Machine Learning Ensemble Method for Discovering Knowledge from Big Data

Majed Farrash

School of Computing Sciences

University of East Anglia

This thesis is submitted for the degree of

Doctor of Philosophy

January 2016

I would like to dedicate this thesis to my loving parents, wife and children …

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor, Dr. Wenjia Wang, who provided valuable encouragement and advice throughout my PhD journey. His attentiveness to my problems and his patience in discussing issues and answering questions were particularly helpful as I refined my research. The completion of this thesis would not have been possible without his constant availability and expert professional support.

My appreciation goes as well to my secondary supervisor, Dr. Beatriz De La Iglesia, for her insightful comments, positive motivation and meaningful guidance. A further special note of thanks must be expressed to all my colleagues in the School of Computing Science at University of East Anglia and all the friends whom I met in Norwich for their inspiration and company.

I am likewise indebted to my parents, brothers and sister, for without them, I would not have accomplished any of my dreams.

Finally, a heartfelt thank you goes to my beloved wife, Abrar, for the love and unremitting support that she provided all my life.

ABSTRACT

Big data, generated from various business internet and social media activities, has become a big challenge to researchers in the field of machine learning and data mining to develop new methods and techniques for analysing big data effectively and efficiently. Ensemble methods represent an attractive approach in dealing with the problem of mining large datasets because of their accuracy and ability of utilizing the divide-and-conquer mechanism in parallel computing environments.

This research proposes a machine learning ensemble framework and implements it in a high performance computing environment. This research begins by identifying and categorising the effects of partitioned data subset size on ensemble accuracy when dealing with very large training datasets. Then an algorithm is developed to ascertain the patterns of the relationship between ensemble accuracy and the size of partitioned data subsets. The research concludes with the development of a selective modelling algorithm, which is an efficient alternative to static model selection methods for big datasets.

The results show that maximising the size of partitioned data subsets does not necessarily improve the performance of an ensemble of classifiers that deal with large datasets. Identifying the patterns exhibited by the relationship between ensemble accuracy and partitioned data subset size facilitates the determination of the best subset size for partitioning huge training datasets. Finally, traditional model selection is inefficient in cases wherein large datasets are involved.

List of Notations

Symbol	Description
ϕ	an ensemble of classifiers
ϕ_{init}	an initial ensemble
$\lvert\phi_{init}\rvert$	the size of the initial ensemble
ϕ_{Rt}	an ensemble of classifiers where data subsets size equal Rt
$acc(\phi)$	the accuracy of an ensemble of classifiers
ϕ_{FE}	full ensemble
ϕ_{FS}	ensemble of classifiers that is constructed using the forward search selection strategy
ϕ_{SM}	ensemble of classifiers that is constructed using the proposed selective modelling
$\Delta acc(\phi_{Rt\,i}, \phi_{Rt\,i-1})$	the difference between the accuracy of the two ensemble $\phi_{Rt\,i}$ and $\phi_{Rt\,i-1}$
N	number of data subsets
M	number of models within an ensemble of classifiers
acc_{min}	minimum accuracy of individual models
$\overline{acc(M)}$	the mean of the accuracy of individual models within an ensemble
Rt	the relative size of a partitioned data subset
Rt_{max}	the maximum relative size of a partitioned data subset
Rt_{min}	the minimum relative size of a partitioned data subset
Rt_{low}	the lower boundary for the Rt value.
Rt_{stop}	the last Rt in the predicated curve.
$acc(\phi_{Rt_i})$	ensemble accuracy when the size of data subsets is equal to Rt_i.

Tr	training dataset.
V	validation dataset.
TS	testing dataset.
C	number of class labels for a dataset.
r	the Pearson product-moment correlation.
r_s	the Spearman's correlation coefficient.
$IR_{(Rt_i,Rt_j)}$	the improvement rate in ensemble accuracy between Rt_i and Rt_j.
α	initial step size.
β	the number of successive Rt points that needs to be identified before terminating the algorithm.
θ	tolerance.
γ	the percentage of the minimum number of models to be checked before the termination criterion is applied.
π	number of models to be added at each iteration.
Mem_{max}	the maximum available memory.
g	growth ratio
$g(\phi_{Rt\,i}, \phi_{Rt\,i-1})$	the accuracy growth ration between the two ensembles $\phi_{Rt\,i}$ and $\phi_{Rt\,i-1}$.
I	total number of iterations.
T	required time to identify the relationship patterns.
T_p	partitioning Time.
T_{Tr}	training time.
T_{ee}	ensemble evaluation time.
T_{oth}	other times (e.g. times required to access files and calculate DF diversity).
$T(\phi)$	ensemble time
T_{SM}	selective modelling time.

List of abbreviations

Symbol	Description
DF	the **D**ouble-**F**ault Diversity.
CFD	**C**oincident **F**ailure **D**iversity.
SVM	**S**upport **V**ector **M**achine.
CVM	**C**ore **V**ector **M**achine.
MP	**M**odel **P**ool.
MSL	**M**emory **S**afe **L**imit.
DP	**D**ata Subset **P**ool.
SM	**S**elective **M**odelling.
MAM	The **M**ost **A**ccurate **M**odel.
FSoutRWC	**F**orward **S**earch with**out** **R**emoval of **W**eak **C**lassifiers.
FSwithRWC	**F**orward **S**earch **with** **R**emoval of **W**eak **C**lassifiers.
SMwithRNK	**S**elective **M**odelling **with** **R**an**k**ing.
SMoutRNK	**S**elective **M**odelling with**out** **R**an**k**ing.

Glossary of Terms

Term	Definition
Training dataset	A set of examples used for learning, that is to fit the parameters of the classifiers [74].
Validation dataset	A set of examples used to tune the parameters of a classifier [74].
Testing dataset	A set of examples used only to assess the performance of a fully-specified classifier [74].
Ensemble of classifiers	A set of classifiers whose individual classification are combined to produce a single classification for a given problem [41][99].
Big datasets	Big datasets, in this research, are those that are impossible to handle and process with the available computing environment because of the large volume of the datasets and the fact that available data mining algorithms were developed under the assumption that entire data are loaded into a computer's main memory.
Model pool	A pool that contains all the models generated during the modelling phase.
DP	A pool that contains all the partitioned data subsets.
P1 relation pattern	Refer to the relation pattern where $acc(\phi)$ decreases with growing Rt, regardless of whether the decrease is continuous or stops at a certain *Rt* and stabilises (see Figure 4.17).
P2 relation pattern	Refer to the relation pattern where $acc(\phi))$ increases with growing Rt, regardless of whether the increase is continuous or stops at a certain *Rt* and stabilises(see Figure 4.17).

Term	Definition
P3 relation pattern	Refer to the relation where $acc(\phi)$ shows no Rt-induced effects or an increase in Rt negligibly contributes to $acc(\phi)$ (see Figure 4.17).
P4 relation pattern	Refer to the relation where $acc(\phi)$ improves up to a particular Rt and then decreases (see Figure 4.17).
Tolerance	The greatest range of variation allowed when the accuracy of two ensembles are compared.
Partitioning Time	Time required to partition the training dataset into N subsets using Rt.
Training time	Training (modelling) time is the time required to train the data subsets to create a base classifiers by using the machine learning algorithm.
Ensemble evaluation time	The time required to evaluate an ensemble over the TS and V datasets.
Ensemble time	The summation of partitioning time, training time and ensemble evaluation time.
Full ensemble	An ensemble that constructed from all the available models.

List of Publications

- Farrash, M. and Wang,W. (2013). How data partitioning strategies and subset size influence the performance of an ensemble? In *Big Data, 2013 IEEE International Conference on*, pages 42–49.

- Farrash, M. and Wang, W. (2015). An algorithm for identifying the learning patterns in big data. In *Trustcom/BigDataSE/ISPA, 2015 IEEE*, volume 2, pages 48–55.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Background

The rapid improvements in data storage technologies and communication infrastructure, as well as the availability of reliable, efficient and affordable computing hardware, have resulted in the exponential growth of data over the past several years. IBM states that 2.5 quintillion bytes of data are created from our daily activities [38], and the International Data Corporation (IDC) estimates the amount of generated data to increase by 40% a year in the next decade [96]. Massive data volumes are also produced from simple everyday activities. Customer transactions at Walmart, for example, total more than a million per hour [88]. Facebook handles more than 250 million photo uploads and the interactions of 800 million active users with more than 900 million objects (pages, groups, etc.) each day [88]. Furthermore, billions of people around the world constantly use their smartphones and digital devices. Also, Internet of things will be a reality in not far future and then the rate of digital data generation will be even greater.

The availability of tremendous amount of data has made the term 'big data' a commonplace in different fields of science research. A problem, however, is that no formal definition of the term has been universally accepted by scholars. One of the labels most frequently associated with big data is '3Vs', which was conceived by

Laney in 2001 [50] to describe the three main attributes of big data: volume, velocity and variety. Later studies show that the 3Vs definition is insufficient to encompass all the attributes of big data, hence the introduction of the designations '4Vs', '6Vs' and '3^2Vs'. These extended definitions are described in more detail in [89] and [104].

The availability of this enormous amount of data makes the process of analysing such data a challenging task. As a consequence, researchers in the field of machine learning and data mining were motivated to develop new techniques and methods to analyse big data effectively and efficiently.

1.2 Motivation

Apart from the availability of big data, ensemble methods represent an attractive approach that can be used to deal with the problem of mining large datasets. These methods are appealing alternatives for two main reasons. Firstly, many studies have confirmed that an ensemble of classifiers, such as bagging ([7]), boosting ([30]) and random forest ([9]), outperform single base classifiers ([11]). Secondly, ensemble methods can be implemented in parallel computing environments, such as [17, 62], which are necessary to process big datasets.

This research focuses on the problems that arise when existing machine learning algorithms are used to handle a single large training dataset. Dividing such a dataset into smaller manageable data subsets often represents a rational solution that can be implemented by using ensemble methods. The succeeding section discusses the principal challenges that we are going to focus on in this study.

1.3 Problem Statement

Although using the divide-and-conquer technique represents a rational solution to deal with a big dataset, adopting it as a means of building an efficient and effective ensemble of classifiers raises several challenges. The first revolves around how data subset size is

chosen when a huge training dataset requires partitioning. Is it necessary to maximise the size of a partitioned data subset to deliver acceptable ensemble performance?

The second challenge emerges when model selection techniques are applied during the ensemble construction process. Model selection is a well-known practice that is aimed at choosing a smaller subset of models from a set that contains all available models to enhance the prediction performance of a created ensemble. In the case of large datasets, model creation on the basis of each available data subset and subsequent model selection constitute resource wastage. This exhaustion of resources arises from the fact that only some of the generated models are selected to formulate a final ensemble prediction.

This study contributes to the resolution of the above-mentioned problems by proposing a method for identifying the patterns of the relationship between the size of partitioned data subsets and ensemble accuracy. Understanding these patterns is expected to facilitate the selection of suitable sizes of partitioned data subsets. Another approach, the selective modelling method, is put forward to minimise the time and resources allocated to constructing an ensemble of classifiers from a huge dataset. Also this method can be used as an alternative to static model selection when dealing with very large datasets.

1.4 Research Questions

To address the research problems described in Section 1.3, the following research questions are investigated:

Q 1: Does the size of partitioned data subsets affect ensemble accuracy?

Illuminating this issue necessitates an intensive experiment on whether the size of partitioned data subsets and ensemble accuracy are related. If such an association exists, an important requirement is to determine whether this

influence can be characterised by certain patterns, which leads to the second research question.

Q 2: Is it possible to categorise the patterns that underlie the relationship between the size of partitioned data subsets and ensemble accuracy?

The importance of categorising relationship patterns lies in its advancement of a an improved understanding of datasets in terms of learning behaviour and its informing of decisions regarding when it is appropriate or inappropriate to use as much data as possible in building an ensemble.

Q 3: How can the aforementioned patterns be identified for big datasets?

After addressing the first two issues, a logical follow-up task is to develop an algorithm that is able to identify the learning patterns that underlie the relationship between the accuracy of an ensemble and the size of partitioned data subsets used to induce an ensemble of classifiers. This algorithm should also be helpful in deciding the best data subset size of a given dataset for use when an ensemble is constructed.

Q 4: How are model section techniques used to build an efficient and effective ensemble of classifiers when dealing with a big dataset?

All the previously identified questions focus on the problem presented by big data that requires partitioning into smaller manageable subsets without taking the number of subsets into consideration. The final question centres on the problem that arises from a large number of data subsets as a result of partitioning a single large dataset. Answering this question entails assessing the efficacy of existing model selection techniques in dealing with numerous models. Subsequently, there is a need to design and develop a novel model selection technique that can efficiently and effectively handle a large number of models.

1.5 Research Aim and Objectives

This research aims to provide efficient and effective methods that help to deal with the problem of mining big datasets using ensemble of classifiers. To these ends, the following objectives are pursued:

1. To empirically investigate and establish the relationship between ensemble performance and the size of partitioned data subsets.
2. To categorise the patterns manifested by the aforementioned relationship.
3. To design and develop a method that identifies the patterns of the relationship between ensemble performance and partitioned data subset size.
4. To design and develop efficient models selection method that can handle large datasets efficiently without negatively affecting the ensemble performance.
5. To evaluate the proposed methods on benchmark datasets.

1.6 Research Methodology

As prompted by the nature of the research problem (see Section 1.2) and the defined research questions and objectives, an appropriate methodology and adaptable platform are essential to carry out the experiments for the identified research tasks. The divide-and-conquer approach is adopted because it is a typical solution in cases wherein a huge dataset cannot be loaded and processed in the main memory of a computing facility. This approach can be applied in machine learning and data mining via ensemble methods given that one of the approaches to create an ensemble of classifiers involves manipulating training datasets. Section 3.2 describes in detail the framework for the ensemble of classifiers adopted in this research. Our framework is run over the high-performance computing cluster supported by the Research and Specialist Computing Support Service at the University of East Anglia [UEA].

As indicated in Section 1.1, there is no formal definition of big data has been provided, hence it is important to delineate what the term means in the context of this research. Specifically, big datasets, in this research, are those that are impossible to handle and process with the available computing environment because of the large volume of the datasets and the fact that available data mining algorithms were developed under the assumption that entire data are loaded into a computer's main memory. This adopted definition of big data is similar to the definition proposed in [27] and [89].

In addition, this chapter describes how the experimental results were statistically evaluated and illustrates the main characteristics and pre-processing procedures that had been implemented for the datasets . During the research period, 13 moderate-to-big datasets were chosen from various dataset repositories to evaluate the proposed methodologies. Variant dataset sizes were chosen due to the limited availability of sizeable benchmark datasets within the machine-learning dataset repositories. It was vital that manageable datasets be made available so that they could be loaded and processed within the main memory to evaluate the efficiency and effectiveness of the proposed methods. The main characteristics and pre-processing procedures that were implemented for the used datasets have been comprehensively described in Chapter 3.

1.7 Research Contributions

This reserach proposes the specific contributions :

1. Identifies and categorises the patterns manifested by the relationship between ensemble performance and the size of partitioned data subsets (Chapter 4). A pilot study for identifying and categorising these patterns was published in the 2013 IEEE International Conference on Big Data [24].

2. Proposes an algorithm that identifies the relationship between ensemble performance and partitioned data subset size. The algorithm can also be used to choose

the best size for partitioned data subsets in cases wherein large datasets are mined (Chapter 5). This algorithm was published in the The 9th IEEE International Conference on Big Data Science and Engineering (IEEE BigDataSE-15) [25].

3. Proposes a selective modelling method, to handle the problem of having large number of data subsets efficiently. The generated ensemble using the proposed method overcomes the performance of ensemble that is constructed from all the available models and it required less time and resources (Chapter 6).

1.8 Outline of the Thesis

The structure of this thesis is outlined in this section.

- **Chapter 1** introduces the study, with particular focus on the motivation, aims and contributions of the research.
- **Chapter 2** presents an overview of related literature. It describes current ensemble methods and introduces some essential ensemble algorithms. The chapter also discusses the studies that have extensively contributed to the data mining field with a focus on mining large datasets.
- **Chapter 3** describes the research methodology and design. Additionally, it describes the datasets used in experiments, as well as the tools and software employed in carrying out this work.
- **Chapter 4** presents the empirical investigation of the effects of partitioned data subset size on ensemble accuracy when huge datasets are handled.
- **Chapter 5** details the proposed algorithm for identifying the learning patterns that underlie the relationship between the accuracy of an ensemble of classifiers and the relative size of data subsets.

- **Chapter 6** this chapter propose the selective modelling method, which is aimed at minimising the time and resources required to construct an ensemble of classifiers from a large dataset.

- **Chapter 7** summarises the results, discusses the research limitations and highlights suggestions for future work.

CHAPTER 2

REVIEW OF THE RELEVANT LITERATURE

2.1 Ensemble of classifiers

An ensemble of classifiers, a committee of classifiers and a multiple classifier system all refer to a set of classifiers whose individual classification are combined to produce a single classification for a given problem [41][99]. Fundamentally, the aim of using an ensemble of classifiers is to generate more reliable and accurate classification than that produced with single classifier-based methods. Dietterich [21] verified why an ensemble can outperform single classifier-based approaches if it is properly constructed. Other empirical studies, such as [7, 11, 30], have also confirmed the superiority of ensemble performance over that of single classifiers.

The success of ensemble methods has inspired their widespread application in different fields; Yu et al. [108] indicates that ensemble approaches are applied in intelligent transportation systems [67, 111], bioinformatics [64, 84, 107], image and video processing [2, 45, 85] and remote sensing [101, 116]. Ensemble methods are also used in predictive toxicology [59], in which the toxicity of chemical compounds is identified.

2.2 How is an ensemble of classifiers created?

An ensemble of classifiers can be constructed in many ways. Common techniques [87, p.279][41, 44] are summarised in the succeeding sections below.

2.2.1 Dividing a training set

Dividing a training set involves generating multiple data subsets from a single training dataset, after which multiple classifiers are constructed from the multiple data subsets. Data subsets are created using sampling and partitioning techniques. The division approach is aimed at generating a set of classifiers characterised by diversity that originates from the training of each classifier on different data subsets. One of the most popular ensembles of classifiers that adopt this technique is Bagging [53].

Bagging, which is also known as bootstrap aggregating, was proposed by Breiman [7]. It is grounded in the idea of creating multiple subsets (bootstraps) by repeatedly extracting samples with replacement from the original dataset. In standard bagging, the size of each bootstrap is equal to that of the original training dataset. Because sampling is conducted with replacement, any training instance may appear in a bootstrap more than once, whereas some training instances may not appear at all. On average, 37% of training set instances do not appear in a bootstrap, especially with large datasets [83]. After generating bootstraps, a base classifier model is built on the basis of each bootstrap by using a decision tree learning algorithm. The final ensemble decision is obtained by majority voting. The pseudo code for bagging is shown in the Algorithm 1.

After the introduction of Bagging in 1996, multiple ensemble algorithms have been proposed on the basis of training dataset manipulation. Some of the most important are summarised below.

- Bauer and Kohavi [4] in 1999 proposed wagging, which can be considered a modified version of bagging. In this approach, sampling is replaced by instance

Algorithm 1: Bagging algorithm. [87, p. 283]

1 **for** $i = 1$ **to** k **do**
2 | Create a bootstrap sample of szie N, D_i.
3 | Train a base classifier C_i on the bootstrap sample D_i.
4 **end**
5 $C^*(x) = \underset{y}{argmax} \sum_i \delta(C_i(x) = y)$
6 $\delta(\cdot) = 1$ if its argument is ture and 0 otherwise.

weighting. All instances are first set to a uniform weight. At each iteration, Gaussian noise is incorporated into the weight of each instance, after which a classifier is induced. The resultant ensemble is more diverse than that derived from the original bagging method.

- Hothorn and Lausen in 2003 put forward double bagging [36], which is described in [113] as follows. A bootstrap with a size equal to that of the original training dataset is randomly drawn from the original training set. The extraction is carried out using sampling with replacement. The set of instances that represents the out-of-bag sample is then used to perform linear discriminant analysis (LDA). The discriminant variables are calculated for each bootstrap and incorporated as additional variables for building a base classifier. The process is repeated multiple times, and the final design of an ensemble is produced by simple majority voting. The authors demonstrated that this method can outperform standard bagging.

- A bagging multitree was proposed in 2004 by [22] to reduce the training time required in constructing an ensemble. This method begins with the construction of a single structure (a multitree) that includes an ensemble of decision trees obtained by bagging but without repeating the parts that are common to the trees. The experimental results of [22] show that this method can exhibit performance that is comparable to that of original bagging but completes ensemble creation at faster speeds .

- CeBag was developed by Wang and Lin in 2007 [100] as an extended version of bagging that uses SVM instead of a decision tree as a base learner. The experimental results confirm that CeBag can outperform single classifier-based SVM.

- Gan and Xiao produced in 2009 a neural network ensemble [115] that presents performance superior to that of bagging. The main principle that underlies this method is the use of a K-means algorithm to generate different data subsets from the original dataset. Classifiers that employ a neural network algorithm are then constructed from each data subset, and the final prediction of the ensemble is produced by majority voting.

2.2.2 Manipulating data distribution

Boosting typically involves the manipulation of data distribution. The first provable polynomial-time boosting algorithm was proposed by Schapire [80], and the most well-known boosting algorithm is AdaBoost, which was put forward by Freund and Schapire [30].

AdaBoost is described in [29] as a process that depends on continual changes to the distribution of a training set in an iterative manner. Instead of dividing a training dataset, multiple classifiers are iteratively constructed from the entire dataset. At each iteration, the new base classifier focuses on training instances that are incorrectly classified in the previous iteration. The final ensemble prediction is made by weighted voting, wherein each classifier's prediction is weighted according to its accuracy on the training datasets. AdaBoost [29] is the most popular algorithm that uses data distribution manipulation in creating an ensemble of classifiers.

- Pasting small votes was put forward by [8] in 1999 as a variation of bagging and boosting that can deal with large datasets and produce accuracy that is comparable to that of AdaBoost.

- SMOTEBoost was proposed by Chawla et al. [18] as a boosting alternative that is designed to deal with highly imbalanced datasets. The experimental results show that SMOTEBoost can outperform AdaBoost in predicting minority classes.

- A new boosting method called AdaBoost.M1.ICV was proposed by Blagus and Lusa in 2015 [57] as a modification to AdaBoost.M1. The authors empirically demonstrated that AdaBoost.M1 can perform poorly in cases that involve high-dimensional training datasets. The modified version exhibits performance that is better or at least similar to that of AdaBoost.M1.

2.2.3 Manipulating input features

In manipulating input features, multiple classifiers are constructed from the same training dataset as each classier is built using different parts of a feature space. Random forest, which uses feature subspace techniques and is intended particularly for a decision tree, was put forward by Breiman [9]. The word 'forest' here refers to a constellation of many tree models that are constructed with no trimming or pruning of fully grown trees [61, 114]. As described in [75], the random forest method is based on the creation of multiple decision trees (classifiers). Each tree is constructed from n instances, which are drawn from the original training dataset by sampling with replacement. In each tree node, a splitting attribute is selected from a randomly chosen sample of the training dataset's attributes. Ensemble prediction is produced using majority voting.

Through a practical experiment, Breiman [9] demonstrated that the accuracy of random forest is as good as or sometimes better than that of AdaBoost; it also performs at a faster rate than do bagging and boosting. His study also reveals that random forest is robust to outliers and noise because it employs random inputs and random features. Figure 2.1 presents the structure of the random forest method.

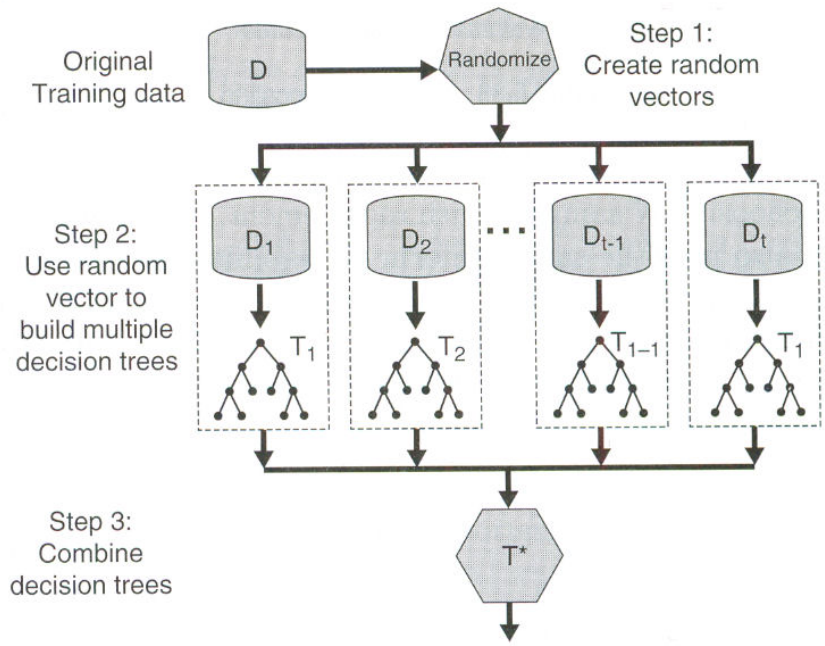


Fig. 2.1 Random Forest [87, p.279]

2.2.4 Manipulating learning algorithms

In all the previously discussed approaches, a single type of learning algorithm is often used to construct an ensemble, which is usually called a homogeneous ensemble. By contrast, manipulating a learning algorithm produces an ensemble through either the manipulation of a single learning algorithm to create different models from the same training dataset or the use of multiple learning algorithms that are each adopted to create a model from the same training dataset. An artificial neural network, for example, can be manipulated to generate different models that are trained on the same training datasets by changing its network topology or the initial weights of the line between neurons [87, p.279]. The ensembles produced by multiple learning algorithms are often called heterogeneous ensembles of classifiers.

Stacking, or stacked generalisation, is an example of an ensemble that uses different types of classifiers [44]. Stacked generalisation was proposed by Wolpert [103] . The main idea behind stacking is to use two levels of classifiers. The classifiers in the first

level are trained on the original training dataset. The output of the learners in the first level is then used to create a new dataset, which is employed to train second-level learning algorithms. The end-product of this process is the final prediction of an ensemble.

2.3 Factors that affect the performance of an ensemble of classifiers

The accuracy of an ensemble is influenced by the following factors:

1. Diversity amongst base classifiers.
2. Accuracy of individual classifiers.
3. Number of models within an ensemble.
4. The decision-making strategy used.
5. Use of model selection strategies.

This section presents a brief discussion of the above-mentioned factors.

2.3.1 Diversity amongst base classifiers

Using an ensemble method to deal with a classification problem is aimed at classifying a dataset as accurately as possible. An ensemble is a collocation of M models; thus, if these models are identical, then the performance of the ensemble will be equal to the individual performance of its base classifiers. Combining multiple models therefore produces satisfactory results only when models are significantly different from one another.

Figure 2.2 illustrates two ensembles, Ensemble A and B (examples proposed by Wang in [98]). Ensemble A is constructed from three base classifiers, each correctly

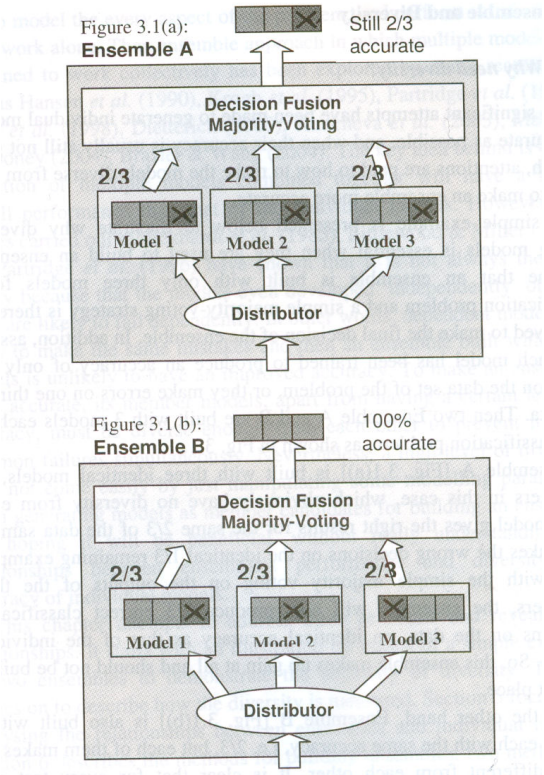


Fig. 2.2 Two Ensemble classifiers [98]

classifying two-thirds of a dataset and misclassifying the rest. All three models commit identical mistakes, indicating that they misclassify the same instances of the dataset. When majority voting is used as a decision-making strategy for the ensemble, two-thirds of the dataset is correctly classified and one-third is misclassified. As a result, using either an ensemble of classifiers or a single classifier exerts an equal effect on classification performance.

Ensemble B is also built from three base classifiers. As in the previous example, each base classifier correctly classifies two-thirds of a dataset and misclassifies the rest. The difference between the base classifiers in the two ensembles is that in Ensemble B, each base classifier makes a mistake that differs from those committed by the other base classifiers within the ensemble. Each base classifier thus misclassifies a different one-third of the examples from the dataset. When a majority voting strategy is used, each instance of the dataset is correctly classified by two base classifiers; that is, the entire dataset is correctly classified by the ensemble. These examples indicate that

diversity plays a key role in the overall performance of an ensemble. Knowing how to measure the diversity between the different models within an ensemble is therefore important.

2.3.1.1 Diversity measures

Numerous methods of measuring diversity are presented in the literature. Kuncheva et al. [48] summarises 10 such approaches, which are briefly discussed in this section. This measures can be classified into pairwise and non-pairwise diversity measures:

1. Pairwise diversity measures. A we have a training set $Z = \{z1, ..., zn\}$. The output of a classifier can be represented in an N-dimensional binary vector. We can measure the diversity between classifiers for the binary classification problem as follows:

   (a) The Q statistics: This measure determines, whether two classifiers are statistically independent or not using the following formula :

   $$Q_{i,k} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}} \tag{2.1}$$

   where $N^{11}N^{00} - N^{01}N^{10}$ can be calculated, as Table 2.1 shows.

   The value of Q will vary between [-1,1]. The Q value is positive, if the two classifiers predict the same example correctly. The Q value is negative if they make a wrong prediction in different examples.

   Table 2.1 The relationship between a pair of classifiers

		classifier k	
		Correct (1)	Incorrect(0)
classifier i	Correct (1)	N^{11}	N^{10}
	Incorrect(0)	N^{01}	N^{00}

(b) The correlation coefficient p. The correlation between two binary classifiers can be calculated as follows:

$$P_{i,k} = \frac{N^{11}N^{00} - N^{01}N^{10}}{\sqrt{(N^{11}+N^{10})(N^{01}+N^{00})(N^{11}+N^{01})(N^{10}+N^{00})}} \tag{2.2}$$

P has a similar sign to Q.

(c) The disagreement measure. “It is the ratio between the number of observations on which one classifier is correct and the other is incorrect to the total number off observations" [48]. The disagreement measure can be computed as follows:

$$Dis_{i,k} = \frac{N^{01}+N^{10}}{N^{11}+N^{10}+N^{01}+N^{00}} \tag{2.3}$$

(d) The double-fault measure. This measure is used to “form a pairwise diversity matrix for a classifier pool and subsequently to select classifiers that are least related" [48]. It is calculated by taking the ratio of the misclassified examples to the total number of classified examples. Equation 2.4 shows that

$$DF_{i,k} = \frac{N^{00}}{N^{11}+N^{10}+N^{01}+N^{00}} \tag{2.4}$$

2. Non-pairwise diversity measures.

(a) The entropy measure E can be calculated as in equation 2.5:

$$E = \frac{1}{N}\sum_{j=1}^{N}\frac{1}{L-\lceil L/2\rceil} min\left\{l\left(z_j\right), L-l\left(z_j\right)\right\} \tag{2.5}$$

where

- N = the total number of classified examples.
- L = number of class labels.
- $l\left(z_j\right)$ = the number of classifiers from D that correctly recognize z_j

- D = a set of classifiers.

(b) Kohavi-Wolpert variance. This measure can be calculated as shown in equation 2.6:

$$KW = \frac{1}{NL^2} \sum_{j=1}^{N} \left(L - l\left(z_j\right)\right) \tag{2.6}$$

(c) Measurement of interrater agreement K. "A static developed as a measure of interrater reliability can be used when different rates (here classifiers) assess subjects (here zj) to measure the level of agreement while correcting for chance" [48]. Measurement of interrater agreement K can be calculated as follows:

$$k = 1 - \frac{\frac{1}{L}\sum_{j=1}^{N} l\left(z_j\right)\left(L - l\left(z_j\right)\right)}{N\left(L-1\right)\bar{\rho}\left(1-\bar{\rho}\right)} \tag{2.7}$$

(d) Generalized diversity (GD). This measure was introduced by Partridge and Krzanowski in 1997 [70]. This measure computes the probability of two classifiers, making similar decisions (correct or incorrect). This measure can be computed as equation 2.8 shows:

$$GD = 1 - \frac{\sum_{i=1}^{L} \frac{i}{L} \frac{(i-1)}{(L-1)} p_i}{\sum_{i=1}^{L} \frac{i}{L} p_i} \tag{2.8}$$

where:

- L =the number of classifiers.
- p_i = the probability that $Y = \frac{i}{L}$
- Y = a random variable expressing the proportion of classifiers (out of L) that are incorrect on a random drawn object x.

The GD value is varies between 0, which mean the two classifiers are independent, and 1, which means the two classifiers are identical.

(e) Coincident Failure Diversity. This measure is a modified version of GD, and it was proposed also by Partridge and Krzanowski in 1997 [70]. Equa-

tion 6.1 shows how to calculate the coincident failure diversity.

$$CFD = \begin{cases} 0 & \text{if } p_o = 1.0 \\ \frac{1}{1-p_0} \sum_{i=1}^{L} \frac{L-i}{L-1} P_i & \text{if } p_o < 1 \end{cases} \tag{2.9}$$

These aforementioned diversity measures have been classified by Bian and Wang in study conducted in 2006 [6] into three groups in term of similarity as follow:

1. *Dis*, *KW* and *E*

2. *GD* and *CFD*

3. *DF*, *Q*,*P* and *k*

Although extensive experimental studies on these diversity measures were conducted by [46, 48, 77, 76, 82], the researchers did not find a strong correlation between them and Acc(ϕ) on all the examined datasets. Other studies, such as [43], put forward new diversity measures, but in most cases, no single measure exhibited a strong correlation with ensemble accuracy.

2.3.2 Accuracy of individual classifiers

A typical recommendation is that each individual classifier within an ensemble should exhibit performance greater than that of a random guess [99, 41]. The accuracy of a random guess equals $\frac{1}{C}$, where C is the number of class labels in a classification problem (e.g. The random guess for a binary classification problem is equal to $\frac{1}{2}$).

2.3.3 Number of models within an ensemble

The size of an ensemble is also critical to accuracy and can be determined on the basis of three factors suggested by [58]:

- Desired accuracy: Some researchers argue that a specific number of classifiers can improve the overall accuracy of an ensemble. [58, P974] summarises the main findings of these studies as follows:

  - Having 10 classifiers within an ensemble is sufficient to improve the accuracy of the ensemble [35].
  - The meta-learning approach was used to build an ensemble of an arbiter tree with subsets varying from two to 64 and discuss the effects of the number of subsets on overall accuracy [14].
  - AdaBoost was run for 10 to 100 iterations. Performance continually improved, even after 10^5 and 10^6 iterations [33].
  - An empirical experiment on AdaBoost shows that ensemble accuracy improved, even with a large number of classifiers (e.g. 25 classifiers) [68].

  Most ensemble approaches assume that enhancing ensemble performance necessitates that each base classifier within an ensemble exhibits performance that is superior to that of a random guess. However, some studies (e.g. [99]) confirm that the performance of an ensemble can be improved even if certain base classifiers are less accurate than a random guess. This outcome is possible only when classifiers are significantly diverse from one another and when the number of models within an ensemble is increased.

- User preferences: Sometimes, user preferences affect the number of classifiers that constitute a cohesive set of classifiers because 'increasing the number of classifiers increases the complexity and decreases the comprehensibility' [58, p. 975]. This phenomenon may be the factor that drives users who establish a specific number of classifiers to avoid overfitting.

- Number of processors available: "In concurrent approaches, the number of processors available for parallel learning could be put as an upper bound on the number of classifiers that are treated in paralleled process" [58, p. 975].

2.3.4 The decision-making strategy used

A fusion strategy used for an ensemble of classifiers is the process that entails integrating or combining the predictions of the individual classifiers to produce the final ensemble output. Fusion usually refers to types of approaches that combine the decisions of all the members of an ensemble. Conversely, ensemble selection approaches combine only a set of available classifiers to produce the final ensemble output. The most frequently used fusion strategies are described in this section, and ensemble selection methods are discussed in 2.3.5.

Majority voting

Majority voting or simple voting is one of the simplest and most commonly used ensemble strategies. Its idea is summarized as, when there is a need to classify an input instance, each member of the ensemble (classifier) votes for one class label, and the final class label is the one that receives the majority (half of the voter + 1)of the votes [49].

Plurality voting

Plurality voting is similar to majority voting but does not require a final label to receive more than half of the votes from ensemble members. The class label that receives the highest number of votes is the final decision of the ensemble [117]. In cases wherein binary classification is conducted, plurality voting is equal to majority voting.

Threshold plurality voting is a generational plurality voting method proposed by Xu et al. [106]. Here, a threshold determines the number of votes needed for a class label to be considered the final decision of an ensemble.

Weighted voting

Depending on the fact that the members of an ensemble often exhibit unequal performance, a rational approach is to assign more power to stronger members. In weighted voting, each classifier is assigned a weight, which often depends on their performance; their weights are taken into consideration when calculating the final decision of an ensemble [47].

2.3.5 Use of model selection strategies

Model selection, ensemble selection and ensemble pruning constitute the process of selecting a small set of classifiers from a large pool of all available classifiers to produce an efficient and effective ensemble [69]. Many researchers, such as [69, 79, 110, 31, 5], presented different techniques for selecting an optimal set of classifiers that improve overall ensemble performance. This process can be performed dynamically or statically. Dynamic model selection is carried out by selecting a specific set of classifiers for each testing instance. By contrast, static model selection uses the same selected set of classifiers to predict entire testing instances. Fundamentally, static and dynamic selection depend on the existence of a model pool that contains an entire set of available models.

2.3.5.1 Static Ensemble Selection

Tsoumakas et al. [94] provided a taxonomy for ensemble selection methods. They classified ensemble selection methods into four categories:

1. Search Based methods: This category includes all the methods that employ a heuristic search in the space of available models. The most common search approaches under this category are greedy search methods.

    - **Greedy search methods**: under this category search an entire space of possible classifiers, with the search going in two directions: forward and

backward selection. In forward selection, a greedy search algorithm begins with an empty ensemble, to which classifiers are added in an iterative manner. In backward selection, the algorithm starts with an ensemble that contains all possible classifiers and then removes classifiers that do not increase the performance of the ensemble. Adding and removing classifiers is performed using the selection (or evaluation) function. This method has been adopted in many studies, including [69, 13].

2. **Clustering based method:** In this method, a clustering algorithm is used to cluster a model space and then each cluster is pruned independently. The aim of pruning each cluster is to increase diversity amongst the members of an ensemble. [72, 52, 40] are examples of methods that use clustering-based selection.

3. **Ranking based method:** Selection methods that belong to this category start by arranging the classifiers in a model pool according to some evaluation functions (e.g. accuracy, diversity, etc.). The top k classifiers are then chosen to produce the final ensemble output. [56, 60] are methods that implement arrangement in selecting classifiers. Selecting the best classifier is a special case of this method, where k=1.

4. **others:** This category encompasses any method that does not fall into any of the previous categories.

2.3.5.2 Dynamic Classifier Selection

Dynamic classifier selection is based primarily on selecting a set of classifiers that classify each data instances in a testing dataset. This feature means that a pool of models is searched before any testing instance is classified to find a suitable set of classifiers that predicts the probability of correctly classifying that instance.

Alceu et al. [10] comprehensively reviewed different dynamic classifier selection methods. Figure 2.3 illustrated a suggested taxonomy for dynamic classifier selection methods ([10]).

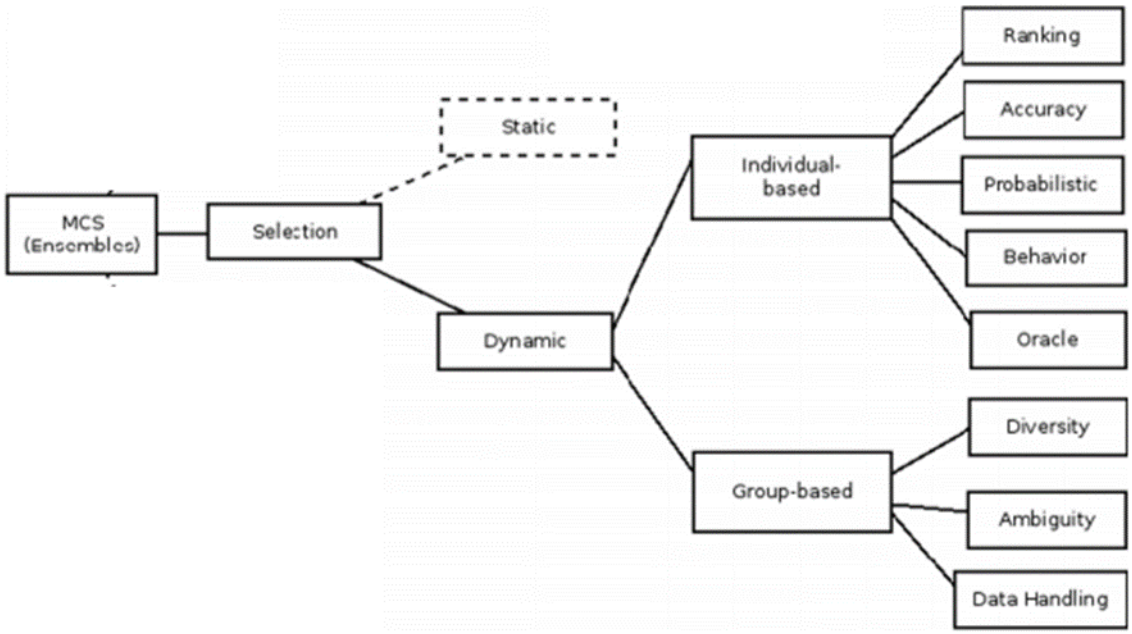


Fig. 2.3 Taxonomy of dynamic classifier selection methods proposed by [10]

.

As indicated in Figure 2.3, dynamic classifier selection methods can be divided into two main categories: individual and group-based methods. Individual methods select models on the basis of the effectiveness of the models, whereas group-based approaches choose models by combining the individual performance of a model with other information related to an entire model group (e.g. overall diversity of an ensemble). Each method and the studies that proposed dynamic classifier selection methods are fully described in [10].

2.4 Mining large datasets

Large datasets are a natural result of our growing dependence on computer systems in our daily lives. Organisations, such as Internet and email service providers, telecommunications enterprises, health research centres, companies that process credit card

transaction data and airlines, store huge amounts of data in their servers. Storing and integrating these data are expensive processes that result in monetary and storage-related losses for companies. Consequently, such enterprises are eager to explore different ways of data handling, specifically to extract hidden or unknown knowledge that may provide benefits to company operations. The vast amount of data and the rising need to comprehensively examine such data motivate machine learning researchers to introduce machine learning and data mining algorithms that can deal with massive datasets. These algorithms can be useful in applications such as webpage categorisation that involves dealing with gigabytes of data. Handwriting, face, and image recognition tasks are other examples of classification that entail handling voluminous data.

When a data mining algorithm is applied to a large dataset, two important factors should be considered: space and time. Despite rapid improvements in computer hardware, the available main memory of a computer still fails to hold huge datasets for mining, thus prompting the need to find data mining methods that deal with large data in such a way without the need of that an entire dataset remains in the main memory.

In addition to the space problem, time (modelling and prediction time) is a critical factor in dealing with very large datasets. "If the learning time does not scale linearly (or almost linearly) with the number of training set instances, it will eventually become infeasible to process very lager datasets" [37, p.346].

This section discusses the approaches that are proposed by researchers to accelerate the mining of large datasets.

2.4.1 Speeding up the data mining process for large datasets

Algorithms that work in an incremental manner are necessary to reduce the time required to build a model and ensure the feasibility of mining under a limited memory and a large dataset size. In the fields of machine learning and data mining, many approaches to speeding up these processes have been put forward. Some of these approaches are as follows ([28]):

- **Sampling:** According to [37], sampling is sometimes called the trivial approach, which is grounded in the principle of building a classification model from a subset of a training set. Researchers who introduced this approach argue that the performance of a classification model often increases very slowly after the model is trained by a significant number of training instances. If this is the case, then verifying model performance is easily accomplished through observation during a holdout test set for training sets of different sizes. This method therefore sacrifices accuracy for speed.

- **Distributed data mining:** In many real-world applications, vast amounts of data are available but are often geographically distributed in different sites. International banking and health information centres, for instance, are located in different regions or countries, and consequently, so are the data that are handled by these companies. The classical data mining approach depends on a central processing strategy. However, central data storage is ineffective and infeasible in some applications for the following reasons ([95, P157]):

  - Storage cost
  - Communication cost
  - Computational cost
  - Data privacy and sensitivity

  The idea of distributed data mining is to build a model from each dataset by using any data mining algorithm, after which the results derived by algorithms are combined. Figure 2.4 illustrates the topical architecture of distributed data mining techniques. Distributed data mining focuses primarily on combining various results for each dataset [58, p.998]. Model interaction and meta-learning are examples of approaches used to combine the results of distributed data mining.

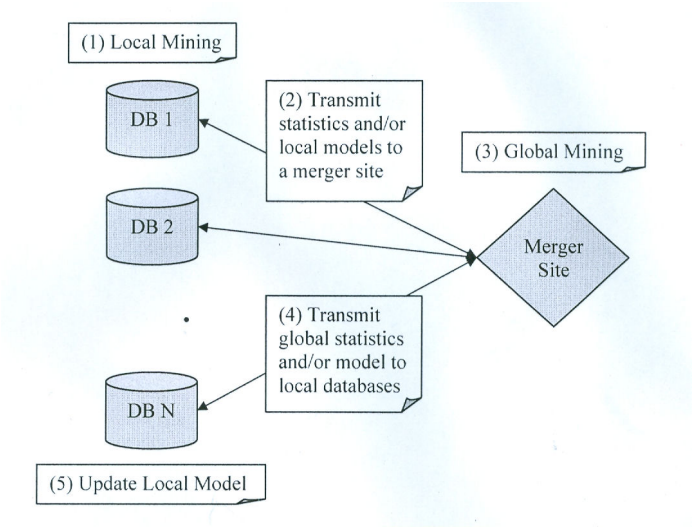


Fig. 2.4 The topical architecture of distributed data mining techniques [26]

- **Parallel data mining:** On massive datasets, which may be measured in terabytes (even bigger), using traditional data mining algorithms with a single processor is infeasible. Employing a single processor to build a model from terabytes of data within an acceptable timeframe is very difficult. To address the problems presented by the immense amount of data available and the limitations of a single computer's power, parallel data mining techniques are proposed as a solution to tackle the problem of of the time needed for mining big dataset and to reduce the computational cost of the data mining process.

- **Ensemble approaches:**

  Bagging and boosting are two of the most popular ensemble calculation techniques. Many researchers have confirmed that these techniques achieve better accuracy than that accomplished with a single classifier over relatively small datasets. For very large datasets, however, bagging and boosting are limited in their effectiveness because they depend on sequential procedures that are unsuitable for such datasets [17].

  To solve the problem of sequential procedures, Lazarevic and Obradovic [51] developed a distributed boosting algorithm for shared memory systems with a

small number of processors that can deal with large datasets. The experiment featured the examination of many datasets; the large set was a Cover Type data that was obtained from the UCI repository and contains 581,012 instances with 54 attributes and 7 class labels [51]. "They achieved the same or slightly better prediction accuracy than standard boosting, and they also observed a reduction in the cost of learning and memory requirements for their data sets" [17]. As a result of memory usage reduction, parallel boosting presents good acceleration for Cover Type datasets [51].

Chawla et al. [17] proposed the DIvote and DRvote algorithms, which are distributed versions of the Ivote and Rvote introduced by [8].In [17], eight datasets were examined. The two largest datasets analysed were the Cover Type and Protein datasets. The Cover Type dataset, was also used in [51], was divided into 149,982 instances for a training set and 431,030 instances for a testing set. The Protein dataset, also called the Jones dataset, was divided into 209,529 for training and 17,731 for testing. The main result indicates DIvote as a promising approach for very large data sets.

Peng et al. [71] put forward an approach that combines stratified random sampling, MCLP classification and a majority vote ensemble as a means of handling the problem of mining a massive dataset. Two different datasets were used and introduced in the KDD classification cups of 1999 and 2004. From the 1999 KDD dataset, 444,000 instances were used for training and 595,080 instances were employed for tasting. From the 2004 KDD dataset, 44,000 instances were used for training and 6,000 were adopted for testing. The core findings show that the proposed approach outperforms standard stratified random sampling on both datasets.

Another promising approach for mining large datasets is the ensemble of core vector machine (CVM), which was proposed by Tsang et al [91] as an upgrade to SVM. In [92], the ensemble version of CVM was used to classify two large

datasets, namely, the usps and face datasets; these can be found in [90]. The usps data set consists of 266,079 training patterns and 75,383 testing patterns with 676 attributes and 2 class labels. The face training set consists of 346,260 training patterns and 24,045 testing patterns with 361 attributes and 2 class labels. The experiment shows that incorporating orthogonality constraints into the CVM ensemble generates a more robust performance than that achieved with bagging based on SVM. Additionally, the CVM ensemble was 10–100 times faster than the original SVM ensemble.

Another method for handling large datasets is described in by Tuv [97]. This method is called the hybrid GTB-RF, which is a combination of a boosting algorithm (gradient tree boosting) and random forest.

2.4.2 Computing frameworks that can deal with big datasets

As mentioned in Section 1.1, the increasing availability of data has driven the development of several computing platforms that are developed to deal with the problem of mining large dataset. Some of these platforms are described below.

2.4.2.1 Hadoop

Hadoop is a Java-programming framework that facilitates the mining of very large datasets in distributed computing environment [63]. It is an open source software sponsored by the Apache Software Foundation. Hadoop works on MapReduce, which is a programming model developed by Google [81] for processing large datasets. For data mining tasks, Hadoop provide a large library of algorithms that can perform classification, clustering and regression. More information about this software can be found on the Hadoop website [HAd].

2.4.2.2 Spark

Spark was also sponsored by the Apache Software Foundation and is regarded as an extension of software that works on the MapReduce model (e.g. Hadoop). One of the main speed-related features of Spark is its ability to run computations in memory, but the system also rapidly performs on MapReduce for complex applications running on disk [42]. Spark can run a programme 100 times faster than Hadoop-MapReduce in the main memory and 10 time faster on disk [Spa]. Spark also provides multiple APIs in Java, Python and SQL, thus making it a highly accessible programme [42].

2.5 Summary

In this chapter, the relevant literature for ensemble methods is reviewed starting by illustrating the different ways to create an ensemble of classifiers. Then the main factors that affect the performance of an ensemble of classifiers are dissected. Among these factors, the most commonly used diversity measures are highlighted, and the methods and techniques used for model selection are also reviewed. The main decision-making strategies for the ensemble method are also summarised.

In addition to the ensemble methods literature, this chapter (see section 2.4) also discusses the challenges which arise from the problem of mining big datasets in a classification context and reviews the works that have been proposed by researchers in order to solve some of these problems. Finally, this chapter concludes by describing some computing platforms that can deal with big data.

CHAPTER 3

METHODOLOGY AND RESEARCH DESIGN

3.1 Introduction

This chapter describes how the research is designed and the methods and tools used to achieve the aim of this work. It also presents the datasets used to test our proposed methods, as well as the main characteristics of the computing facilities, data mining tools used in the experiments and the methods for evaluating the statistical significance of the results.

The chapter is structured as follows. Section 3.2 delineates the manner by which this study is carried out. Section 3.2 illustrates the reserach design and explains the methods and algorithms used in our ensemble of classifiers, and Section 3.3 presents the main characteristic of the datasets and the preprocessing steps that are implemented on the datasets. Section 3.4 describes the statistical significance tests that are carried out, and Section 3.5 provides the specifications of the computing environment in which the experiments are conducted.

