

# Evaluating ensemble classifiers for spam filtering

---

November 10, 2005

James M Carpinter  
Supervisor: Dr. Brent Martin

Computer Science and Software Engineering  
University of Canterbury  
Christchurch, New Zealand

---

### **Abstract**

In this study, the ensemble classifier presented by Caruana, Niculescu-Mizil, Crew & Ksikes (2004) is investigated. Their ensemble approach generates thousands of models using a variety of machine learning algorithms and uses a forward stepwise selection to build robust ensembles that can be optimised to an arbitrary metric. On average, the resulting ensemble out-performs the best individual machine learning models. The classifier is implemented in the WEKA machine learning environment, which allows the results presented by the original paper to be validated and the classifier to be extended to multi-class problem domains. The behaviour of different ensemble building strategies is also investigated. The classifier is then applied to the spam filtering domain, where it is tested on three different corpora in an attempt to provide a realistic evaluation of the system. It records similar performance levels to that seen in other problem domains and out-performs individual models and the naive Bayesian filtering technique regularly used by commercial spam filtering solutions. Caruana et al.'s (2004) classifier will typically outperform the best known models in a variety of problems.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Research objectives . . . . .	2
1.2	Report structure . . . . .	2
1.3	An overview of the library-based ensemble classifier . . . . .	3
<b>2</b>	<b>Background and prior research</b>	<b>4</b>
2.1	Component algorithms . . . . .	4
2.1.1	<i>k</i> -Nearest-Neighbour . . . . .	4
2.1.2	Artificial neural networks . . . . .	5
2.1.3	Decision trees . . . . .	5
2.1.4	Support vector machines . . . . .	6
2.2	Ensemble theory . . . . .	6
2.3	Ensemble algorithms . . . . .	7
2.3.1	Bagging . . . . .	8
2.3.2	Boosting . . . . .	8
2.4	Spam filtering . . . . .	8
2.4.1	Machine learning approaches . . . . .	8
2.4.2	Non-machine learning approaches . . . . .	10
<b>3</b>	<b>Library-based ensemble classifier</b>	<b>11</b>
3.1	Training process . . . . .	11
3.2	Selection strategies . . . . .	11
3.2.1	Deterministic selection . . . . .	11
3.2.2	Non-deterministic selection . . . . .	12
3.3	Optimisation . . . . .	13
<b>4</b>	<b>System overview</b>	<b>15</b>
4.1	WEKA machine learning environment . . . . .	15
4.2	Corpus builder . . . . .	15
4.3	WEKA classifier plugin . . . . .	16
4.4	WEKA model builder . . . . .	19
<b>5</b>	<b>Evaluation</b>	<b>20</b>
5.1	Overview . . . . .	20
5.1.1	Experimental setup . . . . .	20
5.1.2	Reporting . . . . .	22
5.2	Classifier validation . . . . .	22
5.2.1	Aim . . . . .	22
5.2.2	Hypothesis . . . . .	22
5.2.3	Method . . . . .	23
5.2.4	Results . . . . .	23
5.2.5	Discussion . . . . .	26

5.3	Classifier extension . . . . .	28
5.3.1	Aim . . . . .	28
5.3.2	Hypothesis . . . . .	28
5.3.3	Method . . . . .	29
5.3.4	Results . . . . .	29
5.3.5	Discussion . . . . .	30
5.4	Application to spam . . . . .	31
5.4.1	Aim . . . . .	31
5.4.2	Hypothesis . . . . .	31
5.4.3	Method . . . . .	32
5.4.4	Results . . . . .	33
5.4.5	Discussion . . . . .	36
5.5	Experimental observations . . . . .	37
<b>6</b>	<b>Conclusions and further work</b>	<b>38</b>
6.1	Further research and work . . . . .	39
6.2	Acknowledgements . . . . .	39
<b>A</b>	<b>Detailed results</b>	<b>44</b>
A.1	Classifier validation experiments . . . . .	44
A.2	Classifier extension experiments . . . . .	47
A.3	Application to spam experiments . . . . .	51

# Chapter 1

## Introduction

The first email message recognised as ‘spam’ was sent in 1978 to the users of Arpanet, and represented little more than a minor annoyance. Today, spam represents much more than that: it is an ever-present threat to the productivity of users and to the stability of IT infrastructures worldwide. Spam is estimated to represent 80% of all email sent and is expected to cost users and organisations US\$50 billion in 2005, in terms of lost productivity and increased network maintenance (Postini Inc. 2005).

