Department of Computer Science
The University of Auckland
New Zealand

Clustering-based Pruning for Bagging

Boyang, Tang

Jun 2018

Supervisor: Bernhard Pfahringer

A thesis submitted in partial fulfilment of the requirements of

Master of Professional Studies in Data Science

Abstract

Nowadays, more and more people are engaged in the fields of data mining and machine learning, although they have been constantly developed for several decades. While in the last few year, it appears that a new wave is upon us.

Bagging is a meta-algorithm that combines several machine learning techniques into one predictive model in order to decrease variance, bias, or improve predictions.

In this paper, we propose a new approach to improve the performance of regular Bagging by using a pruning method. After reviewing a relatively amount of related work, we use an unsupervised technique (K-means clustering) to prune ensembles. We propose three modified versions of the regular Bagging algorithm: C-Bagging, C-BaggingHO, and BaggingRC. The test output of our experiments shows that the modified versions only achieve small improvements, which are not significant. Thus, there might have other alternative ways to enhance the algorithms, the limitation and potential solutions will be concluded in the future work.

Acknowledgements

The success and final outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along the completion of my project. I would like to acknowledge and thank my supervisor professor Bernhard Pfahringer for his great encouragement and assistance on my dissertation. I am also grateful to the Dean of Data Science department Sebastian Link for his coordination, so that my dissertation can be supervised and proceed normally even Bernhard left the campus in the latter half of the year. Lastly, I would like to thank my parents for supporting me spiritually throughout writing this report and my life in general.

Contents

1 Introduction **1**

1.1 Background 1
1.1.1 Basic Concepts 1
1.1.2 Choosing Appropriate Learning Algorithms 3
1.1.3 Ensemble Methods 6
1.1.4 Application of Ensemble Methods 8
1.2 Motivation 10
1.3 Research Question 11
1.4 The Structure 12

2 Literature Review **13**

2.1 Ensemble Diversity 13
2.1.1 What is Diversity? 13
2.1.2 Diversity Measures 14
2.1.3 Generating Diversity 19
2.2 Ensemble Pruning 20
2.2.1 Many Could Be Better Than All 20
2.2.2 Ordering-Based Pruning 21
2.2.3 Clustering-Based Pruning 25
2.2.4 Optimization-Based Pruning 25

3 Algorithm Design **28**

3.1 Regular Bagging 28
3.2 Risk in Bagging 30
3.3 Proposed Algorithms 31
3.3.1 Clustering 31
3.3.2 K-means Clustering 32
3.3.3 Version 1—Clustering Bagging 33
3.3.4 Version 2—Clustering BaggingHO 35
3.3.5 Version 3—BaggingRC 36

4 Experiments **38**
4.1 WEKA—Data Mining Software in Java 38
4.2 K-fold Cross-Validation 39
4.3 Explore Datasets 40
4.4 Experiments and Results 44
4.5 Summary of Findings 47

5 Conclusion **49**

6 Future Work **51**

7 Appendix **52**
7.1 Extra experiment-output1 52
7.2 Extra experiment-output2 53

8 References **54**

1

Introduction

1.1 Background

1.1.1 Basic Concepts

Data mining and machine learning are not new in nowadays' life, they both are rooted in data science. Although, there is no common agreement on what difference is between data mining and machine learning. But Wikipedia gives a definition of data mining [1] which concludes that machine learning can be regarded as a sub-field of data mining. Data mining is a process of exploring patterns and potential information in massive data sets, and it is applied by interactively using methods from various fields such like machine learning, statistics, and database system. It is clear then that machine learning can be used for data mining. However, data mining can use other techniques besides or on the top of machine learning [2].

In the most cases, a large data set would be unusable unless people could derive useful patterns or relationships behind data. An appropriate tool or application must be able to analyse and interpret huge amounts of data and find recognizable patterns. It is true that data mining can reveal some patterns by doing some analyses on existing data, but it is indeed more than that. Actually, data mining can make a step further by using machine learning algorithms which automatically learn from collected data sets, and then can make predictions for unseen data in the future [2]. As we know, the higher quality of data collections, the higher accuracy of analyses. Data mining involves a sort of extracting and scraping techniques that can pull more accurate data from thousands of resources. This eventually enhances machine learning algorithms to achieve better results. In brief, constructing a good ***model*** from collected ***data sets*** is the prior task of machine learning and data mining [4].

Generally, a "data set" is a collection of data. Most commonly a dataset looks like a data table or data matrix, which is constructed by rows and columns. Each row of the table represents an ***instance***, and each column of the table represents a particular ***variable*** or ***attribute***. Indeed, every instance is a vector of attributes. In some cases, this kind of vectors are also called ***feature vectors*** [4], where each feature vector is a description of an object by using a set of features. For instance, have a look at the data set which consists of three features as shown in Figure 1.1. There is a group of data points denoted as feature vectors such like (.4, .7, circle) or (.6, .8, cross), each point is described by the features x-coordinate, y-coordinate and shape. The above data set is a ***three-dimension*** data set, where the ***dimension*** or ***dimensionality*** of a data set is the number of features of that data set.

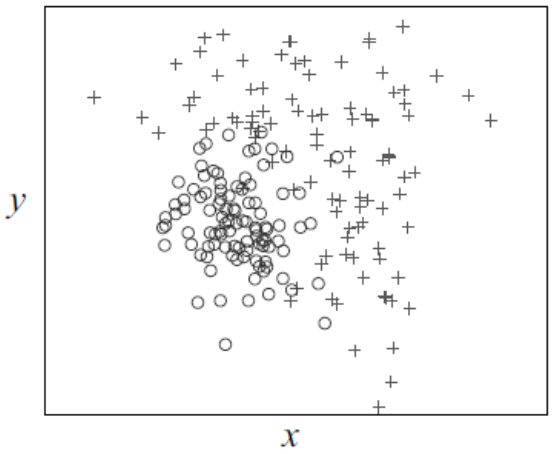

FIGURE 1.1: The synthetic *three-Gaussians* data set [4]

Our goal is to build a good predictive "model" from historical data to make a prediction on new data where we do not have the answer. In general, there are three kinds of data sets which are training data, validation data and testing data, where each kind of data is used for its respective purpose. In machine learning, the process of building models by running machine learning algorithms on training data set is called ***learning***. Usually, a learned model built after leaning is also called a ***learner***. The relationships and differences among above various data sets will be discussed in the later sections.

In supervised learning, the main task is to make predictions for target values on unseen data, thus the learned model is also called a ***predictor***. When the value of a target attribute is categorical or discrete, the task is a ***classification problem*** and the predictor is also called a ***classifier***. Whereas the value of a target attribute is numerical, the task is a ***regression problem*** [5]. For example, in the former three-Gaussians data set, the task is to predict the shape of a data point, where the "circle" and "cross" are called class ***labels***. This will be looked as a classification predictive modelling problem. More specifically, classification is the task of predicting a discrete class label while regression is the task of predicting a continuous quantity. In this case, the three-Gaussians data set

only has two classes, this particular predictive problem is also called ***binary classification***. Moreover, in binary classification, “Yes” and “No” are commonly used to present class labels, the result of a prediction is either positive or negative [4]. Unsupervised learning is a machine learning task of drawing hidden inferences from data sets without using corresponding class labels. ***Clustering*** is the most common unsupervised leaning method, which is aiming to find cluster or group structure of data points by using a measure of similarity [6].

Because of its nonnegligible importance, the performance evaluation of a predictive model is payed more and more attention by users. However, it is not easy to judge whether a model is good enough, since different users would have different issues to be addressed. Even if some expectations might be satisfied by a model that does not mean that it could make good predictions for other purposes [3,4]. There is an empirical strategy that is to evaluate models on ***test data***, and let the users to decide whether a model is acceptable. The process of performing models on test data is called ***testing*** or ***evaluation***. A good learner should have a small ***generalization error***, also called ***prediction error***. But it is impossible to know the exact error directly, since the unseen data lack label information [4]. A ***test error*** is the variance between prediction results and the real target value of test data. All the above mentioned basic concepts will be used for arguments in the later sections of this report.

1.1.2 Choosing Appropriate Learning Algorithms

Machine learning algorithms enable computers to learn information directly from data by themselves. As mentioned in section 1.1.1, machine learning consists of two types of techniques which are supervised learning and unsupervised learning. Almost all machine leaning algorithms can be classified into these two leaning categories. The Figure 1.2 illustrates a tree-based relationship among different learning techniques.

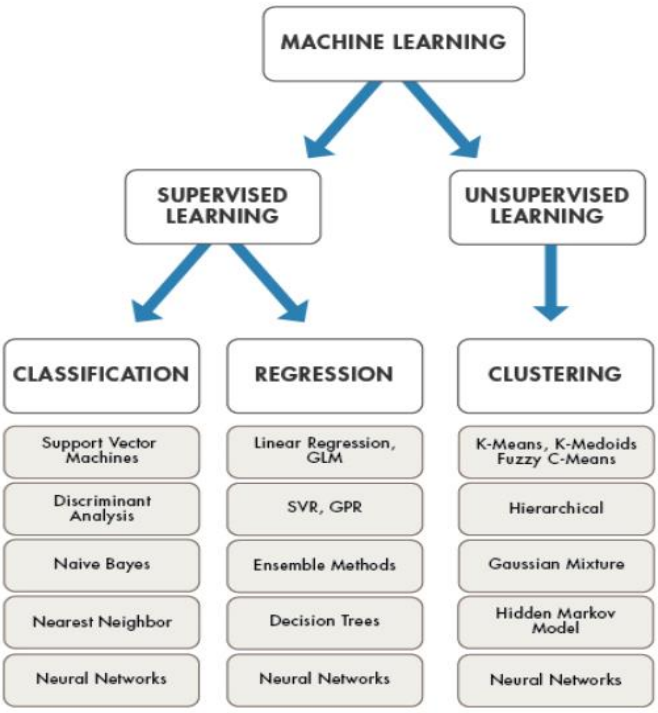

FIGURE 1.2: Machine learning Techniques [6]

In the supervised learning, algorithms are used to generate reliable models for solving classification or regression problems, while in unsupervised learning, algorithms are often used for clustering purpose like drawing inferences from data sets to find hidden patterns or groupings in data [6].

There is no explicit limit of using which machine learning algorithm for which kind of problems. A machine learning algorithm can be applied in supervised learning field and it may also performs well in unsupervised learning field. Take the neural networks for example, it has been widely used for classification and regression problems, and it is also applied to find clusters and groups for clustering problems. Moreover, in machine leaning, no one can state that which algorithm works best for every problem. For instance, no one can simply say neural networks are always better than decision trees or vice-versa.

It is not easy to decide which algorithm is appropriate in some cases. Even an experienced data scientist cannot tell which machine learning algorithm will perform better than others before trying a range of algorithms. Making a choice of the predictive model is sensitive on various factors, such as the quality and structure of data, the expectation of analyses, the urgency of the task, the available computational time, etc [7]. Looking for insight into individual algorithms would help in deeply understanding what they provide and how they are used. Some of the common and popular machine learning algorithms has been shown in the Figure 1.2, such as linear regression & logistic regression, decision tree, neural networks, k-means clustering, ensemble

methods, etc. The descriptions and details of those algorithms will not be discussed here. Assuming a beginner has been familiar with the knowledge or tips of some algorithms, there is a machine learning algorithm cheat sheet as shown in Figure 1.3, which assists him/her in choosing the appropriate algorithm for a specific problem.

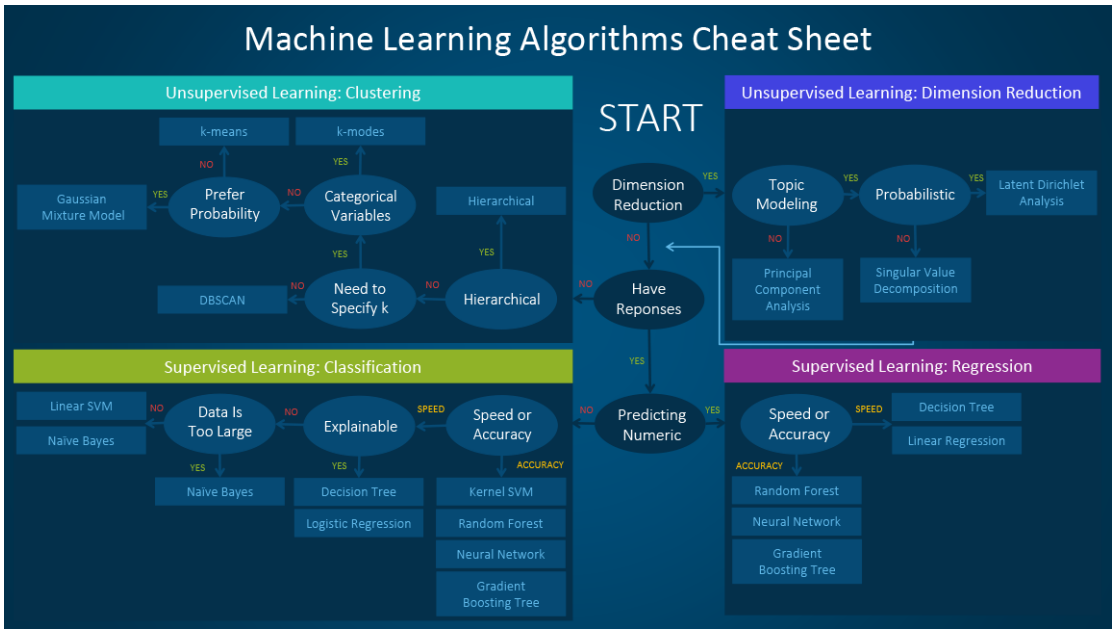

FIGURE 1.3: Machine Learning Algorithm Cheat Sheet [7]

The above cheat sheet is concluded by pruning a diversity of recommendations given from data scientists and machine learning experts. The cheat sheet is not going to advocate any specific one, but to give a guidance on the priority of the trials of different algorithms depending on some clear factors. More specifically, it is more likely to choose a path from the cheat sheet, for instance, if need a quick numeric prediction, use decision tree or linear regression, or if need a high accuracy categorical prediction, use random forest or neural networks. In some cases, more than one path could be applied, and other cases there does not exist a perfect match. If there is no obvious choice, it is better to try all algorithms beginning at one that you are most familiar with [7]. Many data scientists put the accuracy first, but only considering accuracy may not be sufficient. Appropriately choosing an algorithm may require consideration of many factors in addition to accuracy. Some other considerations could be training time, number of parameters, ease of use, etc.

1.1.3 Ensemble Methods

The goal of supervised learning problem is to search through a hypothesis space to find the best suitable hypothesis that will make good predictions, for traditional learning algorithms, they only construct one learned model from training data. In contrast, ensemble methods take a myriad of models into account, this machine learning technique combines multiple base models (learners or classifiers) and finally produces one optimal predictive model [8].

A common ensemble architecture is shown in Figure 1.4. Typically, the term ensemble in machine learning is usually reserved for methods which generate multiple hypotheses using a number of base learners. Base learners are usually obtained from training data on a base learning algorithm, which can be a decision tree, neural network or other kinds of learning algorithms [4,8]. The characteristics of ensembles usually refer to the diversity of base learners. Most ensemble methods produce homogenous ensembles, which contain based learners of the same type. While some other methods like Stacking use multiple learning algorithms to produce heterogenous ensembles, which contain based learners of different types [4].