3.2 Research Design and Methods

As prompted by the nature of the research problem -mining large datasets-and the defined research objectives, an appropriate methodology and adaptable platform are

essential to carry out the experiments for the identified research tasks. The divide-and-conquer approach is adopted because it is a typical solution in cases wherein a huge dataset cannot be loaded and processed in the main memory of a computing facility. This approach can be applied in machine learning and data mining via ensemble methods given that one of the approaches to creating an ensemble of classifiers is to manipulate training datasets (see Chapter 2).

Figure 3.1 shows the framework for an ensemble of classifiers that is constructed from data subsets partitioned from a single training dataset. The framework entails three principal phases: partitioning, modelling and combining. Each phase allows for the use of different methods and requires the consideration of several issues in designing our ensemble. The phases of the framework are described in detail below.

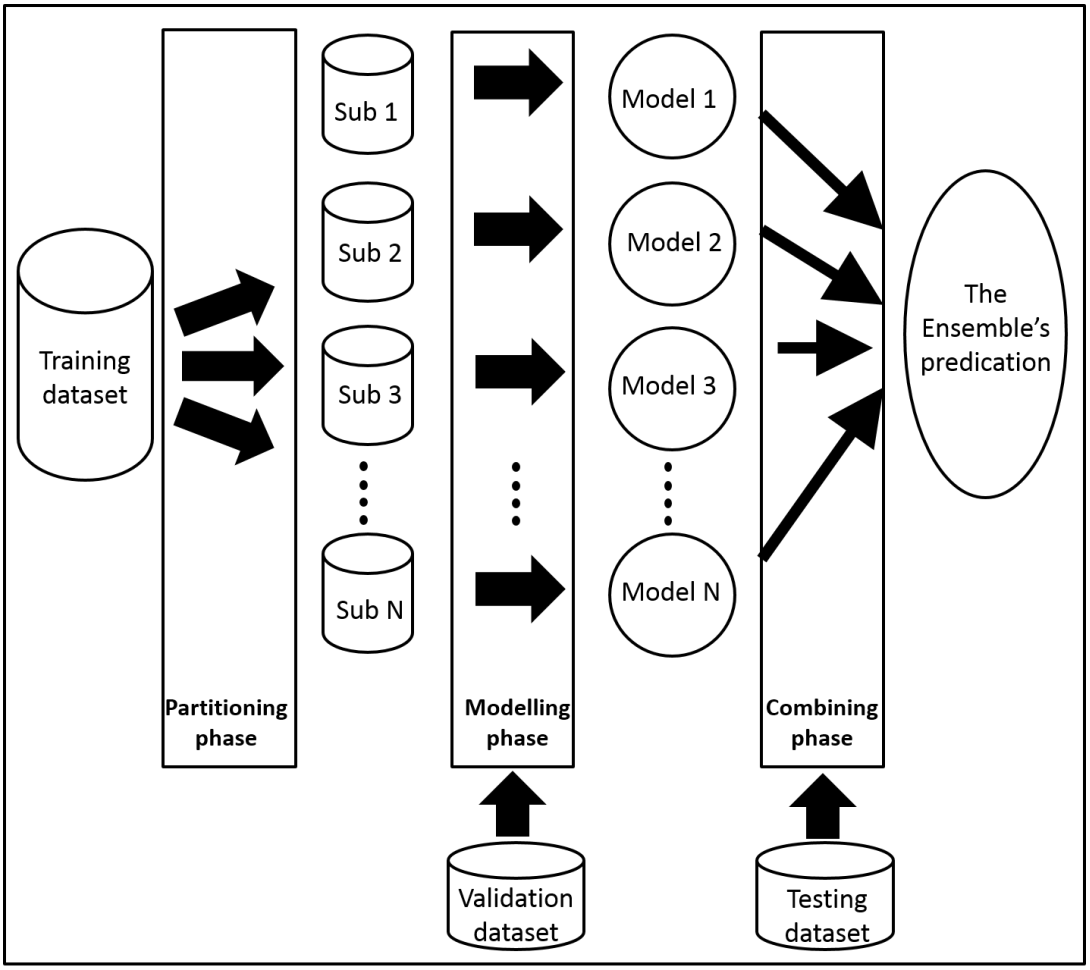


Fig. 3.1 Ensemble framework used in this Reserach

3.2.1 Partitioning phase

The partitioning phase is aimed primarily at dividing a training dataset (Tr) into N data subsets. Two approaches are commonly adopted to create different numbers of small subsets from a single dataset: sampling and partitioning. Sampling can be performed with or without replacement, and partitioning is normally employed to divide an original dataset into a set of N disjoint subsets.Certain issues should be taken into account before proceeding to partitioning. Possible main issues are listed as follows:

- What data partitioning or sampling methods should be used?
- How do we choose a suitable subset size for partitioning a single training dataset?
- How many data subsets should be used?

In order to choose a suitable partitioning strategy for our ensemble, a pilot study [24] is performed using 5 datasets from the 13 datasets mentioned in section 3.3. These datasets include Adult, Cover type, IJCNN01, KDD99 and Web. The aim of this study is to empirically investigate the effect of different partitioning methods and subset size on ensemble accuracy in dealing with big datasets. In our pilot study, three commonly used data partitioning methods are examined. These approaches include Sequential non-overlapping partitioning, Round robin partitioning and Sampling without replacement. The experimental results show that an ensemble of classifiers constructed from disjointed datasets is preferable to other partitioning methods, results which are also confirmed by a study conducted by Chawla [16]. The published version of the pilot study is available in appendix B.1.

Depending on the results from our pilot study in this research, in cases wherein sampling without replacement is unsuitable owing to the specific requirements of an experimental design, a combined partitioning method (i.e. combination of sampling with and without replacement) is adopted; otherwise, sampling without replacement

is used to partition a single big dataset. The used combined partitioning method is described in detail in section 4.3.3.

The second most important issue in this phase is the manner by which an appropriate size for partitioned data subsets is chosen. This matter is one of the predefined objectives of our research. Such size selection is usually encountered in tasks wherein a single large dataset requires division into smaller manageable data subsets. Should a data subset be maximised or minimised? Resolving this problem first necessitates determining whether data subset size affects the performance of an ensemble of classifiers that is constructed from data subsets. These issues are investigated in chapter 4. Following the investigation described in chapter 4, a method is proposed for identifying the patterns of the relationship between the relationship between the relative size (Rt) Rt and the ensemble accuracy $acc(\phi)$, as well as facilitating the selection of the best data subset size for dataset partitioning. The method is detailed in chapter 5.

3.2.2 Modelling phase

The purpose of the modelling phase is to create models (classifiers) from the available data subsets. Models are created by training these data subsets with a machine learning algorithm that creates a homogeneous ensemble of classifiers. Multiple machine learning algorithms may be employed to create a heterogeneous ensemble, but we opt for a homogeneous ensemble because our research objectives do not cover the investigation of the effects of using different machine learning algorithms on the performance of an ensemble. Similar to partitioning, modelling necessitates the examination of important factors:

- What is the suitable base learner to use?
- How many models should be created?
- Is constructing a model from each available data subset necessary?

It should be noticed that determining the best base learner is beyond the scope of this research. In the early stages of this reserach a variant of SVM, Core Vector machine (CVM) [93], was used. However, it was clear that the CVM is very time consuming and have not been efficient for dealing with big data. Thus the decision tree learning algorithm is employed as the base learner given its efficiency and sensitivity to changes in data inputs. More specifically, we use the C4.5 decision tree algorithm [73] to produce a classifier from each dataset.

In this phase, our concern centres on the number of created models, particularly when numerous data subsets are available, and the verification of whether creating a model from each available data subset is necessary. The manner by which these issues are addressed to is described in the next phase because they are related to model selection methods.

3.2.3 Combination phase

The last phase in any ensemble method involves combining the predictions of all the individual models created in the modelling phase. Several approaches are used to combine classifier predictions, with some simply integrating the predictions of all available models and others combining the predictions of a subset of created models. The techniques that merge only the optimal subsets of available models are often called model selection methods. A number of combination and model selection methods are discussed in Chapter 2. The issues relevant to the combination phase are listed below:

- What is the appropriate fusion method to be used?
- Is using a model selection method necessary?
- What is the best model selection technique for ensembles that deal with large datasets?

A common practice in ensemble approaches is to use model selection techniques in the combination phase for the purpose of choosing an optimal subset from a model

pool (MP) that contains all the models generated during the modelling phase. The aim of model selection is to improve the performance of an ensemble by enabling the selection of a set of models that can obtain best (or near the best)) ensemble performance.

Both static and dynamic model selection methods depend on the availability of MPs. In cases wherein a large number of data subsets are to be processed, creating an MP involves considerable time and resources. This expensive task raises important questions regarding the modelling and combination phases: Are training all available data subsets and subsequently applying model selection techniques really necessary? Do training all the data subsets and selecting only some of them to formulate the final prediction of an ensemble translate to time and resource wastage? These questions are investigated in Chapter 6 and a selective modelling method is proposed as an approach that iteratively combines model creation and selection to save time and resources when numerous data subsets are available. While majority voting is employed as a fusion strategy.

3.3 Datasets and Pre-processing

Thirteen datasets are obtained from the UCI Machine Learning Repository [54], the KEEL Dataset Repository [1] and the LIBSVM Dataset Repository [15] for preprocessing. Table 3.1 lists the main characteristics of the 'raw' datasets (i.e. before they are subjected to preprocessing procedures) while Table 3.2 lists the main characteristics of the datasets after applying the pre-processing procedures..

The preprocessing applied to the datasets proceeds as follows:

1. **Creating testing and validation datasets**

   Some of the thirteen datasets are initially divided into training and testing datasets while others comes as a single file as illustrated in Table 3.1. In the cases where testing dataset is already provided, a validation dataset is created

Table 3.1 Main characteristics of the 13 datasets

Dataset	No. of Training Instances	No. of Testing Instances	No. of Attributes	No. of classes
Adult [15]	32,561	16,281	123	2
Census [54]	199,523	99,762	40	2
Connect4 [1]	67,557	NA	42	3
Cover Type [15]	522,910	58,102	54	2
FARS [1]	100,067	NA	30	8
HIGGS [54]	11,000,000	NA	28	2
IJCNN01 [15]	49,990	91,701	22	2
KDD99 [15]	4,898,431	311,029	127	2
Poker [1]	1,025,010	NA	10	10
Shuttle [1]	435,00	14,500	10	7
Skin [54]	245,057	NA	4	2
SUSY [54]	5,000,000	NA	18	2
Web [15]	49,749	14,951	300	2

by randomly selecting 20% of the instances in the training dataset. In situations wherein a single dataset file is available, a testing dataset is first created by selecting 30% of the original file. The remaining 70% of the file is then divided into two datasets that represent training and validation datasets, as described earlier. The validation and testing datasets are created using the StratifiedRemoveFolds filter, which is available in Weka [34]. Using this filter ensures that the generated datasets are characterised by the same class distribution as the original dataset as suggested in [117].

2. **Selection of Attribute**

   Attribute selection technique is applied on the training dataset. The InfoGainAttributeEval filter, which is available in Weka [34], is used to perform attribute selection. All the attributes with score greater or equal to 0.01 were selected.

3. **Balancing the training dataset**

   The class distribution in each training dataset is balanced by resampling the instances of the minority class with the use of the SMOTE filler, which is also available in Weka [34].

4. **Convert multi-class problem to binary-class problem**

   Multi-class classification problem involve assigning each of the data instances into one of k classes [105]. In this reserach our multi-class datasets is out of our reserach scope, so the multi-class dataset is converted into a binary class dataset as follows:

   - Connect4 dataset: The class attribute in this dataset [1] originally contains three different classes. The class values indicate whether a player of the connect4 6×7 grid game is going to win, lose or draw. To convert this dataset into a binary classification problem, the data instances that belong to the loss and draw classes are assigned a single class value called 'loss_or_draw'.
   - Shuttle dataset: The class attribute in this dataset is a nominal attribute, and its values vary between 1 and 7. All instances that belong to the class value that is not equal to 1 are joined under one class label. More details about the original dataset and class values can be found in [54] and [1].
   - FARS dataset: This dataset contains statistical data on car accidents in the US in 2001. The class attribute contains eight nominal values that describe the level of injury suffered. To transform this dataset into a binary classification problem, the class attribute is transformed to describe the existence of an injury. For further details on the original dataset, the reader is referred to [1].
   - Poker dataset: The class attribute in this dataset takes 10 different nominal values that describe the poker hand obtained. To transform this dataset into a binary classification problem, the class attribute is converted in a way that describes whether a player's hand can be recognised as a poker hand. More details about the original dataset are found in [1].

Table 3.2 Main characteristics of the datasets used in this reserach after applying the preprocessing procedures

Dataset	No. Training Instances	No. Validation Instances	No. Testing Instances	No. of Attributes
Adult [15]	34,536	9,768	16,281	60
Census [54]	245,389	59,859	99,762	31
Connect4 [1]	48,173	9,458	20,268	43
Cover Type [15]	366,037	156,873	58,102	39
FARS [1]	89,648	14,010	30,021	30
HIGGES [54]	5,390,000	2,310,000	3,300,000	28
IJCNN01 [15]	60,138	14,997	91,701	7
KDD99 [15]	1,361,892	1,469,530	311,029	47
Poker [1]	574,004	143,502	307,503	11
Shuttle [1]	54,331	8,700	14,500	10
Skin [54]	216,974	34,308	73,518	4
SUSY [54]	2,800,000	700,000	1,500,000	19
Web [15]	64,839	14,925	14,951	262

3.4 Method for Statistical Evaluation

Statistical significance and correlation tests are conducted when necessary (see Chapters 4, 5 and 6). The statistical significance and correlation tests are discussed in Sections 3.4.1 and 3.4.2, respectively.

3.4.1 Statistical significance tests

Statistical significance tests are generally used to assess the performance of a proposed ensemble method against other methods, and test selection is determined by experimental design. In this study, two different tests are used.

3.4.1.1 Wilcox's signed-rank test

Wilcox's signed-rank test [102] is a nonparametric test that can serve as an alternative to a matched-pair t-test. The use of this test in the field of machine learning is described in [39, p.233-238]. It can be used to compare two classifiers on a single or multiple domains.

3.4.1.2 Friedman test

The Friedman test [20] is a nonparametric test that is intended to compare multiple classifiers on multiple domains. The analysis test depends on the rank of each classifier on each dataset and not on accuracy. Because the test compares multiple classifiers on multiple domains, its results indicate whether one should reject the null hypotheses that no statistical difference in performance exists amongst classifiers. In the case of rejection, we use the critical difference (CD) diagram, which graphically represents overall performance; in the diagram, statistically similar classifiers are denoted by a slid bar [20].

3.4.2 Correlation analysis

Correlation coefficient analysis is generally carried out to test the strength and direction of the association between two variables. Pearson's and Spearman's correlation analyses are the two commonly adopted approaches to measuring correlation.

In this study, correlation analyses are conducted in two cases. It is used in Chapter 4 to determine whether Rt and $acc(\phi)$ are correlated. It is also used in chapter 6 to find out whether an association exists between diversity amongst the members of an ensemble and $acc(\phi)$. In both cases, Pearson's correlation analysis is performed.

3.5 The Used Computing Environment

To carry out the experiments for the studies described in Section 3.2 and all the other experimental procedures relevant to this work, the proposed framework is implemented with a multithreaded Java application. We implement our own software because we intend to propose new methods and algorithms, which necessitate finding an open source data mining package that will enable us to modify or incorporate new features into existing methods. WEKA [34], which is one of the most highly regarded open source data mining tools, is very useful but cannot be directly used in our experiments

given that it lacks a methods for measuring the diversity between models within our ensemble. Additionally, it cannot take advantage of parallel computing facilities or multithreading architecture. Nevertheless, because WEKA is an open source software, it is embedded in our Java programme as a JAR file, thus ensuring that all the collection of algorithms and methods available in WEKA are also accessible from our software.

Our Java application runs over the high-performance computing cluster supported by the Research and Specialist Computing Support Service at the University of East Anglia. More specifically, we run our application on a single computing node that uses a dual 8 core Intel E5-2670 2.6 GHz processor system (16 cores) with 32 GB of RAM. More information about the UEA high-performance computing cluster can be found in [UEA].

3.6 Summary

This chapter describes the design of the research and illustrates the ensemble framework (see Figure 3.1) that was adopted to achieve the aim and objectives of the research that were defined in chapter 1.

Our ensemble framework consisted of the following primary phases: partitioning, modelling and combining. In the partitioning phase, sampling without replacement was used as a partitioning method while combined partitioning (e.g., combination of sampling with and without replacements) was applied in the cases where it was unsuitable to use sampling without replacement owing to the specific requirements of the experimental design. While a decision tree was used as a base classifier , majority voting was used as a fusion strategy during the modelling and combining phases, respectively.

Finally, the chapter ends by describing the specifications of the computing environment that was used to conduct the experiments throughout the research period.

CHAPTER 4

HOW DOES DATA SUBSET SIZE INFLUENCE THE PERFORMANCE OF AN ENSEMBLE OF CLASSIFIERS?

4.1 Introduction

Nowadays, data from various digital media channels and activities can rapidly accumulate to such a magnitude that even a hight performance computer system is unstable to load such data into its main memory for analysis. This problem necessitates that large datasets be divided into smaller manageable subsets to overcome the capacity limitations of memory. 'Big dataset' is a relative term; in this research, it refers to any dataset that cannot be loaded or processed with the available main memory. Although many data partitioning strategies have been proposed, no study has systematically examined how the size of a partitioned subset influences the performance of machine learning and data mining methods. This chapter investigates empirically of the effects of partitioned data subset size on ensemble accuracy [$acc(\phi)$] when dealing with huge datasets.

The rest of the chapter is organised as follows. Section 4.2 introduces the problem and the aim of the corresponding experiment. Section 4.3 illustrates the experimental design, and Section 4.4 presents the experimental results. Section 4.5 ends the chapter with a summary.

4.2 Hypothesised Effects of the Relative Subset Size on the Accuracy of an Ensemble of Classifiers

Mining a big dataset is a very challenging task for data miners. As discussed in Chapter 2, even though some studies put forward techniques for handling huge datasets (e.g. [3, 55, 92, 17]), none of them analysed the influence of partitioned data subset size on the performance of an ensemble during mining large dataset. The majority of the literature on classifier ensembles (e.g. [23, 78, 109]) discusses the effects of sampling size on $acc(\phi)$ but does not present experimental studies that demonstrate the partitioned subset size effects on $acc(\phi)$.

Fundamentally, the relationship between the relative size (Rt) of partitioned data subsets and $acc(\phi)$, if such an association exists, is expected to correspond to one of the three hypothesised patterns depicted in Figure 4.1. P1 and P2 represent negative and positive monotonic relationships between Rt and $acc(\phi)$, respectively, and P3 denotes a case wherein a very weak or no relationship exists between Rt and $acc(\phi)$.

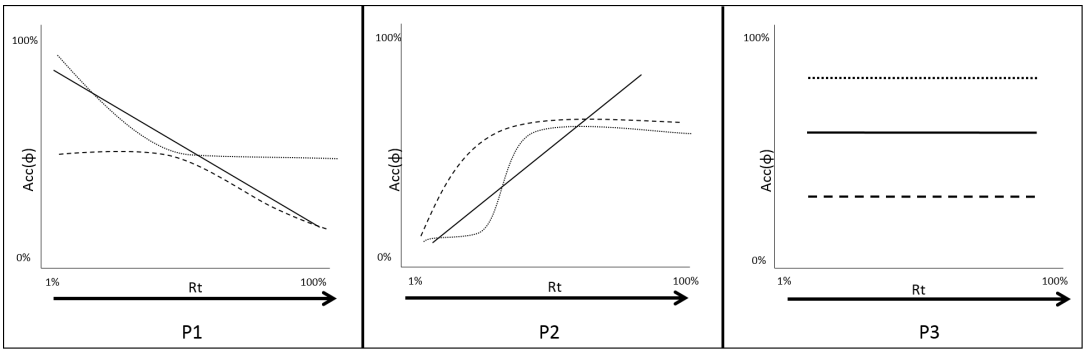


Fig. 4.1 Expected patterns of the relationship between Rt and $acc(\phi)$

This chapter aims to experimentally determine whether Rt and $acc(\phi)$ are related. In cases wherein a relationship exists between Rt and $acc(\phi)$, the objective is to categorise the patterns of the discovered relationship.

When dealing with big datasets, understanding the relationship between Rt and $acc(\phi)$ is a critical requirement because one of the most frequently applied techniques

for resolving memory constraints is the divide-and-conquer technique. Without such an understanding, choosing the best size for partitioning a single large dataset is difficult. A common solution in this case is to maximise data subset size, but a principal issue is whether such maximisation is indeed necessary and achieves better results. This chapter is intended to address this issue, with the experiment conducted to identify the patterns of the Rt–$acc(\phi)$ relationship.

The main concern of this work is the pattern exhibited by the aforementioned relationship and not the accuracy of the generated ensemble of classifiers. The performance of the ensemble on each dataset is therefore excluded from the analysis and discussion.

4.3 Experimental Design

With the goal of unravelling the patterns exhibited by the relationship between Rt and $acc(\phi)$, a necessary requirement is to examine the effects of different Rt values on $acc(\phi)$. To this end, varying Rt values are used to construct different ensembles of classifiers, wherein the only difference between two generated ensembles is the size of the data subsets employed to build the ensembles.

To gain a reliable investigation of the relationship, it is important to vary Rt values while fixing all other factors that may influence $acc(\phi)$ see Figure 4.2.

The details of the experimental design and how the factors are fixed in the experiment are described in Section 4.3.2.

4.3.1 Datasets

Thirteen datasets were used to determine the effects of Rt on $acc(\phi)$. Comprehensive information on the datasets is provided in Chapter 3. Table 3.2 summarises the main characteristics of these datasets.

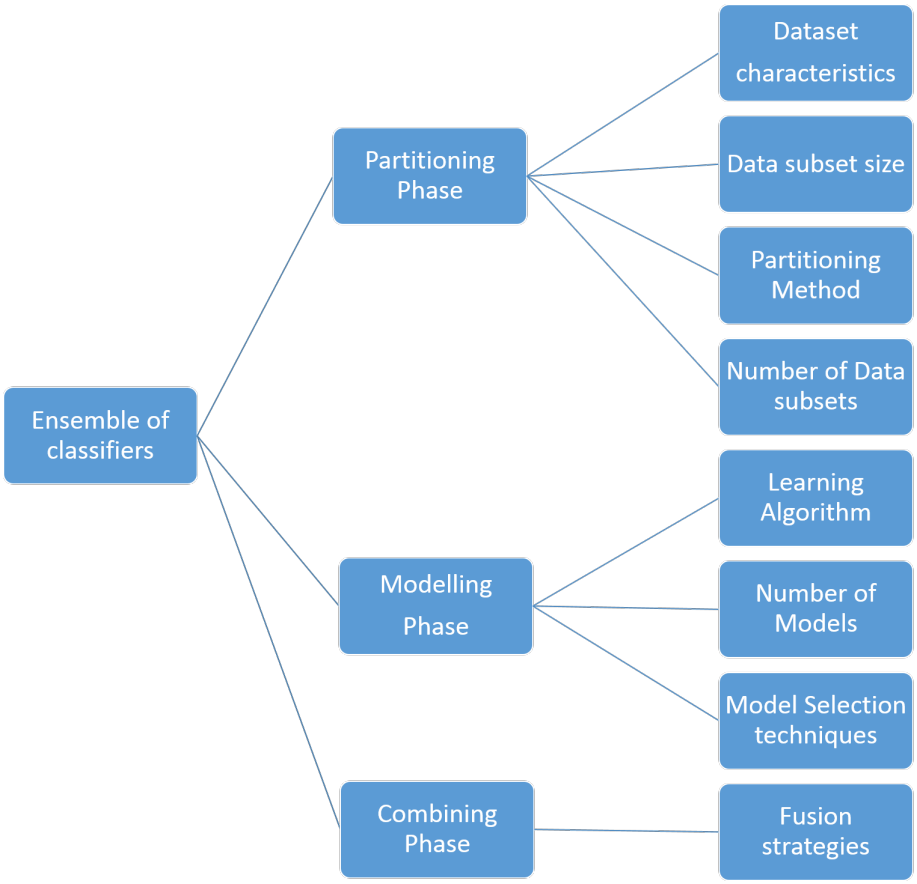


Fig. 4.2 Factors that may influence the performance of an ensemble of classifiers

4.3.2 Experimental procedure and setup:

- The experimental procedure is applied to all 13 datasets.
- To obtain a reliable representation of the relationship between Rt and $acc(\phi)$ (if such a relationship exists), the procedure is repeated 10 times for each dataset. In each run, new training (Tr) and validation (V) datasets are created, as described in Section 3.3. A separate testing dataset (TS) is used to evaluate all 10 ensembles created in the 10 the runs.
- Given that the experiment is aimed at ascertaining whether Rt affects $acc(\phi)$, we use different Rt values ranging from a given minimum Rt (Rt_{min}) to a maximum Rt (Rt_{max}), with a step size = 5%.
- For the 13 datasets, Rt_{min} is set to 1%.
- Rt_{max} depends on available memory of the computing system used. In our case, the maximum available memory (Mem_{max}) is 20 GB.
- Rt_{max} is set to 100% in all the datasets, except for the 3 bigger datatsets :HIGGS, KDD99 and SUSY datasets.
- Because of memory constraints, the Rt_{max} values of HIGGS, KDD99 and SUSY are 25%,34% and 40%, respectively.
- The following steps are performed in sequence as follows:

    1. Tr is partitioned into N data subsets by using the partitioning method described in Section 4.3.3. The size of each data subset is equal to Rt and N = 101.
    2. M models (classifiers) are induced using the j48 classifier, which is the Weka version of the traditional c4.5 decision tree, where M = N.
    3. The created M models are then evaluated over V and TS datasets..

4. Build an ensemble with M models.
5. Majority voting is used to combine the predictions of the M models.
6. Rt is increased by 5%.
7. Steps 1 to 7 are repeated until Rt = Rt_{max}.

4.3.3 Partitioning Method

The experimental design in this study necessitates the observation of ensemble accuracy [$acc(\phi)$] at different Rt values and a fixed N. In other words, an essential requirement is to vary data subset size without affecting number of created data subsets which is a goal that cannot be achieved by simple disjoint partitioning. To address this issue, a combined partitioning method is used. This combined method employ sampling with and without replacement approaches as follows:

Fundamentally, Rt should be varied from Rt_{min} up to Rt_{max}, where Rt_{min} and Rt_{max} are the lowest and maximum required Rt values. Both Rt_{min} and Rt_{max} are incorporated into the experimental design. N should also be specified in the experimental design and should be greater than or equal to $\lceil \frac{1}{Rt_{min}} \rceil$ to ensure that all the training instances are taken into consideration in the partitioning process.

Any time N data subsets with a size equal to Rt need to be created, partitioning is implemented in the following manner:

- If $\lceil \frac{1}{Rt} \rceil = N$, then sampling without replacement will be used to create N disjoint data subsets.
- If $\lceil \frac{1}{Rt} \rceil < N$, then sampling without replacement will be used to create n disjoint data subsets where $n = \lceil \frac{1}{Rt_{min}} \rceil$, and sampling with replacement will be used to create the rest of the K data subsets where K = N-n.

Figure 4.3 illustrates the adopted partitioning method. For clarification, let us assume that the experimental design necessitates the creation of nine data subsets from

a single Tr, where the size of each subset is = 20% of Tr. In this case, N=9 and Rt=0.2 under the proposed combination method. Five data subsets are created using sampling without replacement, and four are produced by sampling with replacement.

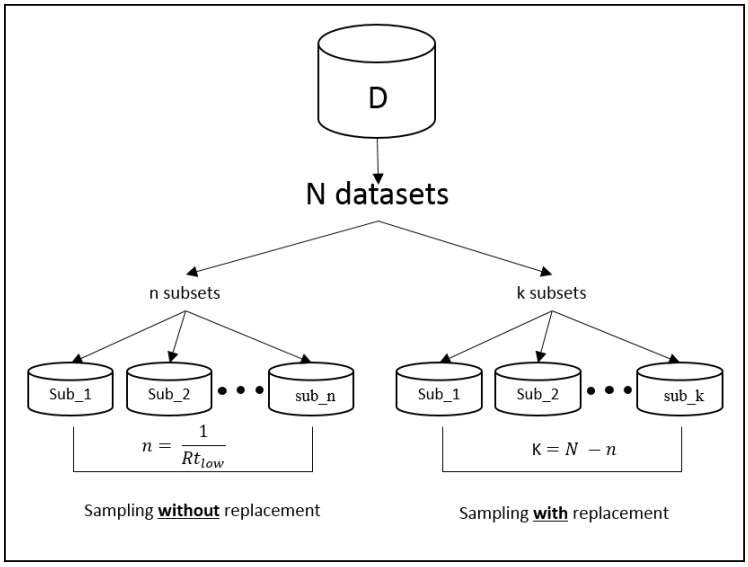


Fig. 4.3 Partitioning method

4.3.4 Evaluation Methods

In this chapter the accuracy is used as a performance measure, while Pearson correlation coefficient is used to confirm whether there is a association between Rt and $acc(\phi)$ when the existence of a linear relationship between Rt and $acc(\phi)$. In the case of the absence of the linear relationship Spearman's correlation.

4.4 Experimental Results and Evaluation

In this section the effects of Rt on $acc(\phi)$ are examined, and the empirical results for the 13 datasets are presented. The findings discussed in this section are an average of 10 runs. The influence of varying Rt values on $acc(\phi)$ for each dataset is discussed in Section 4.4.1, and the effects of Rt over all the 13 datasets are summarised in Section4.4.2.

The following notations are used as follows :

- $\overline{acc(M)}$: is the mean of the accuracy of individual models within an ensemble.
- r: is the Pearson product-moment correlation.
- r_s: is Spearman's correlation correlation coefficient.
- $IR_{(Rt_i,Rt_j)}$: is the improvement rate in ensemble accuracy between Rt_i and Rt_j, where IR can be calculated as follows

$$IR_{Rt_i,Rt_j} = \frac{acc(\phi)_{Rt_j} - acc(\phi)_{Rt_i}}{Rt_j - Rt_i} \tag{4.1}$$

  where

  – Rt_i and Rt_j are the Rt values that determine the period at which improvement in accuracy is calculated. In the calculation, $Rt_j > Rt_i$.

4.4.1 Effects of Rt on $Acc(\phi)$ for each dataset

4.4.1.1 Adult dataset

Figure 4.4 and Table 4.1 show that increasing Rt improves $acc(\phi)$ over both V and TS. The figure also indicates that Rt and $acc(\phi)$ demonstration a nonlinear relationship given that the improvement in accuracy is inconstant across all Rt values. The figure confirms a monotonic relationship between Rt and $acc(\phi)$ over V and TS and reflects two distinct behaviours between $acc(\phi)$ and Rt. These behaviours can divide the curves into two sections thus:

- **When Rt $\leq$ to 10%**: Where $acc(\phi)$ sharply increases as Rt increases, where $IR_{(1\%,10\%)} = 0.56\%$.
- **When Rt $>$ 10%**: Where $acc(\phi)$ minimally increases and levels off with a large Rt, where $IR_{(15\%,100\%)} = 0.02\%$.

The results also show that $acc(\phi)$ outperforms $\overline{acc(M)}$. Although the performance of ensembles is the focus of this chapter, a reliable and accurate $acc(\phi)$ helps to build up the confidence in believing the relationship identified by the ensembles. Furthermore, the ensembles consistently behave over V and TS for the entire range of Rt, implying the remarkable constancy of V and TS. Therefore, V can be used to help determine which Rt is the most appropriate size and select the set of models that best facilitates final ensemble prediction if necessary.

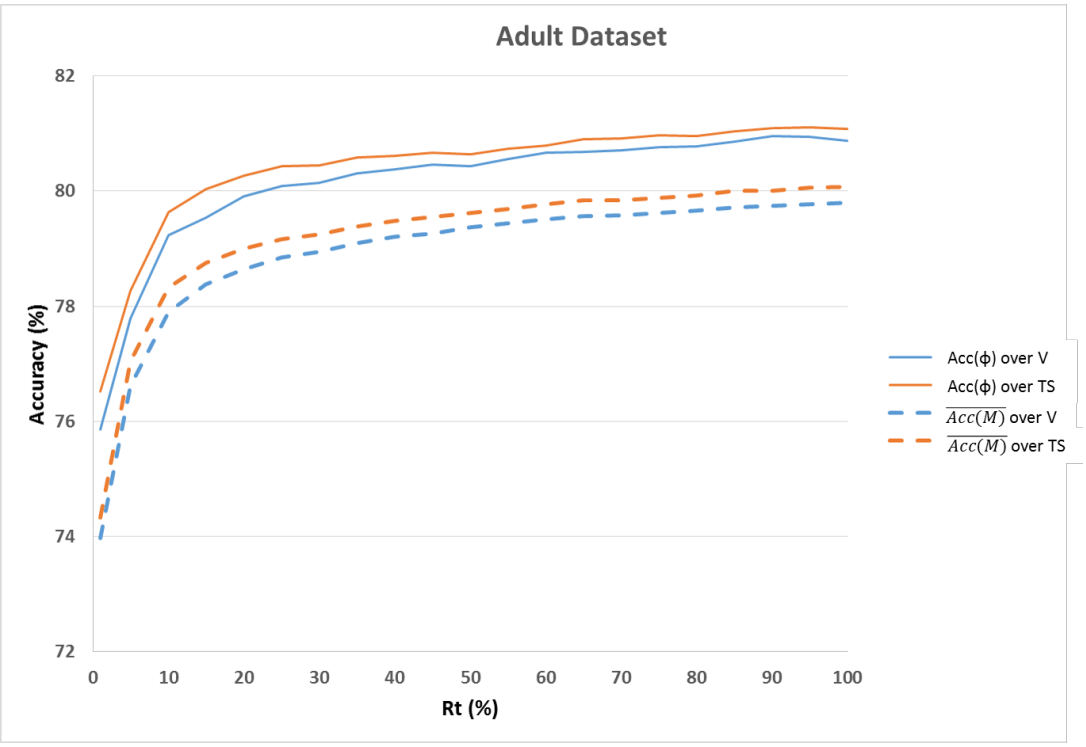


Fig. 4.4 Relationship between Rt and $Acc(\phi)$ over V and TS for the Adult dataset

The statistical evaluation of the relationship

Given the consistent behaviour of the ensembles on V and TS (Figure 4.4), the statistical analysis for the Adult dataset and all succeeding datasets reveals the analytical results on TS only. The complete statistical analysis results for all datasets over V and TS can be found in Appendix A.

We begin our analysis by calculating the correlation between Rt and $acc(\phi)$ at Rt $\leq$10% and Rt $>$10% by using Pearson's correlation coefficient. Then, the correlation

between the entire ranges of Rt and $acc(\phi)$ is calculated using Spearman's correlation coefficient. At $\leq$10%, a strong positive correlation exists between Rt and $acc(\phi)$ over TS, where $r = 0.991$. However, this result is statistically nonsignificant because the p = 0.087. This result is due to the small number of ensembles that were taken into consideration in the calculation of the correlation coefficient; that is, the performance of only three ensembles were adopted in the calculation. Conversely, a very strong positive correlation is identified when Rt >10%, where $r = 0.964$ and $p < 0.0005$. Examining the correlation between the entire ranges of Rt and $acc(\phi)$ values confirms the existence of a very strong positive correlation with $r_s = 0.994$ and $p < 0.0005$.

Table 4.1 $Acc(\phi)$ and $\overline{acc(M)}$ over V and TS for the Adult dataset

	validation		Testing	
Rt	$Acc(\phi)$	$\overline{acc(M)}$	$Acc(\phi)$	$\overline{acc(M)}$
1	75.85	73.97	76.53	74.33
5	77.79	76.62	78.28	77.04
10	79.23	77.90	79.63	78.32
15	79.54	78.37	80.03	78.75
20	79.91	78.65	80.27	78.99
25	80.09	78.84	80.43	79.17
30	80.15	78.95	80.45	79.25
35	80.31	79.09	80.58	79.38
40	80.38	79.20	80.61	79.48
45	80.46	79.27	80.67	79.56
50	80.43	79.37	80.63	79.62

	validation		Testing	
Rt	$Acc(\phi)$	$\overline{acc(M)}$	$Acc(\phi)$	$\overline{acc(M)}$
55	80.55	79.44	80.73	79.69
60	80.66	79.51	80.79	79.77
65	80.68	79.57	80.90	79.83
70	80.71	79.58	80.91	79.84
75	80.76	79.62	80.97	79.88
80	80.78	79.66	80.96	79.93
85	80.86	79.72	81.04	80.00
90	80.96	79.74	81.10	80.01
95	80.94	79.77	81.10	80.06
100	80.88	79.80	81.08	80.07

4.4.1.2 Census dataset

Figure 4.5 illustrates the relationship between Rt and $acc(\phi)$ in the Census dataset over V and TS. This relationship exhibits a pattern similar to that displayed by the relationship found in the Adult dataset. The figure and Table 4.2 show the following distinct behaviours between $acc(\phi)$ and Rt:

- **When Rt $\leq$ to 10%**: Where $acc(\phi)$ sharply increases as Rt increases, where where $IR_{(1\%,10\%)} = 0.53\%$.

- **When Rt $>$ 10%**: Where $acc(\phi)$ gradually increases, where $IR_{(15\%,100\%)} = 0.03\%$.

Given that the discovered association is monotonic in nature, Spearman's correlation coefficient is employed on the basis of the information provided in Table 4.2. A very strong positive correlation is found between Rt and $acc(\phi)$, where $r_s = 1$ and $p < 0.0005$. Pearson's correlation coefficient is used to test the association between $acc(\phi)$ and Rt in the two sections of the curve when Rt $\leq 10\%$ and Rt $> 10\%$. Similar to the Adult dataset, the Census dataset shows a nonsignificant correlation between Rt $\leq 10\%$ and $acc(\phi)$, where $r = 0.980$, $p = 0.126$. A strong correlation between Rt $> 10\%$ and $acc(\phi)$ is identified, where $r = 0.983$, $p < 0.0005$.

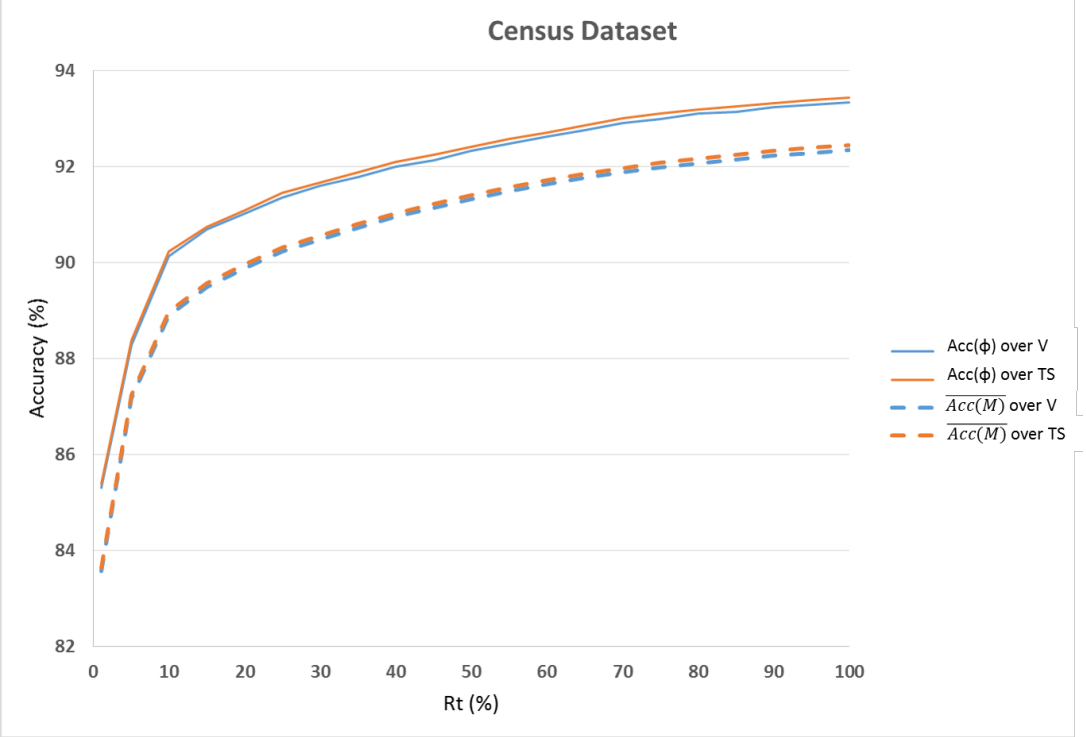


Fig. 4.5 Relationship between $Acc(\phi)$ and Rt over V and TS for the Census dataset

Table 4.2 $Acc(\phi)$ and $\overline{acc(M)}$ over V and TS for the Census dataset

	validation		Testing	
Rt	$Acc(\phi)$	$\overline{acc(M)}$	$Acc(\phi)$	$\overline{acc(M)}$
1	85.30	83.57	85.39	83.63
5	88.28	87.17	88.37	87.24
10	90.12	88.89	90.23	88.97
15	90.69	89.49	90.74	89.57
20	91.02	89.89	91.09	89.97
25	91.36	90.22	91.46	90.31
30	91.60	90.48	91.67	90.57
35	91.77	90.72	91.88	90.81
40	92.00	90.96	92.09	91.03
45	92.14	91.15	92.24	91.23
50	92.32	91.31	92.41	91.39

	validation		Testing	
Rt	$Acc(\phi)$	$\overline{acc(M)}$	$Acc(\phi)$	$\overline{acc(M)}$
55	92.48	91.48	92.57	91.56
60	92.62	91.63	92.71	91.71
65	92.76	91.76	92.85	91.85
70	92.90	91.88	93.00	91.97
75	92.99	91.98	93.10	92.08
80	93.10	92.07	93.18	92.16
85	93.14	92.15	93.26	92.25
90	93.23	92.22	93.32	92.32
95	93.29	92.29	93.39	92.39
100	93.33	92.35	93.44	92.44

4.4.1.3 Connect4 dataset

In the Connect4 dataset, a monotonic relationship again arises, with the dataset reflecting two distinct behaviours (as shown in Figure 4.6 and Table 4.3) between $acc(\phi)$ and Rt:

- **When Rt $\leq$ to 20%**: Where $acc(\phi)$ sharply increases, where $IR_{(1\%,20\%)} = 0.57\%$.
- **When Rt $>$ 20%**: Where $acc(\phi)$ gradually increases, where $IR_{(25\%,100\%)} = 0.05\%$.

Rt and $acc(\phi)$ are very strongly positively correlated, where $r_s = 1$ over TS at a significance of $p < 0.0005$. A strong positive correlation is also identified in the two sections at Rt $\leq 20\%$ and Rt $> 20\%$, where $(r = 0.993; p = 0.001)$ and $(r = 0.955; p < 0.0005)$, respectively.

4.4.1.4 Cover Type dataset

Figure 4.7 depicts the relationship between $acc(\phi)$ and Rt over V and TS for the Cover Type dataset, and Table 4.4 shows the improvement in $acc(\phi)$ at an increasing Rt. A monotonic relationship is yet again identified, similar to observations on the

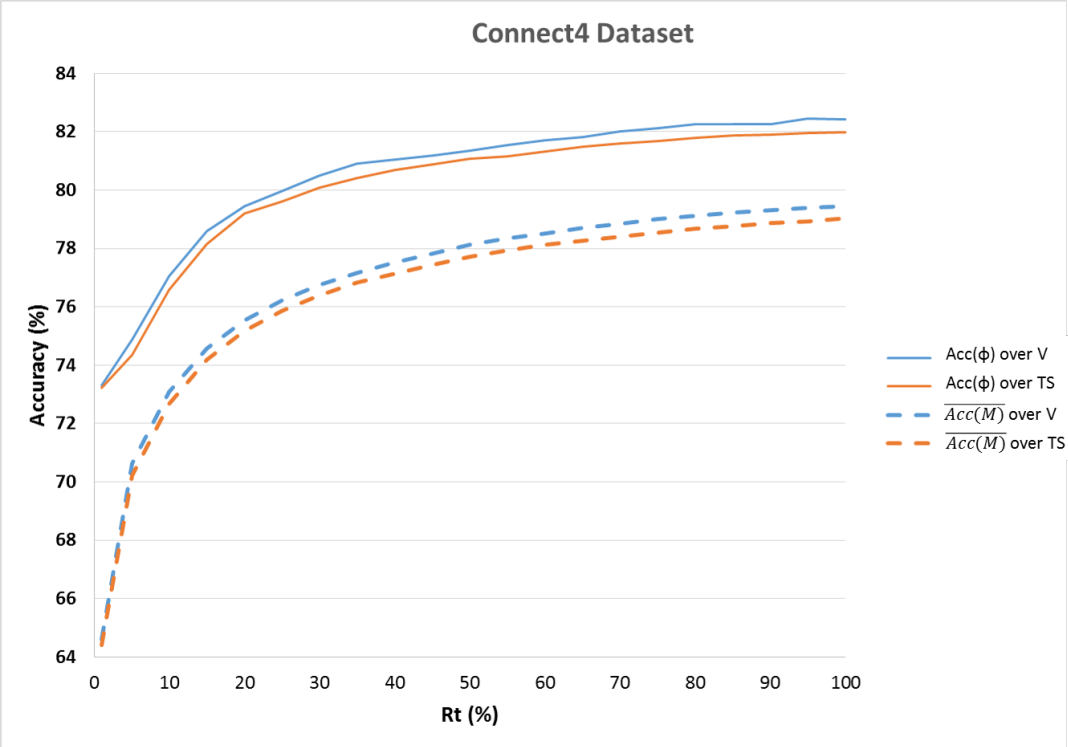


Fig. 4.6 Relationship between $Acc(\phi)$ and Rt over V and TS for the Connect4 dataset

Table 4.3 $Acc(\phi)$ and $\overline{acc(M)}$ over V and TS for The Connect4 dataset

	validation		Testing	
Rt	$Acc(\phi)$	$\overline{acc(M)}$	$Acc(\phi)$	$\overline{acc(M)}$
1	73.30	64.60	73.23	64.41
5	74.88	70.64	74.35	70.22
10	77.06	73.08	76.57	72.67
15	78.60	74.57	78.16	74.19
20	79.45	75.52	79.20	75.17
25	79.97	76.22	79.60	75.87
30	80.48	76.74	80.08	76.38
35	80.90	77.17	80.41	76.82
40	81.04	77.51	80.69	77.14
45	81.19	77.82	80.88	77.44
50	81.35	78.13	81.06	77.72

	validation		Testing	
Rt	$Acc(\phi)$	$\overline{acc(M)}$	$Acc(\phi)$	$\overline{acc(M)}$
55	81.54	78.33	81.16	77.92
60	81.71	78.51	81.33	78.11
65	81.82	78.71	81.47	78.27
70	82.01	78.84	81.59	78.40
75	82.12	79.01	81.67	78.55
80	82.24	79.12	81.79	78.67
85	82.26	79.22	81.87	78.77
90	82.26	79.31	81.90	78.87
95	82.45	79.40	81.96	78.94
100	82.42	79.45	81.98	79.02

previously discussed datasets. Rt and $acc(\phi)$ exhibit a very strong positive correlation over TS, as determined by Spearman's correlation coefficient, where $r_s = 0.999$ with a significance of $p < 0.0005$. Similar to the previously discussed datasets, the increase in $acc(\phi)$ is inconstant during the rise in Rt. Thus, two distinct behaviours of $acc(\phi)$ are observed (Figure 4.7 and Table 4.4):

- **When Rt $\leq$ to 20%**: Where $acc(\phi)$ sharply increases as Rt increases, where $IR_{(1\%,20\%)} = 0.65\%$.
- **When Rt $>$ 20%**: Where $acc(\phi)$ gradually increases, where $IR_{(25\%,100\%)} = 0.03\%$.

A strong positive correlation over TS is also identified using Pearson's correlation coefficient at Rt $\leq$ to 20% and Rt $>$ 20%, where $(r = 0.923; p = 0.025)$ and $(r = 0.936; p < 0.0005)$, respectively.

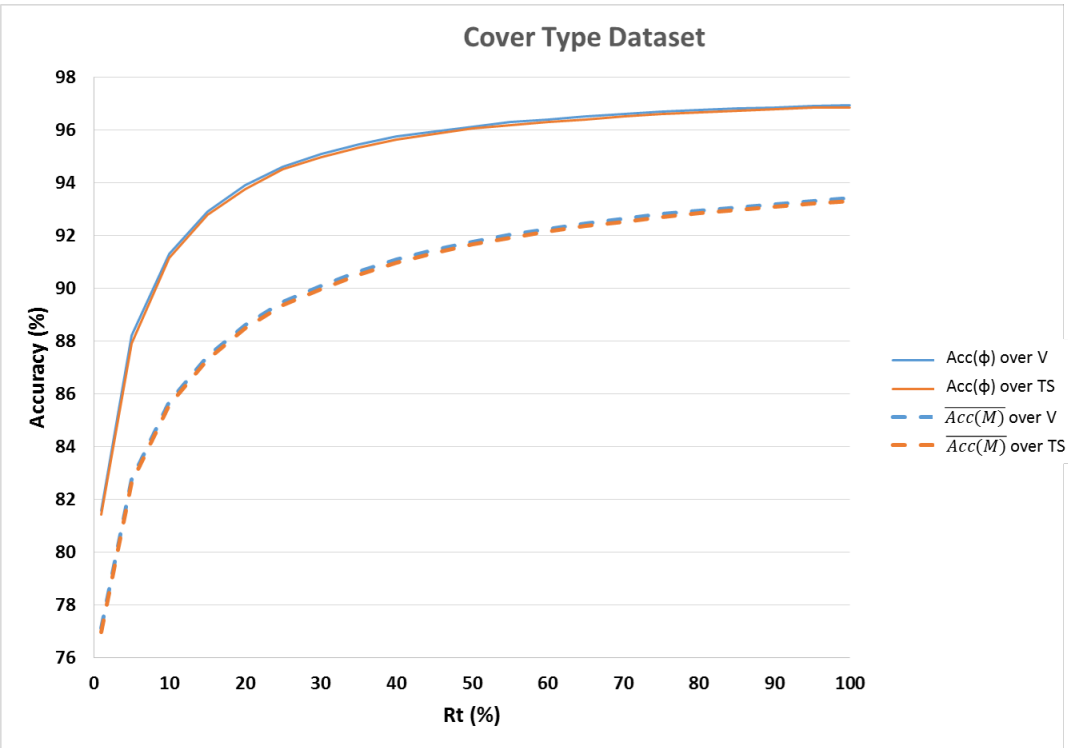


Fig. 4.7 Relationship between $Acc(\phi)$ and Rt over V and TS for the Cover Type dataset

Table 4.4 $Acc(\phi)$ and $\overline{acc(M)}$ over V and TS for the Cover Type dataset

	validation		Testing	
Rt	$Acc(\phi)$	$\overline{acc(M)}$	$Acc(\phi)$	$\overline{acc(M)}$
1	81.58	77.12	81.42	76.96
5	88.21	82.76	87.90	82.62
10	91.31	85.71	91.14	85.58
15	92.92	87.42	92.78	87.30
20	93.91	88.60	93.77	88.47
25	94.59	89.47	94.50	89.35
30	95.08	90.10	94.97	89.97
35	95.44	90.63	95.33	90.52
40	95.74	91.07	95.65	90.97
45	95.94	91.44	95.84	91.33
50	96.13	91.76	96.05	91.65

	validation		Testing	
Rt	$Acc(\phi)$	$\overline{acc(M)}$	$Acc(\phi)$	$\overline{acc(M)}$
55	96.29	92.02	96.19	91.91
60	96.40	92.25	96.29	92.14
65	96.52	92.45	96.39	92.35
70	96.60	92.63	96.50	92.52
75	96.68	92.81	96.59	92.71
80	96.74	92.94	96.67	92.84
85	96.80	93.07	96.71	92.97
90	96.85	93.18	96.77	93.09
95	96.91	93.30	96.83	93.20
100	96.94	93.40	96.83	93.30

4.4.1.5 FARS dataset

The relationship between $acc(\phi)$ and Rt in the FARS dataset is illustrated in Figure 4.8. This relationship pattern differs from those found in the four previous datasets in that it corresponds to a weak linear relationship between $acc(\phi)$ and Rt (Figure 4.8).

