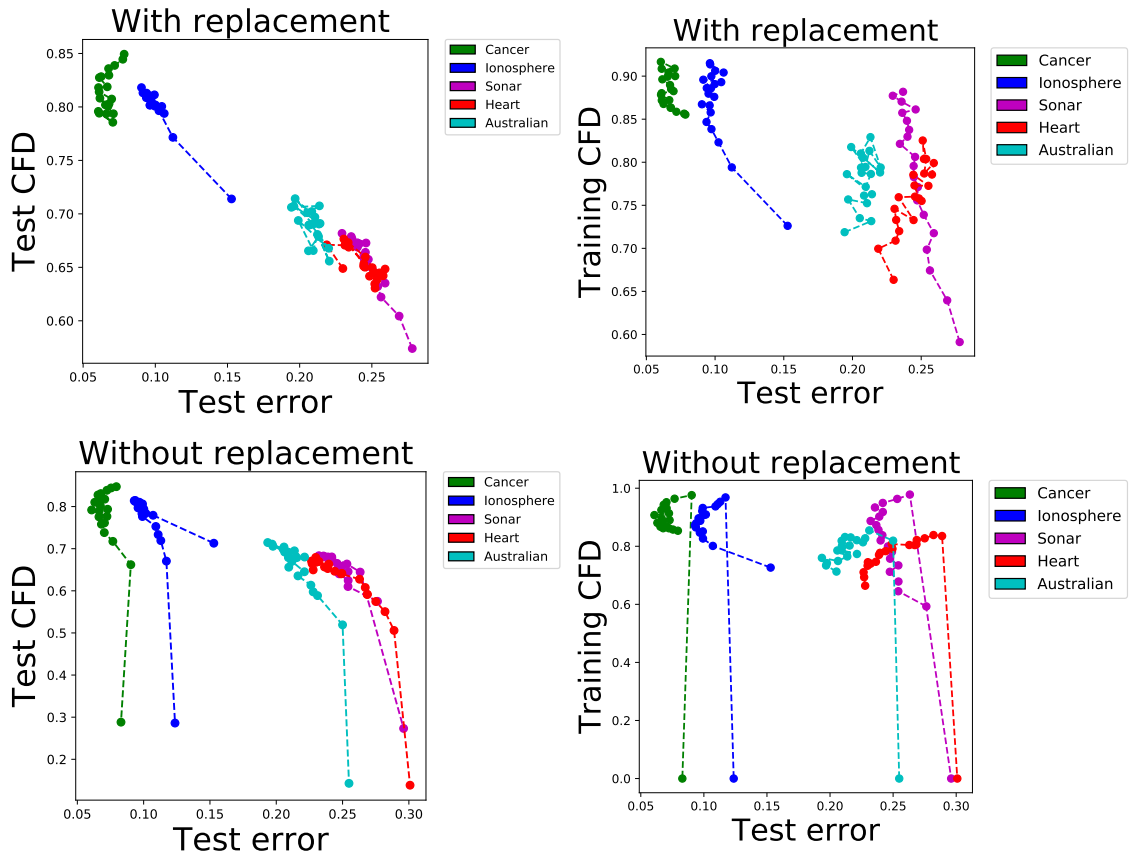

Figure 3.6. Correlation of the training or test CFD with the test error, evaluated on different datasets. The forests were obtained from varying the subset of features. The first row from the panel corresponds to the values obtained by having bootstrapping with replacement, whereas bootstrapping without replacement in the second row respectively. The relationship between test CFD and test error is shown in the first column, whereas training CFD and test error in the second column. The points on the curves represent the sampling rates used for selecting portions of data for training the decision trees.

We plot the test diversity versus the test AUC and the training diversity versus test AUC in Figures 3.9, 3.10. However the behaviour encountered was similar to the previous case, when using test error.

3.4 Diversity zones

In [1] the author presented the following relationship between the ensemble generalisation error, test ambiguity and average test error of the classifiers.

$$L_{01}(Y_n t_n) = \sum_{i=1}^{M} c_i L_{01}(y_{in} t_n) - \frac{t_n}{2} \sum_{i=1}^{M} (\frac{1}{M} Y_n - c_i y_{in}) \tag{3.11}$$

where L_{01} is the 0-1 loss, M is the number of trees/classifiers, Y_n the ensemble prediction for the data point x_n and c_i are the weights of the classifiers. In [1] the authors have compared the ensemble error with the average error of the classifiers and ensemble diversity. Their analysis was conducted in the case of training and test data and for a variety of ranges. We have repeated the experiment. For 20 sampling rates ranging between 0.05 and 1, we generated ensembles of 100 trees. By using 5-fold cross validation, we evaluated the ensemble ambiguity and error and the average error of the trees, evaluated on the train or test data. This process was repeated 20 times. The average for each sampling

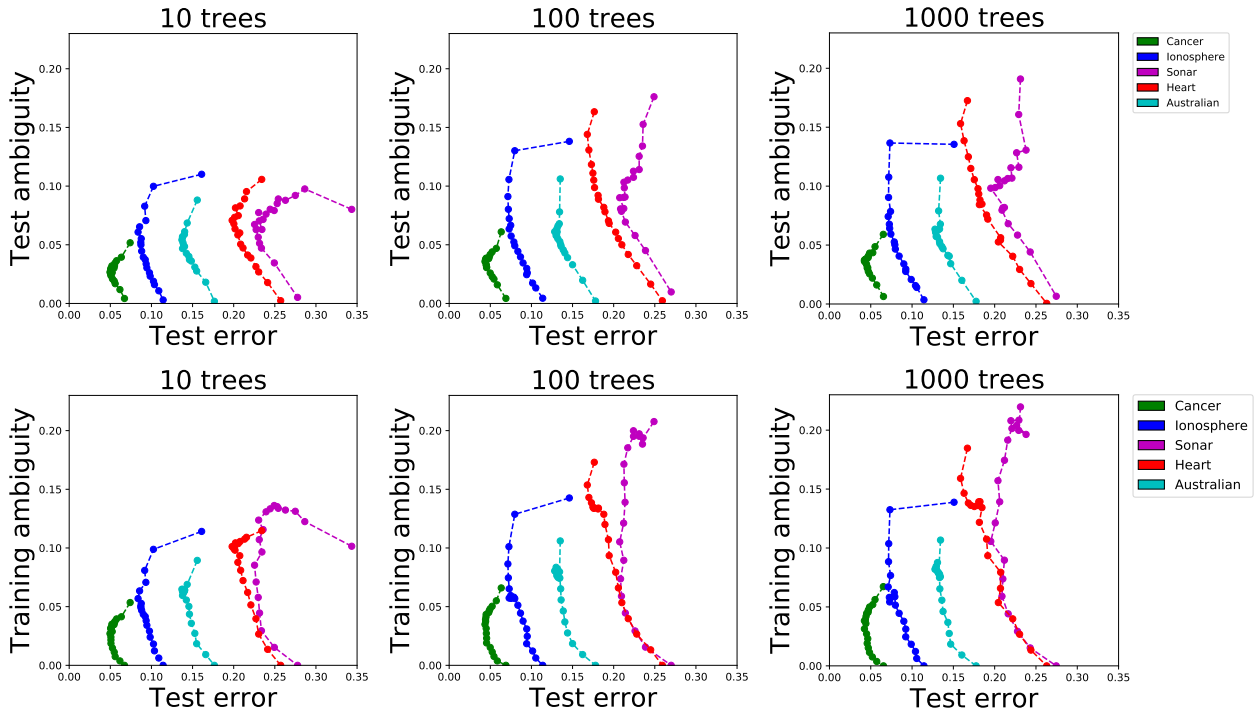

Figure 3.7. Correlation of the training or test amb_{01} with the test error, evaluated on different datasets. The test ambiguity is plotted in the first row, whereas the training ambiguity on the second row. The first column displays the relationship between ambiguity and test error for 10 trees, the middle for 100 trees and the last one for 1000 trees. The points on the curves represent the sampling rates used for selecting portions of data for training the decision trees.

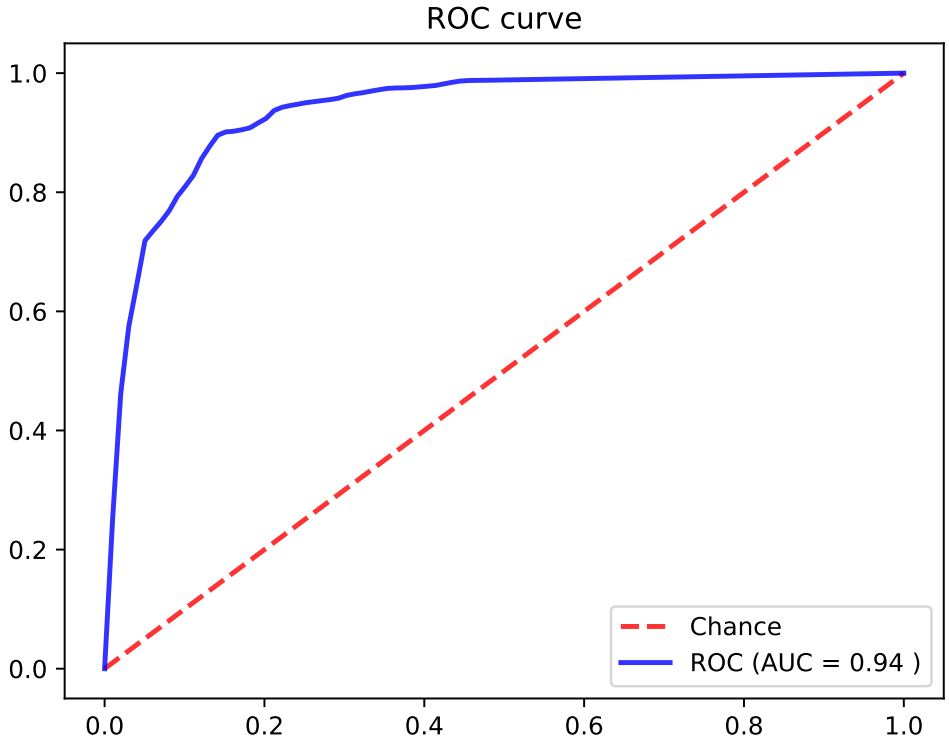

Figure 3.8. ROC curve of a forest of 3 trees obtained on the GMM5 dataset.

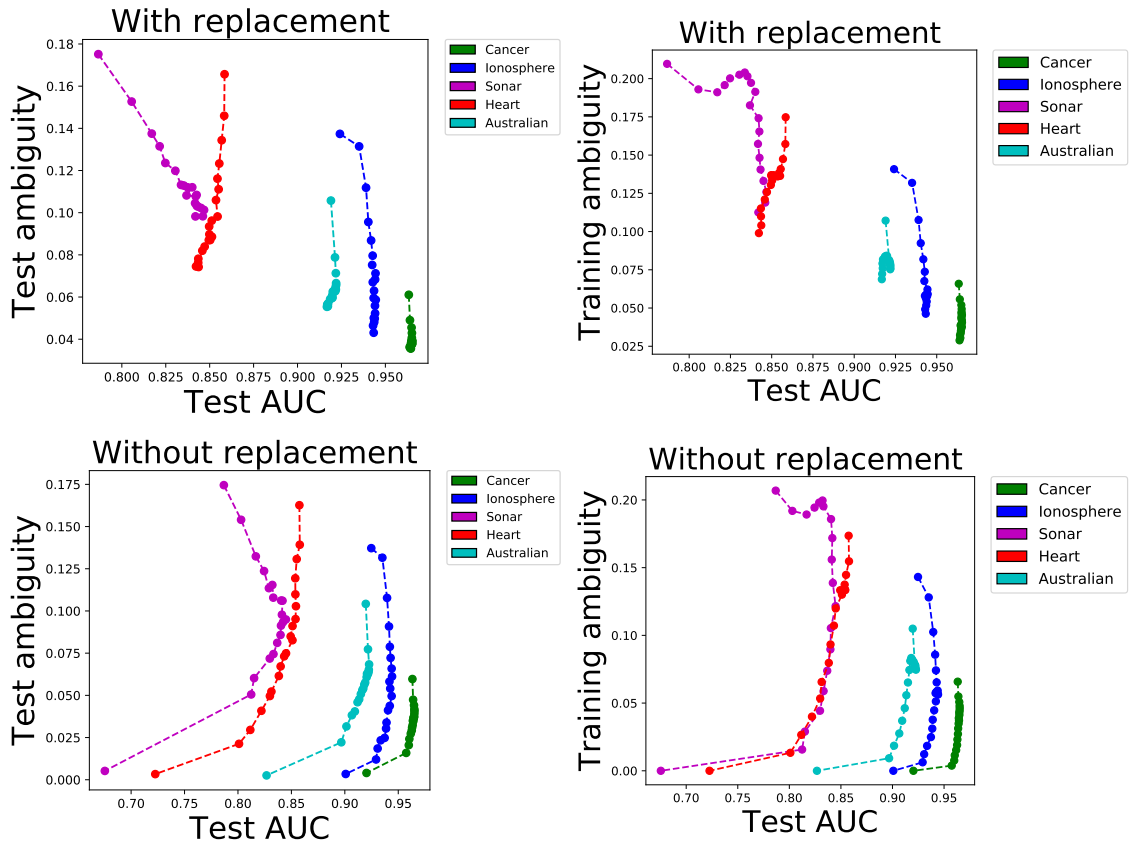

Figure 3.9. Correlation of the training or test ambiguity with the test AUC, evaluated on different datasets. The forests were obtained from varying the subset of features. The first row from the panel corresponds to the values obtained by having bootstrapping with replacement, whereas bootstrapping without replacement in the second row respectively. The relationship between test ambiguity and test AUC is shown in the first column, whereas training ambiguity and test AUC in the second column. The points on the curves represent the sampling rates used for selecting portions of data for training the decision trees.

rate of these three quantities versus the sampling rates are plotted in Figure 3.11. We have analyzed how diversity changes according to different sampling rates and divided the space into different diversity zones (similar to the ones in [1]). Taking into account Equation (3.11) (how diversity evolves along with ensemble error and average error), we obtained the following conclusions:

- Low diversity zone -when the sampling rate is high ($r \in [0.9, 1]$). The generalisation error decreases when changing the rate from 1 to 0.9, because the average error will not change too much whereas the diversity of the trees will increase.

- Medium diversity zone, which corresponds to medium sampling rates ($r \in [0.2, 0.9]$). The generalisation error will not change too much, as the average error and diversity will change at the same speed.

- High diversity zone, corresponding to low sampling rates ($r < 0.2$). In this zone the generalisation error will increase the most, because the average error will change faster than diversity.

Since the generalisation error decreases when the sampling rate is high, it suggests that

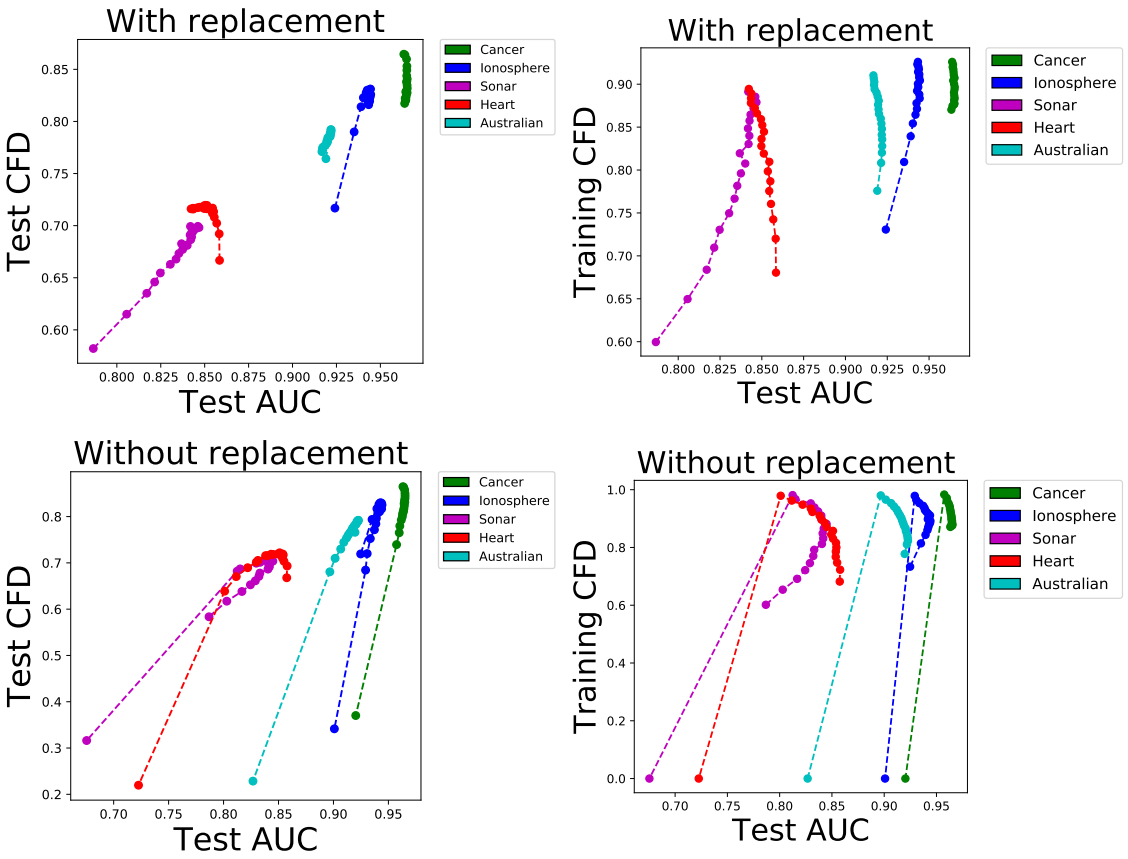

Figure 3.10. Correlation of the training or test CFD with the test AUC, evaluated on different datasets. The forests were obtained from varying the subset of features. The first row from the panel corresponds to the values obtained by having bootstrapping with replacement, whereas bootstrapping without replacement in the second row respectively. The relationship between test CFD and test AUC is shown in the first column, whereas training CFD and test AUC in the second column. The points on the curves represent the sampling rates used for selecting portions of data for training the decision trees.

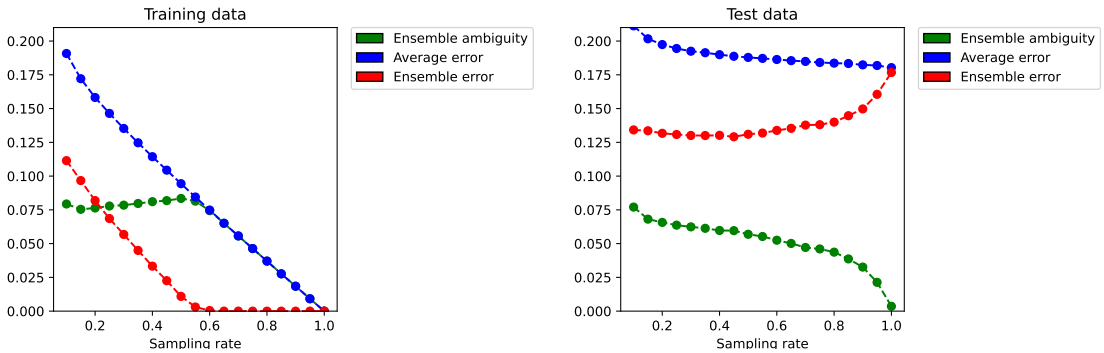

Figure 3.11. These plots present the behavior of the ensemble ambiguity, average error of the classifiers and ensemble error versus different sampling rates. In the first plot the training data was used in order to evaluate these quantities, whereas the test data in the second plot. These results were obtained for the Australian dataset.

the more data that are given, the more the error will decrease. In order to test this hypothesis, we split the training data into different slices and evaluated the ensemble error and ambiguity. We generated forests of 100 trees, trees that were fitted on one sixteenth of the data, one eighth, one quarter, one half and the whole data. The whole test data was used. We plot the ensemble test error versus the ensemble test ambiguity, the

test error versus the training ambiguity and finally the training error versus the training ambiguity in Figure 3.12. The reason why we produced these three plots was to see if it's possible to predict from the training data the point (the knee of the curve) where the negative correlation between ambiguity and error ceases to exist. Also on the plots are the sampling rates for which the specific values for ambiguity and error were obtained. The results confirmed the hypothesis, the more data is given the more the error will decrease. However, these experiments show that it is very difficult to reliably predict from the training data the position of the knee for the test data, for both of them the knee being found for different values:

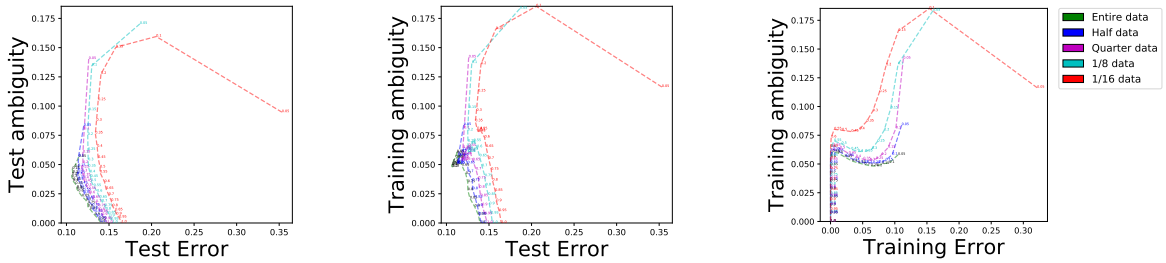

Figure 3.12. The figure in the left represents the test ambiguity versus test error of forests obtained by fitting a proportion of the data. The plot in the middle shows for the same forests their training ambiguity versus test error. The right presents the relationship between the training error and the training ambiguity of the same forests. These results were obtained for the GMM5 dataset

In this section we analyzed the effect that different diversity measures have on the generalisation error. The diversity measure that had the strongest negative correlation with the test error, was the ambiguity. We have generated ensembles by using bagging with different sampling rates. Our results show that ensemble test error is negatively correlated with diversity, when diversity is low. We have tried to determine the threshold up to which ensemble error is negatively correlated with diversity. We have generated ensembles by varying the features or the patterns considered. For both these experiments, we plotted the ensemble training error versus generalisation error, in order to determine that specific threshold. Our results showed that it is impossible to determine from the training data, what would this threshold be. We have repeated the experiment in the case of AUC, however the conclusion remained the same. Another conclusion of our experiments was that test error decreases at high sampling rates, implying that the more data is given, the lower the error will become. Therefore, we varied the ensemble size and evaluated the ensemble error on ensembles of different sizes. Our empirical results showed that larger ensembles yielded lower test errors.

In our experiments we fixed number of $M = 100$ trees. It has been shown that decreasing the number of trees can induce similar performance as well as decreasing the computational time when training models or predicting new data. Next we will present a number of pruning methods which involve ambiguity.

3.5 Pruning methods

In our next set of experiments we start with large ensembles and remove trees one by one with the goal of maintaining generalisation accuracy. Based on the decomposition from equation (3.11) we derive pruning methods which involve ambiguity. Next we will present

a mathematical relation between classifier error and ambiguity, which we will use in our pruning methods.

Equation (3.11) can be modified by adding a summation over all the patterns and obtaining the following equation:

$$L_{01}(Y) = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{M} c_i L_{01}(y_{in}t_n) - \frac{1}{N}\sum_{n=1}^{N}\frac{t_n}{2}\sum_{i=1}^{M}\left(\frac{1}{M}Y_n - c_i y_{in}\right) \tag{3.12}$$

where M is the number of classifiers (trees) and N the total number of patterns
By swapping the two summations, we obtain:

$$L_{01}(\mathcal{Y}) = \sum_{i=1}^{M} c_i \frac{1}{N}\sum_{n=1}^{N} L_{01}(y_{in}t_n) - \sum_{i=1}^{M}\frac{1}{N}\sum_{n=1}^{N}\frac{t_n}{2}\left(\frac{1}{M}Y_n - c_i y_{in}\right) \tag{3.13}$$

This equation can be interpreted that the ensemble error is equal to the average error of all the classifiers minus the sum of individual ambiguities for each classifier. As a result, we obtained:

$$L_{01}(y_i) = \frac{1}{N}\sum_{n=1}^{N} L_{01}(y_{in}t_n) \tag{3.14}$$

where $i \in 1..n$

$$\text{amb}_{01}(y_i) = \frac{1}{N \cdot M}\sum_{n=1}^{N}\frac{t_n}{2}(Y_n - y_{in}) \tag{3.15}$$

In our experiments all classifiers have equal weights, $c_i = \frac{1}{M}$.

Based on Equation (3.11), we can hypothesise that by removing the classifier with the higher difference between its individual error and its ambiguity, the ensemble error will be reduced. To investigate this we plotted the individual error of each classifier versus their individual ambiguity in Figure 3.13.

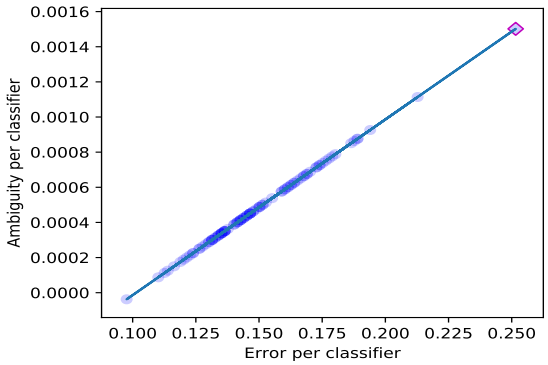

Figure 3.13. Cloud of classifier error vs classifier ambiguity

The square at the top of the line corresponds to the classifier with the highest difference between its individual error and its ambiguity. We investigated the reason why all the points were situated on a straight line and found the following result:

Theorem 2. *Given an ensemble of M classifiers, where Y represents the ensemble outputs and y_i, $i = 1..M$, denote individual classifier outputs. Let L_{01} denote the 0-1 loss function and amb_{01} represent the ambiguity derived from this loss. Then, the following statement holds:*

$$L_{01}(y_i) - M \cdot \text{amb}_{01}(y_i) = L_{01}(Y) \tag{3.16}$$

Proof. If we define the error function as in [1]:

$$L_{01}(x) = \begin{cases} 1 & \text{if } x = -1 \\ 0.5 & \text{if } x = 0 \\ 0 & \text{if } x = 1 \end{cases}$$

where $x = 0$ means the classifiers gave equal votes for both classes
Following the above definition of $L_{01}(x)$, we find that:

$$L_{01}(x) = -\frac{1}{2}(x-1) \Rightarrow 2 \cdot L_{01}(x) + x = 1 \tag{3.17}$$

Using Equations (3.14), (3.15) in (3.16), we obtain:

$$L_{01}(y_i) - M \cdot \text{amb}_{01}(y_i) = \frac{1}{N}\sum_{n=1}^{N} L_{01}(y_{in}t_n) - \frac{t_n}{2}Y_n + \frac{t_n}{2}y_{in}| \cdot 2$$

$$2L_{01}(y_i) - 2M \cdot \text{amb}_{01}(y_i) = \frac{1}{N}\sum_{n=1}^{N} \underbrace{2 \cdot L_{01}(y_{in}t_n) + t_n y_{in}}_{1} - \underbrace{t_n Y_n}_{1 - 2 \cdot L_{01}(t_n Y_n)} | \cdot 2$$

$$2L_{01}(y_i) - 2M \cdot \text{amb}_{01}(y_i) = \underbrace{\frac{1}{N}\sum_{n=1}^{N} 2 \cdot L_{01}(t_n Y_n)}_{2 \cdot L_{01}(Y)} | : 2$$

$$L_{01}(y_i) - M \cdot \text{amb}_{01}(y_i) = L_{01}(Y)$$

□

In our experiments we used forests of $M = 100$ trees, however is has been shown in [81] that a similar performance in a forest can be achieved with a smaller number of trees, via pruning. Generalisation error is very important in machine learning, but also the computational time for training/predicting new data is important, which increases proportionally with the number of models. Therefore finding the optimal number of classifiers is important in the classification process. In the next section, we will analyse different ambiguity based pruning techniques.

3.6 Ambiguity pruning methods

Large ensembles increase the computational time, when making new classifications or when building the model. Therefore, smaller ensembles would be preferred as long as the generalisation error is not severely affected. Smaller ensembles are also preferred due to the memory footprint.

In order to minimise the computational time and to achieve similar classification per-

formance we have developed a series of pruning techniques. We will use the following notations. Let $Y_m = \{y_i\}_{i=1}^m$ be a collection of m classifiers, where $2 \leq m \leq M$ and M is the number of classifiers from the unpruned ensemble. A new ensemble Y_{m-1} is formed by removing one of the classifiers. We define different approaches to select the classifier to be removed from Y_m:

- Remove successive worse. Inspired by the previous hypothesis, the trees that have the highest difference between individual error and individual ambiguity are removed, in order to reduce ensemble error. Mathematically, this approach can be expressed as
$$Y_{m-1} = Y_m \setminus argmax_{y \in Y_m} \left(L_{01}(y) - \text{amb}_{01}(y)\right)$$

- Linear random. At each iteration a random tree is removed. Formally, this method can be defined as
$$Y_{m-1} = Y_m \setminus RandomChoice(Y_m)$$

- Remove successive best. Opposite to the first method, it removes the tree with the smallest difference between individual error and individual ambiguity. In the previous method, the hypothesis was that removing the trees with the smallest difference between individual error and ambiguity would decrease the ensemble training error. This approach tests the opposite effect in order to determine if the previous method overfits and if the current one would have a more significant impact on the generalisation error. Mathematically, this method can be described as :
$$Y_{m-1} = Y_m \setminus agmin_{y \in Y_m} \left(L_{01}(y) - \text{amb}_{01}(y)\right)$$

We have pruned a forest of 100 trees according to these methods and evaluated the training and test error of the new formed ensembles.

We have used 5-fold cross validation and repeated the process 20 times. We have plotted the mean test error, along with the 25th and 75th quartile versus the number of trees. Our results showed a large variability between the two quartiles. Therefore, in order to determine if the variability was due to the change in the data used to fit the models, we repeated the experiment in two scenarios: when the same data was used for building the models, or when different data was used. By using the same data we mean, that the same training and test data was used, 4/5th for training and 1/5th for testing of the original data, (we chose the last fold of the 5-fold cross validation). In this case the variability was given by the ensembles created, which we term conditional variability, because it's conditioned on knowing the data set. The different data scenario implied using the training and test data of each fold, to fit and evaluate the models.

The evolution of the average ensemble error versus the number of trees is presented in Figures 3.14-3.17.

We chose to investigate the effect of providing more or less data has, when applying the pruning methods, therefore, we run the experiment for initial forests generated using low

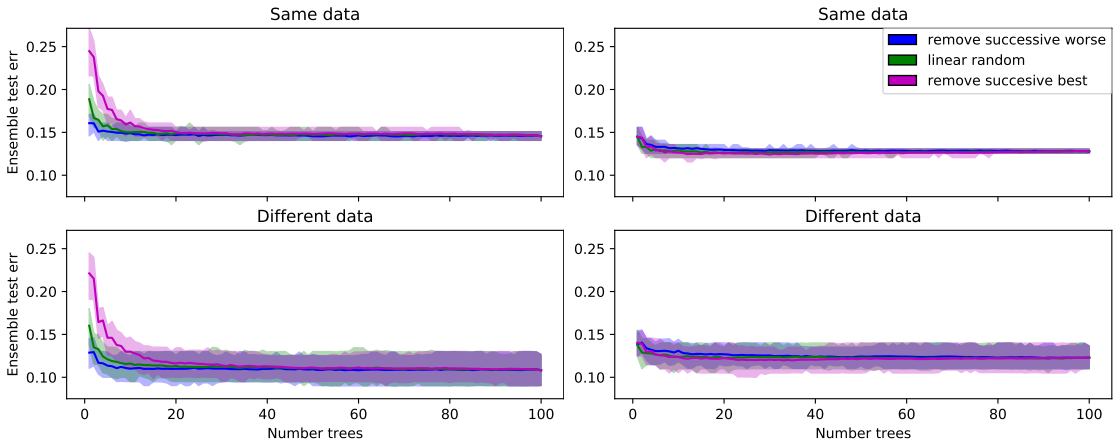

Figure 3.14. Comparisons of the pruning approaches for the Gmm5test dataset, for sampling rates 0.1 and 0.75. The results for the 0.1 sampling rate are presented in the first column, whereas the 0.75 sampling rate in the second column. The top figure of each column displays the results when the same data was used, whereas the bottom plot when different data was used.

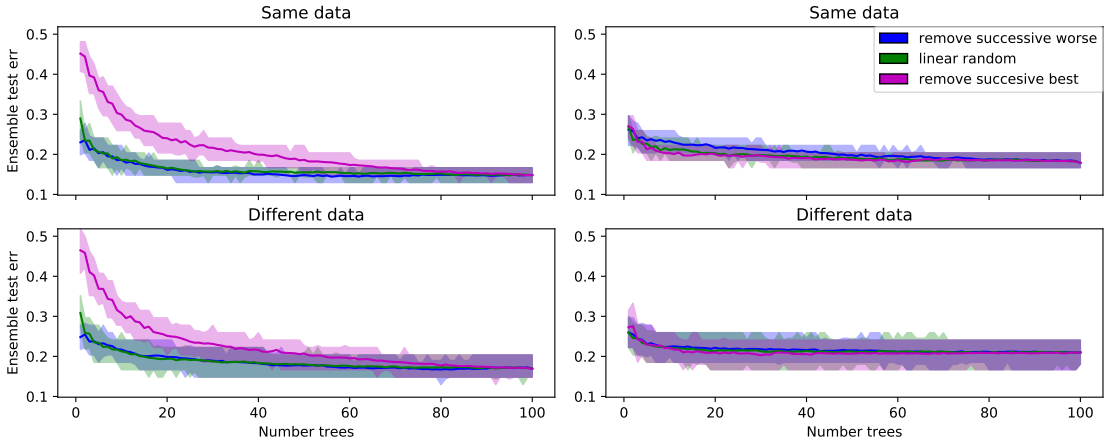

Figure 3.15. Comparisons of the pruning approaches for the Heart dataset, for sampling rates 0.1 and 0.75. The results for the 0.1 sampling rate are presented in the first column, whereas the 0.75 sampling rate in the second column. The top figure of each column displays the results when the same data was used, whereas the bottom plot when different data was used.

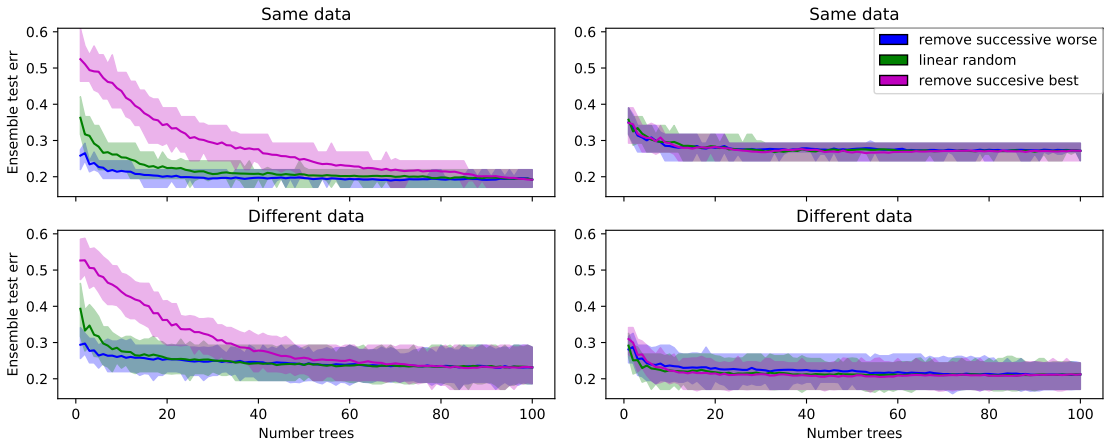

Figure 3.16. Comparisons of the pruning approaches for the Sonar dataset, for sampling rates 0.1 and 0.75. The results for the 0.1 sampling rate are presented in the first column, whereas the 0.75 sampling rate in the second column. The top figure of each column displays the results when the same data was used, whereas the bottom plot when different data was used.

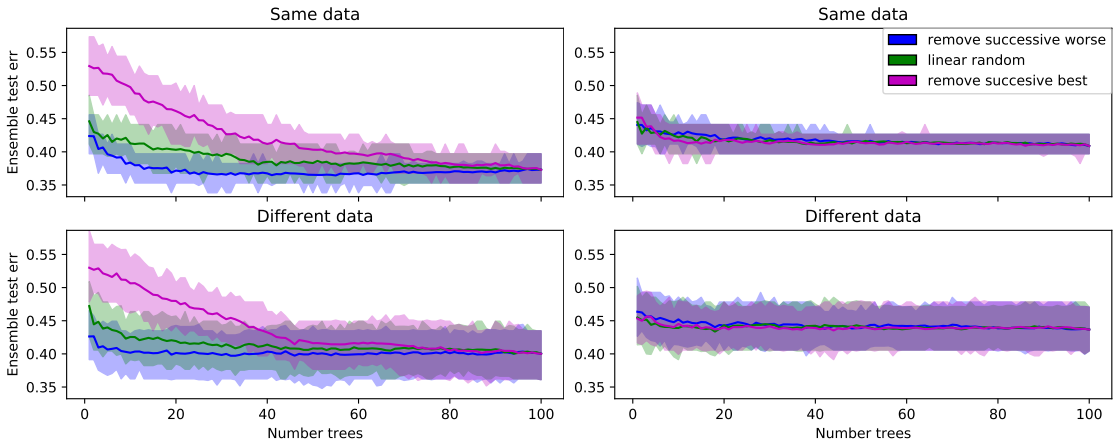

Figure 3.17. Comparisons of the pruning approaches for the Ionosphere dataset, for sampling rates 0.1 and 0.75. The results for the 0.1 sampling rate are presented in the first column, whereas the 0.75 sampling rate in the second column. The top figure of each column displays the results when the same data was used, whereas the bottom plot when different data was used.

and high sampling rates (0.1 and 0.75).

In the left hand side of each panel, we have plotted the results obtained for the 0.1 sampling rate. The results for the 0.75 sampling rate are presented in the right side of the figures.

The first row of each panel, displays the results when the same data was used to build ensembles. The results obtained when different data was used, are presented on the second row.

For the 0.75 sampling rate the number of trees needed seems to be generally fewer than about 40 (although different in exact number for different data sets). So all the errors with more than this number of trees are about the same, regardless of how you remove the trees. Presumably this is because the high sampling rate means that with about 20 or 40 trees the ensemble has effectively seen all the data.

With the 0.1 sampling rate, the "remove successive best" is always worse on the test data than the other methods. This could be explained by the fact that if you remove the tree with the smallest error then the remaining ensemble is made of trees with larger errors, so the resulting ensemble error is larger. Note that removing the tree with the largest difference between individual error and individual ambiguity is equivalent to removing the tree with the largest individual error because the individual error and ambiguity are proportional to each other (see Equation (3.16)).

With the 0.1 sampling rate, "remove successive worse" (the blue line) is better than the other methods in some cases (for the same data case and for small ensembles), but at high sampling rates there is not a statistical difference.

The variability in both cases seems to be smaller for higher sampling rates, due to the fact that higher sampling rates imply providing more data to the models, therefore having a better view of the data and being able to make a better classification.

Overall the average test errors are similar for all cases, however in the case of smaller

sampling rates, you need more trees — which makes sense because you need more trees to "see" the entire dataset and there is more diversity in the ensemble because the trees see more different slices of the data. In general, the two approaches (using the same or different data) seem to provide similar results, in terms of the effect on ensemble test error across the three pruning methods discussed.