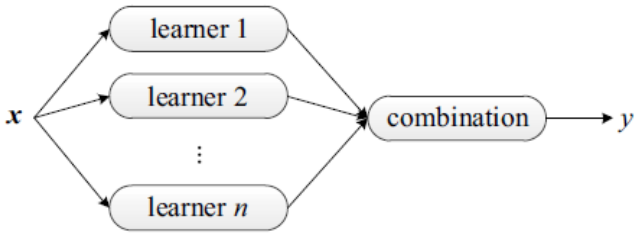

FIGURE 1.4: A common ensemble architecture [4]

Before looking into specific ensemble methods, it is necessary to be aware about some basic combination or integration techniques such like voting and averaging, where voting is usually used for classification and averaging is used for regression [4]. The first step of ensemble methods is to create multiple individual classifiers or regression models by using some training data. After that, we should combine the output of the base learners in the ensemble. Under the situation of voting, each classifier makes a prediction for each instance, and the final prediction is the most voted prediction, this kind of voting is also called majority voting [4,11]. Unlike majority voting, where assuming base classifiers have the same importance, weighted voting is a way to find a reasonable set of weights that emphasize the increased importance of some classifiers who have better performances [11]. In the above procedure, it is likely to count the prediction of the classifiers with higher weights multiple times.

Averaging technique also contains simple averaging and weighted averaging. In simple averaging, the final prediction of each test instance is obtained by averaging the prediction of each base regression model. While the weighted averaging is a slightly modified version of simple averaging, where the predictions is multiplied by a vector of weights and then their average is calculated [4].

Generally, ensemble methods can be divided into two groups according to the style of generating base learners. A parallel ensemble method generates base learners in a parallel style, which means each individual learner is independent from others. For example, in the Bootstrap Aggregating (also known as "Bagging") method, the individual learners are generated by using same base learning algorithm with bootstrap samples of the dataset which are randomly selected from the original dataset with replacement [4,12]. Whereas a sequential ensemble method generates base learners step by step, the latter constructed learners are based on the former ones, this shows the dependence between the base learners [4]. Taking Boosting for instance, it incrementally builds an ensemble by training each learner with the same data, but more weights will be given to the instances that are wrongly predicted by previous learners, therefore the subsequent learners give more attention to them during training [4,13]. This research report is based on Bagging, more details will be introduced in the later sections.

Compared to ordinary machine learning algorithms, the ensemble methods are meta-algorithms which designed to improve the stability and the accuracy of predictions. As the Bagging and Boosting mentioned above, ensemble methods combine a number of weak learners to create a strong learner that can improv predictions [14]. The main causes of error in predictions are due to noise, variance and bias, ensemble methods help in reducing the effects of these factors [8,11].

It is not easy to trace the starting point of the history of ensemble methods, but there were two pieces of pioneering work that made a significant contribution to prompting ensemble methods to become a major learning paradigm in 1990s. The empirical study [9] found that the combination of a set of classifiers often perform better than the best single classifier. The observation is illustrated in Figure 1.5. Another theoretical study [10] has proved that it is better to convert weak learners to strong ones rather than directly obtaining them, since weak learners are easier to get in real practice. This result lays the groundwork for progress in generating strong learners by ensemble methods.

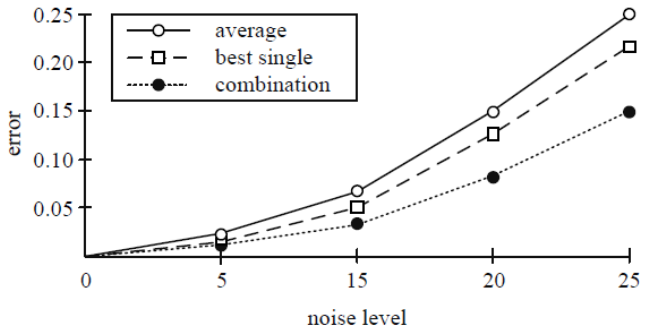

FIGURE 1.5: A simplified illustration of study [9]'s observation: Ensemble is often better than the best single.

1.1.4 Application of Ensemble Methods

In order to know the ensemble methods that have been successfully applied in various real-world tasks, it is good to start from looking through two of the most popular data mining competitions, which are ***KDD-Cup***[1] and ***Netflix Prize***[2].

The KDD-Cup is the annual Data Mining and Knowledge Discovery competition held for attracting the interests of data miners all around world since 1997. This competition provides a comprehensive range of practical tasks, such as direct marketing for profit optimization (1998), biomed document & gene role classification (2002), consumer recommendation (2007), early detection of breast cancer (2008), favourite songs recommendation (2011), identify the best research institutions (2016), etc. Among all techniques utilized by participants, the ensemble methods were far ahead of others and won a majority of the past KDD-Cup competitions. For example, ensemble methods are used by all the first-place and second-place winners in competitions from year 2009 to year 2011.

1http://www.kdd.org/ 2https://www.netflixprize.com/

Netflix held the Netflix Prize open competition for seeking the best algorithm to substantially improve the accuracy of predictions about how much someone is going enjoy a movie based on their movie preferences. In 2009, Netflix awarded a grand prize of $1 million to the research team *BellKor's Pragmatic Chaos*, whose algorithm improved the accuracy of the company's recommendation engine by 10%. The winning algorithm was an ensemble method consisting of various types of classifiers, such as regression models, neighbourhood models, matrix factorization, restricted Boltzmann machines, etc.

Moreover, beyond the impressive results in competitions, many researchers have found that the ensemble methods are also very efficient when exploited in some places such as computer vision, computer security and computer aided medical diagnosis [18-25]. In the case of computer vision, which involves some popular branches such like object detection, recognition, and tracking [18,19,20].

1.2 Motivation

As mentioned above, the ensemble methods have been commonly used in many domains of human society for a long time. Even though it has been proved that ensemble methods usually can produce higher accuracy and more reliable estimates than a single model, but there still exists some uncertainty due to the procedure of building an ensemble. Factors such as types of ensemble, ensemble diversity, and number of based learners should be taken into account in the build of committees [15].

In general, there are two steps to building an ensemble, i.e., generating multiple base learners, and then combining them together. A proper ensemble size would decide whether the method can achieve a great success. If the number of individual learners is very small, the ensemble would not be performed well. While when the number of individual learners become huge, the performance improves but will result in high storage space and computational time [15,16]. It is essential to find a good way that can efficiently select a proper number of individual learners.

In addition to size, ensemble diversity is looked as another fundamental issue which also affects the accuracy of ensemble methods. It is necessary that the ensemble should introduce enough diversity into it, if no difference among the individual learners, there would be no improvement of performance [4]. In fact, the diversity is meaningful only when the individual learners are not very poor. The combination of individual learners would not improve and could even worsen the performance when they have little accuracy [15]. It is difficult to ensure that the individual learners are as accurate as possible and as diverse as possible. Our research tries to find a solution which achieves a good trade-off between the individual performance and diversity.

Some researchers [15,17] suggest that the performance of the ensembles also depends on whether a suitable combination/integration method has been chosen. In the second step of building an ensemble, integration methods are used for combining the decision of the base learners to a single output. The ensemble shows the benefit from combination that dramatically reduce the variance and bias, this has been proved by many empirical studies.

Moreover, if the performance of an ensemble method is just a bit better than that of a single model, is there a way to make any further improvement? In term of optimization process, some techniques such like selective ensemble methods and pruning of ensembles have been proposed by many researches [4]. Ensemble selection is used to increase diversity, which can be utilized in many different ways. For example, feature selection provides different subsets of attributes for training base learners, so that is to enhance independence and reduce redundancy [15]. Sometimes, many could be better than all, ensemble pruning aims to comprise an ensemble by selecting a subset of trained base learners rather than combining all of them. Many literatures have proposed that it is possible to get a smaller yet better ensemble through ensemble pruning [4]. Therefore, our research is tries to create a more competitive model by modifying existing ensemble models by the above techniques.

1.3 Research Question

Our dissertation is based on research about how we can improve further on the existing ensemble method “Bagging”, where the name Bagging is the abbreviation from Bootstrap Aggregating [4,12]. Compared to Boosting, which tries to reduce bias, Bagging is good at reducing variance. On the other hand, Bagging may solve the over-fitting problem, while Boosting can increase it. Our research would pay more attention to classification problems, here are some specifically defined questions:

1. whether an ensemble has enough diversity is one of the most important factors that are responsible for the success of an ensemble. Diversity can be created in many ways, in the case of Bagging, diversity is created through bootstrap resampling, where the individual learners are independent from each other as they are trained on different data subsets (same size as the original dataset but with replacement). Is it enough to significantly improve performance? Does this give a promise of keeping enough diversity?

2. given a set of trained individual learners, the final decision is made by averaging or majority voting in Bagging, where each individual learner is equally weighted. What if the mistakes made by those base learners are correlated, or the accuracy of base learners are poor? Is there a way to pay more weight to some of those with better performance on training data?

3. Our main research question is to look more at whether getting a further improvement is possible when combining Bagging with a pruning technique such as clustering. On the one hand, when we are going to do a big number of iterations, say 1000, it will generate 1000 trained base learners. The size of an ensemble is too large to perform efficiently, thus we need refine the ensemble and then choose a subset of it. On the other hand, it is also necessary to keep enough diversity of the ensemble during the process of pruning. We decide to use a simple K-means clustering method, will it work for above issues?

1.4 The Structure

The rest part of report is structured as follows:

Chapter 2 will introduce related work about ensemble diversity and present existing literature on ensemble pruning techniques, particularly in clustering ensemble methods. Chapter 3 presents the algorithms, system design and architecture that are based on Weka to solve our research problem. The implementation of our algorithms and experiments on Weka will be discussed in Chapter 4, and the findings of our results and some comparisons among algorithms will be also analysed. Chapter 5 summaries our work and gives a conclusion of the report. Lastly, Chapter 6 discusses the limitations of our approach and further improvements which might be done in the future.

2

Literature Review

2.1 Ensemble Diversity

2.1.1 What is diversity?

Typically, ensemble diversity is the quantified estimation of the distinction of making the same mistakes by individual learners in an ensemble. We can also simply view it as the difference between base learners in the ensemble [4,26].

Common intuition suggests that the individual learners in the ensemble must be as accurate as possible, and as diverse as possible. In terms of diversity, the base learners should make errors on different objects. To gain benefits from the combination of multiple learners, they should be diverse, otherwise there would be no performance improvement [26].

In practice, generating diverse individual learners is not that easy, since high correlation among these individual learners is caused by training them on the same dataset. It is good to find out whether the diversity is crucial to ensemble performance, this can be judged after combining base learners. [27] analysed a combination method called simple soft voting, which is commonly used when base learners produce class probability outputs. Suppose the ensemble size is T and the individual classifier h_i produce a n-dimensional vector $(h_i^1(\mathrm{x}), \ldots, h_i^n(\mathrm{x}))^{\mathrm{T}}$ for the instance x, where $h_i^j(\mathrm{x}) \in [0,1]$ can be viewed as the estimate of the probability that object x is classified into class j. Simple soft voting treats all individual classifiers with equal weights, and thus the final probability for class j is simply averaged over all the individual outputs:

$$H^j(x) = (1/T)\sum_{i=1}^{T} h_i^j(\mathrm{x})\ . \qquad (2.1)$$

The study of [27] introduced a coefficient σ ∈ [0,1] to describe the overall correlation among individual classifiers. The expected ensemble error is calculated by

$$err(H) = \frac{1+\sigma(\mathrm{T}-1)}{\mathrm{T}}\mathrm{err(h)}\ , \qquad (2.2)$$

where the err(h) is the expected error of the individual classifier. Here, all individual classifiers were assumed to have equal error, (2.2) reveals the truth that correlation among individual classifiers has an impact on ensemble performance. More specifically, if all individual classifiers are totally unrelated to each other, i.e., σ = 0, then the ensemble error is dramatically decreased by a factor of T. Whereas, if the individual classifiers are entirely correlated, i.e., σ = 1, this cannot make any advance in combination. Therefore, diversity is a vital requirement for the success of an ensemble.

2.1.2 Diversity Measures

Intuitively, diversity is a fundamental issue for ensemble methods. Although the connection between diversity and ensemble performance has been proved, but we still do not deeply understand how to measure the diversity among the members within an ensemble. In fact, there is no common agreement on measuring diversity, since it is not straightforward. Here is a quick review of some popular approaches.

2.1.2.1 Pairwise Measures

For measuring the diversity of an ensemble, many methodologies have been proposed by various studies. In terms of pairwise measures, which consider a pair of individual learners at each time, the overall diversity is obtained from averaging all pairwise measurements, where each measurement is the agreement/disagreement among any pair of base learners.

Suppose there is a data set $\boldsymbol{D} = \{(x_1,y_1), \ldots, (x_m,y_m)\}$, where x_i represents the input variables and y_i denotes the outputs or class labels. For binary classification problems, $y_i \in \{+1, -1\}$, which means the prediction output of an instance is either positive or negative. The agreement or disagreement on a prediction made by a pair of individual classifiers can be illustrated by the following table:

TABLE 2.1: contingency table for two classifiers

	$C_i = +1$	$C_i = -1$
$C_j = +1$	a	c
$C_j = -1$	b	d

The table shows the correlation between two classifiers C_i and C_j, the ***Disagreement Measure*** [29,30] maybe the most intuitive measure of diversity between a pair of classifiers. This measure is denoted as the proportion of instances on which the two classifiers make different predictions, i.e.,

$$D_{ij} = \frac{b+c}{m} . \tag{2.3}$$

Where m = a + b +c + d, D_{ij} is non-negative, and the value of D_{ij} is within [0,1]; the larger the value, the larger the diversity.

There are various statistics to assess the similarity of two classifier outputs. the study of [31] stated the correlation between two binary classifier outputs can be defined as

$$\rho_{ij} = \frac{ad-bc}{\sqrt{(a+b)(c+d)(a+c)(b+d)}} , \tag{2.4}$$

where ρ_{ij} is the ***Correlation Coefficient*** of C_i and C_j. Another classic statistic introduced by [32] is the Q-***Statistic***, which is defined as

$$Q_{ij} = \frac{ad-bc}{ad+bc} , \tag{2.5}$$

For statistically independent classifiers, the expectation of Q_{ij} is 0. The value of Q_{ij} varies in the range of [-1,1]. Q_{ij} is positive if C_i and C_j make similar predictions; Q_{ij} is negative if C_i and C_j make different predictions. It is easy to find that Q and ρ have the same sign, for any pair of classifiers. It can also be proved that $|\rho| \leq |Q|$.

Kappa-Statistic [33] is also regarded as a classic measure in statistical literature, this statistic is developed for measuring interrater reliability. It is used to measure the diversity between two classifiers by comparing the proportion of agreement and the agreement by chance. It is defined as

$$\boldsymbol{\kappa_p} = \frac{\Theta 1 - \Theta 2}{1 - \Theta 2} , \tag{2.6}$$

where Θ_1 is the proportion of examples on which two classifiers make the same predictions, and Θ_2 is the probability that the two classifiers agree by chance [34]. These two variables can be calculated according to the table 2.1.

$$\Theta_1 = \frac{a+d}{m}, \tag{2.7}$$

$$\Theta_2 = \frac{(a+b)(a+c)+(c+d)(b+d)}{m*m}. \tag{2.8}$$

$\boldsymbol{\kappa_p} = 1$ if the two classifiers totally agree on a dataset; $\boldsymbol{\kappa_p} = 0$ if the two classifiers agree by chance; $\boldsymbol{\kappa_p} < 0$ is a rare case where the agreement is even less than what is expected by chance.

Double-Fault Measure was used by [35] to form a pairwise diversity matrix for measuring the proportion of observations that have been misclassified by both classifiers C_i and C_j. i.e.,

$$df_{ij} = \frac{e}{m}, \tag{2.9}$$

where e is the number of instances that satisfy $C_i(x_k) \neq y_k$ and $C_j(x_k) \neq y_k$.

2.1.2.2 Non-Pairwise Measures

Unlike pairwise measures, which assess ensemble diversity by averaging all pairwise measurements, non-pairwise measures aim to consider all the classifiers together and calculate the ensemble diversity directly. In the following we will look through some representative non-pairwise measures that have been proposed in the literature.