Table 4.5 indicates that the $acc(\phi)$ over V and TS slightly increases between Rt = 1% and Rt = 10%, after which no improvement in $acc(\phi)$ occurs with rising Rt. This moderate relationship is confirmed by the correlation analysis, where $r_s = 0.521$ and $p < 0.015$ over TS. The accuracy improvement rate for this dataset is minimal compared with those obtained for the previous datasets, where $IR_{(1\%,100\%)} = 0.002\%$ over V and TS .

4.4.1.6 HIGGS dataset

Given the huge size of the HIGGS dataset and memory constraints, the relationship between $acc(\phi)$ and Rt in this dataset is only partially explored . Figure 4.9 depicts the relationship between $acc(\phi)$ and Rt over V and TS, where Rt increases from 1% to Rt_{max}, which is equal to 25%. Rt_{max} is selected depending on the available memory in the computing environment where the experiment is performed. The specifications of the computing environment are described in Chapter 3.

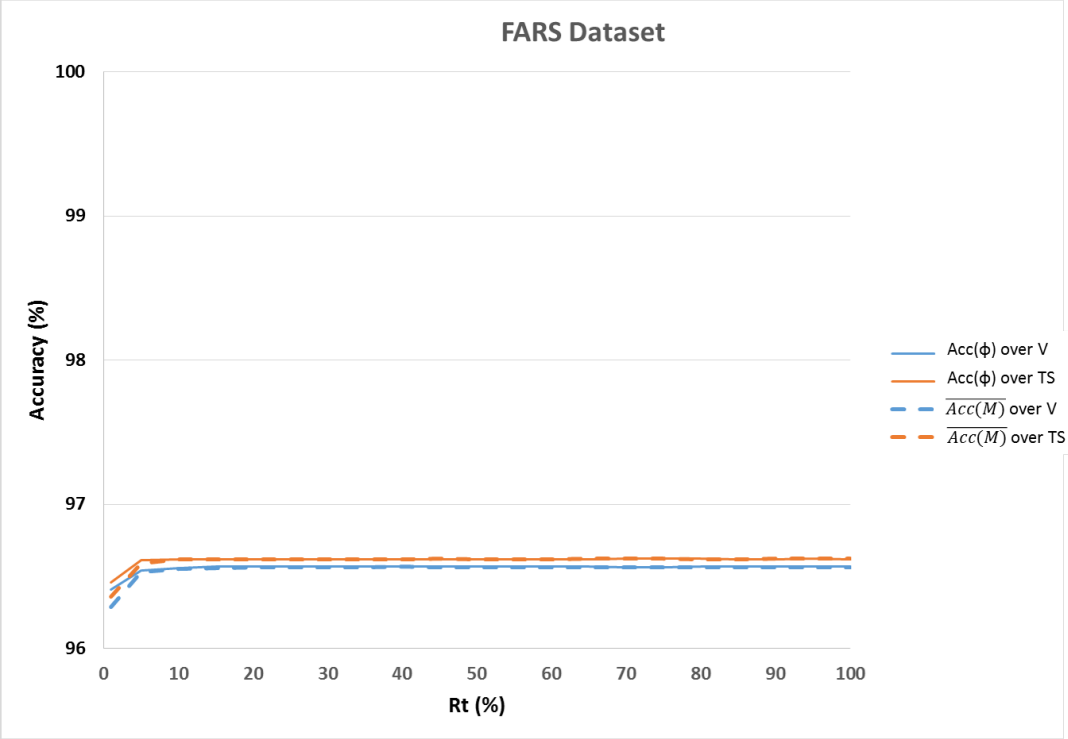


Fig. 4.8 Relationship between $Acc(\phi)$ and Rt over V and TS for the FARS dataset

Table 4.5 $Acc(\phi)$ and $\overline{acc(M)}$ over V and TS for the FARS dataset

	validation		Testing	
Rt	$Acc(\phi)$	$\overline{acc(M)}$	$Acc(\phi)$	$\overline{acc(M)}$
1	96.41	96.29	96.46	96.36
5	96.54	96.53	96.61	96.59
10	96.56	96.55	96.62	96.62
15	96.57	96.56	96.62	96.62
20	96.57	96.56	96.62	96.62
25	96.57	96.56	96.62	96.62
30	96.57	96.56	96.62	96.62
35	96.57	96.56	96.62	96.62
40	96.57	96.57	96.62	96.62
45	96.57	96.56	96.62	96.62
50	96.57	96.56	96.62	96.62

	validation		Testing	
Rt	$Acc(\phi)$	$\overline{acc(M)}$	$Acc(\phi)$	$\overline{acc(M)}$
55	96.57	96.56	96.62	96.62
60	96.57	96.56	96.62	96.62
65	96.57	96.56	96.62	96.62
70	96.57	96.56	96.62	96.62
75	96.57	96.56	96.62	96.62
80	96.57	96.56	96.62	96.62
85	96.57	96.56	96.62	96.62
90	96.57	96.56	96.62	96.62
95	96.57	96.56	96.62	96.62
100	96.57	96.56	96.62	96.62

Figure 4.9 and Table 4.6 indicate a monotonic relationship between $acc(\phi)$ and Rt. $IR_{(1\%,25\%)} = 0.06$, and a very strong positive correlation is identified, where $r_s = 1$ over V and TS with a significance of $p < 0.0005$.

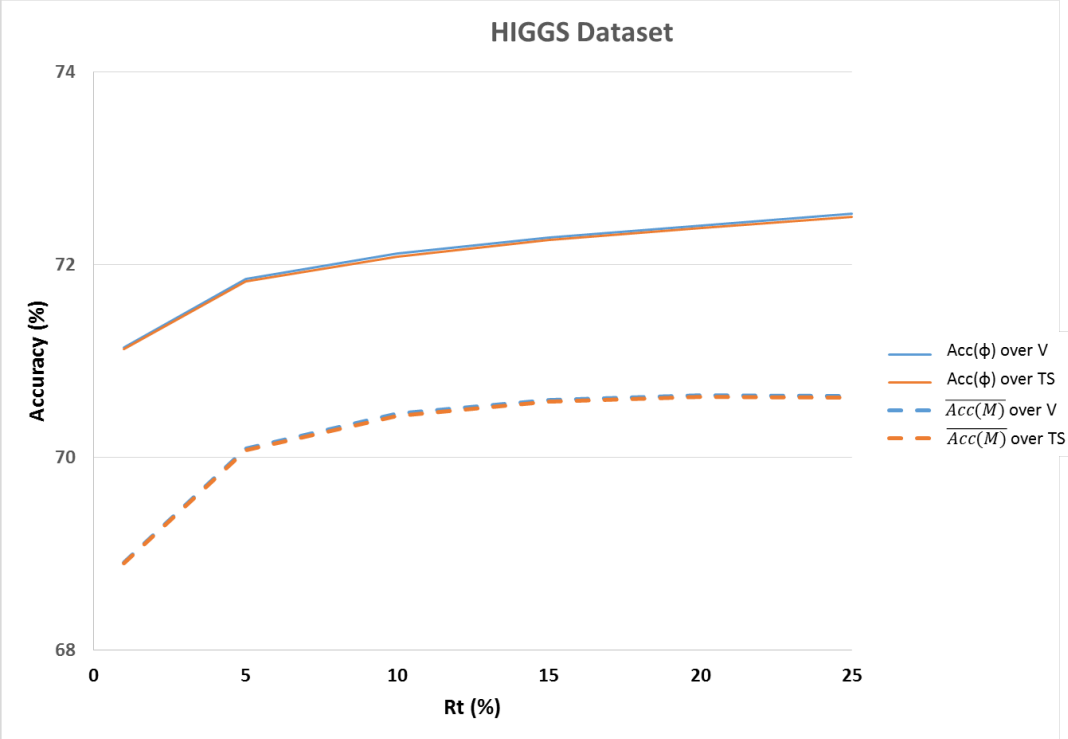


Fig. 4.9 Relationship between $Acc(\phi)$ and Rt over V and TS for HIGGS dataset

Table 4.6 $Acc(\phi)$ and $\overline{acc(M)}$ over V and TS for the HIGGS dataset

	validation		Testing	
Rt	$Acc(\phi)$	$\overline{acc(M)}$	$Acc(\phi)$	$\overline{acc(M)}$
1	71.14	68.91	71.13	68.90
5	71.85	70.09	71.83	70.07
10	72.11	70.45	72.09	70.43
15	72.28	70.60	72.26	70.58
20	72.41	70.65	72.38	70.63
25	72.53	70.64	72.50	70.62

4.4.1.7 IJCNN01 dataset

The relationship between $acc(\phi)$ and Rt in the IJCNN01 dataset is similar to the relationship patterns identified in the Adult, Census, Connect4 and Cover Type datasets.

This dataset reflects a monotonic relationship between $acc(\phi)$ and Rt, where the former continuously improves with increasing Rt (Figure 4.10). Table 4.7 lists the values that reflect the $acc(\phi)$–Rt relationship at each Rt.

Again the behaviour of the enhancement in $acc(\phi)$ with increasing Rt can be divided into two different behaviours:

- **When Rt $\leq$ to 20%**: Where $acc(\phi)$ sharply increases, where $IR_{(1\%,20\%)} = 0.20\%$.
- **When Rt $>$ 20%**: Where $acc(\phi)$ gradually increases, where $IR_{(25\%,100\%)} = 0.02\%$.

The correlation analysis confirms a very strong association between $acc(\phi)$ and Rt, where $r_s = 0.999$ over TS with a significance of $significant\, p < 0.0005$. The same holds for both sections of the curve at Rt $\leq$ to 20% and Rt $>$ 20%, where $(r = 0.942; p = 0.017)$ and $(r = 0.951; p < 0.0005)$, respectively.

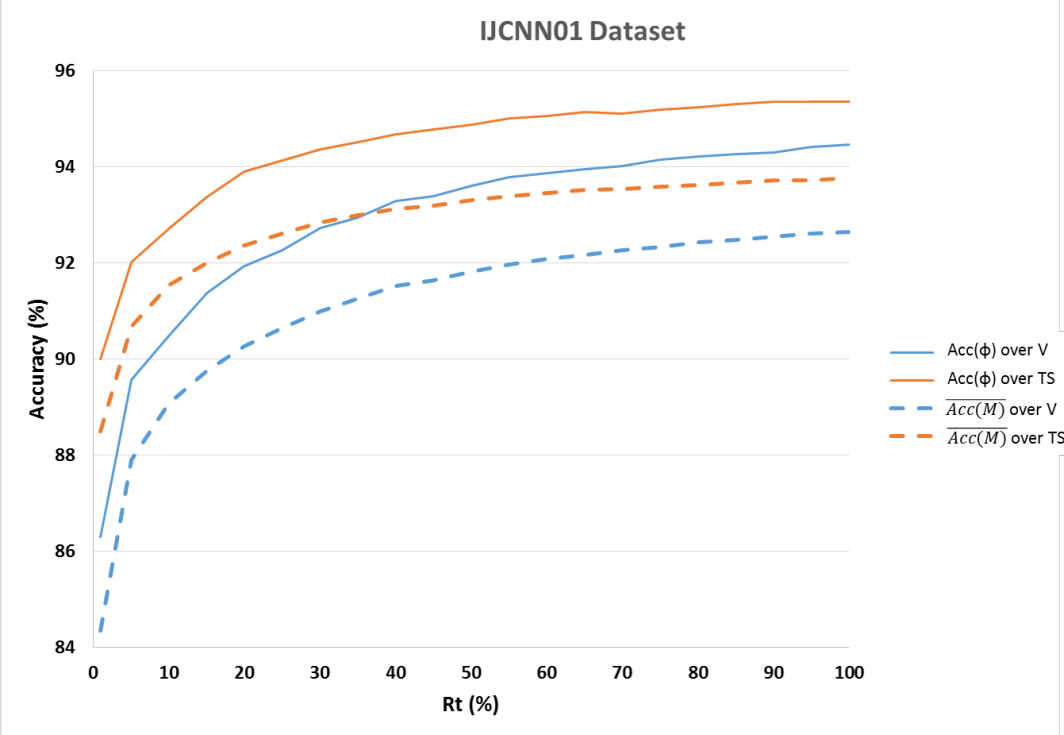


Fig. 4.10 Relationship between $Acc(\phi)$ and Rt over V and TS for the IJCNN01 dataset

Table 4.7 $Acc(\phi)$ and $\overline{acc(M)}$ over V and TS for the IJCNN01 dataset

	validation		Testing	
Rt	$Acc(\phi)$	$\overline{acc(M)}$	$Acc(\phi)$	$\overline{acc(M)}$
1	86.29	84.35	90.01	88.49
5	89.58	87.90	92.01	90.68
10	90.48	89.08	92.71	91.54
15	91.37	89.76	93.38	92.01
20	91.93	90.27	93.89	92.37
25	92.27	90.65	94.13	92.61
30	92.73	90.99	94.37	92.84
35	92.94	91.26	94.52	92.98
40	93.29	91.52	94.68	93.12
45	93.38	91.64	94.78	93.19
50	93.60	91.82	94.87	93.30

	validation		Testing	
Rt	$Acc(\phi)$	$\overline{acc(M)}$	$Acc(\phi)$	$\overline{acc(M)}$
55	93.78	91.96	95.01	93.38
60	93.88	92.08	95.06	93.46
65	93.96	92.17	95.13	93.52
70	94.02	92.26	95.11	93.54
75	94.15	92.34	95.19	93.59
80	94.21	92.43	95.24	93.63
85	94.26	92.48	95.31	93.67
90	94.30	92.55	95.35	93.71
95	94.41	92.61	95.36	93.72
100	94.46	92.64	95.36	93.76

4.4.1.8 KDD99 dataset

Figure 4.11 shows the relationship between $acc(\phi)$ and Rt over V and TS for the KDD99 dataset. This is the only dataset that exhibits a slight difference in the shapes of the relationship patterns over V and TS. Figure and Table 4.8 indicate a linear relationship between Rt and $acc(\phi)$ over the V dataset, where $IR_{(1\%,34\%)} = 0.002$. Over TS, a monotonic relationship between Rt and $acc(\phi)$ is identified, with $IR_{(1\%,34\%)} = 0.01$.

Although the patterns over V and TS differ in terms of shape (Figure 4.11), the correlation analysis does not show a huge difference between the patterns. $r_s = 0.994$ over V, and $r_s = 0.976$ over TS with a significance of $significant\, p < 0.0005$.

Table 4.8 $Acc(\phi)$ and $\overline{acc(M)}$ over V and TS for the KDD99 dataset

	validation		Testing	
Rt	$Acc(\phi)$	$\overline{acc(M)}$	$Acc(\phi)$	$\overline{acc(M)}$
1	93.92	93.91	92.18	92.18
5	93.97	93.96	92.22	92.24
10	93.99	93.98	92.63	92.55
15	93.99	93.98	92.63	92.58
20	93.99	93.98	92.65	92.59
25	93.99	93.98	92.66	92.61
30	93.99	93.98	92.66	92.61
34	93.99	93.98	92.65	92.60

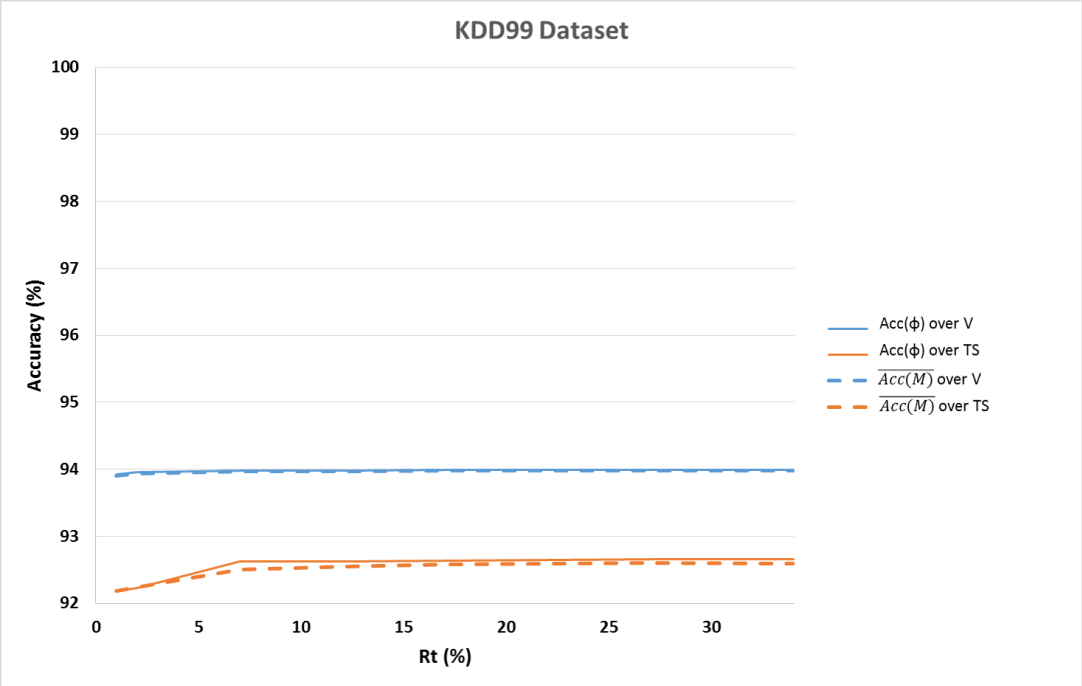


Fig. 4.11 Relationship between $Acc(\phi)$ and Rt over V and TS for the KDD99 dataset

4.4.1.9 Poker dataset

The Poker dataset reflects a unique pattern of the $acc(\phi)$–Rt relationship. Figure 4.12 suggests that the relationship pattern on this dataset is similar to those identified in the Adult and Cover Type datasets. However, a deeper examination of the figure and a consideration of the information in Table 4.9 reveal that $acc(\phi)$ continuously improves with growing Rt until $acc(\phi)$ peaks at Rt=70%. Beyond this point, $acc(\phi)$ gradually deteriorates. This pattern [i.e. improvement in $acc(\phi)$ with increasing Rt up to a certain point, then declining $acc(\phi)$] is not observed in the previous datasets. The curve shown in Figure 4.12 can be classified into four sections, depending on the behaviour of $acc(\phi)$:

- **When Rt $\leq$ to 20%**: $acc(\phi)$ fluctuates with increasing Rt.
- **When 20%<Rt$\leq$40%** : $acc(\phi)$ sharply increases, where $IR_{(25\%,40\%)} = 1.19\%$.
- **When Rt 40%<Rt$\leq$70%**: $acc(\phi)$ gradually increases, where $IR_{(45\%,70\%)} = 0.10\%$.

- **When Rt $>$ 70%**: Where $acc(\phi)$ gradually decreases, where $IR_{(25\%,100\%)} = -0.03\%$.

The decline in $acc(\phi)$ is reflected in the correlation analysis, where $r_s = 0.751$ with a significance of $p < 0.0005$ over V and TS. Conversely, a strong positive correlation is identified when $\leq$ to 20% and 20%<Rt$\leq$40%, where $r = 0.921; p = 0.026$ and $r = 0.998; p = 0.002$, respectively. The correlation is statistically nonsignificant at Rt $>$ 70% $r = -0.804; p = 0.54$.

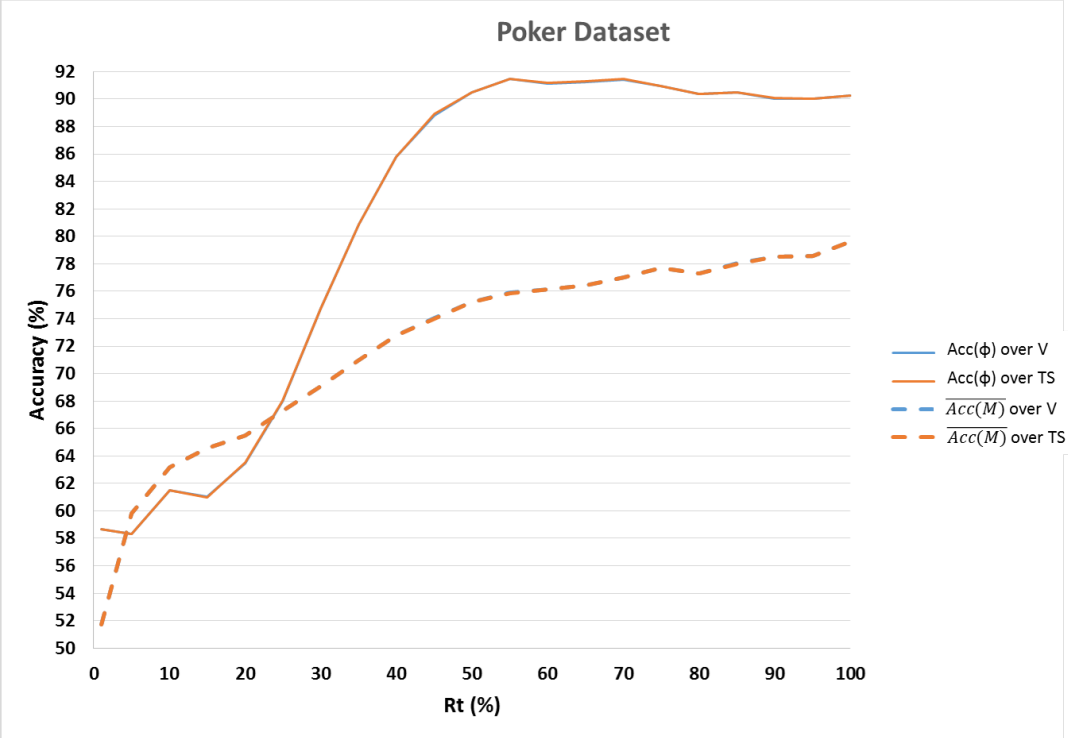


Fig. 4.12 Relationship between $Acc(\phi)$ and Rt over V and TS for the Poker dataset

4.4.1.10 Shuttle dataset

The pattern of the relationship between $acc(\phi)$ and Rt is very similar to that identified in the FARS dataset. Figure 4.13 and Table 4.10 show that the $acc(\phi)$ over the V dataset slightly improves when Rt increases from 1% to 30%. Beyond this value, $acc(\phi)$ no longer shows effects from rising Rt. The pattern over the TS dataset is

Table 4.9 $Acc(\phi)$ and $\overline{acc(M)}$ over V and TS for the Poker dataset

	validation		Testing	
Rt	$Acc(\phi)$	$\overline{acc(M)}$	$Acc(\phi)$	$\overline{acc(M)}$
1	58.69	51.73	58.65	51.73
5	58.32	59.82	58.32	59.80
10	61.48	63.19	61.49	63.17
15	61.02	64.58	61.00	64.57
20	63.46	65.50	63.52	65.49
25	67.99	67.30	68.03	67.28
30	74.74	69.08	74.75	69.07
35	80.87	71.02	80.88	71.01
40	85.78	72.77	85.81	72.76
45	88.83	74.03	88.90	74.02
50	90.46	75.22	90.48	75.19

	validation		Testing	
Rt	$Acc(\phi)$	$\overline{acc(M)}$	$Acc(\phi)$	$\overline{acc(M)}$
55	91.45	75.89	91.45	75.87
60	91.15	76.12	91.16	76.11
65	91.26	76.42	91.29	76.41
70	91.42	77.02	91.45	77.01
75	90.94	77.72	90.95	77.71
80	90.34	77.32	90.39	77.30
85	90.48	78.04	90.50	78.02
90	90.03	78.51	90.06	78.49
95	90.00	78.56	90.03	78.54
100	90.24	79.63	90.23	79.61

similar to that over V, except that $acc(\phi)$ constantly improves with an Rt growth of up to 65%. After this, $acc(\phi)$ remains steady until Rt=100% is reached.

A moderate positive correlation is found between Rt and $acc(\phi)$, where $r_s = 0.687$ with a significance of $p < 0.001$ over V, and $r_s = 0.686$ with $p < 0.0005$ over TS. The overall $IR_{(1\%,100\%)} = 0.004$ over V and TS.

Table 4.10 $Acc(\phi)$ and $\overline{acc(M)}$ over V and TS for the shuttle dataset

	validation		Testing	
Rt	$Acc(\phi)$	$\overline{acc(M)}$	$Acc(\phi)$	$\overline{acc(M)}$
1	99.63	99.64	99.65	99.64
5	99.88	99.78	99.87	99.73
10	99.96	99.87	99.95	99.82
15	99.97	99.91	99.95	99.86
20	99.97	99.93	99.95	99.89
25	99.97	99.94	99.95	99.91
30	99.97	99.95	99.95	99.92
35	99.98	99.96	99.97	99.93
40	99.98	99.96	99.97	99.93
45	99.98	99.96	99.98	99.94
50	99.98	99.96	99.98	99.94

	validation		Testing	
Rt	$Acc(\phi)$	$\overline{acc(M)}$	$Acc(\phi)$	$\overline{acc(M)}$
55	99.98	99.96	99.99	99.95
60	99.99	99.96	99.98	99.95
65	99.99	99.96	99.99	99.95
70	99.98	99.96	99.99	99.96
75	99.98	99.96	99.99	99.96
80	99.98	99.96	99.99	99.96
85	99.98	99.97	99.99	99.96
90	99.98	99.97	99.99	99.96
95	99.98	99.97	99.99	99.97
100	99.98	99.97	99.99	99.97

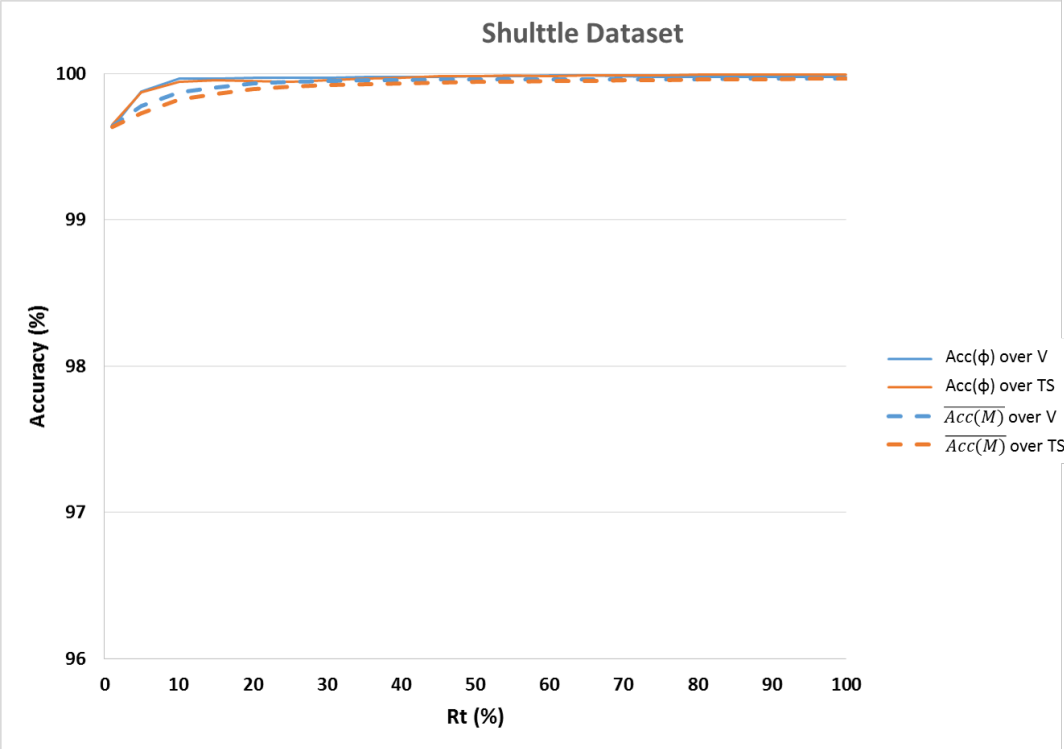


Fig. 4.13 Relationship between $Acc(\phi)$ and Rt over V and TS for the Shuttle dataset

4.4.1.11 Skin dataset

Figure 4.14and Table 4.11 illustrate the relationship between $acc(\phi)$ and Rt over V and TS for the Skin dataset. A slight increase in $acc(\phi)$ is observed with a 5% growth in Rt, after which minimal enhancement and levelling off in $acc(\phi)$ occur with increasing Rt. The curve that describes the relationship between $acc(\phi)$ and Rt can generally be divided into two distinct sections, depending on the type of influence exerted by Rt on $acc(\phi)$. The sections are as follows:

- **When Rt $\leq$ to 5%**: $acc(\phi)$ sharply increases, where $IR_{(1\%,5\%)} = 0.29\%$.
- **When Rt $>$ 5%**: $acc(\phi)$ slightly increases, where $IR_{(10\%,100\%)} = 0.003\%$.

The correlation analysis confirms a very strong correlation between $acc(\phi)$ and Rt, where $r_s = 0.993$ over TS with a significance of $p < 0.0005$.

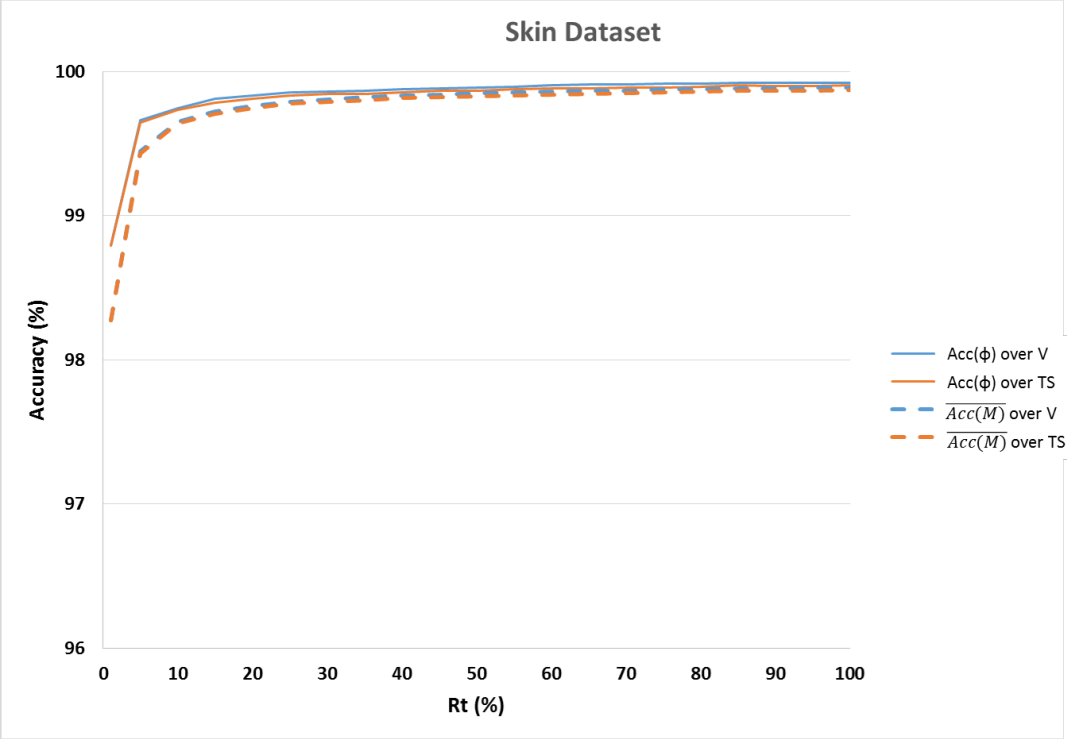


Fig. 4.14 Relationship between $Acc(\phi)$ and Rt over V and TS for the Skin dataset

Table 4.11 $Acc(\phi)$ and $\overline{acc(M)}$ over V and TS for the Skin dataset

	validation		Testing	
Rt	$Acc(\phi)$	$\overline{acc(M)}$	$Acc(\phi)$	$\overline{acc(M)}$
1	98.80	98.28	98.79	98.28
5	99.66	99.45	99.65	99.43
10	99.75	99.65	99.74	99.64
15	99.81	99.72	99.79	99.71
20	99.83	99.76	99.81	99.75
25	99.85	99.79	99.83	99.78
30	99.86	99.81	99.84	99.79
35	99.87	99.82	99.84	99.80
40	99.88	99.83	99.86	99.82
45	99.89	99.84	99.87	99.83
50	99.89	99.85	99.87	99.83

	validation		Testing	
Rt	$Acc(\phi)$	$\overline{acc(M)}$	$Acc(\phi)$	$\overline{acc(M)}$
55	99.90	99.86	99.88	99.84
60	99.90	99.86	99.88	99.84
65	99.91	99.87	99.89	99.85
70	99.91	99.87	99.89	99.85
75	99.92	99.88	99.89	99.86
80	99.92	99.88	99.90	99.86
85	99.92	99.88	99.90	99.87
90	99.92	99.89	99.90	99.87
95	99.92	99.89	99.90	99.87
100	99.92	99.89	99.90	99.87

4.4.1.12 SUSY dataset

The relationship between $acc(\phi)$ and Rt over V and TS for the SUSY dataset is illustrated in Figure 4.15. The pattern is similar to that found in the HIGGS database for multiple reasons. First, because of memory constraints, only part of the relationship pattern is explored from Rt=1% to Rt=40%. Second, $acc(\phi)$ marginally improves with growing Rt, where the overall $IR_{(1\%,40\%)} = 0.03$. This value can be calculated using the information presented in Table 4.15. Finally, a very strong positive correlation is identified, where $r_s = 1$ over V and TS with $p < 0.0005$.

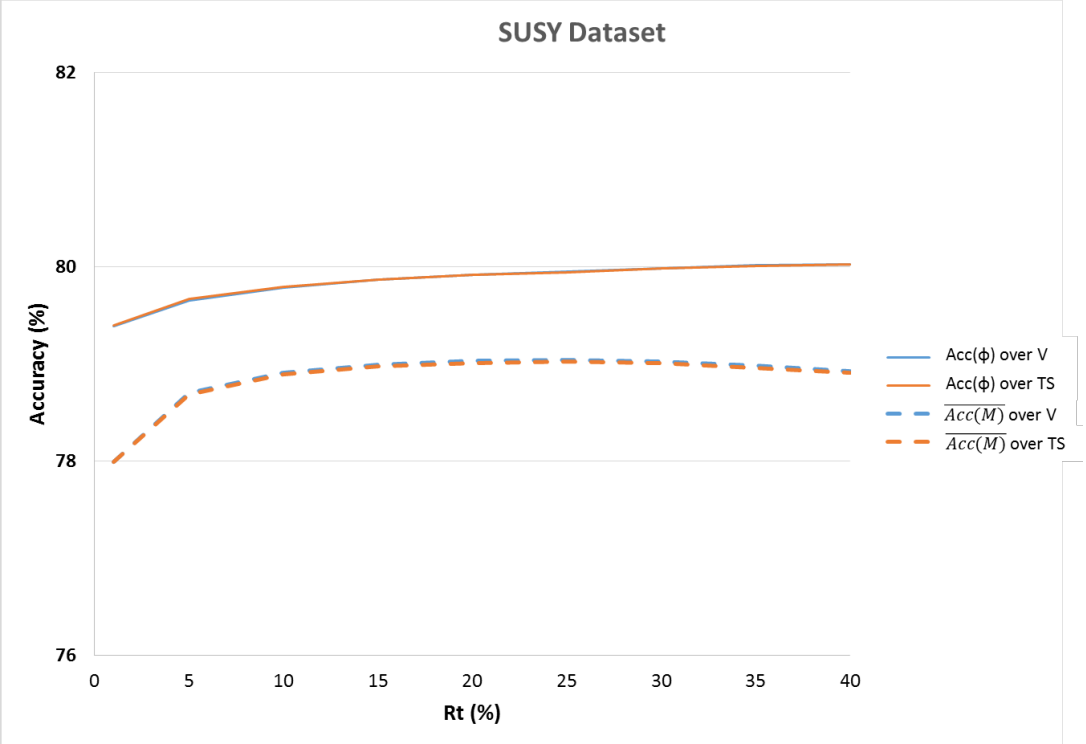


Fig. 4.15 Relationship between $Acc(\phi)$ and Rt over V and TS for the SUSY dataset

4.4.1.13 Web dataset

Figure 4.16 and Table 4.13 illustrate the relationship between $acc(\phi)$ and Rt over V and TS in the Web dataset, which exhibits a monotonic relationship. The relationship pattern is similar to those identified in most of the datasets discussed in this chapter.

Table 4.12 $Acc(\phi)$ and $\overline{acc(M)}$ over V and TS for the SUSY dataset

	validation		Testing	
Rt	$Acc(\phi)$	$\overline{acc(M)}$	$Acc(\phi)$	$\overline{acc(M)}$
1	79.39	78.00	79.40	77.99
5	79.65	78.70	79.67	78.69
10	79.79	78.91	79.79	78.89
15	79.87	79.00	79.87	78.98
20	79.92	79.03	79.91	79.01
25	79.95	79.04	79.95	79.02
30	79.99	79.03	79.98	79.01
35	80.02	78.98	80.01	78.96
40	80.02	78.92	80.02	78.91

The influence of Rt on $acc(\phi)$ forms two distinct sections on the curve depicted in Figure 4.16. These sections are listed below.

- **When Rt $\leq$ to 15%**: $acc(\phi)$ sharply increases as Rt increases, where $IR_{(1\%,15\%)} = 0.83\%$. In this section, $acc(\phi)$ and Rt are very strongly correlated, where $r = 0.954; p = 0.046$.

- **When Rt $>$ 15%**: $acc(\phi)$ slightly increases, where $IR_{(20\%,100\%)} = 0.02\%$. A very strong correlation is also found between $acc(\phi)$ and Rt, where $r = 0.950; p < 0.0005$.

To statistically verify the relationship over the entire range of Rt values, Spearman's correlation is calculated. The computation indicates a very strong correlation between $acc(\phi)$ and the entire Rt range, where $r_s = 1$ over TS with $significant\, p < 0.0005$.

4.4.2 Effects of Rt on $acc(\phi)$ over all the examined datasets

This section presents an overall view of the relationship between $acc(\phi)$ and Rt. This perspective is determined on the basis of the investigation presented in the preceding section.

Figure 4.1 presents three hypothesized patterns for the relationship between $acc(\phi)$ and Rt. The analysis of the effects of Rt on $acc(\phi)$ in the 13 datasets reveals a need

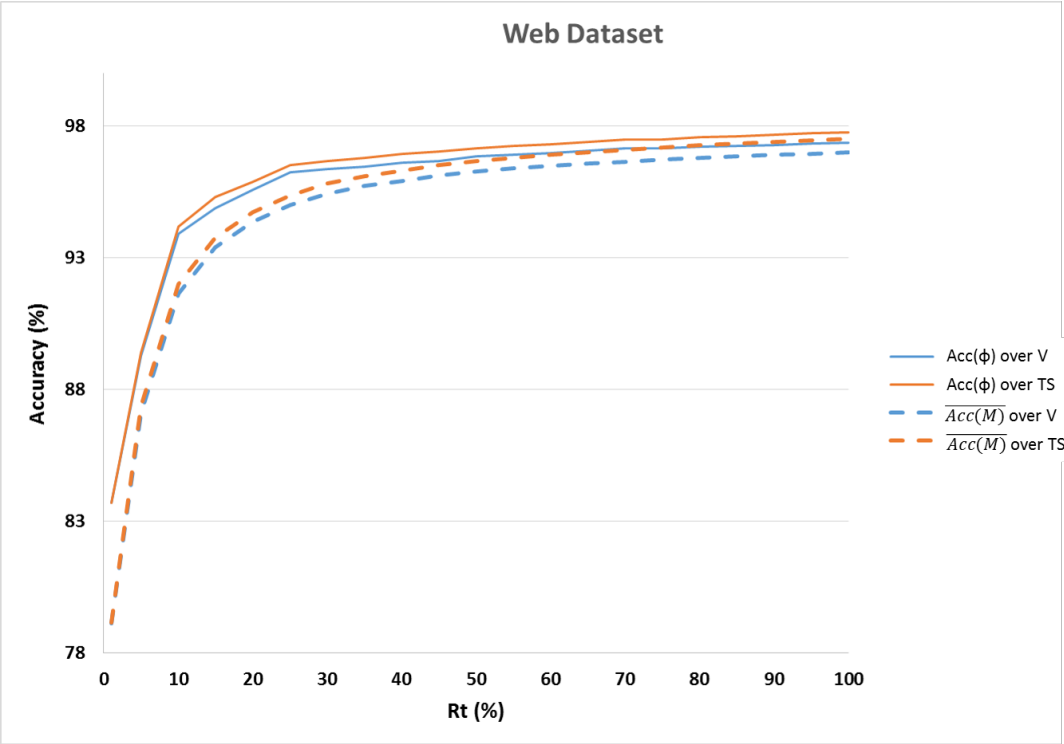


Fig. 4.16 Relationship between $Acc(\phi)$ and Rt over V and TS for the Web dataset

Table 4.13 $Acc(\phi)$ and $\overline{acc(M)}$ over V and TS for the Web dataset

	validation		Testing	
Rt	$Acc(\phi)$	$\overline{acc(M)}$	$Acc(\phi)$	$\overline{acc(M)}$
1	83.71	79.13	83.71	79.15
5	89.27	87.09	89.42	87.37
10	93.93	91.65	94.20	92.02
15	94.90	93.40	95.30	93.78
20	95.60	94.38	95.91	94.76
25	96.25	95.01	96.54	95.39
30	96.39	95.45	96.69	95.83
35	96.48	95.73	96.79	96.11
40	96.62	95.94	96.95	96.33
45	96.67	96.14	97.03	96.53
50	96.86	96.28	97.18	96.69

	validation		Testing	
Rt	$Acc(\phi)$	$\overline{acc(M)}$	$Acc(\phi)$	$\overline{acc(M)}$
55	96.92	96.40	97.25	96.79
60	96.99	96.49	97.33	96.92
65	97.06	96.58	97.41	97.02
70	97.15	96.66	97.50	97.12
75	97.16	96.74	97.51	97.20
80	97.22	96.81	97.58	97.29
85	97.26	96.87	97.61	97.36
90	97.29	96.91	97.67	97.41
95	97.34	96.96	97.74	97.47
100	97.38	97.01	97.78	97.53

to incorporate a new pattern into the figure. The new pattern represents any datasets wherein $acc(\phi)$ increases with rising Rt up to a certain point, after which $acc(\phi)$ starts to decline. Figure 4.17 is the refined image of the patterns exhibited by the $acc(\phi)$–Rt relationship.

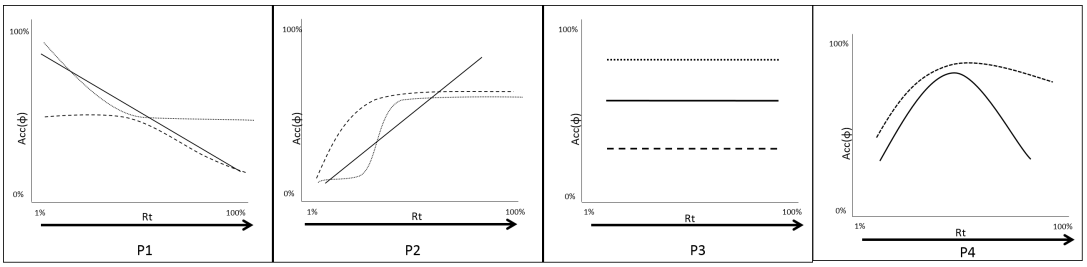


Fig. 4.17 Possible patterns of the relation between Rt and $acc(\phi)$

The 13 examined datasets fall under three out of the four categories displayed in Figure 4.17. The first category is the P2 pattern, which covers all the datasets where $acc(\phi)$ increases with growing Rt, regardless of whether the increase is continuous or stops at a certain Rt and stabilises. The second category is the P3 pattern, which consists of all the datasets where $acc(\phi)$ shows no Rt-induced effects or an increase in Rt negligibly contributes to $acc(\phi)$. In this study, if the absolute difference between $acc(\phi)$ at Rt=1% and $acc(\phi)$ at Rt_{max} is less than 0.5%, then the improvement in accuracy is considered negligible or insignificant. The third category is pattern P4, which includes all the datasets where $acc(\phi)$ improves up to a particular Rt and then decrease. Table 4.14 lists the identified relationship patterns and the datasets belonging under each pattern.

Drawing an adequate visual representation of the patterns identified in the 13 datasets and linking these to the patterns defined in Figure 4.17 necessitate a reproduction of all the charts presented in the preceding sections. The reproduction can be accomplished by using the same scales for the x-axis (Rt) and y-axis (accuracy). Figures 4.18 to 4.21 show the 13 datasets grouped under the three pattern categories.

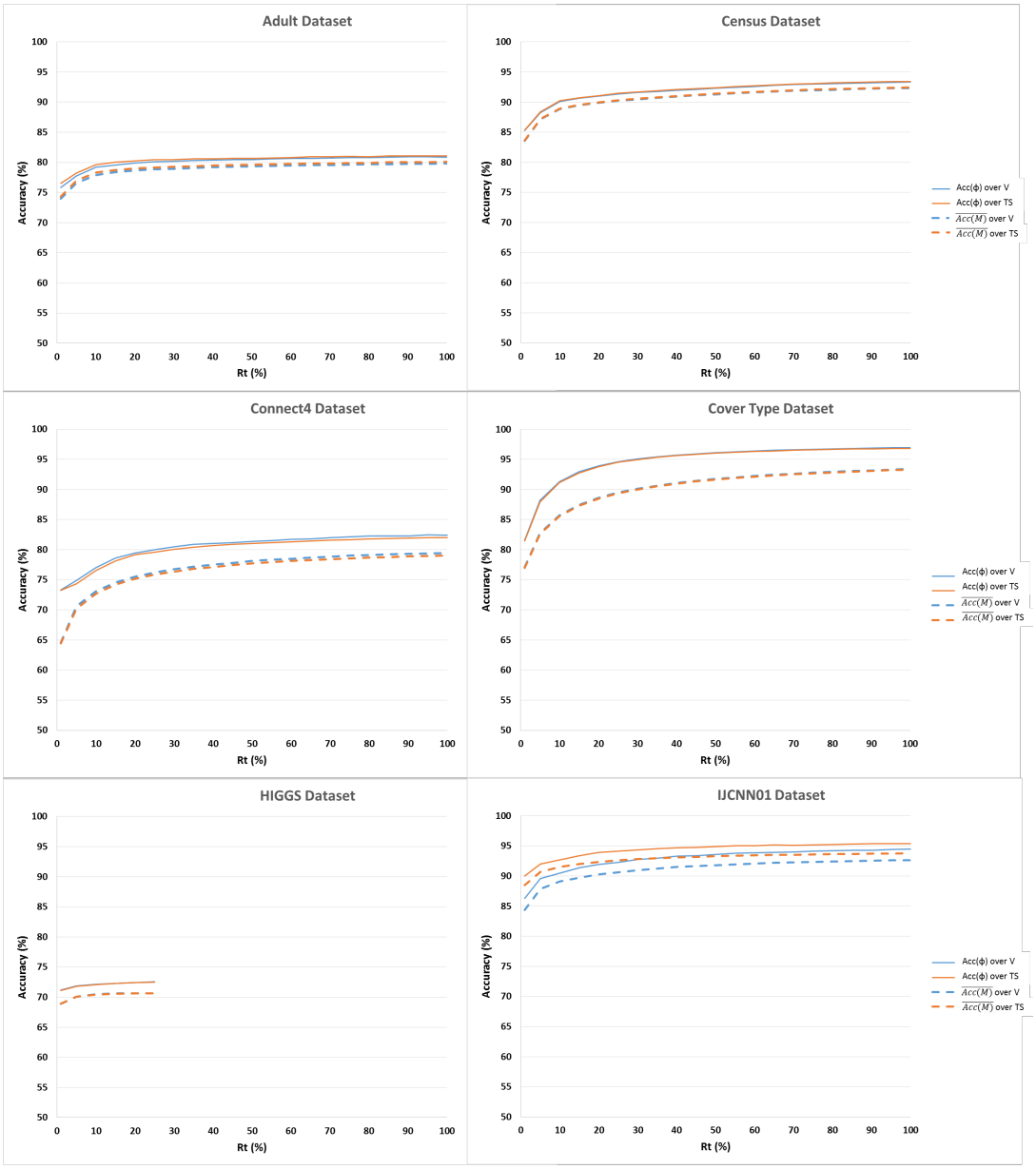


Fig. 4.18 Datasets that exhibit the P2 pattern

Table 4.14 Categorisation of the examined datasets on the basis of the relationship between Rt and $Acc(\phi)$

Relationship Patterns		
P2	P3	P4
Adult	FARS	Poker
Census	KDD99	
Connect4	Shuttle	
Cover Type		
HIGGS		
IJCNN01		
Skin		
SUSY		
Web		

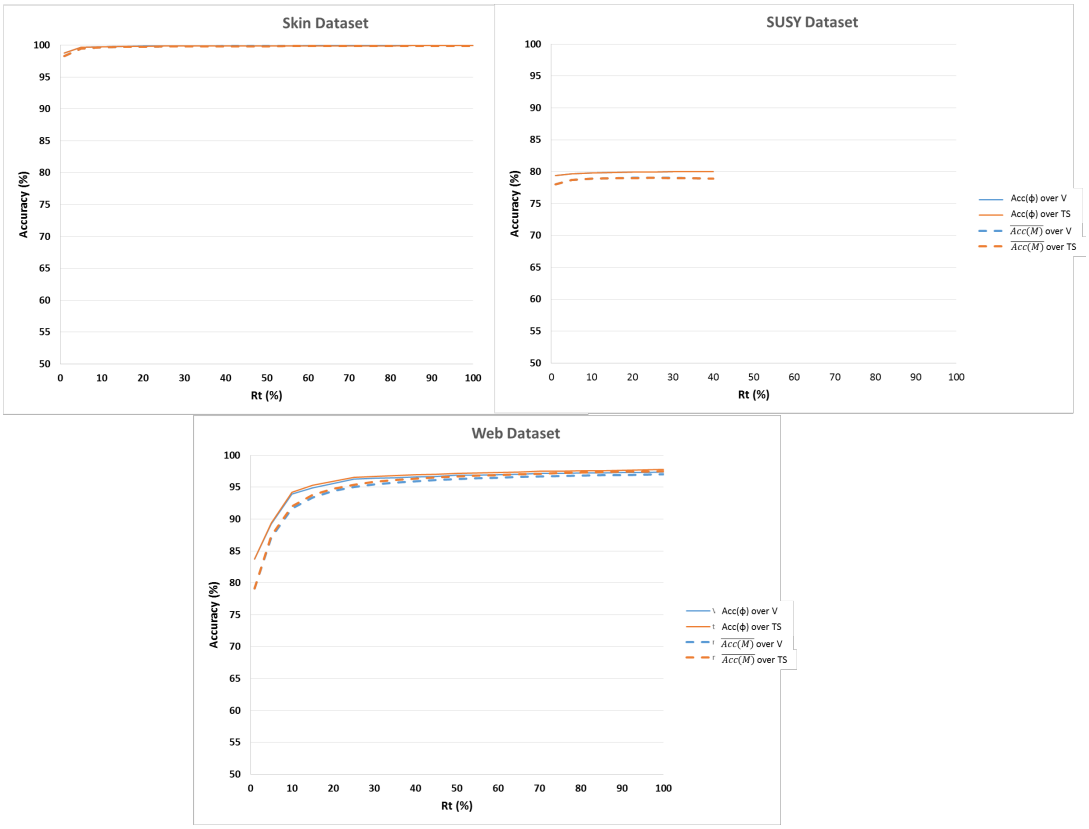


Fig. 4.19 Other datasets that belong to the P2 category

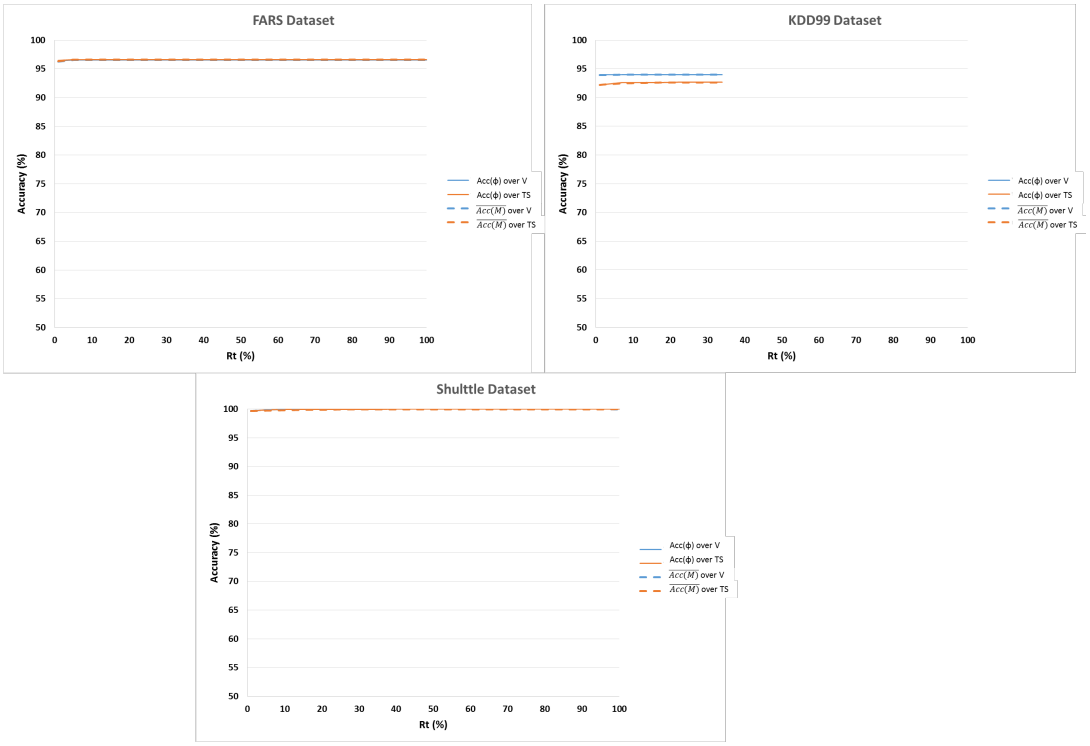


Fig. 4.20 Datasets that fall under the P3 category

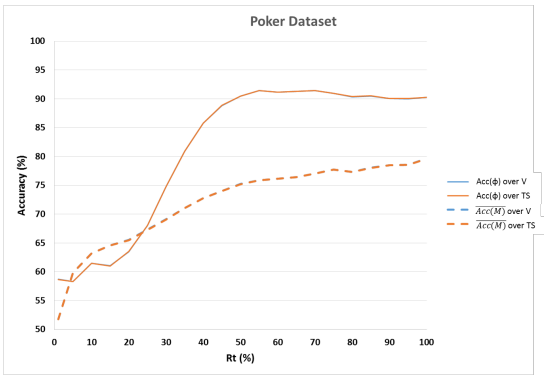


Fig. 4.21 Datasets that fall under the P4 category

4.4.3 Effects of Rt on ensemble time

The results presented in Sections 4.4.1 and 4.4.2 show that using an Rt smaller than Rt_{max} can produce an ensemble that performs comparably with the ensemble generated using Rt_{max}. In this section, we verify whether using a small Rt is more efficient than constructing an ensemble by using Rt_{max}. As expected, increasing Rt adds to the time required to construct an ensemble of classifiers (Figure 4.22). The same relationship between Rt and ensemble time is identified for all the 13 datasets. This section provides the chart for the Adult dataset, and the rest of the charts can be found in Appendix A.