We have repeated the same experiment in the case of the coherence diversity measure (see equation 3.8). The pruning approaches considered were the same, with the only differences that the "remove successive best" and "remove successive worst" methods involved coherence instead of ambiguity. These approaches are defined as follows:

- Remove successive worse:

$$Y_{m-1} = Y_m \setminus argmax_{y \in Y_m} \left(L_{01}(y) - COH(y) \right)$$

- Remove successive best:

$$Y_{m-1} = Y_m \setminus agmin_{y \in Y_m} \left(L_{01}(y) - COH(y) \right)$$

The results of the experiment for the Ionosphere dataset are displayed in Figure 3.18. These plots show that the same conclusions as before hold, that there is no statistical difference between the random approach and the ones involving a diversity measure (coherence or ambiguity). Even though Figure 3.18 displays the results in the case of the Ionosphere dataset, similar results were obtained for the other datasets considered (the remaining plots are presented in the Appendix A.1).

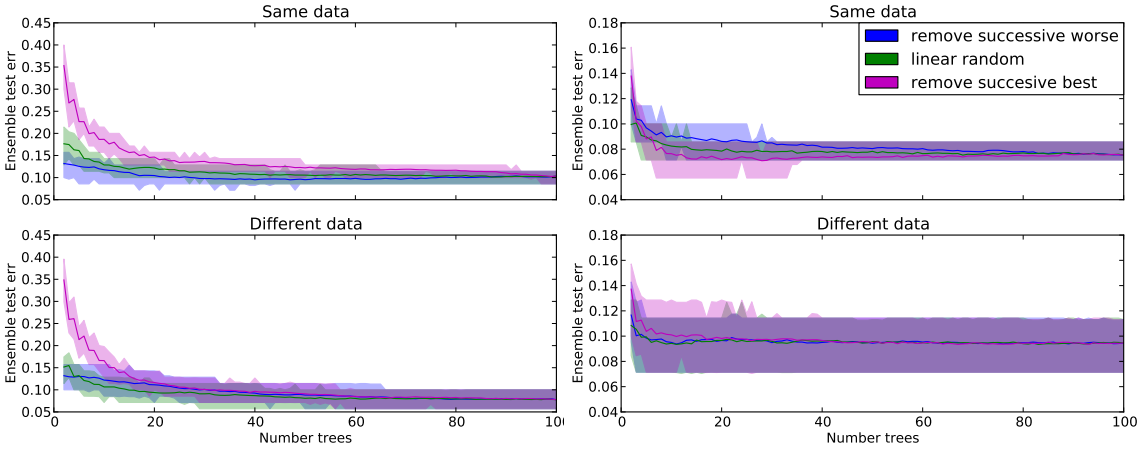

Figure 3.18. Comparisons of the pruning approaches involving the coherence diversity measure for the Ionosphere dataset, for sampling rates 0.1 and 0.75. The results for the 0.1 sampling rate are presented in the first column, whereas the 0.75 sampling rate in the second column. The top figure of each column displays the results when the same data was used, whereas the bottom plot when different data was used.

In this section, we have defined a series of pruning techniques that involved removing either random classifiers, either the ones with the highest/smallest difference between individual error and ambiguity/coherence. We have investigated the effect of these techniques in two scenarios: when the same training/test data was provided to the models or when different

data was used. We have compared the results obtained, when more data was provided (0.75 sampling rate) or when less data was used (0.1 sampling rate).

Our results have shown that in the case of the higher sampling rates, the approaches are not statistically different, whereas when less data was used, the "remove successive worse" approach was slightly better than the other approaches. However, in general the pruning approaches that favour diverse ensembles, are not statistically different than random ensembles.

3.7 Conclusion

We have investigated the correlation that different diversity measures have on the generalisation error. The diversity measures considered were: ambiguity, coincident failure, disagreement, Kohavi-Wolpert and a new measure, called coherence. The coherence measures the angle between the prediction of a classifier and the ensemble's prediction. Our results show that the diversity measure that had the strongest negative correlation with the generalisation error was Chen's ambiguity, followed by the DIS and KW. Another conclusion of these experiments was that at high diversity there is little consistent correlation with the test error.

We have compared these correlations in the case of ensembles formed by bootstrapping with or without replacement. Our results show that at low diversity there is a negative correlation between diversity and test error, particularly for ensembles with bootstrap without replacement. Another conclusion was that bootstrapping with replacement leads to ensembles for which the test error is not correlated with diversity. On the whole bootstrapping without replacement generates ensembles with lower test error. We have also investigated the effect that providing more data has on the generalisation error. The conclusion was that by increasing the size of the dataset the generalisation error will decrease.

Our next set of experiments focused on analysing the effect of different pruning techniques on the generalisation error. The pruning methods considered involved either discarding random trees or selecting trees based on the lowest or highest difference between individual error and ambiguity/coherence. Our results show that, in general, there is no significant difference between removing random trees and the approaches involving ambiguity or coherence.

Chapter 4

Optimising diversity in ensembles of classification trees

4.1 Introduction

In the previous chapter we examined the relationship between the 0-1 loss ambiguity measure defined by Chen (equation 3.2, [1]) and the generalisation error.

In this chapter we further explore the connection between ensemble diversity and generalisation error. Following [21, 1], we define and characterise new ambiguity measures appropriate for the log loss and hinge loss. We investigate empirically the relationship between these ambiguities and the generalisation error. This leads to an evolutionary algorithm for the direct maximisation of the ensemble ambiguity, and indirectly the generalisation error, by optimisation of the patterns that each ensemble member is trained on.

The principal contributions of this chapter are as follows:

1. the derivation of a cross-entropy-based ambiguity measure for ensemble diversity;
2. the derivation of a hinge-loss-based ambiguity measure for ensemble diversity;
3. the empirical assessment of the ambiguity/generalisation error trade-off on a number of widely used classification data sets, using decision trees ensembles;
4. the exploration of the effect of ensemble sampling rates on this trade-off;
5. the exploration of the direct *maximisation* of ensemble ambiguity via an evolutionary optimisation of the training patterns as a way to maximise generalisation performance.

In the next section we present different diversity measures for ensembles using log and hinge losses and establish some of their properties. Section 4.3 analyses the correlation between the ambiguity measures presented and ensemble test error. Section 4.4 presents an evolutionary algorithm for the optimisation of the cross-entropy diversity. Section

4.5 illustrates the performance of the evolutionary optimiser on a range of classification problems. Section 4.6 presents the conclusions.

4.2 Ambiguity measures

Extending the idea of quantifying diversity in regression ensembles [21], Chen [1] by using the 0-1 loss defined a new classifier ensemble diversity measure in terms of how diverse the outputs of the constituent classifiers are compared with the ensemble prediction. Following this line, by employing the log and hinge loss we define new diversity measures as the difference between the average error of the individual classifiers forming the ensemble and the ensemble error; that is we define the ambiguity through the simple relation:

$$\text{Ensemble error} \quad = \quad \text{Average error} \quad - \quad \text{Ambiguity} \tag{4.1}$$

In line with [1], we call these measures of diversity *ambiguity measures*. We define new ambiguities for the log loss and hinge loss and present their properties in the following sections.

4.2.1 Ambiguity measure for log loss

The cross-entropy error or log loss measures the discrepancy between the output of the classifier and the true class when the classifier produces an output between 0 and 1 which may be interpreted as a posterior probability; for convenience we denote the classes as 0 and 1, $t_n \in \{0, 1\}$.

Definition 4.2.1. *The ambiguity measure derived from the log loss has the following formula:*

$$\text{amb}_{CE}(\mathcal{Y}_n) \triangleq t_n \log\left(\frac{\sum_{i=1}^{M} c_i y_{in}}{\prod_{i=1}^{M} y_{in}^{c_i}}\right) + (1 - t_n)\log\left(\frac{\sum_{i=1}^{M} c_i (1 - y_{in})}{\prod_{i=1}^{M} (1 - y_{in})^{c_i}}\right). \tag{4.2}$$

Note that for any t_n only one of the terms will not be zero, so $\text{amb}_{CE}(\mathcal{Y}_n)$ is the logarithm of the ratio between the arithmetic and geometric means of the proximity of the classifiers' outputs to the desired targets. The cross entropy ambiguity for many patterns is just the ambiguity averaged over patterns (4.3):

$$\text{amb}(\mathcal{Y}) = \frac{1}{N}\sum_{n=1}^{N} \text{amb}(\mathcal{Y}_n) \tag{4.3}$$

We note Woodhouse [102] shows that the ratio of the arithmetic mean to the geometric mean is equivalent to a cross-entropy quantifying the amount of information added in an image processing problem. In addition in [103], [104] the ratio of the arithmetic to geometric mean is used to measure homogeneity. A ratio close to 1 might indicate homogeneity between the samples, whereas a value greater than 1 indicates heterogeneity.

Next we will enumerate and prove the properties of the amb_{CE} ambiguity.

Theorem 3. *The log loss ambiguity,* amb_{CE}*, has the following properties:*

1. $\text{amb}_{CE}(\mathcal{Y}_n) \geq 0, \forall n \in \overline{1, N}$

2. $\text{amb}_{CE}(\mathcal{Y}_n) = 0, \Leftrightarrow y_{in} = y_{jn} \forall i, j \in \overline{1, M}$

In conclusion, we can see that the amb_{CE} ambiguity has the desirable property of always being non-negative in contrast to amb_{01} which, as mentioned in [105] can be negative if the ensemble prediction is incorrect. The proofs for Theorem 3 can be found in the Appendix, section A.2.1.

4.2.2 Ambiguity measure for hinge loss

Following the same route, an ambiguity measure can be obtained appropriate for the hinge loss. The hinge-loss is defined as:

$$L_H(y_{in}, t_n) = \max(0, 1 - t_n y_{in}). \tag{4.4}$$

Here y_{in} is the i^{th} classifier score for the n^{th} pattern and t_n is the target, where it is convenient to label the targets as $\{\pm 1\}$. From straightforward substitution we obtain the following formula for the ambiguity measure derived from the hinge loss.

Definition 4.2.2. *The ambiguity measure obtained for the hinge loss is defined as:*

$$\text{amb}_{HL}(\mathcal{Y}_n) = \sum_{i=1}^{M} c_i \max(0, 1 - t_n y_{in}) - \max\left(0, \sum_{i=1}^{M} c_i (1 - t_n y_{in})\right). \tag{4.5}$$

Theorem 4.

The hinge loss diversity has the following properties:

1. $\text{amb}_{HL}(\mathcal{Y}_n) \geq 0 \, \forall n \in \{1 \dots N\}$

2. *If for the pattern* $\mathbf{x}_n$ *we have* $\forall \, i, j \in \{1 \dots M\}, y_{in} = y_{jn} \Rightarrow \text{amb}_{HL}(\mathcal{Y}_n) = 0$

3. *If for the pattern* $\mathbf{x}_n$ *we have* $\text{amb}_{HL}(\mathcal{Y}_n) = 0 \implies$ *that all the classifiers predict the same class.*

We provide an example of the last case, illustrating that $\text{amb}_{HL}(\mathcal{Y}_n) = 0$ does not imply that all the classifiers predict the same class. Consider one pattern x_1 belonging to class 1, so $t_1 = 1$ and 3 classifiers of equal weights, having the following scores: $y_{11} = -2$, $y_{21} = 0.5$, $y_{31} = 1$. Then, we have :

$$\text{amb}_{HL}(\mathcal{Y}_1) = \frac{1}{3} \sum_{i=1}^{3} \max(0, 1 - t_1 y_{i1}) - \max\left(0, \frac{1}{3} \sum_{i=1}^{3} (1 - t_1 y_{i1})\right).$$

The first term is:

$$\begin{aligned} \frac{1}{3} \sum_{i=1}^{3} \max(0, 1 - t_1 y_{i1}) &= \frac{1}{3} \max(0, 1 - 1 \cdot (-2)) + \frac{1}{3} \max(0, 1 - 1 \cdot 0.5) + \frac{1}{3} \max(0, 1 - 1 \cdot 1) \\ &= \frac{1}{3} \max(0, 3) + \frac{1}{3} \max(0, 0.5) + \frac{1}{3} \cdot 0 = \frac{3.5}{3}. \end{aligned}$$

The second term is:

$$\max\left(0, \frac{1}{3}\sum_{i=1}^{3}(1 - t_1 y_{i1})\right) = \max\left(0, \frac{1}{3}(3 + 0.5 + 0)\right) = \max\left(0, \frac{3.5}{3}\right) = \frac{3.5}{3}.$$

As a result, we have $\text{amb}_{HL}(\mathcal{Y}_1) = 0$ even though the first classifier predicts the negative class and the other two predict the positive class.
In this section we have analysed the properties of the three ambiguity measures presented. We can conclude that out of all of them, the amb_{CE} measure satisfies all the desired properties of a diversity measure (being always positive and 0 if and only if all the classifiers predict the same class). Not all of these properties are satisfied by the other two ambiguity measures. While the amb_{01} ambiguity is 0 if and only if all the classifiers predict the same, it can be negative when the ensemble classifies incorrectly a pattern [105]. On the other hand the amb_{HL} ambiguity is always positive, but if the ambiguity is 0 it doesn't necessarily imply that all the classifiers predict the same. The proofs for Theorem 4 are presented in the Appendix, section A.2.2.

4.3 Correlation between ambiguity and generalisation error

Previous studies have investigated the relationship between diversity (measured in a variety of ways) and the error/loss, see Chapter 3, [23, 1]. However, although a negative correlation between generalisation error and ambiguity has been reported [1], it is clear that this cannot be true for all ambiguities. We therefore empirically investigate the relationship between the ambiguity measured on a *training* data set and the error/loss on a test data set (approximating the generalisation error).

Bagging was used in order to control the diversity by sampling different independent samples to train the classifiers in the ensemble. We use 30 sampling rates in the range $[0.01, 1]$. For each sampling rate an ensemble of decision trees, forming a random forest [10], was trained on the sampled patterns. From the 2000 available observations, 1000 were drawn at random and used for training, while the remaining 1000 were used for evaluating the generalisation error; the roles of the training and testing sets were then swapped and the corresponding ambiguities and losses calculated. This process was repeated 50 times and the ambiguities and errors averaged over the resulting 100 instances.

We used the GMM5 dataset [106] which comprises two-dimensional features generated by a Gaussian mixture model with 5 components (an extension of the 4-component model of [107]) allowing a large quantity of data to be synthesised and the Bayes error rate to be calculated exactly. The Bayes error is the lowest possible error that can be achieved for a specific problem and it can be calculated by using the following formula:

$$\mathcal{L}_{Bayes} = 1 - \sum_{i=1}^{C}\int_{C_i} P(t_i)P(x|t_i) \text{ [108]}$$

where C is the number of classes, $P(t_i)$ is the a priori class probability of class i, $P(x|t_i)$ is the class likelihood and C_i is the region where class i has the highest posterior probability.

Figure 4.1 show the variation of the generalisation error with the diversity of the ensemble

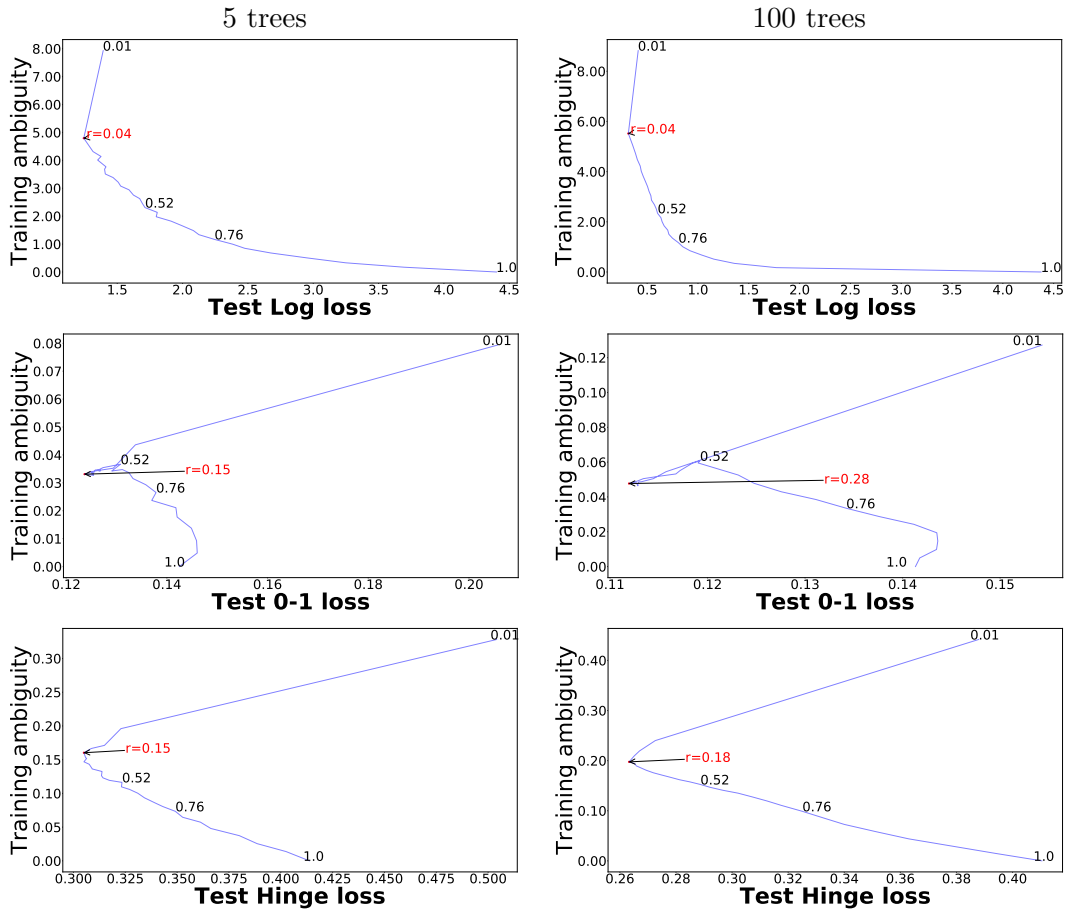

Figure 4.1. Curves of the three types of ambiguities versus the corresponding losses that were derived from these. The test error versus the training ambiguity was plotted for different sampling rates for ensembles formed of 5 trees (left column) and 100 tress (right column) for the GMM5 dataset. The first row shows the behaviour of the test cross entropy versus the training cross entropy ambiguity, in the second row the test 0-1 loss versus its corresponding training ambiguity is plotted, respectively the behaviour of the hinge loss is presented in the third row of panels. The optimal sampling rate is indicated in red.

measured on the training dataset for each of the ambiguity measures discussed. The first column of panels in Figure 4.1 corresponds to a small ensemble of $M = 5$ trees the second column shows the variation for a large ensemble of $M = 100$ trees. Although there is considerable variation between the curves for the different ambiguity measures, they all display common characteristics. At high sampling rates the ambiguity and test error are negatively correlated, as also reported by [1]. In this regime, as the sampling rate increases member classifiers are trained on increasingly similar views of the data and therefore diversity decreases. Since the average error per classifier is approximately constant (because adding more data does not appreciably increase their accuracy), equation (4.1) shows that the ensemble error increases.

Decreasing the sampling rate means that the members of the ensemble are trained on different views of the data, leading to increasing diversity/ambiguity and therefore a smaller ensemble error, c.f. (4.1). However, as the sampling rate is reduced to even lower levels, each component classifier is trained on a very small number of patterns and therefore starts to become inaccurate. In (4.1) the average error increases more rapidly than the diversity and the result is that the ensemble error begins to rise again. Unfortunately, determin-

ing the sampling rate that yields the optimum generalisation error is not straightforward or susceptible to *a priori* analysis. In section 4.4 we therefore describe an evolutionary algorithm to determine this rate.

The same pattern is apparent for both small ($M = 5$, Figure 4.1 left column) and large ($M = 100$, Figure 4.1 right column) ensembles, although the larger ensemble achieves a lower generalisation error. This generalisation error is very close to the Bayes error (0.11 misclassification rate) for this data set. It might be expected that the optimum sampling rate would be at least $1/M$, so that each classifier in the ensemble is trained on N/M examples and each example is used on average in the training of at least one classifier. However, as the panels in Figure 4.1 show, the optimum sampling rate is well below $1/M$, meaning that some of the data is not used at all by the ensemble. This indicates the significant role played by diversity: to achieve best generalisation performance it is better to ensure diversity by exposing classifiers to very different views of the data than to better train them individually by providing more data.

Although only shown here for the GMM5 dataset we emphasise that very similar relationships between ambiguity and generalisation error were observed on a number of additional datasets (Table 4.1, see Appendix, section A.3).

In our next set of experiments we investigated the variation of generalisation error with the number of classifiers forming the ensemble. This was achieved by generating ensembles with 2 to 100 members and training them, as before, with samples at a given rate. This was repeated 20 times for each ensemble size and sampling rate. The average (test) cross entropy error plotted against size of ensemble and sampling rate is shown in the panel of Figure 4.2 for the Sonar data set (Table 4.1, [109, 110]). This figure plainly shows the benefit of a large ensemble: the optimum generalisation error with a large ensemble is obtained over a wide range of sampling rates. The average training cross entropy ambiguity is plotted against size of ensemble and sampling rate in the right panel of Figure 4.2. These two figures together show the relationship between generalisation error and training ambiguity; high ambiguities yield lower test errors, provided the sampling rate is not too small. However, these two plots show the difficulty of predicting from the training ambiguity the optimal rate that will yield the lowest generalisation error.

4.4 An Evolutionary Algorithm to Optimise Ambiguity

As we have shown, provided that the sampling rate is not too low, the generalisation error is reduced for ensembles with high diversity. We therefore use an evolutionary algorithm to maximise the ambiguity of an ensemble of classifiers by selecting the patterns; i.e the particular training examples on which the constituent optimisers are trained. Pseudocode for the algorithm is presented in Algorithm 2.

We use ensembles of M classifiers, each of which is trained on a fraction ρ of the N available training patterns. In common with standard bagging ensembles, each of the classifiers is trained on all the available features. The patterns on which each classifier is trained are represented by a string of N 0s and 1s, where a 1 indicates that the corresponding pattern is used to train the classifier, so that there are exactly $[\rho N]$ 1s in each string and $[\cdot]$ indicates rounding to the nearest integer. The strings representing the training

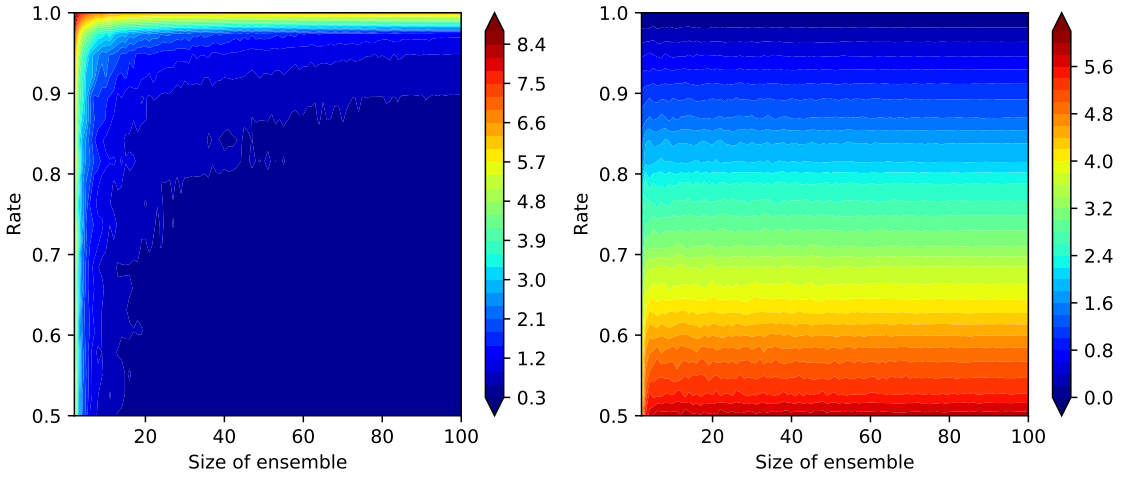

Figure 4.2. The figure in the left of the panel represents the cross entropy generalisation error versus the size of the ensemble and the sampling rate. On the right hand side the training ambiguity derived from the cross entropy versus the size of the ensemble and the sampling rate is plotted. The plots were obtained for the Sonar data.

patterns are initialised using stratified random sampling without replacement so that the class ratios are preserved.

A single ensemble is evolved through mutation. Between 1 and M strings are mutated in one of two ways, chosen with equal probability (line 3 in Algorithm 2). Then a type of mutation is chosen with equal probability (line 5, $U(0,1)$ generates a random number between 0 and 1):

1. A proportion up to $\frac{N}{2}$ of 1s and 0s are flipped at random. This is performed in a stratified manner to preserve the class ratio and so as to maintain the sampling rate as ρ (line 6).

2. The current string is discarded and replaced with a new string chosen in the same way as the initialisation, preserving the class ratio and the sampling rate (line 8).

Following mutation the N_{pop} members with the largest ambiguity are retained to proceed into the next generation. In case of equality, the forest with the lower error will be preferred (line 9).

4.5 Experiments

We ran our algorithm on six standard classification datasets from the UCI Machine Learning Repository: Australian, Cancer, Liver, Heart, Sonar, Ionosphere [76] and an additional synthetic dataset GMM5 [107, 111]. Table 4.1 summarises the dataset characteristics.

Since the results in Fig. 4.2 show that for large ensembles, the generalisation error is small for sufficiently low sampling rates, we concentrate on small ensembles. We used ensembles of $M = 5$ trees, which were implemented by using the `DecisionTreeClassifier` function from the sklearn library [112] in Python and the ambiguity measure $\text{amb}_{CE}(\cdot)$ derived from the log loss (4.2).

Algorithm 2 Evolutionary algorithm for evolving a diverse ensemble

Input: $X = \{\mathbf{x}_n\}_{n=1}^N$ ▷ training data
Input: $t = \{t_n\}_{n=1}^N$ ▷ targets
Input: M ▷ number of trees
Input: g ▷ number of generations
Input: ρ ▷ sampling rate
Output: $\mathcal{T}$ ▷ evolved forest

$\mathcal{T} \leftarrow initialize(X, t, M)$ ▷ generate a random ensemble/forest
for $i = 1 \rightarrow g$ **do**
  $m \leftarrow random(1, M)$ ▷ choose m trees to be changed
  $indices \leftarrow indicesToChange(M, m)$ ▷ choose the indices of the m trees
  if $U(0, 1) < 0.5$ **then**
    $\mathcal{T}' \leftarrow mutate(\mathcal{T}, indices, \rho)$ ▷ mutation type 1
  else
    $\mathcal{T}' \leftarrow genNewTrees(\mathcal{T}, indices, \rho)$ ▷ mutation type 2
  if $(\text{amb}_{CE}(\mathcal{T}') > \text{amb}_{CE}(\mathcal{T}))$ **or**
    $(\text{amb}_{CE}(\mathcal{T}') = \text{amb}_{CE}(\mathcal{T})$ **and** $L_{log}(\mathcal{T}', t) < L_{log}(\mathcal{T}, t))$ **then**
    $\mathcal{T} \leftarrow \mathcal{T}'$
return $\mathcal{T}$

Table 4.1. Dataset characteristics

Datasets	Patterns	Features
GMM5	1000	2
Australian	690	14
Cancer	569	10
Liver	345	6
Heart	270	75
Sonar	208	60
Ionosphere	351	34

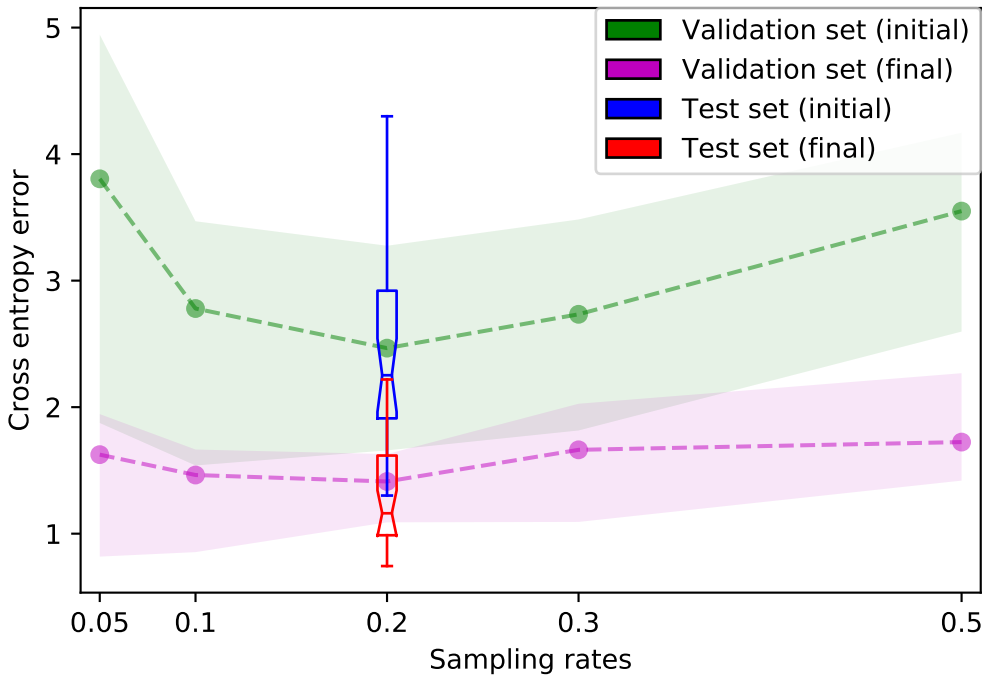

Figure 4.3. Example results on the Liver dataset, using an evolutionary algorithm to optimise the cross-entropy ambiguity.

Evolutionary algorithm

Data was partitioned into the following stratified parts as follows: one half for the test data, a quarter of the data for the training and the remaining quarter for the validation data. The evolutionary algorithm was run using the training data and the resulting ensemble evaluated on the validation data. The forest with the sampling rate that yields the lowest validation error was evaluated on the test data to assess the algorithm's performance.

Figure 4.3 shows example results obtained on the Liver dataset. The optimisation was repeated 30 times for each sampling rate and the figure shows the mean and interquartile range of the cross entropy generalisation error. We have run the experiment for 30 times, 30 being a standard number used in statistical testing, in order for calculating a meaningful statistics. Having a smaller number of repeats might not provide meaningful results, whereas a higher number would provide similar results.

We compared the ensemble's validation error for the initial generation with the optimised ensemble's validation error, for the following sampling rates: 0.05, 0.1, 0.2, 0.3, 0.5.
The green dashed line in Figure 4.3 corresponds to the mean of the 30 runs for the initial population, whereas the purple dashed line represents the mean for the final population. Shading indicates the interquartile range. The blue box plot corresponds to the test error for the initial populations, whereas the red box plots represents the test error for the corresponding final populations. These box plots were generated just for the sampling rate that yielded the lowest average validation error.

We also performed non-parametric statistical tests to assess the significance of the results. We used the Wilcoxon signed rank two-tailed test, $p = 0.05$. In Table 4.2 the mean test error of the initial ensemble for the sampling rate that yielded the smallest validation error

Table 4.2. Results on datasets, mean over 30 runs given (lower and upper quartile in brackets). Bold mean value indicates significant difference (Wilcoxon signed rank two-tailed test, $p = 0.05$).

Datasets	**Initial cross entropy**	**Final cross entropy**
GMM5	1.32 (0.82, 1.75)	**0.73** (0.6003, 0.852)
Australian	1.35 (1.11, 1.52)	1.26 (0.92, 1.39)
Cancer	0.63 (0.35, 0.74)	**0.421** (0.32, 0.45)
Liver	2.41 (1.91, 2.92)	**1.37** (0.98, 1.61)
Heart	1.76 (1.195, 2.17)	**1.32** (0.94, 1.55)
Sonar	2.19 (1.52, 2.953)	**1.21** (0.91, 1.52)
Ionosphere	1.32 (0.97, 1.57)	**1.01** (0.83, 1.195)

is shown, along with the mean test error of the corresponding final evolved ensemble. The values in the parenthesis correspond to the 25th quartile and 75th quartiles. These results show that, in general, the EA performs significantly better than the random sampling from the initial population, and never worse. The ambiguity optimised ensembles have lower test errors on average than the initial ensemble across all test problems.

What patterns are selected?

In our evolutionary algorithm we evolved the patterns that were selected in each tree. As such it would be interesting to see which patterns were actually chosen, and if they have any particular properties. In order to gain an understanding of which are the selected patterns, we analyse a two dimensional case.

A preliminary experiment was to plot the evolved patterns from the final generations of the evolutionary algorithm with their frequency of appearance. We performed this experiment just for the GMM5 dataset, because the distribution of these data are known and we have access to the posterior probabilities. We characterised the patterns according to their distance from the decision boundary. In order to determine how far a pattern is from the decision boundary, we calculated the maximum posterior probability of the pattern belonging to each of the two classes. The patterns belonging to the decision boundary have a minimum maximum posterior probability of 0.5. We averaged the number of appearances for the patterns from the final generation throughout the 30 runs. On the x-axis of Figure 4.4 the maximum of the posterior probability for both classes for each pattern is represented in 20 bins. On the y-axis, the proportion of occurrences is plotted. The green horizontal lines represent the medians of the number of occurrences for the patterns belonging to each of the 20 bins. This plot was obtained from the results of the evolutionary algorithm for the $\rho = 0.1$ sampling rate. Our results suggest that for this particular problem there is no preference for choosing some patterns during the optimisation, and that there is no correlation between whether a pattern is selected and its proximity to the decision boundary. This is contrary to what might be expected *a priori* — that is that points closer to the class boundary might be preferred as they give more information for bracketing the boundary. The results also reveal a high density of points on the right-hand side of the figure, which can be explained by the fact that these patterns are well classified. The errors obtained by the evolutionary algorithm are close to the Bayes error; therefore, these patterns would have high posterior probabilities.

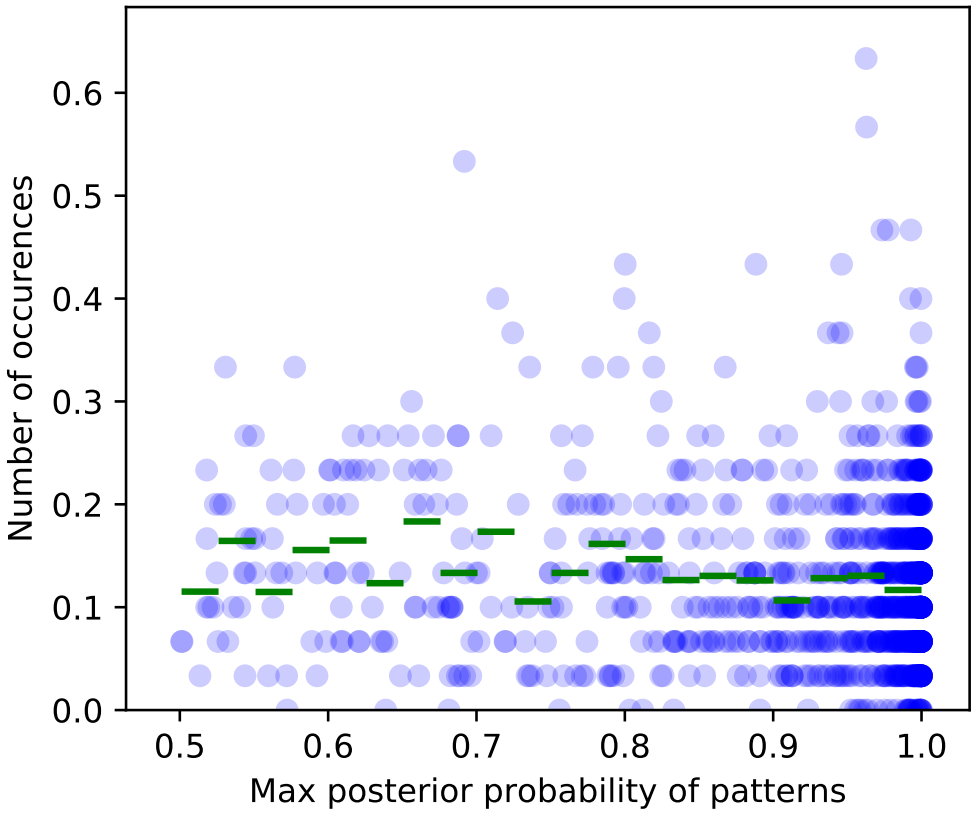

Figure 4.4. Frequency of patterns selected by the evolutionary algorithm at the final generation for the GMM5 dataset, for the 0.1 sampling rate. On the x-axis is the maximum posterior probability of a pattern belonging to each of the two classes. The y-axis represents the average proportion each pattern was selected over the 30 runs of the evolutionary algorithm. The values from the x-axis have been divided into 20 equally-sized bins. The green lines represent the medians of the number of occurrences of the patterns belonging to each bin.

Weight optimisation

Our previous results showed the positive effect that maximising ambiguity has on the test error. In our experiments the weights c_i of the classifiers were equal, but another way of maximising ambiguity would be by optimising the weights. We tried to optimise the ambiguity obtained from the cross entropy, amb_{CE}, from equation 4.2. Since amb_{CE} is not a quadratic function, Quasi-Newton methods are used, which are iterative methods for solving unconstrained nonlinear optimisation problems [113]. These methods use the Jacobian of the function to approximate the Hessian of the function, which is used to determine the step for each iteration.

In our experiments we defined the weights via a softmax function, so each weight c_i was of the form $c_i = \frac{e^{\theta_i}}{\sum_{j=1}^{M} e^{\theta_j}}$, where M is the number of classifiers and θ_i, $i \in 1 \dots M$ were chosen from a normal distribution. This definition of the weights was necessary in order to ensure that they are all positive and their sum is equal to 1.

The Jacobian of the amb_{CE} function is defined as:

$$J_{\text{amb}_{CE}} = \left[\frac{\partial \, \text{amb}_{CE}}{\partial \theta_1} \cdots \frac{\partial \, \text{amb}_{CE}}{\partial \theta_M} \right]$$

Let

$$S = \sum_{j=1}^{M} e^{\theta_j}$$

$$S_1 = \sum_{j=1}^{M} y_{jn} \frac{e^{\theta_j}}{S}$$

$$S_2 = \sum_{j=1}^{M} (1 - y_{jn}) \frac{e^{\theta_j}}{S}$$

$$\frac{e^{\theta_i + \theta_j}}{S^2} = \psi_{i,j}$$

$$\frac{e^{\theta_i}}{S} = \phi_i$$

Then

$$\begin{aligned} \frac{\partial \operatorname{amb}_{CE}}{\partial \theta_i} = \frac{1}{N} \sum_{n=1}^{N} t_n \left(y_{in} \frac{\phi_i}{S_1} - \phi_i - \ln(y_{in})(\phi_i - \phi_i^2) + \sum_{i \neq j} \ln(y_{jn}) \psi_{ij} \right) \\ +(1 - t_n) \left((1 - y_{in}) \frac{\phi_i}{S_2} - \phi_i - \ln(1 - y_{in})(\phi_i - \phi_i^2) + \sum_{i \neq j} \ln(1 - y_{jn}) \psi_{ij} \right) \end{aligned} \tag{4.6}$$

We used the minimize library from `SciPy.optimize` from Python and minimized the negative of the ambiguity.