Interrater agreement is a measure of inter-classifier reliability [59], it should be set apart from Kappa-statistic, and it is not equal to the average of all pairwise $\boldsymbol{\kappa_p}$. However, for the same purpose, this approach is designed for measuring the level of agreement within a set of classifiers directly instead of pairs. Given a set of individual classifiers $\{\boldsymbol{h_1, \ldots, h_T}\}$ and a dataset $\boldsymbol{D} = \{(\boldsymbol{x_1,y_1}), \ldots, (\boldsymbol{x_m,y_m})\}$ where $\boldsymbol{x_i}$ is an instance and $\boldsymbol{y_i} \in \{+1, -1\}$, then this measure can be defined as

$$\kappa = 1 - \frac{\frac{1}{T}\sum_{k=1}^{m} \rho(\boldsymbol{x}_k)(T - \rho(\boldsymbol{x}_k))}{m(T-1)\bar{p}(1-\bar{p})}, \tag{2.10}$$

where the $\rho(\boldsymbol{x_k})$ is the number of classifiers that classify $\boldsymbol{x_k}$ correctly, and

$$\bar{p} = \frac{1}{mT}\sum_{i=1}^{T}\sum_{k=1}^{m} \mathbb{I}(h_i(\boldsymbol{x}_k) = y_k) \tag{2.11}$$

is the average accuracy of individual classifiers [58]. Like $\boldsymbol{\kappa_p}$, $\boldsymbol{\kappa} = 1$ if the classifiers totally agree on $\boldsymbol{D}$, and $\boldsymbol{\kappa} \leq 0$ if the agreement is even less than what is expected by chance.

Intuitively, the ensemble is most diverse for a particular instance $\boldsymbol{x_k}$ only when a half of votes (T/2) are 0s (1s) and the other half of votes are 1s (0s). If the votes all were 0s or all were 1s, there is no disagreement among individual learners, and therefore the classifiers cannot be deemed diverse. ***Entropy*** is another possible non-pairwise measure of diversity based on above concept [60], that is,

$$E = \frac{1}{m}\sum_{k=1}^{m}\frac{\min\left(\rho\left(x_k\right), T-\rho\left(x_k\right)\right)}{T-\lceil T/2\rceil}, \tag{2.12}$$

where $\rho(\boldsymbol{x_k})$ is denoted as the number of classifiers that correctly recognize $\boldsymbol{x_k}$. $\boldsymbol{E}$ varies between 0 and 1, where 0 indicates no difference while 1 indicates the largest the diversity. Notice that (2.12) is not a classical format of entropy, since it does not use the logarithm function. A classical version of this measure by using logarithm transformation is more suitable in practical problems since it is easier to handle and faster to calculate [60].

The measure of "***difficulty***" θ was originally proposed by [9] in 1990. Let a random variable X taking values in $\{0, \frac{1}{T}, \frac{2}{T}, \ldots, 1\}$ denote the proportion of classifiers that correctly classify an instance $\boldsymbol{x}$ that is randomly selected from the distribution of the problem. The probability mass function of X can be estimated by running the T classifiers on the dataset $\boldsymbol{D}$.

The Figure 2.1 shows three patterns of "difficulty" for three classifier teams with T=7 and N=100 data points. If the same points are difficult for all classifiers, and the other points are easy for all classifiers, the distribution shape is with two separated peaks, which should be very similar to the middle plot (b). If the points that are difficult for some classifiers and are easy for other classifiers, the distribution of *X* should be similar as the plot on the right (c). And if each point is equally difficult for all classifiers, the distribution shape on the left is the most likely one (a).

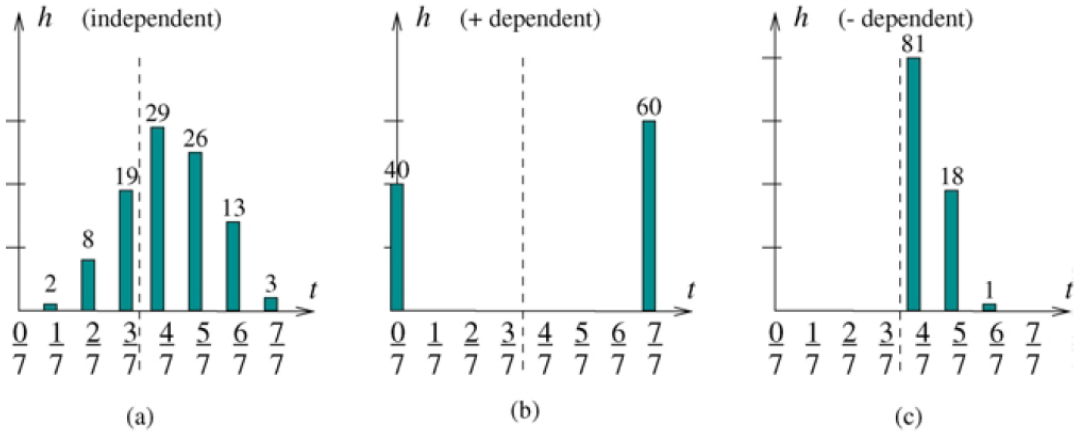

FIGURE 2.1: *Figure 1*. Patterns of "difficulty" for three classifier teams with $T = 7$, $p = 0.6$, and $N = 100$. Assuming all classifiers have individual accuracy $p = 0.6$. The dashed line is the majority vote border. The histograms show the number of points (out of 100) which are correctly labelled by i of the T classifiers. The x-axis is "proportion correct", i.e., i/T.

So, we can measure the diversity based on the distribution of "difficulty", and using the variance of X to capture the distribution shape, the "difficulty" measure can be defined as

$$\theta = variance(X). \tag{2.13}$$

It is obvious that the smaller the θ value, the larger the diversity.

In addition to these non-pairwise measures, many other approaches have been proposed in the past decades, such as ***Generalized Diversity*** [37], ***Coincident Failure*** [37], and ***Kohavi-Wolpert Variance*** [39]. The Figure 2.2 shows a table of the comparison of 12 diversity measures. Notice that the Entropy mentioned above is assumed to know the correctness of the classifiers [60], it is denoted as Ent_{sk} in the table. And the table illustrates whether a measure is pairwise or non-pairwise, whether it requires to know the correctness of classifiers, and whether it is symmetric or non-symmetric. A **symmetric measure** will keep the same when the values of 0 (incorrect) and 1 (correct) in binary classification are swapped [36].

Diversity Measure	Symbol	↑/↓	Pairwise	Known	Symmetric
Disagreement	dis	↑	Yes	No	Yes
Q-statistic	Q	↓	Yes	No	Yes
Correlation coefficient	ρ	↓	Yes	No	Yes
Kappa-statistic	κ_p	↓	Yes	No	Yes
Double-fault	df	↓	Yes	Yes	No
Interrater agreement	κ	↓	No	Yes	Yes
Kohavi-Wolpert variance	kw	↑	No	Yes	Yes
Entropy (C&C's)	Ent_{cc}	↑	No	No	Yes
Entropy (S&K's)	Ent_{sk}	↑	No	Yes	Yes
Difficulty	θ	↓	No	Yes	No
Generalized diversity	gd	↑	No	Yes	No
Coincident failure	cfd	↑	No	Yes	No

FIGURE 2.2: Summary of ensemble diversity measures, where ↑ (↓) indicates that the larger (smaller) the measurement, the larger the diversity ("Known" indicates whether it requires to know the correctness of individual classifiers) [28].

2.1.3 Generating Diversity

When an ensemble is built with identical base learners, there will not be any improvement in terms of accuracy than when a single model is used. Though there is no commonly accepted formal formulation and measure for ensemble diversity, there are many ways to create or increase diversity according to the literatures studied. Intuitively, the basic idea is about randomness that is utilized in ensemble construction process. Here, we will review some popular and effective diversity generation mechanisms such as manipulating the data samples, input features selection, learning parameter tuning, and output representation manipulation.

Data sample manipulation. This may be the most popular mechanism that is wildly used in various areas. Our algorithms design and experiments are based on this kind of approach. Given a dataset, multiple data samples can be randomly generated by *sampling* approaches, and then the base learners are trained from these different data samples. In Bagging [12], diversity is created through *bootstrap resampling*, where each training subset is randomly resampled from the original dataset with replacement, and any training sample is kept separate from each other. While in Boosting [13], the training process is based on *sequential sampling*, each base learner will be trained on a sample without replacement but with more weights to some observations that are misclassified by the previous learners.

Feature selection. Another useful method to achieve diversity is manipulating the feature space. It has been proved that feature manipulation is very suitable for a dataset with a large feature space, which contains a lot of redundant features or irrelevant features. The *Random Subspace* [30] is a famous method that employs this mechanism, which allows the base learners to be trained on data samples with different subsets of features. Therefore, the ensembles attained by using different subspaces are usually diverse and enhances the quality of results.

Learning parameter tuning. Diversity is also achieved through manipulation of training parameters by assigning different values for each of the parameters. Generally, that said, the goal is usually to set those parameters to optimal values that enable to complete a learning task in the best way possible. This mechanism tries to generate diverse individual learners by tuning various parameter settings for the base learning algorithm. To make it more concrete, here are a few examples. Suppose a task for clustering by using KNN [46], during the process you must specify the value of K for models. Tring many different values of K for a model and looking for the best one. Moreover, different initial weights can be assigned to individual neural networks [43], different split percentages can be applied to individual decision trees [44], different candidate rule conditions can be applied to individual FOIL rule learners [45], etc.

Output representation manipulation. This mechanism tries to generate diversity by using different output representations. Some popular approaches have been proposed by various literature studies, such like the *ECOC* technique which employs error correcting output codes [40], the *Flipping Output* method that randomly changes the labels of some training observations [41], etc. It is notable that some people would be more interested in other attempts through using *artificial training data* [42] to encourage diversity. The ensemble will be constructed in an iteratively way. More specifically, in each round, a sort of artificial instances is randomly generated based on the data distribution, and then assigning labels to these artificial instances. Notice that those labels are different maximally from the predictions of the current trained learners. Subsequently, a new learner will be trained on the combination of original data and artificial data. Add a new learner to current ensemble if it can remit the training error, otherwise, discard the new learner and repeat the previous step.

Notice that different diversity generation mechanisms can be used individually or together in practice. For example, *Random Forest* trees [39] adopts both the mechanisms of data sample manipulation and input feature selection.

2.2 Ensemble Pruning

2.2.1 Many could be better than all

Given a set of trained base learners, instead of combining all of them, ***ensemble pruning*** tries to select a subset of the pool to comprise the ensemble, aiming to increase efficiency by reducing the number of learners. Intuitively, the ensemble can be regarded as a second-level learner, or meta-learner, since it is converted from a set of weak learners (base learners). From this aspect, ensemble pruning could be looked as a procedure which sets the weights on some base learners to zero.

It is essential to note that pruning methods differ from sequential ensemble methods [4]. Take AdaBoost for example, it is a conventional sequential ensemble method, some individual learners may be discarded during training process if they cannot pass the check. Whereas, the pruning process happens when all individual learners have been already generated, and no more individual learners will be produced during this process. In addition, once any individual learner has been added into an ensemble, it would not be removed anymore. But for ensemble pruning methods, any individuals can be excluded at any time.

The final purpose of pruning is attempting to reduce the size that may preferably enhance the efficiency without sacrificing generalization ability. In order to prove that it is possible to get a smaller yet better ensemble through ensemble pruning, the study [52] introduced the analyses on both regression and classification that concluded the theorem of MCBTA ("*many could be better than all*"). This conclusion shows that for supervised learning, if given a set of trained base learners, it may be better to combine some instead of all of these base learners.

The work of ensemble pruning has received a lot of attention in the last few decades, some studies also called it selective ensemble, while others called the pruning ensemble selection. However, here, we use "ensemble pruning" as the general designation for referring to all of them. Lately, many studies were devoted to find effective ensemble pruning methods. According to the study of [4,47], it shows that those methods can be classified into three classes, which are ***Ordering-based pruning***, ***Clustering-based pruning*** and ***Optimization-based pruning***.

2.2.2 Ordering-based pruning

Firstly, it is worthwhile to introduce the category named Ordering-based pruning. From the meaning of name, those methods work in a sequential style, individual learners may be ordered by some restrictions and only top ones will be selected and then comprised as the ensemble.

Suppose there is a number of N individual learners, and they are randomly ordered in a sequence. In general, the generalization error of the ensemble is corresponding to the ensemble size, the error will decrease if the size of an ensemble increases to be bigger and bigger. The basic idea of Ordering-based pruning is about how to find an appropriate way to order the given individual learners, and then select T of them to reach the minimum error level. The following Figure 2.3 shows the relation between the size and the error:

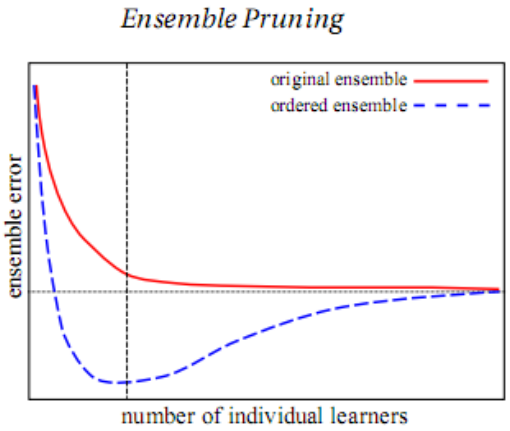

FIGURE 2.3: Illustration of error curves of the original ensemble (aggregated in random order) and ordered ensemble.

In general, making a decision on the best value of T is hard, but fortunately there are many T values that will lead to better performance than including all members of N. Actually, the core problem of this pruning method is how to design a practical rule for ordering individual learners. In most cases, both the accuracy and diversity of individual learners should be taken into account when setting up the ordering strategies. The book "Ensemble Methods Foundations and Algorithms" [4] has mentioned some typical order-based pruning methods, whose advantages and weakness will be illustrated as follows.

Reduce-Error Pruning chooses the individual learner who has the smallest validation error as the start-point. After that, adding front T individual learners into the ensemble, meanwhile; keeping the validation error as low as possible. Lastly, using a technique called "Backfitting" to make an improvement of this ensemble, it is realized by replacing an individual learner which is currently held in the ensemble with an outside learner. This replacement occurs if and only if the outside learner can produce a lower ensemble error. In brief, this method is greedy and "Backfitting" is time consuming, moreover, "Backfitting" may be suitable for sequential style but it could not do better for parallel ensemble pruning methods such as bagging [52]. Bagging can be regarded as a selection of different types of individual learners, a selective ensemble is the product of the pruning of parallel ensembles, and an optimal size of the ensemble leads to a better performance.

There are two pruning methods that draw on Kappa statistic for assistance, they are ***Kappa Pruning*** and ***Kappa-Error Diagram Pruning*** [48]. Kappa statistic is used to measure the agreement between two learners, if the value of Kappa equals to 1, which means those two learners totally agree on the validation set. And if value of Kappa is less than 1, which means there is not a perfect agreement between two learners; while Kappa can be 0, which means two learners agree by chance [33]. In short, the smaller the value of Kappa, the larger diversity is among individual learners. The Kappa Pruning method treats performances of given learners as same, and then, creates a list to save pairs of individual learners in ascending order of Kappa values. Therefore, the first T individual learners will be added into the ensemble. A more advanced version of this method was proposed by [50], in this case, the Kappa statistic is represented by interrater agreement. Now, the Kappa is used to measure the level of agreement among a set of individual learners rather than just two [49]. The first step of this version is same as former one, once the top T individual learners have been put into the ensemble, it then calculates the Kappa value between each unselected individual learner and the subset (T-1) of the current ensemble, replacing the last one of the ensemble with an unselected individual learner which has the smallest Kappa value. The new version of the ensemble will lead to smaller error at the expense of slightly more memory and runtime.