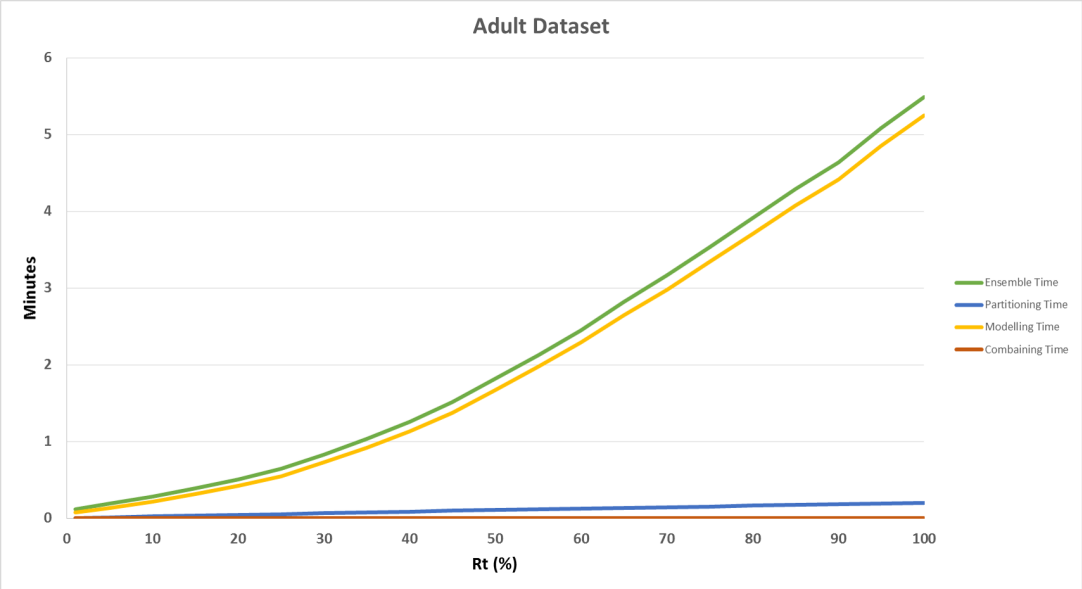


Fig. 4.22 Relationship between Rt and ensemble time for the Adult dataset

4.5 Summary

The empirical results in this chapter confirm that Rt influences $acc(\phi)$, but this influence differs from one dataset to another. Our results also negate the widely accepted idea that having more training data instances generally improves classification performance. This case does not always hold for ensembles of classifiers.

In sum, the identified relationship between $acc(\phi)$ and Rt in the 13 datasets exhibits three different patterns (Table 4.14). Fundamentally, nine datasets exhibit an increasing monotonic relationship between $acc(\phi)$ and Rt to a certain Rt. This association categorises the datasets under P2. In three datasets, $acc(\phi)$ is unaffected by the growth in Rt, a pattern that is defined as P3. Pattern P4, which is initially unexpected before the experiment, is reflected only by the Poker dataset. In this pattern, $acc(\phi)$ continuously improves with increasing Rt until it peaks at a certain Rt, after which it declines even with further Rt growth. Further investigation is required in the future to understand the reasons behind the variation in the relation between $acc(\phi)$ and Rt for different datasets.

Depending on our investigation, when dealing with large datasets, partitioning a single dataset by maximizing data subset size is not the ideal solution because this approach does not necessarily enhance ensemble performance. By contrast, using a suitable data subset size (smaller than Rt_{max}) can produce an efficient ensemble with performance comparable to that gained under the maximum possible size of data subsets.

The succeeding chapter introduces an algorithm that identifies the patterns of the relationship between Rt and $acc(\phi)$ and determines the best Rt value for use in dividing a single large dataset.

CHAPTER 5

IDENTIFYING THE RELATIONSHIP BETWEEN RELATIVE SUBSET SIZE AND ENSEMBLE ACCURACY

5.1 Introduction

Most existing machine learning algorithms were developed under the assumption that all of the available data of a problem under study would be loaded into the main memory. However, this assumption is unrealistic in the case of big data, which is too big to be loaded into the main memory and hence requires the use of a divide-and-conquer strategy. In the context of big data mining, such a technique entails dividing a single big dataset into manageable subsets in order to overcome memory limitation.

In terms of classification problems, an ensemble of classifiers represents one of the best methods of utilising the divide-and-conquer technique to learn from big data. People generally believe that building an ensemble from a large data subset results in more accurate classification, but this is not always the case, according to some researchers [65] [24] [66]. Two important questions arise when the ensemble method is used for classifying a big dataset. First, can an ensemble of classifiers constructed from small data subsets performs as well as or similar to an ensemble constructed from data subsets that are each equal in size to the whole available training dataset? Second,

what is the possible relationship between the size of data subsets and the accuracy of an ensemble?

Therefore, in this chapter, an algorithm is proposed to identify the learning pattern of the relations between the accuracy of an ensemble of classifiers ($acc(\phi)$) and the relative size of the data subsets (Rt) used to train the base classifiers. The aim of the proposed algorithm is to facilitate a better understanding of the data in terms of learning behaviour and decide when it is appropriate to use as much data as possible in building an ensemble and when it is not. This algorithm will also be helpful in deciding the best data subset size of a given dataset to use when an ensemble is being constructed.

The rest of this chapter is organised as follows: Section 5.2 describes the proposed algorithm to identify the relation pattern between $acc(\phi)$ and (Rt). Section 5.3 illustrates the algorithm's design and the pseudocode, while Section 5.4 describes our experimental design and setup. Section 5.5 presents the results, and the last section summarises the study and outlines the conclusions.

5.2 A Novel Algorithm for Identifying the Relation Between Relative Subset Size and Ensemble Accuracy

In this chapter, learning patterns are represented by the relation between the ensemble accuracy $acc(\phi)$ and the relative data subset size (Rt), and they should roughly fall into one of the three patterns illustrated in Figure 4.2 [24]. These patterns can be described as follows: In the first pattern (P1), $acc(\phi)$ has an inverse correlation with Rt; in the second pattern (P2), $acc(\phi)$ has some degree of correlation with Rt; and in the last pattern (P3), $acc(\phi)$ does not show any correlation with Rt.

In machine learning, it is generally assumed in the case of a single base classifier and an ensemble of classifiers that increasing the size of the training dataset improves overall classifier accuracy. However, in practice, this assumption does not always hold,

and different behaviour and relations may occur. As previously stated, the aim of the proposed algorithm is to detect the relation pattern between $acc(\phi)$ and (Rt) in a few search steps and thus, to understand how an ensemble of classifiers may behave as the size of the data subsets varies .

Figure 5.1 shows the stages of the proposed algorithm. The proposed approach is an iterative method consisting of five stages, each of which is repeated multiple times until the termination criteria are met, as follows:

Stage 1: Choose a Rt: In the first iteration, the algorithm will begin with a predefined initial Rt (Rt_0). In consecutive iterations, the value of the Rt is identified by some adaptive rules, as described in Section 5.3.5.

Stage 2: Divide training dataset: In this stage, the training dataset (Tr) is divided into N subsets, where the size of each subset is equal to the Rt set in the previous stage. The method used to partition the Tr is illustrated in Section 5.4.

Stage 3: Build an ensemble: In this stage, an ensemble of classifiers (ϕ) is constructed through the generation of a model from each data subset using a machine learning algorithm.

Stage 4: Evaluate the ensemble: In this stage, the ensemble is eventuated over V.

Stage 5: Check the termination criteria: Following the ensemble evaluation in the previous stage, the termination criteria are checked to decide whether the process should be terminated or whether the Rt should be varied and a new iteration started.

It should be noted that several factors must be considered to achieve the aim of the proposed algorithm. These factors can be classified into two categories according to their effect. The first category involves ensemble-related factors, such as the number of subsets (N), the number of base classifiers (M), the partitioning method and the fusion strategy, all of which have a direct effect on the ensemble's accuracy. The

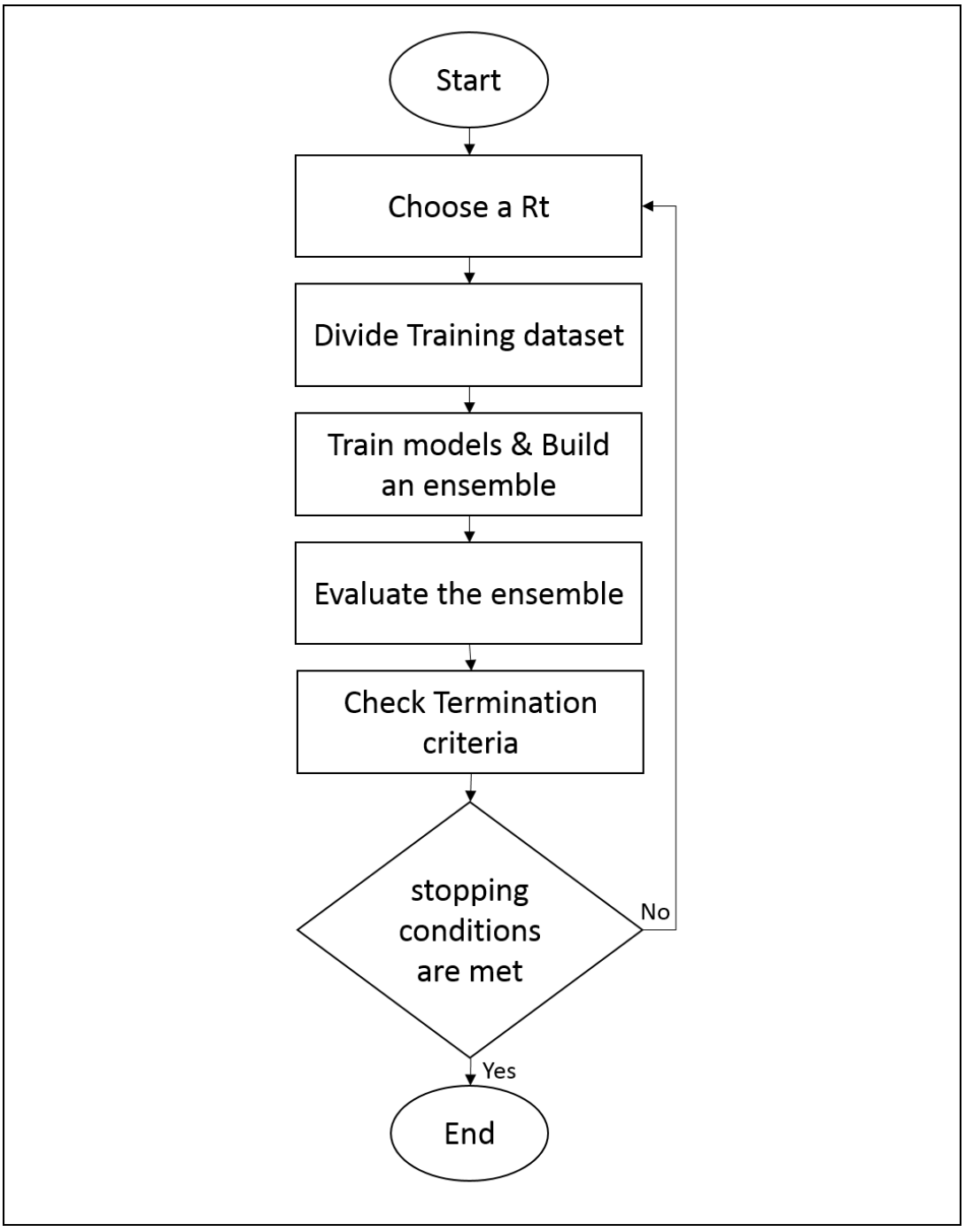


Fig. 5.1 Main stages of the proposed Algorithm for detecting the learning pattern

second category includes algorithm-related factors, such as step size, initial step size and memory limit, which have a direct influence on the performance of the detection algorithm. Most of the factors that relate to the former category have been studied in literature on ensemble methods [99] [112] [94], and for that reason, they will not be discussed in detail as part of this study. Since our focus in this chapter is on the proposed algorithm, the algorithm-related factors, which also represent the inputs of the algorithm, will be discussed in detail here.

5.3 Design of the Proposed Method

In order to achieve the goal of the proposed algorithm, there is a need to observe the $acc(\phi)$ with different Rt values while fixing the number of base classifiers (M). Fixing M in all the iterations during the algorithm's lifetime is a very important factor in eliminating any influence of M on the $acc(\phi)$.

Fundamentally, the proposed method is an iterative approach that starts with the construction of a homogeneous ensemble from data subsets with a size equal to Rt_0. It should be pointed out that different types of base classifier can be used to generate methodologically different models in order to build a heterogeneous ensemble. However, this study only uses one type of base learner for simplicity. Subsequently, the performance of the ensemble on the validation dataset (V) is then taken as a representation of the relationship between $acc(\phi_{Rt_i})$ and Rt_i, where (i) is the number of the iteration . This procedure will be repeated through variations of Rt values until no more improvement in the ensemble performance is observed or until one of the termination conditions is met.

The proposed algorithm is implemented through an ensemble of classifiers, and the ensemble's framework is described briefly alongside the experimental procedure in Section 5.4.2. The detailed algorithm design is discussed in the sections below.

5.3.1 Inputs Parameters

1. Memory safe limit (MSL): The relation between the data subset size and the memory required to build an ensemble of classifiers is not a linear one. Therefore, predicting the amount of memory required to build an ensemble for a specific Rt is extremely difficult as it involves many other uncertain factors such as OS, Kernel, other factors, etc. MSL is used as precaution to prevent the algorithm from crashing due to physical memory constraints. The MSL is used to calculate the maximum amount of memory (Mem_{max}) that can be used by the proposed algorithm. In iteration, if the consumed memory for an ensemble exceeds this limit, the algorithm will be terminated in the following way

$$Mem_{max} = AM - MSL \tag{5.1}$$

   where AM is the size of the available memory.

2. Lower relative subset size Rt_{low}: This parameter is the smallest relative size of the data subset, and it represents the lower boundary for the Rt value, where

$$Rt_{low} = \left\lceil \frac{100\%}{M} \right\rceil \tag{5.2}$$

3. Initial step size (α): This parameter will be used to vary the value of Rt from one iteration to another.

4. Tolerance(θ): This parameter is the greatest range of variation allowed when the accuracy of two ensembles is compared. Therefore, if $acc(\phi_1)$ - $acc(\phi_2) \leq \theta$, then $acc(\phi_1) \approx acc(\phi_2)$.

5. β: This parameter is the number of successive Rt points that has an equal acc(ϕ) and needs to be identified before the algorithm is terminated.

5.3.2 Outputs

As previously stated, in order to achieve the goal of the proposed algorithm, there is a need to observe the $acc(\phi)$ with different Rt values. The output of this algorithm is a set of points which represent the accuracy of the ensemble of classifiers at various Rt(s). These Rt values are selected by the algorithm, as described in the following section (section 5.3.3). The output points can be used to draw a scatter plot which represents the relation between $acc(\phi)$ and Rt, as shown in section 5.5.

5.3.3 Incrementing the Data Subset Size

As there is a need to observe the $acc(\phi)$ with different Rt values, a variable step size is used to increment the Rt in each iteration. The process of incrementing the Rt depends on the growth ratio (g) of the Acc($acc(\phi)$) between two ensembles in three consecutive iterations. If g is found to be approximately constant between three consecutive iterations, then the step size used to calculate the next Rt will be multiplied by 2 ($\alpha = \alpha \times 2$). If g is found to be approximately constant again, then the step size will be multiplied by 3 and so on until one of the termination criteria is met. This incrementing process is illustrated in algorithm 2 lines 17 to 27.

The growth ratio (g) of the Acc($acc(\phi)$) between two ensembles g can be calculated as described below in equation 5.3:

$$g(\phi_{Rt\,i}, \phi_{Rt\,i-1}) = \left| \frac{\Delta acc(\phi_{Rt\,i}, \phi_{Rt\,i-1})}{Rt_i - Rt_{i-1}} \right| \tag{5.3}$$

where:

- $g(\phi_{Rt\,i}, \phi_{Rt\,i-1})$: is the accuracy growth ration between the two ensembles $\phi_{Rt\,i}$ and $\phi_{Rt\,i-1}$.
- $\phi_{Rt\,i}$: An ensemble of classifiers where the size of the data subsets are equal to Rt_i.

- $\Delta acc(\phi_{Rt\,i}, \phi_{Rt\,i-1})$: the difference between the accuracy of the two ensemble $\phi_{Rt\,i}$ and $\phi_{Rt\,i-1}$.

5.3.4 Termination Criteria

1. The memory consumption of an ensemble reaches the specified Mem_{max}.
2. The proposed method finds β successive Rt that have similar accuracy.
3. If the Rt_i for the next iteration is greater than 100%.

Checking the termination Criterion process is illustrated in Algorithm 2 line 9.

5.3.5 Pseudocode:

The proposed method is described in detail in Algorithm 2.

5.4 Experimental Setup

5.4.1 Datasets

Thirteen datasets were used in the experiment to test and evaluate the proposed algorithm. These datasets were obtained from the UCI Machine Learning Repository [54], the KEEL Dataset Repository [1] and the LIBSVM Dataset Repository [15]. Table 3.2 lists the main characteristics of the datasets used. Detailed information about these datasets is presented in Section 3.3.

5.4.2 Experimental procedure

1. Rt_{low} is selected depending on the size of the Tr, and the required number of models (M) can be calculated as described in 5.3.1.

Algorithm 2: Algorithm for Identifying the Relation Patterns between Rt and $Acc(\phi)$

Inputs :

1. N: a set that contains references to all available subsets.
2. TS: testing dataset.
3. V: validation dataset.
4. MSL: Memory Safe Limit.
5. Rt_{low}: lower relative subset size.
6. α: initial step size.
7. θ: tolerance.
8. β: number of successive Rt points that needs to be identified before the algorithm is terminated.

Output :

A prediction of the pattern of the relationship between $acc(\phi)$ and Rt.

Start :

```
Declare counters i = 0 and k = 0
set step_Increment= j =1
Let step = α
Let M = ⌈100% / Rt_low⌉
if M is even number then
    M = M + 1
end
Let N = M
Let Rt_i = Rt_low
Calculate the available memory(AM)
Mem_max = AM − MSL
Divide Tr into N data subsets using Rt_i
Acc(φ_Rt i) and mem(φ_Rt i) ← Build and evaluate φ(Rt_i)
while ( mem(φ_Rt i) < Mem_max and k ≤ β and Rt_i ≤ 100%) do
    i++
    Rt_i = Rt_{i−1} + step
    if (Rt_i > 100) then
        Rt_i = 100
    end
    Divide Tr into N data subsets using Rt_i
    Acc(φ_Rt i) and mem(φ_Rt i) ← Build and evaluate φ(Rt_i)
    if ( mem(φ_Rt i) ≤ Mem_max) then
        Calculate g_i ← g(φ_Rt i, φ_Rt i−1)
        if (i ≥ 2) then
            if (g_i ≈ g_{i−1}) then
                j++
                Let step = α × j
                if g_i ≈ 0 then
                    k++
                end
                else
                    k=0
                end
            end
        end
    end
end
Draw the detected pattern for the relation between Acc(φ) and Rt
End
```

Table 5.1 Datasets Used after the Creation of Validation Datasets

Dataset	Training Instances	Validation Instances	Testing Instances	No. of Attributes
Adult	34,536	9,768	16,281	60
Census	245,389	59,859	99,762	31
Connect4	48,173	9,458	20,268	43
Cover Type	366,037	156,873	58,102	39
FARS	89,648	14,010	30,021	30
HIGGS	5,390,000	2,310,000	3,300,000	28
IJCNN01	60,138	14,997	91,701	7
KDD99	1,361,892	1,469,530	311,029	47
Poker	574,004	143,502	307,503	11
Shuttle	54,331	8,700	14,500	10
Skin	216,974	34,308	73,518	4
SUSY	2,800,000	700,000	1,500,000	19
Web	64,839	14,925	14,951	262

2. If M is an even number, then equation 5.4 needs to be applied to avoid a tie situation in the combination phase of the ensemble.

$$M = M + 1 \tag{5.4}$$

3. As the proposed algorithm is an iterative method, the following steps (step 3 to step 9) will be repeated in all the iterations.

4. Tr is partitioned into N datasets using the partitioning method described previously in section 4.3.3. In this case, the size of each data subset is equal to Rt_i, $N = M$, and i is the number of the iteration.

5. A base classifier is generated from each data subset. In this chapter, the C4.5 decision tree[73] is used to induce models. Specifically, the j48 learning algorithm, which is the Weka [34] implementation of the C4.5 decision tree[73], is used.

6. An ensemble of the classifiers is constructed, where the majority voting is used as a decsion fusion strategy.

7. A hold-out [47, p.9] strategy is adopted to evaluate the generated ensemble over the Validation (V) and Testing (TS) datasets. In this case, the ensemble accuracy was used as a performance measure for the created ensemble.

8. The amount of memory used during the construction and evaluation of the ensemble $mem(\phi)$ is calculated.

9. At the end of each iteration, the termination conditions are checked. If none of these conditions is met, the Rt_i is incremented as described in section 5.3.3, and the steps from 3 to 9 are repeated until one of the termination criteria happens.

5.4.3 Evaluation Method

In order to evaluate our detection algorithm, the predicted relation pattern is compared to the true pattern that represents the true relation between $acc(\phi)$ and Rt. The predicted curve results from evaluating the ensemble at each Rt selected by the algorithm on the V. Conversely, the true curve is formed by evaluating the ensemble with a wide range of Rt values on the testing dataset. This range of Rt values begins with Rt_{low} and increases to Rt_{max}, with a step size of 5%. $Rt_{max} = 100\%$ for most of the datasets examined; the only exceptions were in the KDD, SUSY and HIGGS datasets due to memory constraints, where the Rt_{max} values were 34%, 40% and 25%, respectively.

The aim in the evaluation process is to prove the ability of the proposed method to terminate the algorithm where no more improvement is expected to occur in the $acc(\phi)$. To achieve this goal, the ensemble accuracy of the last Rt in the predicated curve ($acc(\phi_{Rt_{stop}})$) is compared to the ensemble accuracy of the maximum possible Rt on the true curve ($acc(\phi_{Rt_{max}})$). In addition, this comparison is statistically tested to find out whether or not there is a significant difference between ($acc(\phi_{Rt_{stop}})$) and ($acc(\phi_{Rt_{max}})$).

The Wilcoxon's Signed-Rank test [102] for matched pairs is used to compare between $acc(\phi_{Rt_{stop}})$ and $acc(\phi_{Rt_{max}})$ over TS in 10 runs, to find out whether there is significant difference between the performances of $\phi_{Rt_{stop}}$ and $\phi_{Rt_{max}}$ using significant level =0.05. This test is a on-parametric test which is used as alternative of the paired t-test, because it was difficult to fulfilled the normality condition of $acc(\phi)$ over the 13 examined datasets. This test is used as described in [39][P. 233-238] and the SPSS software is used to preform the test.

5.5 Experimental Results and Evaluation

Extensive experiments were conducted on the 13 datasets to evaluate the ability of the proposed algorithm in order to identify the relation patterns between $acc(\phi)$ and Rt. The proposed algorithm requires multiple input parameters. Therefore, our investigation started with an exploratory experiment to understand the effect of these input parameters on the algorithm and to identify the best values for these parameters in the datasets examined. Then, the results of the proposed algorithm was statistically analysed.

5.5.1 Effect of the Input Parameters on the Algorithm

In Section 5.3.1, five different input parameters were mentioned. The values of MSL and Rt_{low} depend on the available physical memory, whereas the remaining three parameters (α,θ and β) can be tuned independently from the memory constraints. To identify the effect of the input parameters on the proposed algorithm, MSL and Rt_{low} were set according to the amount of available memory, which was 20 GB, while the values for α,θ and β were varied and then analysed as follows:

1. The MSL was set to 500 MB; thus, from equation 5.1, the $Mem_{max} = 19980$ MB. .

2. Rt_{low}: Our lower relative subset size will be 1%. From equation 5.2, the number of required models (M) can be calculated, and it will be equal to 100. As specified in Section 5.4.2, majority voting is used as a fusion strategy, so equation 5.4 should be used to correct the value of M and avoid a tie situation. Using equation 5.4, M = 101.

3. α: Varied from 1% up to 6%.

4. Tolerance (θ): Varied from 0.5% up to 2.5%.

5. β: It will be varied from 2 up to 5.

Our extensive experiments indicate the tolerance needed to be less than or equal to 0.5. Increasing the tolerance value too much might lead the algorithm to be terminated before the relation pattern can be adequately detected. The best illustration of this situation is the Poker dataset. Figure 5.2 depicts the predicted curve for the relation between Acc($acc(\phi)$) and Rt when $\theta = 2.5$.

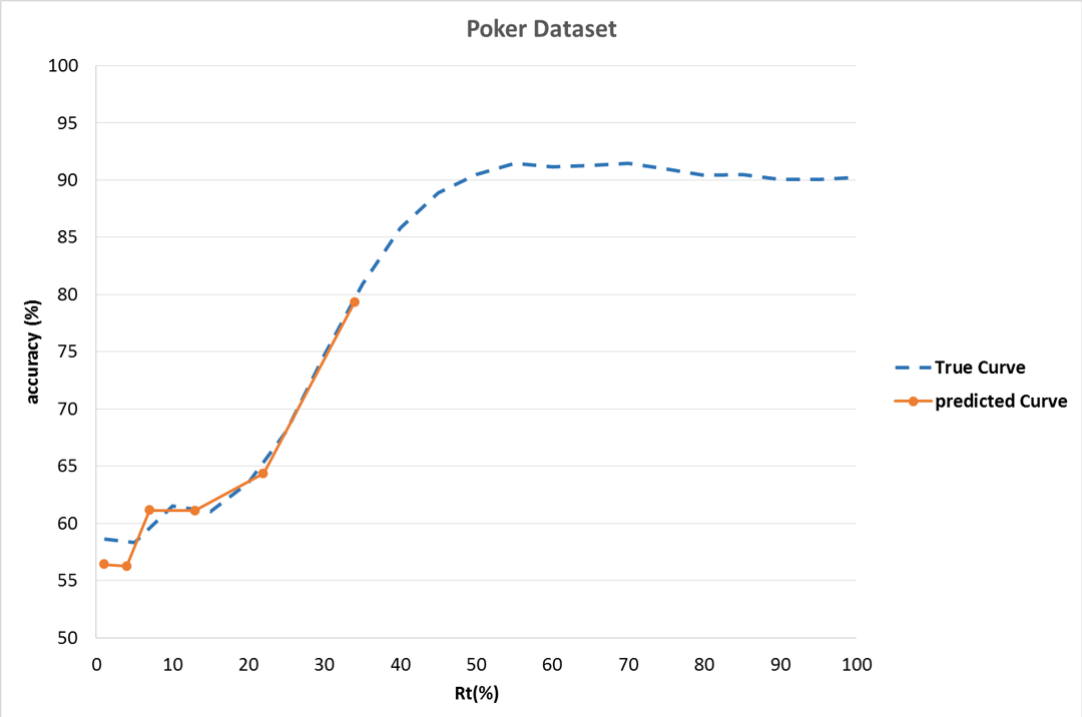


Fig. 5.2 The pattern identified by the proposed algorithm on the Poker dataset where $\theta = 2.5$, $\alpha = 3$ and $\beta = 3$

By contrast, decreasing the tolerance value to a very small value will lead to an unnecessary increase in the number of steps required to detect the relation pattern. The Connect4 dataset is one of the best datasets to show this situation. Figure 5.3 shows that when θ is decreased to 0.05, the algorithm needs 25 steps to detect the relation pattern, whereas it needs only 6 steps when $\theta = 0.5$.

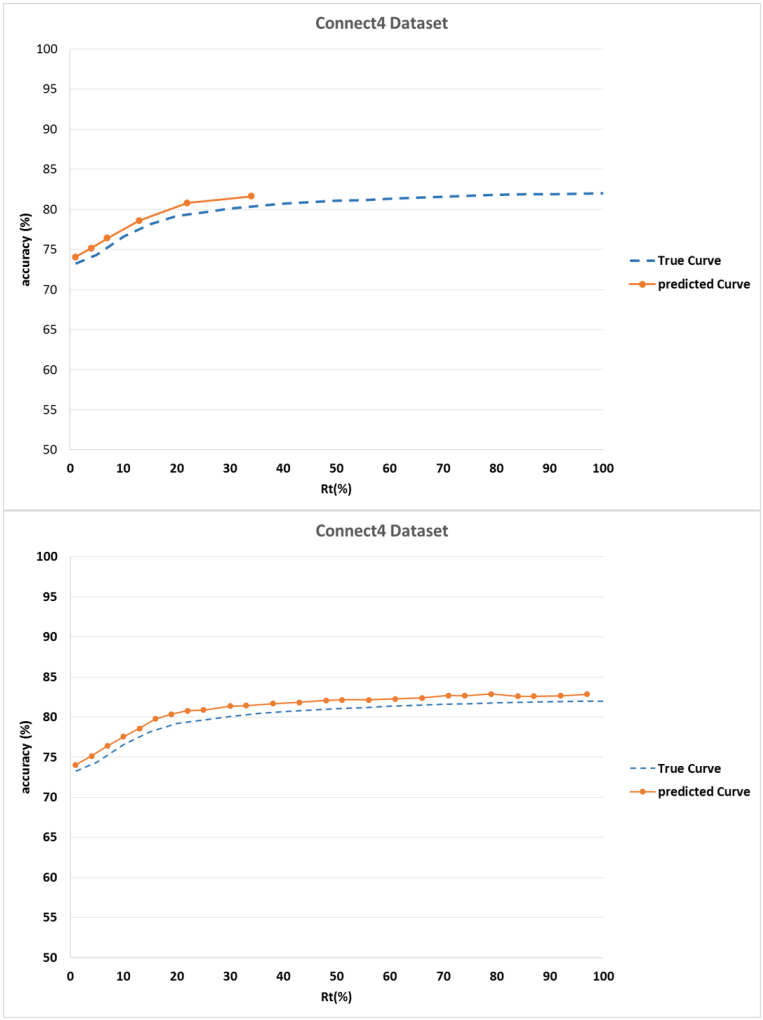


Fig. 5.3 The pattern identified by the proposed algorithm on the Connect4 dataset where $\theta = 0.5$ on the left chart and $\theta = 0.05$ on the right chart. α and β were equal to 3 in both charts

With regard to the α and β, our extensive experiments demonstrate that the suitable α and values β were 3 in all the examined datasets.

5.5.2 Results

Reporting all the results of the 13 datasets with variations in the input parameters is a huge task. Thus the representative results of our extensive experiment will be reported and discussed in this section.

Figures 5.4 and 5.5 illustrate the detected pattern of the relation between $acc(\phi)$ and (Rt), which was generated by the proposed method, compared with the true relation curve, as described in Section 5.4.3. Datasets are grouped in the former figures depending on the type of patterns identified in the previous chapter 4. The 13 datasets examined belong to two groups of patterns which was identified earlier in chapter 4. eight and five datasets belong to P2 and P3, respectively. The datasets categorized as P2 are Adult, Census, Connect4, Cover type, IJCNN01, poker , Skin and Web. FARS, KDD99, shuttle, SUSY and HIGGS are categorised as P3. For any dataset belonging to the P2 category, the curve detected by the proposed method provides a hint regarding the best relative subset size (Rt) that needs to be chosen when an ensemble is being constructed from one of these datasets. For datasets that belong to the P3 category, our method gives an indication after a few steps that increasing the subset size will not improve the ensemble's accuracy.

After displaying the type of the patterns in Figures 5.4 and 5.5, it is important to evaluate the detected pattern, as discussed in section 5.4.3, to test the ability of the proposed method to stop when there is no more enhancement in the is expected to occur. Tables 5.2 and 5.3 show $acc(\phi_{Rt_{stop}})$ compared to $acc(\phi_{Rt_{max}})$ as an averaged accuracy of 10 runs over V and TS where Mem_{max} =20GB ,$Rt_{low} = 1\%$, $\alpha = 3$, $\beta = 3$ and θ =0.5.

It is clear that the difference between $acc(\phi_{Rt_{stop}})$ and $acc(\phi_{Rt_{max}})$, noted by $\Delta acc(\phi)$), was less than or equal to the predefined tolerance (θ =0.5) in 9 out of the 13 examined datasets. This indicates that the proposed method was able to stop in cases where no more improvement in the $acc(\phi)$ was expected to take place in 69 %

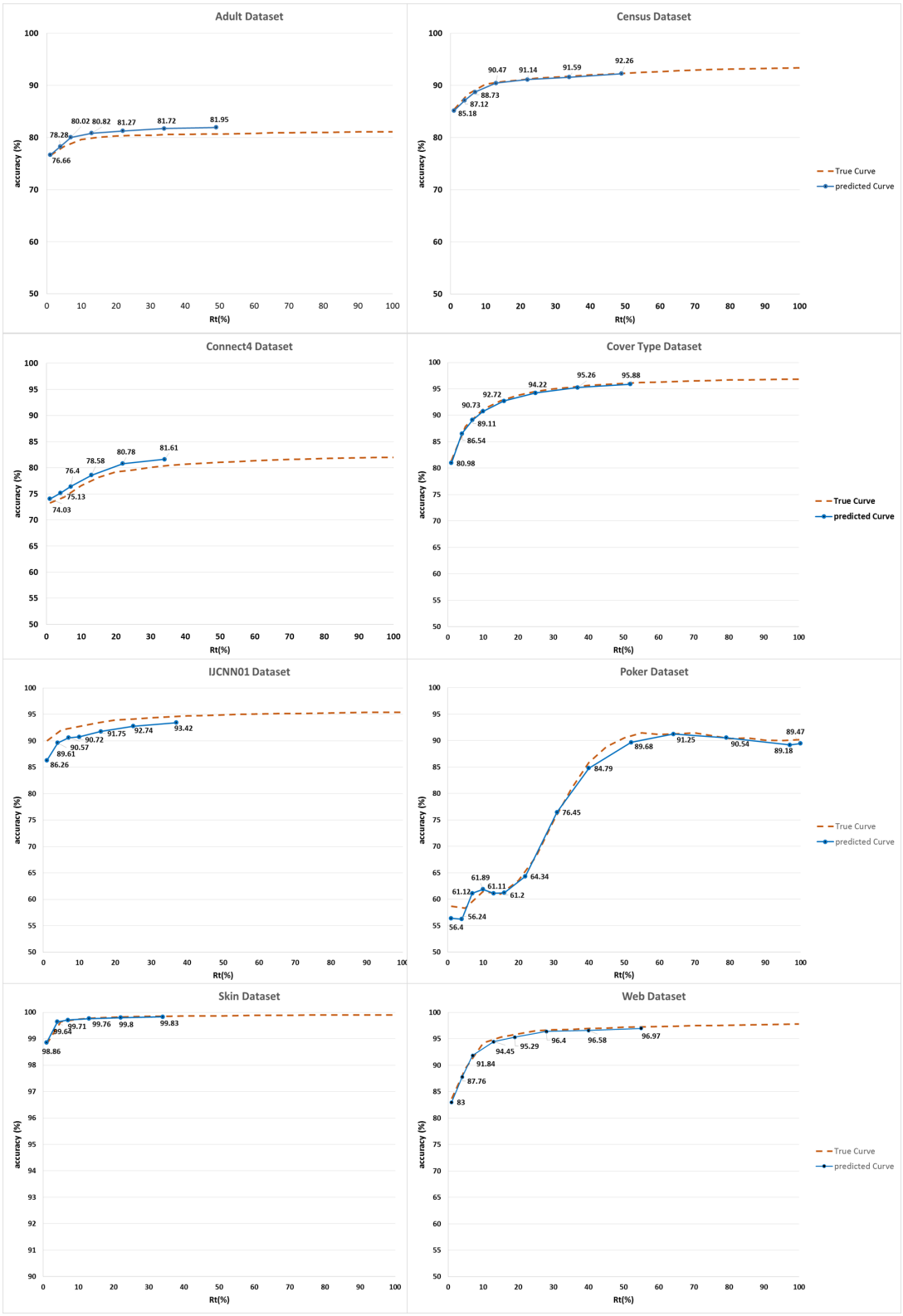


Fig. 5.4 Detected pattern of the relation between $acc(\phi)$ and (Rt) for datasets that belong to Pattern P2

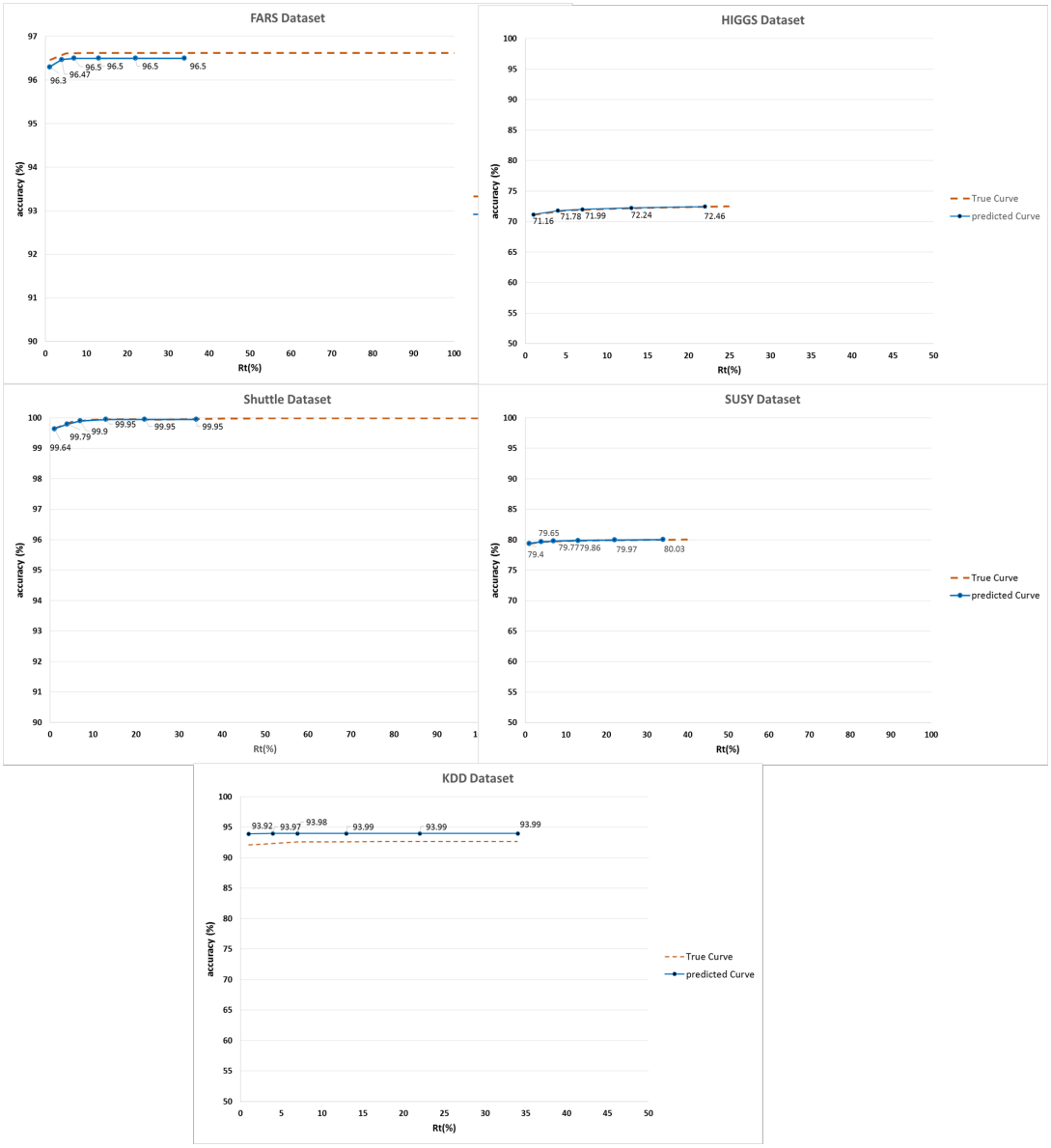


Fig. 5.5 Detected pattern of the relation between $acc(\phi)$ and (Rt) for datasets that belong to Pattern P3

of the examined datasets. On the other hand, the only exception was in the Census, Connect4, Cover Type and IJCNN01 datasets, where $\Delta acc(\phi)$ was greater than θ in both V and TS. These datasets are highlighted in Tables 5.2 and 5.3.

Table 5.2 Results of the proposed algorithm for the 13 datasets over V, where Mem_{max}=20GB ,$Rt_{low} = 1\%$, $\alpha = 3$, $\beta = 3$ and θ =0.5

Dataset	Rt_{stop}	$Rt_(max)$	$Acc(\phi_{Rt_{stop}})$	$Acc(\phi_{Rt_{max}})$	$\lvert\Delta acc(\phi)\rvert$
adult	49%	100%	80.71	81.15	0.44
Census	49%	100%	92.29	93.32	1.03
connect4	34%	100%	80.84	82.56	1.72
Covert Type	52%	100%	96.14	96.90	0.76
FARS	34%	100%	96.55	96.55	0.00
HIGGES	22%	25%	72.46	72.52	0.06
IJCNN01	37%	100%	93.17	94.57	1.40
KDD99	34%	34%	93.99	93.99	0.00
Poker	100%	100%	88.93	88.93	0.00
Shuttle	34%	100%	99.95	99.97	0.02
Skin	34%	100%	99.86	99.91	0.05
SUSY	34%	40%	80.01	80.03	0.02
Web	55%	100%	96.82	97.25	0.43

Table 5.3 Results of the proposed algorithm for the 13 datasets over the TS, where Mem_{max}=20GB ,$Rt_{low} = 1\%$, $\alpha = 3$, $\beta = 3$ and θ =0.5

Dataset	Rt_{stop}	$Rt_(max)$	$Acc(\phi_{Rt_{stop}})$	$Acc(\phi_{Rt_{max}})$	$\lvert\Delta acc(\phi)\rvert$
adult	49%	100%	80.72	81.17	0.45
Census	49%	100%	92.35	93.43	1.08
connect4	34%	100%	80.35	82.00	0.65
Covert Type	52%	100%	96.05	96.83	0.78
FARS	34%	100%	96.62	96.62	0.00
HIGGES	22%	25%	72.42	72.49	0.07
IJCNN01	37%	100%	94.55	95.34	0.79
KDD99	34%	34%	92.65	92.65	0.00
Poker	100%	100%	89.93	89.93	0.00
Shuttle	34%	100%	99.96	99.98	0.02
Skin	34%	100%	99.85	99.91	0.06
SUSY	34%	40%	80.00	80.02	0.03
Web	55%	100%	97.24	97.71	0.47

These four datasets, where the proposed method stopped whenever $acc(\phi_{Rt_{stop}})$ was greater than θ, were examined further. As a result, it was found that the proposed method is able to detect the point of no more enhancement in $acc(\phi)$, which is expected

to happen when the value of α is increased. Tables 5.4 and 5.5 show the suitable α for each of these datasets, in addition to all of the other information illustrated in Tables 5.2 and 5.3.

Table 5.4 Results of the proposed algorithm for the 4 datasets that need further examinations over the V, where Mem_{max}=20GB ,$Rt_{low} = 1\%$, $\beta = 3$ and θ =0.5, with α varied

Dataset	α	Rt_{stop}	$Rt_(max)$	$Acc(\phi_{Rt_{stop}})$	$Acc(\phi_{Rt_{max}})$	$\lvert\Delta acc(\phi)\rvert$
Census	4	73%	100%	92.96	93.32	0.36
connect4	6	67%	100%	82.07	82.56	0.49
Covert Type	6	73%	100%	96.61	96.90	0.29
IJCNN01	6	73%	100%	94.25	94.57	0.32

47

Table 5.5 Results of the proposed algorithm for the 4 datasets that need further examinations over the TS, where Mem_{max}=20GB ,$Rt_{low} = 1\%$, $\beta = 3$ and θ =0.5, α varied

Dataset	α	Rt_{stop}	$Rt_(max)$	$Acc(\phi_{Rt_{stop}})$	$Acc(\phi_{Rt_{max}})$	$\lvert\Delta acc(\phi)\rvert$
Census	4	73%	100%	93.06	93.43	0.37
connect4	6	67%	100%	81.53	82.00	0.47
Covert Type	6	73%	100%	96.52	96.83	0.31
IJCNN01	6	73%	100%	95.19	95.34	0.15

Table 5.6 illustrates that less than 10 steps were needed to determine the pattern relation in most of the examined datasets. The only case in which the number of steps exceeded 10 was in the Poker dataset, which required 14 steps. A possible explanation for this increase in I can be noted from Figure 5.4, which shows that the $acc(\phi)$ continuously improved until Rt = 70% for both datasets. In addition, the Table 5.6 shows that the proposed method in nine datasets was terminated due to the identification of three successive equally performing ensembles, while the method ended in the HUGGES, KDD99 and SUSY datasets due to $Mem_(max)$ being reached. While the proposed method was ended due to reach 100% in the Poker dataset.

Table 5.6 The number of steps (I) required to identify the relation pattern using the proposed method and the termination criteria for each of the 13 examined datasets

Dataset	I	Termination Criteria
adult	7	2
Census	9	2
connect4	6	2
Covert Type	7	2
FAR	6	2
HUGGES	5	1
IJCNN01	7	2
KDD	6	1
Poker	14	3
Shuttle	6	2
Skin	6	2
SUSY	6	1
Web	8	2

5.5.2.1 Statistical Analysis

Previously, the ability of the proposed method to stop where no more enhancement $acc(\phi)$ was expected to occur was confirmed by comparing $Acc(\phi_{Rt_{stop}})$ to $Acc(\phi_{Rt_{max}})$. In this section, the Wilcoxson test is used to determine whether there is a significant difference between $(acc(\phi_{Rt_{stop}}))$ and $(acc(\phi_{Rt_{max}}))$ when running 10 times over TS . Detailed information about the Wilcoxson test can be found in Chapter 3.

Table 5.7 shows the Rt_{stop}, Rt_{max} , $|\Delta acc(\phi)|$, z statistic, and the related p-value for each dataset. The statistical results obtained by applying the Wilcoxon signed-rank test in our experiment confirm that there was no significant difference between $(acc(\phi_{Rt_{stop}}))$ and $(acc(\phi_{Rt_{max}}))$ in 5 of the 13 datasets when the significance level is 0.05. These five datasets are the FARS, KDD99, Poker, Shuttle, and SUSY datasets, where the p-values were equal to 0.414, 1, 1, 0.360, and 0.670, respectively. The table also reveals an expected result, but one that does not often occur when using statistical tests. That is, the p-value in the cases of KDD99 and the Poker datasets was equal to 1, and $z = 0$. The reason the p-value was equal to 1 in both datasets was because Rt_{stop} was equal to $Rt_(max)$, which means the compressions were done between two identical ensembles, and that this was reflected in the statistical test results.

No significant difference was found between $acc(Rt_{Stop})$ and $acc(Rt_{max})$ in the present experiment means, that the proposed method succeeded in terminating the algorithm where no more enhancement in the $acc(\phi)$ was expected by increasing the Rt.

Table 5.7 The results of the statistical analysis

Dataset	Rt_{stop}	$Rt_(max)$	$\|\Delta acc(\phi)\|$	z	p-value
adult	49%	100%	0.45%	-2.805	0.005
Census	73%	100%	0.37%	-2.809	0.005
connect4	67%	100%	0.47%	-2.803	0.005
Covert Type	73%	100%	0.31%	-2.809	0.005
FAR	34%	100%	0.00%	-0.816	0.414
HUGGES	22%	25%	0.07%	-2.836	0.005
IJCNN01	73%	100%	0.15%	-2.81	0.005
KDD99	34%	34%	0.00%	0	1.000
Poker	100%	100%	0.00%	0	1.000
Shuttle	34%	100%	0.02%	-2.099	0.360
Skin	34%	100%	0.06%	-2.848	0.004
SUSY	34%	40%	0.03%	-0.426	0.670
Web	55%	100%	0.47%	-2.823	0.005

On the other hand, the statistical analysis results show that there was a significant difference between $(acc(\phi_{Rt_{stop}}))$ and $(acc(\phi_{Rt_{max}}))$ in 8 of the datasets at a significance level of 0.05. These results suggest that the proposed method was terminated where there was a possible improvement in the $acc(\phi)$ by increasing the Rt. Although the statistical results confirm that there was a possibility of enhancing $acc(\phi)$ by increasing the Rt, this enhancement (represented by $|\Delta acc(\phi)|$)) for all eight of the datasets was within the predefined tolerance ($\theta = 0.5\%$). Thus, this difference in the performance can be considered an insufficient enhancement when considering the predefined tolerance. In that case, the proposed method can be considered to have succeeded in stopping at the suitable Rt where no more improvement in the accuracy would occur.

5.5.3 Efficiency Investigations

Table 5.8 shows α, the total number of iterations (I), and the required time (T) taken to identify the relationship patterns for the 13 datasets used. It is apparent from this table that T varies from one dataset to another. For instance, there is a need for about 23 hours to detect the relation between $acc(\phi)$ and Rt in the HIGGES dataset, while only 1.41 minutes were required to identify the relationship patterns in the Shuttle dataset. As a result of this wide variation in T, there is a need to break down T to its basic components to facilitate an understanding of the reasons behind this variation in times from one dataset to the next .