We split the data into two stratified halves, one used for testing and the other half was divided as well into two stratified halves, one for training and the other for validation data . We initialised random values from a normal distribution (with zero mean and unit variance) for the values θ_i, $i \in 1 \ldots M$. We generated random forests of $M = 5$ decision trees and used sampling rates (representing the proportion of data used for each tree) in the range [0.05, 0.1, 0.2, 0.3, 0.4, 0.5]. We have fitted the ensembles on the training data and optimised the weights on the validation data. We repeated this experiment 20 times and computed the box plots for the training amb_{CE} and test error, see Figure 4.5. The top left plot compares the validation ambiguity of the ensembles with unoptimised weights (blue box plots) versus the validation ambiguity of the ensembles with optimised weights (red box plots) over 20 runs. We can see that the ambiguity was maximised in the case of optimised weights. The top right plot shows the test errors of the corresponding ensembles (blue with unoptimised weights and red with optimised weights). Unfortunately, the ensemble with optimised weights (with the highest ambiguity respectively), in general had a higher test error than the initial ensemble. We suspect that a possible reason could be that during the optimisation process, the ambiguity exceeded the level at which it was negatively correlated with the test error.

The bottom plots shows the curve of the average ensemble test error versus average validation ambiguity over 20 runs and for all sampling rates. The blue box plot shows the results when unoptimised weights were used, whereas red denotes the case when the op-

timised weights were used. This figure confirms the results obtained in the first two plots (generated for the 0.05 sampling rate), indicating that the use of optimised weights leads to increased ensemble ambiguity and test error compared to when unoptimised weights are used. The plots for the other sampling rates show a similar behaviour.

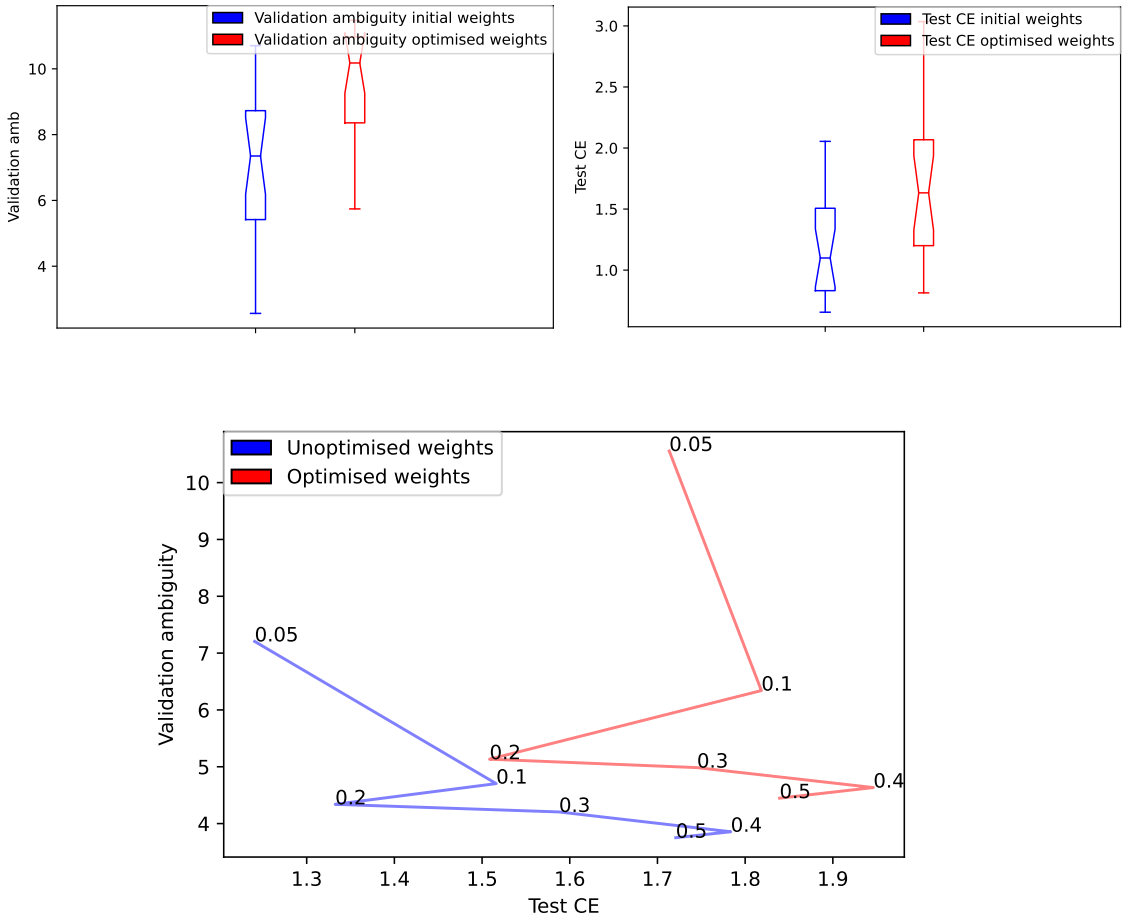

Figure 4.5. Optimised weights for the GMM5 dataset, obtained by using a sampling rate of 0.05. The first plot corresponds to the box plots of the validation ambiguity, the blue box plots for the validation ambiguity with initial weights whereas the red box plots with optimised weights. The second plot represents the box plots for the test error, the colors having the same meaning. The third plot represents the average behaviour of the validation ambiguity versus the test error for all sampling rates, blue denotes the case when unoptimised weights were used, whereas red when the optimised weights were used.

4.6 Conclusion

In this chapter we introduced two ambiguity measures using the bias-variance decomposition and the cross-entropy error or the hinge loss. Together with the ambiguity corresponding to the 0-1 loss, we established the properties of these new diversity measures. We evolved the training patterns of the classifiers in order to maximise the ambiguity obtained from the cross-entropy (amb_{CE}) and our results show that the evolved ensemble generally has a better generalisation error than the initial ensemble. Hence, our results support the influence that the diversity has on minimising generalisation error. Also the ambiguity measure obtained by using the cross-entropy error satisfies all the required properties of a diversity measure (being always positive and being zero if and only if the predictions of the classifiers are all the same). This property of being always positive is not present in the ambiguity obtained by using the 0-1 loss (see [1]), which can be negative if the

ensemble prediction is wrong [105].

Our results show that if random sampling is used to select patterns on which ensemble members are trained, we find that generalisation error is negatively correlated with diversity at high sampling rates; conversely generalisation error is positively correlated with diversity when the sampling rate is low and the diversity high.

Our experiments have shown that evolving the patterns that determine the trees of a forest, in order to maximise ambiguity, has a positive effect on the testing error of the ensemble.

Also, we found that there is no correlation between whether a pattern is selected and its proximity to the decision boundary (at least for the problem we considered where we had direct access to the posterior probabilities and therefore could determine the 'true' decision boundary precisely).

Our results were obtained when all the weights of classifiers were equal, as a result our last set of experiments analysed the effect of varying the weights has on the ensemble test error. We have optimised the weights by using BFGS, in order to maximise ambiguity and investigated the impact on generalisation error. Even though in the previous experiments maximising ambiguity improved the ensemble generalisation error, our results show that by optimising the weights the same conclusion does not stand. A possible explanation could be that the threshold up to which the ambiguity is negative correlated with the ensemble error, is exceeded by the optimisation. Also, by analysing the last plot in Figure 4.5 the relationship between sampling rates, test error and ambiguity is not as clear as the one obtained by maximising the ambiguity with equal weights, where there was a negative correlation when sampling rates were low and positive for higher sampling rates, as seen in Figure 4.1 .

In this chapter the main results were obtained by optimising a subset of patterns of fixed trees. Hence, our next set of experiments will focus on selecting a subset of trees from a set of trees in order to optimise ambiguity and analyse its effect on ensemble generalisation error.

Chapter 5

Optimising diversity by tree selection

5.1 Introduction

In the previous chapter the focus was on optimising the patterns on which the trees were built, trees which formed small ensembles. In some cases it has been shown that the more trees in an ensemble, the better its performance will be [81]. However the amount of time needed for training such an ensemble, increases linearly with the number of trees. In [81] the authors have shown that the ensemble size can be reduced substantially and still obtain a similar performance. In this chapter we investigate different tree selection methods in order to find the best performance for a given number of trees. In section 5.2 we present pruning algorithms that reduce ensemble size by discarding either the classifiers that would minimize or maximise ensemble diversity or error, either the classifiers with the highest or lowest error. Section 5.3 presents tree selection methods for a fixed number of trees. In section 5.4 an evolutionary algorithm is presented, which optimises different subsets of trees, in order to maximise diversity. This section is followed by a comparison of all the methods presented in this chapter with the results of the evolutionary algorithm that selected patterns, presented in Section 4.4.

The main contributions of this chapter are the following:

1. The definition of various pruning techniques, a number of which promote ambiguity or ensemble accuracy

2. The exploration of the direct maximisation of ensemble ambiguity, via an evolutionary optimisation algorithm for selecting trees, as a way to maximise generalisation performance

3. The comparison of the above approaches in order to determine the most effective ones in reducing generalisation error

5.2 Pruning methods

In this section we will present novel pruning techniques that take into account the amb_{CE} of an ensemble. We will use the following notations. Let t be the target vector and $Y_m = \{y_i\}_{i=1}^m$ be a collection of m classifiers, where $2 \leq m \leq M$, where M is the number of classifiers from the unpruned ensemble. We will denote the ambiguity of the ensemble by $\text{amb}_{CE}(Y_m)$. A new ensemble Y_{m-1} is formed by removing one of the classifiers. We define different schemes to select the classifier to be removed from Y_m:

1. Keep most ambiguous ensemble. At each iteration the tree that would yield the most ambiguous ensemble (evaluated on the training data) is removed. Formally, this approach could be defined as:

$$Y_{m-1} = \arg\max_{y \in Y_m} \text{amb}_{CE}(Y_m \setminus y).$$

2. Keep least ambiguous ensemble. At each iteration the tree that would yield the least ambiguous ensemble, evaluated on the training data, will be removed. Mathematically, this method can be expressed as:

$$Y_{m-1} = \arg\min_{y \in Y_m} \text{amb}_{CE}(Y_m \setminus y).$$

3. Random. At each iteration a random tree is removed:

$$Y_{m-1} = Y_m \setminus \mathit{RandomChoice}(Y_m).$$

4. Remove tree highest error. The tree with the highest training error is removed :

$$Y_{m-1} = Y_m \setminus \arg\max_{y \in Y_m}(L_{log}(y, t)).$$

5. Remove tree lowest error. Conversely, the tree with the smallest training error is removed :

$$Y_{m-1} = Y_m \setminus \arg\min_{y \in Y_m}(L_{log}(y, t)).$$

6. Keep ensemble highest error. The next approach discards the tree that would yield the ensemble with the highest training error:

$$Y_{m-1} = \arg\max_{y \in Y_m} L_{log}(Y_m \setminus y, t).$$

7. Keep ensemble lowest error. Conversely, the last approach discards the tree that would yield the smallest training error respectively :

$$Y_{m-1} = \arg\min_{y \in Y_m} L_{log}(Y_m \setminus y, t).$$

In our experiments we used $M = 100$ trees and at each iteration a tree is discarded, according to the above methods.

The function *getAmbEns* (see below) at each iteration receives an ensemble of size m (m decreases after every iteration) and discards each tree in turn to form subsets of $m-1$ trees and evaluates the ambiguity of all these m ensembles. It returns the list containing the ambiguities of all the m subsets. Then according to the pruning scheme chosen ("Keep most ambiguous ensemble" or "Keep least ambiguous ensemble"), the algorithm will discard the tree of the index that is either the arg max or arg min.

```
1: procedure getAmbsEns(Y_m)
2:     amb_list ← list()
3:     for y ∈ Y_m do
4:         Y' ← Y_m \ y
5:         amb ← amb_CE(Y')
6:         add(amb_list, amb)
7:     return amb_list
```

Similarly the function *getErrEns* (see below) calculates the training error of the m subensembles. At each iteration, these approaches make m comparisons, which makes the method expensive in terms of time.

```
1: procedure getErrsEns(Y_m,t)
2:     err_list ← list()
3:     for y ∈ Y_m do
4:         Y' ← Y_m \ y
5:         err ← L_log(Y', t)
6:         add(err_list, err)
7:     return err_list
```

The trees were obtained via bagging for sampling rates $\rho \in [0.05, 0.1, 0.2, 0.3, 0.4, 0.5]$. We evaluate the performance of the pruning over 50 runs and produce the medians and the interquartile ranges for each of the 7 approaches, the results are displayed in Figure 5.1. The top plot from the panel shows the variation of the test cross entropy with the ensemble size across all 7 methods. The figure in the middle corresponds to the training cross entropy, whereas the bottom figure displays the behaviour of the training amb_{CE}.

By analysing the plot of the test cross entropy from Figure 5.1, we can see that the best approaches are "Keep most ambiguous ensemble", "Remove tree lowest error" and "Keep ensemble lowest error". We suspect that the reason for this similar behaviour between the first two approaches, is that by removing the tree with the lowest cross entropy the average cross entropy of the remaining classifiers increases much faster than the ensemble cross entropy, leading to an increase of the ensemble ambiguity, as seen in Figure 5.2. Figure 5.2 also shows the average tree training/test cross entropy versus ensemble training/test cross entropy and ensemble training/test amb_{CE} for three approaches "Remove tree lowest error", "Remove tree highest error", and "Random". The first row of each set of figures represents the results for the training data, the second shows results for the test data. We can see from these plots that the average tree cross entropy is increasing for the

"Remove tree lowest error" method, decreasing for the "Remove tree highest error" and almost constant for the "Random" scheme. Also, in the "Random" case the ensemble cross entropy increases slightly faster (as trees are removed) than removing the lowest error trees.

Another explanation for why the "Remove tree lowest error" approach has such a good generalisation performance, could be that the lowest error trees are overfitted to the training data and so removing them from the ensemble leaves an ensemble of "less overfitted" trees, that are therefore better able to generalise. Conversely, we could argue that removing the tree with the highest error is actually removing the tree that is least overfitted to the training data.

The first plot from Figure 5.1 also reveals that amongst the worst approaches is "Keep least ambiguous ensemble". These results demonstrate empirically once more the influence that diversity has on reducing test error.

We have compared the "Keep most ambigous ensemble" approach with the algorithms presented in Section 2.8. Since some algorithms either start from from 2 or 3 classifiers, either from a fixed number M of classifiers and eliminate one classifier at a time, we have ran the algorithms on the same set of trees and have used the same training and test data. We have plotted the test error versus the number of trees for all the algorithms presented in Section 2.8 versus the "Keep most ambiguous ensemble" method. The results are presented in Figure 5.3. The "Keep most ambiguous ensemble" method is presented in dark blue, the "Complementary" and "Margin distance" from [84] in red and green respectively, the "FES" method is highlighted in turqoise, whereas the "Kappa pruning" and "Reduce error with backfitting" methods from [81] in yellow and black respectively. These plots were obtained for the Australian dataset, the results for the sampling rates 0.1 and 0.5 are presented in Figure 5.3. Even though these results were obtained for the Australian dataset, the behaviour seen is consistent for the other datasets.

A similar pattern has been observed throughout the plots. The "Complementary" method is the least successfull, followed by "Kappa statistics", "Backfitting" approach. The "FES" approach seems to be sucessfull for smaller sampling rates since at each generation it prefers the ensemble with the lowest training error, however for higher sampling rate its effect on the test error seems to decrease. This could be explained by overfitting. Overall the most successful methods seem to be "Margin distance" and "Keep most ambiguous ensemble". We have not included in our analysis the "ADP" method, because the algorithm at each iteration checks if the accuracy of the pruned ensemble is higher than the one of from the previous iteration. If the accuracy is not higher, then the algorithm stops and returns the ensemble from the previous iteration, which in most cases led to only one iteration.

In terms of computational complexity, the most time consuming is the "Backfitting" approach, with a complexity of $\theta(M^2Q)$, where Q is the number of iterations in order to determine which classifier to be selected per iteration. The other approaches have a computation complexity of $\theta(M^2)$.

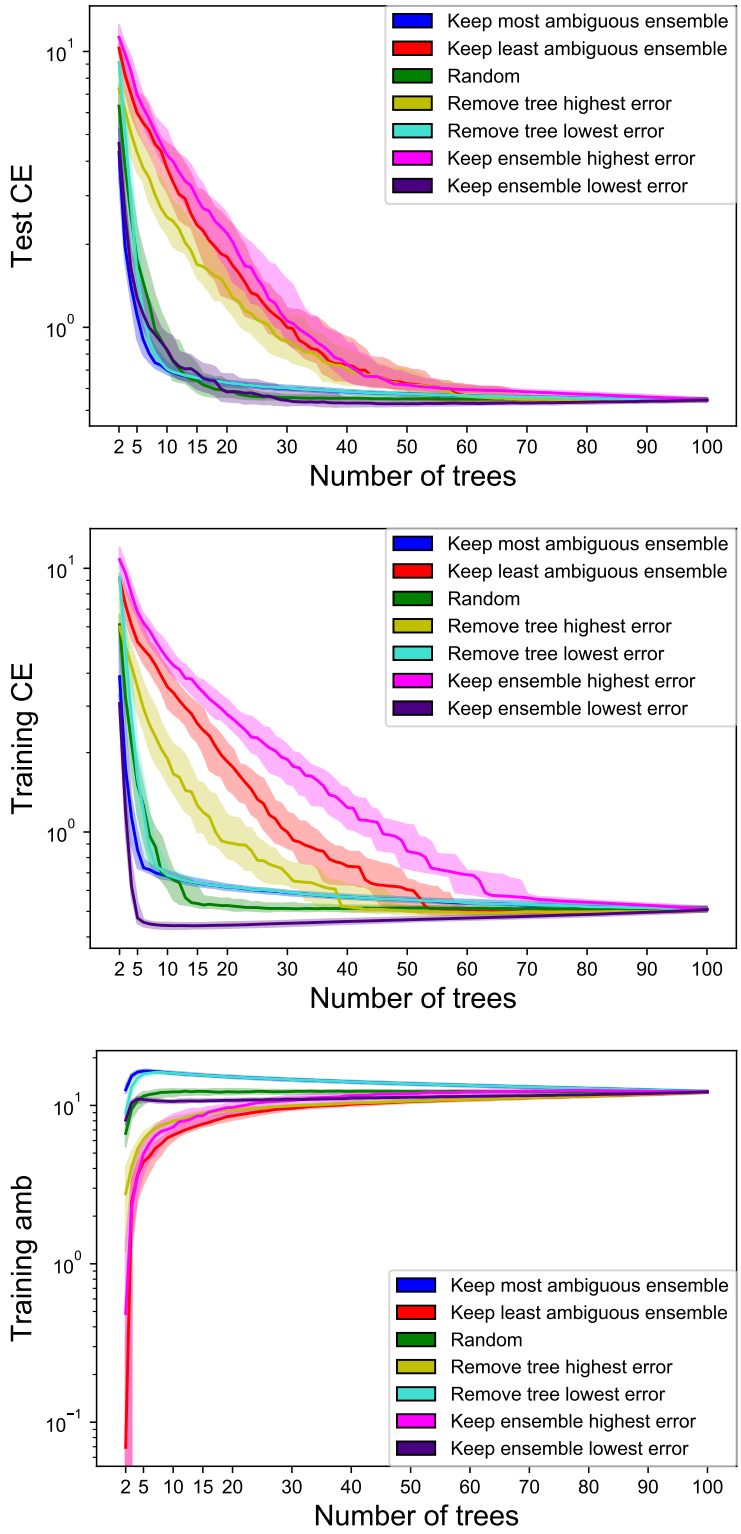

Figure 5.1. The top set of curves displays the variation of the test error with the ensemble size for each of the mentioned pruning approaches. The middle figure shows the variation of the training error, whereas the bottom plot shows for the training ambiguity. The results displayed are for the German dataset and for the 0.05 sampling rate

5.3 One in, One Out approach

The pruning method described in the previous subsection (5.2) — that starts from M trees and reduces the ensemble size down to two trees — has the disadvantage of being costly in terms of time. Let m_n be the size of the ensemble at iteration n. Then for the pruning

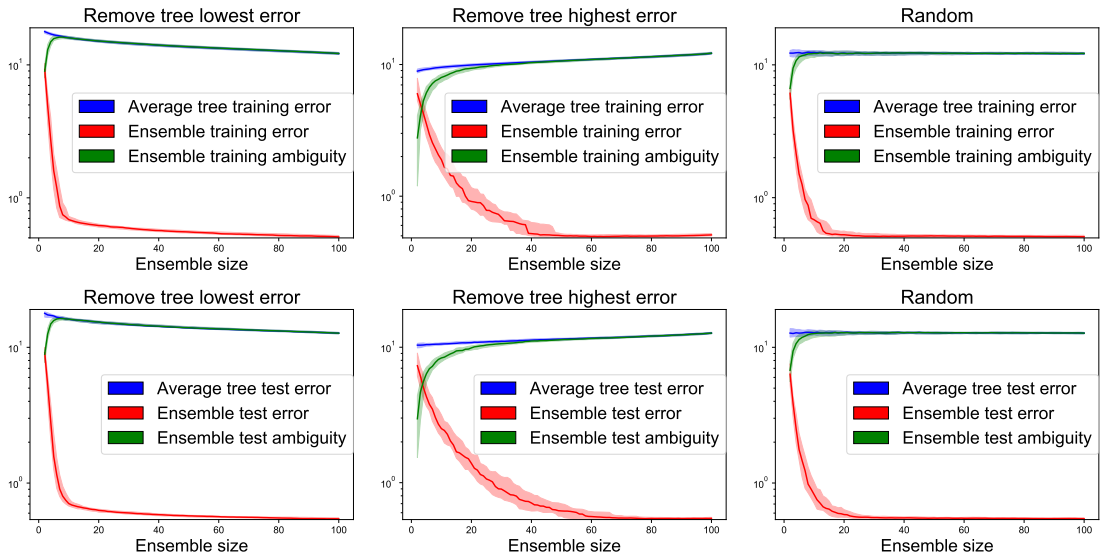

Figure 5.2. Curves of the average tree training/test error versus ensemble training/test error and ensemble training/test ambiguity. These plots were obtained for the German dataset and the 0.05 sampling rate.

schemes that keep at each iteration an ensemble that satisfies a certain criteria ("Keep most ambiguous ensemble","Keep least ambiguous ensemble", etc), at the nth iteration there will be m_n comparisons made. Obviously $m_1 = M$, $m_2 = M - 1$, as a result the program will finish after making $\frac{M(M+1)}{2} - 3$ comparisons, which increases quadratically with the initial size of the ensemble, M.

An alternative to this method would be to keep at each iteration a fixed number of trees, m, selected a priori, $m \ll M$. At each iteration a random tree would be added from a pool of remaining trees, and a tree will also be discarded according to the approaches presented in Section 5.2 (it is possible for the discarded tree to be the new entrant). Hence, we have named this method the "One In, One Out (OIOO)" approach. The total number of comparisons made by the OIOO method will be $(m + 1)(M - m)$. For $m = 5$ and $M = 100$, the total number of comparisons made is 570, whereas the pruning algorithm when reaching 5 trees, it would have made $\frac{M(M+1)}{2} - 4 - 3 - 2 - 1 = 5040$ comparisons, almost 9 times more calculations.

As before, we will consider $Y_n = \{y_i\}_{i=1}^{m}$ to be the collection of m classifiers, where $2 \leq m \leq M$ and n denotes the n^{th} iteration. We try to form a new ensemble Y' by considering ensembles formed by adding a new randomly generated classifier y' to the ensemble

$$Y' = Y_n \cup \{y'\}$$

and then removing one of the classifiers from Y' to form Y_{n+1}. One important aspect to remark, is that as opposed to the previous method, the index n from Y_n does not denote the size of the ensemble (because the size of the ensemble, m, is fixed), it denotes the current iteration.

We will consider in our experiments the same type of approaches as in the previous section, but with the following definitions:

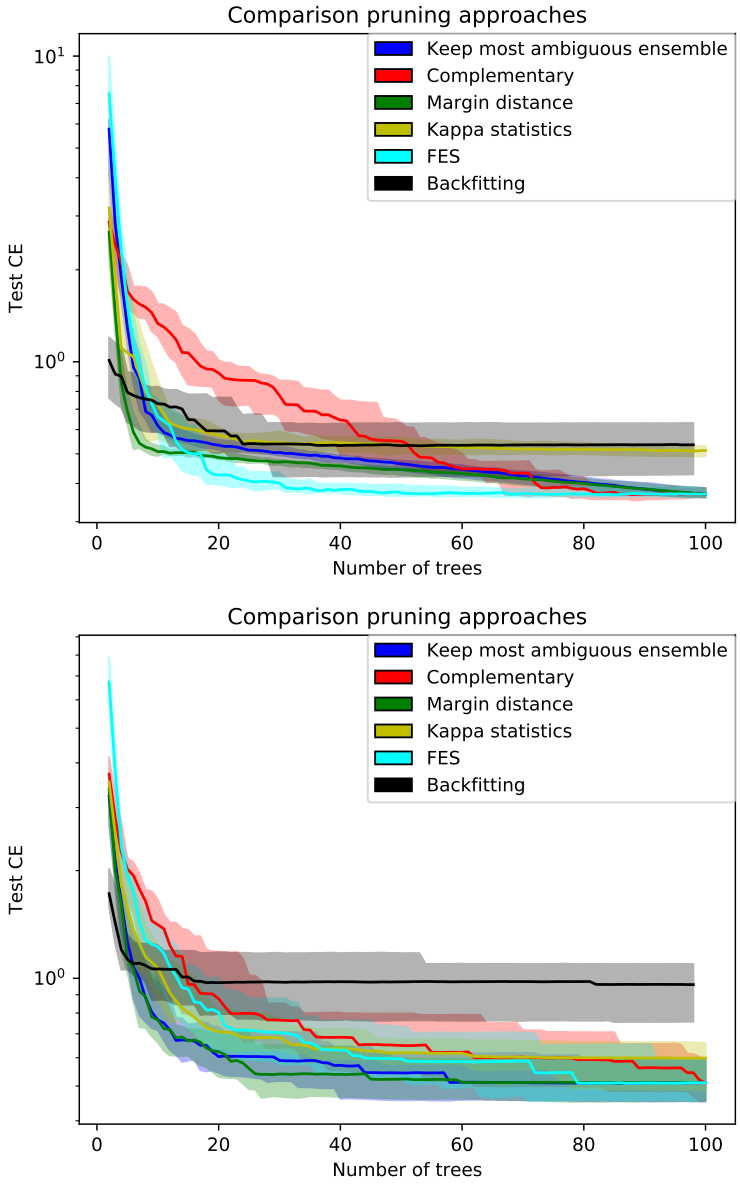

Figure 5.3. These set of curves displays the variation of the test error with the ensemble size for each of the mentioned pruning approaches. The top plots shows the rsults for the 0.1 sampling rate, whereas the bottom plot for the 0.5 sampling rate, both for the Australian dataset.

1. Keep the most ambiguous ensemble:

$$Y_{n+1} = \arg\max_{y \in Y'} \text{amb}_{CE}(Y' \setminus y).$$

2. Keep the least ambiguous ensemble :

$$Y_{n+1} = \arg\min_{y \in Y'} \ \text{amb}_{CE}(Y' \setminus y).$$

3. Remove a random classifier:

$$Y_{n+1} = Y' \setminus RandomChoice(Y').$$

4. Remove the classifier with the largest training error:

$$Y_{n+1} = Y' \setminus \arg\max_{y \in Y'}(L_{log}(y,t)).$$

5. Remove the classifier with the lowest training error:

$$Y_{n+1} = Y' \setminus \arg\min_{y \in Y'}(L_{log}(y,t)).$$

6. Keep ensemble with the highest training error:

$$Y_{n+1} = \arg\max_{y \in Y'} L_{log}(Y' \setminus y, t).$$

7. Keep ensemble with the lowest training error:

$$Y_{n+1} = \arg\min_{y \in Y'} L_{log}(Y' \setminus y, t).$$

We have tested this approach on forests formed of $m \in \{5, 10, 20, 50\}$ trees which were randomly chosen from the same pool of $M = 100$ trees as in Section 5.2. We ran the algorithm 50 times and produced box plots of the test data, see Figure 5.4. The same training and test data as in the previous experiments were used, but with different random startup ensembles for each run.

5.4 Evolving ensemble membership

Building on the ideas from the last two sections, we define an evolutionary algorithm which maximises the training ambiguity, by selecting a fixed number of trees, m, at each generation, see Algorithm 3. The trees selected are represented via a string of 0s and 1s, where 1 on the i^{th} position signifies that the i^{th} tree is selected and 0 that is not. At each generation we mutate the current string according to mutation rate μ. For each bit of the string a random number between 0 and 1 is generated, if the value is less than μ, then the current bit is mutated. If the total number of trees selected after the mutation is different than m, then random trees will be either added or removed in order to have

the same total number of trees, line 4 from Algorithm 3. The training ambiguity of the new formed ensemble is calculated, if it is higher than that of the previous ensemble the current ensemble will be kept and the old one discarded. In case of equality, the ensemble with the lower training error is kept.

Algorithm 3 Evolutionary algorithm for selecting trees by optimising diversity

Input: $X = \{\mathbf{x}_n\}_{n=1}^N$ ▷ training data
Input: $t = \{t_n\}_{n=1}^N$ ▷ targets
Input: m ▷ number of trees desired
Input: $tree\_list$ ▷ list of all possible trees
Input: g ▷ number of generations
Input: μ. ▷ mutation rate
Output: $\mathcal{T}$ ▷ evolved forest

1: $M \leftarrow len(tree\_list)$
2: $\mathcal{T} \leftarrow initialize(X, t, M)$ ▷ generate a random ensemble/forest
3: **for** $i = 1 \rightarrow g$ **do**
4: $\quad \mathcal{T}' \leftarrow mutate(\mathcal{T}, m, \mu)$
5: $\quad$ **if** $(\text{amb}_{CE}(\mathcal{T}') > \text{amb}_{CE}(\mathcal{T}))$ **or**
$\quad\quad (\text{amb}_{CE}(\mathcal{T}') = \text{amb}_{CE}(\mathcal{T})$ **and** $L_{log}(\mathcal{T}', t) < L_{log}(\mathcal{T}, t))$ **then**
6: $\quad\quad \mathcal{T} \leftarrow \mathcal{T}'$
7: **return** $\mathcal{T}$

To evaluate its performance the evolutionary algorithm was run for $g = 25000$ generations 50 times, and for the sampling rates $\{0.05, 0.1, 0.2, 0.3, 0.5\}$. The pool of trees from which the ensemble is composed at each generation is the same as the one used in Sections 5.2 and 5.3.

We compared the performance of the evolutionary algorithm with the approaches described in Sections 5.2 and 5.3 and with the evolutionary algorithm from Section 4.4. Box plots of the test errors for all the approaches over the 50 runs are shown in Figure 5.4. The results for 5 trees are displayed in the top left panel, for 10 trees in the top right, for 20 in the bottom left and 50 in the bottom right. We have grouped the box plots according to the tree selection method used. For example, in each group of a similar colour, the first box plot will always denote one of the tree selection methods that start with an ensemble of M trees and discard a tree at each iteration, according to the pruning approaches mentioned in section 5.2. The second box plot from the group (the darker colour) will always the be the corresponding method, but by using the OIOO approach from section 5.3 . We have labelled the box plots of similar approaches with the same letter, the ones representing the methods from Section 5.2 by 1 and the corresponding OIOO approach by 2. For example, the first group of box plots shown (coloured in navy) represent the test cross entropy values for the "Keep most ambiguous ensemble" method. The first box plot, A1 represents the values for the "Keep most ambiguous" ensemble method presented in Section 5.2. The second box plot A2, displays the results for the "Keep most ambiguous ensemble OIOO", presented in section 5.3. The meanings of these labels are presented in Table 5.1. The black vertical line separates the tree selection algorithms from the evolutionary algorithm that selects patterns (from Section 4.4). These boxplots were obtained for the German dataset and for $\rho = 0.05$.

We can see from these plots that in general the best approaches are "Keep most ambiguous ensemble","Keep most ambiguous ensemble OIOO","Remove tree lowest error","Remove tree lowest error OIOO" and the evolutionary algorithms, followed by "Keep ensemble lowest error","Keep ensemble lowest error OIOO","Random","Random OIOO". Another aspect which is visible from these plots is that as we increase the number of trees, the test errors given by the best approaches get closer to the unpruned ensemble.

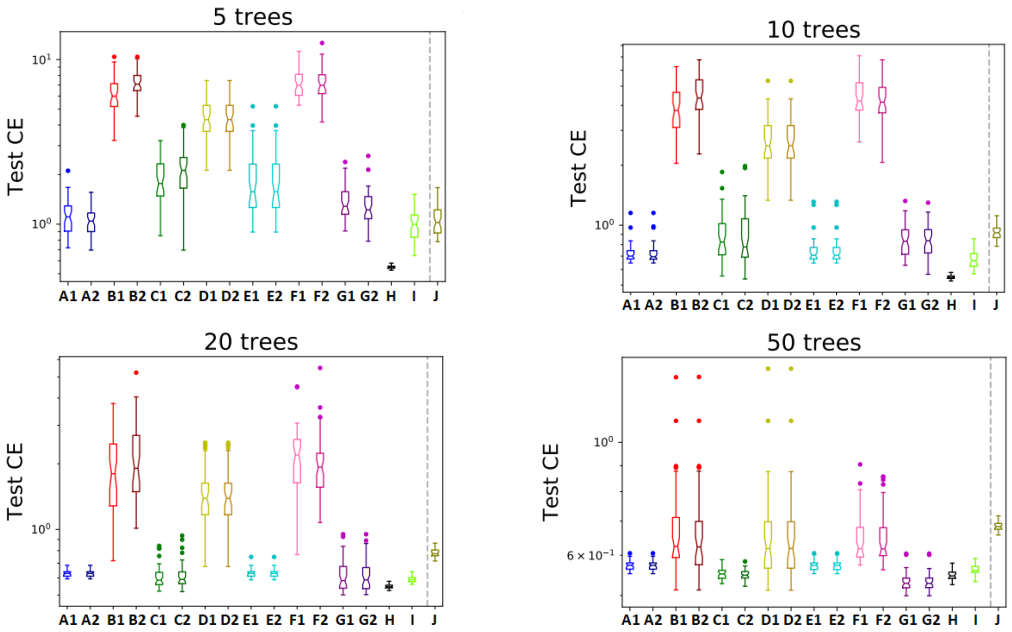

Figure 5.4. Box plots of the test CE for the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the German dataset for the 0.05 sampling rate. The first plot in the panel displays the results for 5 trees, the second plot for 10 trees, the third of 20 trees, whereas the last one of 50 trees. The symbols from the x-axis correspond to the tree selection methods and pattern selection approaches and their meaning is displayed in Table 5.1.

In order to correctly rank all these schemes, we performed statistical tests. We used the Wilcoxon signed rank two-tailed test [114] along with the Holm–Bonferroni correction method [115] for a p-value of 0.05. The results of the statistical tests are displayed in Table 5.2. This table contains two analyses. First it determines the best approach and all those approaches which are statistically indistinguishable from it out of the tree selection approaches. The best one is underlined in red, whereas the statistically similar ones are underlined in blue. The second analysis compares all approaches, i.e. the tree selection approaches with the pattern selection one (the evolutionary algorithm from Section 4.4). The best performing approach is shaded with a dark grey, whereas the statistically similar ones are highlighted in lighter grey.

We also present a visual comparison, by using critical difference diagrams, as in [116]. In these diagrams the best classifiers are ranked in an ascending order and they are connected with each other if they are statistically similar, see Figures 5.5–5.7 for these for the German dataset.

Even though these comparisons are just for the German dataset and for the $\rho \in \{0.05, 0.3, 0.5\}$ sampling rates, the ranking of the approaches is similar across all datasets (see the rest of

Table 5.1. Symbols denoting the tree selection methods from Figure 5.4

Approach	Symbol
Keep most ambiguous ensemble	A1
One In, One out	A2
Keep least ambiguous ensemble	B1
One In, One out	B2
Random	C1
One In, One out	C2
Remove tree highest error	D1
One In, One out	D2
Remove tree lowest error	E1
One In, One out	E2
Keep ensemble highest error	F1
One In, One out	F2
Keep ensemble lowest error	G1
One In, One out	G2
Unprunned ensemble	H
EA select trees	I
EA select patterns	J

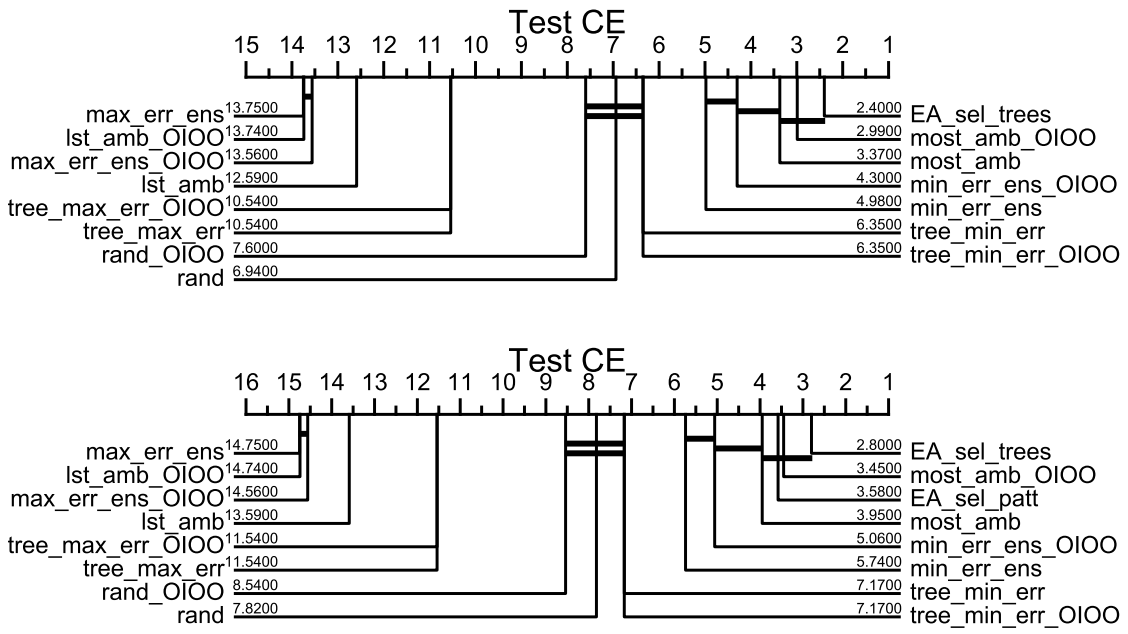

Figure 5.5. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the German dataset for the 0.05 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

Sampling rate	Number trees	+ most amb ensemble		+ least amb ensemble		Random		- tree max error		- tree min error		+ max error ensemble		+ min error ensemble		EA sel trees	EA sel patterns
0.05	5	1.11	1.04	5.99	7.09	1.76	2.12	4.32	4.32	1.57	1.57	6.98	6.97	1.28	1.22	1.00	1.02
0.1	5	1.29	1.18	3.73	4.36	1.88	1.81	3.12	3.12	1.49	1.49	4.29	4.50	1.67	1.60	1.29	1.13
0.2	5	1.48	1.46	2.98	3.13	1.88	1.89	2.63	2.63	1.60	1.60	2.64	2.86	1.91	1.87	1.51	1.39
0.3	5	1.59	1.55	2.76	2.93	1.96	1.96	2.58	2.58	1.57	1.57	2.32	2.58	2.14	2.19	1.65	1.36
0.4	5	1.67	1.61	2.68	2.71	2.16	2.12	2.67	2.63	1.76	1.76	2.14	2.26	2.33	2.41	1.86	1.50
0.5	5	1.90	1.91	2.71	2.70	2.17	2.25	2.63	2.63	1.92	1.92	2.44	2.55	2.44	2.50	1.98	1.75
0.05	10	0.70	0.69	3.77	4.36	0.82	0.78	2.51	2.51	0.71	0.71	4.21	4.15	0.83	0.84	0.66	0.92
0.1	10	0.66	0.66	1.99	2.10	0.87	0.85	1.82	1.82	0.66	0.66	1.90	1.89	0.99	0.99	0.66	0.75
0.2	10	0.71	0.71	1.61	1.63	0.88	0.86	1.45	1.45	0.72	0.72	0.97	1.27	1.05	1.05	0.72	0.66
0.3	10	0.78	0.78	1.43	1.42	0.99	0.97	1.44	1.46	0.76	0.76	1.01	1.12	1.22	1.17	0.82	0.68
0.4	10	0.86	0.86	1.43	1.41	1.06	1.08	1.44	1.44	0.86	0.86	1.03	1.09	1.28	1.28	0.91	0.71
0.5	10	1.00	1.00	1.49	1.53	1.17	1.24	1.48	1.48	1.00	1.00	1.15	1.22	1.31	1.31	1.05	0.87
0.05	20	0.63	0.63	1.80	1.91	0.59	0.59	1.39	1.39	0.63	0.63	2.19	1.93	0.58	0.59	0.59	0.77
0.1	20	0.60	0.60	1.02	1.07	0.59	0.58	0.95	0.95	0.60	0.60	0.78	0.80	0.66	0.67	0.57	0.67
0.2	20	0.56	0.56	0.91	0.92	0.61	0.60	0.88	0.88	0.57	0.57	0.62	0.62	0.75	0.73	0.57	0.60
0.3	20	0.59	0.59	0.86	0.89	0.65	0.66	0.85	0.85	0.58	0.59	0.65	0.65	0.75	0.73	0.59	0.57
0.4	20	0.62	0.62	0.92	0.92	0.72	0.78	0.90	0.90	0.64	0.64	0.68	0.69	0.83	0.83	0.65	0.57
0.5	20	0.69	0.69	0.97	0.97	0.83	0.80	0.96	0.96	0.68	0.68	0.75	0.74	0.86	0.85	0.72	0.61
0.05	50	0.57	0.57	0.62	0.62	0.55	0.55	0.62	0.62	0.57	0.57	0.62	0.62	0.53	0.53	0.56	0.68
0.1	50	0.55	0.55	0.57	0.57	0.53	0.53	0.57	0.57	0.55	0.55	0.56	0.56	0.53	0.53	0.54	0.61
0.2	50	0.53	0.53	0.57	0.57	0.52	0.52	0.57	0.57	0.53	0.53	0.53	0.53	0.53	0.53	0.53	0.56
0.3	50	0.52	0.52	0.58	0.58	0.53	0.53	0.58	0.58	0.52	0.52	0.53	0.53	0.53	0.53	0.53	0.54
0.4	50	0.54	0.54	0.60	0.60	0.58	0.57	0.60	0.60	0.55	0.55	0.56	0.56	0.59	0.59	0.55	0.53
0.5	50	0.60	0.60	0.64	0.64	0.61	0.61	0.64	0.64	0.60	0.60	0.60	0.60	0.62	0.62	0.59	0.53

Table 5.2. Statistical comparisons of the tree selection schemes and the evolutionary algorithm that selects patterns from Section 4.4. For each method the median of the test cross entropy over the 50 runs is displayed. The columns of the tree selection methods from Sections 5.2, 5.3 have two values, with the following meaning: the first one denotes the median of the test cross entropy for the pruning method, whereas the second the value indicates the median of the corresponding OIOO approach. Dark shading indicates the best approach across all methods, whereas the lighter grey the ones statistically indistinguishable. The blue underlining denotes the best approach across the tree selection methods, whereas the red underlining indicates the approaches statistically similar. The results shown are for the German dataset.