Kappa-Error Diagram Pruning method is based on both accuracy and diversity. A Kappa-Error diagram was introduced by [4], and this is a two-dimension grid, where the x-axis represents the Kappa statistic of a pair of individual learners, and the y-axis represents the error rate. Each point of the plot is a pair of two individual learners. The following figure shows two examples of Kappa-Error diagram:

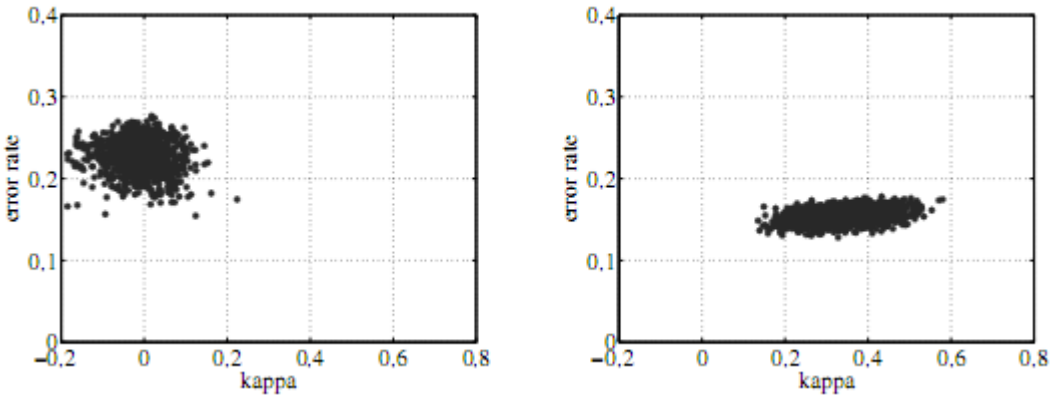

FIGURE 2.4: Examples of two ensembles' kappa-error diagrams on *credit-g* data set, where each ensemble comprises 50 C4.5 decision trees.

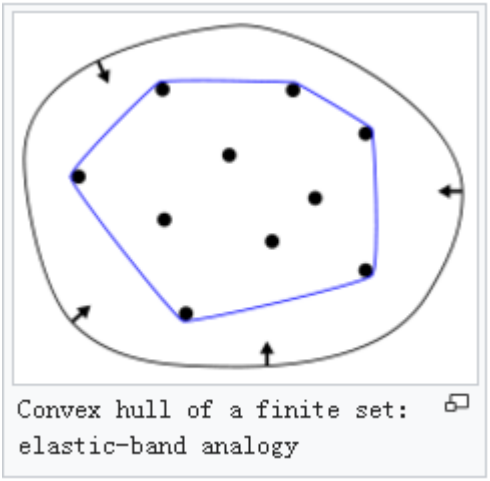

FIGURE 2.5: The convex hull of a sort of points in the Euclidean plane.

From Figure 2.4, where the left diagram is obtained by using Bagging, while the right one is obtained from Boosting. It can be seen that the kappa-error diagram visualizes the accuracy-diversity trade-off of different ensemble methods. The higher the point clouds, the less accurate the individual classifiers; the more right-hand the point clouds, the less diverse the individual classifiers.

Figure 2.5 illustrates an example of constructing the *convex hull* of a number of points in the *Euclidean plane*. In this method, it constructs the *convex hull* according to the points on the kappa-error diagram, and the final product of a pruned ensemble consists all individual learners that appears on the convex hull.

Moreover, there are many other methods mentioned in [4], such as ***Complementariness Pruning***, ***Boosting-Based Pruning*** and etc. Each of them considers an aspect that is not involved in former methods. Complementariness Pruning takes account of the individual learners whose performance is complementary to that of the current pruned ensemble. More specifically, the first step of this method is similar to that of Reduce-Error Pruning, every subset of the ensemble grows from the individual learner whose validation error is the smallest. Suppose it is at *t*th round, the subset of ensemble H*t-1* with size *t-1*, and the last element of this subset is h*t*. Then, this pruning method selects the individual learner h*t* from unselected set, and h*t* must satisfy

$$h_t = \arg\max_{h_k} \sum_{(\boldsymbol{x},y)\in V} \mathbb{I}\left(h_k(\boldsymbol{x}) = y \text{ and } H_{t-1}(\boldsymbol{x}) \neq y\right), \tag{2.14}$$

V is the validation data set [53].

Boosting-Based Pruning introduces a powerful strategy *boosting with restart*. This method uses the AdaBoost algorithm to determine the order of the classifiers. It is not generating any new base classifiers from the training data, instead only the classifier with smallest validation error will be selected. Under this situation, each classifier has a weight, and this pruning method also starts with the classifier with lowest weighted validation error. Then, using boosting to prune the ensemble, once the aggregate weight is larger than 0.5, the strategy is triggered. For example, there are five classifiers C1, C2, C3, C4, and C5 with weights 0.2, 0.1, 0.5, 0.3, and 0.4 respectively. Ordering them in increasing by weights, the classifier with smallest weight is C2 (0.1<0.5), and the second order is C1 (0.1+0.2<0.5). The third order is C4 (0.1+0.2+0.3>0.6), while this time the aggregate weight is bigger than 0.5, the Boosting with restart approach is applied, that is, the weight is reset to zero and the current selected classifier C4 is removed. The pruning method restarts from the classifier with smallest weight in the rest, which is C5 (0+0.4<0.5). Then, it is C3 (0.4+0.5>0.5), the weight is larger than 0.5, therefore; the Boosting with restart is applied again. Lastly, the pruned ensemble is ordered as C2, C1 and C5. This helps the pruned ensemble keep a low level of error rate as much as possible [54].

2.2.3 Clustering-based pruning

Generally, there are two steps when working with this kind of ensemble pruning methods. Firstly, using clustering techniques to classify individual learners into different groups or clusters. Then, selecting some representative learners from each cluster to construct the ensemble. In the second step, it is essential to keep a fair representative of the given individual learners, as this would keep the diversity of ensemble. K-means clustering [55] is a widely used technique in different fields for its simplicity. The study of [51] showed an implementation of clustering-based pruning with k-means clustering approach. They increased the number of clusters k stage by stage until the diversity among the clusters of individual learners become worse than before. The diversity among the clusters of individual learners is refer to the level of similarity between the cluster centres.

In the second step, this study also showed a strategy for pruning the ensemble by removing individual learners inside each of clusters. This approach breaks the task down into a number of steps: ordering the individual learners in an increasing way inside each cluster; start from the learner with least accurate to the most accurate; only removing the individual learner who cannot overcome the presupposed marginal value of disagreement between itself and the rest; repeat above steps until the accuracy of the pruned ensemble begins to decrease.

2.2.4 Optimization-based pruning

This kind of method treats ensemble pruning as an optimization problem. In general, working on an optimization problem is trying to find the best solution from all possible solutions. While, for ensemble pruning problems, it aims to find the best subset of the classifiers ensemble that will lead to better generalization performance. There are various optimization techniques mentioned by [4], such as heuristic optimization methods, mathematical methods and probabilistic methods. So, the optimization-based pruning methods can be divided into three classes according to above optimization approaches.

Take ***Heuristic optimization pruning*** for example, genetic algorithms are commonly used for solving heuristic pruning problems. Later, many researches were devoted to other novel techniques, such as greedy hill-climbing, artificial immune algorithms, case similarity search, etc [4]. In some cases, these heuristic optimization techniques are not practical or realizable. An efficient and feasible method named ***GASEN*** was introduced by Zhou et al. in 2002 [52] for heuristic optimization pruning. The key idea is to make a standard for individual learners, this standard is denoted by weights. Suppose there are a number of N individual learners in total, initializing an N-dimensional vector which contains each individual learner's weight. Whether an individual learner should be included depends on how much the weight is, a small weight denotes that the corresponding learner is potentially to be removed. One N-dimensional weight vector is a feasible solution for ensemble pruning.

There is a set of randomly initialized weight vectors in ***GASEN***, and it employs genetic algorithm to find the most appropriate weight vector according to the corresponding ensemble performance on validation set. Lastly, removing the individual learners with small weight until the performance of the pruned ensemble declines.

During the past decade, ***mathematical programming*** was proposed for solving machine learning problems. Ensemble pruning based on mathematical programming was suggested by [4]. Unlike the previous heuristic approach, the study [56] formulated ensemble pruning as a quadratic integer programming problem to find a subset of individual learners. In order to get the optimal accuracy-diversity trade-off, they proposed an effective solution by Semi-Definite Programming (SDP). Suppose N individual learners and m training instances, using a matrix P to record errors of each individual learner on the given instances:

$$P_{ij} = \begin{cases} 0 & \text{if } h_j \text{ classifies } \boldsymbol{x}_i \text{ correctly} \\ 1 & \text{otherwise.} \end{cases} \tag{2.15}$$

Then, initialize a matrix G, and let G=$P^T P$. The diagonal term of the matrix G is denoted as G*ii*, which is used to present the total number of mistakes made by individual learner *i*. And the number of common mistakes made by individual learners *i* and *j* is denoted by off-diagonal term G*ij*. A normalized matrix G can be presented as

$$\tilde{G}_{ij} = \begin{cases} \frac{G_{ij}}{m} & i = j \\ \frac{1}{2}\left(\frac{G_{ij}}{G_{ii}} + \frac{G_{ji}}{G_{jj}}\right) & i \neq j\ . \end{cases} \tag{2.16}$$

Thus, all elements in above matrix are enclosed within [0 ,1]. And the normalized element $\tilde{G}_{ii}$ is the error rate of individual learner *i*. So, $\sum_{i=1}^{N} \tilde{G}_{ii}$ is used to measure the overall performance of the ensemble. Whereas, $\sum_{i,j=1;i\neq j}^{N} \tilde{G}_{ij}$ is used to measure the diversity of the ensemble. Therefore, the sum of the above two terms can be used appropriately to measure the ensemble error. In the next step, the pruning problem is formulated as the quadratic integer programming problem.

$$\min_{\boldsymbol{z}} \ \boldsymbol{z}^{\top} \tilde{\mathbf{G}} \boldsymbol{z} \quad \text{s.t.} \sum_{i=1}^{N} z_i = T,\ z_i \in \{0,1\}\ , \tag{2.17}$$

Where the binary variable Z_i is used to determine whether the individual learner *i* will be selected, and T is the number of individual learners involved in pruned ensemble. Since the computation of (2.17) is generally NP-hard, a more efficient way to compute this optimization problem is given by [12]. Let $V_i = 2Z_i - 1 \in \{-1,1\}$,

$$\mathbf{V} = \boldsymbol{v}\boldsymbol{v}^\top, \quad \mathbf{H} = \begin{pmatrix} 1^\top \tilde{\mathbf{G}} 1 & 1^\top \tilde{\mathbf{G}} \\ \tilde{\mathbf{G}} 1 & \tilde{\mathbf{G}} \end{pmatrix}, \text{ and } \mathbf{D} = \begin{pmatrix} N & 1^\top \\ 1 & \mathbf{I} \end{pmatrix}, \tag{2.18}$$

where an all-one column vector is denoted by 1, and the identify matrix is represented by **I**, then (2.17) can be rewritten as

$$\begin{aligned} \min_{\mathbf{V}} \quad & \mathbf{H} \otimes \mathbf{V} \\ \text{s.t.} \quad & \mathbf{D} \otimes \mathbf{V} = 4T, \ \mathrm{diag}(\mathbf{V}) = 1, \ \mathbf{V} \succeq 0 \\ & \mathrm{rank}(\mathbf{V}) = 1 \,, \end{aligned} \tag{2.19}$$

Now, the problem is relaxed into the following convex SDP problem by dropping the rank constraint.

$$\begin{aligned} \min_{\mathbf{V}} \quad & \mathbf{H} \otimes \mathbf{V} \\ \text{s.t.} \quad & \mathbf{D} \otimes \mathbf{V} = 4T, \ \mathrm{diag}(\mathbf{V}) = 1, \ \mathbf{V} \succeq 0 \,. \end{aligned} \tag{2.20}$$

The idea about ***probabilistic pruning*** is also employing the concept of weights. This pruning method is based on Bayesian framework and expectation-maximization (EM) algorithm. By introducing a sparsity-inducing prior over the combination weight vector w, in order to lead an approximation of the sparse weight vector, EM algorithm can be employed to generate an MAP solution. Where the MAP is the Maximum a posteriori (MAP) estimation of weights [57]. The output of a classifier *i* on the instance x is denoted as $h_i(x)$. Suppose there are N individual classifiers $h_1, \ldots, h_N$, and then, the output vector of N individual classifiers on x is denoted by $h(x) = (h_1(x). \ldots h_N(x))^T$. The output of the ensemble contains all individual classifiers' output with a sparse weight vector $\boldsymbol{w}$ as $H(x) = \boldsymbol{w}^T h(x)$, where $\boldsymbol{w} = [\boldsymbol{w_1}, \ldots, \boldsymbol{w_N}]^T$ is a non-negative weight vector. To make the weight vector w sparse and non-negative, a left-truncated Gaussian prior will be adopted for w. Owing to the sparsity-inducing prior, many of the posteriors of weights are sharply distributed at zero, which results in pruning those irrelevant individual learners.

3

Algorithm Design

3.1 Regular Bagging

Our novel algorithm aims to make an improvement on the regular Bagging, in other words, it a modified version of the traditional Bagging algorithm. Bagging came from the abbreviation of *Bootstrap Aggregating* [12], which was first introduced by Breiman in 1996. As the name implies, the core idea behind Bagging is the combination of bootstrap and aggregation. Simply, the first step of Bagging is to build the ensemble by training multiple base learners on bootstrap replicates of the original dataset, and then, the final outputs are combined by majority voting.

As mentioned in chapter 2, ensemble diversity leads to a dramatic decrease of generalization errors, therefore, we want to get the members of an ensemble as independent as possible. Given a training dataset, one possible way is to train a base learner on each of the subsets, which are sampled non-overlapped data obtained from the original training set. However, since we cannot afford an infinite labelled training set in practice, such a process will produce very small and unrepresentative samples, leading to poor performance of base learners. Fortunately, we can get individual learners with less dependence by introducing randomness into the learning process, in which the individual learners are trained on a number of subsets sampled with replacement from the original training set (bootstrap resampling) [61], each subset has the same size as the original one.

After making sure of using the variations of training sets, it is also necessary to prevent the resultant ensemble from using stable learners, such as k-nearest neighbour. The individual learners should be *unstable*, that is, small changes in the training set should lead to large changes in the prediction outputs. Otherwise, the combination will be a range of almost identical learners. Two famous examples of unstable learners are neural networks and decision trees.

For Bagging, the most popular combination strategies such as voting and averaging can be used for aggregating the outputs of the individual learners. For example, in classification problems, Bagging feeds each instance to its individual classifiers and collects all of their outputs, and takes the label with majority votes as the final prediction. The Bagging algorithm pseudo code is summarised in Figure 3.1.

Bagging

Input: Data set $\boldsymbol{D} = \{(\boldsymbol{x_1}, \boldsymbol{y_1}), (\boldsymbol{x_2}, \boldsymbol{y_2}), \ldots, (\boldsymbol{x_m}, \boldsymbol{y_m})\}$;

Base learning algorithm **£;**

The number of base learners **T**

The ensemble E = ∅

Process:

1. **For** t = 1, . . . , T:

2. **£(**$\mathbf{C_t}$, $\mathbf{D_{bs}}$), build a learner $\mathbf{C_t}$ using $\mathbf{D_{bs}}$ as the training set. % $\mathbf{D_{bs}}$ is a bootstrap distribution.

3. Add the learner to the current ensemble, **E** = **E U** $\mathbf{C_t}$.

4. **End.**

Output:

Return ensemble **E**. And the final prediction: majority vote for classification, averaging for regression.

FIGURE 3.1: The Bagging algorithm.

Notice that Bagging are mostly applied for dealing with binary classification as well as multi-class classification problems. In our research, we proposed the variations of regular Bagging that focus on binary classification problems.