Table 5.8 The average required times (T) of 5 runs in minutes and number of iterations (I) to predict the relation pattern using the proposed method for all the 13 datasets examined (α)

Dataset	α	I	T
adult	3	7	3.99
Census	4	9	44.76
connect4	6	6	2.23
Covert Type	6	7	161.66
FAR	3	6	3.29
HIGGES	3	5	1383.29
IJCNN01	6	7	5.67
KDD	3	6	783.41
Poker	3	14	174.92
Shuttle	3	6	1.41
Skin	3	6	4.47
SUSY	3	6	488.01
Web	3	8	100.12

Fundamentally, T is the sum of the time required to create and evaluate all the ensembles of the classifiers that were constructed during the lifetime of the proposed method . T can be calculated as shown in equations 5.5 and 5.6.

$$T = \sum_{i=1}^{I} T(\phi)_i \tag{5.5}$$

Where $T(\phi)_i$ is the ensemble time for the ith iteration which can be calculated $T(\phi)$ as shown in 5.6.

$$T(\phi) = T_p + T_{Tr} + T_{ee}. \tag{5.6}$$

Where

- T_p : Partitioning Time, which is the time required to partition Tr into N subsets using Rt_i.
- T_{Tr} : is the training (modelling) time, which is the time required to train the data subsets to create a base classifiers by using the machine learning algorithm;

T_{ee} : denotes the ensemble evaluation time, which pertains to the time required to evaluate an ensemble over the TS and V datasets;

As can be seen in equations 5.5 and 5.6, T is mainly affected by the $T(\phi)$, which can be influenced by many factors such as the size of data subsets, the partitioning method used, and the learning algorithm used to induce models. To highlight the effect of $T(\phi)$ and its basic elements on the time of the proposed method, the detailed $T(\phi)_i$ for each iteration are presented in Tables 5.9 and 5.10 for the datasets that required the longest and shortest times.

Table 5.9 The breakdown of (T) in minutes to its basic elements as an average of 5 runs in HIGGS dataset (α)

i	Rt_i	T_P	T_{Tr}	T_{ee}	$T(\phi)_i$
1	1	0.31	6.32	5.98	12.61
2	4	1.19	23.92	6.83	31.95
3	7	2.12	62.65	7.10	71.87
4	13	4.36	219.98	7.00	231.25
5	22	6.89	1021.74	6.98	1035.61
Total		14.88	1334.52	33.89	1383.29

The results obtained from both former tables show that T_{Tr} is the factor that most contributed to the $T(\phi)$ in each iteration. Thus, in the HIGGS dataset, T_{Tr} represents 96.47% of T which is represented in Table 5.9 as the total of $T(\phi)_i$. In the same way,

Table 5.10 The breakdown of (T) in minutes to its basic elements as an average of 5 runs in Shuttle dataset (α)

i	Rt_i	T_P	T_{Tr}	T_{ee}	$T(\phi)_i$
1	1	0.004	0.127	0.062	0.193
2	4	0.010	0.128	0.052	0.189
3	7	0.016	0.122	0.047	0.186
4	13	0.032	0.146	0.048	0.226
5	22	0.049	0.171	0.052	0.272
6	34	0.076	0.221	0.043	0.340
Total		0.186	0.914	0.305	1.405

T_{Tr} represents 65.05% of the T for the shuttle dataset as shown in Table 5.10. The data from both tables also shows that $T(\phi)$ had a direct relation with Rt, and this is very obvious in the HIGGS dataset, which is very big compared to the shuttle dataset.

The results discussed above provide further support to what was mentioned previously regarding the factor that influences the $T(\phi)$, and as a consequence, the T. Fundamentally, the proposed method in this chapter provides a generic approach that helps to identify the pattern of the relationship between $acc(\phi)$ and Rt without specifying a particular type of learning algorithm or partitioning strategy. Thus, if a researcher proposes an efficient method for building an ensemble of classifiers, the new ensemble approach can be adopted in our proposed method without the need for any modification in the design of the proposed method.

5.6 Summary

The results of our previous chapter 4 and the results of this study suggest that the relation between $acc(\phi)$ and Rt for a given dataset often belongs to the P2 or P3 categories.

In this chapter, a method was proposed to identify the relation pattern between the accuracy of an ensemble of classifiers, acc(ϕ), and the relative size of the data, (Rt). The algorithm was evaluated empirically using 13 big datasets. The results show that the proposed method can detect the relation pattern in less than 10 steps in 92% of

the datasets investigated. This method can also be used to provide a suggestion for choosing the best data subset size when an ensemble of classifiers that is needed to deal with a big dataset is being developed.

In regard to the efficiency of the proposed method, the results show that the time required to identify the pattern of the relationship is dependent on the time needed to construct and evaluate an ensemble of classifiers for the provided dataset using different Rt.Thus any enhancement in constructing time of the ensemble will be reflected positively in the proposed method.

Furthermore, the Wilcoxson test was to determine whether there is a significant difference between $Acc(\phi_{Rt_{stop}})$ to $Acc(\phi_{Rt_{max}})$ using 5 runs over TS. The results of this test show that there is no significant difference between $Acc(\phi_{Rt_{stop}})$ and $Acc(\phi_{Rt_{max}})$ in FARS, KDD99, Poker, Shuttle, and SUSY datasets, while there is a significant difference between $Acc(\phi_{Rt_{stop}})$ and $Acc(\phi_{Rt_{max}})$ in the remaining 8 datasets at a significance level of 0.05. Finding no significant difference between $Acc(\phi_{Rt_{stop}})$ and $Acc(\phi_{Rt_{max}})$ means that the proposed method succeeded in terminating the algorithm where no more improvement in the $acc(\phi)$ is expected. Also, the findings of the significant difference between $Acc(\phi_{Rt_{stop}})$ and $Acc(\phi_{Rt_{max}})$ can be ignored in cases where the absolute difference between them are within the predefined tolerance, which was the case for the eight datasets where a significant difference between $Acc(\phi_{Rt_{stop}})$ and $Acc(\phi_{Rt_{max}})$ was identified.

In further work, it will be interesting to reduce the Rt to a very small value, such as 0.1%, 0.01% or 0.001%, especially for datasets that belong to the P3 pattern, in order to understand how the relation between $acc(\phi)$ and Rt will be affected in the case of using a very small Rt. Currently, the input parameters for the proposed algorithm were chosen depending on the empirical results of our experiments. Further work needs to be done to design an adaptive rule for choosing the best input values for each dataset. Furthermore, the current experiments were conducted on a single machine. Accordingly, more improvements to the proposed detection algorithm are needed to

develop a new version of the algorithm that is compatible with a parallel computing environment; this will reduce the time required to identify the relationship pattern.

CHAPTER 6

SELECTIVE MODELLING FOR BUILDING EFFECTIVE ENSEMBLES

6.1 Introduction

Model selection, ensemble selection and ensemble pruning constitute the process of selecting a small set of classifiers from a large pool of all available classifiers to produce an efficient and effective ensemble [69]. This process can be performed dynamically or statically. Dynamic model selection is carried out by selecting a specific set of classifiers for each testing instance. By contrast, static model selection uses the same selected set of classifiers to predict entire testing instances.

As discussed in previous chapters, using an ensemble of classifiers to address the problem of mining a big dataset requires employing the 'divide-and-conquer' strategy, which necessitates the division of a single large dataset into numerous subsets of manageable sizes. A model is then constructed from each of these subsets, and an ensemble selection method is used to select the set of classifiers that best improves ensemble performance. Finally, the predictions of the selected models are combined using a fusion method designed to generate ensemble predictions.

Applying traditional ensemble selection methods in dealing with a big dataset is a challenging task. Obstacles appear to be that numerous subsets are required.

This requirement, in turn, necessitates the construction of a large number of models. Training these massive numbers of data subsets entails considerable expense in terms of time and resources. Traditional ensemble selection methods depend on the principle of selecting models from a pool of models that contain all possible models. The primary drawback of these approaches lies in the creation of a pool that contains all possible models; this task can result in time and resource wastage because only some of the generated models are selected to form the final ensemble.

To enable the conservation of time and resources, this chapter proposes a selective modelling (SM) method, whose aim is to minimise the time and resources during the construction of model pools. Time and resources reduction is made possible by the iterative building of the models pool and sustained construction more models until no further improvement than a pre-set tolerance in ensemble performance is observed.

6.2 A Novel Selective Modelling Algorithm for Mining Big dataset

As previously explained, the main problem encountered in most existing techniques of model selection is their reliance on an existing pool of models that consists of at least one model for each available data subset. Given the excessive time and resource costs associated with creating a pool of classifiers, especially for big datasets, the proposed SM method is developed in a way that allows the combination of creating models pool creation and model selection concurrently in each iteration.

The process of the SM method starts after partitioning a big training dataset into manageable data subsets and putting them into data subsets pool (DP). The proposed SM is an iterative method that consists of 4 main stages. These stages are depicted by Figure 6.1 and can be summarised as follows:

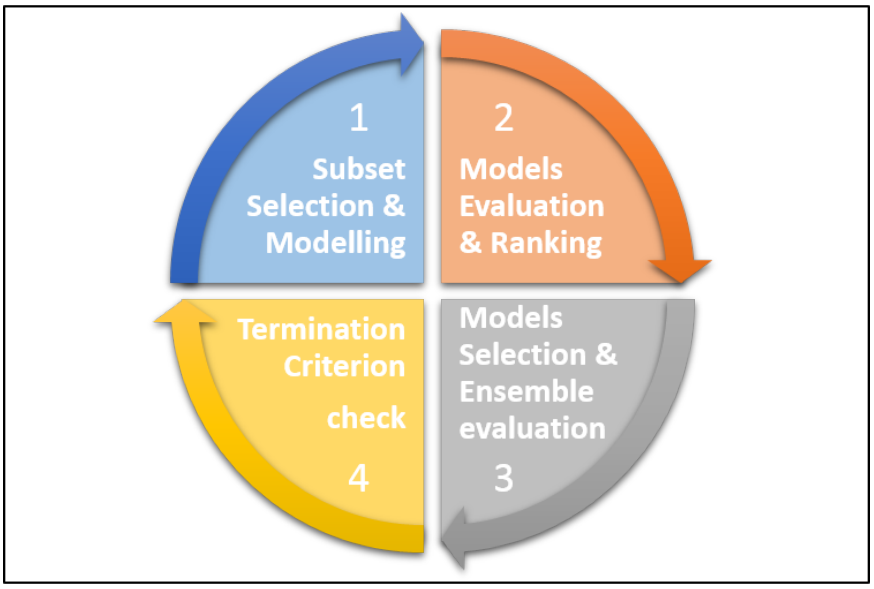


Fig. 6.1 Selective modelling process

Stage 1: **Subset selection & modelling** In this stage, several data subsets are randomly selected from the DP, and then a base classifier from each of the selected data subsets is built using a machine learning algorithm.

Stage 2: **Model evaluation & ranking** The models created in the previous stage are evaluated using a validation dataset (V), whilst weak classifiers are discarded and remaining classifiers are added to the model pool (MP). The models in the MP are then ranked using a pairwise diversity measure.

Stage 3: **Model selection & ensemble evaluation** The models that are ranked highest in the preceding stage are used to construct an intermediate ensemble. The intermediate ensemble is subsequently evaluated on V. After this step, the ensemble evaluation and selection criteria are employed to facilitate the decision of the proposed method on whether to retain the selected models as members of the ensemble or remove them.

Stage 4: **Checking termination criterion** The last stage entails checking the algorithm termination criterion to decide on whether to terminate the process or create another iteration. Generally, the algorithm will continue to work until no further

improvement in ensemble performance is observed. The final ensemble is evaluated on a testing dataset (TS).

Each of the previously described stages consists of several steps. These steps are described in more detail later in section 6.2.3.

6.2.1 Relationship between Ensemble Accuracy and Diversity

This section presents the experimental and (previous) literature-based exploration of the influence of diversity between ensemble members and ensemble accuracy Acc(ϕ). This investigation aims to determine whether diversity is to be considered as an effective selection criterion in the proposed approach. Fundamentally, generating a set of diverse models is an important element in creating an accurate ensemble of classifiers. Although many researchers agree on the theoretical importance of diversity for an ensemble of classifiers, no unified and formal definition of diversity has been found.

Diversity measures are generally divided into two categories: pairwise and non-pairwise measures. Kuncheva [48] summarised and studied 10 of the most commonly used diversity measures. Although extensive experimental studies were conducted in [46, 48, 77, 76, 82], the researchers did not found a strong correlation between the aforementioned diversity measures and the Acc(ϕ) on all the examined datasets. Other studies, such as Ko et al.[43], have put forward new diversity measures, but in most cases, no single measure exhibited a strong correlation with ensemble accuracy.

In light of previous studies, the coincident failure diversity (CFD) [70] has been considered to be arguably the best theoretical representation of the diversity among classifiers [6] and therefore is chosen to examine the relationship between CFD and Acc(ϕ) under an increasing number of models in our research.

6.2.1.1 Experimental Design and Procedure

This experiment is intended to determine the relationship between CFD and Acc(ϕ). Achieving this aim requires observation of the association between CFD and Acc(ϕ) whilst ensemble size is increased. The detailed experimental procedure is described in the succeeding section.

CFD, is proposed by Partridge and Krzanowski [70]. It is defined by [48] in the manner illustrated in equation 6.1.

$$CFD = \begin{cases} 0 & \text{if } p_0 = 1 \\ \frac{1}{1-p_0} \sum_{i=1}^{L} \frac{L-i}{L-1} p_i & \text{if } p_0 < 1 \end{cases} \tag{6.1}$$

Where L is the number of base classifiers in the ensemble, and p_i denotes the probability that i randomly chosen classifiers will fail on a randomly chosen instance. Maximum diversity occurs when CFD=1, which indicates that the members of the ensemble create unique misclassifications. Minimum diversity is generated when CFD = 0, which means that all the member of the ensemble are identical.

6.2.1.1.1 Experimental procedure

1. Six datasets were used to examine the relationship between Acc(ϕ) and CFD. These datasets are Adult, Census, Cover Type, FARS, Poker and Web datasets. The detailed characteristics of these datasets are explained in Section 3.3.

2. To create a DP, a training dataset (Tr) is partitioned into 91 disjoint data subsets using sampling without a replacement technique. Each data subset contains 1.1 % from the instances in the Tr.

3. An MP is created by generating a base classier for each subset in the DP by using the j48 learning algorithm, which is the Weka [34] implementation of the C4.5 decision tree[73].

4. The individual models in the MP are evaluated over V, and any classifier that performs to a degree inferior to the accuracy of a random guess (50%) for binary classification problem is removed from the MP

5. An initial ensemble is built by randomly choosing three models from the MP.

6. The ensemble performance and its diversity are evaluated over V and TS on the basis of accuracy as a performance measure. Equation 6.1 is used to calculate the CFD diversity, and majority voting is adopted as an ensemble fusion strategy.

7. Two other classifiers are randomly chosen from the MP.

8. A new ensemble of classifiers is constructed, and steps 6 and 7 are repeated until no further improvement is achieved or models are no longer available in the MP.

6.2.1.2 Experimental Results

Figures 6.2 and 6.3 show the relationship between $Acc(\phi)$ and CFD over V and TS for the six examined datasets. Figure 6.2 illustrates that CFD is somewhat correlated with $Acc(\phi)$ the Adult, Cover Type and Poker datasets. This correlation varies in strength and direction of relationship from one dataset to another. On the other hand, Figure 6.3 indicates that $Acc(\phi)$ is unaffected by the change in CFD value on the Census, FARS and Web datasets.

To more accurately examined the relationship between CFD and $Acc(\phi)$, the Pearson correlation test [19] is conducted. Tables 6.1 and 6.2 list the Pearson correlation test results for the six datasets evaluated over V and TS, respectively. Both tables show a statistically significant relationship between $Acc(\phi)$ and CFD on the Adult, Cover Type and Poker datasets. $Acc(\phi)$ is strongly correlated with CFD over the Adult and Cover Type datasets but moderately correlated with that on the Poker dataset.

By contrast, no statistically significant relationship is found between $Acc(\phi)$ and CFD on the Census dataset. The test is also inapplicable to the FARS and Web datasets

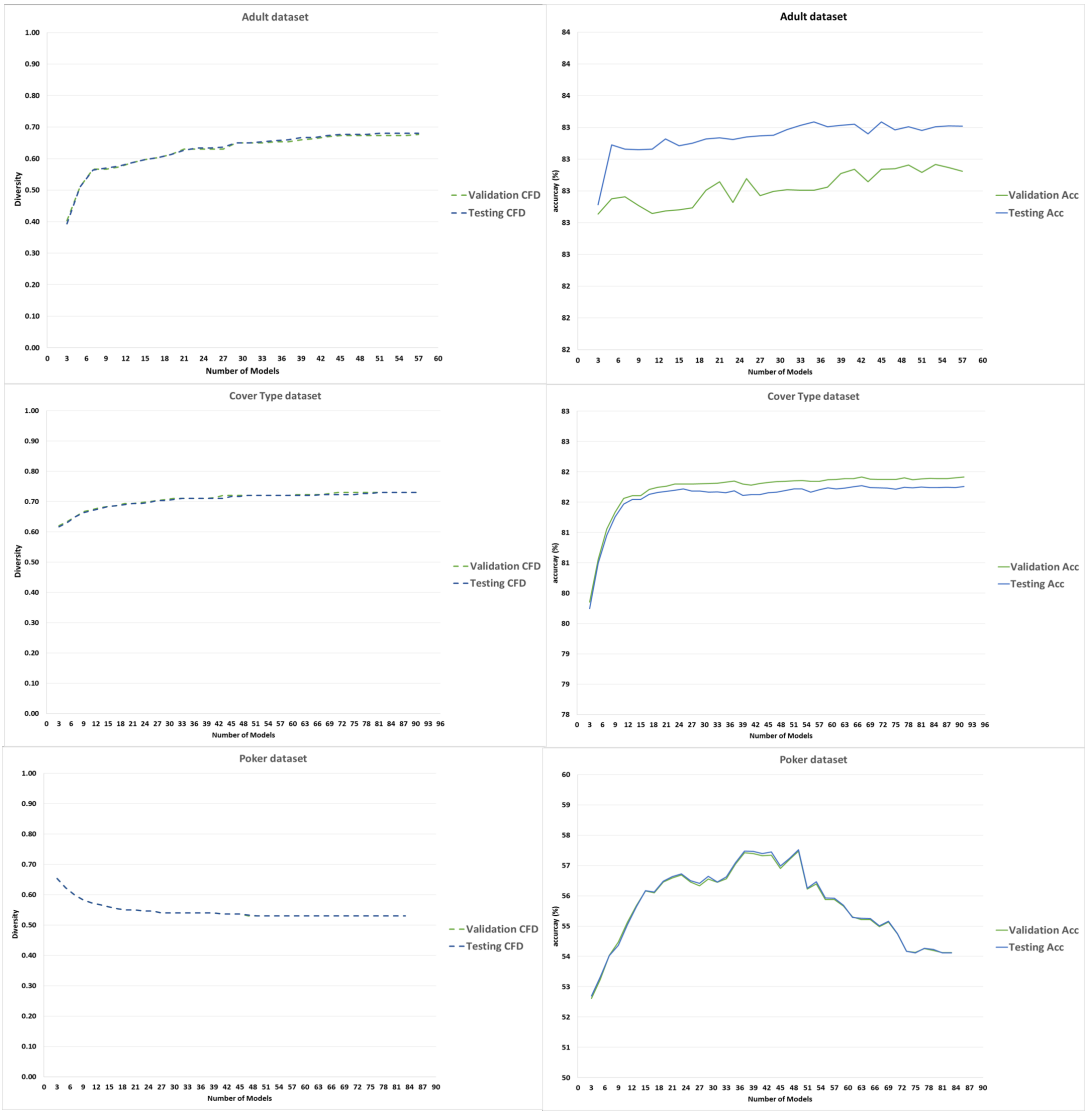


Fig. 6.2 Relationship between $Acc(\phi)$ and CFD over the validation and testing datasets for the Adult, Cover Type and Poker datasets. The diversity chart is on the left, whereas the accuracy chart is on the right.

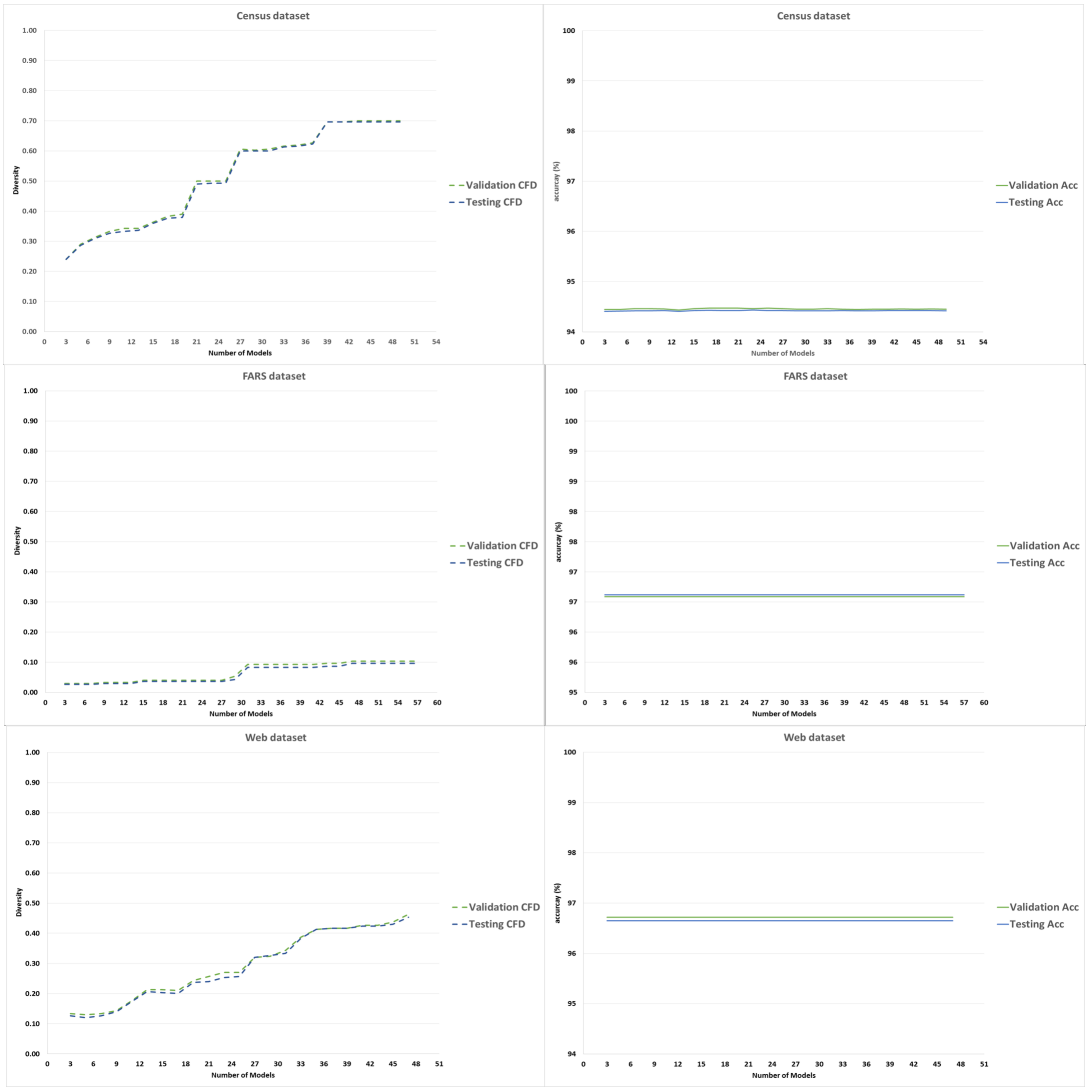


Fig. 6.3 Relationship between $Acc(\phi)$ and CFD over the validation and testing datasets for the Census, FARS and Web datasets. The diversity chart is on the left, whereas the accuracy chart is on the right.

because their ensemble performance is constant under an increasing number of models (Figure 6.3).

Table 6.1 Results of correlation analysis for the used datasets over the validation dataset

Dataset	r	P value	Pearson strength
Adult	0.740	<0.005	Strong
Census	-0.073	0.733	Not significant
Cover Type	0.891	<0.005	Very strong
FARS	NA	NA	NA
Poker	-0.459	0.003	Moderate
Web	NA	NA	NA

Table 6.2 Results of correlation analysis for the used datasets over the testing dataset

Dataset	r	P value	Pearson strength
Adult	0.931	<0.005	Very strong
Census	0.350	0.093	Not significant
Cover Type	0.864	<0.005	Very strong
FARS	NA	NA	NA
Poker	-0.452	0.003	Moderate
Web	NA	NA	NA

Interestingly, although a statistically significant relationship exists between Acc(ϕ) and CFD on the Adult, Cover Type and Poker datasets, the direction of the relationship is non-homogeneous over the three datasets. Acc(ϕ) and CFD are positively related on the Adult and Cover Type datasets and inversely related over the Poker dataset.

6.2.1.3 Summary

As mentioned earliest in this section 6.2.1, this brief investigation aims to decide on whether diversity can be regarded as a selection criterion in the proposed method. The experimental results show that using CFD as a selection criterion in the developed SM is difficult. Given that the relationship between CFD and Acc(ϕ) is unclear over the six examined datasets, CFD is excluded as a contribution evaluator criterion in our approach.

6.2.2 Design of the Selective Modelling Algorithm

6.2.2.1 Inputs

The proposed algorithm requires nine input parameters. These parameters are as follows:

1. DP: A set that contains references to all available data subsets. These subsets are created by partitioning a single huge training dataset.

2. TS: Testing dataset.

3. V: Validation dataset.

4. Accuracy tolerance (θ): In the proposed algorithm, the accuracy tolerance is a percentage range value that represents the amount of acceptable difference between two compared accuracy levels. θ affects two stages out of the four previously illustrated in Figure 6.1. These stages are model selection & ensemble evaluation (stage 3) and termination criterion check (stage 4). During the model selection steps in stage 3, a wide θ increases the chances that a selected model from an MP will be used as a member of the final ensemble. A narrow θ constrains the selection step, thereby restricting the number of models in the final ensemble. θ also influences stage 4, wherein a huge θ can terminate the algorithm in a few iterations. A small value increases the number of iterations before algorithm termination.

5. The size of the initial ensemble $|\phi_{init}|$: This refers to the smallest number of models required to construct an ensemble. The value of $|\phi_{init}|$ is related to the fusion strategy used in the constructed ensemble. Because the majority voting approach is adopted as a fusion strategy, three is the minimum number of models required to construct an ensemble.

6. Minimum accuracy of individual models (acc_{min}) : This value represents the lower bound for the acceptable accuracy of a model. During the model evaluation step in the second stage of the SM process , any model that performs lower than that of (acc_{min}) is discarded.

7. Number of models to be added at each iteration (π): In the proposed algorithm, ensemble size is expanded in each iteration. The value of π indicates the number of models added to an existing ensemble of classifiers at each iteration. This parameter affects stage 3, specifically in model selection tasks. a very big π is undesirable because the aim of SM is to gradually construct the final ensemble. The value of π is also influenced by the employed fusion strategy. In addition, it is used to regulate the number of models within the ensemble to generate an even or odd number of models at each iteration.

8. γ: This refers to the percentage of the minimum number of models to be checked before the termination criterion is applied. This parameter affects stage 4, and it is used as a protection from algorithm termination in early stages. A large γ value increases the number of required iterations for the algorithm.

9. λ: This pertains to the minimum number of successive ensembles with equal accuracy to be identified for the purpose of creating a checking point. The value of λ also affects the required iterations for the algorithm in stage 4.

6.2.2.2 Outputs:

The output of the SM algorithm is an ensemble of classifiers that consists of a few number of models and has a performance better or at last similar to the performance of the full ensemble that consists of all the available models.

6.2.2.3 Models selection Criteria:

For ensemble selection task, the ensemble performance is used as a selection criterion. At any iteration during the algorithm lifetime, If adding new models reduces the ensemble performance, then the latest added models should be discarded. Otherwise, the newly added models should be kept as members of the ensemble. As a result of the investigation in section 6.2.1, the diversity is not used as a selection Criteria.

6.2.2.4 Termination criterion:

Generally, the algorithm will be terminated if the process of adding new models to the ensemble is not enhancing the performance of the ensemble anymore. The details of how to measure the ensemble performance during the algorithm lifetime is illustrated in the next section.

6.2.3 Detailed Algorithm and Pseudocode:

Previously Figure 6.1 shows the 4 main stages for the SM algorithm. In this section the detailed steps for each main stage are presented and the pseudocode for the selective modelling algorithm is also presented. Figure 6.4 shows the conceptual framework for the SM algorithm. The presented framework is divided into four main steps as detailed below:

Stage 1: Data subsets selection and modelling:

At the beginning of each iteration, the number of models available in the MP is checked. If the number of models in MP is bigger or equal to π, there is no need to select a new batch of subsets; consequently, there is no need to build any new models. On the other hand, if the size of MP is less than π, then the following steps need to be performed:

(a) Select P subsets randomly from DP, where P = the number of processors.

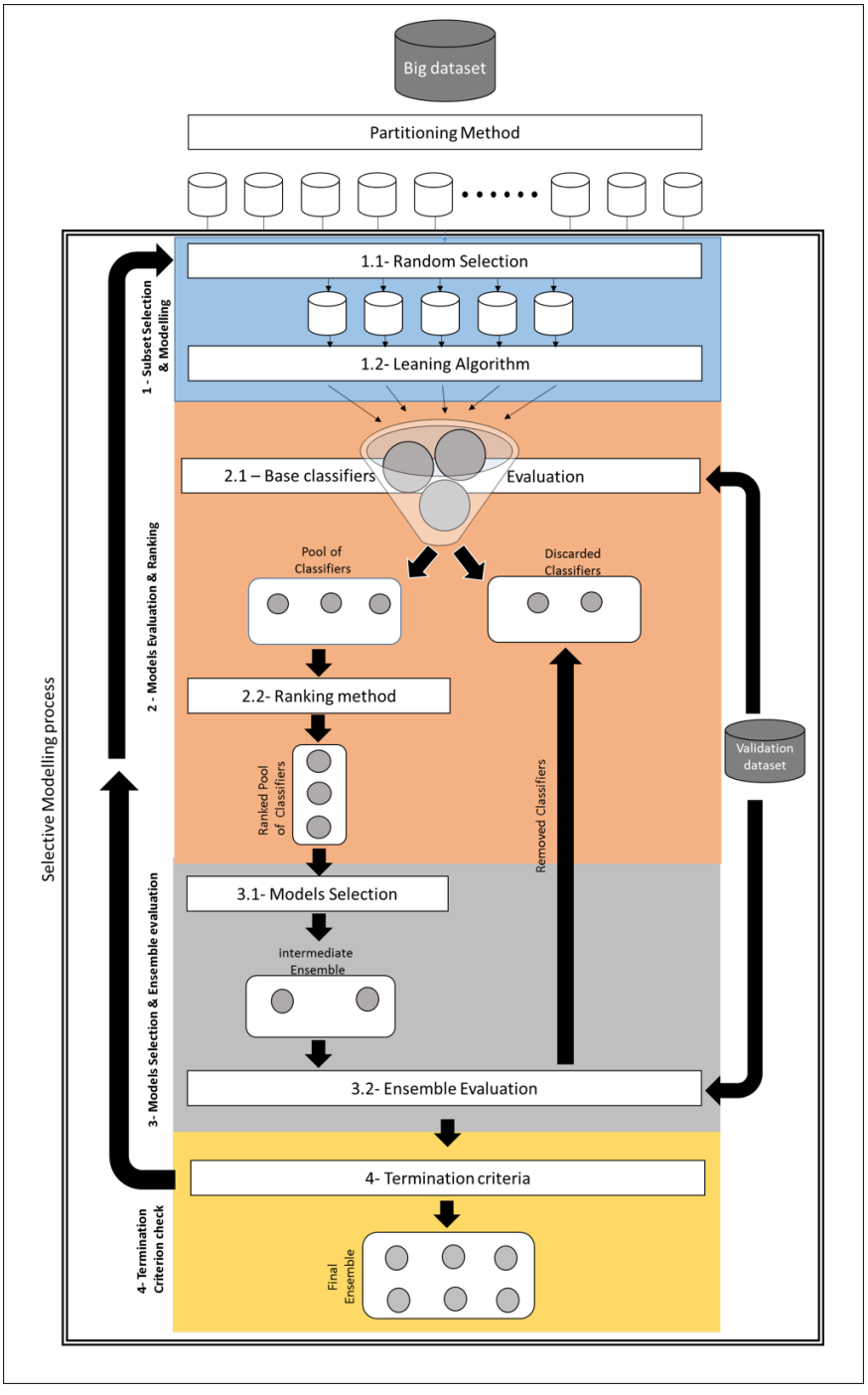


Fig. 6.4 Conceptual Framework for Selective Modelling

(b) Build a base classifier (a model) from each of the selected subsets in parallel, using a machine learning algorithm.

Stage 2: Models evaluation & Ranking:

(a) The generated models from the previous steps are evaluated on V.

(b) Weak classifiers will be discarded. Any model that performs on V less than (acc_{min}) will be discarded.

(c) The remained classifiers are added to MP.

(d) Make sure that MP contains models greater than or equal to π before moving to the next step. Otherwise, go back to step 1 (in stage 1) and select a new batch of subsets.

Stages 1 and 2 are expressed by lines 11 to 24 in Algorithm 3 .

Stage 3: Models selection & Ensemble evaluation:

Before applying any model selection technique, there is a need to construct the initial ensemble (θ_{init}). If the θ_{init} is not constructed, then step (a) is required to be performed. otherwise move to steps b and c.

(a) **Construct the initial ensemble ($\phi init$):** The ($|\phi_{init}|$) is determined as input argument to the proposed algorithm. To construct the θ_{init}, the following steps are need to be followed:

i. Choose the most accurate model (MAM) from MP to be the first member of the ϕ_{init}.

ii. Calculate the pairwise diversity between the MAM and the other models in MP. In the proposed method the Double-fault [32] (DF) pairwise diversity measure is used.

iii. Rank the models in MP according to their DF diversity from MAM.

iv. Choose the most ($|\phi init| - 1$) diverse models from MP. It should be noted that the diversity is used to select a possible candidate for the

constructed ensemble, but the final selection criteria for the member of the ensemble is decided using the ensemble accuracy. The ensemble selection criteria is described in step **Stage 3:**c.

v. Construct and evaluate the $\phi init$ on V.

vi. Begin a new iteration.

Algorithm 4 illustrates the detailed steps for the initial ensemble creation.

(b) **Increase the size of the ensemble:** After constructing and evaluating the ϕ_{init}, the $|\phi_{init}|$ will be gradually increased by π models at each iteration. The process of expanding the ensemble is continued until either the algorithm termination criterion is fulfilled or all the available subsets in DP have been checked. The following steps are required to enlarge the size of the ensemble. These steps are also expressed in the algorithm 3 lines 28 to 38:

i. Calculate the diversity between each model in MP and the last constructed ensemble.

ii. Rank the models in MP according to their diversity.

iii. Select the most π diverse models and add them to the ensemble.

iv. Construct and evaluate the ensemble on V.

(c) **Checking the selection criteria:** After adding new models to the ϕ_{init}, the performance of the new ensemble is compared to the performance of the preceding ensemble. If adding the new models reduces the ensemble performance, then the latest added models should be discarded. Otherwise, the new added models should be kept as members of the ensemble. This step is illustrated in the algorithm 3 lines 40 to 50:

Stage 4: Termination criterion:

The proposed algorithm will continue until the process of adding more models to the ensemble is not improving its performance any more. Fundamentally the termination criterion should be checked at the end of each iteration, but in order to prevent the algorithm from being terminated in very early stages the checking of the termination criterion will only be started after the selection of a specified number of models. This number of models is determined as an input parameter γ.

During the algorithm life cycle, a series of checkpoints for observing the ensemble performance is identified. The algorithm termination criterion check is applied at each checkpoint to decide whether to stop or to continue the selective modelling algorithm. A checkpoint consists of several successive ensembles that have similar performance. The required number of successive ensembles for a checkpoint is determined by input parameter λ. The performance at the checkpoint is calculated as an average of the performance of the enclosed ensembles. The algorithm will be terminated if the performance of two checkpoints is approximately equal. Algorithm 5 highlights the main tasks required for checking the termination criteria of the selective modelling algorithm.

6.3 Experimental Setup

6.3.1 Datasets

As indicated in this section, 13 datasets are used to evaluate the proposed SM algorithm. The main characteristics of the used datasets are summarised in Table 6.3. Detailed information about these datasets is presented in Section 3.3.

Algorithm 3: Selective Modelling

Inputs :

1. N: a set that contains references to all available subsets.
2. TS: Testing dataset.
3. V: Validation dataset.
4. θ: accuracy tolerance.
5. $|\phi_{init}|$: initial ensemble size.
6. Acc_{min}: the minimum accuracy for individual models.
7. π: number of models to be added at each iteration.
8. γ : the minimum no. of models needed to be checked before examining the stopping condition.
9. λ : the minimum no. of successive ensembles with equal accuracy needed to be achieved before ending the algorithm.

Output :

An ensemble of classifiers that consists of a few number of models and has a performance at last similar to the performance of the full ensemble.

Start :

```
1  Let φ_init = false.
2  Let φ_update = false.
3  Let φ_count = 0.
4  Let MpoolSorted = false.
5  Let p = number of available processors.
6  Let the maximum possible number of models (M_max) = |N|.
7  Let checkedM= 0.
8  Let equalAcc =0.
9  Let checkPoints is a list that contains the Acc(φ) on V for the all possible stopping points for the algorithm.
10 while termination criterion is not true do
11     if |MP| < π then
12         if (|MP| + |N|) ≥ π then
13             Select p subsets randomly from N.
14             Generate, in parallel, a base classifier(m) from each of the p selected subsets.
15             Evaluate the p models on the V.
16             Discard any base classifier which its acc(m_i) < Acc_min.
17             Add the remaining models to MP.
18             if |MP| < π then
19                 go to line 12.
20             end
21         else
22             exit.
23         end
24     end
25     if φ_init is not completed then
26         Construct the initial ensemble.
27     end
28     if ensemble not Updated then
29         if MP not Sorted then
30             Sort the models in MP in descending order according to their diversity.
31             MpoolSorted = true.
32         end
33         Select the highest π models from the MP, and add them to the φ.
34         checkedM = checkedM + π.
35         φ_update= true.
36     end
37     Increment φ_count.
38     Evaluate the φ on the V.
39     if ensembleCounter ≠ 1 then
40         if selection criteria is true; then
41             Evaluate the φ on the TS.
42             MpoolSorted = false.
43             if the current Acc(φ) ≅ the previous Acc(φ) then
44                 equalAcc ++.
45             else
46                 equalAcc =0.
47             end
48         else
49             Remove the last π models added to the φ.
50         end
51     else
52         Evaluate the φ on the TS.
53         MpoolSorted = false.
54         add the Acc(φ) on V to checkPoints list.
55     end
56 end
57 End
```

Algorithm 4: Creating the initial ensemble

Inputs :
1. MP: a set that contains references to the available models.
2. $|\phi_{init}|$: initial ensemble size.

Output :
An initial ensemble of classifiers where the size of the created ensemble = $|\phi_{init}|$.

Start :
while ϕ_{init} *is not completed* ***and*** *MP is not empty* **do**
  if ϕ_{init} *is empty* **then**
    let MAM = the most accurate model in MP.
    add MAM to the ϕ.
    checkedM++.
    Calculate the pairwise diversity between MAM and the models in MP.
    Rank the models in MP according to their diversity.
    Mpoolsorted = true.
  else
    let MDM = the most diverse model in MP.
    add MDM to the ϕ.
    checkedM++
  end
  if $|\phi| = |\phi_{init}|$ **then**
    ϕ_{init} = true.
  end
  ϕ_{update} = true.
end
End

Algorithm 5: Termination Criterion Method

Inputs :
1. M_{max} : the maximum possible number of models.
2. checkedM: the number of models that have been checked by the selective Modelling algorithm.
3. checkPoints List: contains the validation accuracy for all the check points before calling this method.
4. γ : the minimum no. of models that needed to be checked before examining the stopping condition.
5. λ : the minimum no. of successive ensembles with equal accuracy that needed to be achieved before ending the algorithm

Output :
A Boolean value which represents whether or not to terminate the algorithm.

Start :
if $checkedM \geq (M_{max} \times \gamma)$ ***and*** $equalAcc = \lambda$ **then**
  let avr = the average of the validation accuracy of the last λ ensembles.
  let LastCheckP = the last entry in the checkPoints List.
  if $avr \approx LastCheckP$ **then**
    return true.
  end
else
  return false.
end
End

Table 6.3 Main characteristics of the datasets used in the experiment

Dataset	Training Instances	Validation Instances	Testing Instances	No. of Attributes
Adult	34,536	9,768	16,281	60
Census	245,389	59,859	99,762	31
Connect4	48,173	9,458	20,268	43
Cover Type	366,037	156,873	58,102	39
FARS	89,648	14,010	30,021	30
HUGGS	5,390,000	2,310,000	3,300,000	28
IJCNN01	60,138	14,997	91,701	7
KDD99	1,361,892	1,469,530	311,029	47
Poker	574,004	143,502	307,503	11
Shuttle	54,331	8,700	14,500	10
Skin	216,974	34,308	73,518	4
SUSY	2,800,000	700,000	1,500,000	19
Web	64,839	14,925	14,951	262

6.3.2 Experimental procedure

- The values of the input parameters are illustrated in Table 6.4. These values are obtained by testing different values for these parameters in several runs of the algorithm over a small set from the 13 datasets.

Table 6.4 Parameters values

Input Parameter	Value
θ	0.5%
$\lvert\phi_{init}\rvert$	3
acc_{min}	50%
π	2
γ	20%
λ	3

- A DP is created by dividing a Tr into 91 disjoint datasets by sampling without replacement as a partitioning method.

- The Weka [34] implementation (j48) of the traditional C4.5 decision tree [73] is used as a learning algorithm to generate base classifiers.

- Majority voting is used as a fusion strategy.

- To estimate ensemble accuracy, the Hold out [47, p. 9] evaluation strategy is adopted. The derived accuracy (see Results section) is an average of 10 runs.
- In each run, the original dataset is divided into training and validation datasets, with 70% of the original dataset size used for training and the remaining 30% employed for validation. Sampling without replacement is also carried out as a partitioning method.
- The performance of the proposed algorithm is then evaluated and statistically tested, as described in Section 6.3.3.

6.3.3 Evaluation Method

To evaluate the effectiveness and efficiency of the proposed SM algorithm, the performance of the constructed ensemble that uses the SM is compared with the performance of the following:

- The full ensemble: An ensemble that consists of all the available models (FE).
- An ensemble that is constructed using the forward search selection strategy (FS): Forward search or forward stepwise selection is a traditional ensemble selection method that proceeds as follows [77, 12, 94, 86]: Start with an empty ensemble. Then, at each iteration, add the model that maximises the ensemble performance on a validation dataset. Continue selecting more models until all the available models are examined. The implementation of the forward search algorithm in this experiment is similar to that described earlier, with one exception; instead of selecting the model that maximises ensemble accuracy, random selection is performed to select π models in each iteration. This step is intended to save time in the evaluation of the created ensemble; such savings translates to a huge difference, especially in situations wherein a very large MP is used.

The Friedman test [20] is also carried out to statistically compare the performance of the proposed method and that of other ensemble methods over the 13 datasets.

The critical difference (CD) diagram [20] is used to graphically summarise whether a significant difference exists between the proposed algorithm and the other methods. Significance is considered at the 5% level. Detailed information on the statistical tests is provided in Section 3.4.

6.4 Experimental Results and Evaluation

The empirical results derived with the proposed SM algorithm are discussed in this section. To examine the quality of classification made by the ensemble that is constructed using SM, the performance of the constructed ensemble is compared to that of two other ensembles: 1) an ensemble that contains all available models (FE) and 2) an ensemble that is constructed using FS. The performance of SM is comprehensively discussed in Section 6.4.1.

In addition to the performance comparison, the efficiency of the proposed method is compared with that of the two mentioned ensembles in Section 6.4.2.

6.4.1 Performance of Selective Modelling

This section reports the experimental results on accuracy. The classification accuracy of the ensembles ($Acc(\phi)$) that are constructed by SM, FS and FE are illustrated using the 13 datasets mentioned in Section 6.3. To obtain reliable statistics over these findings, each result represents an average of 10 runs. The averaged ensemble accuracy $Acc(\phi_{ts})$ over TS of the 13 datasets, are shown in Table 6.5.

Table 6.5 shows the number of models (M) in each ensemble, $Acc(\phi)$ and standard deviations. The proposed SM was more accurate than the two other ensembles on seven out of the 13 datasets. Figures 6.5 and 6.6 indicate an improvement in $Acc(\phi_{ts})$ over the algorithm's lifetime on seven datasets where the SM achieved the highest accuracy amongst the three ensembles.

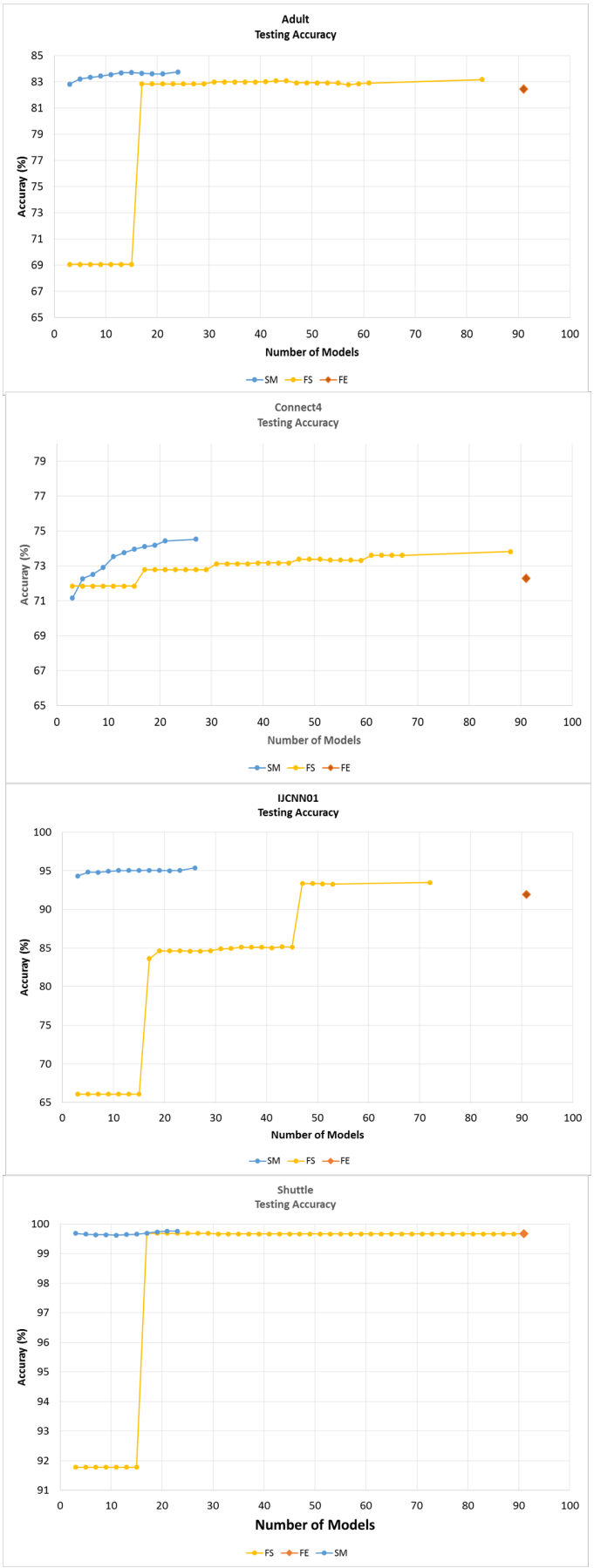


Fig. 6.5 Datasets wherein SM achieves the highest accuracy in the comparison of ensemble methods

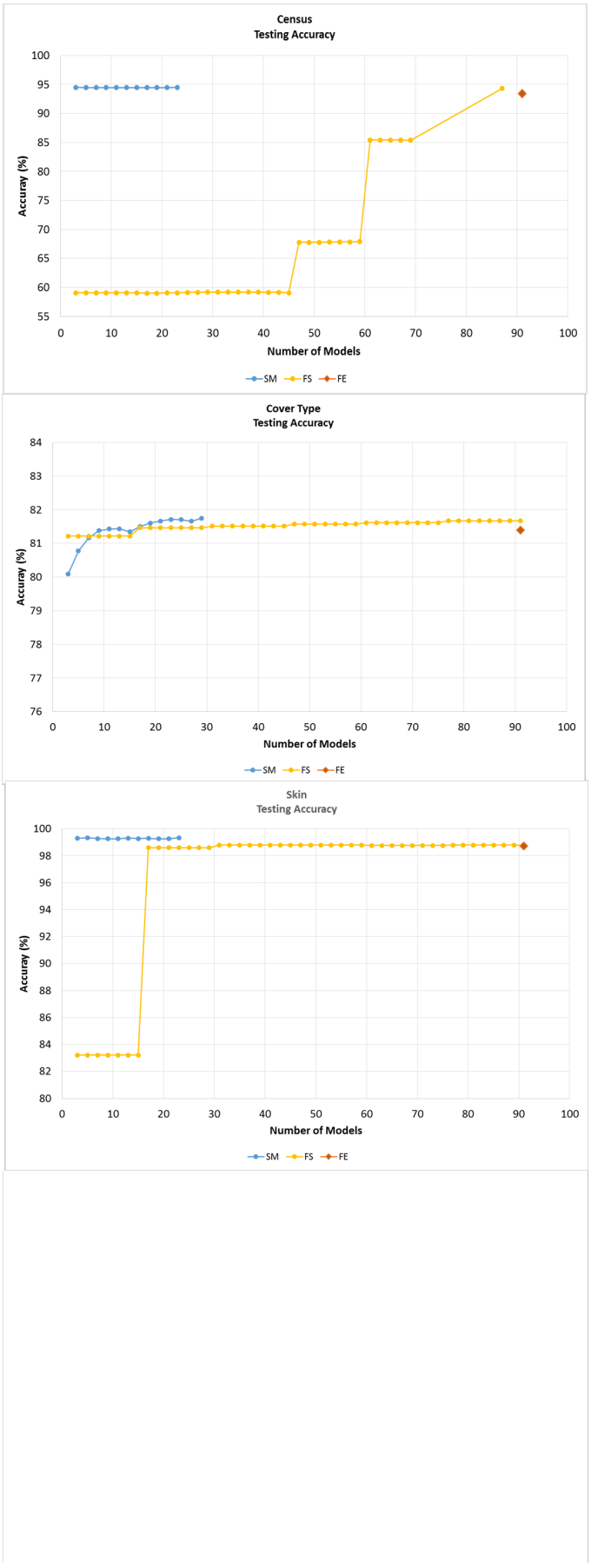


Fig. 6.6 Datasets wherein SM achieves the highest accuracy in the comparison of ensemble methods

Table 6.5 Average testing ensemble accuracy and number of models over 13 datasets

	FE		FS		SM	
Datasets	M	$acc(\phi)_{FE}$	M	$acc(\phi)_{FS}$	M	$acc(\phi)_{SM}$
adult	91	82.43 ±0.38	83	83.16 ±0.30	26	83.75 ±0.30
Census	91	93.95 ±0.18	88	94.29 ±0.17	23	94.48 ±0.02
Connect4	91	72.28 ±0.87	88	73.82 ±0.94	27	74.54 ±0.55
Cover Type	91	81.40 ±0.13	91	81.68 ±0.14	29	81.75 ±0.21
FARS	91	96.62 ±0.00	91	96.62 ±0.00	23	96.62 ±0.00
HIGGS	91	71.05 ±0.04	91	71.17 ±0.02	29	71.12 ±0.04
IJCNN01	91	91.89 ±0.40	72	93.48 ±0.49	26	95.37 ±0.16
KDD99	91	92.11 ±0.02	91	92.11 ±0.01	23	92.11 ±0.03
Poker	91	57.09 ±2.30	79	61.70 ±1.85	39	60.35 ±0.02
Shuttle	91	99.68 ±0.10	91	99.66 ±0.10	23	99.71 ±0.08
Skin	91	98.73 ±0.03	91	98.78 ±0.03	23	99.33 ±0.06
SUSY	91	79.38 ±0.01	91	79.39 ±0.01	26	79.38 ±0.04
Web	91	96.65 ±0.01	89	96.94 ±0.13	23	96.65 ±0.00

Furthermore, SM performed at a level equal to the performance of FE and FS over the FARS and KDD99 datasets. Figure 6.7 shows $Acc(\phi_{ts})$ over the algorithm's lifetime on the aforementioned datasets.