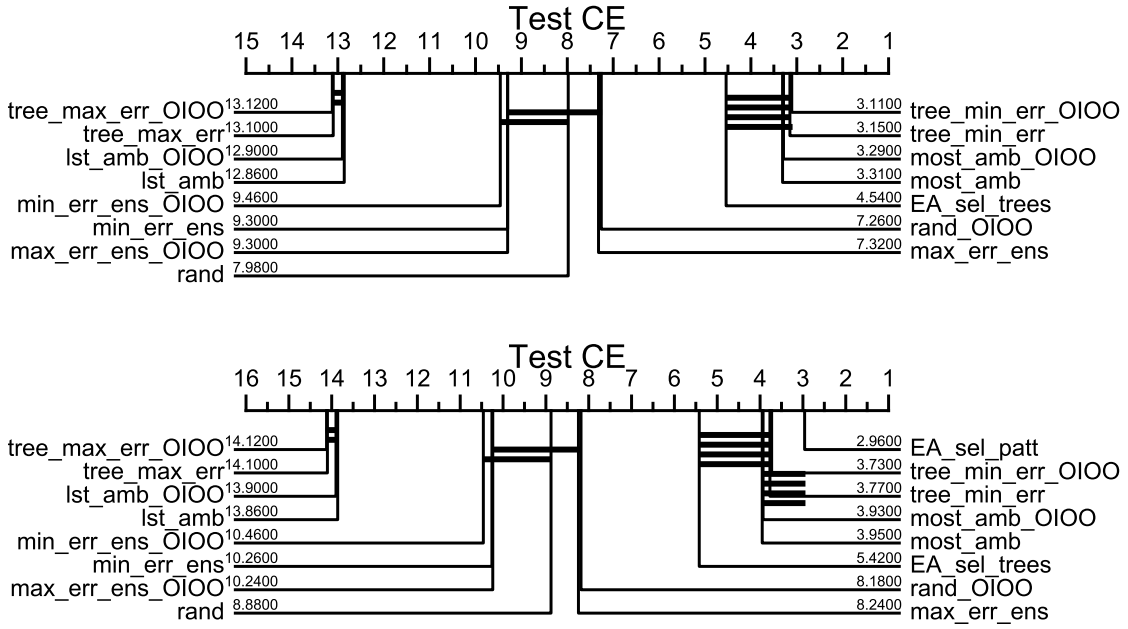

Figure 5.6. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the German dataset for the 0.3 sampling rate, for 10 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

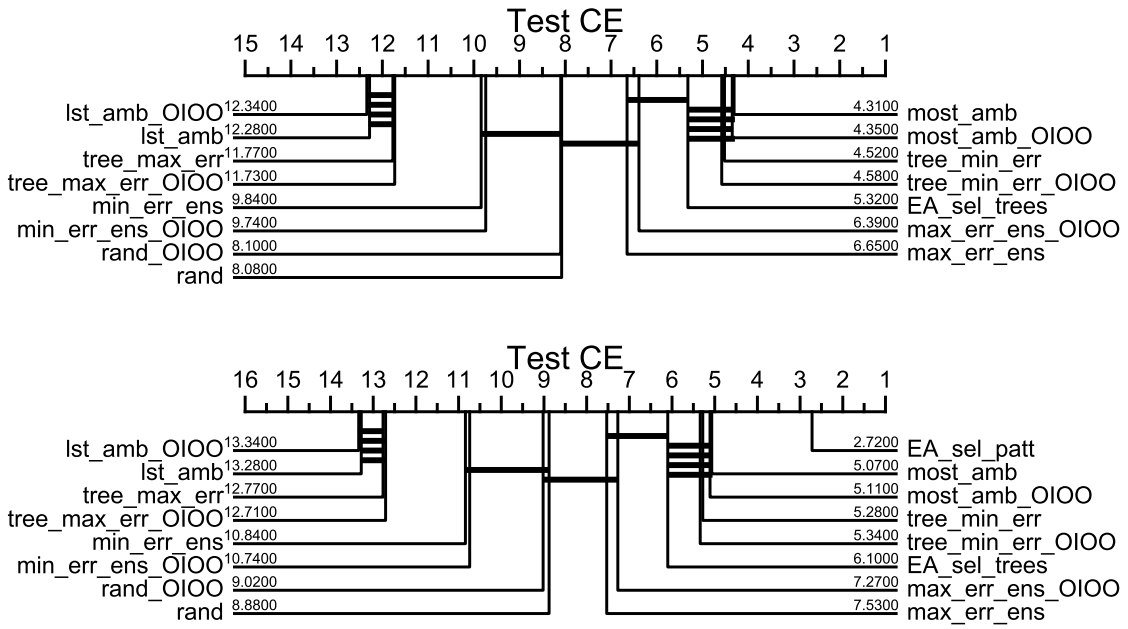

Figure 5.7. Comparisons of the tree selection approaches versus the pattern selection approach(the evolutionary algorithm from Section 4.4) on the German dataset for the 0.5 sampling rate, for 20 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

the figures in Appendix, Section A.4). In general the best approaches are the evolutionary algorithms (since they explore more the space of solutions). The evolutionary algorithms are statistically similar to the "Keep most ambiguous ensemble","Keep most ambiguous ensemble OIOO", "Remove tree lowest error","Remove tree lowest error OIOO" approaches. Since the evolutionary algorithms are expensive (they have a computational cost of $\mathcal{O}(\rho N)$, where $\rho = 25000$ is the number of generations and N the population size) and also the pruning approaches ($\mathcal{O}(M^2)$), arguably the best approaches would be "Keep most ambiguous ensemble OIOO" and "Remove tree lowest error OIOO", since their computational cost is $\mathcal{O}((m+1)(M-m))$ and is less than the costs of the other approaches considered ($\mathcal{O}((m+1)(M-m)) \approx \mathcal{O}(mM) < \mathcal{O}(M^2) < \mathcal{O}(\rho N)$, since $m << M$).

5.5 Conclusion

We have defined different methods of tree selection, some that favoured ambiguous/accurate ensembles and some that promoted less ambiguous/accurate ensembles. We compared the performance of these methods with the results of the evolutionary algorithm from Section 5.4 and from Chapter 4, Section 4.4. Our main findings are the following:

- In general the evolutionary algorithms yield the lowest generalisation errors, however they are the most computational expensive.
- The results of the "Keep most ambiguous ensemble","Keep most ambiguous ensemble OIOO", "Remove tree lowest error" and "Remove tree lowest error OIOO" are statistically similar to the results of the evolutionary algorithms.
- The results of the approaches that favour ambiguity or remove the trees with the lowest errors, are statistically similar to the results of the evolutionary algorithms.
- Our results demonstrate once more the usefulness of ambiguity in error reduction, since the approaches that favour ambiguity are amongst the most successful ones (being also similar in performance to the evolutionary algorithms and less expensive).

Chapter 6

Asymmetric impurities

6.1 Introduction

In previous chapters our experiments were based on forests of trees built with the same impurity function and diversity was based on the classifiers' predictions. In this chapter we explore a different path of ensuring diversity in a forest. We will define diverse forests by building trees that employ different impurities. Our experiments investigate the effect that building trees with different impurities has on the generalisation error. We will focus the investigation in the case of imbalanced datasets, where asymmetric impurities are of interest. In section 6.2 we will investigate the effect that different asymmetric impurities have on the generalisation error, in contrast to symmetric impurities and then continue by varying the asymmetric impurities used in building the trees from a forest.

The principal contributions of this chapter are as follows:

1. The empirical comparison of the effect of different asymmetric impurities versus symmetric impurities on the generalisation error

2. The empirical assessment of the effect of building trees from a forest with different asymmetric impurities on the generalisation error, as a way of injecting diversity

3. The numerical explanation of why symmetric and asymmetric impurities might yield similar splits, particularly within a specific family of impurity measures.

6.2 Experiments

In machine learning the focus is to generate models that would minimise a certain loss and would achieve a good generalisation performance. In practice, we are interested not just in optimising one loss/objective, but to optimise multiple losses/objectives at the same time, which can be conflicting. For example, one would like to have ensembles formed of accurate members, but also diverse. A common strategy is to compose linearly these losses/objectives, by using different weights and parameters and determine those that would minimise their combination. This can be achieved by training for each set of parameters or weights an individual model and store it in the memory, in order to analyse it later.

More formally, given a training distribution of pairs $x, y \sim P_{x,y}$ with $x \in X \subset R^{dX}$; $y \in Y \subset R^{dY}$ and a loss function $L(\cdot, \cdot) : Y \times Y \to R$, the aim is to learn a model $F : X \to Y$, with parameters θ, such that its predictions $\hat{y} = F(x; \theta)$ minimize the expected value of the loss $L(y, \hat{y})$ over the dataset.

This approach can severely affect the time of execution, as a result in [117] the authors suggest a method which would train a model just once, not on a certain parameter, but on distribution of parameters.

As opposed to the previous example when one loss function is considered $L(\cdot; \cdot)$ in this case a family of losses is considered $L(\cdot; \cdot; \lambda)$, parameterised by a vector $\lambda \in \Lambda \in R^{d_\lambda}$.

In general, such a family of losses are represented by a weighted sum of several loss terms:

$$L(\cdot, \cdot, \lambda) = \sum_i \lambda_i L^i(\cdot, \cdot, \lambda_i)$$

Instead of having a fixed λ, an optimisation problem is considered where the parameters λ are sampled from a distribution P_λ.

The parameter θ can be determined in this way:

$$\theta^* = \underset{\theta}{argmin} \sum_{i=1}^{n} L(y_i, F(x_i, \theta, \lambda_i), \lambda_i), \lambda_i \sim P_\lambda$$

In [117] the models considered were convolutional neural networks (CNN). Inspired by this approach, we investigated the extension of this method to decision trees.

As an equivalent to the family of losses, we defined a family of impurities, characterised by different parameters. To reiterate, an impurity function, $f : [0, 1] \to R$, is a concave function, which is continuous on [0,1] and for which the values at the end points coincide.

Inspired by the code from [44], we built decision trees by using different impurity functions.

6.2.1 Beta distribution

The first impurity used in our experiments was the beta function, defined as:

$$b(p) = p^{\alpha-1}(1-p)^{\beta-1}, \alpha, \beta > 0 \tag{6.1}$$

In order for the beta distribution to have a bell shape, or to be concave, the necessary and sufficient conditions are $1 < \alpha < 2$ and $1 < \beta < 2$. The beta distribution is symmetric when $\alpha = \beta$ and asymmetric when $\alpha \neq \beta$.

The beta distribution obtained for the parameters $\alpha = 1.2$, $\beta = 1.8$ is displayed in Figure 6.1.

Our initial experiment was to compare the behaviour of forests built with symmetric impurities versus asymmetric impurities, in the case of imbalanced classes. As mentioned earlier, when dealing with imbalanced datasets sometimes of more importance is considered

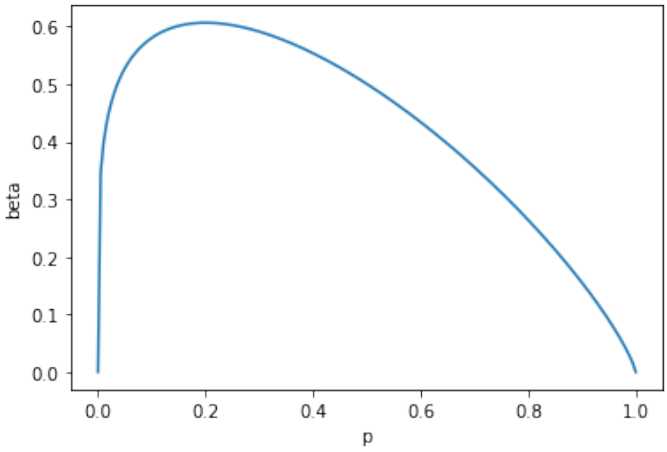

Figure 6.1. The values of the beta distribution are plotted on the y-axis, whereas the corresponding probabilities are shown on the x-axis.

the minority class and calculating the accuracy is not considered to be the best metric of performance. Instead, the true positive rate and false positive rate are of more interest, since they quantify much better the algorithm's capability of predicting the minority class. As a result we compared these values and the ROC curves of the forests built with symmetric impurities with the forests built with asymmetric impurities.

We built forests of 100 trees, each tree having the same values for the α and β values. We used 2-fold stratified cross validation, so that a stratified half of the data was used for the training and the remaining part was used for the test. We modified the data, so that there was an imbalance of 10% for the positive class. We ran the experiment 30 times and plotted the box plots of the forests' true positive rate, false positive rate, true negative rate and true positive rate and the mean ROC curve. The values from the box plots were obtained for the test data and are presented in Figure 6.2. These plots were obtained for parameters $\alpha = \beta = 1.5$ for the symmetric case and $\alpha = 1.9$ and $\beta = 1.1$ for the asymmetric case. We can see from these pictures that the behaviour is very similar in the symmetric and asymmetric case. As a result, we changed the parameters and produced the same box plots but for $\alpha = \beta = 1.6$ for the symmetric case and $\alpha = 1.2$ and $\beta = 1.8$, results shown in Figure 6.3.

Since these sets of parameters did not show the desired result, we tried to determine which ones would be the best parameters. We generated two evenly spaced array of values ranging between 1 and 2, one for the α values and one for β. For each of them, we generated forests of 100 trees in the symmetric case when $\alpha = \beta$.

In the asymmetric case, we investigated for a range of α and β. We used the same technique as before in dividing the data into two stratified halves, representing the training and test data. We ran the experiment 30 times and produced the box plots of the corresponding true positive, true negative, false positive and false negative rates which resulted from applying the test data, results shown in Figure 6.4. The blue box plots correspond to the symmetric case, whereas the red ones to the asymmetric case. The values from the red box plots were obtained for α values equal to the ones from the symmetric case and the β parameters were equal to the values shown on the x-axis. On the y-axis the true positive

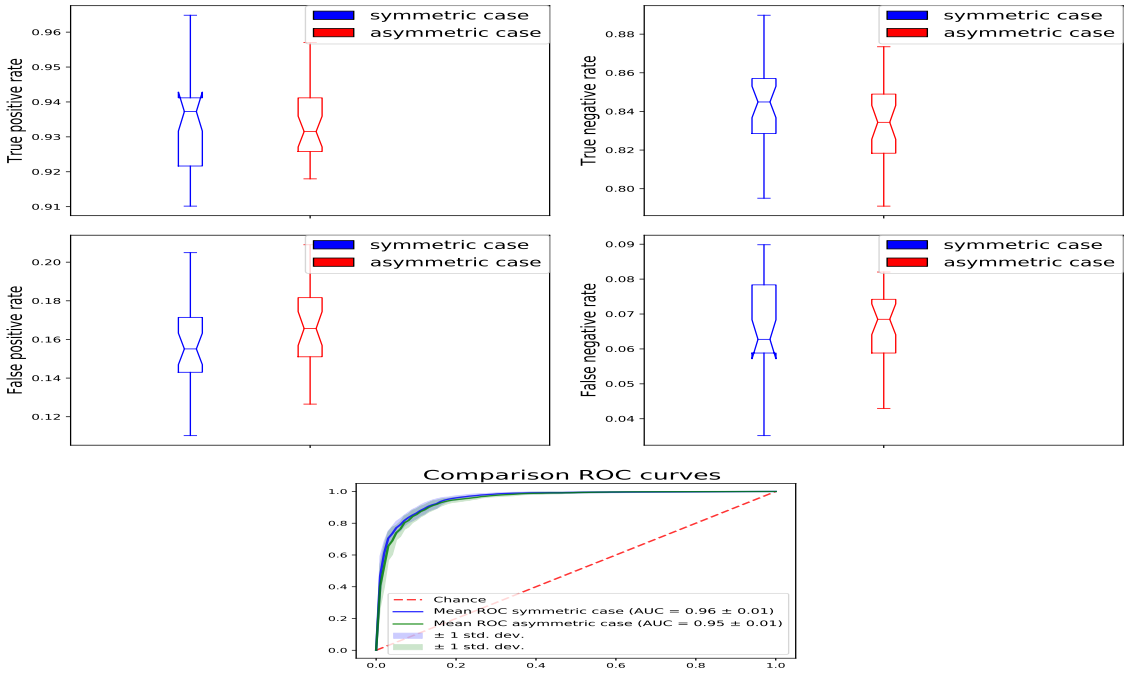

Figure 6.2. Results obtained on the Gmm5test dataset. The ROC plots for the symmetric and the asymmetric case are presented in blue, respectively green. The parameters used for the beta distribution were $\alpha = \beta = 1.5$ for the symmetric case and $\alpha = 1.9$ and $\beta = 1.1$ for the asymmetric case.

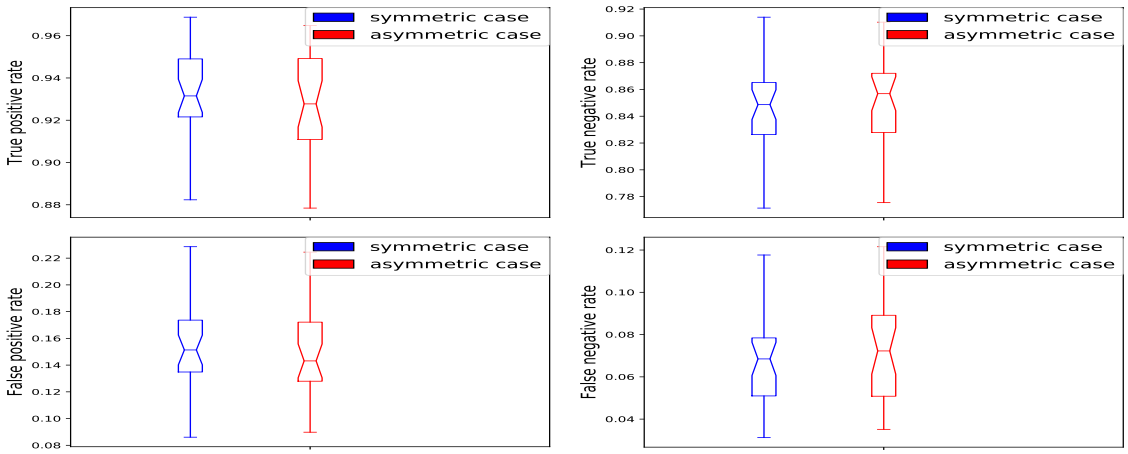

Figure 6.3. Results obtained on the Gmm5test dataset.The parameters used for the beta distribution were $\alpha = \beta = 1.6$ for the symmetric case and $\alpha = 1.1$ and $\beta = 1.8$ for the asymmetric case.

rate is displayed.

As we can see from these plots, in general the error bars are large, leading to the AUCs being similar for all ranges of β for fixed α and for different α. The most significant difference between the symmetric and asymmetric case was achieved for the end points $\alpha = 1$ and and $\beta = 2$. However, for these values the concavity of the function is not satisfied. Similar behavior has been seen for the true negative rate, false positive or false negative rate and for all datasets considered.

Our empirical results can be justified by a theoretical finding from [43]. Zimmerman states that applying an asymmetric impurity to decision trees is equivalent to setting a

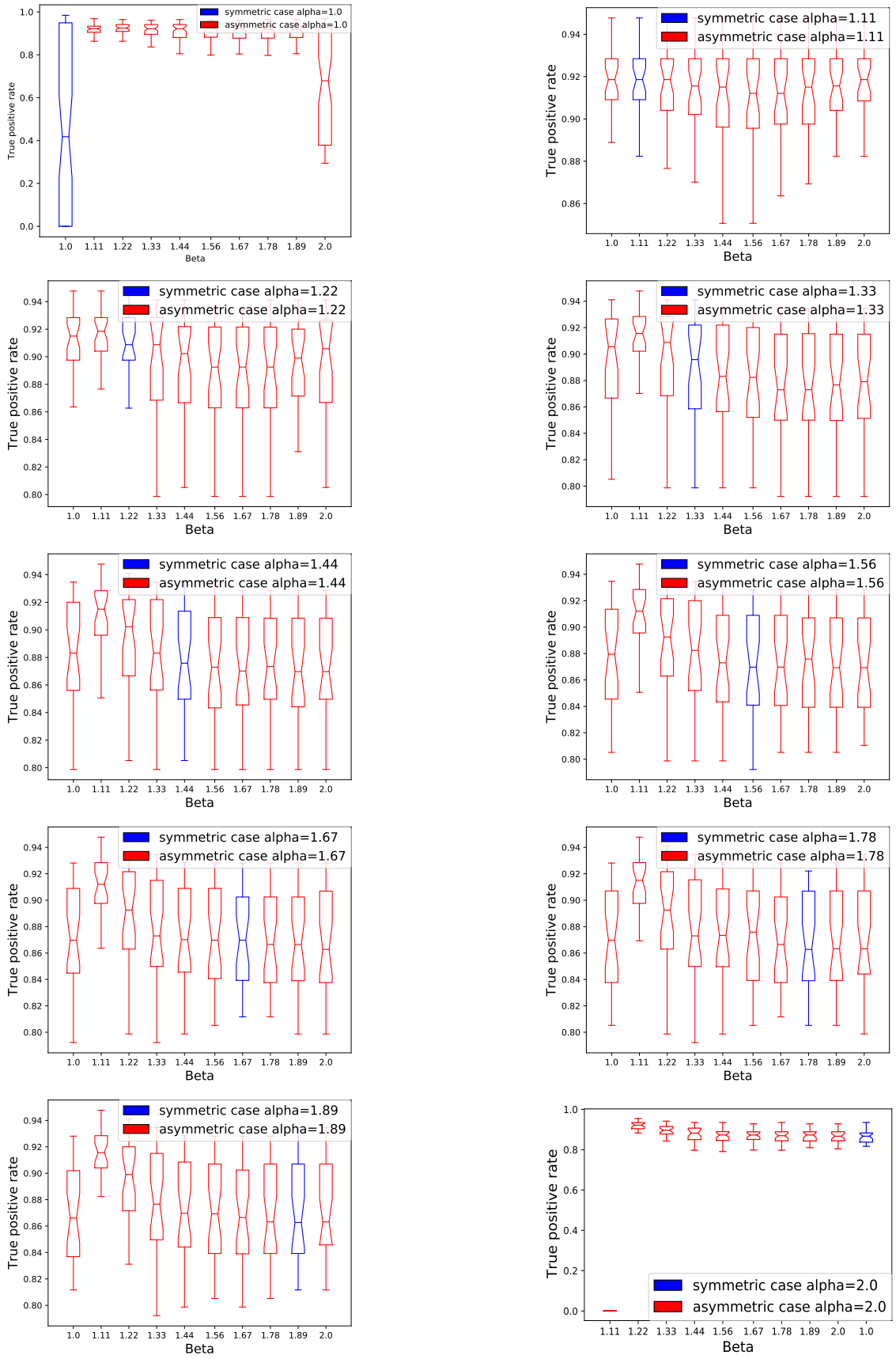

Figure 6.4. Results obtained on the Australian dataset for the true positive rate.

cost for each of the classes. Also, he proves mathematically that functions equal to a scalar multiplied to a function of the form $f(p) = p^{\alpha}(1-p)^{1-\alpha}$ are cost-insensitive. A function

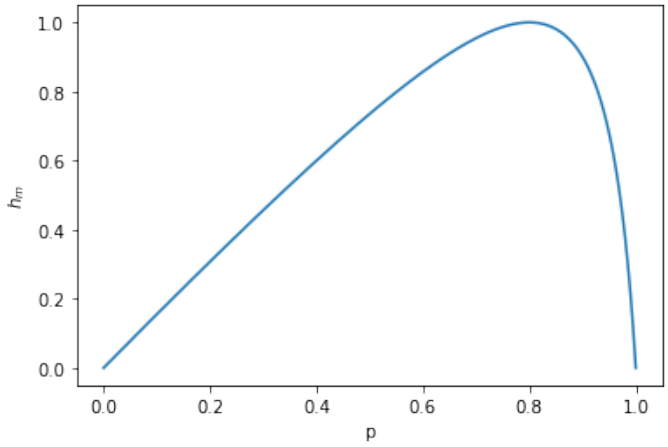

Figure 6.5. The values of the h_m distribution are plotted on the y-axis, whereas the corresponding probabilities are shown on the x-axis. This plot was obtained for $m = 0.8$

Table 6.1. Dataset characteristics

Datasets	Patterns	Features	Imbalance of positive class
Hepatitis	155	19	21%
Satimage	6435	36	10%
Hypothyroid	3772	28	8%
Segment	2310	19	14%
Breast	699	9	34%

that is cost-insensitive is not affected by the class weighting and therefore it will not bias any of the classes. Although the ROC curves are the "same", the same (TP, FP) point on each curve is obtained for different costs or equivalently for different parameterisations of the beta distribution.

6.2.2 h_m impurity

We continue our study with a different impurity function, defined as:

$$h_m(p) = \frac{p(1-p)}{(-2m+1)p + m^2},\ m \in (0,1) \tag{6.2}$$

The curve of this function is displayed in Figure 6.5.

We repeated the initial experiments from the previous section with the beta distribution on the datasets displayed in Table 6.1.

The box plots of the true positive, true negative, false positive and false negative rates and the ROC curves of the forests are displayed in Figures 6.6 and 6.7. Even though the box plots show significant difference, they only represent the values for the 0.5 threshold. The ROC curves corresponding to several thresholds show similar behaviour between the symmetric and asymmetric case.

A justification for this behavior can be found in [43]. The authors have shown that the impurity function h_m from Equation (6.2) can be written as a transformation applied to

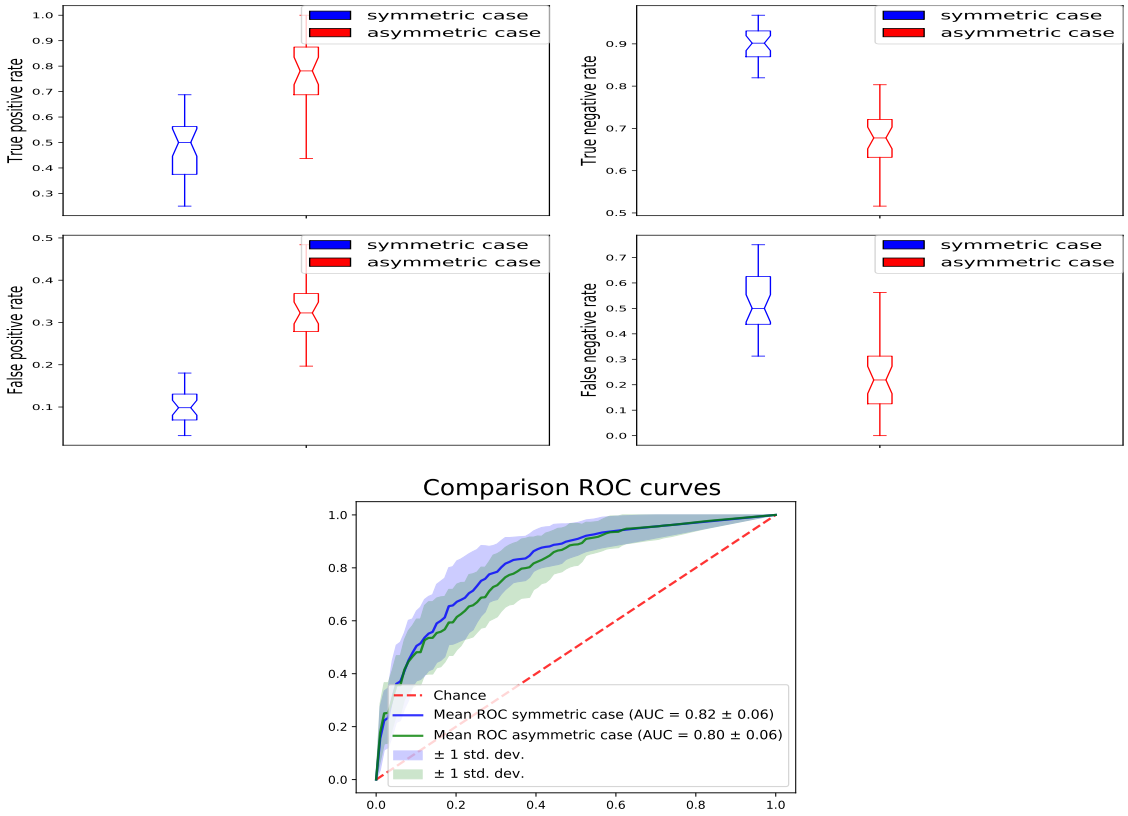

Figure 6.6. Results obtained on the Hepatitis dataset. The ROC plots for the symmetric and the asymmetric case are presented in blue, respectively green. These results were obtained for $m = 0.12$.

the Gini impurity and therefore produce similar results:

$$h_m(p) = \frac{1}{2(1-m)^2} T_w g \tag{6.3}$$

where $w = \left(\frac{1}{m} - 1\right)^2$ and g is the Gini function.

Using this result, we tried to analyze the effect that varying the weights, has on the ROC curves. Since some asymmetric impurities produce the same splits as a tree built with the Gini index but with different weights, we built forests of trees having different sample weights and compared them with the forests having weight one (obtained by using the Gini index). The ROC curves obtained are presented in Figure 6.8.

The line in orange denotes the forest for which the weights were uniformly distributed between 10^{-3} and 10^3. The other forests were formed of samples of equal weights: 0 denoted by the blue curve, -3 by the green and 3 by the red.

The ROC curves for the equal sample weights and the random sample weights are essentially the same. The ROC curves for the extremely asymmetric weights are similiar, but if the weights are biased away from the positive class (the "-3" case) then part of the ROC curve is not accessible, because there are no trees that can classify the positive examples as positive. The results show that for these particular weights there is not much significant difference between symmetric and asymmetric weights, therefore could be a question of

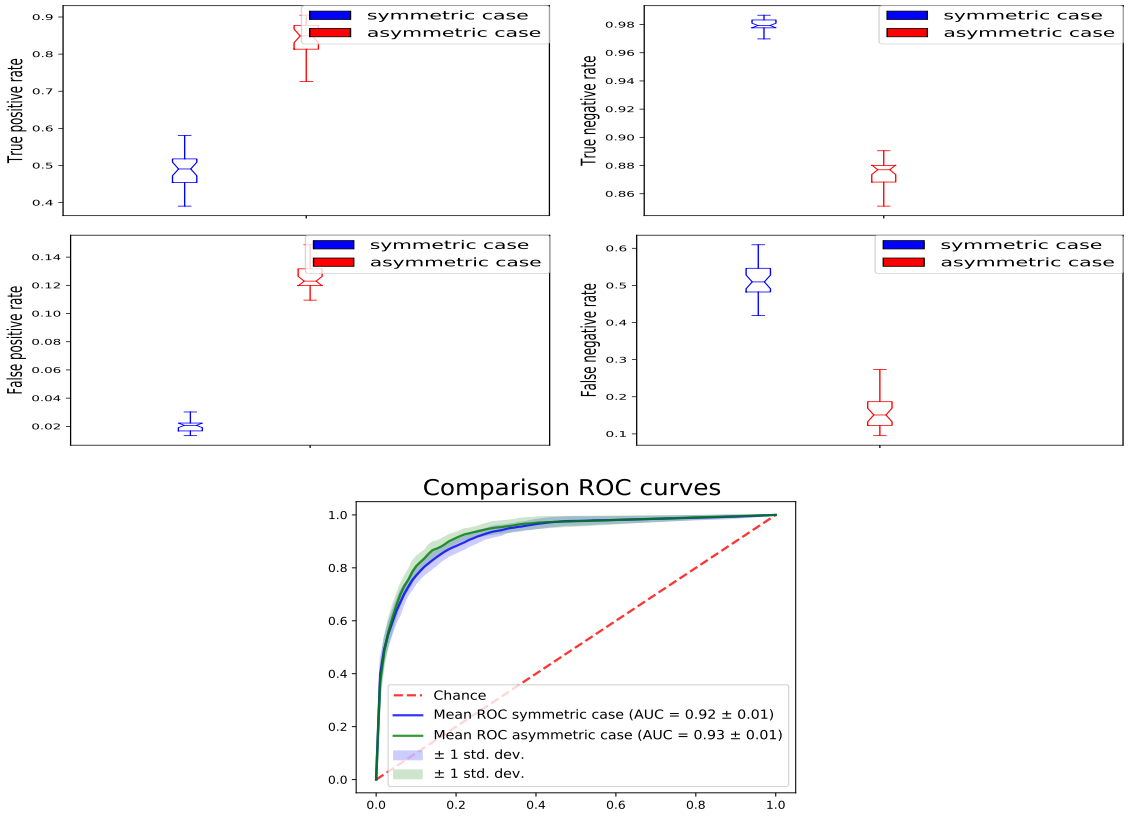

Figure 6.7. Results obtained on the Satimage dataset. The ROC plots for the symmetric and the asymmetric case are presented in blue, respectively green. These results were obtained for $m = 0.12$.

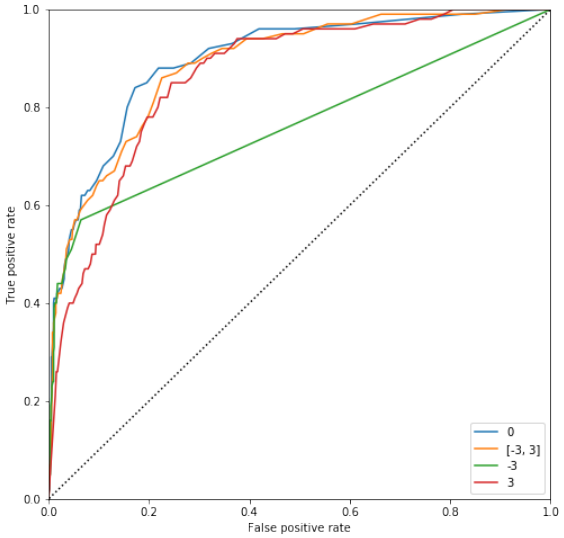

Figure 6.8. Comparison of the ROC curves of the forests obtained for different sample weight ranges. The line in orange denotes the case when the weights were uniformly distributed between 10^{-3} and 10^{3}. The other ROC curves were obtained when the weights were equal. We have compared for the following weight values: 0 denoted by the blue curve, -3 by the green line and 3 by the red line. The results are obtained on the GMM5 dataset.

choosing the right weights.

Since these results were obtained for specific weights and for a specific threshold fixed (0.5), next we tried to optimise the sample weights and threshold. We optimised them

in a bi-objective problem of minimising the false positive rate and the complement of the true positive rate (1-TPR, where TPR stands for true positive rate). The weights, w_i, $i = 1 \dots M$ were represented as a softmax:

$$w_i = \frac{e^{\theta_i}}{\sum_{i=1}^{M} e^{\theta_i}} \tag{6.4}$$

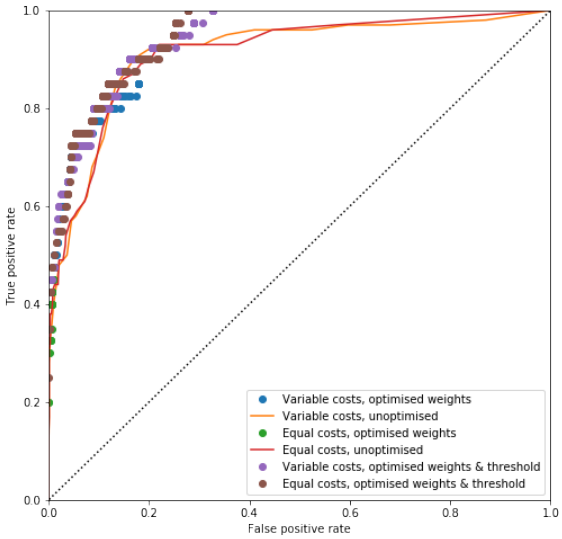

Figure 6.9. Comparison of the ROC curves of the forests obtained for equal weights (denoted in red), random weights (orange), optimised random weights (blue), optimised equal weights (green), optimised random weights and threshold (purple) and optimised equal weights and threshold (brown). The results are obtained on the GMM5 dataset.

Figure 6.9 essentially shows that the optimisation of the weights within the forest works better for the forest with the randomly weighted samples (equivalently, randomly asymmetric impurity functions) as a greater range of the ROC curve is accessible. However, each of the forests with unoptimised tree-weights has an ROC curve that appears to be equivalent to the optimised curves.