3.2 Risk in Bagging

Bagging aims at developing independent learners by taking bootstrap replicates as the training sets. However, it is practically impossible to achieve this purpose, since the sample subsets are drawn from the same training dataset. As in fact, these samples are pseudo-independent, even if they were taken independently from the distribution of the problem, the learners built on these training sets would bring about a potential risk that the outputs might not independent.

The study of [26] illustrates a conclusion about the correlation and the testing error rates by applying Bootstrap of Bagging and real independent sampling respectively. The Figure 3.2 (a) shows the average correlation among the base learners VS. the ensemble size. The correlation between individual learners build on true independent samples is lower than that generated by Bootstrap method. Figure 3.2 (b) shows the testing error rates VS. the ensemble size. It indicates the improved testing error rates by using really independent training sets, compared to using Bootstrap samples.

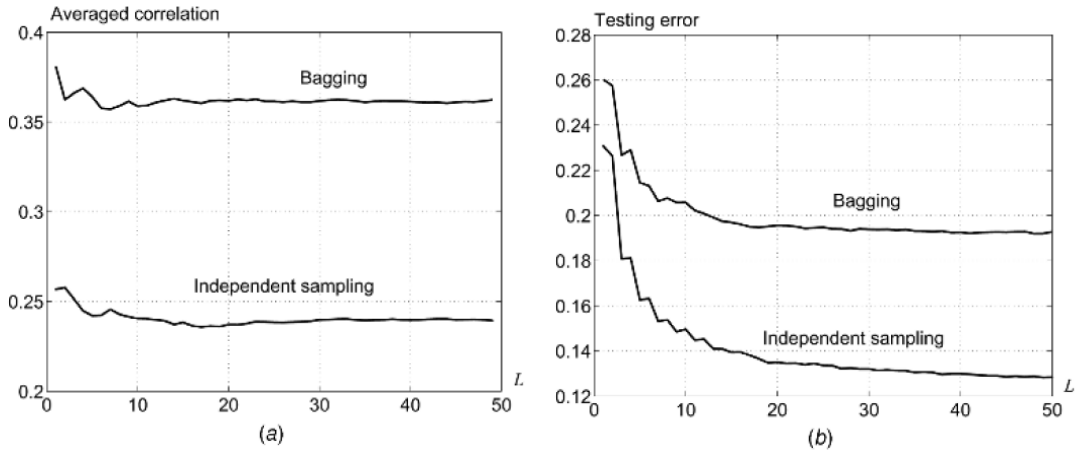

FIGURE 3.2: Bagging for the rotated check-board data using bootstrap samples and independent samples: (a) averaged pairwise correlation versus the ensemble size; (b) testing error rate versus the ensemble size.

Notice that the correlation for the true independent samples is not zero, which means we probably cannot avoid this issue at all, but it is possible to reduce the correlation by increasing diversity. In addition to the conclusion of above experiments, the individual learners solve approximately the same problem, the final success of bagging is the effect of the majority vote combiner (suppose doing classification problems). More specifically, given 1000 trained base classifiers, if the correlation is high among those classifiers and suppose 600 of them give the same outputs, once they all predicted in a

wrong direction, the bad prediction output has a higher frequency than others and therefore win the votes. In this case, the majority vote goes to a wrong direction, since we give an equal weight to each member of the ensemble. In our proposed version, we attempted to use the clustering method for improving the voting combiner, a simple reason is that selecting base classifiers from various groups can dramatically increase the diversity and reduce the risk in false majority vote.

Note that the accuracy and the diversity will not be affected any longer once the ensemble size has become large enough. From this point, it would be better to refine the current ensemble by reducing the number of members, this was the argument "MCBTA" (many could be better than all) which mentioned in section 2.2.1. Another reason of deploying a clustering technique in our proposed algorithm is that the benefit of ensemble pruning enhances the efficiency by reducing both computational resources and storage resources.

3.3 Proposed Algorithms

3.3.1 Clustering

There is a common scenario that the similarity among individual members' outputs is high, thus the final prediction based on the majority vote seems likely to be wrong in this situation. Our goal is to find a way that reduce the similarity of individual learners.

Generally, clustering aims to find the inherent structure of the unbalanced observations by grouping then into clusters of objects [64]. Clustering gives a shortcut to look insight into the hidden patterns of the data, properly using the clustering will maximize the difference among the clusters and maximize the similarity among members within each cluster.

Firstly, we generally have a look at the result after applying a clustering method on a dataset. Formally, given the dataset $\boldsymbol{D} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_m}\}$ where the $\boldsymbol{i}$th example is $\boldsymbol{x_i} = (\boldsymbol{x_{i1}}, \boldsymbol{x_{i2}}, \dots, \boldsymbol{x_{id}})^{\mathrm{T}} \in \boldsymbol{R^d}$ is a d-dimensional feature vector, using clustering method to group $\boldsymbol{D}$ into k disjoint clusters $\{\boldsymbol{C_j} \mid \boldsymbol{j = 1, \dots, k}\}$ with $\mathbf{U}_{\mathbf{j=1}}^{\mathbf{k}}\boldsymbol{C_j} = \boldsymbol{D}$ and $\boldsymbol{C_i} \cap_{\mathrm{i}\neq\mathrm{j}} \boldsymbol{C_j} = \boldsymbol{\emptyset}$. The clustering results returned by a clustering algorithm L can be represented as a label vector $\boldsymbol{\lambda} \in \boldsymbol{N^m}$, with the $\boldsymbol{i}$th element $\boldsymbol{\lambda_i} \in \{\boldsymbol{1, \dots, k}\}$ indicating the cluster **assignment** of $\boldsymbol{x_i}$.

3.3.2 k-means clustering

Although there are diverse categories of clustering methods have been proposed for different purposes, such as dealing with different data types, addressing unbalanced data problems, etc. In our research, we picked the k-means clustering method [62] as our solution, since it is a fast and efficient in terms of computational cost. In addition, it practically works well even when some assumptions are broken, and it is easy to interpret the clustering results.

K-means clustering is a type of unsupervised learning algorithms, which is used when you have unlabelled data, where the data is without defined categories or groups. This algorithm aims to find a number of K groups, and it works interactively to assign each instance to one of K groups based on the similarity. More specifically, it randomly selects k instances from the dataset ***D*** as the initial cluster centres. Then, for each instance in the rest is assigned to the cluster whose centre is the nearest. After that, the cluster centres will be updated and then resigning the instances to their nearest clusters. These two steps will be repeated until convergence. A more readable pseudo code of k-means clustering is summarised in Figure 3.3.

Input: ***k*** (the number of clusters),

D (data points without labels)

Output: a set of clusters

Method:

Arbitrarily choose ***k*** objects from ***D*** as the initial cluster centres;

Repeat:

1. (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;

2. Update the cluster means, i.e., calculate the mean value of the objects for each cluster

Until no change;

FIGURE 3.3: K-means clustering algorithm.

3.3.3 Version 1 – Clustering Bagging

We proposed a novel algorithm which aims to facilitate the performance of regular Bagging by using the unsupervised learning technique - k-means clustering. Notice that our algorithm should be distinct from clustering ensembles, also called cluster ensembles or consensus clustering, are a kind of ensemble whose base learners are clusters, generated by clustering methods [63]. Our research still focused on supervised learning tasks (classification problems), as the name implies, clustering Bagging (**C-Bagging** as a short name) indicates that it introduces the idea of unsupervised learning to refine the regular Bagging method. And the core functionality of the idea is to deploy k-means clustering after combining the ensemble, and then selecting a subset from the individual classifiers to form a new small ensemble.

Our algorithm C-Bagging is based on the *weka.classifiers.meta.Bagging*, thus the training process is exactly the same as the regular Bagging. We kept the core part of the original Bagging and trimmed the unnecessary pieces. In details, C-Bagging uses multiple bootstrap samples to train base classifiers independently, and then testing them by using 10-cross validation technique instead of an entirely new test set.

Suppose there is a multi-classes dataset, where the number of classes is 4, then the prediction result of ***i***th instance classified by ***j***th classifier is denoted as a probability vector of class labels, i.e., [0.17, 0.2, 0.1, 0.53], where the sum of vector members should be equal to 1. Notice that the C-Bagging is designed to mainly address binary-classification problems, in this scenario, the probability vector should be look like [a, b], and a + b = 1. Thereafter, we created multiple 2-demensional arrays which are used for housing the matrix of prediction results. More specifically, each row represents an individual classifier, and each column represents the first probability (a) of an instance, each array can be simple looked as a data point. The reason why we only select probability "a" from the vector [a, b] is because of making it easier for later clustering process. Implementing a procedure of k-means clustering for grouping trained base classifiers by using those probability arrays rather than classifiers themselves. Lastly, C-Bagging choose only one classifier from each group to comprise a new ensemble, where the chosen classifiers are nearest to their cluster centroids in each group. Therefore, the final prediction for each instance is the majority vote of class label based on the new ensemble. The C-Bagging algorithm is summarized in Figure 3.4.

C-Bagging

Input: Data set $\boldsymbol{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$;

Base learning algorithm **£;**

The number of clusters **K** (default value is sqrt(**T**));

Ensemble **E**.

Training phase:

1. Initialize the parameters

- **£** = decision trees (***REPTree***).
- **E** = ∅.
- **T**, the number of base classifiers to train (num of iterations).

2. For k = 1, … , T:

- Take a bootstrap sample $\mathbf{D_{bs}}$ from **D**.
- **£**($\mathbf{C_k}$, $\mathbf{D_{bs}}$), build a classifier $\mathbf{C_k}$ using $\mathbf{D_{bs}}$ as the training set.
- Add the classifier to the current ensemble, $\mathbf{E} = \mathbf{E} \cup \mathbf{C_k}$.

Classification phase:

4. Clustering:

- Create a matrix for housing the prediction probabilities (testing on **D**).
- Using k-means clustering to group the base classifiers into multiple clusters

5. For m = 1, … , K:

- Choose the classifier $\mathbf{C_m}$ which is nearest to its centre
- Add the classifier to a subset $\mathbf{E_S} = \mathbf{E_S} \cup \mathbf{C_m}$.

6. Run each classifier of $\mathbf{E_S}$ on the input **x**.

Output:

7. ***Return*** $\mathbf{E} = \mathbf{E} \cap \mathbf{E_S}$.

8. The class with the maximum number of the votes is chosen as the label for **x**.

FIGURE 3.4: The C-Bagging algorithm.

It is worth mentioning that the number of cluster K is set to a default value which is equal to the square root of the T (the number of base classifiers). However, it is not such efficient for us to figure out what the best choice of the K is, while the K directly affects the performance of ensembles. As in fact, finding the most appropriate K is the primary problem in the later experiments, in order to test and make comparisons of the accuracy with various K values, we added an extra option (-K) for setting an arbitrary number of K.

3.3.4 Version 2 – Clustering BaggingHO

There might have a limitation of the first version algorithm (C-Bagging), since the training set and testing set are from the same dataset, this can cause the problems in later evaluation. In fact, the data points that used for clustering are the prediction results tested from the same dataset, which has been used for training. Thus, the trained base classifiers might not independent or they have already been familiar with the dataset. This would cause overfitting and make it hard for the clustering process, since the data points (probability vectors) are very centralized.

C-BaggingHO

Input: Data set **D** = {(x_1, y_1), (x_2, y_2), … , (x_m,y_m)}, $\mathbf{D_1}$ = 90% of **D**, $\mathbf{D_2}$ = 10% of **D;**

Base learning algorithm **£;**

The number of clusters **K** (default value is sqrt(**T**));

Ensemble **E**.

Training phase:

1. Initialize the parameters

- **£** = decision trees (***REPTree***).
- **E** = ∅.
- **T**, the number of base classifiers to train (num of iterations).

2. For k = 1, … , T:

- Take a bootstrap sample $\mathbf{D_{bs}}$ from $\mathbf{D_1}$.
- **£(**$\mathbf{C_k}$, $\mathbf{D_{bs}}$), build a classifier $\mathbf{C_k}$ using $\mathbf{D_{bs}}$ as the training set.
- Add the classifier to the current ensemble, $\mathbf{E} = \mathbf{E} \cup \mathbf{C_k}$.

Classification phase:

4. Clustering:

- Pre-test on $\mathbf{D_2}$
- Create a matrix for housing the prediction probabilities.
- Using k-means clustering to group the base classifiers into **K** clusters

5. For m = 1, … , K:

- Choose the classifier $\mathbf{C_m}$ which is nearest to its centre
- Add the classifier to a subset $\mathbf{E_S} = \mathbf{E_S} \cup \mathbf{C_m}$.

6. Run each classifier of $\mathbf{E_S}$ on the input **x**.

Output:

7. ***Return*** $\mathbf{E} = \mathbf{E} \cap \mathbf{E_S}$.

8. The class with the maximum number of the votes is chosen as the label for **x**.

FIGURE 3.5: The C-BaggingHO algorithm.

It does make sense to hold out a small part of the dataset for clustering process, in default, we split the dataset as the 90% for training and 10% for the pre-test before clustering. The second version algorithm is called Clustering Bagging with hold-out or **C-BaggingHO** for short, it is summarized in the Figure 3.5.

3.3.5 Version 3 – BaggingRC

BaggingRC

Input: Data set **D** = {(x_1, y_1), (x_2, y_2), … , (x_m,y_m)};

Base learning algorithm **£;**

The number of clusters **K** (default value is sqrt(**T**));

Ensemble **E**.

Training phase:

1. Initialize the parameters

- **£** = decision trees (***REPTree***).
- **E** = ∅.
- **T**, the number of base classifiers to train (num of iterations).

2. For k = 1, … , T:

- Take a bootstrap sample $\mathbf{D_{bs}}$ from **D**.
- **£**($\mathbf{C_k}$, $\mathbf{D_{bs}}$), build a classifier $\mathbf{C_k}$ using $\mathbf{D_{bs}}$ as the training set.
- Add the classifier to the current ensemble, $\mathbf{E} = \mathbf{E} \cup \mathbf{C_k}$.

Classification phase:

4. Clustering:

- Generating random dataset **RD**.
- Validation on **RD.**
- Create a matrix for housing the prediction probabilities.
- Using k-means clustering to group the base classifiers into **K** clusters

5. For m = 1, … , K:

- Choose the classifier $\mathbf{C_m}$ which is nearest to its centre
- Add the classifier to a subset $\mathbf{E_S} = \mathbf{E_S} \cup \mathbf{C_m}$.

6. Run each classifier of $\mathbf{E_S}$ on the input **x**.

Output:

7. ***Return*** $\mathbf{E} = \mathbf{E} \cap \mathbf{E_S}$.

8. The class with the maximum number of the votes is chosen as the label for **x**.

FIGURE 3.6: The BaggingRC algorithm.

Although the hold-out method seems to be helpful in validation with a fully independent dataset, but the performance evaluation is subject to the size of data. Especially for the smaller datasets, taking away 10% of the training might hurt the accuracy. So, it would be interest to find an alternative approach, we attempted to use some artificial random data for the calibration. We proposed the third version algorithm that is called **BaggingRC**, where **RC** encodes random data and clustering. BaggingRC implements a simple randomiser, that takes a dataset and generates random instances by randomly picking values from the dataset. In addition, the random data must have the same structure (number of attributes, and attribute types, etc). The third version algorithm is summarized in Figure 3.6.

It is worth mentioning that the bootstrap sampling also offers Bagging another advantage. As the study of [12] indicated, given a training dataset which contains k observation examples, some original examples appear more than once, while some original examples are not present in the sample. Each example has the same probability to be sampled, thus the probability that an arbitrary example is selected 0, 1, 2, ... times is approximately Poisson distributed with $\lambda=1$. In other words, there is $1-(1/e) \approx 0.632$ of chance that an example will occur at least once, or say, there are around 36.8% original training examples that have never been used in training process. This small part of the training dataset is called ***out-of-bag***, which is used for estimating the goodness of base learners, and thereafter the generalization error of the ensemble can be estimated.