The SM algorithm performed at a lower level than FS did on four datasets (HUGGS, Poker, SUSY and Web) but did not register an accuracy level lower than that of FE. The worst performance of SM ensembles occurs in the evaluation against FS on the Poker dataset with $\Delta(Acc(\phi)_{SM}, Acc(\phi)_{FS})$ = -1.34%. SM performance is also lower on the HUGGS, SUSY and Web datasets, but its difference from the other models is minimal; that is, it differs from $\Delta(Acc(\phi)_{SM}, Acc(\phi)_{FS})$ by –0.05%, –0.01% and –0.29%, respectively . Figure 6.8 illustrates the four datasets wherein SM exhibits a performance worse than that of FS.

6.4.1.1 Assessment of Selective Modelling Performance

As previously stated, the accuracy of SM on the 13 examined datasets is higher than that of the other two ensembles on seven datasets (Figures 6.5 and 6.6). However, its performance is lower than that of FS on four datasets (Figure 6.8); nevertheless,

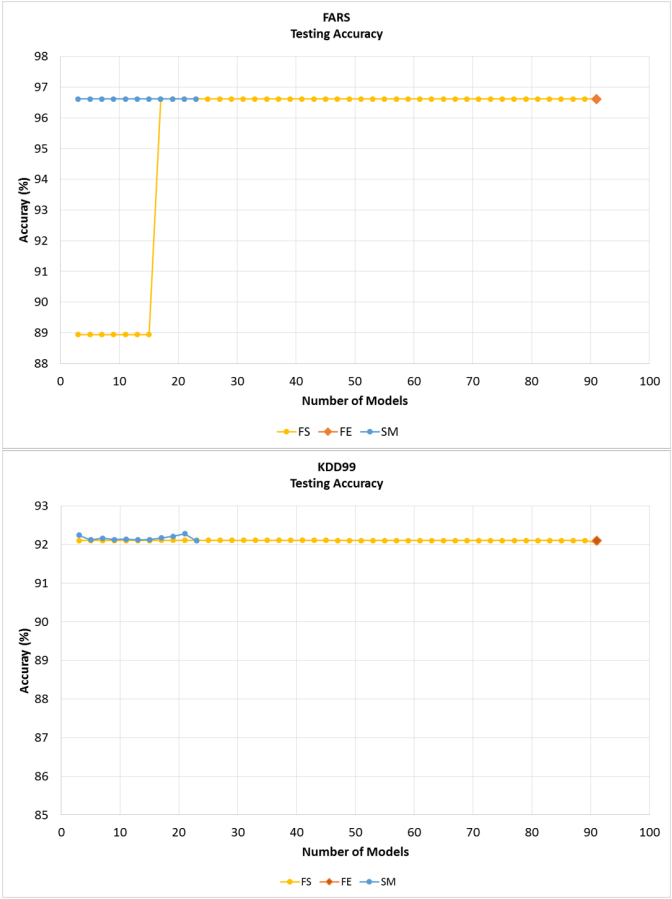


Fig. 6.7 SM achieves a performance equal to that of FE and FS on two datasets

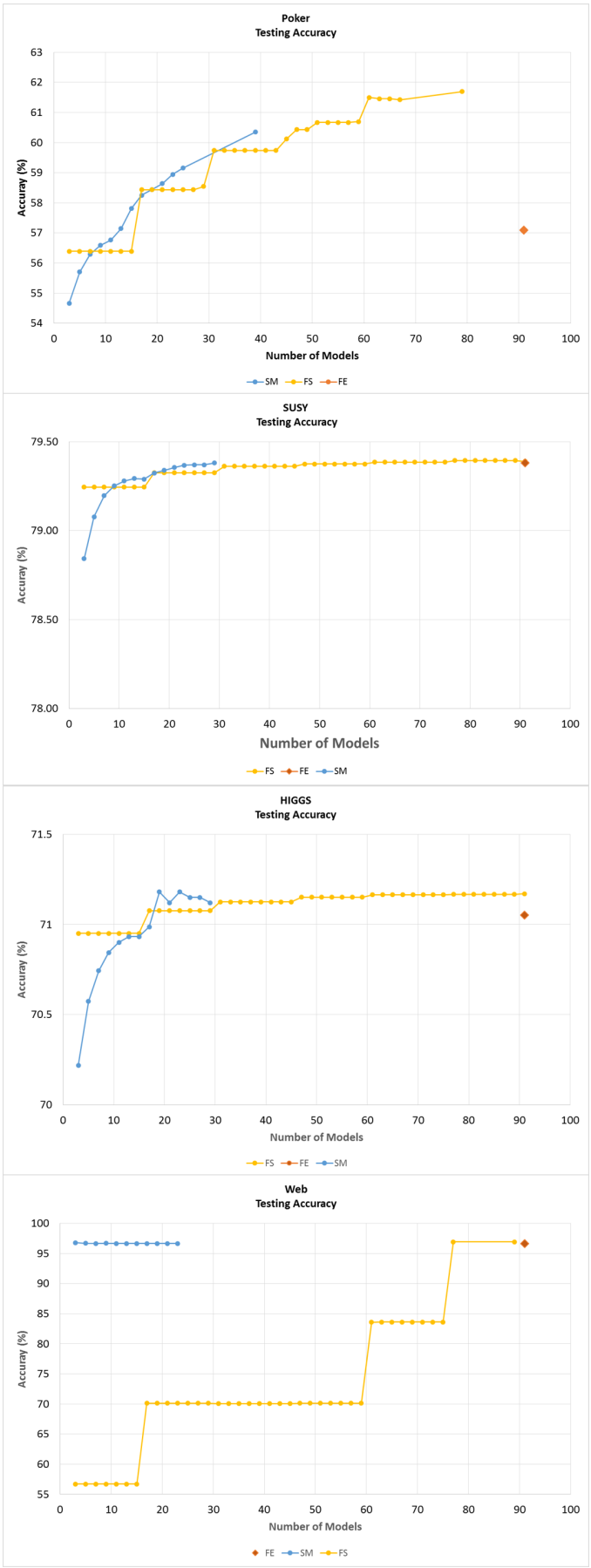


Fig. 6.8 The performance of SM is worse than that of FS on 4 datasets

it consistently exhibits a higher performance than does FE. So it is important to understand the reason for this different behaviour across varied datasets.

Three main factors distinguish the SM method from the FS selection method in terms of accuracy. These factors are the 1) removal of weak classifiers, 2) ranking and 3) termination criteria, which are described in detail in Section 6.2. To examine the effects of these elements on ensemble accuracy, the comparison was extended by adding two different versions from each of the selection methods analysed in Section 6.4.1. The extension proceeds as follows:

- FE: This is the same ensemble used previously.
- Forward search selection without removal of weak classifiers (FSoutRWC): This is also the same ensemble previously adopte d.
- Forward search selection with removal of weak classifiers (FSwithtRWC): Weak classifiers are removed in a similar manner as that done for the SM discussed in Section 6.2.
- SM with ranking (SMwithRNK): This modelling method is that proposed in this chapter and the approach used in the preceding chapter.
- SM without ranking (SMoutRNK): This approach is the proposed method but without the ranking phase described in Section 6.2.

Table 6.6 shows the averaged testing ensemble accuracy $Acc(\phi_{ts})$ over the 13 datasets for 10 runs, the number of models in each ensemble and the rank of each ensemble method on each dataset.

As indicated in Table 6.6, although introducing the removal of weak classifiers in the FS improves the performance of forward search selection (FSwithRWC) on the Adult, Census, IJCNN01 and Skin datasets, the introducing also results in lower performance than that achieved by FSoutRWC over the Connect4, Poker, Shuttle and Web datasets. While removing weak classifiers does not affect the performance of

Table 6.6 Average testing ensemble accuracy (%), rank and number of models over 13 datasets

	FE		FSoutRWC		FSwithRWC		SMwithRNK		SMoutRNK	
	M	$acc(\phi)$	M	$acc(\phi)$	M	$acc(\phi)$	M	$acc(\phi)$	M	$acc(\phi)$
Adult	91	82.43 (5)	83	83.16 (4)	57	83.40 (2)	26	83.75 (1)	23	83.37 (3)
Census	91	93.95 (5)	86	94.29 (4)	49	94.44 (2)	23	94.48 (1)	23	94.4 (3)
Connect4	91	72.28 (5)	88	73.82 (2)	59	72.65 (3)	27	74.54 (1)	28	72.4 (4)
CoverType	91	81.40 (5)	91	81.68 (2.5)	91	81.68 (2.5)	29	81.75 (1)	29	81.47 (4)
FARS	91	96.62 (3)	91	96.62 (3)	57	96.62 (3)	23	96.62 (3)	23	96.62 (3)
HIGGS	91	71.05 (5)	91	71.17 (1.5)	91	71.17 (1.5)	29	71.12 (3)	29	71.08 (4)
Ijcnn	91	91.89 (5)	72	93.48 (4)	51	95.07 (2)	26	95.37 (1)	26	94.91 (3)
KDD99	91	92.11 (3)	91	92.11 (3)	91	92.11 (3)	23	92.11 (3)	23	92.11 (3)
Poker	91	57.09 (5)	79	61.70 (1)	70	60.76 (2)	49	60.35 (3)	37	58.95 (4)
Shuttle	91	99.68 (2)	91	99.66 (3)	59	99.63 (4.5)	23	99.71 (1)	23	99.63 (4.5)
Skin	91	98.73 (5)	91	98.78 (4)	59	99.23 (3)	23	99.33 (1)	23	99.26 (2)
SUSY	91	79.38 (3.5)	91	79.39 (1.5)	91	79.39 (1.5)	26	79.38 (3.5)	26	79.33 (5)
Web	91	96.65 (3.5)	89	96.94 (1)	74	96.65 (3.5)	23	96.65 (3.5)	23	96.65 (3.5)
Average rank		4.23		2.65		2.58		2		3.54
Overall rank		5		3		2		1		4

FS over the remaining five datasets. The loss of accuracy upon the removal of weak classifies need to be investigated in further study. While the removal of weak classifiers does not influence performance on five other datasets (Cover Type, FARS, HUGGS, KDD99 and SUSY) because no weak classifiers require removal, as indicated in the definition of weak classifiers in Section 6.2. This can be confirmed from Table 6.6, where M is constant on any dataset that exhibits equal performance for FSwithRWC and FSoutRWC.

As discussed in section 6.2, taking model ranking into consideration prevents SMoutRNK from achieving a performance that surpasses that of SMwithRNK (Table 6.6), suggesting that using ranking in the proposed SM enhances performance or at least prevents a decline in such performance.

To facilitate the comparison of ensemble performance on different datasets, the Friedman test and the CD diagram [20] are used, as described in Section 3.4. Figure 6.9 shows the CD diagram for the accuracy levels ranked in Table 6.6.

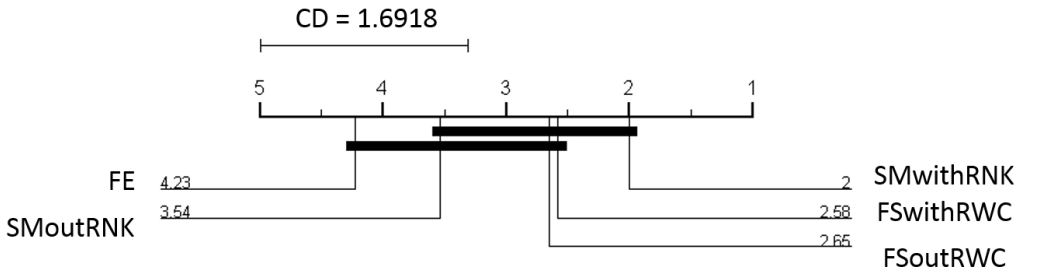


Fig. 6.9 CD diagram of the average ranks for different ensemble performance levels over 13 datasets (derived from the results in Table 6.6).

The results reveal the grouping of ensemble classifiers into two. The first group comprises SMwithRNK, SMoutRNK, FSwithRWC and FSoutRWC, indicating that no significant statistical difference in performance exists amongst these classifiers on the 13 datasets. The second group consists of FSwithRWC, FSoutRWC, SMoutRNK and FE, also indicating no significant statistical difference in accuracy on all the datasets. The only significant difference is that between the proposed method (SMwithRNK) and FE. Fundamentally, SMwithRNK is the best ensemble in terms of accuracy, with

the classifier achieving an average rank of 2. By contrast, FE is the worst ensemble, having a rank of 4.23.

6.4.2 Efficiency of Selective Modelling

Given that SM is an iterative method (see section 6.2), the time required to create an ensemble of classifiers by using SM is the sum of iteration time and the time needed to create a DP, as shown in equation 6.2.

$$T_{SM} = T_p + \sum_{i=1}^{I} T(i) \tag{6.2}$$

where T_{SM} is the SM time, T_p denotes the partitioning time, I represents the number of iterations and $T(i)$ is the time for the i^{th} iteration.

Iteration time can be calculated thus:

$$T(i) = T_{Tr} + T_{ee} + T_{oth} \tag{6.3}$$

where:

T_{Tr} is the training (modelling) time, which is the time required to train the data subsets selected from a DP to create a base classifier by using the machine learning algorithm;

T_{ee} denotes the ensemble evaluation time, which pertains to the time required to evaluate an ensemble over the TS and V datasets;

T_{oth} represents other times, which refer to the times required to access files, calculate DF diversity, rank the models in an MP and check the termination criterion.

Table 6.7 and Figure 6.10 show the averaged total time for the same ensembles whose accuracy levels over the 13 datasets are summarised in Table 6.6. Clearly, the SM algorithm, in both versions, with and without ranking, performs more rapidly than FS and FE do on almost all the datasets.

FS with and without the removal of weak classifiers exhibits the worst efficiency level. That FS exhibits a performance lower than that of FE is unsurprising because it requires considerably more iterations than do FE and SM. Given that FS needs a much

Table 6.7 Average time (in minutes) required by the different ensemble methods that illustrated in Table 6.6 and the standard division for the 13 datasets

	FE	FSoutRWC	FSwithRWC	SMwithRNK	SMoutRNK
adult	0.23 ±0.01	0.44 ±0.02	0.28 ±0.01	0.16 ±0.02	0.14 ±0.01
Census	0.81 ±0.03	2.23 ±0.02	1.13 ±0.03	0.62 ±0.03	0.57 ±0.03
Connect4	0.23 ±0.01	0.58 ±0.02	0.37 ±0.04	0.18 ±0.04	0.18 ±0.03
Cover Type	1.10 ±0.01	4.46 ±0.18	4.04 ±0.17	0.90 ±0.02	0.76 ±0.02
FARS	0.37 ±0.01	0.95 ±0.03	0.55 ±0.01	0.26 ±0.13	0.25 ±0.01
HIGGS	20.19 ±0.39	70.23 ±04.53	63.85 ±2.63	13.79 ±0.46	12.04 ±0.36
Ijcnn	0.55 ±0.01	0.79 ±0.05	0.57 ±0.01	0.31 ±0.04	0.31 ±0.04
KDD99	6.36 ±0.18	37.85 ±1.76	33.56 ±1.71	5.40 ±0.19	4.30 ±0.15
Poker	1.84 ±0.05	6.03 ±0.04	4.40 ±0.32	2.96 ±0.95	1.67 ±0.48
Shuttle	0.19 ±0.01	0.50 ±0.02	0.50 ±0.02	0.13 ±0.01	0.12 ±0.01
Skin	0.54 ±0.01	1.61 ±0.07	0.88 ±0.03	0.36 ±0.01	0.32 ±0.02
SUSY	9.41 ±0.24	31.74 ±1.40	28.36 ±1.11	5.88 ±0.60	5.04 ±0.48
Web	0.29 ±0.02	0.63 ±0.15	0.35 ±0.04	0.22 ±0.01	0.20 ±0.01

longer time than that required by the other two ensembles, the investigation in this section is focused on the efficiency levels of SM and FE.

Although SMoutRNK is the fastest amongst the compared ensembles, our focus in this section is the efficiency of SMwithRWC because it is the most accurate method amongst the approaches (see section 6.4.1). SMwithRWC was faster than FE on 12 out of the 13 examined datasets. The highest improvement rate was registered on IJCNN01, where SMwithRWC performed faster than FE by 44%. The lowest improvement occurs over KDD99, where the proposed method was more rapidly than FE by about 15%. The only exception was the Poker dataset, in which SMwithRWC exhibits a performance lower than that of FE by 60%. This variation in the behaviour of SMwithRWC compared with FE needs further analysis to determine why SMwithRWC is efficient over 12 datasets and inefficient on one. This issue is discussed in Section 6.4.2.1.

Figure 6.11 shows the CD diagram for the ranked average time shown in Table 6.7 where SM based (SMoutRNK and SMwithRNK) methods are statistically faster than FS-based methods. Although the CD diagrams reflect that the efficiency of SMoutRNK is superior to that of SMwithRWC, the diagram also confirms that no

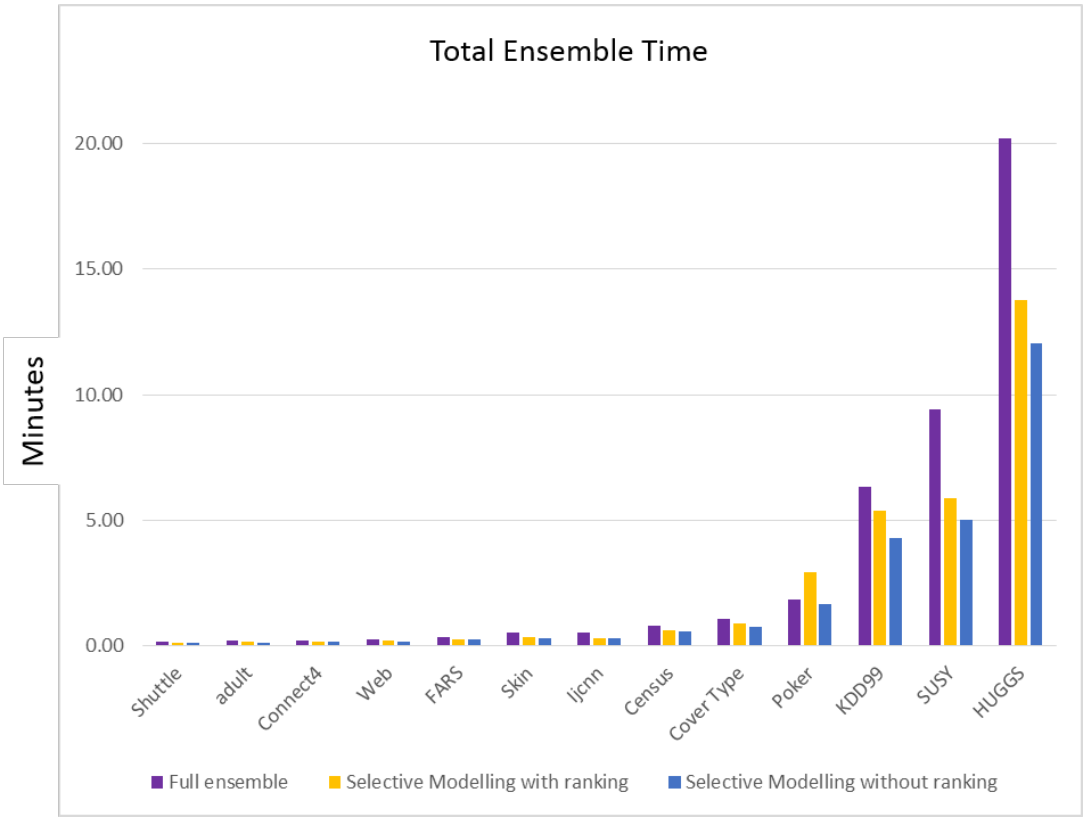


Fig. 6.10 Total time (in minutes) for the 13 examined datasets

significant difference in time exists between the two methods. Additionally, the diagram illustrates that no significant difference in time is found between SMwithRWC and FE; however, a statically significant difference in time exists between FE and SMoutRNK.

6.4.2.1 Factors Affecting the Time of the Selective Modelling

Table 6.7 shows the time elapsed in generating an ensemble of classifiers by using four different model selection methods in comparison to FE. As indicated in the previous section, the proposed SMwithRWC method is more efficient than FS over all the examined datasets, except the Poker dataset, wherein FE is faster than SMwithRWC. Elucidating the factors behind this exception necessitates a breakdown of the total time for both methods to its basic elements. The time spent by the SM method can be

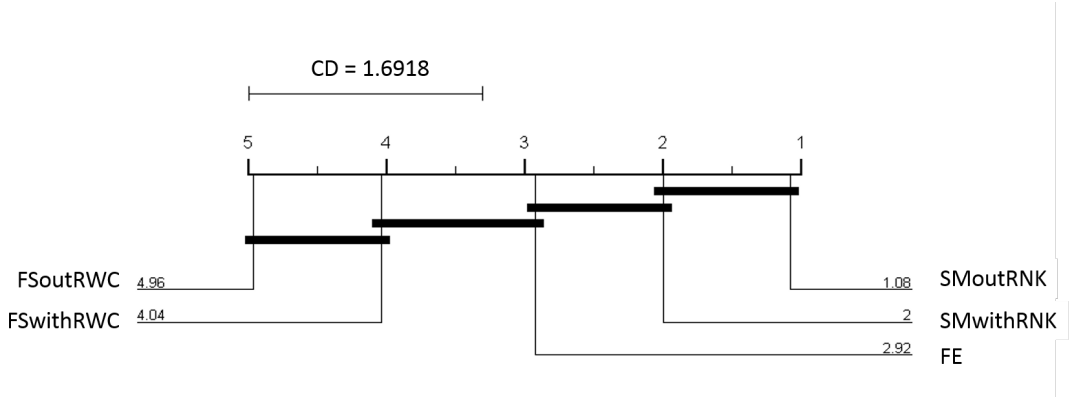


Fig. 6.11 CD diagram of the average ranks for different ensemble methods over 13 datasets (derived from the results in Table 6.7).

broken down using equations 6.2 and 6.3, whereas that spent by FE can be calculated using equation 6.4.

$$T_{FE} = T_p + T_{Tr} + T_{ee} + T_{oth} \tag{6.4}$$

Table 6.8 illustrates the broken down total times of FE and SMwithRWC. Fundamentally, the T_{Tr} derived with SMwithRNK was smaller than the T_{Tr} obtained with FE over all the used datasets. The lower training time for SMwithRNK is due to the small number of models and the principle of building models when they are needed (see Section 6.2). Conversely, T_{ee} varies as follows: T_{ee} was slower with SMwithRNK than with FE over eight out of the 13 datasets; the time was equal to FE on the Connect4 dataset. T_{ee} entails a longer time with SMwithRNK than that required by FE over four datasets (Cover Type, HUGGS, KDD99 and Poker).

Although SMwithRNK need more evaluation time than FE on four datasets, its total time was lower on all of them, except the Poker dataset. An extensive examination reveals that SMwithRNK exhibits a performance worse than that of FE in terms of time under conditions wherein the difference between the evaluation time in the two methods is larger than that between their training times, as indicated in equation 6.5. In other words, SMwithRNK will exhibit a performance worse than that of FE with

Table 6.8 Detailed average times (in minutes) for FE and SMwithRNK for 13 used datasets

	FE					SMwithRNK				
	M	I	T_{Tr}	T_{ee}	T_{oth}	M	I	T_{Tr}	T_{ee}	T_{oth}
adult	91	1	0.16	0.05	0.02	25	12	0.08	0.03	0.04
Census	91	1	0.42	0.20	0.19	23	11	0.23	0.17	0.21
Connect4	91	1	0.15	0.07	0.01	25	15	0.08	0.07	0.03
Cover Type	91	1	0.73	0.22	0.14	29	14	0.24	0.40	0.26
FARS	91	1	0.19	0.09	0.09	23	11	0.09	0.07	0.10
HUGGS	91	1	12.35	6.02	1.83	29	14	4.05	6.50	3.24
Ijcnn	91	1	0.35	0.16	0.04	24	12	0.18	0.08	0.05
KDD99	91	1	3.60	1.60	1.15	23	11	1.23	2.19	1.99
Poker	91	1	1.16	0.58	0.09	49	27	0.89	1.70	0.37
Shuttle	91	1	0.13	0.05	0.02	23	11	0.07	0.04	0.03
Skin	91	1	0.32	0.16	0.05	23	11	0.17	0.12	0.07
SUSY	91	1	5.58	2.88	0.95	26	12	1.87	2.40	1.61
Web	91	1	0.15	0.04	0.09	23	11	0.08	0.03	0.10

respect to time if the duration of ensemble evaluation is longer than the saved time during training.

$$\Delta(FE(T_{Tr}), SM(T_{Tr})) < \Delta(FE(T_{ee}), SM(T_{ee})) \tag{6.5}$$

Where:

$\Delta(FE(T_{Tr}), SM(T_{Tr}))$: is the difference between the training time in FE and in SMwithRNK.

$\Delta(FE(T_{ee}), SM(T_{ee}))$: is the difference between the time of evaluation in FE and in SMwithRNK.

Evaluation time is related to M and I; the higher the number of models in an ensemble, the longer the time required to evaluate these model and calculate the final decision of the ensemble. Moreover, a high number of iterations results in more time required because evaluating the created ensemble is needed at the end of each iteration. Table 6.8 shows that the ensemble created from the Poker dataset with the use of the proposed method consists of 49 models, which accounts for about 53% of the maximum possible number of models (91). The table also indicates that SMwithRNK

requires 27 iterations (representing about 60% of the maximum *I*) to build the final ensemble from the Poker dataset; this value is nearly twice the required number of iterations for the other dataset. As a result of requiring a large number of M and I, SMwithRNK performs worse than does the FE ensemble.

To evaluate the influence of T_{Tr} and T_{ee} on the total time of the SMwithRNK and FE ensemble methods as described in equation 6.5, the same experiment is repeated multiple times on the Poker dataset (see Section 6.3) but with a gradual increase in DP size from 91 to 591 models. Then, an ensemble is built from each DP by using FE and SMwithRNK. Table 6.9 illustrates the average time required by 10 runs to create an ensemble of classifiers from the Poker dataset with different DP sizes. Although T_{ee} in the proposed method remains less than that in FE even after DP size is increased, the total time required by SMwithRNK decreases with increasing DP size. This improvement in SMwithRNK is due to the huge difference between the training times of both methods; this difference confirms the relationship between T_{Tr} and T_{ee} described in equation 6.5.

Table 6.9 Average time (in minutes) spent to create an ensemble from the Poker dataset by using FE and SMwithRNK under varying DP sizes

	DP	FE					SMwithRNK				
		M	I	T_{Tr}	T_{ee}	T_{FE}	M	I	T_{Tr}	T_{ee}	T_{SM}
Poker	91	91	1	1.16	0.58	1.84	49	27	0.89	1.70	2.96
	191	191	1	2.39	1.21	3.70	59	30	1.00	2.34	3.77
	291	291	1	3.69	1.86	5.66	62	36	1.01	2.52	3.99
	391	391	1	4.91	2.46	7.49	86	47	1.36	4.60	6.58
	491	491	1	6.17	2.99	9.31	103	55	1.64	6.27	8.62
	591	591	1	7.46	3.61	11.23	119	65	1.88	8.39	10.58

6.5 Summary

In this chapter, an SM algorithm is proposed as an alternative to traditional model selection methods for building an efficient ensemble of classifiers when large datasets are used. Static and dynamic ensemble selection methods depend on the principle of

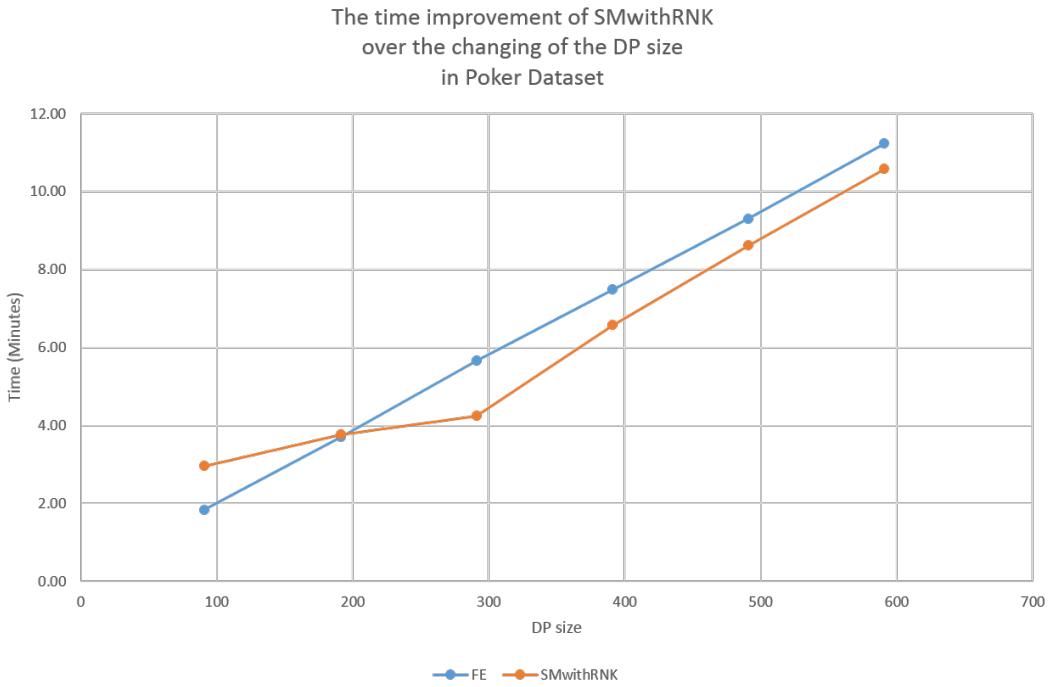


Fig. 6.12 The time improvement of SMwithRNK over the changing of the DP size in Poker Dataset

selecting models from a pool of models that contains all possible models. Creating such a pool is very costly in terms of time and resources, particularly when dealing with big datasets that need to be divided into numerous subsets.

The proposed SM is an iterative method that aims to save time and resources in building an ensemble of classifiers from a large dataset. This time saving advantage is achieved by the iterative combination of MP creation and selection. The main idea in SM is to build what is needed. That is, instead of creating models from all the available datasets in a DP, the SM algorithm iteratively generates models to reduce the number of unused models. As a result, it save resources which is useful when dealing with extreme large dataset.

The performance and efficiency of the proposed method is compared to two other ensemble methods: FE and FS strategy . The proposed SM method creates more accurate ensembles of classifiers than did the other two methods on 54% of the examined datasets. It also consistently attains an accuracy higher than that of the FE

ensemble. Additionally, SM can generate an ensemble of classifiers more rapidly than can FS on all the examined datasets; the same holds in the comparison between SM and FE on 92% of the examined datasets. The extended investigation in Section 6.4.2.1 illustrates that the proposed SM method surpasses FE in cases wherein large datasets that required partitioning into numerous data subsets is used.

CHAPTER 7

CONCLUSIONS AND FURTHER WORK

7.1 Introduction

This chapter begins with a summary of the entire research and presents the conclusions drawn from the findings. It concludes with a list of the research limitations and suggested directions for future investigations.

7.2 Summary

This study aimed at developing methods that facilitate a resolution to the problem of mining big datasets on the basis of ensemble methods. To realise this aim and its associated objectives a framework for an ensemble of classifiers has been proposed and implemented to carry out three sets of experiments. This section summarises the works that have been conducted in the entire study to answers the predefined research questions in Section 1.4 as follows:

Q 1: Does the size of partitioned data subsets affect ensemble accuracy?

As divide-and-conquer strategy is adopted to handle the problem of mining large dataset, we started our research by investigating experimentally whether the size of the partitioned data subset (Rt) has any relationship with ensemble accuracy $acc(\phi)$. In Chapter 4, we examined the relationship between Rt and $acc(\phi)$ in

thirteen datasets. The experimental results presented in section 4.4 confirm that Rt can have some influence in $acc(\phi)$ in several patterns. However, the pattern of the influence of Rt on $acc(\phi)$ was not regular in all the examined datasets. Thus there was a need to categorize the pattern of this relationship which it was our second research question.

Q 2: Is it possible to categorise the patterns that underlie the relationship between the size of partitioned data subsets and ensemble accuracy?

Three patterns out of the four conceptually hypothesised patterns for the relationship between Rt and $acc(\phi)$ are identified in Chapter (4) and depicted by Figure 4.17. These three patterns are :

(a) The first identified pattern represent a monotonic relationship up to certain Rt (in most cases this Rt between 10% and 30% of the size of Training dataset Tr) after this certain Rt the improvement in the $acc(\phi)$ become negligible. This pattern is illustrated in Figure 4.17 as P2. Nine of the thirteen examined datasets were found to belong to this pattern.

(b) The second identified pattern is P3, which represents no relationship or very weak linear relationship between Rt and $acc(\phi)$. Three out of the thirteen evaluated datasets were found to belong to this category.

(c) The last identified category is (P4), which represents a monotonic relationship between Rt and $acc(\phi)$ up to a particular Rt; after that Rt the $acc(\phi)$ starts to descend . Poker dataset was the only dataset that found to have this pattern of relationship.

Having identified these three patterns do exit in big data, the next logical task is to develop an algorithm to do so efficiently and effectively.

Q 3: How can the aforementioned patterns be identified for big datasets?

In Chapter 5, a method to identify the relationship between acc(ϕ) and Rt

is proposed. The aim of the proposed method is to facilitate an improved understanding of the relationship between $acc(\phi)$ and Rt and help to decide when using as much data as possible in the construction of an ensemble of classifiers is appropriate and when it is not. The proposed method is also helpful in deciding the best data subset size (for a specified available memory capacity) of a given dataset to use when an ensemble is being constructed

The experimental results discussed in Section 5.5 show that the proposed method can detect the relational pattern in a few iterations. Also, the proposed method was able to stop at Rt, where no more enhancement in acc(ϕ) occurs with the increase in Rt. The Rt where the algorithm is terminated can be considered as the best Rt to be use when partitioning the examined dataset.

Q 4: How are model section techniques used to build an efficient and effective ensemble of classifiers when dealing with a big dataset?

When a large training dataset is divided into a big number of subsets, the process of creating a model (classifier) from all the available data subsets becomes a very expensive task in term of time and resources. In Chapter 6, the selective modelling (SM) method is proposed to reduce the time and resources required during the process of creating models pool (MP). The proposed SM is an alternative method to the traditional model selection methods to build an efficient ensemble of classifiers when a large number of datasets are required. The SM aims to save time and resources in building an ensemble of classifiers from a large dataset. This aim advantage is achieved by the iterative combination of MP creation and selection. The main idea in SM is to build what is needed. Instead of creating models from all the available datasets in a data pool, the SM algorithm iteratively generates models to reduce the number of generating unused models.

The experimental results in Section 6.4 show that SM can outperform the ensemble constructed from all the available models in the 13 examined datasets. The extended investigation in Section 6.4.2.1 illustrates that the proposed SM method surpasses other compared methods, especially in the cases where large datasets must be partitioned into numerous data subsets.

7.3 Research Novelty

The novelty of our research can be summarised as follows:

- This research empirically investigated the relationship between ensemble accuracy and the size of partitioned data subsets when dealing with big datasets. It also categorizes the patterns exhibited by this relationship. Most the available studies, such as [3, 55, 92, 17], focus on the effects of sampling ratio on small datasets, whereas the current research probes into this matter in relation to huge datasets and analyses the influence of partitioned data subsets. To the best of our knowledge, no other study has classified the patterns that are manifested by the relationship between ensemble accuracy and partitioned data subset size. A pilot study (see appendix B.1) for identifying and categorising these patterns was published in the 2013 IEEE International Conference on Big Data [24].

- Developed a novel an algorithm that can identify the patterns of the relationship between ensemble accuracy and partitioned data subset size. This algorithm adds to the novelty of the study given that no other research has attempted to categorise the patterns reflected by the ensemble accuracy–data subset size relationship. This algorithm was published in the The 9th IEEE International Conference on Big Data Science and Engineering (IEEE BigDataSE-15) [25]. A copy from the published paper is available in appendix B.2.

- Also a selective modelling method is developed that handles the problem of dealing with large datasets. This method is an efficient alternative to traditional

ensemble selection methods because it is applicable to very large datasets. Its novelty stems from the fact that it does not require the construction of a model pool that contains an entire set of possible models, which as is essential in conventional model selection.

7.4 Conclusions

The main findings of this research can be summarised as follows:

- **Maximising the size of the partitioned data subset does not necessarily lead to better results when analysing large datasets**

  The main finding from the experimental study presented in Chapter 4, is there are three patterns of relationships between $acc(\phi)$ and Rt. These three identified patterns show that maximising Rt (as much as possible) is not necessary in most of the examined datasets to produce an accurate and efficient ensemble of classifiers. For example, all nine datasets that belong to pattern P2 show that $acc(\phi)$ is no longer improved after certain level of Rt, that mostly between 10% and 30% of training data Tr. Thus, constructing an ensemble of classifiers with the use of the maximum possible Rt will not produce a more accurate and more efficient ensemble than that created with the use of that certain Rt. Furthermore, datasets that belong to P3 give good support to our finding, where $acc(\phi)$ does not gain any enhancement with the increase in Rt; which means the maximum Rt is not necessary at all for the datasets that belong to this pattern.

  This finding shows the importance of the proposed method in Chapter 5, that identified the pattern of the relationship between $acc(\phi)$ and Rt.

- **Identifying the pattern of the learning relationship between $acc(\phi)$ and Rt can help to determine the best sub set size when partitioning a big training dataset.**

The proposed method (in chapter 4) to identify the pattern of relationship between $acc(\phi)$ and Rt can be used to determined the best Rt when dividing a big dataset. This advantage is due to fact that the proposed method is iteratively examined influence of increasing Rt on $acc(\phi)$ and it is terminated at a Rt where no more improvement on $acc(\phi)$ is expected to occur. This Rt where the algorithm has stopped can be considered as the best partitioning size of the data subset for the examined dataset.

- **Traditional model selection is not efficient when dealing with large datasets.** The large number of models, which is a consequence of dividing a large training dataset to a large number of data subsets, represents a challenge in the application of model selection techniques. Traditional model selection techniques depend on the principle of selecting models from a pool of models (MP) containing all possible models. In the case of a large number of data subsets, creating MP will be an expensive task in terms of time and resources. Moreover, creating a MP might result in wastage in time and resource because only some of the generated models are selected by the used model selection method to form the final ensemble predication.

7.5 Limitations and Further Work

The aim and the objectives of this study were achieved by carrying out series of experiments illustrated in chapters 4,5 and 6. However this study has some aspects that can still be improved as follows:

- All the experiments conducted in this research employ a homogeneous ensemble of classifiers. It will be interesting to know if the proposed methods can produce similar oe even better results in the case of using heterogeneous ensembles by using different types of classifiers.

- Decision tree was as a base learner for our ensemble, more other algorithms are could be explored.

- Although the majority voting was adopted in our ensemble as a decision fusion strategy for its simplicity, the effect of other strategies could be investigated in further research.

- In chapter 4, the 13 examined datasets were placed into 3 categories depending on the pattern of the relationship between $acc(\phi)$ and Rt for each dataset. Further investigation is required in the future to understand the reasons behind the variation in the patterns of the relationship between $acc(\phi)$ and Rt for different datasets.

- In chapter 4, the smallest Rt used in the study of the relationship between Rt and $acc(\phi)$ was equal to 1%, it will be interesting in further work to reduce the Rt to a very small value, such as 0.1%, 0.01% or 0.001%, especially for datasets that belong to the P3 pattern, in order to understand how the relation between $acc(\phi)$ and Rt will be affected in the case of using a very small Rt.

- In the two proposed methods in chapter 5 and 6, the input parameters for the proposed algorithms were chosen after trying multiple values for these inputs to find out what the best values for these input parameters. Further work needs to be done to design an adaptive rule for choosing the best input values for each examined dataset.

REFERENCES

[HAd] Hadoop, howpublished = https://hadoop.apache.org/, note = Accessed: 2015-11-30.

[UEA] High Performance Computing Cluster supported by the Research and Specialist Computing Support service at the University of East Anglia. http:/http://rscs.uea.ac.uk/high-performance-computing/using-grace. Accessed: 2015-12-20.

[Spa] Spark, howpublished = http://spark.apache.org/. Accessed: 2015-11-30.

[1] Alcalá, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., and Herrera, F. (2010). Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, 17(255-287):11.

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

[3] Basilico, J. D., Munson, M. A., Kolda, T. G., Dixon, K. R., and Kegelmeyer, W. P. (2011). Comet: A recipe for learning and using large ensembles on massive data. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 41–50. IEEE.

[4] Bauer, E. and Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine learning*, 36(1-2):105–139.

[5] Bhatnagar, V., Bhardwaj, M., Sharma, S., and Haroon, S. (2014). Accuracy–diversity based pruning of classifier ensembles. *Progress in Artificial Intelligence*, 2(2-3):97–111.

[6] Bian, S. and Wang, W. (2006). Investigation on diversity in homogeneous and heterogeneous ensembles. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pages 3078–3085. IEEE.

[7] Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140.

[8] Breiman, L. (1999). Pasting small votes for classification in large databases and on-line. *Machine Learning*, 36(1-2):85–103.

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

[10] Britto, A. S., Sabourin, R., and Oliveira, L. E. (2014). Dynamic selection of classifiers—a comprehensive review. *Pattern Recognition*, 47(11):3665–3680.

[11] Brown, G., Wyatt, J., Harris, R., and Yao, X. (2005). Diversity creation methods: a survey and categorisation. *Information Fusion*, 6(1):5–20.

[12] Caruana, R., Munson, A., and Niculescu-Mizil, A. (2006). Getting the most out of ensemble selection. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 828–833. IEEE.

[13] Caruana, R., Niculescu-Mizil, A., Crew, G., and Ksikes, A. (2004). Ensemble selection from libraries of models. In *Proceedings of the twenty-first international conference on Machine learning*, page 18. ACM.

[14] Chan, P. and Stolfo, S. (1993). Toward parallel and distributed learning by meta-learning. In *AAAI workshop in Knowledge Discovery in Databases*, pages 227–240.

[15] Chang, C. and Lin, C. (2012). Libsvm – a library for support vector machines. http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/. Accessed: 02/06/2012.

[16] Chawla, N., Moore Jr, T. E., Bowyer, K. W., Hall, L. O., Springer, C., and Kegelmeyer, P. (2001). Bagging is a small-data-set phenomenon. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages II–II. IEEE.

[17] Chawla, N. V., Hall, L. O., Bowyer, K. W., and Kegelmeyer, W. P. (2004). Learning ensembles from bites: A scalable and accurate approach. *The Journal of Machine Learning Research*, 5:421–451.

[18] Chawla, N. V., Lazarevic, A., Hall, L. O., and Bowyer, K. W. (2003). Smoteboost: Improving prediction of the minority class in boosting. In *Knowledge Discovery in Databases: PKDD 2003*, pages 107–119. Springer.

[19] Cohen, L., Manion, L., and Morrison, K. (2000). Action research. *Research methods in education (5th ed.), eds. L. Cohen, L. Manion, and K. Morrison. London: Routledge-Falmer*.

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

[21] Dietterich, T. G. (2000). Ensemble methods in machine learning. In *Multiple classifier systems*, pages 1–15. Springer.

[22] Estruch, V., Ferri, C., Hernández-Orallo, J., and Ramírez-Quintana, M. J. (2004). Bagging decision multi-trees. In *Multiple Classifier Systems*, pages 41–51. Springer.

[23] Faisal, Z. and Hirose, H. (2010). A comparative study on the performance of several ensemble methods with low subsampling ratio. In *Intelligent Information and Database Systems*, pages 320–329. Springer.

[24] Farrash, M. and Wang, W. (2013). How data partitioning strategies and subset size influence the performance of an ensemble? In *Big Data, 2013 IEEE International Conference on*, pages 42–49.

[25] Farrash, M. and Wang, W. (2015). An algorithm for identifying the learning patterns in big data. In *Trustcom/BigDataSE/ISPA, 2015 IEEE*, volume 2, pages 48–55.

[26] Fiolet, V. and Toursel, B. (2005). Distributed data mining. *Scalable Computing: Practice and Experiences*, 6(1):99–109.

[27] Fisher, D., DeLine, R., Czerwinski, M., and Drucker, S. (2012). Interactions with big data analytics. *interactions*, 19(3):50–59.

[28] Freitas, A. and Lavington, S. (1998). *Mining very large databases with parallel processing*. Springer.

[29] Freund, Y., Schapire, R., and Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612.

[30] Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.

[31] Fu, B., Wang, Z., Pan, R., Xu, G., and Dolog, P. (2013). An integrated pruning criterion for ensemble learning based on classification accuracy and diversity. In *7th International Conference on Knowledge Management in Organizations: Service and Cloud Computing*, pages 47–58. Springer.

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

[33] Grove, A. and Schuurmans, D. (1998). Boosting in the limit: Maximizing the margin of learned ensembles. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, pages 692–699. JOHN WILEY & SONS LTD.

[34] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

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

[36] Hothorn, T. and Lausen, B. (2003). Double-bagging: Combining classifiers by bootstrap aggregation. *Pattern Recognition*, 36(6):1303–1309.

[37] Ian H. Witten, E. F. (2005). *Data mining Practical machine learning tools and techniques*. Elsevier.

[38] IBM (2014). What is big data?

[39] Japkowicz, N. and Shah, M. (2011). *Evaluating learning algorithms: a classification perspective*. Cambridge University Press.

[40] Jia, J., Xiao, X., Liu, B., and Jiao, L. (2011). Bagging-based spectral clustering ensemble selection. *Pattern Recognition Letters*, 32(10):1456–1467.

[41] Jurek, A., Bi, Y., Wu, S., and Nugent, C. (2014). A survey of commonly used ensemble-based classification techniques. *The Knowledge Engineering Review*, 29(05):551–581.

[42] Karau, H., Konwinski, A., Wendell, P., and Zaharia, M. (2015). *Learning Spark: Lightning-Fast Big Data Analysis*. " O'Reilly Media, Inc.".

[43] Ko, A. H.-R., Sabourin, R., and de Souza Britto Jr, A. (2009). Compound diversity functions for ensemble selection. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(04):659–686.

[44] Kumar, G. and Kumar, K. (2012). The use of artificial-intelligence-based ensembles for intrusion detection: a review. *Applied Computational Intelligence and Soft Computing*, 2012:21.

[45] Kuncheva, L., Rodríguez, J. J., Plumpton, C. O., Linden, D. E., Johnston, S. J., et al. (2010). Random subspace ensembles for fmri classification. *Medical Imaging, IEEE Transactions on*, 29(2):531–542.

[46] Kuncheva, L. and Whitaker, C. (2001). Ten measures of diversity in classifier ensembles: limits for two classifiers. In *COLLOQUIUM DIGEST-IEE*, pages 16–25. IEE; 1999.

[47] Kuncheva, L. I. (2004). *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons.

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

[49] Lam, L. and Suen, C. Y. (1997). Application of majority voting to pattern recognition: an analysis of its behavior and performance. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 27(5):553–568.

[50] Laney, D. (2001). 3d data management: Controlling data volume, velocity and variety. *META Group Research Note*, 6:70.

[51] Lazarevic, A. and Obradovic, Z. (2002). Boosting algorithms for parallel and distributed learning. *Distributed and parallel databases*, 11(2):203–229.

[52] Li, G.-z., Yang, J., Kong, A., and Chen, N.-y. (2004). Clustering algorithm based selective ensemble. *Journal of Fudan University*, 5:003.

[53] Liang, G., Zhu, X., and Zhang, C. (2011). An empirical study of bagging predictors for different learning algorithms.

[54] Lichman, M. (2013). UCI machine learning repository.

[55] Louppe, G. and Geurts, P. (2012). Ensembles on random patches. In *Machine Learning and Knowledge Discovery in Databases*, pages 346–361. Springer.

[56] Lu, Z., Wu, X., Zhu, X., and Bongard, J. (2010). Ensemble pruning via individual contribution ordering. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 871–880. ACM.

[57] Lusa, L. et al. (2015). Boosting for high-dimensional two-class prediction. *BMC bioinformatics*, 16(1):1.

[58] Maimon, O. and Rokach, L. (2005). *Data mining and knowledge discovery handbook*. Springer-Verlag New York Inc.

[59] Makhtar, M., Yang, L., Neagu, D., and Ridley, M. (2012). Optimisation of classifier ensemble for predictive toxicology applications. In *Computer Modelling and Simulation (UKSim), 2012 UKSim 14th International Conference on*, pages 236–241. IEEE.

[60] Martinez-Muoz, G., Hernández-Lobato, D., and Suarez, A. (2009). An analysis of ensemble pruning techniques based on ordered aggregation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(2):245–259.

[61] Meng, Y., Yu, Y., Cupples, L., Farrer, L., and Lunetta, K. (2009). Performance of random forest when SNPs are in linkage disequilibrium. *BMC bioinformatics*, 10:78.

[62] Merler, S., Caprile, B., and Furlanello, C. (2007). Parallelizing adaboost by weights dynamics. *Computational statistics & data analysis*, 51(5):2487–2498.

[63] Nandakumar, A. and Yambem, N. (2014). A survey on data mining algorithms on apache hadoop platform. *International Journal of Emerging Technology and Advanced Engineering*, 4(1):563–565.

[64] Nanni, L., Lumini, A., Gupta, D., and Garg, A. (2012). Identifying bacterial virulent proteins by fusing a set of classifiers based on variants of chou's pseudo amino acid composition and on evolutionary information. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 9(2):467–475.

[65] Oates, T. and Jensen, D. (1997). The effect of training set size on decision tree complexity. In *Proceedings of the 14th International Conference on Machine Learning*, pages 254–262.

[66] Oates, Tim and Jensen, David (1998). Large datasets lead to overly complex models: an explanation and a solution. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 294–298.

[67] Oliveira, L., Nunes, U., and Peixoto, P. (2010). On exploration of classifier ensemble synergism in pedestrian detection. *Intelligent Transportation Systems, IEEE Transactions on*, 11(1):16–27.

[68] Opitz, D. and Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11(1):169–198.

[69] Partalas, I., Tsoumakas, G., and Vlahavas, I. P. (2008). Focused ensemble selection: A diversity-based method for greedy ensemble selection. In *ECAI*, pages 117–121.

[70] Partridge, D. and Krzanowski, W. (1997). Software diversity: practical statistics for its measurement and exploitation. *Information and software technology*, 39(10):707–717.

[71] Peng, Y., Kou, G., Shi, Y., and Chen, Z. (2005). Using Optimization-Based Classification Method for Massive Datasets. *AMCIS 2005 Proceedings*, page 110.

[72] Qiang, F., Shang-Xu, H., and Sheng-Ying, Z. (2005). Clustering-based selective neural network ensemble. *Journal of Zhejiang University Science A*, 6(5):387–392.

[73] Quinlan, J. R. (1993). *C4. 5: programs for machine learning*. Elsevier.

[74] Ripley, B. D. (2007). *Pattern recognition and neural networks*. Cambridge university press.

[75] Robnik-Šikonja, M. (2004). Improving random forests. In *Machine Learning: ECML 2004*, pages 359–370. Springer.

[76] Ruta, D. and Gabrys, B. (2001). Analysis of the correlation between majority voting error and the diversity measures in multiple classifier systems. In *Soft Computing and Intelligent Systems for Industry: Proceedings and Scientific Program: : Fourth International ICSC Symposium*, page 50. ICSC;2001.

[77] Ruta, D. and Gabrys, B. (2005). Classifier selection for majority voting. *Information fusion*, 6(1):63–81.