This figure shows that by optimising the threshold along with the weights, a better TPR/FPR trade-off is obtained. However, having variable weights for the samples (equivalently, asymmetric impurity) doesn't give any benefit in this case. By analysing the curves we can see that there is no significant difference between the symmetric forests or forests built with asymmetric impurities equivalent to a reweighting of the Gini function.

6.2.3 $p - p^\alpha$ impurity

An impurity function that does not fall in any of the previous mentioned categories (being equivalent to a transformation applied to a symmetric function or being of the form $ap^\alpha(1-p)^{1-\alpha}, \alpha \in (0,1)$, where a is positive constant) is the following:

$$f(p) = p - p^\alpha,\ \alpha \geq 3 \tag{6.5}$$

The curve of this impurity function is displayed in Figure 6.10.

We generated for values of $\alpha \in [3,4,5,6]$ forests formed of 100 trees and produced the

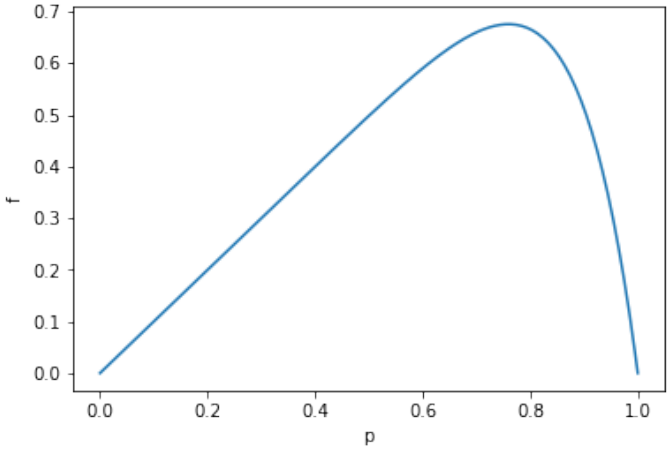

Figure 6.10. The values of the $p - p^{\alpha}$ are plotted on the y-axis, whereas the corresponding probabilities are shown on the x-axis. This plot was obtained for $\alpha = 9$.

corresponding ROC curves from Figure 6.11. The green ROC curve represents the asymmetric case, whereas the blue one represents the symmetric case. The plots were obtained by averaging over 10 folds, the bold line corresponds to the mean, and the upper and lower bound represent the values of the mean $\pm$ the standard deviation. The yellow stars represent the number of times the true positive rate for the asymmetric case was higher than the true positive rate of the symmetric case.

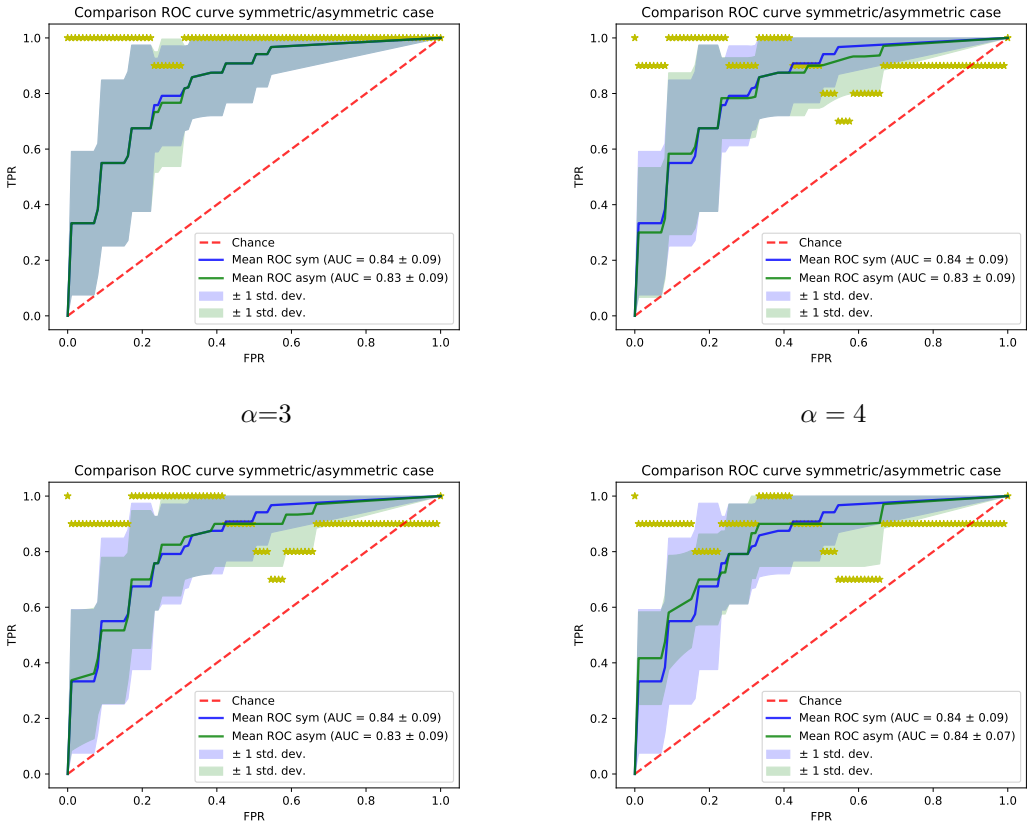

Figure 6.11. Hepatitis dataset

Again, there was no significant difference in behaviour between the symmetric and asym-

metric case. Next, we tried to compare the behaviour of the impurities when the parameters α are varied for each tree, which can be considered a way of ensuring diversity. In these experiments we generated the ROC curves for forests of 100 trees, in which the impurity function was either the Gini index for the symmetric case and $p - p^{\alpha}$ for the symmetric case.

Each tree was built with a randomly chosen $\alpha \in [3, 4, 5, 6, 7, 8, 9]$. These plots were obtained for the 0.1 threshold. We investigated the influence that each α value has on the true positive or false positive rates. We conducted this analysis per α value and also grouped sub-ensembles of trees that were built with the same α value. The left figure in the panel from Figure 6.12 shows the mean and interquartile ranges for the ROC curves obtained for the ensemble with mixed α values in blue and for the sub-ensembles formed with the same α in the remaining colours. The right plot from the same figure, shows the average true positive and false positive rate of the sub-ensembles for the 0.1 threshold, per α. These results were obtained for the Satimage dataset. For the Hepatitis dataset, the corresponding plots are presented in Figures 6.14.

By comparing these two set of plots, we can conclude that varying the alpha per tree, does not have a significant impact on the ROC curve. Also by comparing the right figures of the average FPR and TPR for each sub-ensemble, we cannot see a pattern for a specific α positively influencing the prediction. We compared the behaviour of the forest built with different values of α with the forests obtained in the previous experiments. Figure 6.13 presents a comparison of the ROC curves for the symmetric case, the asymmetric cases, where α was the same for all trees, versus the asymmetric case when α was randomly selected, in the case of the Satimage dataset. For the Hepatitis dataset, the corresponding plots are presented in Figure 6.15. These plots do not show a significant difference between the behaviour of the three approaches.

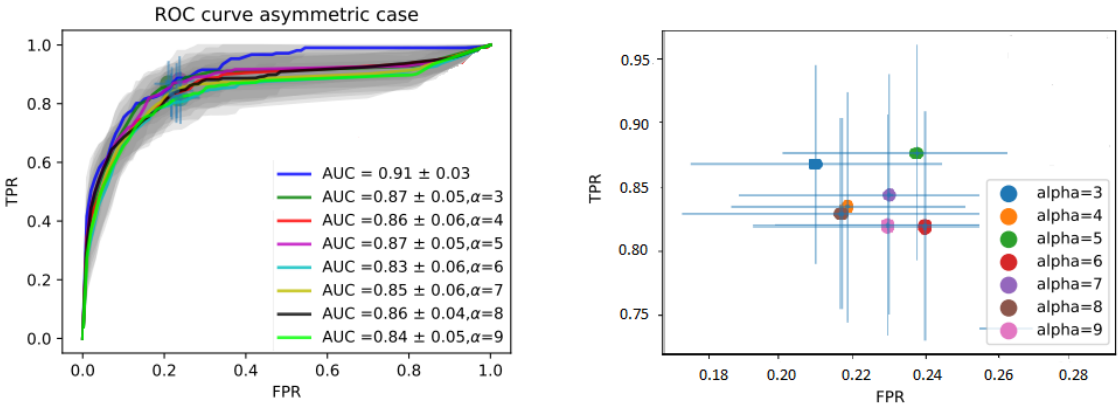

Figure 6.12. The left plot shows the mean ROC curves for the ensemble build with different values of α and for the sub-ensembles built with the same values of α. The values for α ranged between [3,4,5,6,7,8,9]. The right figure presents the average FPR vs TPR for each ensemble built with a specific α. These results were obtained for the Satimage dataset.

We could not express the impurity function f as being the result of a transformation T_{ω} applied to a symmetric function, as in Definition 2.4.5. Therefore, in order to justify the similarity in performance of the symmetric impurity and the $p - p^{\alpha}$ impurity, by

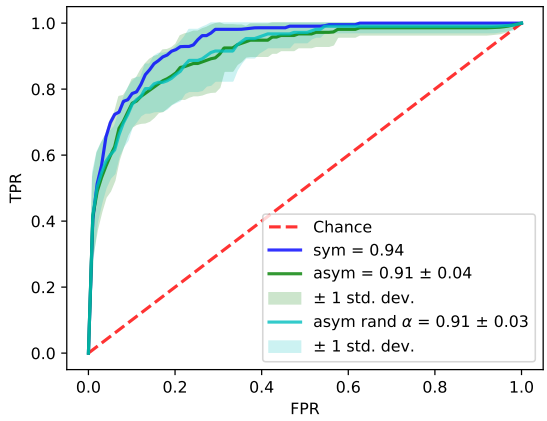

Figure 6.13. Comparison ROC curves for ensembles built with random α in turquoise, Gini index in blue or $\alpha = 7$ in green. There results were obtained for the Satimage dataset.

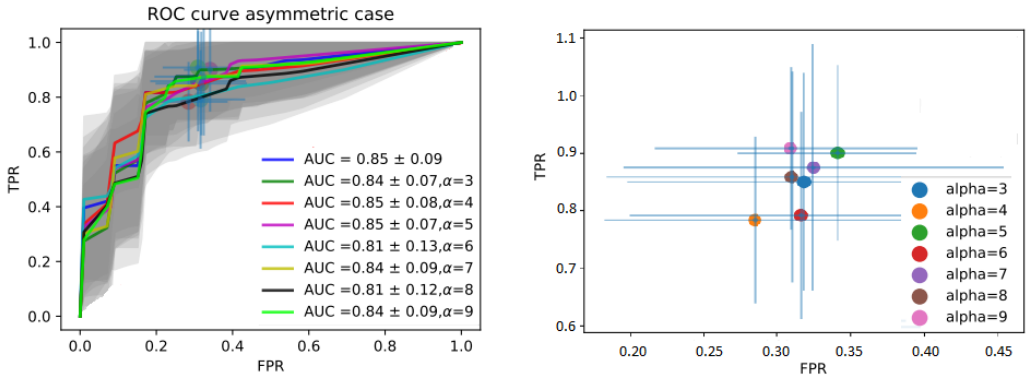

Figure 6.14. The left plot shows the mean ROC curves for the ensemble build with different values of α and for the sub-ensembles built with the same values of α. The values for α ranged between [3,4,5,6,7,8,9]. The right figure presents the average FPR vs TPR for each ensemble built with a specific α. These results were obtained for the Hepatitis dataset

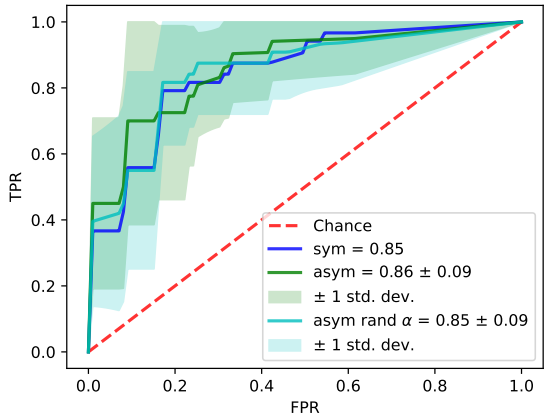

Figure 6.15. Comparison ROC curves for ensembles built with random α in turquoise, Gini index in blue or $\alpha = 7$ in green. There results were obtained for the Hepatitis dataset.

the aid of numerical experiments we found functions that are a result of a transformation applied to symmetric functions, which bound the impurity f. With the aid of optimisation much closer bounds can be found, so that the distance between the target function, f

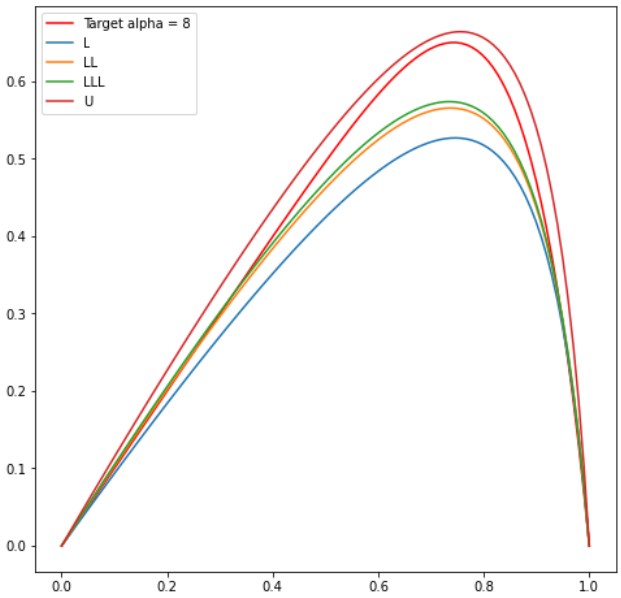

Figure 6.16. Bracketing of the impurity $f = p - p^{\alpha}$, where $\alpha = 8$

and the bounds to be close to an $\epsilon > 0$. Taking into account the result from Equation (6.3), that function h can be defined as a transformation applied to the Gini function, a transformation applied to it, will be another transformation applied to the Gini function. By applying sequentially the transformation, we define the following functions:

$$L(p, \omega, m) = (1 + (\omega - 1)p)h(\Phi(p, \omega), m) \tag{6.6}$$

where h is the function from Equation 6.2 and Φ from Definition 2.4.4.

$$LL(p, \omega, m) = (1 + (\omega_L - 1)p)L(\Phi(p, \omega_{LL}), \omega, m) \tag{6.7}$$

$$LLL(p, \omega, m) = (1 + (\omega_{LL} - 1)p)LL(\Phi(p, \omega_{ll}), \omega, m) \tag{6.8}$$

$$LLLL(p, \omega, m) = (1 + (\omega - 1)p)LLL(\Phi(p, \omega_{ll}), \omega, m) \tag{6.9}$$

$$U(p, \omega, m) = (1 + (\omega - 1)p)h(\Phi(p, \omega), m) \tag{6.10}$$

A possible bracketing of the function $f = p - p^{\alpha}$, where $\alpha = 8$ can be seen in Figure 6.16.

In this section we have compared the results of forests built with a symmetric impurity versus an asymmetric impurity $p - p^{\alpha}$. We have compared the behaviour for ensembles of trees built with the same value of α or when each tree was built with a different value of α. Our results have shown that there is not a statistical difference in performance between the impurities used, on the particular problems we considered.

The impurity function $p - p^\alpha$ cannot be written as a transformation applied to a symmetric function, as in Equation 6.3. Therefore, in order to justify the similarity between the predictions, we have shown numerically that this function can be bracketed by functions which are a transformation of the Gini index and therefore produce similar results. However, we could not find analytically functions to bound this impurity function. One possible way of dealing with this problem would be to define analytically functions which can be written as a transformation applied to the Gini index and which bracket the function $p - p^\alpha$. In this way their splits will be similar and it can justify why there was no significant difference between the results of the Gini index and the impurity function $p - p^\alpha$. More details are presented in chapter 7.

6.3 Conclusion

Imbalanced datasets often occur in real world problems, where the minority class is of greater interest. In such scenarios asymmetric impurities have been used in order to predict the minority class. In our experiments we have tried to asses how well a series of asymmetric impurities performs on predicting the minority class. We have compared the predictions of forests build with symmetric impurities versus ensembles formed with asymmetric impurities, however in the case of asymmetric impurities there was no significant bias towards the minority class and the behaviour was statistically indistinguishable to the case of symmetric impurities. Amongst the impurities considered, most of them were proven to be equivalent to a symmetric impurity [43]. In the case of the impurity function $f = p - p^\alpha$ we have showed numerically that it can be bracketed by a series of impurity functions obtained by applying a transformation to a symmetric impurity, hence the similar prediction performance. However, we could not establish the bounds of this function analytically. Even though our results could not demonstrate the benefit of using asymmetric impurities, it might still be a question of choosing the right impurity function.

Chapter 7

Conclusion and future work

This chapter summarizes the results in the previous chapters and presents the main results obtained in this thesis. Also a number of ideas which could be explored in future work are presented.

Ensembles are important models used in machine learning, because of their prediction capabilities which surpass those of individual classifiers. In Chapter 2 we present methods for reducing ensemble error, including the bias and variance trade-off. It has been shown that the prediction performance (accuracy) is positively influenced by diversity. We present a number of state-of-the-art algorithms that optimise accuracy and diversity.

However there is a question about which diversity measure to use. In Chapter 3, we investigate empirically the effect that different diversity measures have on ensemble performance. The diversity measures considered were: ambiguity [1], coincident failure (CFD) [24], disagreement (DIS) [25], Kohavi-Wolpert (KW) [26] and a new measure, called coherence. The coherence measures the angle between the prediction of a classifier and the ensemble's prediction. The diversity measures that had the strongest negative correlation with the generalisation error were the ambiguity, followed by the DIS and KW. Our experiments have shown that at high diversity there is little consistent correlation with the test error. We have analysed the effect that bootstrapping with or without replacement has on the the generalisation error. Our empirical results showed that at low diversity there is a negative correlation between diversity and test error, in the case of bootstrapping without replacement. Another conclusion was that the ensembles generated by bootstrapping with replacement had test errors which were not correlated with diversity. On the whole bootstrapping without replacement generated ensembles with lower generalisation error.

We have also investigated the effect that providing more data has on the generalisation error. The conclusion was that by increasing the size of the dataset the prediction performance will increase. Since providing more data will decrease the computational speed on fitting new models and predicting on unseen data, pruning techniques have been used in the literature.

In our next set of experiments we have defined a series of pruning techniques: "Remove

successive best" (trees that have the highest difference between individual error and individual ambiguity/coherence are removed), "Remove successive worst" (the opposite of the previous one) and "Linear random" (at the $n+1$th iteration the same n trees from the previous iteration are removed, plus a random new one). Our experiments revealed that generalisation error was not considerably better when pruning according to diversity than when choosing random ensembles. Hence, in the experiments in the following chapter we focused on small ensembles.

Based on the bias-variance decomposition, in Chapter 4 we introduced two new ambiguity measures, obtained by using the cross-entropy error or the hinge loss. We have analysed the properties of these new diversity measures and found that only the ambiguity derived from the cross-entropy error satisfies all the properties of a diversity measure, being always positive and zero if and only if all the classifiers' predictions agree.

We presented an evolutionary algorithm which evolves the training patterns of the classifiers, in order to maximise the ambiguity obtained from the cross-entropy (amb_{CE}). Our experiments have shown that in general, the evolved ensemble has a better generalisation error than the initial ensemble. Hence, our results support the influence that the diversity has on minimising generalisation error. We also investigated the influence that using random sampling on selecting patterns on which ensemble members are trained has on the generalisation error. We found that generalisation error is negatively correlated with diversity at high sampling rates; conversely generalisation error is positively correlated with diversity when the sampling rate is low and the diversity high.

We have used in our experiments decision trees, therefore a possible extension of our work would be to use different learners. Deep neural networks have achieved a high predictability performance, however they are expensive to train. Since our experiments were designed in the case of small ensembles, deep neural networks seem to be a good candidate for a possible extension. A possible route could be to analyse the influence that diverse/ambiguous ensembles of deep neural networks have on the generalisation error, by using the amb_{CE} measure. One starting point could be to evolve ambiguous ensembles of deep neural networks, with the aim of achieving low generalisation error. We can start from a set pool of M deep neural networks and evolve the ensemble membership, by selecting a fixed subset of them of size m throughout the generations. The deep neural networks selected can be represented via a string of 0s and 1s, where 1 on the ith position signifies that the ith neural network is selected and 0 that is not. At each generation the current string can be mutated according to mutation rate μ. One approach could be to keep the most ambiguous ensemble at each generation and in case of equality the one with the lowest training error to be preferred. The pseudocode of this approach is presented in Algorithm 4:

Our experiments were performed in the classification case, another possible extension could be to determine different ambiguity measures in the regression case and examine their impact on the ensemble error. In [21] an ambiguity measure was defined by using the bias-variance decomposition and the quadratic error $L_{quadratic} = \sum_{i=1}^{N}(t_n - Y_n)^2$. Other possible losses that can be used in the regression case in order to obtain new diversity

Algorithm 4 Evolutionary algorithm for selecting deep neural networks by optimising diversity

Require: $X = \{\mathbf{x}_n\}_{n=1}^{N}$ ▷ training data
Require: $t = \{t_n\}_{n=1}^{N}$ ▷ targets
Require: m ▷ number of deep neural networks desired
Require: NN_list ▷ list of all possible deep neural networks
Require: g ▷ number of generations
Require: μ. ▷ mutation rate
Ensure: $\mathcal{NN}$ ▷ evolved ensemble

$M \leftarrow len(NN\_list)$
$\mathcal{NN} \leftarrow initialize(X, t, M)$ ▷ generate a random ensemble
for $i = 1 \rightarrow g$ **do**
  $\mathcal{NN}' \leftarrow mutate(\mathcal{NN}, m, \mu)$
  if $(\text{amb}_{CE}(\mathcal{NN}') > \text{amb}_{CE}(\mathcal{NN}))$ **or** $(\text{amb}_{CE}(\mathcal{NN}') = \text{amb}_{CE}(\mathcal{NN})$ **and** $L_{log}(\mathcal{NN}', t) < L_{log}(\mathcal{NN}, t))$ **then**
    $\mathcal{NN} \leftarrow \mathcal{NN}'$
return $\mathcal{NN}$

measures are: the $\log - \cosh$ loss (which can be used in robust statistics) and the quantile loss [118].

The $\log - \cosh$ loss for the nth target and ith classifier is defined as:

$$L_{cosh}(y_{in}, t_n) = \log(\cosh(y_{in} - t_n)) \tag{7.1}$$

By using equation 4.1, the ambiguity derived from the $\log - \cosh$ loss of an ensemble $\mathcal{Y}_n$ for the nth pattern is:

$$amb_{\log-\cosh}(\mathcal{Y}_n, t_n) = \log\left(\frac{\prod_{i=1}^{M}\cosh^{c_i}(y_{in} - t_n)}{\cosh(\sum_{i=1}^{M} c_i y_{in} - t_n)}\right) \tag{7.2}$$

The quantile loss for the qth quantile, the nth target and ith classifier is defined as follows:

$$L_{QL}(y_{in}, t_n) = \max\left[q(y_{in} - t_n), (q-1)(y_{in} - t_n)\right] \tag{7.3}$$

As before, we can define the ambiguity derived from the quantile loss, of ensemble $\mathcal{Y}_n$, on the nth target as:

$$amb_{QL}(\mathcal{Y}_n, t_n) = \sum_{i=1}^{M} c_i \max\left[q(y_{in} - t_n), (q-1)(y_{in} - t_n)\right] - \max[q(\sum_{j=1}^{M} c_j y_{jn} - t_n), \tag{7.4}$$

$$(q-1)(\sum_{j=1}^{M} c_j y_{jn} - t_n)] \tag{7.5}$$

These new types of ambiguities can be employed in evolutionary algorithms, like the ones suggested in Algorithm 2 from Chapter 4 and Algorithm 3 from Chapter 5.

Since in Chapter 4 we focused on small ensembles, in Chapter 5 we present pruning techniques and investigate their impact on reducing generalisation error. We have defined different methods of tree selection, a number of them that favoured ambiguous/accurate ensembles and others that promoted less ambiguous/accurate ensembles. We compared the performance of these methods with the results of the evolutionary algorithm from Section 5.4 and from Chapter 4, Section 4.4. Our results show that in general the evolutionary algorithm achieves good performance, however its performance is similar to the pruning approaches that take into account ambiguity, which are also faster in terms of computational speed, hence being preferred. This result shows once more the usefulness of ambiguity in error reduction.

In the previous chapters our results were based on forests of trees, built with the same impurity, where diversity was quantified based on the predictions of the individual trees. In Chapter 6 we inject diversity in the forests by building trees with different impurities. We built our experiments by choosing different families of impurities which are characterized by different parameters and analysed their impact on the generalisation performance. By tuning the parameters we can define symmetric or asymmetric impurities. Asymmetric impurities have been proven to be beneficial in the case of imbalanced datasets, where the minority class is usually of greater importance. We compare the behaviour of the forests by using symmetric and asymmetric impurities with forests of trees built with different impurities (different parameters). Our experiments have shown that in the case of asymmetric impurities there was no significant bias towards the minority class and the behaviour was very similar to the case of symmetric impurities. Amongst the impurities considered, most of them were proven to be equivalent to a symmetric impurity [43], hence the similar behaviour with the symmetric impurities. In the case of the impurity function $f = p - p^{\alpha}$, in order to justify the similarity in predictions with the symmetric case, we have showed empirically that it can be bracketed by a series of impurity functions obtained by applying a transformation to a symmetric impurity. Even though our results could not present evidence of the benefit of using asymmetric impurities, it might still be a question of choosing the right impurity function.

As further avenue for future work, it would be useful to find an analytical explanation of why the asymmetric impurity $f = p - p^{\alpha}$ has a similar behaviour to the symmetric impurity. One possible approach to addressing this problem would be to identify functions that can be expressed as transformations applied to the Gini index and which bound the function $p - p^{\alpha}$. In this way their splits will be similar and it can prove the similarity between the results of the Gini index and the impurity function. In [43] the authors have defined the transformation of a function $f : [0, 1] \to R$ as

$$T_{\omega} f(p) = (1 + (\omega - 1)p) \, (f \circ \Phi_{\omega}) \, (p) \tag{7.6}$$

where $\omega > 0$ and $\Phi_{\omega} : [0, 1] \to [0, 1]$ is defined as :

$$\Phi_{\omega}(p) = \frac{\omega p}{1 + (\omega - 1)p} \tag{7.7}$$

The next step would be to find symmetric impurities whose transformation will bracket the function $f(p) = p - p^\alpha$. We are looking for functions of the form

$$G(p) = a(T_\omega g)(p) = a(1 + (\omega - 1)p)\,(g \circ \Phi_\omega)\,(p) \tag{7.8}$$

where g is a symmetric function and a a constant.

We will consider the Gini impurity $g(p) = 2p(1-p)$. As a result from Equation (7.8), we will have :

$$\begin{aligned} G(p) = a(1 + (\omega - 1)p)2\Phi_\omega(p)(1 - \Phi_\omega(p)) &= 2a(1 + (\omega - 1)p)\frac{\omega p}{1 + (\omega - 1)p}\left(1 - \frac{\omega p}{1 + (\omega - 1)p}\right) \\ &= 2a(1 + (\omega - 1)p)\frac{1 + (\omega - 1)p - \omega p}{1 + (\omega - 1)p} = \frac{2a\omega p(1-p)}{1 + (\omega - 1)p} \end{aligned}$$

We will define two functions U and L to be the upper bound function and lower bound, respectively, of the function $f(p) = p - p^\alpha$. In order for a function to be bounded by another, we can impose a set of conditions on the tangents of the functions in some points, for example 0 and 1. For the case of L, which we want to serve as the lower bound, we can impose that its tangent's values at 0 and 1 be smaller than f's tangents at 0 and 1, ensuring that the curve of L lies below that of f. Similarly, for the upper bound, where we would like the tangents of U to be greater than the ones of f. Mathematically this can be expressed via the following conditions :

- $\frac{\partial L}{\partial p}(0) \leq \frac{\partial f}{\partial p}(0) = 1$
- $\frac{\partial L}{\partial p}(1) \leq \frac{\partial f}{\partial p}(1) = 1 - \alpha$
- $\frac{\partial U}{\partial p}(0) \geq \frac{\partial f}{\partial p}(0) = 1$
- $\frac{\partial U}{\partial p}(1) \geq \frac{\partial f}{\partial p}(1) = 1 - \alpha$

Next steps would involve optimising the parameters a and w in order to find closer bounds of function f.

Appendices

Appendix A

A.1 Pruning plots involving the coherence diversity measure

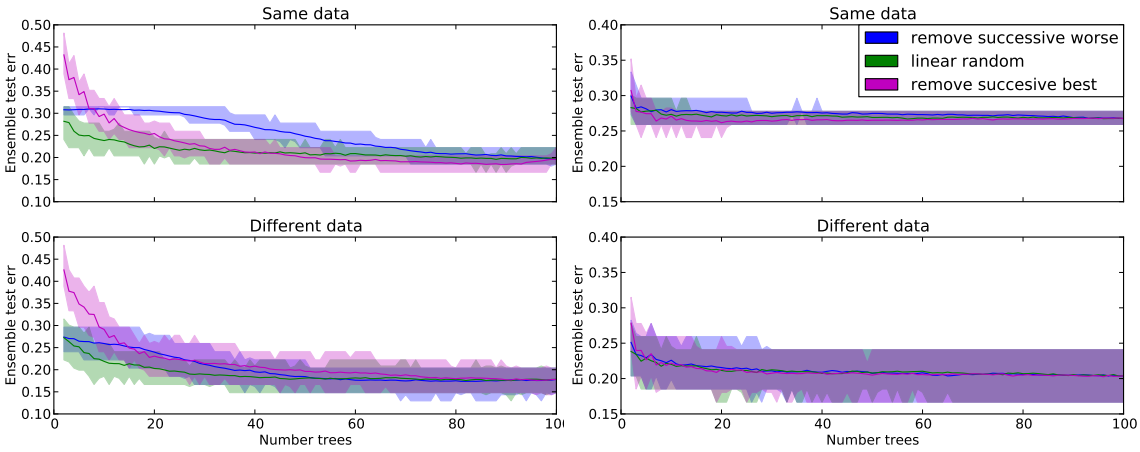

Figure A.1. Comparisons of the pruning approaches involving the coherence diversity measure for the Heart dataset, for sampling rates 0.1 and 0.75. The results for the 0.1 sampling rate are presented in the first column, whereas the 0.75 sampling rate in the second column. The top figure of each column displays the results when the same data was used, whereas the bottom plot when different data was used.

A.2 Ambiguity measures

Let us consider the ensemble prediction to be:

$$Y_n \equiv Y(x_n) = \sum_{i=1}^{M} c_i y_{in} \tag{A.1}$$

where M is the number of classifiers, x_n the n^{th} pattern, y_{in} is the i^{th} classifier's prediction for the n^{th} pattern and c_i the non-negative weights which have the following property $\sum_{i=1}^{M} c_i = 1$.

We denote the outputs of the ensemble members when classifying patterns x_n by

$$\mathcal{Y}_n = \{y_{in}\}_{i=1}^{M}.$$

Also we denote the targets as t_n, where $n \in \{1 \dots N\}$ and N is the total number of

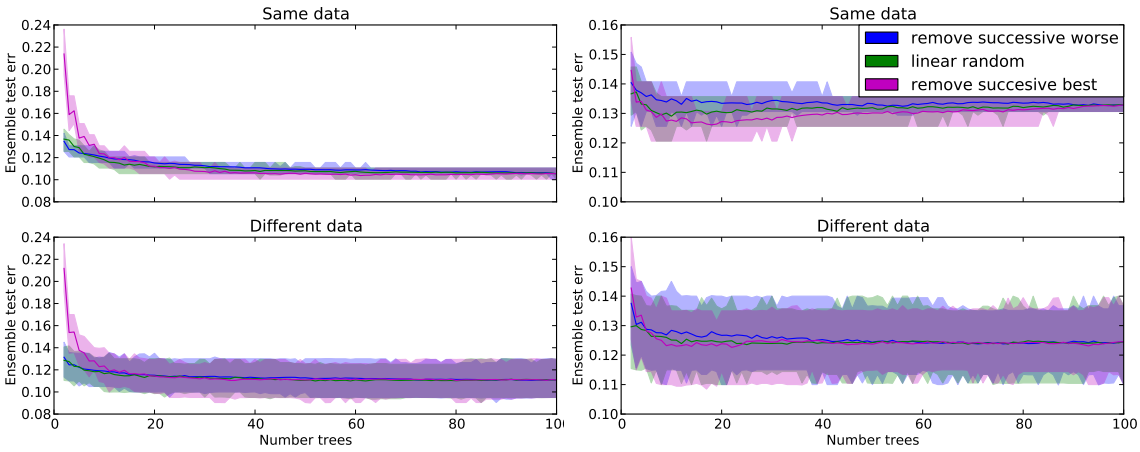

Figure A.2. Comparisons of the pruning approaches involving the coherence diversity measure for the GMM5 dataset, for sampling rates 0.1 and 0.75. The results for the 0.1 sampling rate are presented in the first column, whereas the 0.75 sampling rate in the second column. The top figure of each column displays the results when the same data was used, whereas the bottom plot when different data was used.

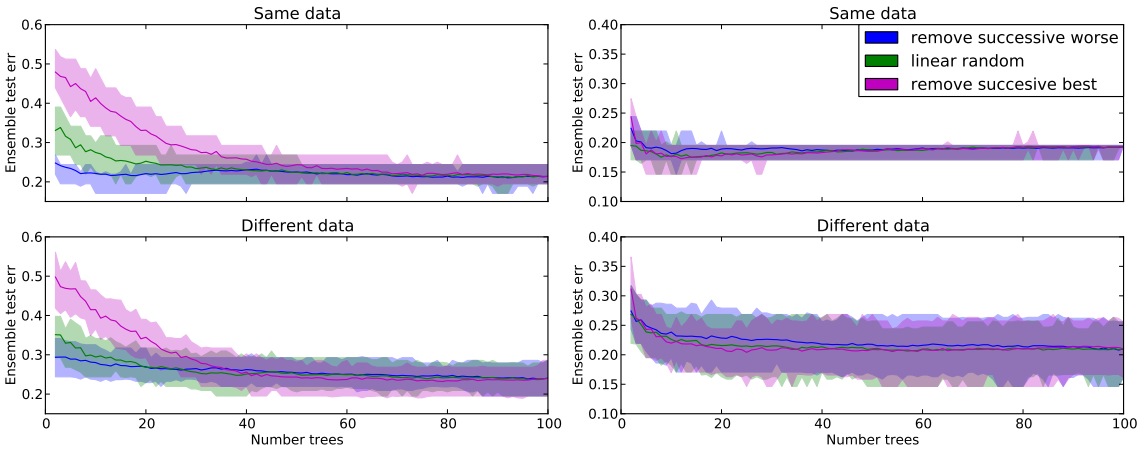

Figure A.3. Comparisons of the pruning approaches involving the coherence diversity measure for the Sonar dataset, for sampling rates 0.1 and 0.75. The results for the 0.1 sampling rate are presented in the first column, whereas the 0.75 sampling rate in the second column. The top figure of each column displays the results when the same data was used, whereas the bottom plot when different data was used.

patterns.

A.2.1 Ambiguity measure obtained from the cross-entropy

Lemma 1. *The formula for the diversity measure obtained from the cross entropy error is :*

$$\text{amb}_{CE}(\mathcal{Y}_n) \triangleq t_n \log\left(\frac{\sum_{i=1}^{M} c_i y_{in}}{\prod_{i=1}^{M} y_{in}^{c_i}}\right) + (1-t_n)\log\left(\frac{\sum_{i=1}^{M} c_i (1-y_{in})}{\prod_{i=1}^{M} (1-y_{in})^{c_i}}\right). \tag{A.2}$$

Proof. The cross-entropy error is defined as:

$$L_{log}(y_{in}, t_n) = -\left[t_n \log(y_{in}) + (1-t_n)\log(1-y_{in})\right]. \tag{A.3}$$

In this case, y_{in} is the i^{th} classifier's probability prediction that the n^{th} pattern belongs

to class 1. From equation (A.3), we could express the average error of the classifiers as:

$$\sum_{i=1}^{M} c_i L_{log}(y_{in}, t_n) = -\sum_{i=1}^{M} c_i [t_n \log(y_{in}) + (1 - t_n) \log(1 - y_{in})]. \tag{A.4}$$

Then from (A.3) we can define the ensemble error as being:

$$L_{log}(Y_n) = -[t_n \log(Y_n) + (1 - t_n) \log(1 - Y_n)]. \tag{A.5}$$

From equations (A.1) and (A.5) we can define the ensemble error as:

$$L_{log}(Y_n) = -\left[t_n \log\left(\sum_{i=1}^{M} c_i y_{in} \right) + (1 - t_n) \log\left(1 - \sum_{i=1}^{M} c_i y_{in} \right) \right]. \tag{A.6}$$

Using the above equations, the difference between the average error of the classifiers and the ensemble error can be expressed as:

$$\begin{aligned} \sum_{i=1}^{M} c_i L_{log}(y_{in}, t_n) - L_{log}(Y_n) = &-\sum_{i=1}^{M} c_i [t_n \log(y_{in}) + (1 - t_n) \log(1 - y_{in})] \\ &+ t_n \log\left(\sum_{i=1}^{M} c_i y_{in} \right) + (1 - t_n) \log\left(1 - \sum_{i=1}^{M} c_i y_{in} \right). \end{aligned} \tag{A.7}$$

Taking into account the logarithm's properties:

1. $\log(a \cdot b) = \log(a) + \log(b)$
2. $\log(\frac{a}{b}) = \log(a) - \log(b)$
3. $x \log(a) = \log(a^x)$

we obtain:

$$\sum_{i=1}^{M} c_i L_{log}(y_{in}, t_n) - L_{log}(Y_n) = t_n \log\left(\frac{\sum_{i=1}^{M} c_i y_{in}}{\prod_{i=1}^{M} y_{in}^{c_i}} \right) + (1 - t_n) \log\left(\frac{1 - \sum_{i=1}^{M} c_i y_{in}}{\prod_{i=1}^{M} (1 - y_{in})^{c_i}} \right). \tag{A.8}$$

Since $\sum_{i=1}^{M} c_i = 1$ we can rewrite the second logarithm as:

$$\log\left(\frac{1 - \sum_{i=1}^{M} c_i y_{in}}{\prod_{i=1}^{M} (1 - y_{in})^{c_i}} \right) = \log\left(\frac{\sum_{i=1}^{M} c_i (1 - y_{in})}{\prod_{i=1}^{M} (1 - y_{in})^{c_i}} \right). \tag{A.9}$$

As a result the ambiguity in x_n be defined as:

$$\text{amb}_{CE}(\mathcal{Y}_n) = t_n \log\left(\frac{\sum_{i=1}^{M} c_i y_{in}}{\prod_{i=1}^{M} y_{in}^{c_i}} \right) + (1 - t_n) \log\left(\frac{\sum_{i=1}^{M} c_i (1 - y_{in})}{\prod_{i=1}^{M} (1 - y_{in})^{c_i}} \right). \tag{A.10}$$

Taking into account equation A.10, the formula of the diversity for all the patterns is:

$$\text{amb}_{CE}(\mathcal{Y}) = \frac{1}{N} \sum_{n=1}^{N} t_n \log\left(\frac{\sum_{i=1}^{M} c_i y_{in}}{\prod_{i=1}^{M} y_{in}^{c_i}} \right) + (1 - t_n) \log\left(\frac{\sum_{i=1}^{M} c_i (1 - y_{in})}{\prod_{i=1}^{M} (1 - y_{in})^{c_i}} \right). \tag{A.11}$$

□

Theorem 5. *The log loss ambiguity,* amb_{CE}*, has the following properties:*

1. $\text{amb}_{CE}(\mathcal{Y}_n) \geq 0, \forall n \in \overline{1,N}$
2. $\text{amb}_{CE}(\mathcal{Y}_n) = 0, \Leftrightarrow y_{in} = y_{jn} \forall i,j \in \overline{1,M}$

Proof. First we show that $\text{amb}_{CE}(\mathcal{Y}_n) \geq 0, \forall n \in \overline{1,N}$.