4

Experiments

4.1 WEKA- Data Mining Software in Java

The WEKA workbench is a collection of machine learning algorithms and data pre-processing tools that used for data mining tasks. WEKA enables to deal with main data mining problems: regression, classification, clustering, attribute selection, etc. Generally, it provides a range of functionalities that almost support the whole process of experimental data mining, including preparing the input data, evaluating learning algorithms statistically, and visualizing the input data and the test output of learning. Simply, you can feed your own dataset into a learning scheme and analyse the resulting classifier and its performance, and it is also well-suited for developing new machine learning schemes. The algorithms can either be applied directly to a dataset or called from your own Java code. In our experiments, we deploy the proposed algorithms in the WEKA and then evaluate them on a variety of datasets.

There are several kinds of graphical user interfaces (GUI) within WEKA, the easiest way to use it is through a GUI called *Explorer*. This allows you quickly to load a dataset from a file or randomly generating a set of data by itself, using *Filter* tool for pre-processing the data. And then, selecting any learning algorithm you want from the WEKA interface menu, evaluating it on your dataset. In addition, you can also use the *Explorer* to look insight into the data, visualising the details of data such as the total number of instances, the number of attributes, missing values, type of attribute value, etc.

Another WEKA interface, the *Experimenter*, is designed to help you find the method that best addresses the given problems. In practice, there is usually no way to answer which method is the best, while WEKA provides an environment that enables users to compare a variety of learning techniques. The *Experimenter* allows users to automate the process by making it easy to run classifiers and filters with different parameter settings on a corpus of datasets, collect performance statistics, and perform significance tests. Our experiments totally rely on this interface, by feeding various datasets into our proposed algorithms, which can be called from our own code. The WEKA GUI Chooser panel is shown in Figure 4.1.

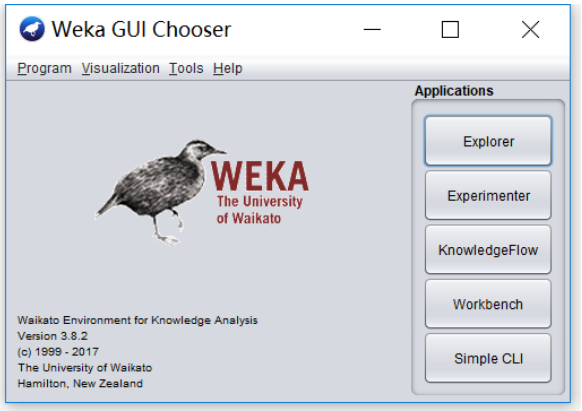

FIGURE 4.1: The WEKA GUI Chooser.

4.2 K-fold Cross-Validation

The success of finding a robust model for making predictions on unseen data is not only about using which machine learning techniques but also refers to data itself. It is not reliable to evaluate the model on the same dataset which has been used for training, since the performance would be misleadingly optimistic. This situation is called *overfitting*. More specifically, learning the parameters of a prediction function and testing it on the same data is a methodological mistake, a model that would just repeat the labels of the samples that it has just seen. This would have a perfect score but would fail to predict anything useful on yet unseen data.

To avoid it, it is common practice when performing a machine learning experiment to hold out part of the available data as a test set. But it is not suitable for smaller datasets. K-fold cross-validation is a systematic way of doing hold-out process multiple times, thus it dramatically decreases the variance of measures. Take 10-fold cross-validation for example, it divides the training set into 10 subsets and trains the model 10 times, each time leaving a different subset out of the training data and using it instead as a validation set. At the end, the performance metric is averaged across all 10 tests. Lastly, as before, once the best parameter combination has been found, the model is retrained on the full data.

In practice, obtaining another fully independent testing set is expensive and time-consuming, even though the K-fold cross-validation method needs higher computational costs, but it is very helpful in smaller datasets. There is a certain amount of variance in an evaluation, because it is all statistical underneath. To keep the variance of the estimation as low as possible, we applied the K-fold cross-validation technique in our experiments.

4.3 Explore Datasets

As mentioned before, the Weka API has a great ability to explore, summarize and visualize data. The goal of our experiments is to evaluate whether our proposed algorithms achieve a better performance than the regular Bagging. Although some example datasets are included in the Weka distribution, but for binary-classification problems, some of those seem to be unsuitable. In order to implement the algorithms on a wide range of datasets, we downloaded a set of useful classification datasets from the UCI Machine Learning repository. Eventually, we selected 15 datasets that involves various problem domains, such as medical, food, election, game, credit, etc.

In addition, it is also necessary to evaluate them on datasets with diverse sizes. It aims to find the data size, for which the proposed algorithms are more suitable. Thus the 15 selected datasets include small size data as well as large size data. For example, the ***labor*** dataset is the smallest one of all, this dataset was used to learn the description of an acceptable and unacceptable contract and it contains only 57 instances. While the ***mushroom*** dataset includes 8124 instances and it is the biggest one that was used to describe whether a species of mushroom is edible or poisonous.

Moreover, whether a dataset has missing attribute values is another determinant of the achievement. In the learning process, missing values may cause a loss of knowledge. Especially for the clustering process during running our algorithms, the influence of pattern of missing data would lead to a reduction of robustness of the clusters. The real-world datasets are almost always burdened with missing values.

The goal of experiments is to compare our proposed algorithms with the regular Bagging, but beyond that it is also interested to explore the sensitivity of missing values for our algorithms. The analysis of [65] has proven that the data quality is important to classification models' induction, and a common quality problem relates to lack of complete data which can be cause by many reasons. In addition, their empirical analysis shows that the missing values on datasets do have an influence on solving classification problems, the classification error rises after significant insertion of missing values in ten UCI datasets tested. The following histogram (Figure 4.2) shows the proportion of missing values in each of our chosen datasets.

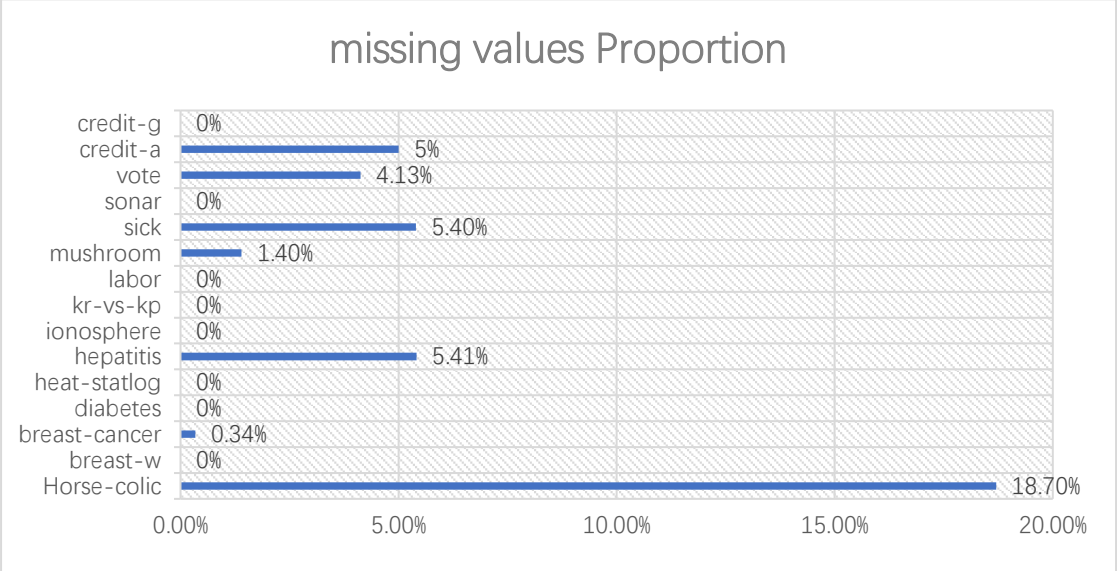

FIGURE 4.2: Missing values proportion vs. datasets.

Dimensionality in data mining or statistic refers to how many attributes a dataset has, if a dataset has many feature variables, we call it high-dimensional data. For example, medical data is notorious for having vast amounts of variables (e.g. heart rate, blood pressure, blood glucose, weight, cholesterol level, etc). Another famous high dimensional data example is microarrays, which measure gene expression, and the number of attributes can exceed the number of observations.

Higher dimensional data is very commonly seen in machine learning datasets, and our 15 chosen datasets have a variety of dimensions, some of them have a relatively higher dimension than others. The Figure 4.3 illustrates a summary of dimensionalities of our datasets. Intuitively, a higher dimensional data will cause problems, such as increasing the computational burden or reducing the predictive power. The more dimensions you add to a dataset, the more difficult it becomes to predict certain quantities. Our experiments are also trying to find out whether the proposed algorithms can perform better than the regular Bagging on a higher dimensional data. Or, for what dimension the dataset is, our proposed algorithms have the greatest improvement.

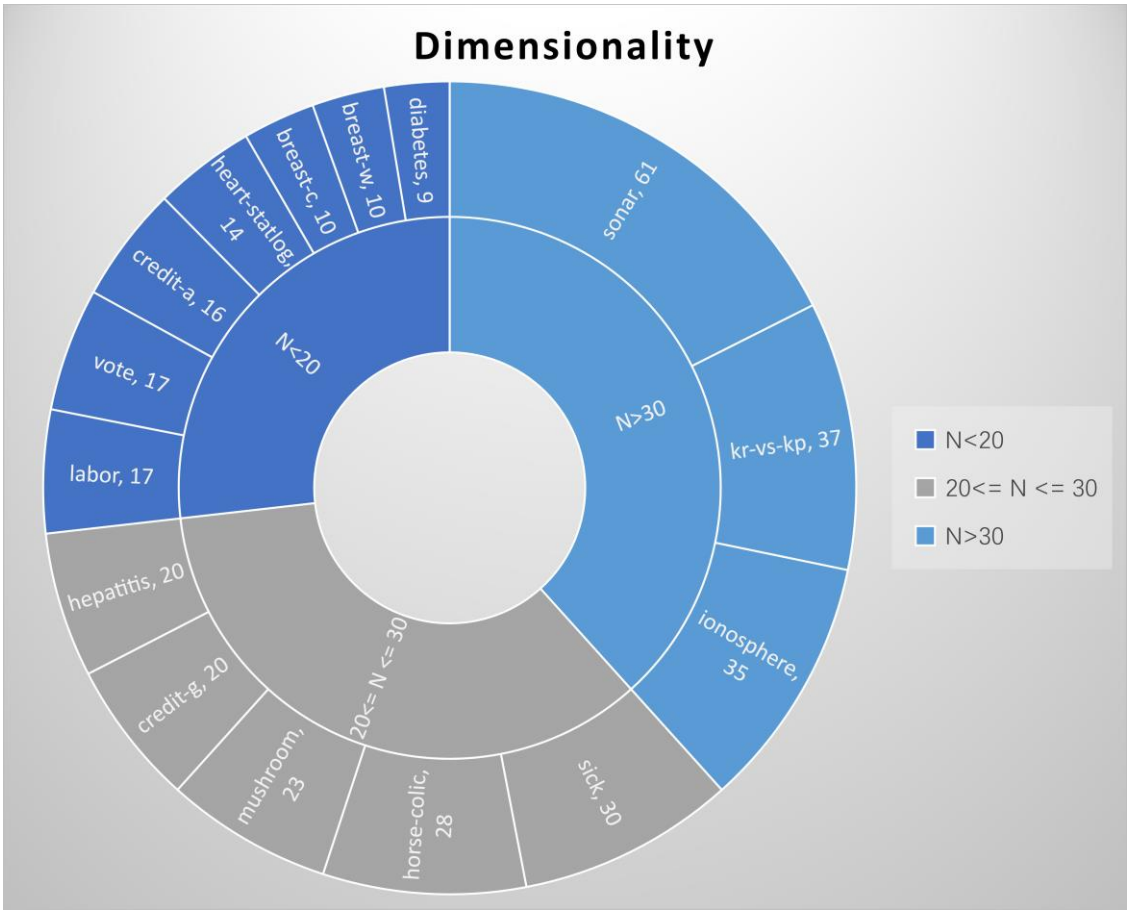

FIGURE 4.3: A summary of dimensionality (including class attribute).

Note that our experiments are trying to solve classification problems. Obviously, the class attributes in datasets should be categorical. However, some of our datasets are mixed, which means they consist of numeric features as well as categorical features. Generally, it is better to deploy a form of data transformation that promise to solve the classification problems more properly. And many empirical analyses have proven that the non-linear classifier such as tree-based classifiers are better at dealing with mixed data.

For real-world data, it is hard to avoid having mixed features, for example, a dataset records about buying pattern of customers, age is a numeric feature while gender and the type of goods are categorical or text data, etc. Our datasets refer to a range of areas in practice, it is the same situation as above. The following experiments are also trying to evaluate whether the proposed algorithms can handle this kind of issue better than the regular Bagging. We do not change anything of our datasets for the moment, and the Figure 4.4 shows a summarized structure of features for each dataset.

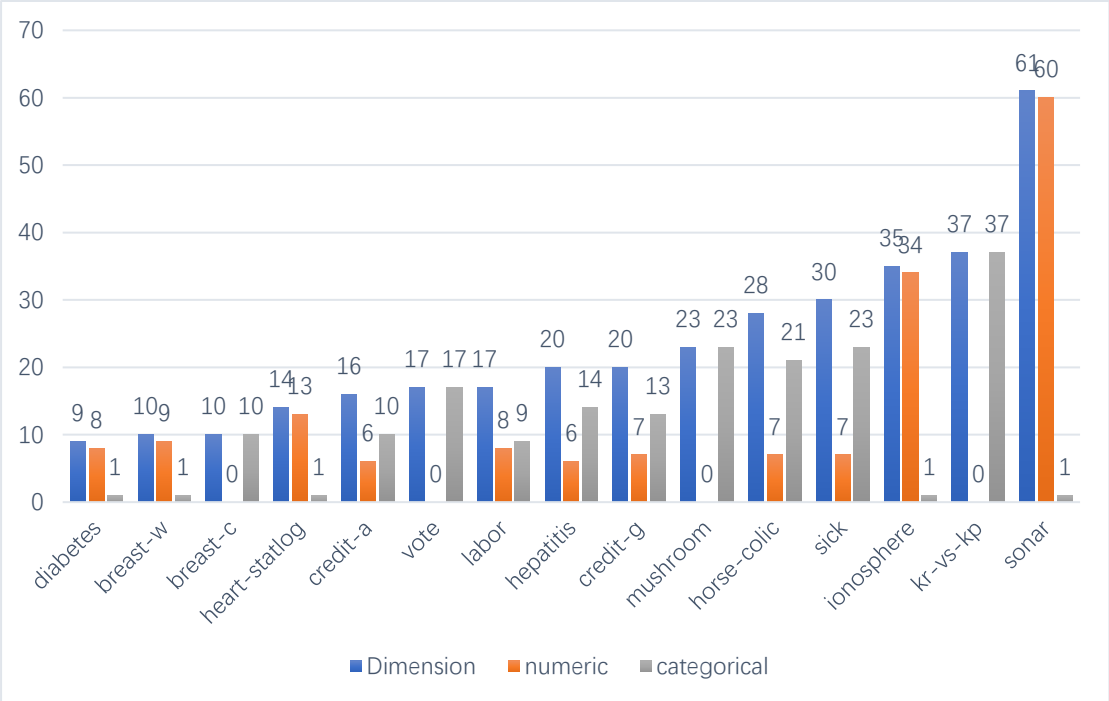

FIGURE 4.4: The feature structure of datasets.

4.4 Experiments and Results

For all experiments scenarios, implementing four algorithms on 15 chosen datasets by using *Weka Experiments Environment*. In this case, we deployed the options as follows: set the number of iterations to 10 for the regular Bagging algorithm while 100 iterations for the three proposed algorithms, in the meanwhile, set the -K option to be 10 (-K option for setting the number of clusters). Thereafter, we can evaluate those methods on the same level, since all ensembles have the same size which is equal to 10 (each ensemble consists of 10 individual classifiers). And we selected the REPTree as the base classifier for all algorithms.