[78] Sabzevari, M., Martınez-Munoz, G., and Suárez, A. (2014). Improving the robustness of bagging with reduced sampling size. In *Proceedings of the European Symposium on Artificial Neural Networks, ESANN*, pages 667–682.

[79] Santana, A., Soares, R. G., Canuto, A. M., and de Souto, M. C. (2006). A dynamic classifier selection method to build ensembles using accuracy and diversity. In *Neural Networks, 2006. SBRN'06. Ninth Brazilian Symposium on*, pages 36–41. IEEE.

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

[81] Shim, K. (2012). Mapreduce algorithms for big data analysis. *Proceedings of the VLDB Endowment*, 5(12):2016–2017.

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

[83] Skurichina, M. and Duin, R. (2002). Bagging, boosting and the random subspace method for linear classifiers. *Pattern Analysis & Applications*, 5(2):121–135.

[84] Soda, P. and Iannello, G. (2009). Aggregation of classifiers for staining pattern recognition in antinuclear autoantibodies analysis. *Information Technology in Biomedicine, IEEE Transactions on*, 13(3):322–329.

[85] Song, Y., Zheng, Y.-T., Tang, S., Zhou, X., Zhang, Y., Lin, S., and Chua, T.-S. (2011). Localized multiple kernel learning for realistic human action recognition in videos. *Circuits and Systems for Video Technology, IEEE Transactions on*, 21(9):1193–1202.

[86] Sun, Q. and Pfahringer, B. (2011). Bagging ensemble selection. In *AI 2011: Advances in Artificial Intelligence*, pages 251–260. Springer.

[87] Tan, P.-N., Steinbach, M., and Kumar, V. (2005). *Introduction to Data Mining*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[88] Troester, M. (2012). Big data meets big data analytics. *Cary, NC: SAS Institute Inc*.

[89] Tsai, C.-W., Lai, C.-F., Chao, H.-C., and Vasilakos, A. V. (2015). Big data analytics: a survey. *Journal of Big Data*, 2(1):1–32.

[90] TSANG, I. (2009). Core vector machine. http://www.cs.ust.hk/~ivor/cvm.html.

[91] Tsang, I., Kwok, J., and Cheung, P. (2005a). Core vector machines: Fast SVM training on very large data sets. *Journal of Machine Learning Research*, 6(1):363–392.

[92] Tsang, I. W., Kocsor, A., and Kwok, J. T. (2006). Diversified svm ensembles for large data sets. In *Machine Learning: ECML 2006*, pages 792–800. Springer.

[93] Tsang, I. W., Kwok, J. T., and Cheung, P.-M. (2005b). Core vector machines: Fast svm training on very large data sets. In *Journal of Machine Learning Research*, pages 363–392.

[94] Tsoumakas, G., Partalas, I., and Vlahavas, I. (2008). A taxonomy and short review of ensemble selection. In *ECAI 2008, workshop on supervised and unsupervised ensemble methods and their applications*, pages 41–46.

[95] Tsoumakas, G. and Vlahavas, I. (2009). *Distributed Data Mining*. Information Science Reference-Imprint of: IGI Publishing Hershey, PA.

[96] Turner, V., Gantz, J. F., Reinsel, D., and Minton, S. (2014). The digital universe of opportunities: Rich data and the increasing value of the internet of things. *International Data Corporation, White Paper, IDC_1672*.

[97] Tuv, E. (2006). Ensemble learning. *STUDIES IN FUZZINESS AND SOFT COMPUTING*, 207:187.

[98] Wang, W. (2006). *diversity and accuracy of data mining ensemble*. World Scientific.

[99] Wang, W. (2008). Some fundamental issues in ensemble methods. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 2243–2250. IEEE.

[100] Wang, Y. and De Lin, C. (2007). Learning by bagging and adaboost based on support vector machine. In *Industrial Informatics, 2007 5th IEEE International Conference on*, volume 2, pages 663–668. IEEE.

[101] Waske, B. and Benediktsson, J. A. (2007). Fusion of support vector machines for classification of multisensor data. *Geoscience and Remote Sensing, IEEE Transactions on*, 45(12):3858–3866.

[102] Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics bulletin*, pages 80–83.

[103] Wolpert, D. H. (1992). Stacked generalization. *Neural networks*, 5(2):241–259.

[104] "Wu, C., Buyya, R., and Ramamohanarao, K. (2016). "big data analytics= machine learning+ cloud computing". In *"Big Data: Principles and Paradigms"*, chapter 1. "Morgan Kaufmann".

[105] Wu, T.-F., Lin, C.-J., and Weng, R. C. (2004). Probability estimates for multi-class classification by pairwise coupling. *The Journal of Machine Learning Research*, 5:975–1005.

[106] Xu, L., Krzyżak, A., and Suen, C. Y. (1992). Methods of combining multiple classifiers and their applications to handwriting recognition. *Systems, man and cybernetics, IEEE transactions on*, 22(3):418–435.

[107] Yu, Z., Deng, Z., Wong, H.-S., and Tan, L. (2010). Identifying protein-kinase-specific phosphorylation sites based on the bagging–adaboost ensemble approach. *NanoBioscience, IEEE Transactions on*, 9(2):132–143.

[108] Yu, Z., Li, L., Liu, J., and Han, G. (2015). Hybrid adaptive classifier ensemble. *Cybernetics, IEEE Transactions on*, 45(2):177–190.

[109] Zaman, F. and Hirose, H. (2009). Effect of subsampling rate on subbagging and related ensembles of stable classifiers. In *Pattern Recognition and Machine Intelligence*, pages 44–49. Springer.

[110] Zeng, X., Wong, D. F., and Chao, L. S. (2014). Constructing better classifier ensemble based on weighted accuracy and diversity measure. *The Scientific World Journal*, 2014.

[111] Zhang, B. (2013). Reliable classification of vehicle types based on cascade classifier ensembles. *Intelligent Transportation Systems, IEEE Transactions on*, 14(1):322–332.

[112] Zhang, C.-X. and Duin, R. P. (2011). An experimental study of one-and two-level classifier fusion for different sample sizes. *Pattern Recognition Letters*, 32(14):1756–1767.

[113] Zhang, C.-X., Zhang, J.-S., and Zhang, G.-Y. (2009). Using boosting to prune double-bagging ensembles. *Computational Statistics & Data Analysis*, 53(4):1218–1231.

[114] Zhang, H. and Wang, M. (2009). Search for the Smallest Random Forest. *Statistics and Its Interface*, 2:381–388.

[115] Zhi-Gang, G. and Nan-Feng, X. (2009). A new ensemble learning algorithm based on improved k-means for training neural network ensembles. In *Intelligent Information Technology and Security Informatics, 2009. IITSI'09. Second International Symposium on*, pages 8–11. IEEE.

[116] Zhong, P. and Wang, R. (2007). A multiple conditional random fields ensemble model for urban area detection in remote sensing optical images. *Geoscience and Remote Sensing, IEEE Transactions on*, 45(12):3978–3988.

[117] Zhou, Z.-H. (2012). *Ensemble methods: foundations and algorithms*. CRC Press.

APPENDIX A

EXTENDED RESULTS FOR CHAPTER 4

A.1 Correlation test result

The detailed correlation analysis results is provided in Figure A.2.

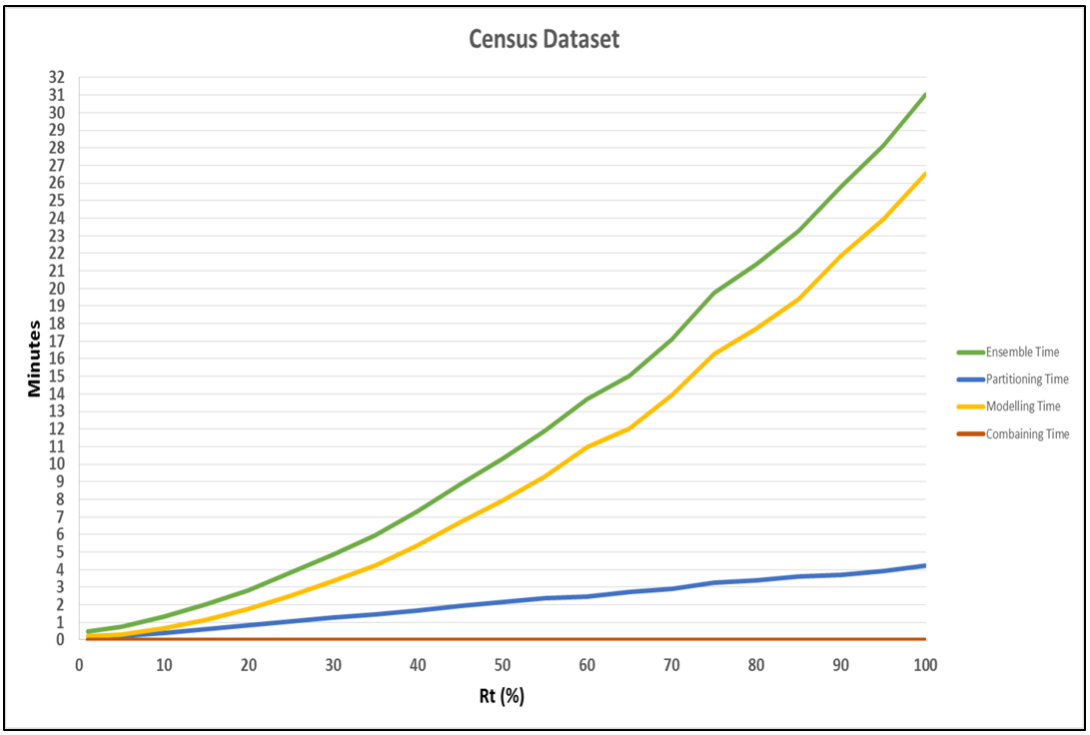
A.2 Effects of Rt on ensemble time

In section 4.4.3 the effects of increasing Rt on ensemble time on adult dataset was illustrated. In this section the charts the depict the influence of Rt on ensemble time for the other datasets are presented in Figures.

dataset	Rt range	crrolation method *	Validation		Testing	
			correlation coefficient	p-value	correlation coefficient	p-value
adult	Rt <= 10%	P	0.989	p=0.094	0.991	p=0.087
	Rt > 10%	P	0.944	p<0.0005	0.965	p<0.0005
	all Rt	S	0.994	p<0.0005	0.994	p<0.0005
	all Rt	P	0.753	p<0.0005	0.735	p<0.0005
census	Rt <= 10%	P	0.98	p=0.127	0.98	p=0.126
	Rt > 10%	P	0.982	p<0.0005	0.983	p<0.0005
	all Rt	S	1	p<0.0005	1	p<0.0005
	all Rt	P	0.822	p<0.0005	0.824	p<0.0005
connect4	Rt <= 10%	P	0.987	p=0.002	0.993	p=0.001
	Rt > 10%	P	0.961	p<0.0005	0.955	p<0.0005
	all Rt	S	0.998	p<0.0005	1	p<0.0005
	all Rt	P	0.841	p<0.0005	0.999	p<0.0005
cover type	Rt <= 20%	P	0.92	p=0.027	0.923	p=0.025
	Rt > 20%	P	0.935	p<0.0005	0.936	p<0.0005
	all Rt	S	1	p<0.0005	0.999	p<0.0005
	all Rt	P	0.74	p<0.0005	0.742	p<0.0005
FARS	all Rt	S	0.295	P=0.195	0.521	P=0.015
	all Rt	P	0.429	P=0.052	0.394	P=0.077
HIGGS	all Rt	S	1	p<0.0005	1	p<0.0005
	all Rt	P	0.914	P=0.011	0.915	P=0.011
ijcnn	Rt <= 20%	P	0.914	p=0.030	0.942	p=0.017
	Rt > 20%	P	0.951	p<0.0005	0.951	p<0.0005
	all Rt	S	1	p<0.0005	0.999	p<0.0005
	all Rt	P	0.834	p<0.0005	0.818	p<0.0005
kdd99	all Rt	S	0.994	p<0.0005	0.976	p<0.0005
	all Rt	P	0.761	P=0.028	0.744	P=0.034
poker	Rt <= 20%	P	0.921	p=0.026	0.921	p=0.026
	20% < Rt >= 40%	P	0.998	p=0.002	0.998	p=0.002
	40% < Rt > =70%	P	0.794	p=0.0.59	0.801	p=0.055
	Rt > 70	P	-0.766	p=0.74	-0.804	p=0.54
	all Rt	S	0.751	p<0.0005	0.751	p<0.0005
	all Rt	P	0.875	p<0.0005	0.875	p<0.0005
shuttle	Rt <= 5%	P	NA		NA	
	Rt > 5%	P	0.879	p<0.0005	0.904	p<0.0005
	all Rt	S	0.995	p<0.0005	0.993	p<0.0005
	all Rt	P	0.555	p<0.0005	0.558	p<0.0005
skin	Rt <= 5%	P	NA		NA	
	Rt > 5%	P	0.879	p<0.0005	0.904	p<0.0005
	all Rt	S	0.995	p<0.0005	0.993	p<0.0005
	all Rt	P	0.555	p=0.009	0.558	p=0.009
SUSY	all Rt	S	1	p<0.0005	1	p<0.0005
	all Rt	P	0.893	P=0.001	0.891	P=0.001
Web	Rt <= 15%	p	0.951	p=0.049	0.954	p=0.046
	Rt > 15%	p	0.929	p<0.0005	0.94	p<0.0005
	all Rt	s	1	p<0.0005	1	p<0.0005
	all Rt	p	0.666	p<0.0005	0.667	p<0.0005

* Where :(p) is pearson correlation & (s) detone to is spearman correlation

Fig. A.1 Results of correlation analysis for all the examined datasets using Pearson and Spearman correlation



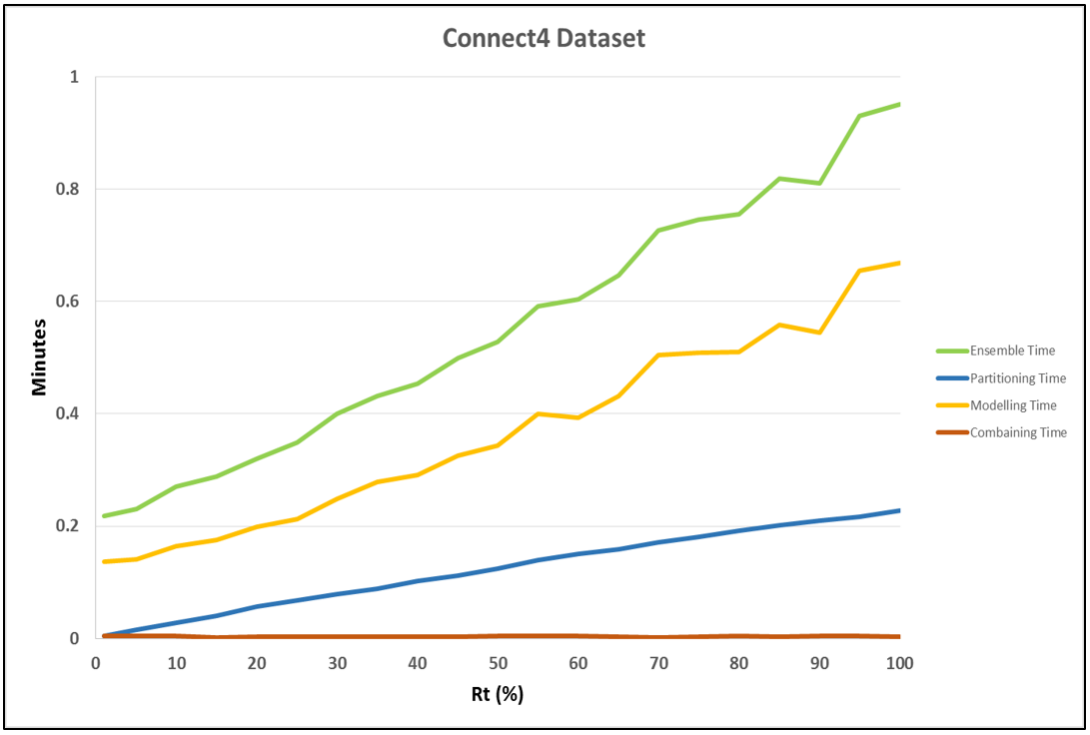


Fig. A.2 Relationship between Rt and ensemble time for the Census and Connect4 datasets

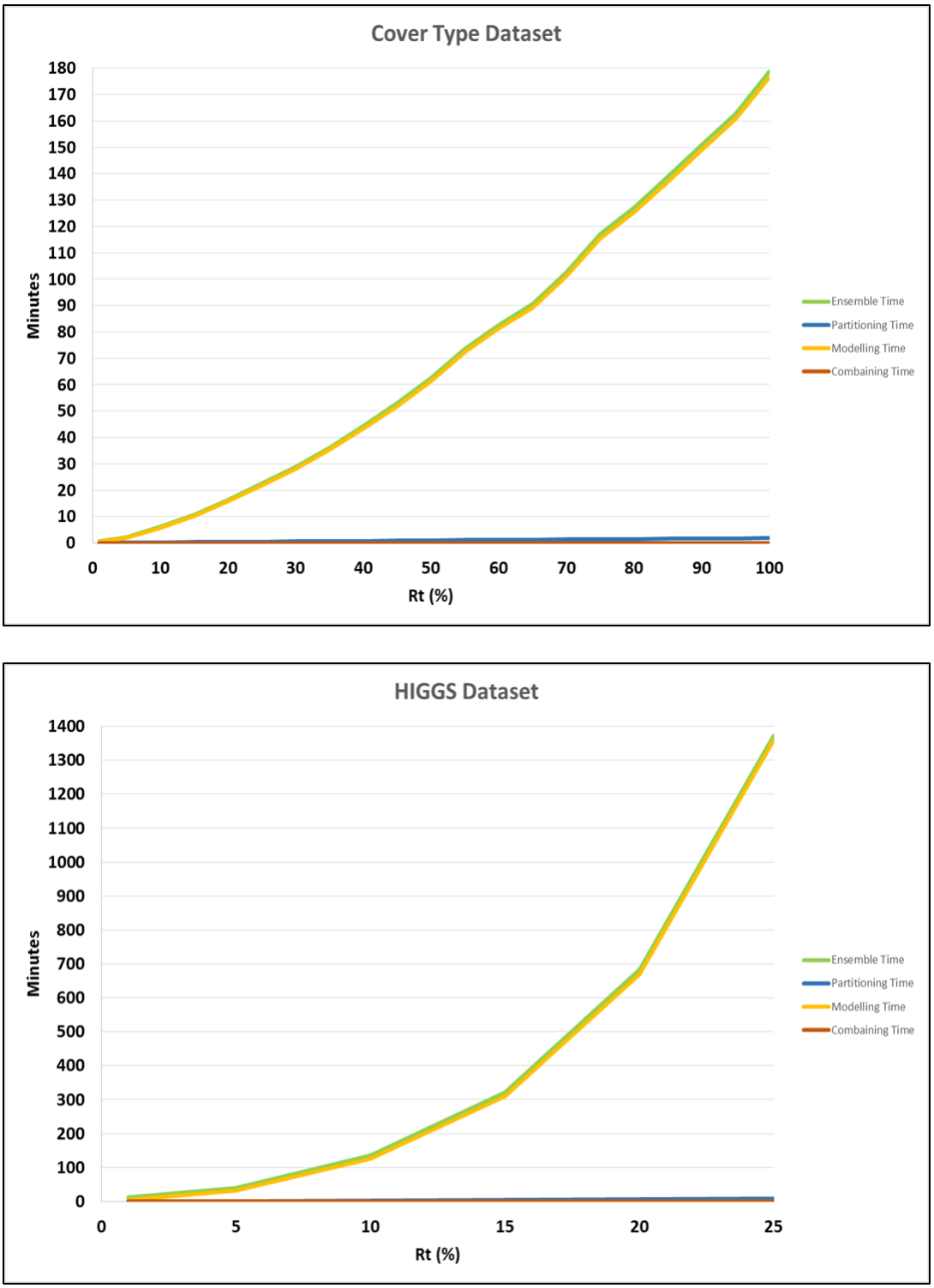


Fig. A.3 Relationship between Rt and ensemble time for the Cover type and HIGGS datasets

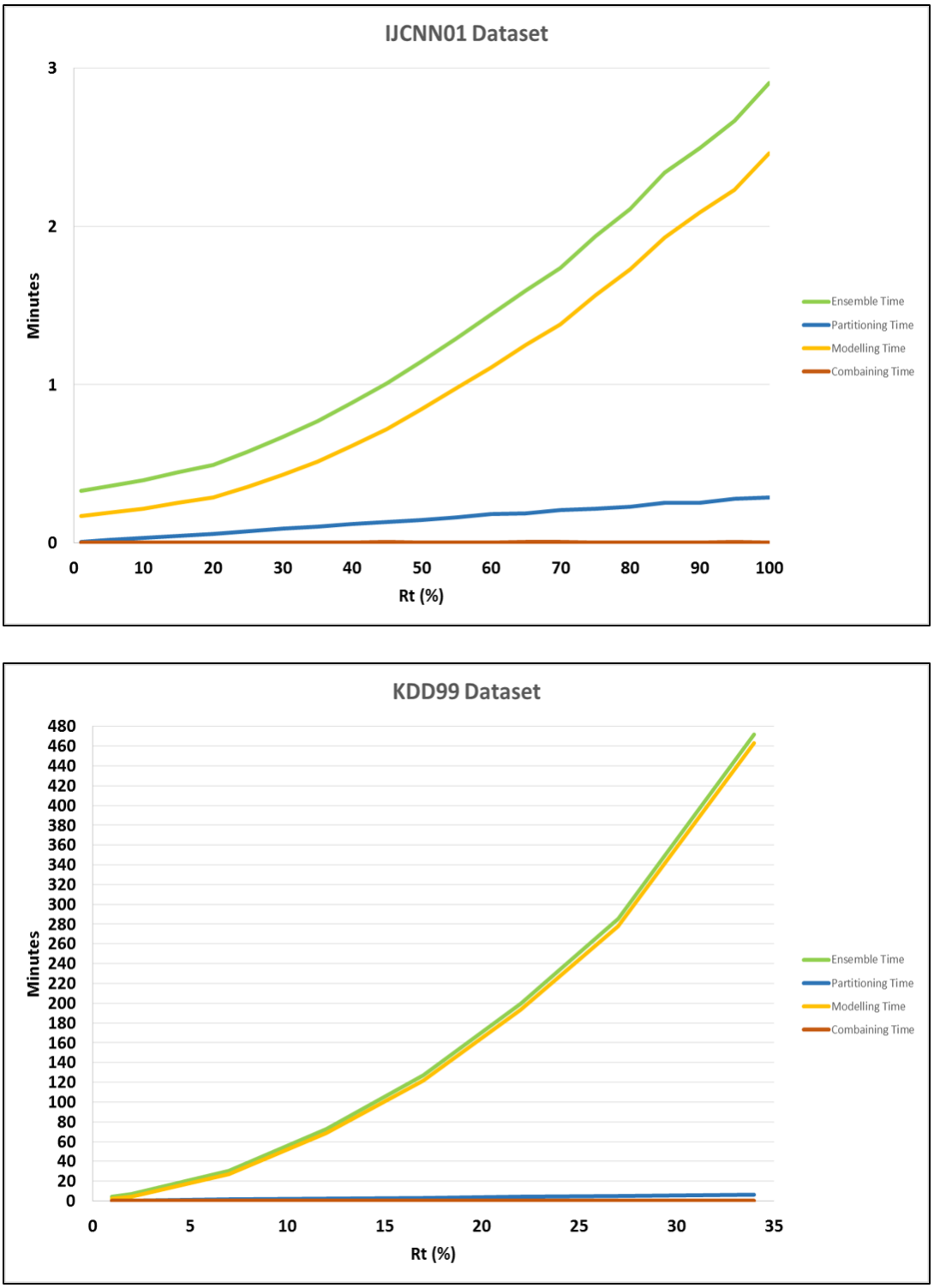


Fig. A.4 Relationship between Rt and ensemble time for the IJCNN01 and KDD99 datasets

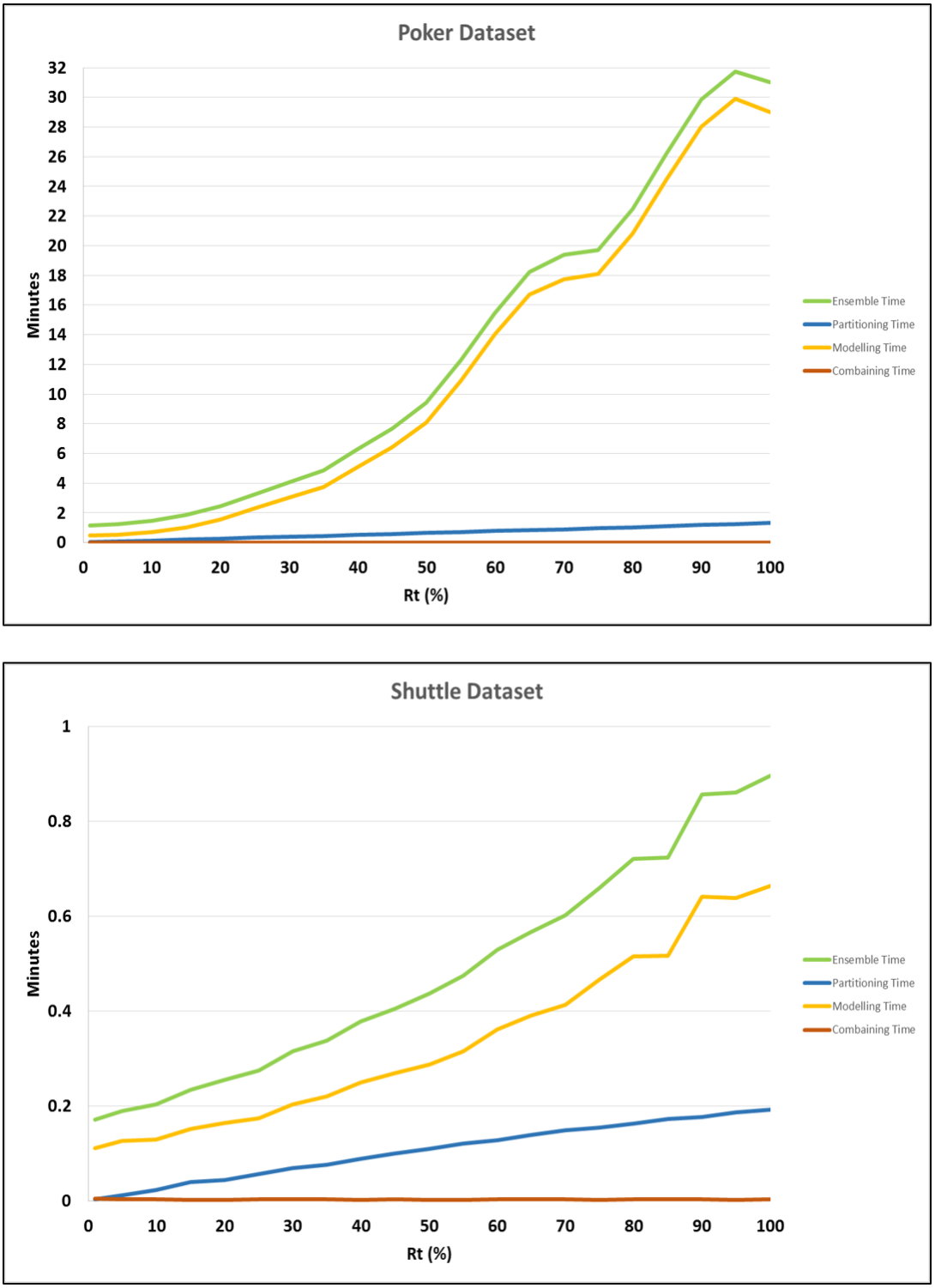


Fig. A.5 Relationship between Rt and ensemble time for the Poker and Shuttle datasets

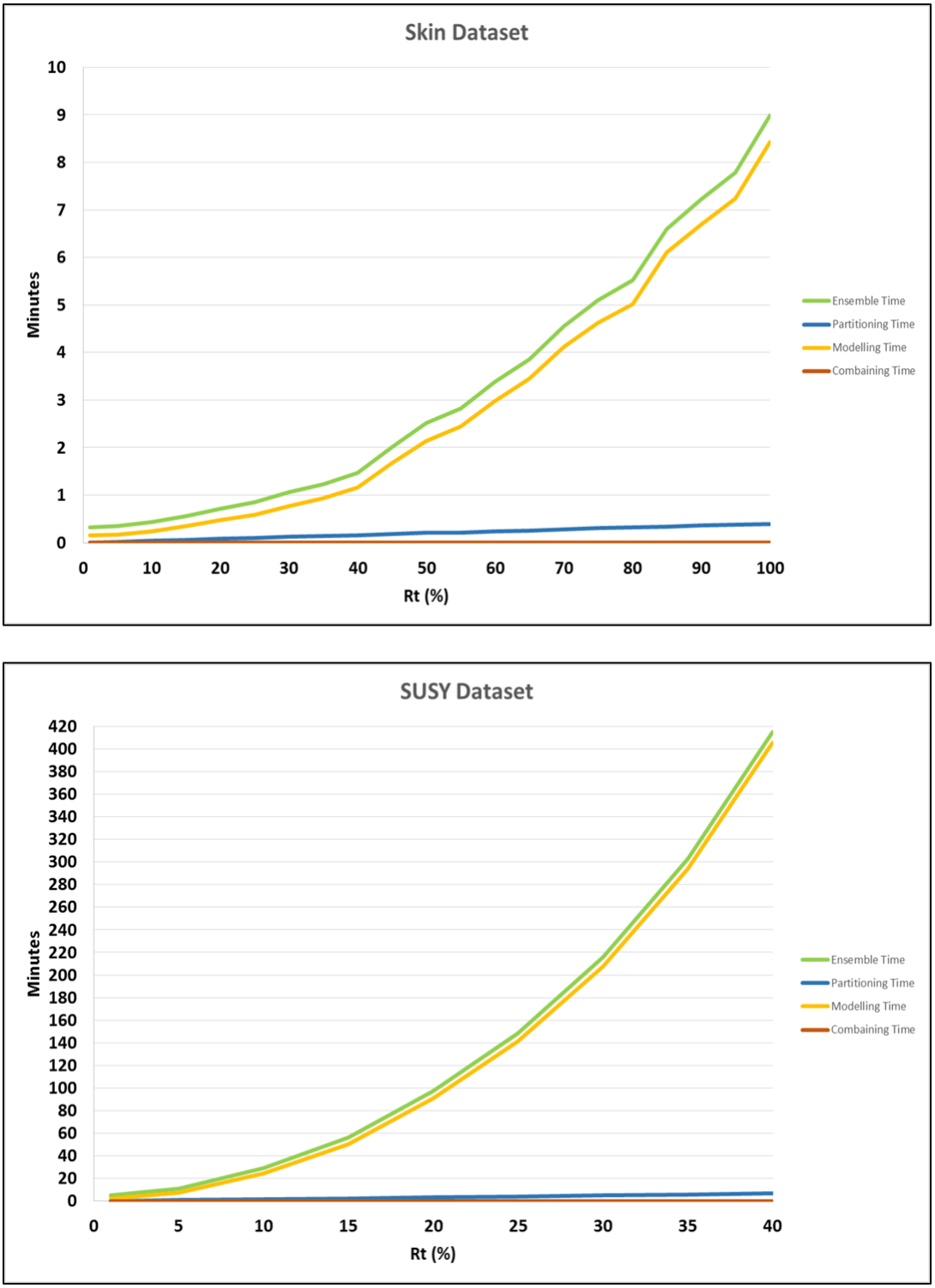


Fig. A.6 Relationship between Rt and ensemble time for the Skin and SUSY datasets

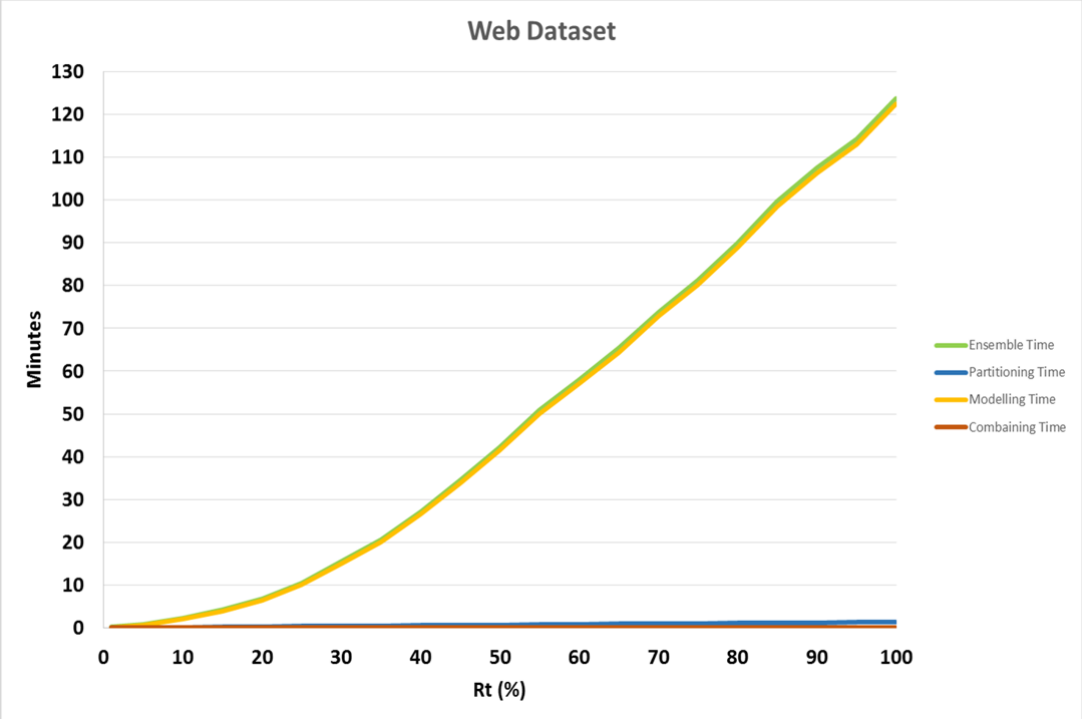


Fig. A.7 Relationship between Rt and ensemble time for the Census and Connect4 datasets

APPENDIX B

RESERACH PAPERS

B.1 How Data Partitioning Strategies and Subset Size Influence the Performance of an Ensemble?

2013 IEEE International Conference on Big Data

How Data Partitioning Strategies and Subset Size Influence the Performance of an Ensemble?

Majed Farrash
School of Computing Sciences
University of East Anglia
Email: M.Farrash@uea.ac.uk

Wenjia Wang
School of Computing Sciences
University of East Anglia
Email: Wenjia.Wang@uea.ac.uk

***Abstract*—When dealing with big data, "divide and conquer" is the most commonly used strategy in practice to partition a big dataset into such smaller subsets that each subset can be handled by a computer or a node of cluster or cloud computing systems. However, among many existing partitioning or sampling techniques, it is not clear which one is suitable and how the size of subset may affect the performance of further analysis. In this paper, after presenting a generic framework of ensemble approach for learning from big data, we focus our investigations on systematically evaluating the effect of partitioning strategies and subset size on ensemble performance. The experimental results have demonstrated that three investigated partitioning / sampling strategies behaved statistically similar but the subset size may affect the performance of the ensemble in very drastically different ways, which are grouped into three patterns, rather than just one default perception - the bigger the better.**

***Keywords*—Big data, partitioning, subset size, ensemble learning.**

I. Introduction

Nowadays, the quantity of data collected through various digital media and activities can quickly become so large that any existing computers alone cannot load the data into their memory all together for analysis. Therefore, a big dataset has to be divided into smaller and manageable subsets to overcome the limit of memory. However, although there exist many data partitioning and sampling strategies, there is no systematic study on them examining firstly how these techniques perform when dealing with big data, and secondly how the size of subset may influence the performance of machine learning and data mining methods. In practice, the second aspect is more important as machine learning methods may have quite different performance in dealing with smaller datasets and big datasets. Studies such as, distributed data mining [1], parallel data mining [2], incremental data mining [3] and ensemble methods [4] have shown the ability to deal with small datasets, but most of them do not show the same performance when dealing with single massive dataset. A very plausible reason is that the learning algorithms employed by these methods are developed under the assumption of having the whole data loaded into the main memory, and obviously this assumption cannot be upheld when dealing with large datasets that cannot be loaded into the main memory all together.

In this paper, we firstly propose a generic ensemble framework that consists of three phases for partitioning big data, generating models from the partitioned data subsets, and combining the generated models to produce a final answer. Then we focus our empirical investigations on the effect of diffident partitioning methods and subset size on ensemble accuracy in dealing with big datasets.

The rest of the paper is organized as follows. Section 2 briefly describes some related works. Section 3 presents an ensemble framework. The empirical investigation design and experimental results are presented in sections 4 and 5 respectively. Then the discussion is given in section 6 and conclusions in Section 7.

II. Related work

As previously mentioned, partitioning a big dataset into smaller subsets that are manageable is a basic strategy used by different data mining methods. Some researchers, such as Oates and Jensen [5] and [6], have shown that increasing the size of a training set does not greatly increase classification accuracy. Another research proposed by Hall and Chawla [7] showed some promising results in relatively small datasets. Their proposed method was to build a single decision system after learning is done independently on N disjoint subsets of data in parallel computing. In addition, Nittaya and Kittisak Kerdprasop [3] proposed an algorithm to partition the training dataset into manageable and learning effective subsets and tested the algorithm using the incremental data mining method. They found that the size of a subset should be between 10% and 50% of the whole data set. Moreover, Tsang et al. [4] proposed a SVM ensemble and sampling technique that was used to partition the dataset and showed that the use of orthogonal constraints in the SVM ensemble leads to better performance than bagging. An algorithm proposed by Patil and Bichkar [8] using a round robin partitioning method to classify large data sets, showed that the proposed algorithm achieved equivalent performance to the performance on a complete dataset. COMET is the MapReduce algorithm [9] for learning from large datasets. COMET builds multiple ensembles on distributed blocks of data and merges them into a mega ensemble. Their study showed that COMET compares favourably to subsampling Random Forests run serially on a single block of data. However their study fixed the subset size to 100K which was chosen by running IVoting algorithm for 1000 iteration. It is not clear whether different subset size affects the performance of the classification and in what ways if it does. So there is still need to investigate the effect of different subset sizes on the ensemble accuracy and find out how to choose the best subset size when partitioning a big dataset.

978-1-4799-1293-3/13/$31.00 ©2013 IEEE

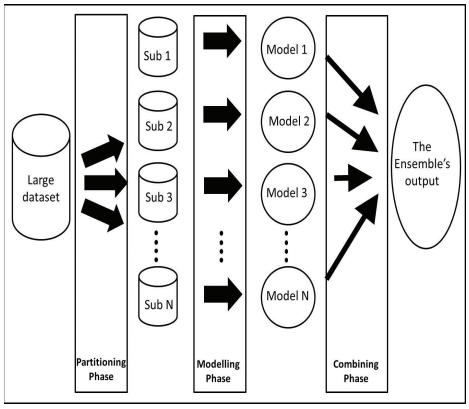


Fig. 1. Proposed Ensemble Framework

III. Proposed Ensemble Framework

In this research, an ensemble method framework is proposed as shown in Figure 1 to deal with very large datasets. It consists of three phases partitioning, modelling and combining which will be described in detail below. The operations of the framework are described in Algorithm 1.

Algorithm 1: Proposed Ensemble Algorithm

Inputs : Training Dataset Tr.
validation dataset V
Testing dataset Ts.
Partitioning Method P.
Learning algorithm L.
Fusion strategy F.
The relative size of a subset R_t as a percentage of the whole size of the Tr, number of subset $N = \left\lceil \frac{1}{R_t} \right\rceil$

Outputs: Ensemble Φ.
Ensemble accuracy $Acc(\Phi)$.
The mean accuracy of individual models $(\overline{Acc(m_i)})$.
Ensemble time (TM_ϕ) = Partitioning Time (TM_p)+ Learning Time (TM_ln) + Classification Time (TM_c).

Declare counter i=0
Call P (Tr , R_t) , and receive a set of N training subsets SUB= $\{t_0, t_1, t_2, ..., t_{N-1}\}$
while *i less than N* **do**
  Call L (t_i) and receive a base classifier m_i.
  add m_i to the ensemble Φ.
end
Evaluate Φ with V and Ts using F and receive $Acc(\Phi, V)$,$Acc(\Phi, Ts)$, $(\overline{Acc(m_i)})$, (TM_ϕ), (TM_p), (TM_{ln}) and (TM_c).
End

A. Partitioning Phase

In this phase a given large dataset will be divided into smaller subsets. However, there is no study to show which partitioning method should be used, and how they may affect the results of mining data as a whole.

In our experiments, three commonly used data partitioning methods, Sequential non-overlapping partitioning, Round robin partitioning and Sampling without replacement are examined.

1) **Sequential non-overlapping partitioning:** This method simply, divides the data instances of a training dataset (Tr) into a given number of subsets (N) in sequential manner. The size of each subset is equal to $\left\lceil \frac{|Tr|}{N} \right\rceil$. except the last subset which might take up to $\left\lceil \frac{|Tr|}{N} \right\rceil$ instances, where $|Tr|$ is the size of Tr . After calculating the size of subsets, data instances will be read from Tr sequentially, so the first $\left\lceil \frac{|Tr|}{N} \right\rceil$ instances goes to the first subset and so on until the end of the file of Tr. Algorithm 2 illustrate the procedure for this partitioning method.

Algorithm 2: Sequential non-overlapping partitioning

Input : Training Dataset Tr.
The relative size of a subset R_t as a percentage of the whole size of the Tr

Output: a Set of partitions, SUB $= \{t_0, t_1, t_2, ..., t_{N-1}\}$

Start:
Declare counter i =0, as counter to keep track of created file.
Calculate number of subset $N = \left\lceil \frac{1}{R_t} \right\rceil$.
Calculate number of instances in a subset $|t| = \left\lfloor \frac{|Tr|}{N} \right\rfloor$, where $|Tr|$ is number of instances in Tr.
while *not reach the end of Tr* **do**
  Create a subset t_i
  if $(i = (N-1))$ **then**
    Read the rest of the instances.
    Save them into subset t_i
  end
  Read the first $|t|$ unseen instances from Tr.
  save them into t_i
end
End

2) **Round robin partitioning method:** In round robin partitioning, the first instance of a training dataset goes to the first subset, the second to the second subset, and so on until the last subset. If the last subset is reached, then the method starts over from the first subset [10]. Algorithm 3 shows how to round robin partitioning method works.
3) **Sampling without replacement:** Sampling is a technique used to create subsets by selecting a random instance from a dataset. Random selection means that all the data instances have the same probability of being selected at any time. In sampling without replacement, if an instance was chosen to be

Algorithm 3: Round robin partitioning method.

Input : Training Dataset Tr.
The relative size of a subset R_t as a percentage of the whole size of the Tr

Output: A set of partitions, SUB $= \{t_0, t_1, t_2, ..., t_{N-1}\}$

Start:
Declare counter i =0, as counter to keep track of created file.
Declare counter j =0, as counter to keep track of training instances.
Calculate number of subset $N = \left\lceil \frac{1}{R_t} \right\rceil$.
Calculate $noInstances = |Tr|$, where $|Tr|$ is number of instances in Tr.
while $j < noInstances$ **do**
  reset counter i.
  while $i < N$ **do**
    x_j = Read data instance (j) from Tr.
    Save x_j into subset t_i.
  end
end
End

Algorithm 4: Sampling without replacement

Input : Training Dataset Tr.
The relative size of a subset R_t as a percentage of the whole size of the Tr

Output: a Set of partitions, SUB $= \{t_0, t_1, t_2, ..., t_{N-1}\}$

Start:
Declare counter i =0, as counter to keep track of created file.
Declare $availableSubSets$, as array that contains pointers to available subsets.
Calculate number of subset $N = \left\lceil \frac{1}{R_t} \right\rceil$.
Calculate number of instances in a subset $|t| = \left\lceil \frac{|Tr|}{N} \right\rceil$, where $|Tr|$ is number of instances in Tr.
while $i < N$ **do**
  create subset t_i
  size of t_i, $|t_i| = 0$
  add t_i into $availableSubSets$
end
while *not reach the end of Tr* **do**
  x = Read data instance from Tr.
  select a random subset t form $availableSubSets$
  save x into subset t
  $|t| = |t| + 1$
  if $|t| = N$ **then**
    Remove t from $availableSubSets$.
  end
end
End

a member of a subset, this instance cannot be chosen to be a member of any other subset during the sampling process. Algorithm 4 describes sampling without replacement method.

B. Modelling Phase

The modelling phase is the second stage of constructing an ensemble. In this phase a learning algorithm is chosen to learn from each subset of the data generated in the partitioning phase and then to create a data model. In principle, any learning algorithm can be used in this phase.

In this study a core vector machine (CVM) [11] is used as the base classifier because it performs very well with large datasets and its source code is available for free from the authors website for research purposes [12].

Core Vector Machine (CVM) Algorithm
Tsang et al. proposed CVM as an enhanced version of the standard SVM to accelerate the learning process. A standard support vector machine (SVM) has $O(|Tr|^3)$ time and $O(|Tr|^2)$ space complexities where $|Tr|$ is the size of the training dataset. While CVM has a linear time complexity in $|Tr|$ and space complexity independent of $|Tr|$, this enhancement was achieved by formulating "a kernel (including the soft-margin one-class and two-class SVMs) as the equivalent minimum enclosing ball (MEB) problem and then obtain approximately optimal solutions efficiently with the use of core sets"[11]. Tsang and his colleagues have proven experimentally that their proposed CVM algorithm achieves a similar performance to standard SVM but is much faster especially with large datasets.

In 2007, BVM with enclosing balls was introduced [13]as a faster version of CVM.

As mentioned earlier, other types of learning algorithms can be used, and the base learners choice depends on the suitability for the data. This choice and other related issues in this phase will not be discussed in this paper as they are not the focus of the study at this stage.

C. Combining Phase

The combining phase is the last phase of the ensemble's operation. The aim of this phase is to produce the final result of the ensemble by combining the predictions of individual models. Majority voting will be used as a fusion strategy in this experiment. Similar to the modelling phase, other types of fusion strategies can be used, their effect on the ensemble performance will not be discussed in this paper, as the focus of this paper is on partitioning phase.

IV. Experimental Design

A. The aim of the experiments

In this paper, the experiments are designed and carried out with an aim to examine the effect of three partitioning strategies on the ensemble performance when mining large datasets using the proposed ensemble method. In addition, the influence of the subset's size on the ensemble performance will be examined too.

B. Experimental Design:

1) Datasets: Five datasets of different sizes were used to evaluate the performance of our ensemble. The main characteristics of these datasets are listed in Table I. The Cover

type dataset is available from the UCI Machine Learning Repository. In our experiments the used adult dataset is a preprocessed version of the original adult dataset which is available from the UCI Repository. The details of the preprocessing process for the adult dataset can be found in [14]. The Web and IJCNN01 datasets are available on Libsvm website [1]. In addition, IJCNN01 is the preprocessed version of the original IJCNN01; full information about the preprocessing process can be found in [15]. The KDDCUP-99 dataset was downloaded from [12], and the description of the dataset is available from the KDD website [2].

All 5 datasets come with separate testing datasets. For evaluation purposes, a validation set was created from each training set. The size of validation dataset is 30% of the training set. Table II shows the characteristics of used datasets after the creation of the validation dataset.

TABLE I. DATASETS USED

Dataset	Training Instances	Testing Instances	No. of Attributes	Numeric Attributes	categorical Attributes
Adult	32,561	16,281	123	0	123
Web	49,749	14,951	300	0	300
IJCNN01	49,990	91,701	22	2	20
Cover type	522,910	58,102	54	51	3
KDDCUP-99	4,898,431	311,029	127	120	7

TABLE II. DATASETS USED AFTER THE CREATION OF VALIDATION DATASETS

Dataset	Training Instances	Validation Instances	Testing Instances	No. of Attributes
Adult	22,793	9,768	16,281	123
Web	34,825	14,924	14,951	300
IJCNN01	34,993	14,997	91,701	22
Cover type	366,037	156,873	58,102	54
KDDCUP-99	3,428,902	1,469,529	311	127

2) Experiment procedure and setups: :

1) Three series of experiments were conducted, one for each partitioning method. Figure 2 illustrates how each experiment is performed.
2) For each experiment, five datasets will be used.
3) Although for each of the five datasets a separate testing dataset is provided for testing the performance of our ensemble system, a validation dataset will be created from each training dataset to be used to estimate the learning performance of our ensemble before testing.
4) For a given dataset, each experiment will be repeated several times by varying the relative size of the subset Rt in a systematic manner as shown in the following points.
5) In all the experiments, the relative size of the subset should be chosen to create an odd number of partitions to avoid a potential tie situation when using majority voting as a fusion strategy.
6) The relative size of a subset Rt, is varied from a given minimum $Rt_{min} = 1\%$, to maximum $Rt_{max} = 49\%$ subject to the above requirements in the 5th and 6th steps. The value of Rt varied as follow Rt = 4%,6%,7%, 8%, 12%, 15%, 16%, 20%, 21%, 22%, 23%, 24%, 34%, 35%, 36%, 37% 38%, 39%, 40%, 41%, 42%, 43,% 44%, 45%, 46%, 47% and 48%.
7) CVM and majority voting will be used as the base classifier and the fusion strategy in each experiment respectively.
8) For each experiment, the results of the ensemble and individual models in training, validation, and testing datasets will be calculated. Also, the ensemble time will be recorded.
9) Statistical significance test will be carried out to analyse the results of experiments.

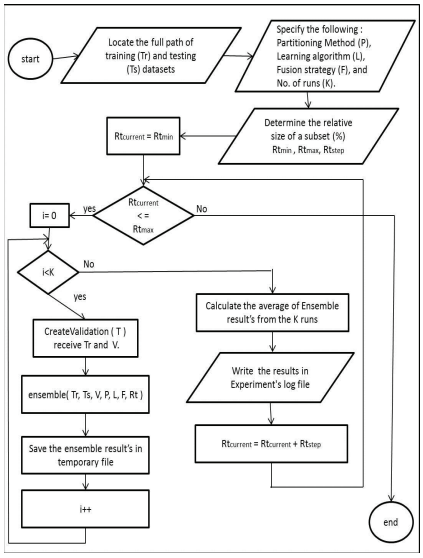


Fig. 2. The experiment procedure

3) Evaluation Methods:

1) **Accuracy and Efficiency**
The accuracy is used as a performance measure in this study. In addition partitioning time is recorded as a measure of efficiency but it is not discussed in this stage of the research due to the three used partitioning strategies have very similar time.
2) **Statistical significance analysis**
McNemar's [16] test is used in this experiment to compare between the classification errors of two classifiers (ensembles) to find out if they are statistically different or not. In addition, Friedman test [17], will be used later to compare between more than two classifiers over multiple datasets.

V. EXPERIMENTAL RESULTS

To evaluate the impact of subset size and partitioning strategy on the $Acc(\Phi)$, We run our proposed ensemble on each dataset for different runs by varying the Rt and the partitioning strategy. The results for each dataset are presented by three charts, one for each partitioning strategy. For each strategy

[1] http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/
[2] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

we monitored the ensemble accuracy $Acc(\Phi)$, and the mean accuracy of individual models of the ensemble ($(\overline{Acc(m_i)})$) on testing and validation datasets. The experimental results section is divided into thee subsections. Firstly the effect of Rt on $Acc(\Phi)$, then comparison of the accuracy of the three partitioning methods and finally statistical analysis section.

A. The effect of Rt on $Acc(\Phi)$

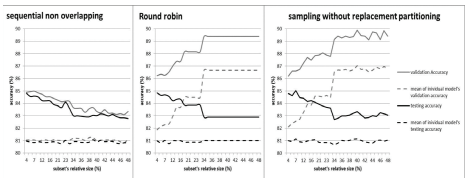


Fig. 3. Adult dataset experimental results, accuracy of ensemble, the mean accuracy of individual models within the ensemble on validation and testing datasets, against the size of subset

1) Result of Adult dataset:

- **Relationship between $Acc(\Phi)$ and Rt**
  On the adult dataset, the highest $Acc(\Phi)$ was achieved in the three partitioning methods when a small Rt was used as a size for the subsets. Figure 3 shows the $Acc(\Phi)$ on testing and validation datasets for adult dataset when the three partitioning methods were used. The left chart in Figure 3 illustrates the result when the sequential non-overlapping partitioning was used, and it shows that the $Acc(\Phi)$ on the testing dataset fell from 84.82% to 82.77% when the subset size was increased from 4% to 48% respectively. In the same manner, the middle graph and the right shows that the $Acc(\Phi)$ decreased when the subset size increased and round robin partitioning and sampling without replacement were used, respectively.
- **Relationship between $Acc(\Phi)$ and $(\overline{Acc(m_i)})$**
  Another important result that can be observed from Figure 3 is that the $Acc(\Phi)$ achieved better performance than the mean of the accuracy of individual models of the ensemble ($(\overline{Acc(m_i)})$) in all the examined cases.