Let us consider the following inequality:

$$\frac{\lambda_1 x_1 + \lambda_2 x_2 + \ldots + \lambda_n x_n}{\lambda} \geq \sqrt[\lambda]{x_1^{\lambda_1} x_2^{\lambda_2} \cdot \ldots \cdot x_n^{\lambda_n}} \textbf{ [119]} \tag{A.12}$$

where $\sum_{i=1}^{N} \lambda_i = \lambda, \forall i \in \{1 \ldots n\}, \lambda_i \geq 0$. The equality holds if and only if all the x_i are equal.

In our case $\lambda_i = c_i$ and $\sum_{i=1}^{M} c_i = 1$, as a result the arguments of both logarithms from the formula are greater then one and then the logarithms are positive. Since $t_n \in \{0,1\}$ the diversity is the sum of two positive numbers.

We now show that $\text{amb}_{CE}(\mathcal{Y}_n) = 0, \Leftrightarrow y_{in} = y_{jn} \forall i,j \in \overline{1,M}$. If we have $y_{in} = y_{jn} = \bar{y} \, \forall i,j \in \{1...N\}$, then:

$$\frac{\sum_{i=1}^{M} c_i y_{in}}{\prod_{i=1}^{M} y_{in}^{c_i}} = \frac{\bar{y} \cdot \sum_{i=1}^{M} c_i}{\bar{y}^{\sum_i^M c_i}} = \frac{\bar{y}}{\bar{y}} = 1.$$

The same principle may be applied to the argument of the second logarithm, as a result:

$$\text{amb}_{CE}(\mathcal{Y}_n) = t_n \log(1) + (1 - t_n) \log(1) = 0.$$

For the reverse case, $\text{amb}_{CE} = 0 \Rightarrow y_{in} = y_{jn} \forall i,j \in \{1...N\}$, we distinguish two cases: $t_n = 1$ and $t_n = 0$. When $t_n = 1$, then:

$$\text{amb}_{CE}(\mathcal{Y}_n) = \log\left(\frac{\sum_{i=1}^{M} c_i y_{in}}{\prod_{i=1}^{M} y_{in}^{c_i}}\right) = 0 \Leftrightarrow \frac{\sum_{i=1}^{M} c_i y_{in}}{\prod_{i=1}^{M} y_{in}^{c_i}} = 1 \overset{(A.12)}{\Leftrightarrow} y_{in} = y_{jn} \forall i,j \in \{1...N\}.$$

The proof is similar for the case $t_n = 0$. □

A.2.2 Ambiguity measure obtained from the hinge loss

Lemma 2. *The formula for the diversity measure obtained from the hinge loss error is :*

$$\text{amb}_{HL}(\mathcal{Y}_n) = \sum_{i=1}^{M} c_i \max(0, 1 - t_n y_{in}) - \max\left(0, \sum_{i=1}^{M} c_i (1 - t_n y_{in})\right).$$

Proof. The hinge loss is defined as:

$$L_H(y_{in}, t_n) = \max(0, 1 - t_n y_{in}) \tag{A.13}$$

where y_{in} is the i^{th} classifier score and t_n is the target, $t_n \in \{\pm 1\}$.

Using equations (A.1) and (A.13) we obtain the error of the ensemble prediction in x_n:

$$L_H(Y_n) = \max\left(0, 1 - t_n \sum_{i=1}^{M} c_i y_{in}\right). \tag{A.14}$$

The weighted average error of the classifiers be: $\sum_{i=1}^{M} c_i L_H(y_{in}, t_n) = \sum_{i=1}^{M} c_i \max(0, 1 - t_n y_{in})$.
Using the decomposition equation we obtain:

$$\text{amb}_{HL}(\mathcal{Y}_n) = \sum_{i=1}^{M} c_i \max(0, 1 - t_n y_{in}) - \max\left(0, 1 - t_n \sum_{i=1}^{M} c_i y_{in}\right). \tag{A.15}$$

Since $\sum_{i=1}^{M} c_i = 1$ we can rewrite the previous equation as:

$$\text{amb}_{HL}(\mathcal{Y}_n) = \sum_{i=1}^{M} c_i \max(0, 1 - t_n y_{in}) - \max\left(0, \sum_{i=1}^{M} c_i (1 - t_n y_{in})\right). \tag{A.16}$$

□

Theorem 6.

The hinge loss diversity has the following properties:

1. $\text{amb}_{HL}(\mathcal{Y}_n) \geq 0 \, \forall n \in \{1 \dots N\}$
2. *If for the pattern* $\mathbf{x}_n$ *we have* $\forall \, i, j \in \{1 \dots M\}, y_{in} = y_{jn} \Rightarrow \text{amb}_{HL}(\mathcal{Y}_n) = 0$
3. *If for the pattern* $\mathbf{x}_n$ *we have* $\text{amb}_{HL}(\mathcal{Y}_n) = 0 \not\Longrightarrow$ *that all the classifiers predict the same class.*

Proof.

1. We first show that $\text{amb}_{HL}(\mathcal{Y}_n) \geq 0 \, \forall n \in \{1 \dots N\}$.

   We can distinguish two sub-cases:

   a) If $\max(0, \sum_{i=1}^{M} c_i(1 - t_n y_{in})) = 0$ then from (4.5):

   $$\text{amb}_{HL}(\mathcal{Y}_n) = \sum_{i=1}^{M} c_i \max(0, 1 - t_n y_{in}) \geq 0. \tag{A.17}$$

   b) If $\max(0, \sum_{i=1}^{M} c_i(1 - t_n y_{in})) = \sum_{i=1}^{M} c_i(1 - t_n y_{in}) > 0$

   Let $K_n = \{i \in \{1 \dots M\} | \, c_i(1 - t_n y_{in}) < 0\}$ which can be rewritten as:
   $K_n = \{i \in \{1 \dots M\} | \, |y_{in}| > 1 \, and \, t_n y_{in} > 0\}$.

Then we have

$$\sum_{i=1}^{M} c_i(1 - t_n y_{in}) = \sum_{i \notin K_n} c_i(1 - t_n y_{in}) + \sum_{j \in K_n} c_j(1 - t_n y_{jn}). \tag{A.18}$$

And by using this equation above, we obtain:

$$\max(0, \sum_{i=1}^{M} c_i(1 - t_n y_{in})) = \sum_{i \notin K_n} c_i(1 - t_n y_{in}) + \sum_{j \in K_n} c_j(1 - t_n y_{jn}). \tag{A.19}$$

Also since $\max(0, 1 - t_n y_{in})$ is always positive, the following equation holds:

$$\sum_{i=1}^{M} c_i \max(0, 1 - t_n y_{in}) = \sum_{i \notin K_n} c_i(1 - t_n y_{in}). \tag{A.20}$$

From equations (4.5), (A.19) and (A.20) we obtain:

$$\begin{aligned} \mathrm{amb}_{HL}(\mathcal{Y}_n) &= \sum_{i \notin K_n} c_i(1 - t_n y_{in}) - \sum_{i \notin K_n} c_i(1 - t_n y_{in}) - \sum_{j \in K_n} c_j(1 - t_n y_{jn}) \\ &= - \sum_{j \in K_n} c_j(1 - t_n y_{jn}). \end{aligned} \tag{A.21}$$

From the definition of the set K_n we obtain:

$$\mathrm{amb}_{HL}(\mathcal{Y}_n) > 0. \tag{A.22}$$

From (A.17) and (A.22) we therefore have that $\mathrm{amb}_{HL}(\mathcal{Y}_n) \geq 0$.

2. We show that if for the pattern x_n we have:
$\forall\, i, j \in \{1 \ldots M\}, y_{in} = y_{jn} \Rightarrow \mathrm{amb}_{HL}(\mathcal{Y}_n) = 0.$

If $\forall\, i, j \in \{1 \ldots M\}, y_{in} = y_{jn} \Rightarrow\ 1 - t_n y_{in} = 1 - t_n y_{jn} = 1 - t_n y_{1n}$. As a result

$$\max\left(0, \sum_{i=1}^{M} c_i(1 - t_n y_{in})\right) = \max\left(0, (1 - t_n y_{1n}) \sum_{i=1}^{M} c_i\right) = \max(0, 1 - t_n y_{1n}) \tag{A.23}$$

and

$$\sum_{i=1}^{M} c_i \max(0, 1 - t_n y_{in}) = \sum_{i=1}^{M} c_i \max(0, 1 - t_n y_{1n}) = \tag{A.24}$$

$$\max(0, 1 - t_n y_{1n}) \sum_{i=1}^{M} c_i = \max(0, 1 - t_n y_{1n}). \tag{A.25}$$

From equations (4.5), (A.23) and (A.25) we obtain

$$\mathrm{amb}_{HL}(\mathcal{Y}_n) = 0. \tag{A.26}$$

3. If $\text{amb}_{HL}(\mathcal{Y}_n) = 0 \Rightarrow \sum_{i=1}^{M} c_i \max(0, 1 - t_n y_{in}) - \max(0, \sum_{i=1}^{M} c_i(1 - t_n y_{in})) = 0 \Rightarrow$

$$\sum_{i=1}^{M} c_i \max(0, 1 - t_n y_{in}) = \max\left(0, \sum_{i=1}^{M} c_i(1 - t_n y_{in})\right). \tag{A.27}$$

We can distinguish two sub-cases:

a) $\max(0, \sum_{i=1}^{M} c_i(1 - t_n y_{in})) = 0$

From (A.27) we get that:

$$\sum_{i=1}^{M} c_i \max(0, 1 - t_n y_{in}) = 0 \Rightarrow \max(0, 1 - t_n y_{in}) = 0 \, \forall i \in \{1 \dots M\}.$$

As a result

$$1 - t_n y_{in} \leq 0 \, \forall i \in \{1 \dots M\}. \tag{A.28}$$

Equation (A.28) implies that $t_n y_{in} > 0 \, \forall i \in \{1 \dots M\}$ and $|y_{in}| \geq 1$. These two conditions are satisfied in two sub-cases

i. $t_n = 1 \Rightarrow y_{in} \geq 1 \, \forall i \in \{1 \dots M\}$, then all classifiers predict class 1.

ii. $t_n = -1 \Rightarrow y_{in} < 0$ and $|y_{in}| \geq 1 \, \forall i \in \{1 \dots M\}$, then all classifiers predict class -1.

b) $\max(0, \sum_{i=1}^{M} c_i(1 - t_n y_{in})) = \sum_{i=1}^{M} c_i(1 - t_n y_{in})$

From (A.27) we get that

$$\sum_{i=1}^{M} c_i \max(0, 1 - t_n y_{in}) = \sum_{i=1}^{M} c_i(1 - t_n y_{in}).$$

Since $\max(0, 1 - t_n y_{in}) \geq 0 \, \forall i \in \{1 \dots M\}$, we obtain:

$$1 - t_n y_{in} \geq 0 \, \forall i \in \{1 \dots M\}. \tag{A.29}$$

This inequality can be verified in the four following sub-cases:

A $\exists j \in \{1 \dots M\}, t_n y_{jn} < 0$ which means that these classifiers do not predict correctly the class.

B $\exists k \in \{1 \dots M\}, 0 < y_{kn} < 1$ and $t_n = 1$ in which case all these classifiers identify correctly the class.

C $\exists l \in \{1 \dots M\}, -1 < y_{ln} < 0$ and $t_n = -1$ in which case all these classifiers identify correctly the class.

D $\exists q \in \{1 \dots M\}, t_n = y_{qn}$, for the case when $1 - t_n y_{in} = 0$. In this case all these classifiers identify correctly the class.

Since cases (A, B, D) or (A, C, D) can be true at the same time and still the

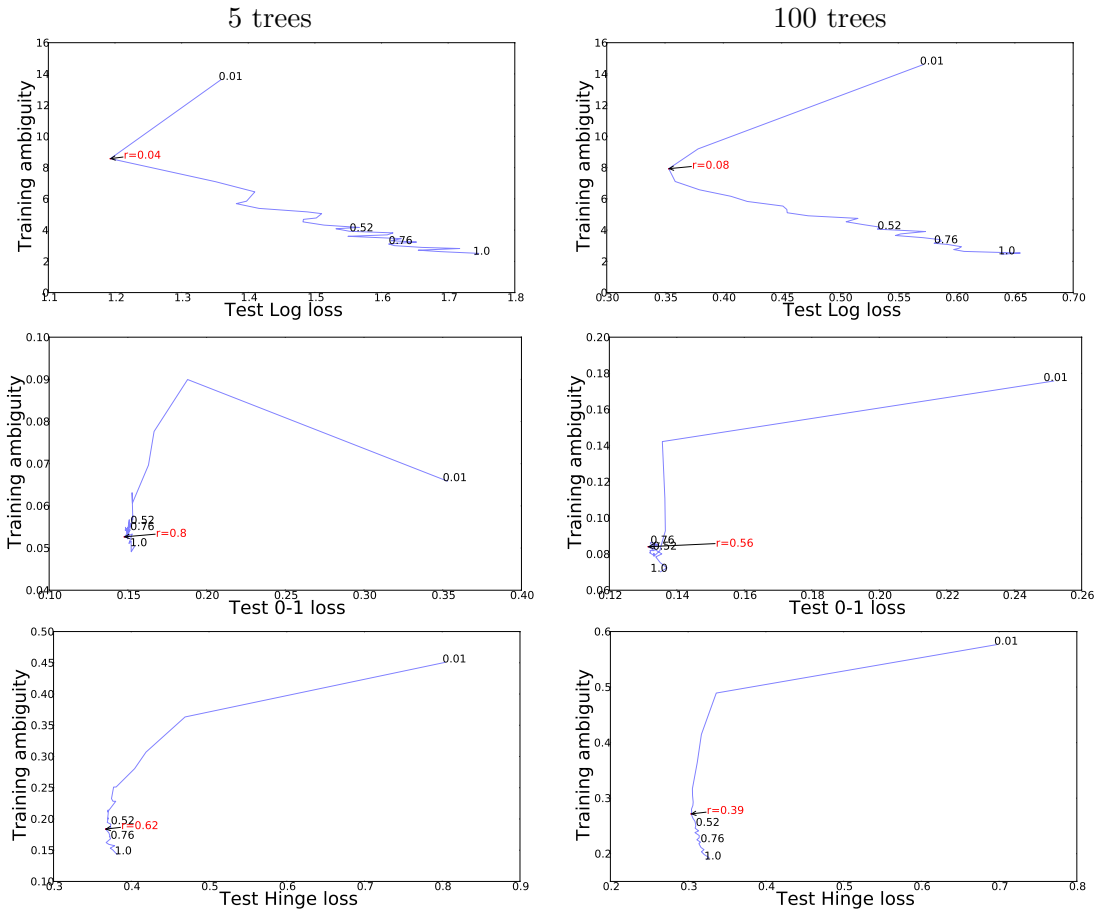

Figure A.4. Curves of the three types of ambiguities versus the corresponding losses that were derived from these. The test error versus the training ambiguity was plotted for different sampling rates for ensembles formed of 5 trees (left column) and 100 tress (right column) for the Australian dataset. The first row shows the behaviour of the test cross entropy versus the training cross entropy ambiguity, in the second row the test 0-1 loss versus its corresponding training ambiguity is plotted, respectively the behaviour of the hinge loss is presented in the third row of panels. The optimal sampling rate is indicated in red.

condition (A.29) to be held, we cannot draw the conclusion that $\text{amb}_{HL}(\mathcal{Y}_n) = 0 \Rightarrow$ all the classifiers predict the same class.

□

A.3 Random splits

This section presents the plots of the variation of the three types of ambiguities versus the corresponding tests errors from Chapter 4, for all the databases investigated. The plots are displayed in Figures A.4 - A.7.

A.4 Critical diagrams

This section presents the critical diagrams obtained in Chapter 5 for all the databases investigated. The plots are displayed in Figures A.8 - A.35.

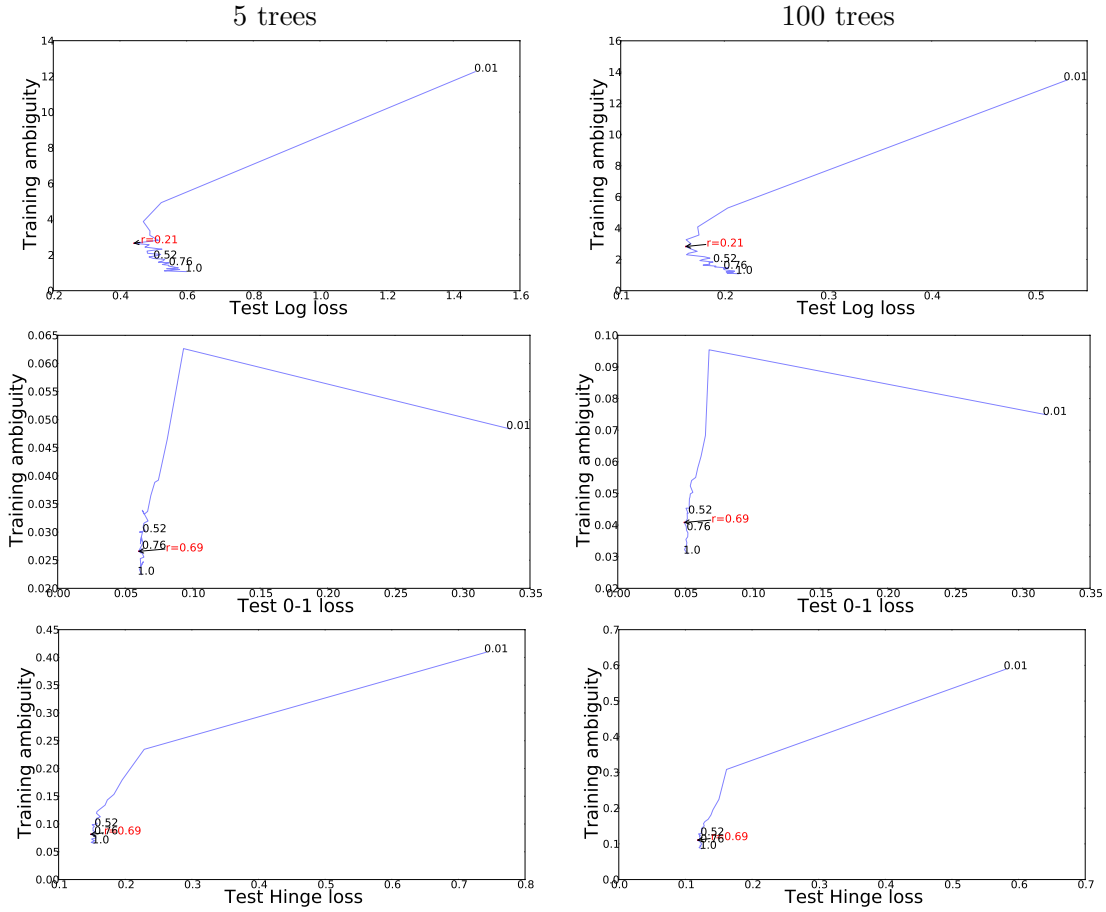

Figure A.5. Curves of the three types of ambiguities versus the corresponding losses that were derived from these. The test error versus the training ambiguity was plotted for different sampling rates for ensembles formed of 5 trees (left column) and 100 tress (right column) for the Cancer dataset. The first row shows the behaviour of the test cross entropy versus the training cross entropy ambiguity, in the second row the test 0-1 loss versus its corresponding training ambiguity is plotted, respectively the behaviour of the hinge loss is presented in the third row of panels. The optimal sampling rate is indicated in red.

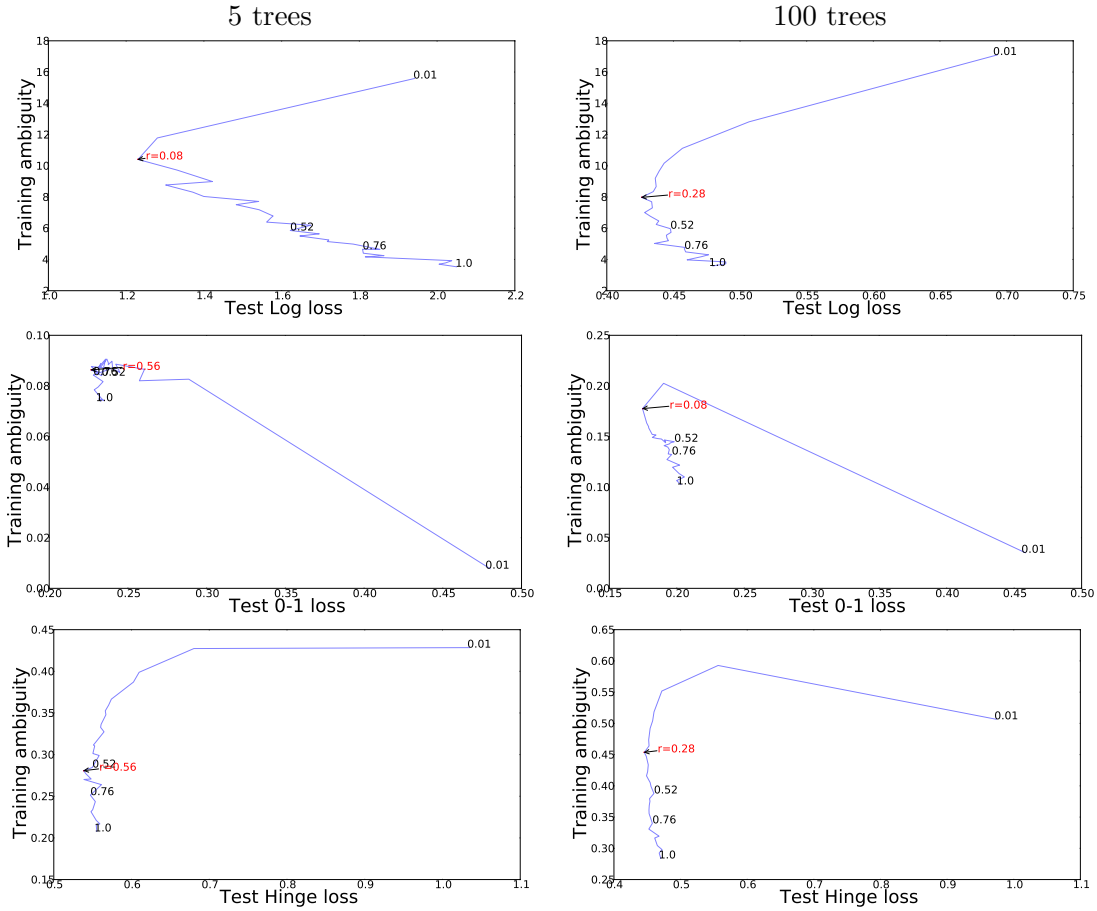

Figure A.6. Curves of the three types of ambiguities versus the corresponding losses that were derived from these. The test error versus the training ambiguity was plotted for different sampling rates for ensembles formed of 5 trees (left column) and 100 tress (right column) for the Heart dataset. The first row shows the behaviour of the test cross entropy versus the training cross entropy ambiguity, in the second row the test 0-1 loss versus its corresponding training ambiguity is plotted, respectively the behaviour of the hinge loss is presented in the third row of panels. The optimal sampling rate is indicated in red.

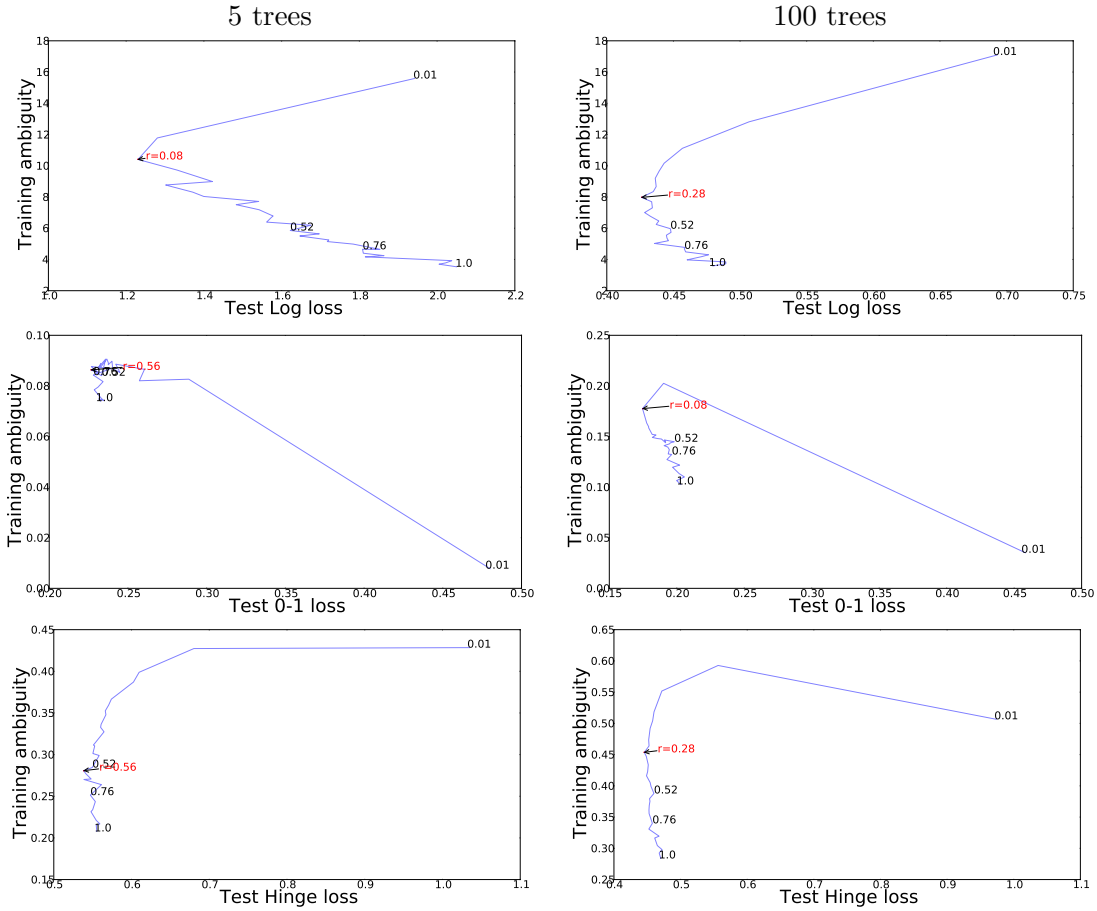

Figure A.7. Curves of the three types of ambiguities versus the corresponding losses that were derived from these. The test error versus the training ambiguity was plotted for different sampling rates for ensembles formed of 5 trees (left column) and 100 tress (right column) for the Ionosphere dataset. The first row shows the behaviour of the test cross entropy versus the training cross entropy ambiguity, in the second row the test 0-1 loss versus its corresponding training ambiguity is plotted, respectively the behaviour of the hinge loss is presented in the third row of panels. The optimal sampling rate is indicated in red.

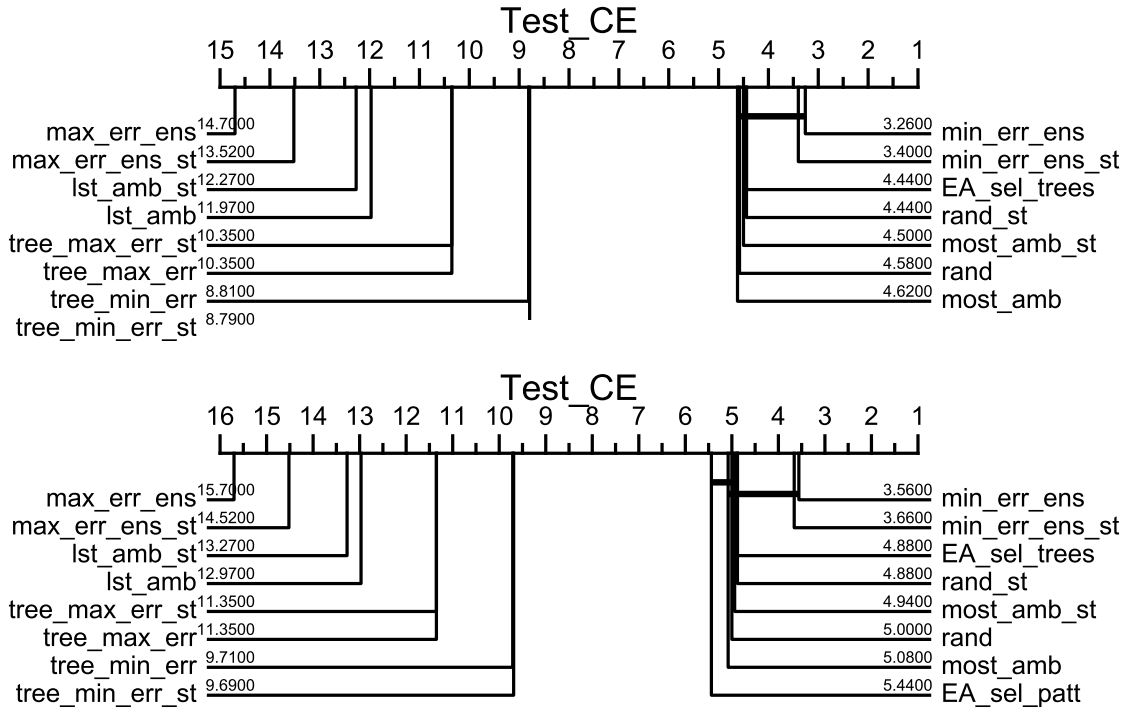

Figure A.8. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the Australian dataset for the 0.05 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

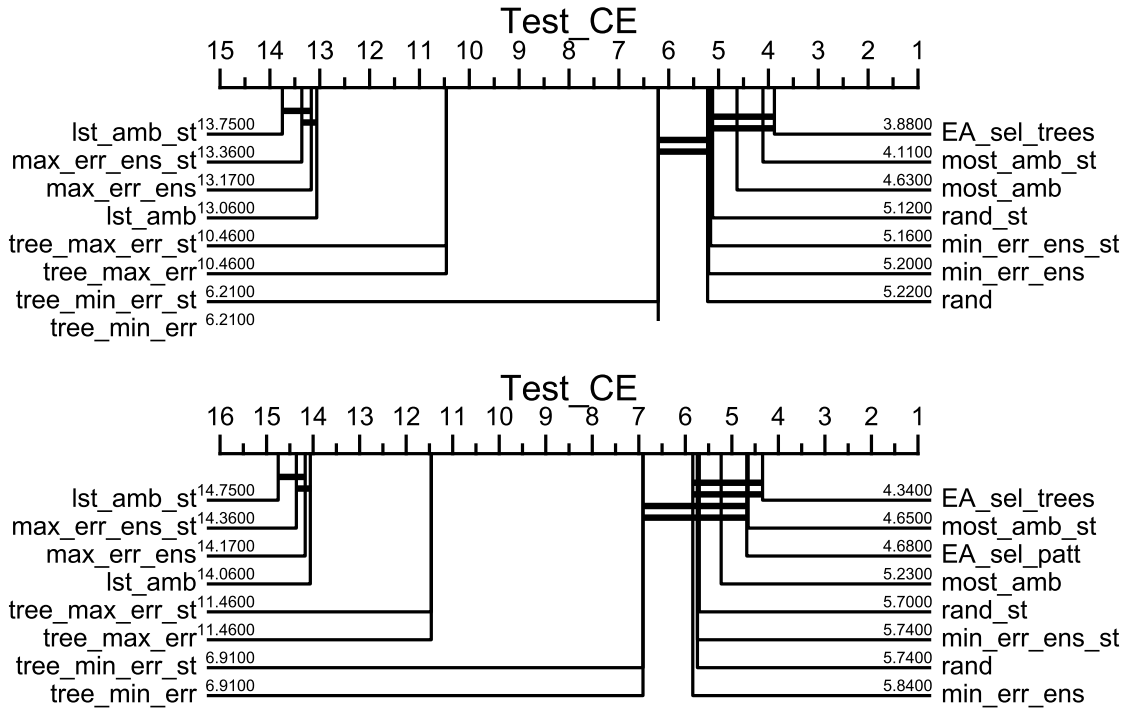

Figure A.9. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the Australian dataset for the 0.1 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

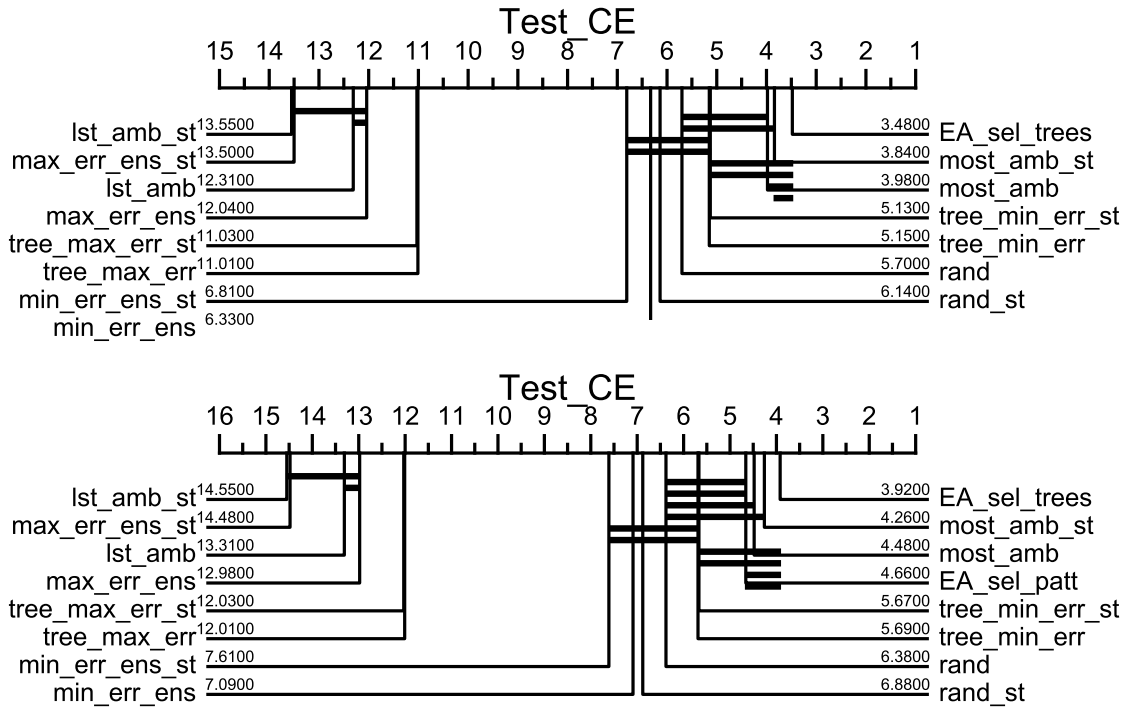

Figure A.10. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the Australian dataset for the 0.2 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

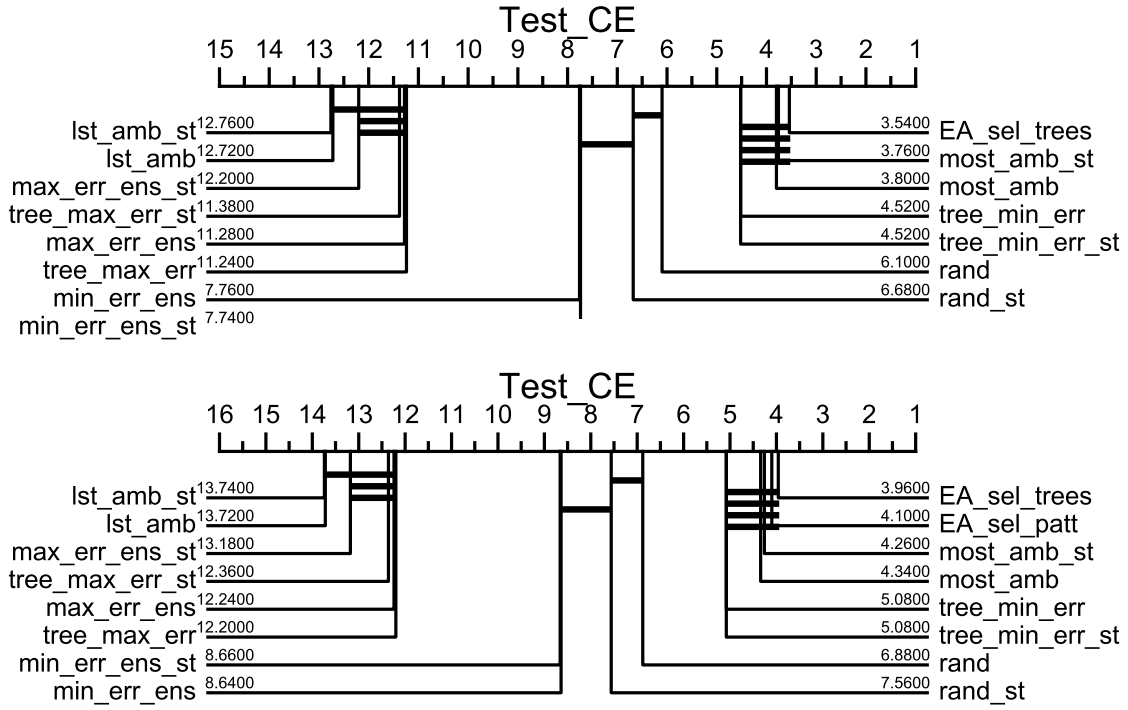

Figure A.11. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the Australian dataset for the 0.3 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

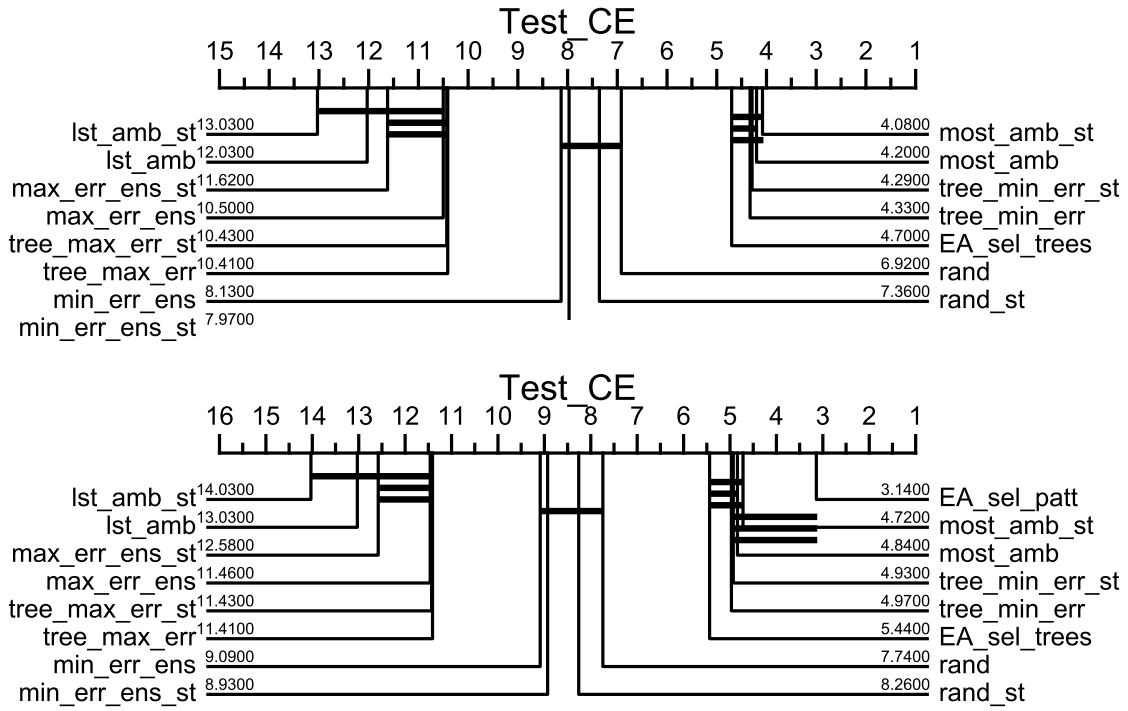

Figure A.12. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the Australian dataset for the 0.4 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