In addition, we set the experiment type as cross-validation, and the number of folds is 10. In order to get a more robust result, we repeated the measure of each algorithm on each dataset for 10 times. Thus, the final result is the average of the 10 outputs, the details of outputs can be saved as a CSV document for later work. The following Figure 4.5 shows a spreadsheet example of the results.

Key_Datas	Key_Run	Key_Fold	Key_Scher	Key_Scher	Key_Scher	Date_time	Number_c	Number_c	Number_c	Number_i	Number_u	Percent_c
horse-coli	1	1	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	331	37	34	3	0	91.89189
horse-coli	1	2	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	331	37	34	3	0	91.89189
horse-coli	1	3	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	331	37	30	7	0	81.08108
horse-coli	1	4	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	331	37	33	4	0	89.18919
horse-coli	1	5	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	331	37	30	7	0	81.08108
horse-coli	1	6	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	331	37	33	4	0	89.18919
horse-coli	1	7	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	331	37	35	2	0	94.59459
horse-coli	1	8	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	331	37	31	6	0	83.78378
horse-coli	1	9	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	332	36	30	6	0	83.33333
horse-coli	1	10	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	332	36	28	8	0	77.77778
horse-coli	2	1	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	331	37	29	8	0	78.37838
horse-coli	2	2	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	331	37	31	6	0	83.78378
horse-coli	2	3	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	331	37	32	5	0	86.48649
horse-coli	2	4	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	331	37	32	5	0	86.48649
horse-coli	2	5	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	331	37	32	5	0	86.48649
horse-coli	2	6	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	331	37	34	3	0	91.89189
horse-coli	2	7	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	331	37	31	6	0	83.78378
horse-coli	2	8	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	331	37	30	7	0	81.08108
horse-coli	2	9	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	332	36	26	10	0	72.22222
horse-coli	2	10	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	332	36	30	6	0	83.33333
horse-coli	3	1	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	331	37	29	8	0	78.37838
horse-coli	3	2	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	331	37	31	6	0	83.78378
horse-coli	3	3	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	331	37	30	7	0	81.08108
horse-coli	3	4	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	331	37	32	5	0	86.48649
horse-coli	3	5	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	331	37	33	4	0	89.18919
horse-coli	3	6	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	331	37	31	6	0	83.78378
horse-coli	3	7	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	331	37	31	6	0	83.78378
horse-coli	3	8	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	331	37	31	6	0	83.78378
horse-coli	3	9	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	332	36	30	6	0	83.33333
horse-coli	3	10	weka.class	'-P 100 -S 1	-1.2E+17	2.02E+07	332	36	32	4	0	88.88889

FIGURE 4.5: A spreadsheet of prediction results.

The above table shows the first 3 repetitions of the first algorithm, there still a lot of information has not been shown in the above, but the most important part has been presented, such as dataset name, algorithm name, index of repetition, index of fold, prediction accuracy, etc. Notice that there are 10 folds results in each repetition, since we applied 10 cross-validation during the experiment. As in fact, the final prediction is derived from the 10*10=100 results. In order to make comparison between the algorithms more intuitive, the Figure 4.6 shows the final prediction accuracy of each algorithm for the selected 15 datasets.

```
Analysing:  Percent_correct
Datasets:   15
Resultsets: 4
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       18-5-9 12:57pm

Dataset                   (1) meta.Bagg | (2) meta. (3) meta. (4) meta.
------------------------------------------------------------------------
horse-colic             (100)    84.77 |   84.80     84.67      84.66
wisconsin-breast-cancer (100)    95.88 |   95.72     95.58      95.82
breast-cancer           (100)    68.41 |   69.12     68.51      69.53
pima_diabetes           (100)    75.86 |   76.23     75.69      75.84
heart-statlog           (100)    81.59 |   81.19     81.15      81.04
hepatitis               (100)    82.08 |   82.22     81.98      82.34
ionosphere              (100)    91.32 |   91.71     91.68      91.51
kr-vs-kp                (100)    99.01 |   99.04     98.98      99.06
labor                   (100)    84.93 |   85.33     83.60      84.53
mushroom                (100)   100.00 |   99.92 *   99.92     100.00
sick                    (100)    98.67 |   98.63     98.53      98.65
sonar                   (100)    76.80 |   76.60     75.50      76.58
vote                    (100)    95.68 |   95.45     95.63      95.63
credit-rating           (100)    85.14 |   85.41     85.52      85.39
german_credit           (100)    74.19 |   74.17     73.86      73.87
------------------------------------------------------------------------
                                 (v/ /*) |  (0/14/1)  (0/15/0)   (0/15/0)

Key:
(1) meta.Bagging '-P 100 -S 1 -num-slots 1 -I 10 -W trees.REPTree -- -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0' -115879962237199696
(2) meta.C-Bagging '-K 10 -S 1 -num-slots 1 -I 100 -W trees.REPTree -- -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0' -50587962237199704
(3) meta.C-BaggingHO '-K 10 -S 1 -num-slots 1 -I 100 -W trees.REPTree -- -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0' -50587962237199705
(4) meta.BaggingRC '-K 10 -S 1 -num-slots 1 -I 100 -W trees.REPTree -- -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0' -50587962237199706
```

FIGURE 4.6: Experiment result (ensemble with 10 base classifiers).

In this experiment, the three proposed algorithms selected 10 out of 100 base classifiers to form the final ensemble. Compared to the regular Bagging, it seems that they don't make much improvement. In some cases, they perform even a little bit worse than the regular Bagging method.

We assumed the cause of unexpected result was that the ensemble size too small, then we decided to redo the experiments by setting the number of clusters (-K) to be 30, selecting 30 classifiers from a 1000 individual classifiers pool. While for the regular Bagging, setting the number of iterations to be 30 as well. The Figure 4.7 presents the results that obtained by testing the ensemble with 30 base classifiers on the same range of datasets.

```
Analysing:  Percent_correct
Datasets:   15
Resultsets: 4
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       18-5-9 6:34pm

Dataset                   (1) meta.Bagg | (2) meta. (3) meta. (4) meta.
------------------------------------------------------------------------
horse-colic              (100)    84.85 |   84.58     84.85      84.58
wisconsin-breast-cancer  (100)    95.97 |   95.88     95.71      95.77
breast-cancer            (100)    69.23 |   69.59     68.84      69.63
pima_diabetes            (100)    76.05 |   76.11     76.16      76.18
heart-statlog            (100)    81.41 |   81.93     82.04      81.78
hepatitis                (100)    82.21 |   82.15     82.16      82.72
ionosphere               (100)    91.60 |   92.19     91.54      91.85
kr-vs-kp                 (100)    99.04 |   99.02     98.98      99.03
labor                    (100)    85.73 |   85.57     86.13      86.33
mushroom                 (100)   100.00 |   99.91 *   99.92 *   100.00
sick                     (100)    98.66 |   98.67     98.55      98.68
sonar                    (100)    76.98 |   77.09     76.53      76.23
vote                     (100)    95.58 |   95.40     95.54      95.54
credit-rating            (100)    85.70 |   85.58     85.45      85.42
german_credit            (100)    74.30 |   74.82     74.12      74.66
------------------------------------------------------------------------
                                 (v/ /*) |  (0/14/1)  (0/14/1)   (0/15/0)

Key:
(1) meta.Bagging '-P 100 -S 1 -num-slots 1 -I 30 -W trees.REPTree -- -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0' -115879962237199696
(2) meta.C-Bagging '-K 30 -S 1 -num-slots 1 -I 1000 -W trees.REPTree -- -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0' -50587962237199704
(3) meta.C-BaggingHO '-K 30 -S 1 -num-slots 1 -I 1000 -W trees.REPTree -- -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0' -50587962237199705
(4) meta.BaggingRC '-K 30 -S 1 -num-slots 1 -I 1000 -W trees.REPTree -- -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0' -50587962237199706
```

FIGURE 4.7: Experiment result (ensemble with 30 base classifiers).

According to the test output, we find that the overall performance of the ensemble methods with 30 base classifiers has been enhanced. While the difference between the proposed algorithms and the regular Bagging is still not significant. Only selecting 30 out of 1000 individual classifiers might cause a low representation for its coverage in community. Thus, we did another experiment that constructing bigger ensembles with 100/1000 individual classifiers. The test output was summarized in Figure 4.8.

```
Analysing:  Percent_correct
Datasets:   15
Resultsets: 4
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       18-5-13 2:09pm

Dataset                   (1) meta.Bagg | (2) meta. (3) meta. (4) meta.
------------------------------------------------------------------------
horse-colic              (100)    84.75 |   84.72     84.69      84.72
wisconsin-breast-cancer  (100)    96.08 |   96.00     95.97      95.94
breast-cancer            (100)    69.51 |   70.01     69.40      69.88
pima_diabetes            (100)    76.29 |   76.26     76.30      76.63
heart-statlog            (100)    82.30 |   82.19     82.56      82.48
hepatitis                (100)    81.45 |   81.82     82.30      81.77
ionosphere               (100)    91.94 |   92.31     91.77      92.20
kr-vs-kp                 (100)    99.03 |   99.03     99.00      99.01
labor                    (100)    85.50 |   86.07     88.07      87.20
mushroom                 (100)   100.00 |   99.91 *   99.91 *   100.00
sick                     (100)    98.66 |   98.68     98.56      98.69
sonar                    (100)    77.66 |   77.66     77.91      77.23
vote                     (100)    95.52 |   95.52     95.47      95.54
credit-rating            (100)    85.86 |   85.78     85.46      85.74
german_credit            (100)    74.58 |   74.92     74.93      74.86
------------------------------------------------------------------------
                                 (v/ /*) |  (0/14/1)  (0/14/1)   (0/15/0)

Key:
(1) meta.Bagging '-P 100 -S 1 -num-slots 1 -I 100 -W trees.REPTree -- -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0' -115879962237199696
(2) meta.C-Bagging '-K 100 -S 1 -num-slots 1 -I 1000 -W trees.REPTree -- -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0' -50587962237199704
(3) meta.C-BaggingHO '-K 100 -S 1 -num-slots 1 -I 1000 -W trees.REPTree -- -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0' -50587962237199705
(4) meta.BaggingRC '-K 100 -S 1 -num-slots 1 -I 1000 -W trees.REPTree -- -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0' -50587962237199706
```

FIGURE 4.8: Experiment result (ensemble with 100 base classifiers).

4.5 Summary of Findings

From all above experiments, we can conclude that there is a little improvement achieved by introducing clustering into the regular Bagging, but it is not significant or sufficient. The test output illustrates that whether the regular Bagging or our proposed algorithms has performed very well on the ***mushroom*** dataset, which has only 1.4% missing values. And it is a 23-dimensional dataset, all feature attributes are categorical, it seems to have no extra space for strengthening our algorithms, as the regular Bagging can get a 100% accuracy.

All algorithms have the worst performance on the ***breast-cancer*** dataset, their percentages of correct-classification are close to 70%, which is even worse than performing a single model. But this is confusing and difficult to understand or explain, as there is almost no missing value (0.34%) in this dataset, and it has a relatively lower dimensionality (10) with only categorical features. The only possible is that some essential features are missing during the process of training.

By comparing with other datasets, we noticed that some feature variables in this dataset are not purely categorical variables, while they are ordinal variables or interval variables. An ordinal variable is similar to a categorical variable. The difference between the two is that there is a clear ordering of the variables. For instance, assume there is a variable that represents the economic status, with three categories (low, medium, high). The interval variable is another type of the ordinal variable, the only difference between them is the values of the interval variables are equally spaced. We assume that the low prediction accuracy is due to the classifiers cannot fully understand those kinds of data, since there exists uneven space or an average of a categorical variable does not make much sense.

In terms of high dimensional data, the ***sonar*** dataset has the highest value of dimensionality. Obviously, the test results show that all algorithms perform not well on this data with a relatively low percent-correct (around 77%). It is essential to take a good control of data dimensionality, since it does have an influence on the performance of ensemble methods when it is large enough. We also noticed that a dataset with high level of missing values will hurt the accuracy. Among our 15 datasets, the ***horse-colic*** dataset is the one with the highest level of missing values. Its' 18.7% feature values are missing, which will result in insufficient training.

Moreover, we picked out all datasets on which we evaluated our algorithms with accuracy over 90%. We found that all of them have a lot in common, such as low level of missing values, low or medium dimensionality (no more than 40), and the feature structure is almost made of exclusively categorical or numeric variables. A summary is concluded in the Figure 4.9.

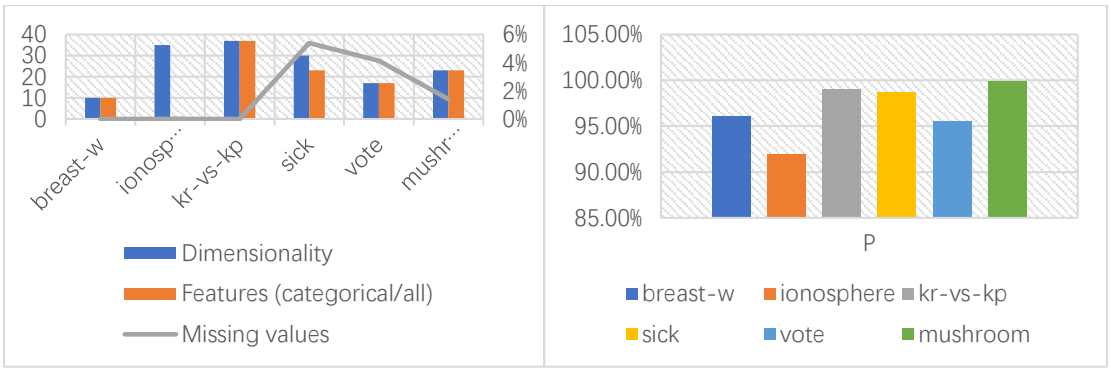

FIGURE 4.9: A summary of findings.

In addition to above findings, we need go back to our major question: does the clustering technique help in improving Bagging? Unfortunately, our experimental test output shows that there is no significant achievement. There are some advices and alternative approaches will be discussed in the future work section.

5

Conclusion

Compared to a single model, the ensemble methods are meta-algorithms that facilitate the improvement of accuracy and stability. And the ensemble methods have been successfully used in a variety of real-world tasks, such as computer vision, computer security, and computer medical diagnosis.

However, the efficiency of ensemble methods might be influenced by several aspects, which could be ensemble size, difficulty of generating diversity, ensemble combination method, ensemble selection, etc. The major purpose of our research is to find a way that can further improve the Bagging method. In the case of Bagging, diversity can be generated by bootstrap resampling with replacement. But for classification problems, if given a set of trained base classifiers, it may be better to combine some instead of all of them (many could be better than all). Thus, we proposed an algorithm that combines the regular Bagging with a pruning method, which is K-means clustering. The clustering technique can also help in increasing the diversity of ensembles. There is a risk in our first version algorithm, that is the training set and the validation set used for clustering are the same dataset. Then, we proposed a second version that involves hold-out technique, which puts aside a small part of data for clustering process.

Although the hold-out method works well, but the performance might be hurt and the evaluation is subject to the size of data. Therefore, we proposed the third version algorithm, which introduces the random artificial data, rather than using hold-out method. Notice that using random artificial data is another efficient way to increase ensemble diversity.

In our experiments, we selected 15 dataset that involves various problem domains, moreover, these datasets are diverse in many aspects: data size, dimensionality, feature types, missing value. Considering the difficulty of obtaining a fully independent test set, we used the 10-fold cross-validation technique in our experiments, since it is efficient to keep the variance of the evaluation as low as possible, especially suitable for small datasets. And our tests were totally performed on WEKA *Experimenter* GUI.