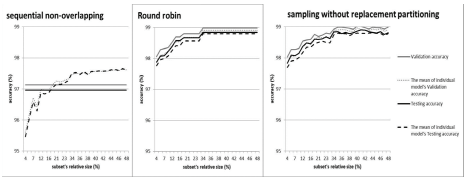


Fig. 4. Web dataset experimental results, accuracy of ensemble, the mean accuracy of individual models within the ensemble on validation and testing datasets, against the size of subset

2) Results of web dataset:

- **Relationship between $Acc(\Phi)$ and Rt**
  The effect of the size of subset on $Acc(\Phi)$ in the web dataset was completely different from its effect on the adult dataset. Figure 4 illustrates the relationship between $Acc(\Phi)$ and Rt for the three used partitioning strategies.
  When round robin and sampling without replacement were used, the $Acc(\Phi)$ had a proportional relationship to Rt. The only difference between the behaviour of the two partitioning methods on this dataset is that in round robin partitioning, the $Acc(\Phi)$ increased as a result of increasing Rt up to $Rt = 34\%$. Then after that point, (Rt = 34%) the $Acc(\Phi)$ became steady and no improvement was achieved by the growth of Rt. When sampling without replacement was used, the $Acc(\Phi)$ continued to increase by the increase of Rt. The big difference occurred when the sequential non-overlapping partitioning was used, which shows that $Acc(\Phi)$ was not affect by Rt at all and that the $Acc(\Phi)$ was steady when the value of Rt varied from 4% to 48%.
- **Relationship between $Acc(\Phi)$ and $(\overline{Acc(m_i)})$**
  Figure 4 also shows that when round robin and sampling without replacement were used, the $Acc(\Phi)$ achieves still better performance than the mean of the accuracy of individual models ($(\overline{Acc(m_i)})$). While in sequential non-overlapping partitioning, although the $Acc(\Phi)$ was steady and did not improve during the experiment, the $(\overline{Acc(m_i)})$ has a direct relationship to Rt and $(\overline{Acc(m_i)})$ where $(\overline{Acc(m_i)})$ = 95.47% when Rt =4% up to 97.62% when Rt=48%. In addition the $(\overline{Acc(m_i)})$ achieved better performance than the $Acc(\Phi)$ between Rt = 21% and Rt=48%.

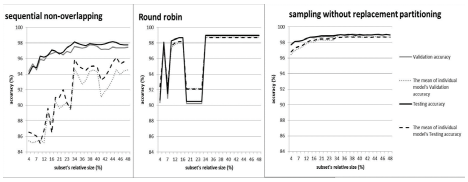


Fig. 5. IJCNN dataset experimental results, accuracy of ensemble, the mean accuracy of individual models within the ensemble on validation and testing datasets, against the size of subset

3) Results of IJCNN dataset:

- **Relationship between $Acc(\Phi)$ and Rt**
  Figure 5 shows that when the sequential non-overlapping partitioning was used, the $Acc(\Phi)$ on the testing dataset increased from 94% to 98% when Rt also increased from 4% to 48%. When round robin partitioning was used, $Acc(\Phi)$ on testing and validation datasets varied up and down, with no obvious pattern or trend, starting from Rt = 4% up to 34%; after 34%, the $Acc(\Phi)$ become steady and was not affected by the Rt, whatever its size. When sampling without replacement was applied, there was an improvement to $Acc(\Phi)$ on validation and testing datasets by increasing of Rt, as illustrated by the right chart in Figure 5.
- **Relationship between $Acc(\Phi)$ and $(\overline{Acc(m_i)})$**

By focusing on the relationship between $Acc(\Phi)$ and mean of the accuracy of individual models ($(\overline{Acc(m_i)})$), Figure 8 shows that our ensemble performs better than ($(\overline{Acc(m_i)})$) on the sequential non-overlapping and sampling methods. When round robin was used, the $Acc(\Phi)$ achieved nearly similar performance to ($(\overline{Acc(m_i)})$) in almost all cases except between Rt = 21% and 34% where ($(\overline{Acc(m_i)})$) showed better performance than $Acc(\Phi)$.

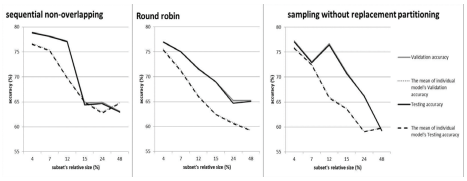


Fig. 6. Cover type dataset experimental results, accuracy of ensemble, the mean accuracy of individual models within the ensemble on validation and testing datasets, against the size of subset

4) Results of Cover type dataset:

- **Relationship between $Acc(\Phi)$ and Rt**
  The relationship between $Acc(\Phi)$ and Rt for this dataset is very similar to the relationship between $Acc(\Phi)$ and Rt on adult dataset. The highest $Acc(\Phi)$ was achieved when a small Rt was used. Figure 6 shows the $Acc(\Phi)$ on testing datasets fell dramatically from 78.86% to 62.94% when Rt varied from 4% to 48% when the sequential non-overlapping partitioning was used. The same relationship pattern appears when round robin partitioning and sampling without replacement were used.
- **Relationship between $Acc(\Phi)$ and $(\overline{Acc(m_i)})$**
  Figure 6 also illustrates that $Acc(\Phi)$ achieves better performance than the mean of the accuracy of individual models ($(\overline{Acc(m_i)})$) in all the examined cases in the three partitioning strategies.

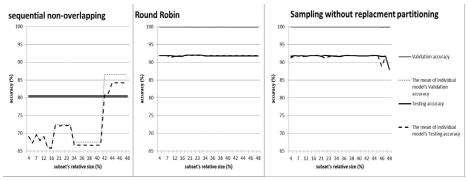


Fig. 7. KDDCUP-99 intrusion dataset experimental results, accuracy of ensemble, the mean accuracy of individual models within the ensemble on validation and testing datasets, against the size of subset

5) Results of KDDCUP-99 intrusion dataset:

- **Relationship between $Acc(\Phi)$ and Rt**
  In this dataset, the $Acc(\Phi)$ for the testing and validation dataset in all three partitioning strategies has not been affected by the relative size of the subset Rt. The $Acc(\Phi)$ was steady during the whole experiment as Figure 7 shows.
- **Relationship between $Acc(\Phi)$ and $(\overline{Acc(m_i)})$**
  Figure 7 also illustrates that $Acc(\Phi)$ achieved better performance than the mean of the accuracy of individual models ($(\overline{Acc(m_i)})$) in all the examined cases in which the round robin and sampling without replacement were used. While sequential non-overlapping partitioning was used on testing and validation datasets, the $Acc(\Phi)$ achieved better performance than $(\overline{Acc(m_i)})$ from Rt=4% up to Rt=40%. The $(\overline{Acc(m_i)})$ achieved better performance from Rt=42% than the $Acc(\Phi)$.

B. Comparing the accuracy of the three partitioning methods

1) Adult dataset: Figure 8 shows that the sequential non-overlapping partitioning and round robin partitioning yield similar pattern on the testing dataset, while sampling without replacement achieves lowest accuracy compared to the previous two partitioning strategies.

2) Web dataset: By comparing the performance of the three partitioning methods depending on testing accuracy, it is clear from Figure 8 that round robin and sampling without replacement have very similar performance, whereas the sequential non-overlapping partitioning yields lowest performance and has no effect on accuracy of ensemble on the test data.

3) IJCNN dataset: Figure 8 shows that the round robin partitioning achieved the worst accuracy in general, compared to the other two strategies. Sampling without replacement achieved better accuracy in most cases.

4) Cover type dataset: By comparing the accuracy of the three partitioning methods depending on testing accuracy, it is clear from Figure 8 that the sequential non-overlapping partitioning and sampling without replacement have very similar accuracy, while the round robin partitioning achieves the lowest accuracy when a small subset size is used, and the highest accuracy when large subsets are used compared to the other two partitioning strategies.

5) KDDCUP-99 intrusion dataset: By comparing the accuracy of the three partitioning methods depending on testing accuracy, it is clear from Figure 8 that round robin and sampling without replacement have very similar accuracy and much better accuracy than that achieved by the sequential non-overlapping partitioning.

C. Statistical analysis

We statistically evaluated our ensembles using two non-parametric approaches: the McNemars and Friedman tests.

1) McNemar's Test: McNemars test [16] is suitable to test two classifiers on a single domain. In this set of experiments McNemars test was used to determine if the classification errors of two ensembles with different Rt are statistically different.

For all datasets and for each partition , we compared the ensemble created from the smallest Rt of 4% denoted by ensemble4 and the largest Rt of 48% denoted by ensmble48.

In most of the 15 examined cases the results of McNemars test provide strong evidence to reject the null hypothesis with

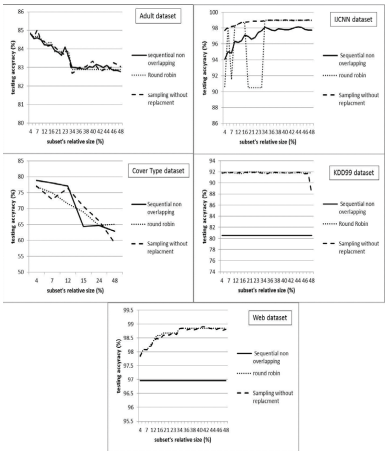


Fig. 8. Compare the effect of different partitioning methods on Adult,Web,IJCNN, Cover Type and KDDCUP-99 datasets

a P value less than 0.0005, which means that the ensemble created from the training datasets with Rt = 4% is significantly different than the ensemble created from the training datasets with Rt = 48%.

Four cases did not present enough evidence to reject the null hypothesis: the Web dataset when the sequential non-overlapping partitioning was used and in the intrusion dataset when all the three partitioning methods were used.

2) Friedman Test: The Friedman test can be used to evaluate multiple classifiers for multiple datasets [17]. In this set of experiments the Friedman test was used to determine if the three partitioning strategies used in this experiment are statically different.

The results show that there is not a statistical difference in the performance of the three partitioning methods on the five datasets with p=0.819.

VI. Discussion

Our results confirm that subset size affects the ensemble accuracy $Acc(\Phi)$ in different ways, which can be roughly classified into three groups of patterns as shown in Figure9. In P1, $Acc(\Phi)$ decreased when increasing the subset size (Rt). In P2, the $Acc(\Phi)$ increases with the growth of Rt. In P3, the $Acc(\Phi)$ does not improve or decrease with the increase of Rt.

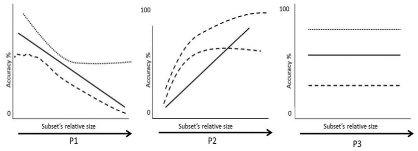


Fig. 9. Possible conceptual relationship patterns between $Acc(\Phi)$ and Rt

Figure 10 show the true relationship patterns between the relative subset size Rt and $Acc(\Phi)$in the three partitioning

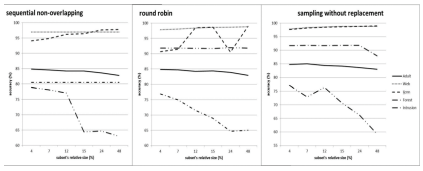


Fig. 10. Relationship patterns between $Acc(\Phi)$ and Rt which were observed in our experiments

strategies examined in this experiment for all the used datasets. It is clear from Figure 10 that the relationship between Rt and $Acc(\Phi)$in Adult and Cover type datasets is similar to pattern P1, while in web and IJCNN datasets their relationship is similar to pattern P2 and the relationship in the intrusion dataset is similar to P3.

In this experiment we intended to find out possible relation between the pattern group and dataset. It seems that the number of instances and number of attributes are not the only factors that determine the relationship pattern. Adult and Cover types belong to the same pattern, while they are very different in terms of number of instances and number of attributes. Also, Web and IJCNN belong to the same pattern; although they have a similar number of instances, but a huge difference between their numbers of attributes. For these reasons, there is a need for further investigation to find how to determine the pattern of the relationship and how to find the suitable subset size.

In addition, our results show that sampling without replacement and round robin partitioning achieves better accuracy in 80% of the examined datasets. Table III shows the best partitioning strategy for each dataset depending on their $Acc(\Phi)$ on testing datasets.

TABLE III. Best Partitioning Strategies for each dataset

Dataset	Best partitioning strategies
Adult	sequential non-overlapping and round robin
Web	round robin and sampling without replacement
IJCNN01	round robin and sampling without replacement
Cover type	sequential sampling without replacement
KDDCUP-99	round robin and sampling without replacement

VII. Conclusions

In this paper, we explored the relationship between partitioning methods and $Acc(\Phi)$. For this task we examined three partitioning strategies: sequential non-overlapping partitioning, round robin partitioning and sampling without replacement. Although sampling without replacement and round robin partitioning achieved better accuracy in 80% of the examined datasets, as shown in Table III, the results of the statistical test do not present enough evidence to conclude if there is a significant difference in the performance of the three partitioning strategies on the five examined datasets. Nevertheless in practise, sampling should be considered in the first place if there is no specific knowledge about the other two.

We also investigated the relationship between the relative size of the subset (Rt) and the ensemble accuracy ($Acc(\Phi)$). We found that Rt can have effect on $Acc(\Phi)$. As illustrated in Figure 9, this effect can be represented by one of three

relationship patterns between Rt and $Acc(\Phi)$. Our results show that the well accepted idea that having more training data instances generally leads to the best classification performance is not always true, as the best accuracy for in Adult and Cover Type datasets were achieved by having small subset sizes. The relationship patterns of a dataset cannot be detected by the number of its instances and attributes because it is found they are not the only factors that affect the behaviour of the relationship between $Acc(\Phi)$ and Rt for a dataset.

Further study should include more investigation to discover the relationship type between ensemble accuracy ($Acc(\Phi)$) and the relative size of a subset (Rt), as the number of instances and number of attributes do not provide an interpretation for this relationship. For that reason, it is necessary to repeat the experiment on larger datasets and apply some vertical petitioning techniques and conduct an in-depth and detailed analysis to find out how to determine the best size of a subset and what type of partitioning strategies is suitable for a dataset.

REFERENCES

[1] S. Totad, R. Geeta, C. Prasanna, N. Santhosh, and P. Reddy, "Scaling data mining algorithms to large and distributed datasets," *Intl J Database Manag Syst*, vol. 2, pp. 26–35, 2010.

[2] N. Amano, J. o Gama, and F. Silva, "Exploiting parallelism in decision tree induction," in *Proceedings from the ECML/PKDD Workshop on Parallel and Distributed computing for Machine Learning*, 2003, pp. 13–22.

[3] N. Kerdprasop and K. Kerdprasop, "Data partitioning for incremental data mining," in *The 1st International Forum on Information and Computer Science*. Citeseer, 2003, pp. 114–118.

[4] I. Tsang, A. Kocsor, and J. Kwok, "Diversified SVM ensembles for large data sets," *Machine Learning: ECML 2006*, pp. 792–800, 2006.

[5] T. Oates and D. Jensen, "The effect of training set size on decision tree complexity," in *Proceedings of the 14th International Conference on Machine Learning*, 1997, pp. 254–262.

[6] Oates, Tim and Jensen, David, "Large datasets lead to overly complex models: an explanation and a solution," in *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, 1998, pp. 294–298.

[7] L. Hall, N. Chawla, K. Bowyer *et al.*, "Combining decision trees learned in parallel," in *Working Notes of the KDD-97 Workshop on Distributed Data Mining*, 1998, pp. 10–15.

[8] D. Patil and R. Bichkar, "An optimistic data mining approach for handling large data set using data partitioning techniques," *International Journal of Computer Applications*, vol. 24, no. 3, pp. 29–33, 2011.

[9] J. Basilico, M. Munson, T. Kolda, K. Dixon, and W. Kegelmeyer, "COMET: A recipe for learning and using ensembles on massive data," *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pp. 41–50, 2011.

[10] C. Ranichandra and J. Vijayashree, "Efficient like commands for dynamic data retrieval and report generation from parallel databases." *Asian Journal of Computer Science and Information Technology*, vol. 2, no. 04, 2012.

[11] I. Tsang, J. Kwok, and P. Cheung, "Core vector machines: Fast SVM training on very large data sets," *Journal of Machine Learning Research*, vol. 6, no. 1, pp. 363–392, 2005.

[12] I. Tsang, A. Kocsor, and J. Kwok, "Libcvm toolkit," http://c2inet.sce.ntu.edu.sg/ivor/cvm.html, 2011, accessed: 02/02/2012.

[13] I. W. Tsang, A. Kocsor, and J. T. Kwok, "Simpler core vector machines with enclosing balls," in *Proceedings of the 24th international conference on Machine learning*, ser. ICML '07. ACM, 2007, pp. 911–918.

[14] J. Platt *et al.*, "Sequential minimal optimization: A fast algorithm for training support vector machines," 1998.

[15] C.-c. Chang and C.-J. Lin, "Ijcnn 2001 challenge: Generalization ability and text decoding," in *Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on*, vol. 2. IEEE, 2001, pp. 1031–1036.

[16] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural computation*, vol. 10, no. 7, pp. 1895–1923, 1998.

[17] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

B.2 An Algorithm for Identifying the Learning Patterns in Big Data

2015 IEEE Trustcom/BigDataSE/ISPA

An Algorithm for Identifying the Learning Patterns in Big Data

Majed Farrash
School of Computing Sciences
University of East Anglia
Email: M.Farrash@uea.ac.uk

Wenjia Wang*
School of Computing Sciences
University of East Anglia
Email: Wenjia.Wang@uea.ac.uk

***Abstract*—Divide-and-Conquer is probably the most commonly used strategy to deal with a big data that is too big to be loaded into any computing systems memory as a whole for analysis. It partitions such a big dataset into many smaller subsets that can be loaded into computer memory separately to induce models, which can be combined by machine learning ensemble methods. However, it is not clear that how the size of subsets may affect the learning performance of individual models and their ensemble. This paper proposes an ensemble based algorithm to quickly detect their relational patterns in terms of ensemble accuracy and the size of partitioned data subset. An ensemble framework of the algorithm is implemented and tested on 12 relatively big benchmark datasets. The experimental results indicate that it is able to identify the relation patterns accurately and efficiently in less than 10 steps. The identified patterns show that in most cases it is not necessary to use the whole big dataset for analysis as few smaller subsets are already sufficiently representative of the underlying problem, which is obviously a useful knowledge in big data analysis.**

***Keywords*—Big data, partitioning, subset size, ensemble learning.**

I. Introduction

The rapid improvements in data storage technologies, communication infrastructure and the availability of reliable and efficient computing hardware at an affordable cost for individuals and organisations have resulted in the exponential growth of available data. IBM states that 2.5 quintillion bytes of data are created from our daily activities [1]. The International Data Corporation (IDC) estimates that the amount of generated data is growing by 40% a year into the next decade [2]. A massive amount of data is also produced from our simple daily activities. For example, Walmart handles more than a million customer transactions each hour [3]. Facebook handles more than 250 million photo uploads and the interactions of 800 million active users with more than 900 million objects (pages, groups, etc.) each day [3]. Furthermore, billions of people around the world use their smartphones and devices daily. The availability of this enormous amount of data makes the process of analysing such data a challenging task. As a result, researchers in the field of machine learning and data mining were motivated to develop new techniques and methods to analyse big data effectively and efficiently.

Most existing machine learning algorithms have been developed under the assumption of having the whole data loaded into the main memory. However, this assumption is unrealistic in the case of dealing with big data, the divide-and-conquer technique needs to be employed. In the context of big data mining, this technique entails dividing a single big dataset into manageable subsets to overcome memory limitations. For a classification problem, an ensemble, i.e. a committee of classifiers, represents one of the best methods that utilise the divide-and-conquer technique to learn from big data. The questions in this problem are how a single big dataset should be divided and whether an ensemble of classifiers constructed from subsets can perform equally well or similar to an ensemble constructed from the whole dataset. People generally believe that building an ensemble from a large dataset results in more accurate classification, but this is not always the case according to some researchers [4] [5] [6]. Therefore, new techniques are needed to help understanding more about the data in terms of learning behaviour and decide when to use as much data as possible in building an ensemble and when not to.

In this study, an algorithm is proposed to identify the learning relation of patterns between the accuracy of an ensemble of classifiers ($acc(\phi)$) and the relative size of data (Rt) used to train classifiers. This algorithm will be helpful in deciding the best subset size of a given dataset to use when an ensemble is constructed.

The rest of this paper is organised as follows. Section II review briefly related works. Section III describes the proposed algorithm to identify the relation pattern between $acc(\phi)$ and (Rt). Section IV presents an ensemble framework that uses the detection algorithm. Section V describes our experimental design and setup. Section VI presents the results, and the last section summarises the study and gives the conclusions.

II. Related work

As mentioned previously, the ensemble method is one of the best strategies that take advantage of the divide and conquer principle. Many ensemble frameworks have been proposed in recent years to overcome the problem of learning from big datasets. Ensemble on random patches [7] is an example of an ensemble framework that was proposed by Louppe and Geurts to solve the problem of having a very strong memory constraint or having a very big dataset. Louppe and Geurts constructed an ensemble from random patches of data created through random selection of subsets of both instances and attributes from the original dataset. The authors of the random patches (RP) algorithm showed that their proposed method achieves comparable performance in terms of ensemble accuracy, with less memory requirement, compared with other randomised

978-1-4673-7952-6/15 $31.00 © 2015 IEEE
DOI 10.1109/Trustcom.2015.561

schemes. COMET [8] is MapReduce algorithm; it is an example of a distributed data mining technique that uses the ensemble framework to learn from a massive dataset. Experimental results show that COMET obtains results comparable to those of subsampling random forests that work serially on a single dataset. A SVM ensemble for large datasets was also proposed by Tasng et al. [9], and it demonstrated robust performance and a much faster training time than bagged SVMs. In addition, Chawla et al. [10] proposed a distributed ensemble framework that can be used for big datasets. Their proposed method performed at least as the performance of bootstrap bagging [11]. A distributed version of Ivotes, which was originally proposed by Breiman [12], was developed by Chawla et al. [13]; this version can be used to deal with the problem of having a large dataset. The proposed method achieves better classification accuracy than the prediction of a single classifier.

Although all the aforementioned studies represent significant contributions in the field of learning from big data, none of them have analysed the influence of data subset size on the performance of the ensemble during learning from a big dataset. The authors in the foregoing studies used either sampling (with or without replacement) or partitioning of a single training dataset into a number of disjoint subsets, whereas some of them, such as [10], used both methods. In this study, an algorithm is proposed to identify the effect of data subset size on ensemble accuracy.

III. The Proposed Algorithm for Identifying Learning patterns

In this context, learning patterns are represented by the relation between the ensemble accuracy $acc(\phi)$ and the relative subset size (Rt), and they should roughly fall into one of the three patterns illustrated in Figure 1[5]. These patterns can be described as follows: In the first pattern (P1), $acc(\phi)$ has an inverse correlation with Rt, while in the second pattern (P2), $acc(\phi)$ has some degree of correlation with Rt and in the last pattern (P3), $acc(\phi)$ does not show any correlation with Rt. The results of our previous work [5] and the results of this study suggest that the relation between $acc(\phi)$ and Rt for a given dataset often belongs to the P2 or P3 categories.

In machine learning, it is generally assumed for single base classier and an ensemble of classifiers that increasing the size of the training dataset improves the overall classifier accuracy. However, in practise this assumption does not always hold, and different behaviour and relations may occur. As previously stated, the aim of our algorithm is to detect the relation pattern between $acc(\phi)$ and (Rt) in a few search steps and thus understand how an ensemble of classifiers may behave as the size of the training data varies.

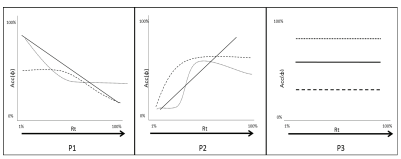


Fig. 1. Possible conceptual relationship patterns between $Acc(\Phi)$ and Rt [5]

Figure 2 shows the stages of the proposed algorithm. It works by choosing an initial Rt ($Rt_{i=0}$), which is then used to divide a given training dataset into smaller subsets. Then, these subsets are used to induce M classifiers, which belong to the same type of learning algorithm, to build a homogeneous ensemble of classifiers. It should be pointed out that different types of base classier can be used to generate methodologically different models to build heterogeneous ensemble. But this study only uses one type of base learner, i.e Decision tree, for simplicity. The outputs of the ensemble on the validation dataset is then taken as the representation of the relationship between $acc(\phi_{Rt_i}))$ and the Rt_i. This procedure will be repeated through a variation of Rt values until one or more of the stopping conditions is met. The numbers and the actual value of the selected Rt are determined by some heuristic and adaptive rules, as depicted in algorithm 1. This algorithm is implemented through an ensemble of classifiers, and the ensemble's framework is described in the next section.

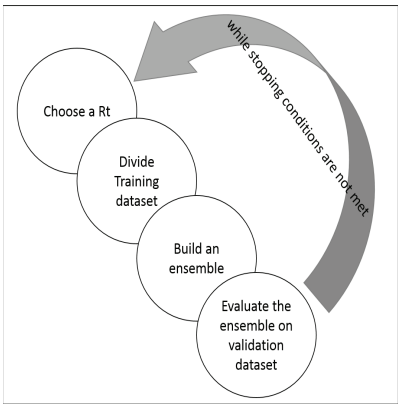


Fig. 2. The stages of the proposed algorithm for detecting the learning pattern

It should be noted that, several factors should be considered to achieve the aim of the proposed algorithm. These factors can be classified into two categories according to their effect. The first category is ensemble-related factors, such as the number of subsets (N), number of base classifiers (M), partitioning method, fusion strategy and others, which have a direct effect on the ensemble accuracy. The second category is algorithm-related factors, such as the step size, initial step size and memory limit, which have a direct influence on the performance of the detection algorithm. Most of the factors that belong to the former category have been studied in the ensemble methods literature [14] [15] [16], and for that reason they will not be discussed in detail in this paper. As our focus in this paper is on the proposed algorithm, the algorithm-related factors that also represent the inputs of the algorithm will be discussed elaborately in this section.

A. Algorithm Inputs

1) Memory safe limit (MSL): The relation between the data subset size and the memory required to build

an ensemble of classifiers is not a linear relationship. Therefore, predicting the amount of memory required to build an ensemble for a specific Rt is extremely difficult. MSL is used as precaution to prevent the algorithm from crashing due to the physical memory constraints. The MSL is used to calculate the maximum amount of memory (max_mem) that can be used by the proposed algorithm. At any time, if the consumed memory for an ensemble exceeds this limit, the algorithm will be terminated, where

$$max\_mem = AM - MSL \tag{1}$$

where AM is the available memory.

2) Lower relative subset size Rt_{low}: This parameter is the smallest relative size of the data subset, and it represents the lower bound for the Rt value, where

$$Rt_{low} = \left\lceil \frac{100\%}{M} \right\rceil \tag{2}$$

3) Initial step size (α): This parameter will be used to vary the value of Rt from one iteration to another.
4) Tolerance(θ): This parameter is the greatest range of variation allowed when the accuracy of two ensembles is compared. Therefore, if $acc(\phi)_1$ - $acc(\phi)_2 \leq \theta$, then $acc(\phi)_1 \approx acc(\phi)_2$.
5) β: This parameter is the number of successive Rt points that has an equal acc(ϕ) and needs to be identified before the algorithm is terminated.

Algorithm 1: An Algorithm for Identifying the Relation Patterns between Rt and $Acc(\phi)$

Start :
Declare counters $i = 0$ and $k = 0$.
set step_Increment =1.
Let step = α.
Let $Rt_{low} = \left\lceil \frac{100\%}{M} \right\rceil$.
Let Rt_i = Rt_{low}
Calculate the available memory(AM).
$max\_mem = AM - MSL$
$Acc(\phi_{Rt_i})$ and $mem(\phi_{Rt_i})$ ← BulidEnsemble(Rt_i)
while ($mem(\phi_{Rt_i}) < max\_mem$ *and* $k \leq \beta$ *and* $Rt_i \leq 100\%$) **do**
  i++.
  $Rt_i = Rt_{i-1} + step.$
  if ($Rt_i > 100$) **then**
    $Rt_i = 100$
  end
  $Acc(\phi_{Rt_i})$ and $mem(\phi_{Rt_i})$ ← BulidEnsemble(Rt_i) using the ensemble framework
  if ($mem(\phi_{Rt_i}) \leq max\_mem$) **then**
    Calculate $\Delta_i \leftarrow \Delta acc(\phi_{Rt_i}, \phi_{Rt_{i-1}})$
    if ($i \geq 2$) **then**
      if ($\Delta_i \approx \Delta_{i-1}$) **then**
        j++
        Let $step = \alpha \times j$.
        if $\Delta_i \approx 0$ **then**
          k++
        end
        else
          k=0
        end
      end
    end
  end
end
Draw the detected pattern for the relation between $Acc(\phi)$ and Rt .
End

B. Stopping Conditions

1) The memory consumption of an ensemble reaches the specified max_mem.
2) The search algorithm finds K successive Rt points and K = β.
3) Rt_i is greater than 100%.

IV. Methodology : Ensemble of Classifiers

An ensemble of classifiers is used in this study as a technique to address the problem of classifying a big dataset(D)that cannot be loaded as whole in the main memory. An ensemble of classifiers can be constructed in different ways, as described in [17, p.279] as follows:

- Constructing an ensemble of classifiers by manipulating the training set, such as bagging [11] and boosting [18].
- Constructing an ensemble of classifiers by manipulating the input features, such as random forest [19].
- Constructing an ensemble of classifiers by manipulating the learning algorithm to generate a heterogeneous ensemble.

In principle, the ensemble framework that is used in this study is similar to bagging. The only differences between bagging and the used ensemble are in the size of the subsets and the method of creating these subsets.

The ensemble works as follows. Firstly, it partitions the big dataset into smaller, manageable subsets. Secondly, it builds a model from each subset separately by using a single learning algorithm to form a homogenise ensemble or by using different learning algorithms to form a heterogeneous ensemble . Lastly, it combines the predictions of these models to produce the final result of the ensemble by using a fusion strategy. Figure 3 illustrates the framework of the used ensemble. The methods and techniques used in the partitioning, modelling and combining phases of the ensemble are described as follows:

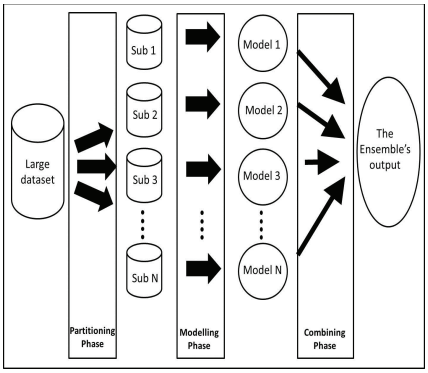


Fig. 3. Proposed Ensemble Framework [5]

A. Partitioning Phase

For the partitioning phase, there are two commonly used approaches to create different numbers of smaller subsets (N), which are derived from a single larger dataset: sampling and partitioning. Sampling can be preformed with or without replacement, while partitioning normally is used to divide the original dataset into a set of N disjoint subsets. Sampling with replacement generates subsets that have some overlaps between them, hence the models trained with these subsets can be highly correlated, or less diverse, which is no good for building an ensemble. But it has to be used when sampling without replacement will not generate enough disjoint subsets for training, even though which may lead to more diverse models.

This study aims to understand the patterns of the relation between $acc(\phi)$ and (Rt), so observing $acc(\phi)$ with different values of Rt while fixing the number of base classifiers is needed. To address this issue, a combination of partitioning into disjoint and sampling with replacement is used: Let D is the training dataset, Rt_{low} is the lowest relative subset size and N is the required number of subsets. N should be greater than or equal to $\lceil \frac{1}{Rt_{low}} \rceil$ to ensure that all the training instances are taken into consideration in the partitioning process.

- If $N = \lceil \frac{1}{Rt_{low}} \rceil$, then sampling without replacement will be used to create N disjoint data subsets.
- If $N > \lceil \frac{1}{Rt_{low}} \rceil$, then sampling without replacement will be used to create n disjoint data subsets where $n = \lceil \frac{1}{Rt_{low}} \rceil$, and sampling with replacement will be used to create the rest K data subsets where K = N-n.

Figure 4 illustrates the used partitioning method.

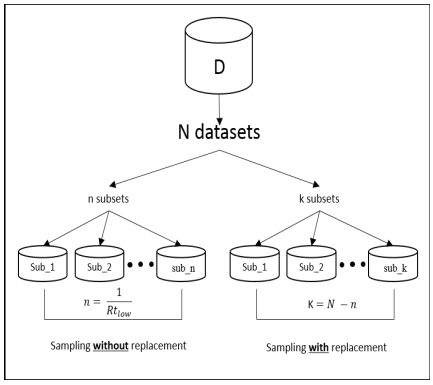


Fig. 4. Our Partitioning Method

B. Modelling and Combining phases

The C4.5 decision tree algorithm [20] is used as a base classifier. As a homogeneous ensemble of classifiers is in this experiment, the same base classifier will be used to generate the members of the ensemble. Model selection technique is not used in this experiment, so number of models(M) is equal the number of available data subsets (N).

$$M = N \tag{3}$$

To combine the predications of the individual models, majority voting is used. Because majority voting is used as a fusion strategy, M should be an odd number to avoid the tie situation. At any time, if M is an even number, then

$$M = M + 1 \tag{4}$$

V. Experimental Design and Setup

A. Datasets

Twelve datasets were used in the experiment. These datasets were obtained from the UCI Machine Learning Repository [21], the KEEL dataset repository [22] and the LIBSVM Dataset repository [23]. Table I lists the main characteristics of the used datasets. The following pre-processing steps were performed on the datasets:

1) Create the testing and validation datasets if they were not provided. The cross valuation method will not be suitable in dealing with big datasets, so the hold-out method was used to estimate the performance of the ensemble. Some of the used datasets were originally divided to training and testing datasets. In such cases, a validation dataset was created by random selection of 20% from the training dataset. In those cases in which a single dataset file was provided, the testing dataset was first created by selection of 30% of the original file. The remaining 70% were divided into two datasets to represent the training and validation datasets, as described earlier.
   The process of creating validation and testing datasets was performed with the StratifiedRemoveFolds filter, which is available in Weka [24], to ensure that the generated datasets have the same class distribution as the original dataset.
2) Attribute selection was applied on the training dataset. The InfoGainAttributeEval filter, which is available in Weka [24], was used to perform the attribute selection.
3) The class distribution in each training dataset was balanced by resampling of the instances of the minority class with the use of the SMOTE filler, which is also available in Weka [24].
4) The multiclass dataset was transformed to the binary classification dataset as follows:
   - Connect4 dataset: The class attribute in this dataset [22] originally contains three different classes. These class values represent whether a player of the game of connect-4 6x7 grid is going to win, lose or draw. To transform this dataset into a binary classification problem, the data instances that belong to the loss and draw classes were given a single class value called "loss_or_draw".

- Shuttle dataset: The class attribute in this dataset is a nominal attribute, and its values vary between 1 and 7. All instances that belong to the class value that is not equal to 1 are joined together under one class label. More details about the original dataset and class values can be found in [21] and [22].
- FARS dataset: This dataset contains statistical data on car accidents in the US in 2001. The class attribute contains eight nominal values that describe the level of injury suffered. To transform this dataset into a binary classification problem, the class attribute is transformed to describe the existence of the injury. More details about the original dataset can be found in [22].
- Poker dataset: The class attribute in this dataset takes 10 different nominal values that describe the poker hand obtained. To transform this dataset into a binary classification problem, the class attribute is transformed to describe whether a players hand can be recognised as a poker hand. More details about the original dataset can be found in [22].

TABLE I. DATASETS USED AFTER THE CREATION OF VALIDATION DATASETS

Dataset	Training Instances	Validation Instances	Testing Instances	No. of Attributes
Adult [21]	34,536	9,768	16,281	60
Census [21]	245,389	59,859	99,762	31
Connect4 [22]	48,173	9,458	20,268	43
Cover Type [21]	366,037	156,873	58,102	39
FARS [22]	89,648	14,010	30,021	30
HUGGS [21]	5,390,000	2,310,000	3,300,000	28
IJCNN01 [23]	60,138	14,997	91,701	7
Poker [22]	574,004	143,502	307,503	11
Shuttle [22][21]	54,331	8,700	14,500	10
Skin [22]	216,974	34,308	73,518	4
SUSY [21]	2,800,000	700,000	1,500,000	19
Web [23]	64,839	14,925	14,951	262

VI. EXPERIMENTAL RESULTS

Extensive experiments were conducted on the 12 datasets to evaluate the ability of the proposed algorithm to identify the relation patterns between $acc(\phi)$ and Rt. The proposed algorithm requires multiple input parameters. Therefore, our investigation started with an explanatory experiment to understand the effect of these input parameters on the algorithm and to identify the best values for these parameters in the examined datasets. Then, the validity of the proposed algorithm was statistically analysed.

A. Effect of the Input Parameters on the Algorithm

In Section V, five different input parameters were mentioned. The values of MSL and Rt_{low} depend on the available physical memory, whereas the remaining three parameters (α,θ and β) can be tuned independently from the memory constraints. To identify the effect of the input parameters on the proposed algorithm, MSL and Rt_{low} were set according to the amount of available memory, which was 20 GB, while the values of α,θ and β were varied and then analysed as follows:

1) The MSL was set to 500 MB; thus, from Equation 1, the $max\_mem = 19980$ MB.
2) Rt_{low}: Our lower relative subset size will be 1%. From Equation 2, the number of required models (M) can be calculated, and it will be equal to 100. As specified in Section IV, majority voting is used as a fusion strategy, so Equation 4 should be used to correct the value of M and avoid a tie situation. Using Equation 4, M = 101.
3) α: Varied from 1% up to 6%.
4) Tolerance (θ): Varied from 0.5% up to 2.5%.
5) β: It will be varied from 2 up to 5.

Our extensive experiments indicate that the tolerance should be less than or equal to 0.5. Increasing the tolerance value will stop the algorithm before the relation pattern can be adequately detected. By contrast, decreasing the tolerance value to a very small value will lead to an unnecessary increase in the number of steps required to detect the relation pattern.

With regard to the α and β, our extensive experiments demonstrate that the suitable α values were between 1% and 6%, whereas the value of β in most datasets was 3, except for the census and web datasets, where the β was equal to 5.

B. The Evaluation of the Discovered Relation Patterns

In order to evaluate our detection algorithm, the predicted curve is compared to the true curve that represents the true relation between $acc(\phi)$ and Rt. The predicted curve results from evaluating the ensemble at each Rt selected by the algorithm on the validation dataset. On the other hand, the true curve is formed by evaluating the ensemble with wide range of Rt values on the testing dataset. This range of Rt values begins with Rt_{low} and increases to Rt_{max}, with a step size of 2%. $Rt_{max} = 100\%$ for most of the examined datasets; the only exceptions were in the SUSY and HUGGS datasets due to memory constrains, where the Rt_{max} values were 40% and 25%, respectively. Our aim is to prove that there is no statistical different between the ensemble accuracy of the last Rt in the predicated curve ($acc(\phi_{Rt_{stop}})$) and the ensemble accuracy of maximum possible Rt on the true curve ($acc(\phi_{Rt_{max}})$).

In our experiment, McNemars test [25] was used, as described in [26], to compare the errors of two ensembles with two different Rt. The probability threshold to reject the null hypothesis was p=0.05.

C. Results

Reporting all the results of the 12 datasets with variations in the input parameters is unrealistic because of space constraints. The representative results of our extensive study will be reported and discussed in this section.

Figure 5 illustrates the detected relation between $acc(\phi)$ and (Rt), which was generated by our proposed algorithm, compared with the true relation curve, as described in section VI-B. As can be seen from the Figure, the examined datasets belong to two groups of patterns (described in section III). Seven and five datasets belong to P2 and P3, respectively. Datasets categorized as P2 are: adult, census, connect4, cover type, IJCNN01, poker and web. While FARS, shuttle, skin, SUSY and HUGGS are categorized as P3. For any dataset

belonging to the P2 category, the curve detected by our algorithm provides a hint for the best relative subset size (Rt) that needs to be chosen when constructing an ensemble from one of these datasets. For datasets that belongs to the P3 category, our algorithm gives an indication after a few steps that increasing the subset size will not improve the ensemble accuracy.

Table II shows the values of the input parameters, the number of search points required and the search time. The results obtained from Table II shows that in 83% of the investigated datasets, the number of steps required to determine the relation pattern was below 10 steps. The only two cases in which the number of steps exceeded 10 steps were in the poker and web datasets. A possible explanation for this increase in the number of steps can be noted from Figure 5, which shows that the ensemble accuracy continuously improved until Rt =70% for both datasets.

TABLE II. RESULT OF THE PROPOSED ALGORITHM, WHERE MAX_MEM =20GB ,$Rt_{low} = 1\%$ AND (θ) =0.5

Dataset	α	β	algorithm time	number of steps
Adult	3	3	3:99	7
Census	3	5	65:52	9
Connect4	6	3	2:23	6
Cover Type	5	3	161:66	7
FARS	1	3	1:83	6
HUGG	2	3	1646:84	6
IJCNN01	8	3	5:85	6
Poker	3	3	174:92	14
Shuttle	1	3	1:14	6
Skin	6	3	8:77	6
SUSY	3	3	488:01	6
Web	2	5	254:05	11

Table III shows, for each dataset, the last Rt(Rt_{stop}) where the algorithm was terminated, the absolute difference between the $acc(Rt_{Stop})$ and $acc(Rt_{max})$ and the p-value of McNemars test comparing $acc(Rt_{Stop})$ and $acc(Rt_{max})$. Looking at the absolute difference between the $acc(Rt_{Stop})$ and $acc(Rt_{max})$ for the 12 examined datasets, it is clear that our proposed algorithm has stopped at points where no more further improvement is likely to occur. The statistical test results have also confirmed that, in 9 of the 12 datasets, the proposed algorithm has stopped at Rt_{Stop}, which shows no statistically significant difference ($p > 0.05$) between $acc(\phi_{Rt_{stop}})$ and $acc(\phi_{Rt_{max}})$ at a significance level of 0.05. The only three datasets in which a significant difference was confirmed statistically are on cover type, SUSY and HUGGS datasets. But the absolute differences between $acc(Rt_{Stop})$ and $acc(Rt_{max})$ of the datasets are very small (less than 1%) even smaller than the absolute differences of some of the insignificant datasets, which implies that the statistical test used may be inappropriate for these cases, because the p values are also influenced by their large number of instances.

Another finding was that the required time to detect the relation pattern was considerably different from one dataset to another. These variations in duration can be explained by the differences in the dataset characteristics, which led to the different time durations required to build the ensemble for each dataset; this topic is outside the scope of this paper.

VII. CONCLUSION

In this paper, an algorithm has been proposed to identify the relation pattern between the accuracy of an ensemble of classifiers, acc(ϕ), and the relative size of the data, (Rt). The algorithm was evaluated empirically with 12 big datasets. Results show that the algorithm can detect the relation pattern in less than 10 steps in 80% of the investigated datasets. The algorithm can also be used to provide a suggestion for choosing the best data subset size when the ensemble of classifiers that is needed to deal with a big dataset is developed.

TABLE III. THE RESULTS OF THE STATISTICAL ANALYSIS

Dataset	Rt_{stop}	$\lvert\Delta acc(Rt_{stop}, Rt_{max})\rvert$	Rt_{stop} VS Rt_{max}
Adult	43%	0.44	p value = 0.06
Census	88%	0.11	p value = 0.07
Connect4	67%	0.23	p value = 0.208
Cover Type	61%	0.49	p value $<$ 0.0001
FARS	12%	0.01	p value = 0.625
HUGGS	23%	0.05	p value $<$ 0.0001
IJCNN01	89%	0.13	p value = 0.225
Poker	97%	0.29	p value = 0.225
Shuttle	12%	0.01	p value = 0.625
Skin	67%	0.08	p value=0.125
SUSY	34%	0.05	p value $<$ 0.0001
Web	81%	0.09	p value = 0.263

In further work, it will be interesting to reduce the Rt to a very small value, such as 0.1%, 0.01% and 0.001%, especially for datasets that belong to the P3 pattern, to understand how the relation between $acc(\phi)$ and Rt will be affected in the case of using a very small Rt. At the moment, the input parameters for the proposed algorithm were chosen depending on the empirical results of our experiments. Further work needs to be done to design an adaptive method to choose the best input values for each dataset. Furthermore, the current experiments were conducted on a single machine. Accordingly, more improvements to the proposed detection algorithm are needed to develop a new version of the algorithm that is compatible with a parallel computing environment to reduce the required time to find the relationship pattern.

REFERENCES

[1] IBM. (2014) What is big data? [Online]. Available: http://www-01.ibm.com/software/data/bigdata/what-is-big-data.html

[2] V. Turner, J. F. Gantz, D. Reinsel, and S. Minton, "The digital universe of opportunities: Rich data and the increasing value of the internet of things," *International Data Corporation, White Paper, IDC_1672*, 2014.

[3] M. Troester, "Big data meets big data analytics," *Cary, NC: SAS Institute Inc*, 2012.

[4] T. Oates and D. Jensen, "The effect of training set size on decision tree complexity," in *Proceedings of the 14th International Conference on Machine Learning*, 1997, pp. 254–262.

[5] M. Farrash and W. Wang, "How data partitioning strategies and subset size influence the performance of an ensemble?" in *Big Data, 2013 IEEE International Conference on*. IEEE, 2013, pp. 42–49.

[6] Oates, Tim and Jensen, David, "Large datasets lead to overly complex models: an explanation and a solution," in *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, 1998, pp. 294–298.

[7] G. Louppe and P. Geurts, "Ensembles on random patches," in *Machine Learning and Knowledge Discovery in Databases*. Springer, 2012, pp. 346–361.

[8] J. D. Basilico, M. A. Munson, T. G. Kolda, K. R. Dixon, and W. P. Kegelmeyer, "Comet: A recipe for learning and using large ensembles on massive data," in *Data Mining (ICDM), 2011 IEEE 11th International Conference on*. IEEE, 2011, pp. 41–50.

[9] I. W. Tsang, A. Kocsor, and J. T. Kwok, "Diversified svm ensembles for large data sets," in *Machine Learning: ECML 2006*. Springer, 2006, pp. 792–800.

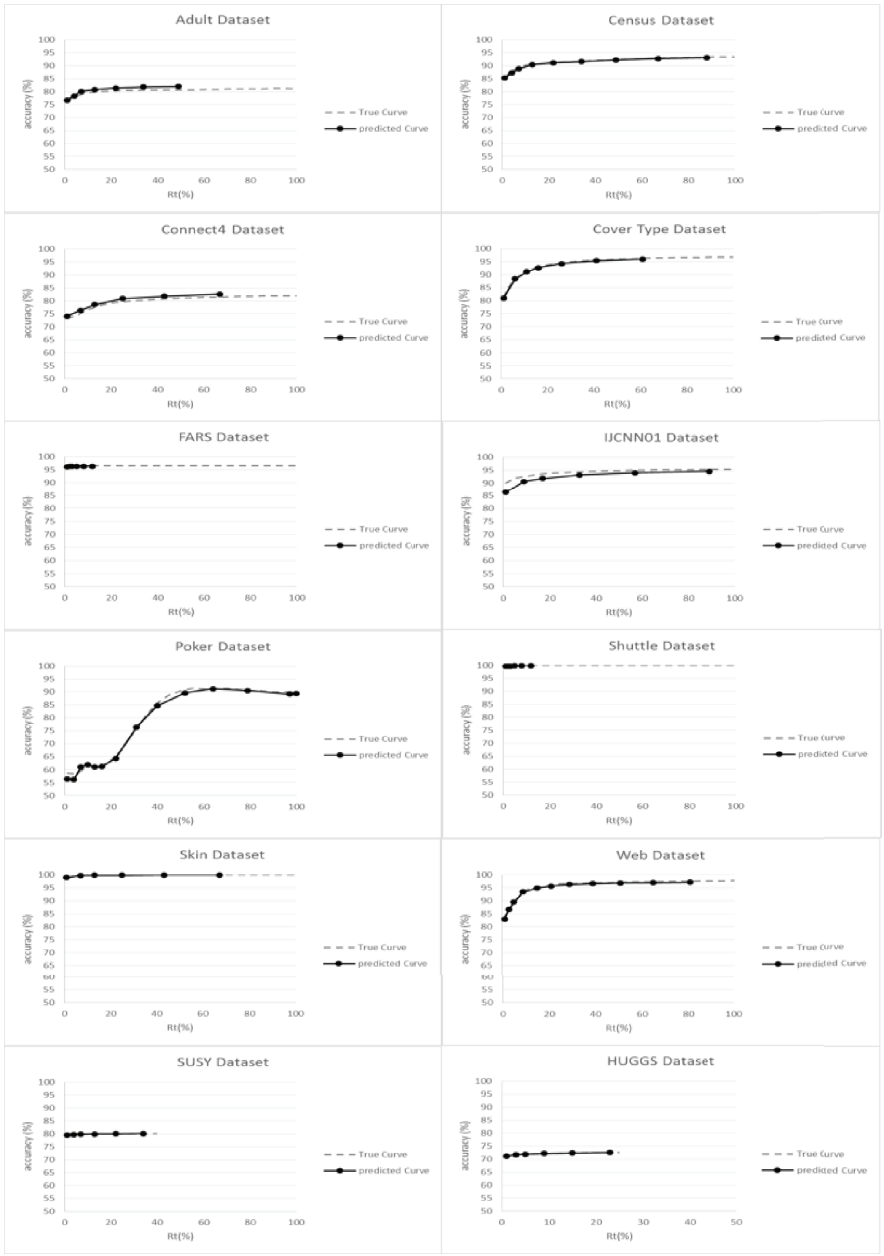


Fig. 5. The patterns identified by the proposed algorithm on the 12 datasets.

[10] N. V. Chawla, T. E. Moore, L. O. Hall, K. W. Bowyer, W. P. Kegelmeyer, and C. Springer, "Distributed learning with bagging-like performance," *Pattern recognition letters*, vol. 24, no. 1, pp. 455–471, 2003.

[11] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[12] ——, "Pasting small votes for classification in large databases and online," *Machine learning*, vol. 36, no. 1-2, pp. 85–103, 1999.

[13] N. V. Chawla, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, "Learning ensembles from bites: A scalable and accurate approach," *The Journal of Machine Learning Research*, vol. 5, pp. 421–451, 2004.

[14] W. Wang, "Some fundamental issues in ensemble methods," in *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*. IEEE, 2008, pp. 2243–2250.

[15] C.-X. Zhang and R. P. Duin, "An experimental study of one-and two-level classifier fusion for different sample sizes," *Pattern Recognition Letters*, vol. 32, no. 14, pp. 1756–1767, 2011.

[16] G. Tsoumakas, I. Partalas, and I. Vlahavas, "A taxonomy and short review of ensemble selection," in *ECAI 2008, workshop on supervised and unsupervised ensemble methods and their applications*, 2008, pp. 41–46.

[17] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.

[18] Y. Freund, R. E. Schapire *et al.*, "Experiments with a new boosting algorithm," in *ICML*, vol. 96, 1996, pp. 148–156.

[19] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[20] J. R. Quinlan, *C4. 5: programs for machine learning*. Elsevier, 2014.

[21] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[22] J. Alcalá, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 17, no. 255-287, p. 11, 2010.

[23] C. Chang and C. Lin, "Libsvm – a library for support vector machines," http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/, 2012, accessed: 02/06/2012.

[24] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[25] Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages," *Psychometrika*, vol. 12, no. 2, pp. 153–157, 1947.

[26] N. Japkowicz and M. Shah, *Evaluating learning algorithms: a classification perspective*. Cambridge University Press, 2011.