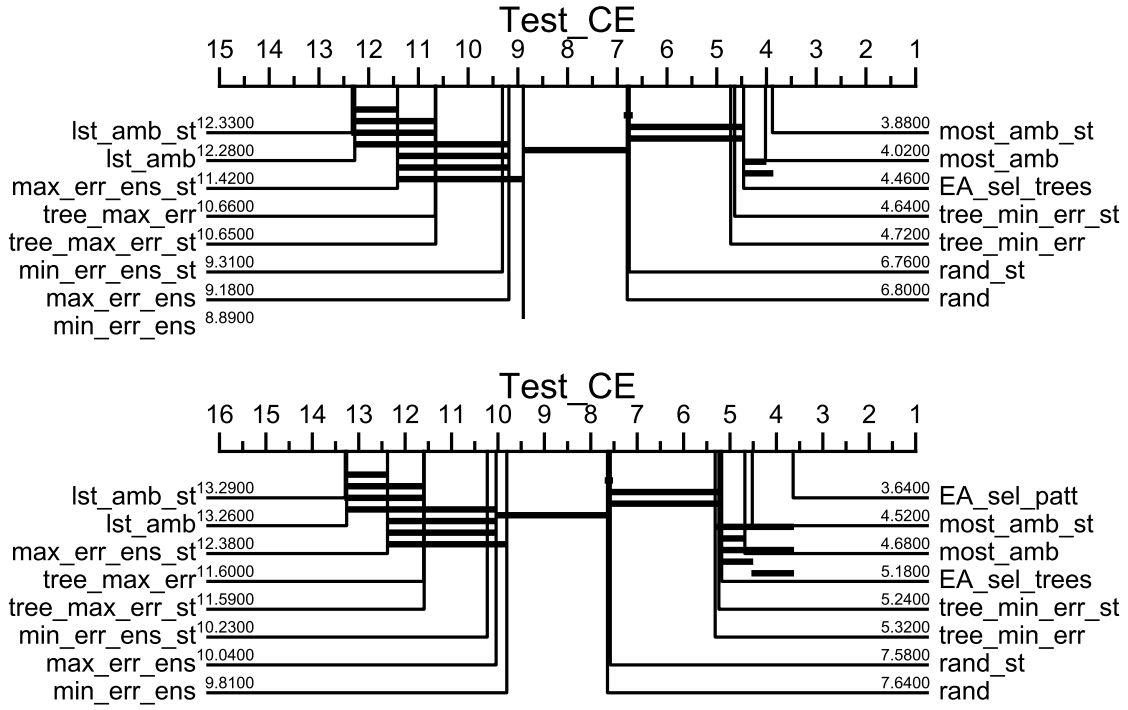

Figure A.13. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the Australian dataset for the 0.5 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

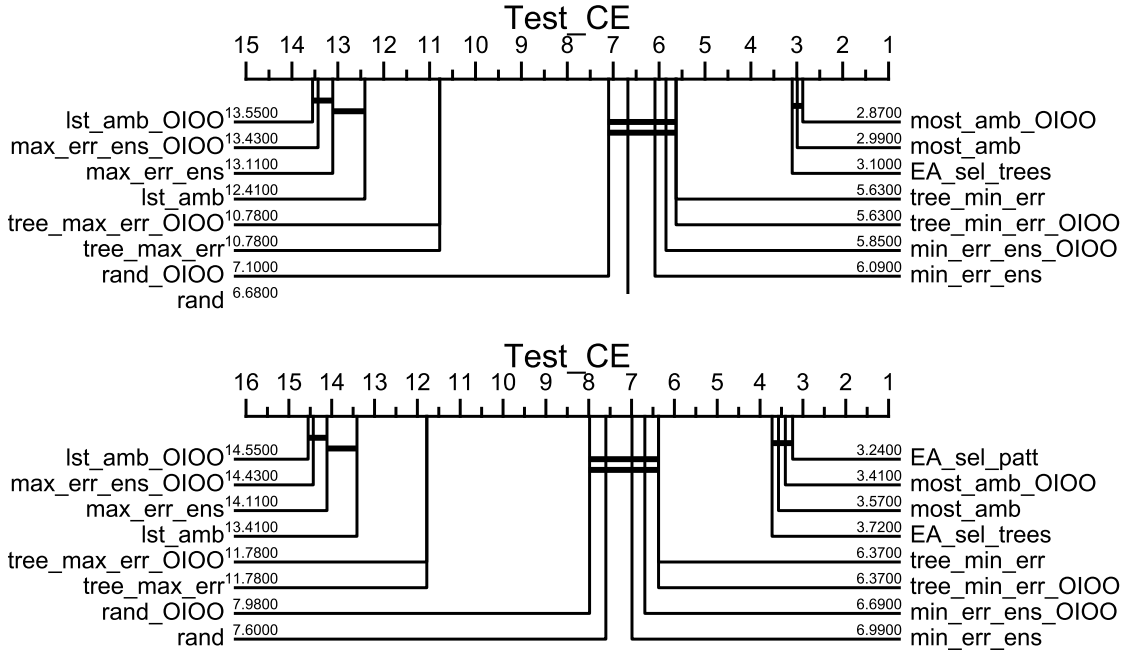

Figure A.14. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the German dataset for the 0.1 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

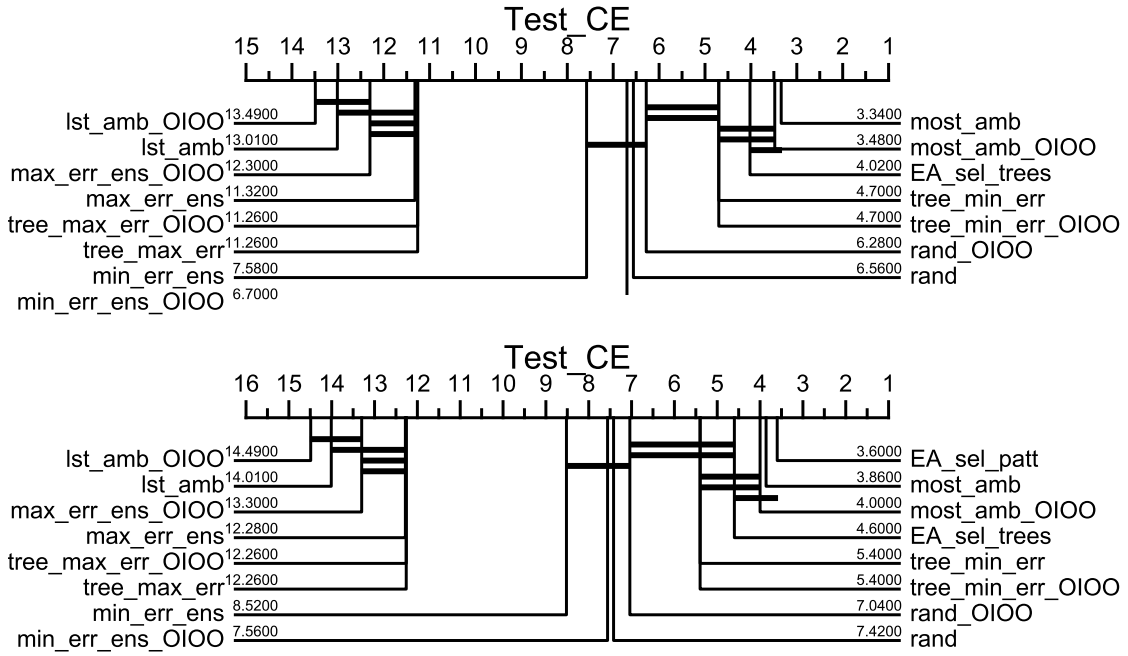

Figure A.15. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the German dataset for the 0.2 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

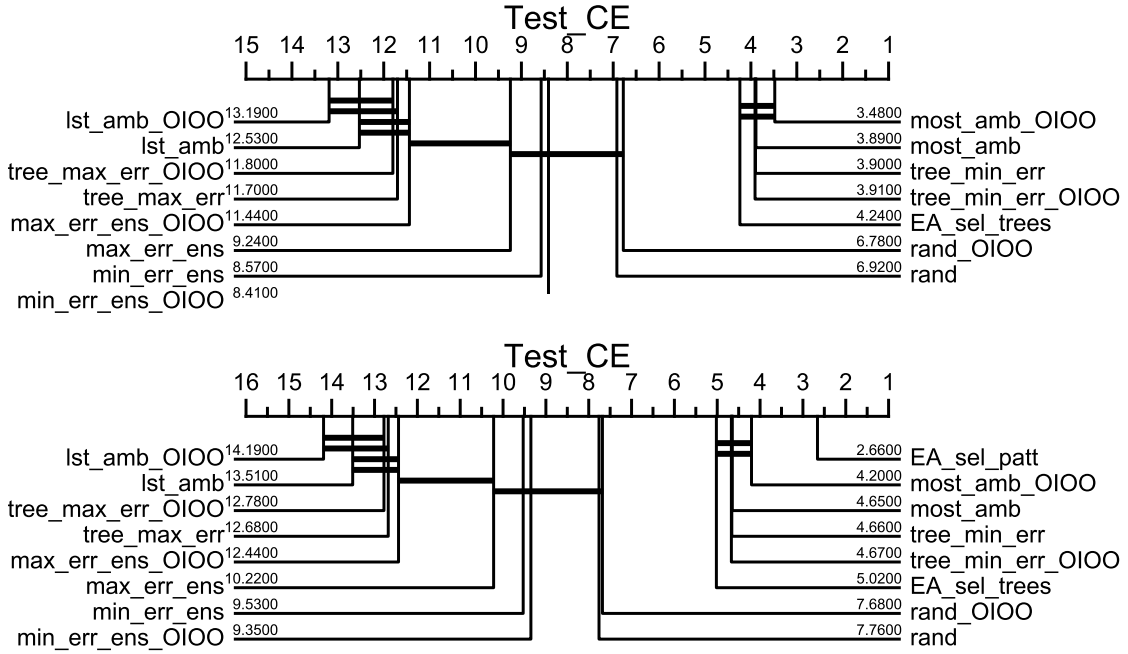

Figure A.16. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the German dataset for the 0.3 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

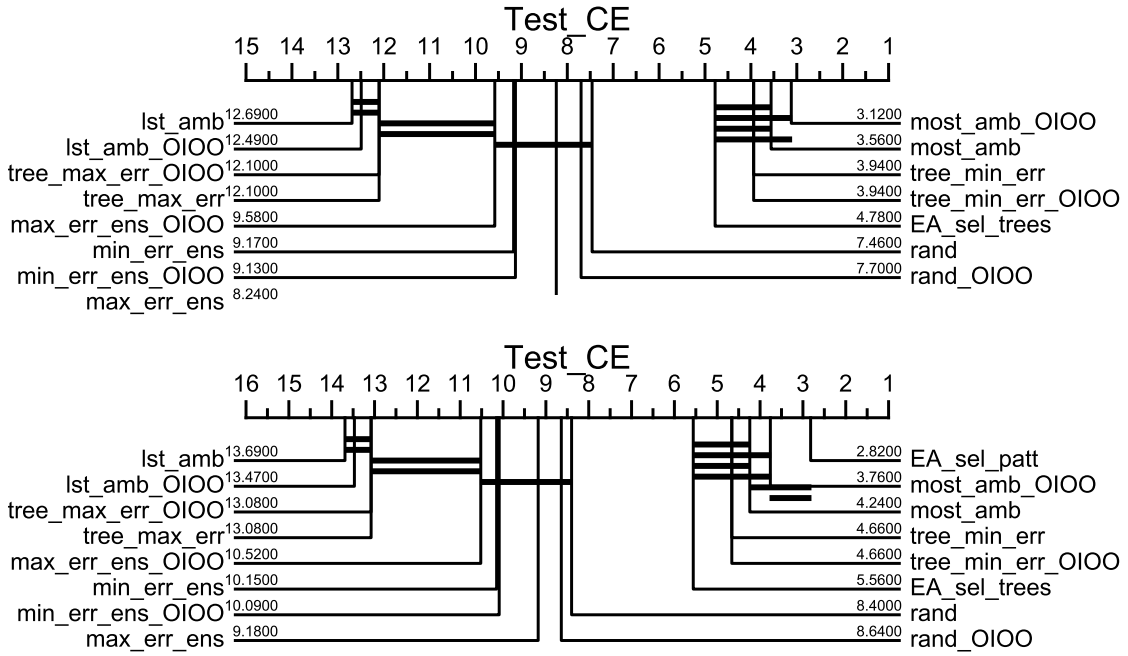

Figure A.17. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the German dataset for the 0.4 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

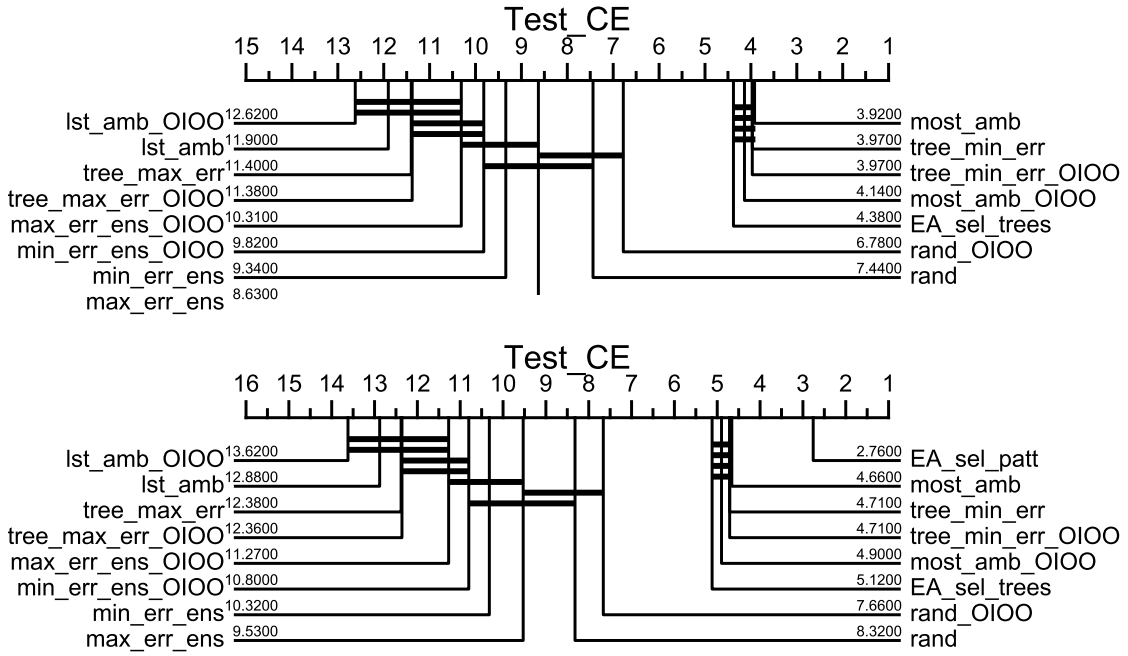

Figure A.18. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the German dataset for the 0.5 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

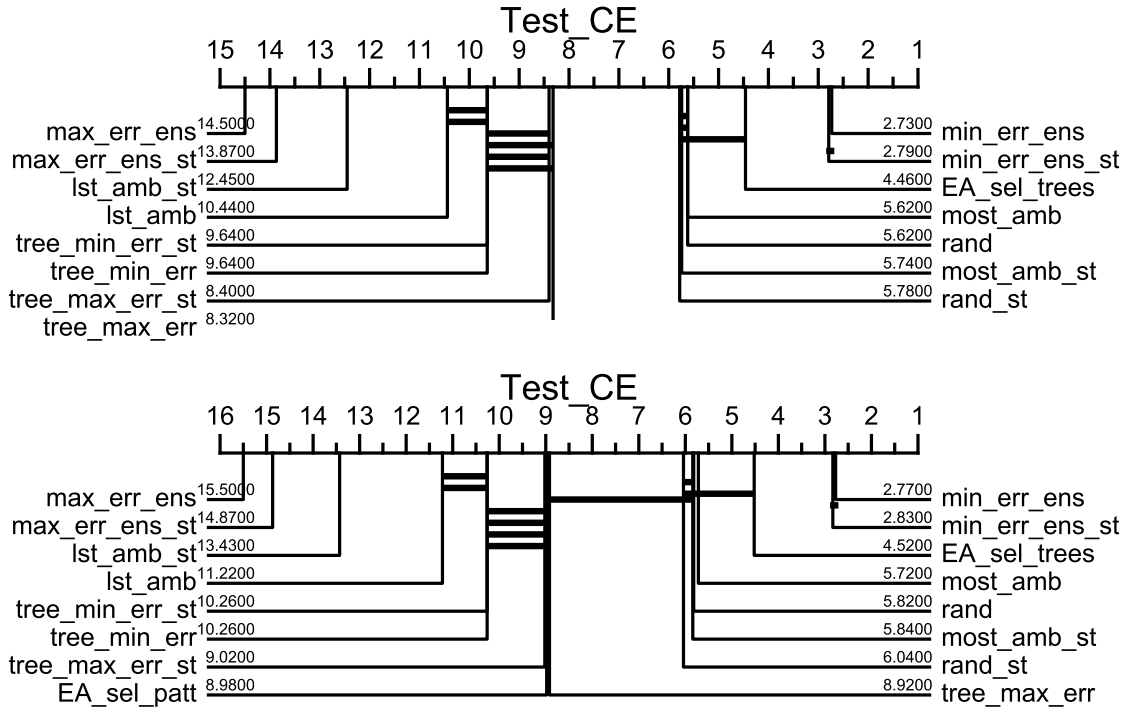

Figure A.19. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the Cancer dataset for the 0.05 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

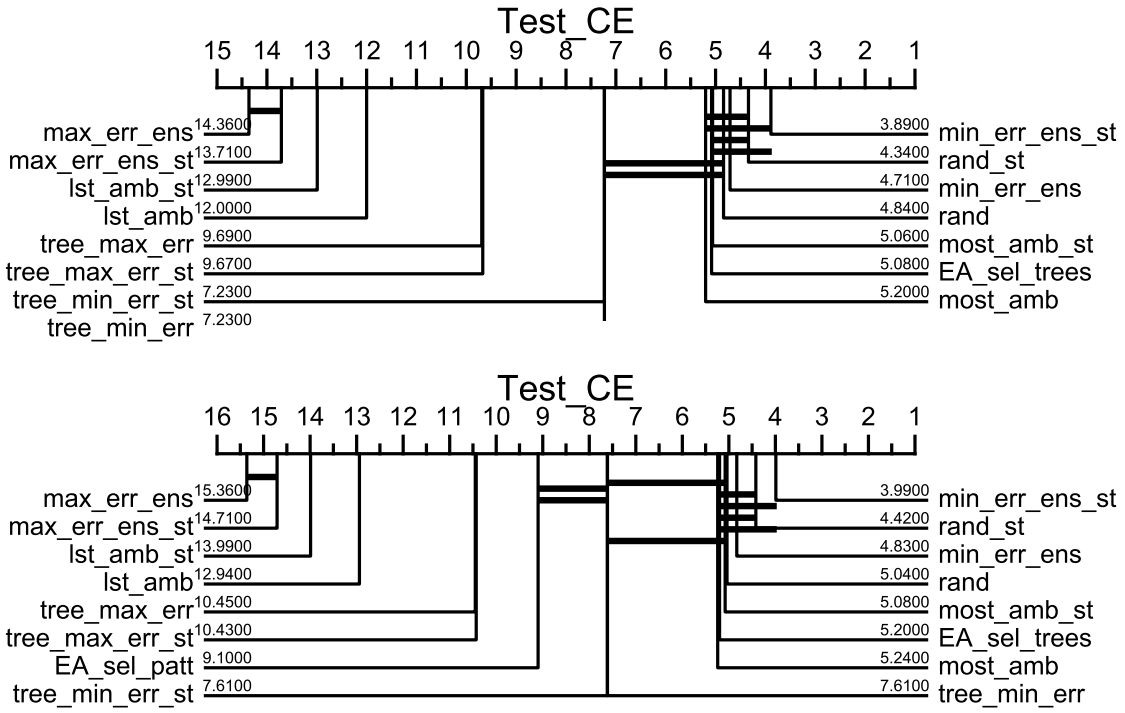

Figure A.20. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the Cancer dataset for the 0.1 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

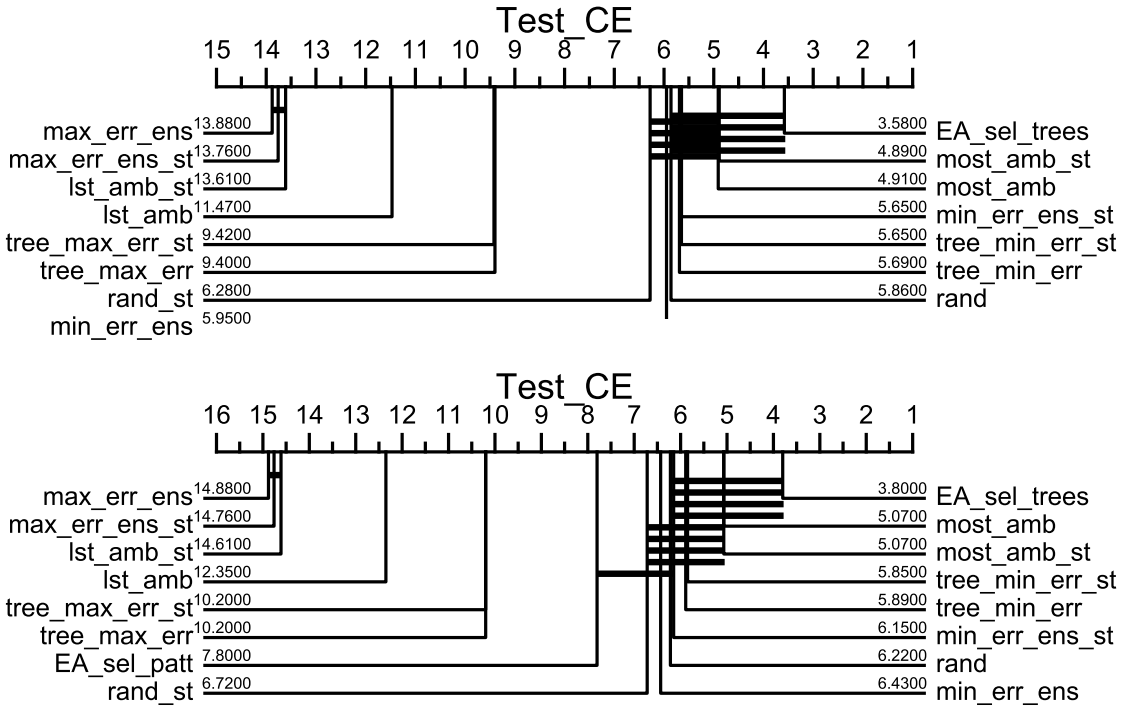

Figure A.21. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the Cancer dataset for the 0.2 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

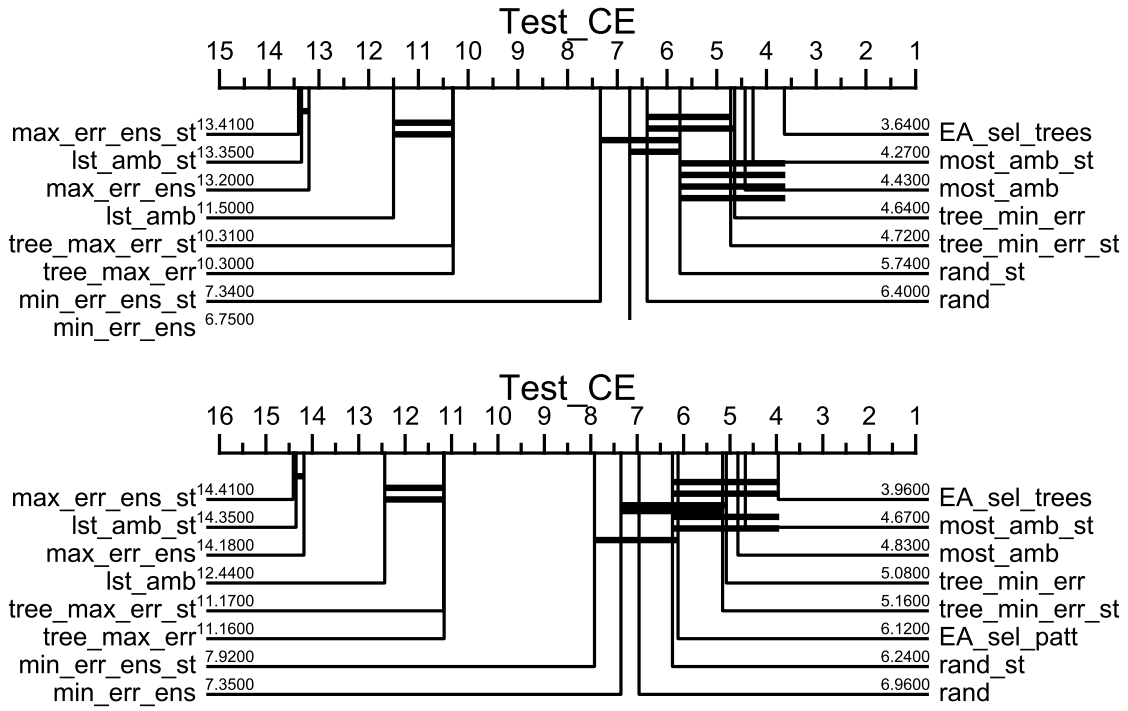

Figure A.22. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the Cancer dataset for the 0.3 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

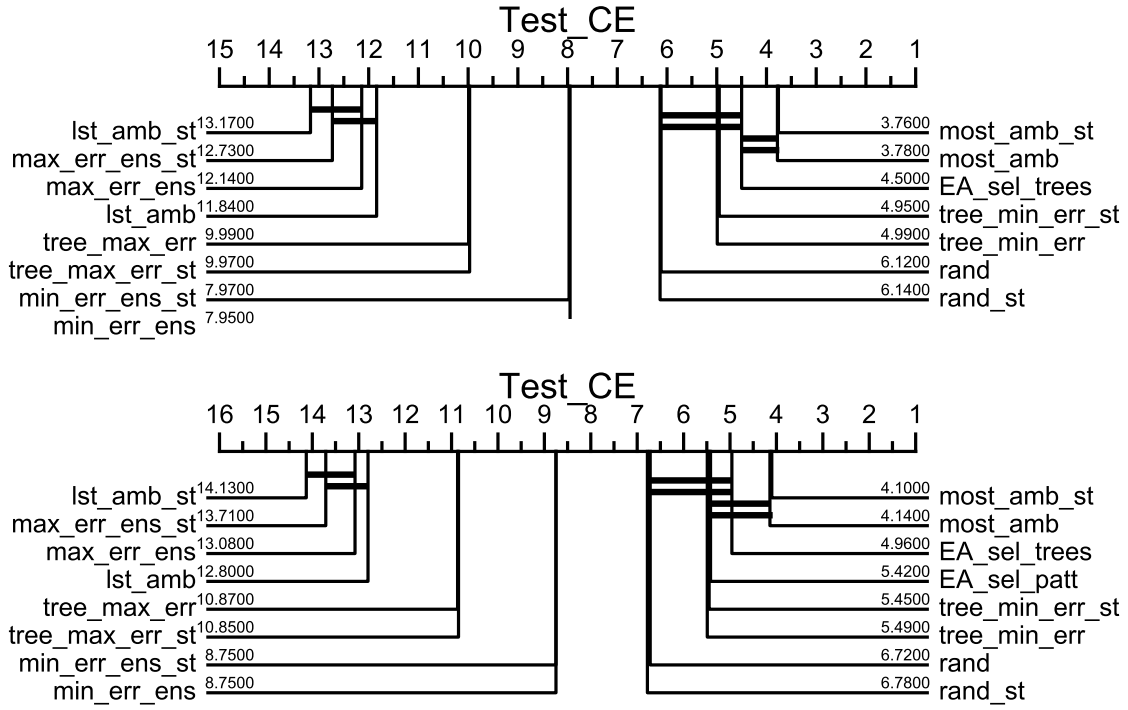

Figure A.23. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the Cancer dataset for the 0.4 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

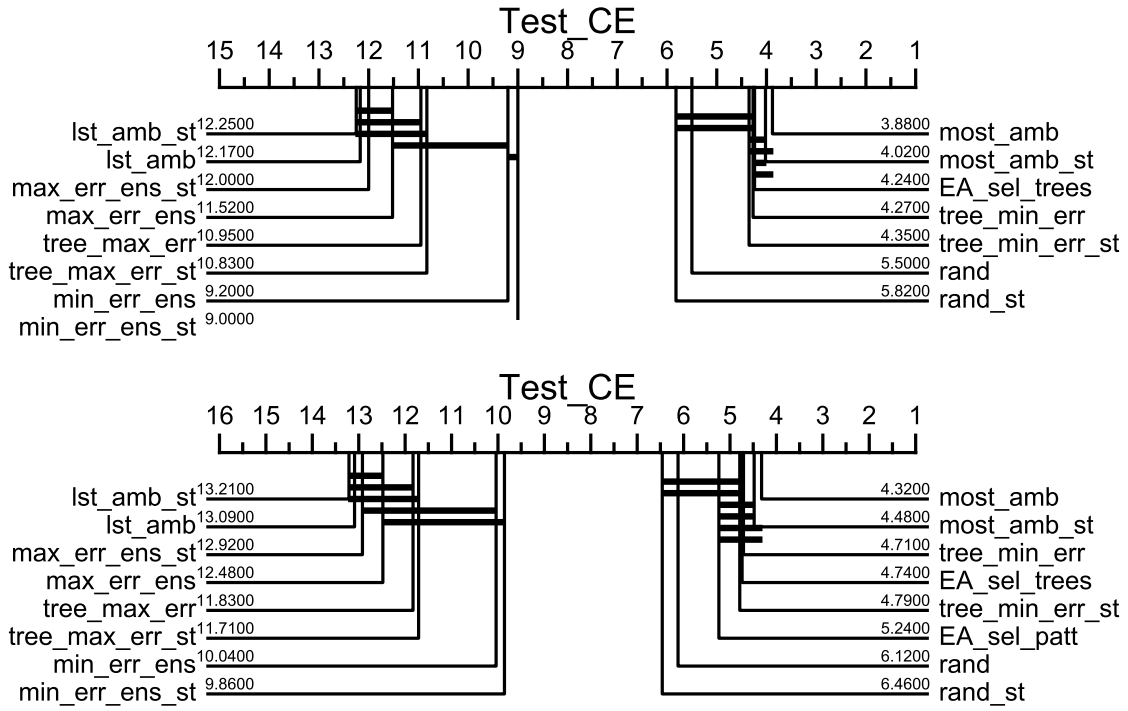

Figure A.24. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the Cancer dataset for the 0.5 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

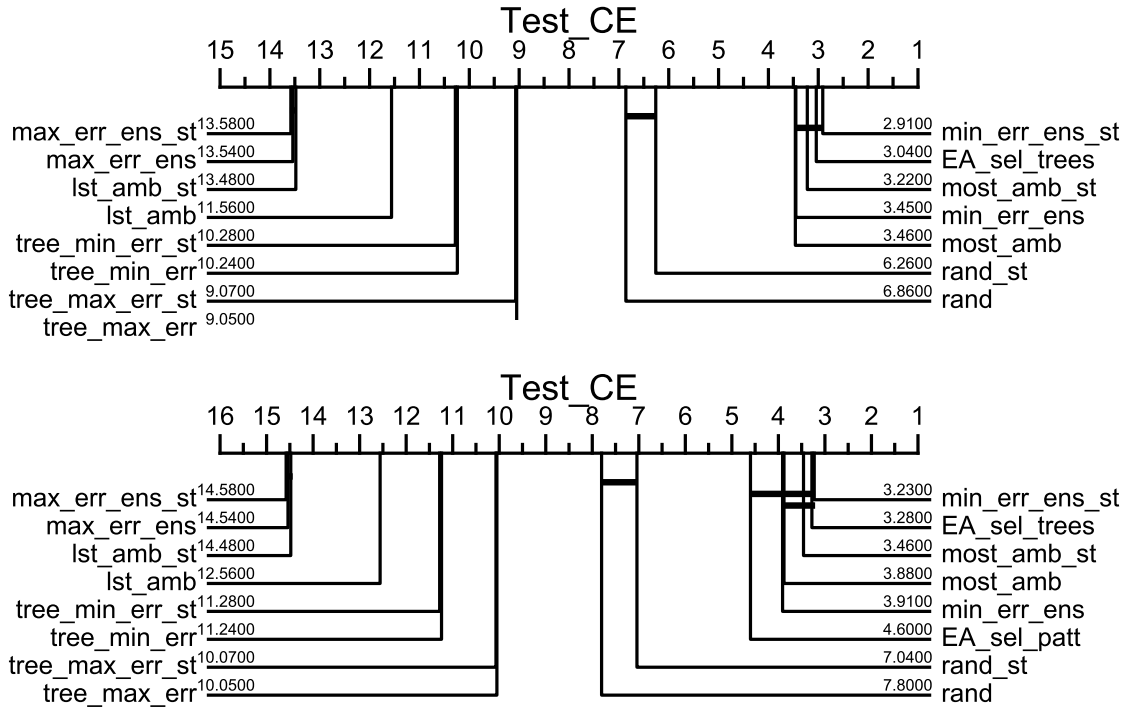

Figure A.25. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the Liver dataset for the 0.05 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

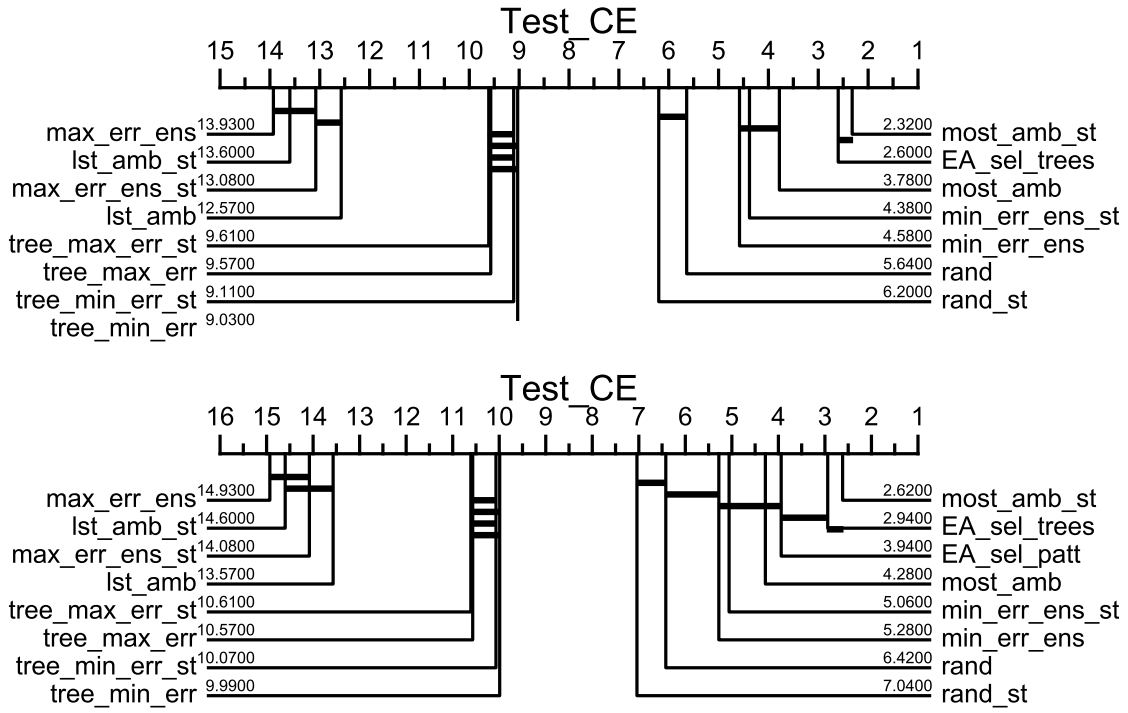

Figure A.26. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the Liver dataset for the 0.1 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

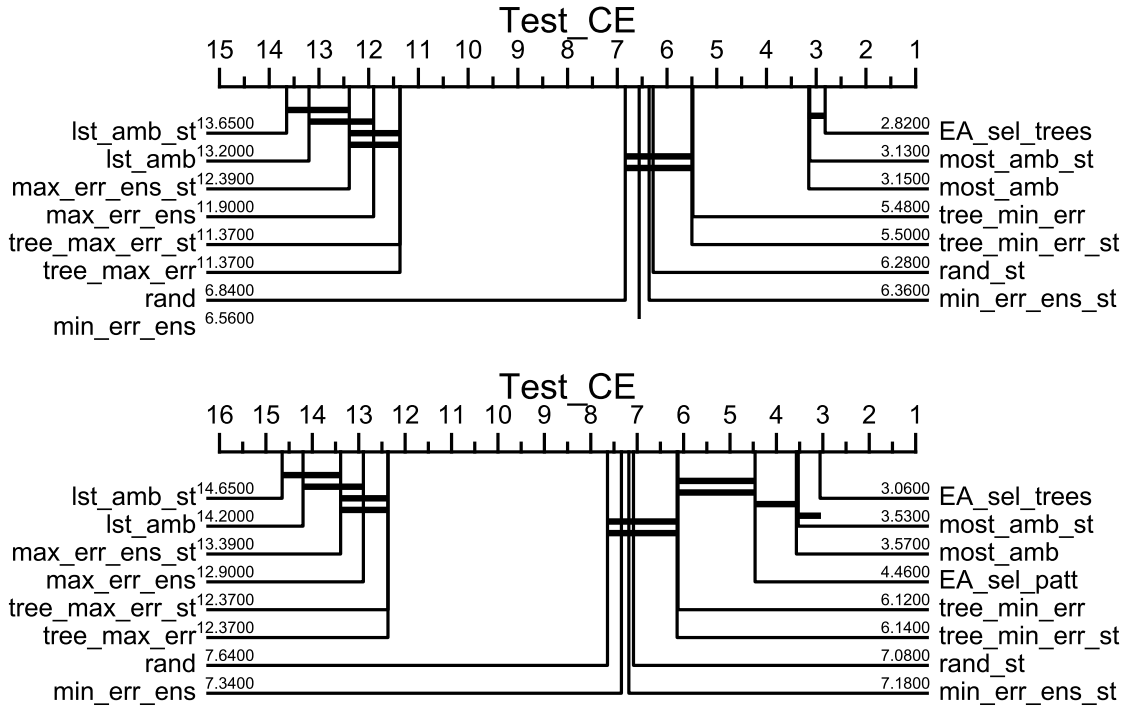

Figure A.27. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the Liver dataset for the 0.2 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