We repeated the test 10 times for running each algorithm on each dataset, and deployed different values of the -K option (the number of clusters) for three experiment scenarios as ensemble with size 10, 30, and 100 respectively. Overall, we found that all algorithms (including the regular Bagging) better performed on datasets which with low level of missing values, relative low dimensionality, and the feature variables are exclusively categorical or numeric. However, it is unexpected that our proposed algorithms do not achieve a significant improvement. It might be interesting to find other alternative ways for the continued perfection of our experiments, and the details will be summarized in the next chapter of future work.

6

Future Work

From the perspective of prediction accuracy, unfortunately, our experimental test output shows that there is no significant achievement. However, there are some alternative experiments that we can do for enriching our conclusion. Firstly, we have done several experiments with different ensemble size, but it is still possible to find a more suitable value of K (the number of clusters), as the best proper ensemble size will dramatically increase the performance. Secondly, we generally set the *REPTree* as the base classifier in default, however, it might be interesting to try other alternatives such as *RandomTree* or *J48 tree*. We did another two experiments with same configuring settings expect the base classifier, one is set as *J48 tree* (unpruned tree) and the other is *RandomTree*. The performance test outputs are shown in Appendix.

Furthermore, in our proposed algorithms, the final ensemble is comprised by a subset of individual classifiers after clustering, where each of them is nearest to its cluster centre. Also, there are two more alternative ways, one is randomly selecting a classifier from each cluster, and another is choosing the one with the best performance from each cluster. We hope that the research objective could be attained by these changes.

There is a limitation of the proposed algorithms, which are only designed for addressing binary-classification problems. We have only tried them on binary datasets, it would be also interesting to explore whether the regular Bagging can better deal with multi-classification or regression problems by using a pruning method.

Appendix:

Extra experiment output - ensemble with 100 base classifiers (J48 Unpruned tree):

```
Analysing:  Percent_correct
Datasets:   15
Resultsets: 4
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       18-5-29

Dataset                 (1) meta.Bagg | (2) meta. (3) meta. (4) meta.
-------------------------------------------------------------------------
horse-colic               (100)   85.10 |   85.81    84.72    85.59
wisconsin-breast-cancer   (100)   96.37 |   96.28    96.28    96.31
breast-cancer             (100)   70.85 |   71.75    71.14    71.82
pima_diabetes             (100)   75.52 |   76.02    76.20    76.41
heart-statlog             (100)   81.93 |   82.07    81.19    82.00
hepatitis                 (100)   82.98 |   82.72    81.69    83.22
ionosphere                (100)   92.94 |   92.77    92.60    93.00
kr-vs-kp                  (100)   99.44 |   99.44    99.37    99.44
labor                     (100)   88.20 |   87.60    87.93    86.87
mushroom                  (100)  100.00 |   99.92 *  99.96   100.00
sick                      (100)   98.93 |   98.99    98.80    98.97
sonar                     (100)   80.04 |   78.70    77.93    79.07
vote                      (100)   96.91 |   96.87    96.68    96.73
credit-rating             (100)   86.55 |   86.38    85.86    86.38
german_credit             (100)   74.92 |   74.26    74.06    74.42
-------------------------------------------------------------------------
                                        (v/ /*) |  (0/14/1)  (0/15/0)   (0/15/0)

Key:
(1) meta.Bagging '-P 100 -S 1 -num-slots 1 -I 100 -W trees.J48 -- -U -M 2' -1158799622371199696
(2) meta.C-Bagging '-K 100 -S 1 -num-slots 1 -I 1000 -W trees.J48 -- -U -M 2' -5058796223711997O4
(3) meta.C-BaggingHO '-K 100 -S 1 -num-slots 1 -I 1000 -W trees.J48 -- -U -M 2' -5058796223711997O5
(4) meta.BaggingRC '-K 100 -S 1 -num-slots 1 -I 1000 -W trees.J48 -- -U -M 2' -5058796223711997O6
```

Extra experiment output - ensemble with 100 base classifiers (RandomTree):

```
Analysing:  Percent_correct
Datasets:   15
Resultsets: 4
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       18-5-28

Dataset                 (1) meta.Bagg | (2) meta. (3) meta. (4) meta.
-----------------------------------------------------------------------
horse-colic       (100)   85.81 |    85.75     85.42     85.48
wisconsin-breast-cancer (100)   96.57 |    96.60     96.40     96.68
breast-cancer     (100)   69.32 |    69.74     69.33     69.60
pima_diabetes     (100)   76.28 |    76.41     76.09     76.36
heart-statlog     (100)   82.81 |    83.11     83.11     83.19
hepatitis         (100)   83.64 |    83.90     83.03     84.03
ionosphere        (100)   93.40 |    93.74     93.34     93.74
kr-vs-kp          (100)   99.17 |    99.18     99.06     99.19
labor             (100)   88.67 |    88.93     88.33     89.33
mushroom          (100)  100.00 |   100.00    100.00    100.00
sick              (100)   98.32 |    98.37     98.32     98.36
sonar             (100)   83.68 |    84.06     82.72     83.37
vote              (100)   96.50 |    96.54     96.41     96.59
credit-rating     (100)   86.64 |    86.41     86.46     86.58
german_credit     (100)   75.82 |    76.18     75.52     76.46
-----------------------------------------------------------------------
                                (v/ /*) |   (0/15/0)   (0/15/0)   (0/15/0)


Key:
(1) meta.Bagging '-P 100 -S 1 -num-slots 1 -I 100 -W trees.RandomTree -- -K 0 -M 1.0 -V 0.001 -S 1' -1158799622371996696
(2) meta.C-Bagging '-K 100 -S 1 -num-slots 1 -I 1000 -W trees.RandomTree -- -K 0 -M 1.0 -V 0.001 -S 1' -5058796223719704
(3) meta.C-BaggingHO '-K 100 -S 1 -num-slots 1 -I 1000 -W trees.RandomTree -- -K 0 -M 1.0 -V 0.001 -S 1' -5058796223719705
(4) meta.BaggingRC '-K 100 -S 1 -num-slots 1 -I 1000 -W trees.RandomTree -- -K 0 -M 1.0 -V 0.001 -S 1' -5058796223719706
```

Reference:

[1] WIKIPEDIA "*Data Mining*". [Online]. Available: https://en.wikipedia.org/wiki/Data_mining

[2] "*Data Mining vs. Machine Learning: What's The Difference*? Import.io, 2017. [Online]. Available: https://www.import.io/post/data-mining-machine-learning-difference/

[3] Han, Jiawei, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[4] Zhou, Zhi-Hua. *Ensemble methods: foundations and algorithms*. CRC press, 2012.

[5] Jason Brownlee. "*Difference Between Classification and Regression in Machine Learning*". In Start Machine Learning, 2017. [Online]. Available: https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/

[6] "Machine Learning in MATLAB", MathWorks. [Online]. Available: https://in.mathworks.com/help/stats/machine-learning-in matlab.html?w.mathworks.com

[7] Hui Li. "Which machine learning algorithm should I use?", 2017. [Online]. Available: https://blogs.sas.com/content/subconsciousmusings/2017/04/12/machine-learning-algorithm-use/#prettyPhoto

[8] Dietterich, Thomas G. "Ensemble methods in machine learning." In *International workshop on* multiple *classifier systems*, pp. 1-15. Springer, Berlin, Heidelberg, 2000.

[9] L. K.Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

[10] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2): 197–227, 1990.

[11] Necati Demir, Turkey. "*Ensemble Methods: Elegant Techniques to Produce Improved Machine Learning Results*," 2015. [Online]. Available: https://www.toptal.com/machine-learning/ensemble-methods-machine-learning

[12] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996d.

[13] L. Breiman. Population theory for boosting ensembles. *Annals of Statistics*, 32(1):1–11, 2004.

[14] "What is the difference between Bagging and Boosting". In QuantDare. [Online]. Available:https://quantdare.com/what-is-the-difference-between-bagging-and boosting/

[15] Baba, Nura Muhammad, et al. "Current Issues in Ensemble Methods and Its Applications." *Journal of Theoretical and Applied Information Technology* 81.2 (2015): 266.

[16] Neto, A. A. F., & Canuto, A. M. (2014).“*Meta-Learning and Multi-Objective Optimization to Design Ensemble of Classifiers”,* 2014 Brazilian Conference on Intelligent Systems. pp. 91 – 96, IEEE.

[17] Wang, W. (2008). “*Some Fundamental Issues in Ensemble Methods”,* In Neural Networks, IJCNN. IEEE World Congress on Computational Intelligence. IEEE International Joint Conference on pp. 2243-2250. IEEE.

[18] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[19] F.-J.Huang, Z.-H. Zhou,H.-J. Zhang, and T. Chen. Pose invariant face recognition. In *Proceedings of the 4th IEEE International Conference on Automatic Face and Gesture Recognition*, pages 245–250, Grenoble, France, 2000.

[20] S. Avidan. Ensemble tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(2):261–271, 2007.

[21] G. Giacinto, F. Roli, and L. Didaci. Fusion of multiple classifiers for intrusion detection in computer networks. *Pattern Recognition Letters*, 24(12): 1795–1803, 2003.

[22] G. Giacinto, R. Perdisci, M. D. Rio, and F. Roli. Intrusion detection in computer networks by a modular ensemble of one-class classifiers. *Information Fusion*, 9(1):69–82, 2008.

[23] J. Z. Kolter andM. A.Maloof. Learning to detect and classify malicious executables in the wild. *Journal ofMachine Learning Research*, 7:2721–2744, 2006.

[24] Z.-H. Zhou, Y. Jiang, Y.-B. Yang, and S.-F. Chen. Lung cancer cell identification based on artificial neural network ensembles. *Artificial Intelligence in Medicine*, 24(1):25–36, 2002a.

[25] R. Polikar, A. Topalis,D. Parikh, D.Green, J. Frymiare, J. Kounios, and C.M. Clark. An ensemble based data fusion approach for early diagnosis of Alzheimer’s disease. *Information Fusion*, 9(1):83–95, 2008.

[26] L. I. Kuncheva, (2005). *"Combining Pattern Classifiers: Methods and Algorithms"*, New York: Wiley.

[27] G. Fumera and F. Roli. A theoretical and experimental analysis of linear combiners for multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):942–956, 2005.

[28] Kuncheva, Ludmila I., and Christopher J. Whitaker. "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy." *Machine learning* 51, no. 2 (2003): 181-207.

[29] D. B. Skalak. The sources of increased accuracy for two proposed boosting algorithms. In *Working Notes of the AAAI'96Workshop on Integrating Multiple Learned Models*, Portland, OR, 1996.

[30] T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8): 832–844, 1998.

[31] P. H. A. Sneath and R. R. Sokal. *Numerical Taxonomy: The Principles and Practice of Numerical Classification*. W. H. Freeman, San Francisco, CA, 1973.

[32] G. Yule. On the association of attributes in statistics. *Philosophical Transactions of the Royal Society of London*, 194:257–319, 1900.

[33] J. Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

[34] D. D.Margineantu and T. G. Dietterich. Pruning adaptive boosting. In *Proceedings of the 14th International Conference on Machine Learning*, pages 211–218, Nashville, TN, 1997.

[35] G. Giacinto and F. Roli. Design of effective neural network ensembles for image classification purposes. *Image and Vision Computing*, 19(9-10): 699–707, 2001.

[36] D. Ruta and B. Gabrys. Application of the evolutionary algorithms for classifier selection in multiple classifier systems with majority voting. In *Proceedings of the 2nd International Workshop on Multiple Classifier Systems*, pages 399–408, Cambridge, UK, 2001.

[37] D. Partridge and W. J. Krzanowski. Software diversity: Practical statistics for its measurement and exploitation. *Information & Software Technology*, 39(10):707–717, 1997.

[38] R. Kohavi and D. H. Wolpert. Bias plus variance decomposition for zero one loss functions. In *Proceedings of the 13th International Conference on Machine Learning*, pages 275–283, Bari, Italy, 1996.

[39] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[40] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[41] L. Breiman. Randomizing outputs to increase prediction accuracy. *Machine Learning*, 40(3):113–120, 2000.

[42] P.Melville and R. J.Mooney. Creating diversity in ensembles using artificial data. *Information Fusion*, 6(1):99–111, 2005.

[43] J. F. Kolen and J. B. Pollack. Back propagation is sensitive to initial conditions. In R. Lippmann, J. E.Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pages 860–867.Morgan Kaufmann, San Francisco, CA, 1991.

[44] S. W. Kwok and C. Carter. Multiple decision trees. In *Proceedings of the 4th International Conference on Uncertainty in Artificial Intelligence*, pages 327–338, New York, NY, 1988.

[45] K. M. Ali and M. J. Pazzani. Error reduction through learning multiple descriptions. *Machine Learning*, 24(3):173–202, 1996.

[46] B. V. Dasarathy, editor. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, Los Alamitos, CA, 1991.

[47] T. Windeatt and C. Zor, "Ensemble Pruning Using Spectral Coefficients", IEEE Transactions on Neural Networks and Learning Systems, vol. 24, no. 4, pp. 673-678, 2013.

[48] D. D.Margineantu and T. G. Dietterich. Pruning adaptive boosting. In Proceedings of the 14th International Conference on Machine Learning, pages 211–218, Nashville, TN, 1997.

[49] J. L. Fleiss. Statistical Methods for Rates and Proportions. John Wiley & Son New York, NY, 2nd edition, 1981.

[50] D. Hernandez-Lobato, G. Martinez-Munoz, and A. Suarez. Statistical instance-based pruning in ensembles of independent classifiers. IEEE Transaction on Pattern Analysis and Machine Intelligence, 31(2):364–369, 2009.

[51] A. Lazarevic and Z. Obradovic. Effective pruning of neural network classifier ensembles. In Proceedings of the IEEE/INNS International Joint Conference on Neural Networks, pages 796–801, Washington, DC, 2001.

[52] Z.-H. Zhou, J. Wu, and W. Tang. Ensembling neural networks: Many could be better than all. *Artificial Intelligence*, 137(1-2):239–263, 2002b.

[53] G. Martínez-Muñoz and A. Suárez. Aggregation ordering in bagging. *In Proceedings of the IASTED International Conference on Artificial Intelligence and Applications*, pages 258–263, Innsbruck, Austria, 2004.

[54] G. Martínez-Muñoz and A. Suárez. Using boosting to prune bagging ensembles. *Pattern Recognition Letters*, *28*(1), 156-165, 2007.

[55] Kaufman, L., Rousseeuw, P., Finding groups in data: an introduction to cluster analysis, Willey, New York, 1990.

[56] Y. Zhang, S. Burer, and W. N. Street. Ensemble pruning via semi-definite programming. Journal of Machine Learning Research, 7:1315–1338, 2006.

[57] H. Chen, P. Ti ˜ no, and X. Yao. A probabilistic ensemble pruning algorithm. In Working Notes of ICDM'06 Workshop on Optimization-Based Data Mining Techniques with Applications, pages 878–882, Hong Kong, China, 2006.

[58] L. I. Kuncheva and C. J.Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2):181–207, 2003.

[59] J. L. Fleiss. *StatisticalMethods for Rates and Proportions*. JohnWiley & Sons, New York, NY, 2nd edition, 1981.

[60] C. A. Shipp and L. I. Kuncheva. Relationships between combination methods and measures of diversity in combining classifiers. *Information Fusion*, 3(2):135–148, 2002.

[61] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman&Hall, New York, NY, 1993.

[62] S. P. Lloyd. Least squares quantization in PCM. IEEE Transactions on Information Theory, 28(2):128¨C137, 1982.

[63] A. Strehl and J. Ghosh. Cluster ensembles – A knowledge reuse framework for combining multiple partitions. Journal of Machine Learning Research, 3:583–618, 2002.

[64] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, San Francisco, CA, 2nd edition, 2006.

[65] Blomberg LC, Ruiz DDA (2013) Evaluating the influence of missing data on classification algorithms in data mining applications. Paper presented at SBSI 2013: Simpósio Brasileiro de Sistemas de Informação, João Pessoa, 22–24 May 2013.