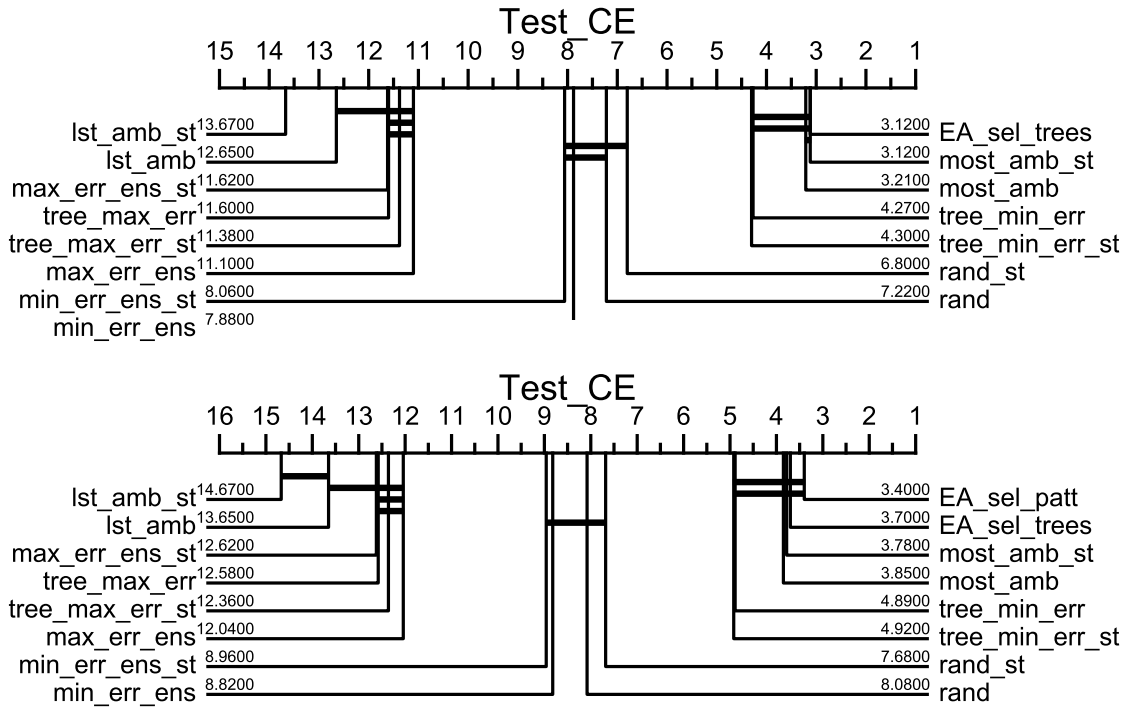

Figure A.28. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the Liver dataset for the 0.3 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

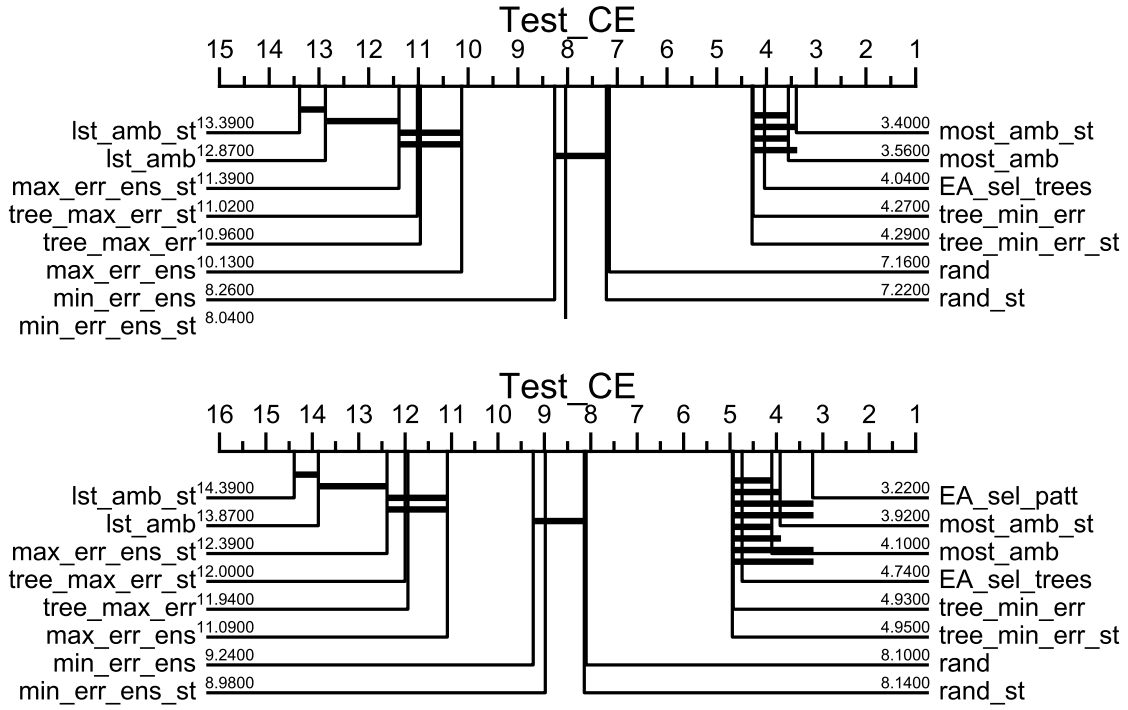

Figure A.29. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the Liver dataset for the 0.4 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

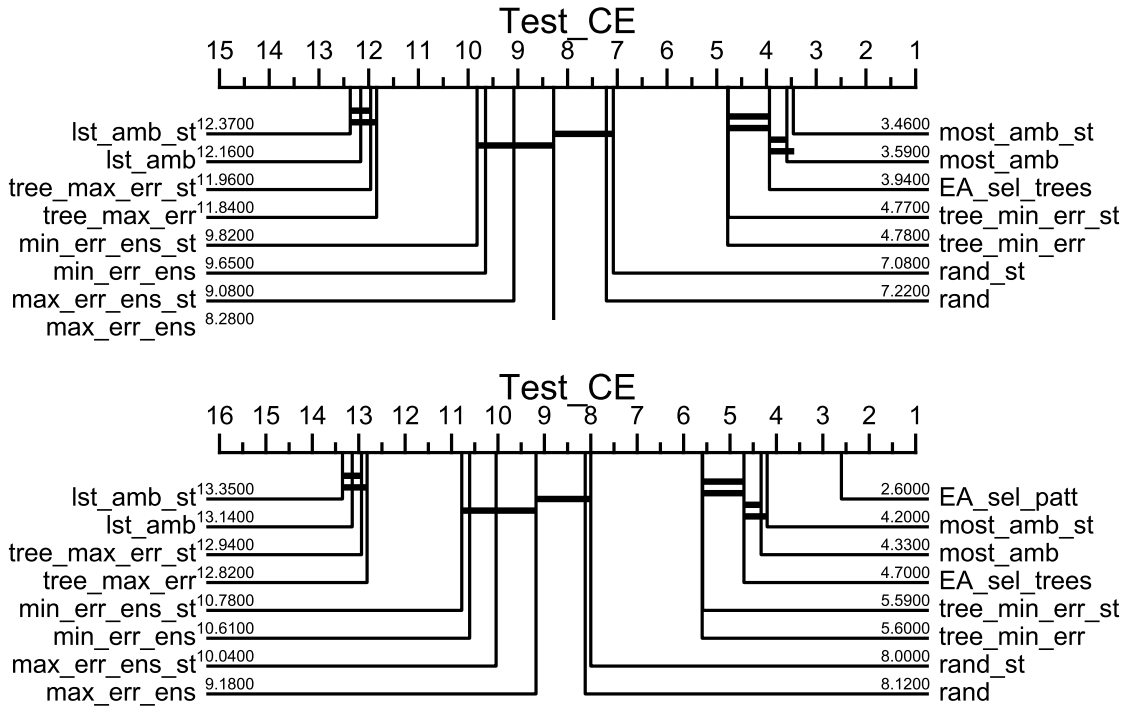

Figure A.30. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the Liver dataset for the 0.5 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

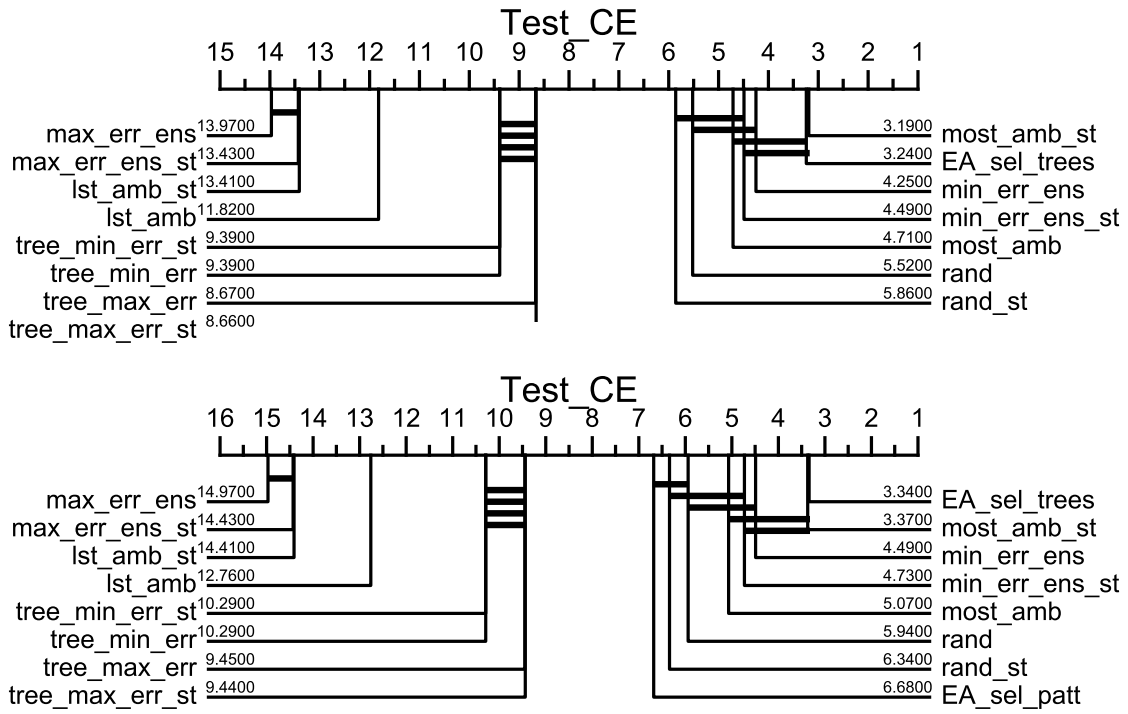

Figure A.31. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the Sonar dataset for the 0.1 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

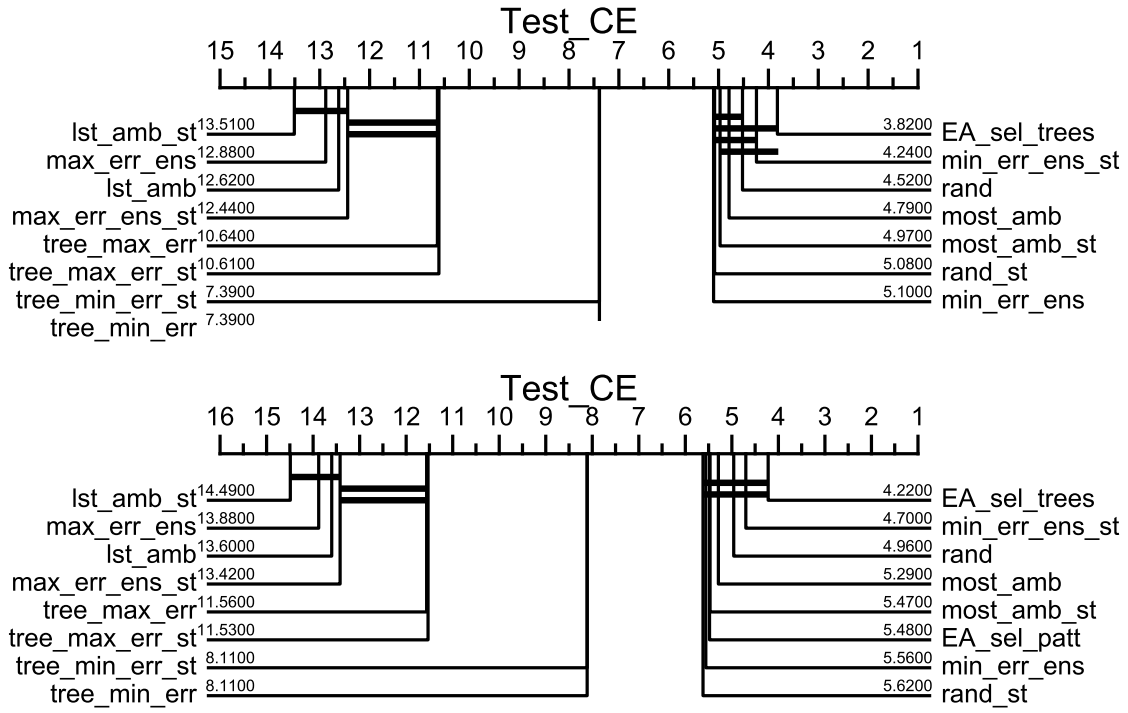

Figure A.32. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the Sonar dataset for the 0.2 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

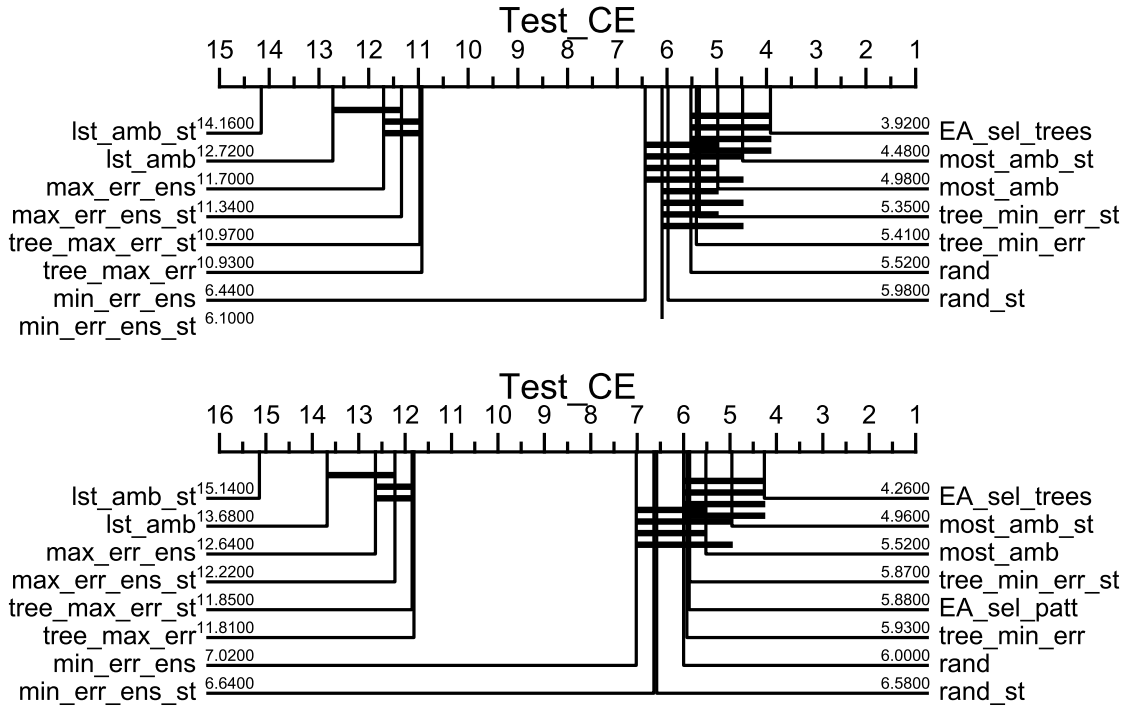

Figure A.33. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the Sonar dataset for the 0.3 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

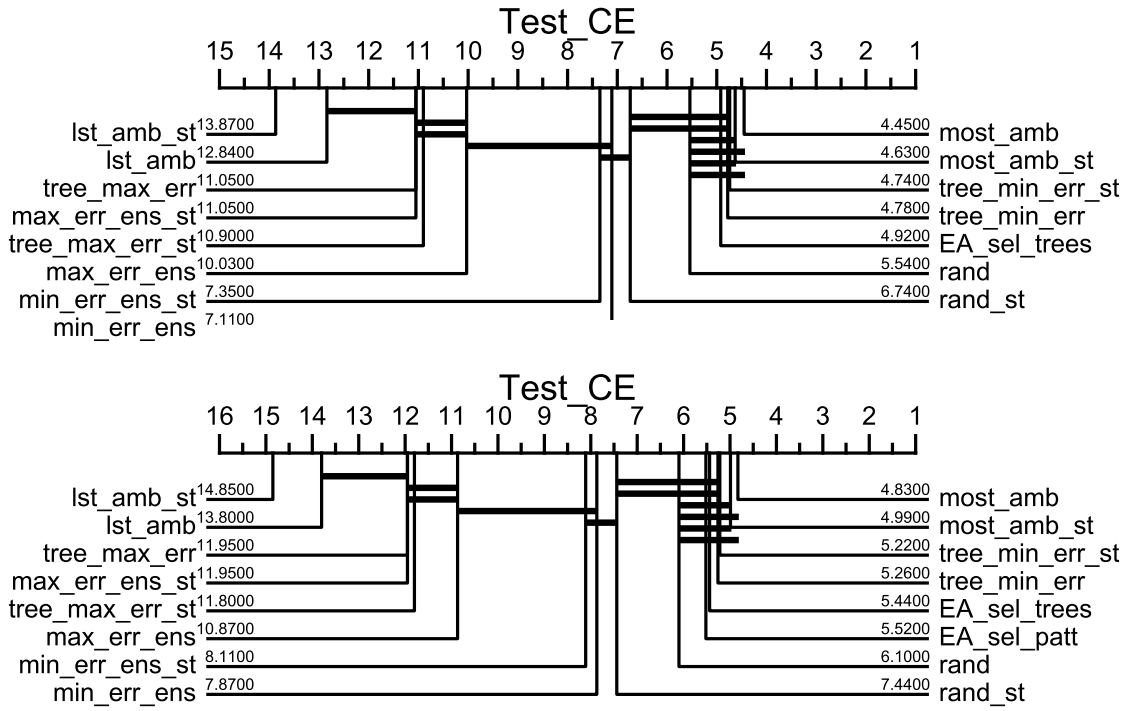

Figure A.34. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the Sonar dataset for the 0.4 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

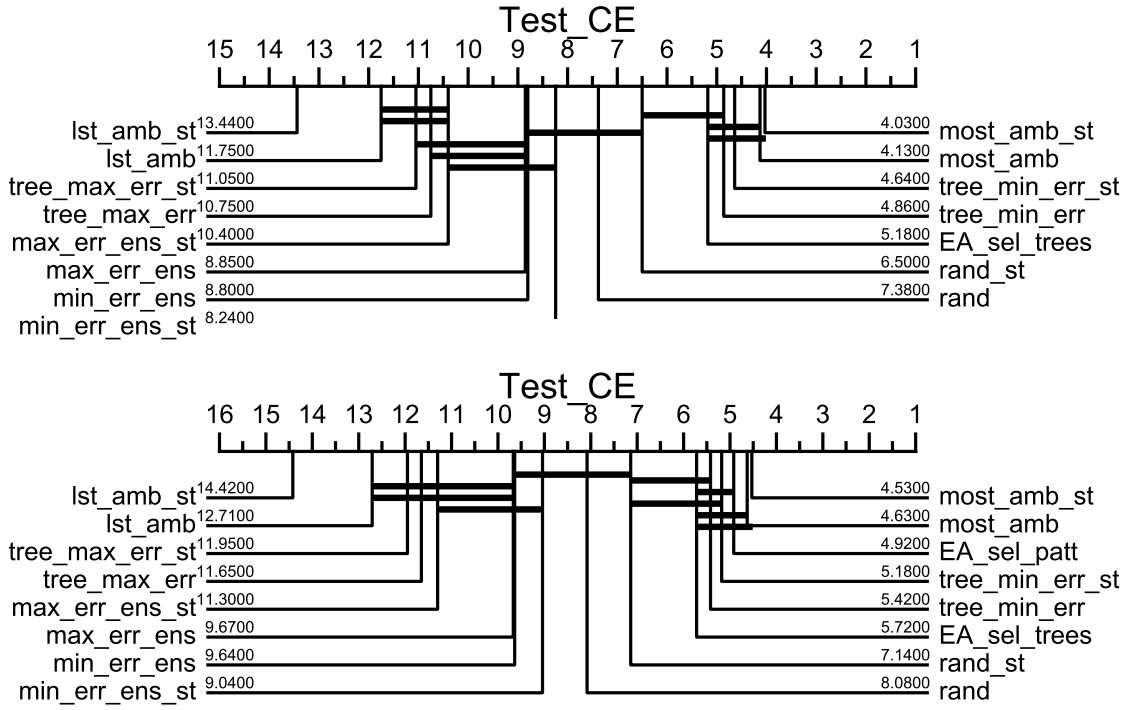

Figure A.35. Comparisons of the tree selection approaches versus the pattern selection approach (the evolutionary algorithm from Section 4.4) on the Sonar dataset for the 0.5 sampling rate, for 5 trees. The top plot shows the ranking and the statistical similarities for all the approaches except for the evolutionary algorithm from Section 4.4. The bottom plot analyses all the approaches.

Bibliography

[1] Chen H. *Diversity and Regularization in Neural Network Ensembles.* PhD thesis, University of Birmingham, 2008.

[2] Y. Jin and B. Sendhoff. Pareto-based multiobjective machine learning: An overview and case studies. *IEEE Transactions on Systems, MAN, AND CYBERNETICS-PART C: APPLICATIONS AND REVIEW*, 38:397–415, 2008.

[3] Bishop C. *Pattern Recognition and Machine Learning.* Springer, 2006.

[4] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528. IEEE, 2011.

[5] Sashikanta Prusty, Srikanta Patnaik, and Sujit Kumar Dash. Skcv: Stratified k-fold cross-validation on ml classifiers for predicting cervical cancer. *Frontiers in Nanotechnology*, 4, 2022.

[6] Duda R. O., Hart P. E., and Stork D. G. *Pattern Classification.* Wiley, New York, 2 edition, 2001.

[7] Swarnalatha Purushotham and BK Tripathy. Evaluation of classifier models using stratified tenfold cross validation techniques. In *International conference on computing and communication systems*, pages 680–690. Springer, 2011.

[8] N Vapnik Vladimir and Vlamimir Vapnik. Statistical learning theory. *Xu JH and Zhang XG. translation. Beijing: Publishing House of Electronics Industry, 2O04*, 1998.

[9] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001, 1990.

[10] Breiman L. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[11] Freund Y. and Schapire R. E. A Short Introduction to Boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999.

[12] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[13] Yong Liu and Xin Yao. Negatively correlated neural networks can produce

best ensembles. *Australian journal of intelligent information processing systems*, 4(3/4):176–185, 1997.

[14] Yanjun Qi. Random forest for bioinformatics. *Ensemble machine learning: Methods and applications*, pages 307–323, 2012.

[15] Sofia Benbelkacem and Baghdad Atmani. Random forests for diabetes diagnosis. In *2019 International Conference on Computer and Information Sciences (ICCIS)*, pages 1–4, 2019.

[16] Zhongheng Zhang, Yiming Zhao, Aran Canes, Dan Steinberg, Olga Lyashevska, et al. Predictive analytics with gradient boosting in clinical medicine. *Annals of translational medicine*, 7(7), 2019.

[17] HIND DAORI, Manar Alharthi, ALANOUD ALANAZI, GHAIDA ALZAHRANI, MAJED ABOROKBAH, and Nojood Aljehane. Predicting stock prices using the random forest classifier, 11 2022.

[18] Yan Wang and Yuankai Guo. Forecasting method of stock market volatility in time series data based on mixed model of arima and xgboost. *China Communications*, 17(3):205–221, 2020.

[19] Kevin J Cherkauer. Human expert-level performance on a scientific image analysis task by a system using combined artificial neural networks. In *Working notes of the AAAI workshop on integrating multiple learned models*, volume 21. Citeseer, 1996.

[20] Lars Kai Hansen, Christian Liisberg, and Peter Salamon. Ensemble methods for handwritten digit recognition. In *Neural Networks for Signal Processing II Proceedings of the 1992 IEEE Workshop*, pages 333–342. IEEE, 1992.

[21] Krogh A. and Vedelsby J. Neural network ensembles, cross validation and active learning. *Neural Information Processing Systems*, 7:231–238, 1995.

[22] Tang E. K., Suganthan P. N., and Yao X. An analysis of diversity measures. *Machine Learning*, 65:247–271, 2006.

[23] Kuncheva L. and Whitaker C. J. Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy. *Machine Learning*, 51:181–207, 2003.

[24] Derek Partridge and Wojtek Krzanowski. Software diversity: practical statistics for its measurement and exploitation. *Information and software technology*, 39(10):707–717, 1997.

[25] David B Skalak et al. The sources of increased accuracy for two proposed boosting algorithms. In *Proc. American Association for Artificial Intelligence, AAAI-96, Integrating Multiple Learned Models Workshop*, volume 1129, page 1133. Citeseer, 1996.

[26] Ron Kohavi and David Wolpert. Bias plus variance decomposition for zero-one loss functions. 09 1997.

[27] Richard Everson. Machine learning course.

[28] A.Chandra and X.Yao. Multi-objective ensemble construction, learning and evolution. *Proc. PPSN Workshop Multi-objective Problem Solving from Nature (part of the 9th International Conference on Parallel Problem Solving from Nature: PPSN-IX). Citeseer*, pages 9–13, 2006.

[29] Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

[30] Jason Brownlee. Gentle Introduction to the Bias-Variance Trade-Off in Machine Learning. `https://machinelearningmastery.com/gentle-introduction-to-the-bias-variance-trade-off-in-machine-learning/`.

[31] Ron Kohavi and David Wolpert. Bias plus variance decomposition for zero-one loss functions. 09 1997.

[32] Gareth James. Variance and bias for general loss functions. *Machine Learning*, 51:115–135, 05 2003.

[33] David Pfau. A generalized bias-variance decomposition for bregman divergences. *Unpublished Manuscript*, 2013.

[34] Wikipedia. Boosting (machine learning). `https://en.wikipedia.org/wiki/Boosting_(machine_learning)`.

[35] Wikipedia. Bootstrap aggregating. `https://en.wikipedia.org/wiki/Bootstrap_aggregating`.

[36] Dragos D. Margineantu and Thomas G. Dietterich. Pruning adaptive boosting. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, page 211–218, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[37] Yong Liu and Xin Yao. Ensemble learning via negative correlation. *Neural networks*, 12(10):1399–1404, 1999.

[38] Gavin Brown, Jeremy L Wyatt, Peter Tino, and Yoshua Bengio. Managing diversity in regression ensembles. *Journal of machine learning research*, 6(9), 2005.

[39] Wikipedia. Random forest. `https://en.wikipedia.org/wiki/Random_forest`.

[40] A. Criminisi, J. Shotton, and E. Konukoglu. Decision forests for classification, regression, density estimation, manifold learning and semi-supervised learning. *Microsoft Research technical report TR-2011-114*, 2011.

[41] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1:81–106, 1986.

[42] Corrado Gini. *Variabilità e mutabilità: contributo allo studio delle distribuzioni e delle relazioni statistiche.[Fasc. I.]*. Tipogr. di P. Cuppini, 1912.

[43] David Zimmermann. Asymmetric Impurity Functions, Class Weighting, and Optimal Splits for Binary Classification Trees. *ArXiv*, abs/1904.12465, 2019.

[44] Machine Learning Mastery. How To Implement The Decision Tree Algorithm From Scratch In Python. `https://machinelearningmastery.com/implement-decision-tree-algorithm-scratch-python/`, 2021.

[45] G.Ritschard S.Marcellin, D.A.Zighed. An asymmetric entropy measure for decision trees. *11th Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 1292–1299, 2006.

[46] J. Ross Quinlan. C4.5: Programs for machine learning. *Morgan Kaufmann Publishers*, 1993.

[47] G.Ritschard S.Marcellin, D.A.Zighed. Evaluating decision trees grown with asymmetric entropies. *Foundations of Intelligent Systems. Springer*, pages 58–67, 2008.

[48] R.Guermazi I.Chaabane. Impact of Sampling on Learning Asymmetric-Entropy Decision Trees from Imbalanced Data. *PACIS*, 2019.

[49] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[50] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: A new over-sampling method in imbalanced data sets learning. In De-Shuang Huang, Xiao-Ping Zhang, and Guang-Bin Huang, editors, *Advances in Intelligent Computing*, pages 878–887, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[51] Krystyna Napierala and Jerzy Stefanowski. Types of minority class examples and their influence on learning classifiers from imbalanced data. *Journal of Intelligent Information Systems*, 46:563–597, 2016.

[52] Gustavo Batista, Ronaldo Prati, and Maria-Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explorations*, 6:20–29, 06 2004.

[53] A. Chandra and X. Yao. Divace:diverse and accurate ensemble learning algorithm. *Intelligent Data Engineering and Automated Learning–IDEAL 2004, Springer*, pages 619–625, 2004.

[54] S. Gu and Y. Jin. Generating Diverse and Accurate Classifier Ensembles using Multi-Objective Optimisation. *Proc. IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM 14). IEEE*, pages 9–15, 2014.

[55] G. Brown. Diversity in neural network ensembles. *Ph.D. dissertation, School of Computer Science, University of Birmingham*, 2004.

[56] Giorgio Giacinto and Fabio Roli. Design of effective neural network ensembles for image classification purposes. *Image and Vision Computing*, 19(9-10):699–707, 2001.

[57] E.K Tang, P.N. Suganthan, and Y.Xao. An analysis of diversity measures. *Machine Learning*, 65:247–271, 2006.

[58] Whitaker C. Kunecheva L. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 5:181–207, 2003.

[59] D.W. Opitz and J.W. Shavlik. Generating accurate and diverse members of a neural-network ensemble. *Advances in Neural Information Processing Systems*, 8:535–541, 1996.

[60] Ayman Jarrous and Benny Pinkas. Secure hamming distance based computation and its applications. In *Applied Cryptography and Network Security: 7th International Conference, ACNS 2009, Paris-Rocquencourt, France, June 2-5, 2009. Proceedings 7*, pages 107–124. Springer, 2009.

[61] Jie Li, Wei Wei Shan, and Chao Xuan Tian. Hamming distance model based power analysis for cryptographic algorithms. *Applied Mechanics and Materials*, 121:867–871, 2012.

[62] Mary L McHugh. Interrater reliability: the kappa statistic. *Biochemia medica*, 22(3):276–282, 2012.

[63] Mona Choi, Saelom Kong, and Dukyoo Jung. Computer and internet interventions for loneliness and depression in older adults: a meta-analysis. *Healthcare informatics research*, 18(3):191–198, 2012.

[64] SRBH Chaturvedi and RC Shweta. Evaluation of inter-rater agreement and inter-rater reliability for observational data: an overview of concepts and methods. *Journal of the Indian Academy of Applied Psychology*, 41(3):20–27, 2015.

[65] GG Landis JRKoch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159174, 1977.

[66] Mina Mohammadi Kambs, Kathrin Hölz, Mark M Somoza, and Albrecht Ott. Hamming distance as a concept in dna molecular recognition. *ACS omega*, 2(4):1302–1308, 2017.

[67] Esteban Garzón, Roman Golman, Zuher Jahshan, Robert Hanhan, Natan Vinshtok-Melnik, Marco Lanuzza, Adam Teman, and Leonid Yavits. Hamming distance tolerant content-addressable memory (hd-cam) for dna classification. *IEEE Access*, 10:28080–28093, 2022.

[68] B. Littlewood and D.R. Miller. Conceptual modeling of coincident failures in multiversion software. *IEEE Transactions on Software Engineering*, 15(12):1596–1614, 1989.

[69] D.E. Eckhardt and L.D. Lee. A theoretical basis for the analysis of multiversion software subject to coincident errors. *IEEE Transactions on Software Engineering*, SE-11(12):1511–1517, 1985.

[70] H. Chen and X. Yao. Evolutionary multiobjective ensemble learning based on bayesian feature selection. *IEEE Congress on Evolutionary Computation*, 2006.

[71] Wikipedia. Multi-objective optimization. `https://en.wikipedia.org/wiki/Multi-objective_optimization`.

[72] Wikipedia. Genetic algorithm. `https://en.wikipedia.org/wiki/Genetic_algorithm`.

[73] Wikipedia. Genetic operator. `https://en.wikipedia.org/wiki/Genetic_operator`.

[74] J. Fieldsend and R.M. Everson. Multi-objective optimisation in the presence of uncertainty. *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC'05)*, pages 476–483, 2005.

[75] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[76] Dua D.and Graff C. UCI Machine Learning Repository, 2017.

[77] Y. Ren, L. Zhang, and P.N. Suganthan. Ensemble classification and regression-recent developments, applications and future directions. *IEEE Computational Intelligence Magazine*, 2016.

[78] H. Chen and X. Yao. Multiobjective neural network ensembles based on regularized negative correlation learning. *IEEE Trans. Knowl. Data Eng.*, 22:1738–1751, 2010.

[79] A. Rosales-Perez, S. Garcia, J.A. Gonzalez, C.A. Coello, and F. Herrera. An evolutionary multiobjective model and instance selection for support vector machines with Pareto-based ensembles. *IEEE Transactions On Evolutionary Computation*, 21, 2017.

[80] Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007.

[81] Dragos D. Margineantu and Thomas G. Dietterich. Pruning adaptive boosting. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, page 211–218, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[82] Vasudha Bhatnagar, Manju Bhardwaj, Dr Shivam, and Sufyan Haroon. Accuracy–diversity based pruning of classifier ensembles. *Progress in Artificial Intelligence*, pages 1–15, 06 2014.

[83] Ioannis Partalas, Grigorios Tsoumakas, and I. Vlahavas. Focused ensemble selection: A diversity-based method for greedy ensemble selection. pages 117–121, 01 2008.

[84] Gonzalo Martínez-Muñoz and Alberto Suárez. Aggregation ordering in bagging. 01 2004.

[85] Grigorios Tsoumakas, Ioannis Partalas, and I. Vlahavas. *An Ensemble Pruning Primer*, volume 245, pages 1–13. 01 1970.

[86] Tumer K. and Ghosh J. Error correlation and error reduction in ensemble classifiers. *Connection Science*, 8:385–404, 1996.

[87] Brown G., Wyatt J., Harris R., and Yao X. Diversity creation methods: A survey and categorisation. *Information Fusion*, 6:5–20, 2005.

[88] Tumer K. and Ghosh J. Error Correlation and Error Reduction in Ensemble Classifiers. *Connect. Sci.*, 8:385–404, 1996.

[89] Chandra A. and Yao X. Multi-objective ensemble construction, learning and evolution. In *PPSN Workshop Multi-objective Problem Solving from Nature (Part 9th Int. Conf. Parallel Problem Solving from Nature: PPSN-IX)*, pages 9–13, 2006.

[90] Michael P Perrone and Leon N Cooper. When networks disagree: Ensemble methods for hybrid neural networks. In *How We Learn; How We Remember: Toward An Understanding Of Brain And Neural Systems: Selected Papers of Leon N Cooper*, pages 342–358. World Scientific, 1995.

[91] Dieter Schlee. Numerical taxonomy. the principles and practice of numerical classification, 1975.

[92] Padraig Cunningham and John Carney. Diversity versus quality in classification ensembles based on feature selection. In *Machine Learning: ECML 2000: 11th European Conference on Machine Learning Barcelona, Catalonia, Spain, May 31–June 2, 2000 Proceedings 11*, pages 109–116. Springer, 2000.

[93] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001, 1990.

[94] Ross Quinlan. Statlog (australian credit approval. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C59012.

[95] William Wolberg, Olvi Mangasarian, Nick Street, and W. Street. Breast cancer wisconsin (diagnostic). UCI Machine Learning Repository, 1995. DOI: https://doi.org/10.24432/C5DW2B.

[96] Statlog (heart). UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C57303.

[97] Terry Sejnowski and R. Gorman. Connectionist bench (sonar, mines vs. rocks). UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5T01Q.

[98] V. Sigillito, S. Wing, L. Hutton, and K. Baker. Ionosphere. UCI Machine Learning Repository, 1989. DOI: https://doi.org/10.24432/C5W01B.

[99] Charles Spearman. The proof and measurement of association between two things. 1961.

[100] Trevor F Cox and Michael AA Cox. *Multidimensional scaling*. CRC press, 2000.

[101] Wojtek J Krzanowski and David J Hand. *ROC curves for continuous data*. Crc Press, 2009.

[102] Woodhouse I. H. The ratio of the arithmetic to the geometric mean: a cross-entropy interpretation. *IEEE Transactions on Geoscience and Remote Sensing*, 39(1):188–189, January 2001.

[103] Chen C.C., Sen P.K., and Wu K.Y. Robust permutation tests for homogeneity of fingerprint patterns of dioxin congener profiles. *Environmetrics*, 23(285-294), 2012.

[104] M. Beauchemin, K.P.B. Thomson, and G. Edwards. The ratio of the arithmetic to the geometric mean: a first-order statistical test for multilook sar image homogeneity. *IEEE Transactions on Geoscience and Remote Sensing*, 34(2):604–606, 1996.

[105] Yijun Bian and Huanhuan Chen. When does diversity help generalization in classification ensembles? *IEEE Transactions on Cybernetics*, 52(9):9059–9075, 2021.

[106] Fieldsend J.E., Bailey T.C., Everson R.M., Krzanowski W.J., Partridge D., and Schetinin V. Bayesian inductively learned modules for safety critical systems. *Computing Science and Statistics*, 35:110–125, 2003.

[107] Ripley B. D. Neural networks and related methods for classification (with discussion). *Journal of the Royal Statistical Society Series B*, 56(3):409–456, 1994.

[108] Kagan Tumer and Joydeep Ghosh. Bayes error rate estimation using classifier ensembles. *International Journal of Smart Engineering System Design*, 5(2):95–109, 2003.

[109] Gorman R. P. and Sejnowski T. J. Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets. *Neural Networks*, 1:75–89, 1988.

[110] UCI Machine Learning Repository. Connectionist Bench (Sonar, Mines vs. Rocks) Data Set. `https://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+(Sonar,+Mines+vs.+Rocks)`, 2021.

[111] Krzanowski W. J., Fieldsend J. E., Bailey T. C., Everson R. M., Partridge D., and Schetinin V. Confidence in Classification: A Bayesian Approach. *Journal of Classification*, 23(2):199–220, 2006.

[112] Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M., Prettenhofer P., Weiss R., Dubourg V., Vanderplas J., Passos A., Cournapeau

D., Brucher M., Perrot M., and Duchesnay E. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[113] Wikipedia. Broyden–Fletcher–Goldfarb–Shanno algorithm. `https://en.wikipedia.org/wiki/Broyden\%E2\%80\%93Fletcher\%E2\%80\%93Goldfarb\%E2\%80\%93Shanno_algorithm`, 2021.

[114] Frank Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in Statistics: Methodology and Distribution*, pages 196–202. Springer, 1992.

[115] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70, 1979.

[116] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, 2019.

[117] A. Dosovitskiy and Josip Djolonga. You only train once: Loss-conditional training of deep networks. In *ICLR*, 2020.

[118] Roger Koenker and Kevin F Hallock. Quantile regression. *Journal of economic perspectives*, 15(4):143–156, 2001.

[119] Isaac Newton. *The Method of Fluxions and Infinite Series: With Its Application to the Geometry of Curve Lines*. Nourse, 1736.