Stanford University estimates that the average internet user loses ten days a year dealing with incoming spam (Zeller Jr. 2005). However, this estimate does not consider the potential harm that some spam email messages expose users to: 15% of spam messages contain some kind of virus payload and 1 in 3,418 contained pornographic images (Wagner 2002). Ongoing investment in IT infrastructure can also be largely attributed to increasing volumes of spam: 50–60% of email was estimated to be spam in 2003 and this has risen to an estimated 80% in 2005 (Zeller Jr. 2005). Email servers worldwide are finding themselves increasingly unequipped for rapidly escalating traffic flows and the associated security risks.

The business model of the ‘spammer’ (a person who sends spam) differs from that of the postal junk mailer and forms an infinitely more appealing business proposition. Spam can be sent with virtually no cost to the sender and the economic realities that regulate postal junk mail do not apply online. Commissions to spammers of 25–50% are not unusual and given a \$50 product, a collection of 200 million email addresses and a response rate of 0.001%, the return to the spammer would be \$25,000 (Zeller Jr. 2005). Furthermore, legal remedies are limited as it is not difficult to send spam outside the jurisdiction of local law enforcement or to avoid leaving a trace.

Spam can be loosely defined as unsolicited bulk email; a more comprehensive definition is provided by Mail Abuse Prevention Systems, which identifies three key spam characteristics (Mail Abuse Prevention Systems 2004):

1. “The recipient’s personal identity and context are irrelevant because the message is equally applicable to many other potential recipients; AND,” (simple, automated customisation does not make the recipient’s identity relevant),
2. “The recipient has not verifiably granted deliberate, explicit, and still-revocable permission for it to be sent; AND,” (failure by the user to explicitly opt-out during web registration does not convey permission, except when the opt-in selection is not default, and the impact on the user’s mailbox is explicitly stated),
3. “The transmission and reception of the message appears to the recipient to give a disproportionate benefit to the sender.” (“disproportionate benefit” is solely determined by the recipient; this will exist unless the email was authorised by the recipient)

Non-spam email is referred to as ‘legitimate’ or ‘ham’ email.

The extent of the spam problem has forced many organisations to deploy a spam filtering solution. Current solutions are generally based on a static, regular expression-style, rule set, which is often complemented by a naive Bayes filter. Most enterprise-scale solutions are expensive to deploy, further negatively

impacting IT budgets, and are far from perfect; the existence of false positives often requires a user to double check filtered messages (somewhat defeating the point of the filter). However, it is understandable that this vexing problem has not yet been satisfactorily solved: the classification task is complex, constantly changing and often user-specific. Constructing a single model to classify the broad range of spam sent to a group of diverse users is difficult; this task is almost impossible with the realisation that the contents and structure of spam emails evolve on a regular basis, with changing consumer trends and the introduction of new, anti-detection HTML structures.

The need for a robust spam filtering technique should by now be obvious, as should the difficulty of implementing such a system. Any solution must be able to identify the complex boundary between spam and non-spam, and adjust the boundary as necessary. Machine learning techniques were first applied to this problem in 1998, and have been followed by a significant amount of academic research and commercial success (Sahami, Dumais, Heckerman & Horvitz 1998, Pantel & Lin 1998). The ability of machine learning algorithms to infer relationships that are not necessarily obvious to the user and to develop their model over time has presented the opportunity for the development of a substantially more robust filtering solution than what is currently available. Current research focuses on identifying machine learning algorithms which are best suited to the task; no complete solution has been identified, and the work is ongoing.

Caruana et al. (2004) present a new, ensemble-based, machine learning technique with positive preliminary results. An ensemble classifier consists of a number of individual machine learning models, each of which contributes to the overall classification decision reached by the ensemble. The performance of their ensemble was shown to reliably out-perform that of the component algorithms, reducing loss<sup>1</sup> by 8.76% on average. It is interesting to note that many of the component algorithms used have been shown to be effective in the spam classification domain (see section 2.4). Furthermore, the classifier was shown to be effective on a wide range of datasets.

The application of Caruana et al.'s (2004) ensemble classifier to the spam filtering domain is logical: the spam filtering problem has already been shown to be well-suited to machine learning-based solutions, and their ensemble classifier has been shown to provide reductions in loss in a variety of circumstances. It is hypothesised that a spam filter based on their ensemble classifier design will more accurately classify spam and legitimate email than any of its individual (component) classifiers. The ability of an ensemble classifier to address some of the limitations of existing machine learning algorithms should result in an increase in accuracy (compared with the accuracy of individual algorithms). If this hypothesis is proved valid, an ensemble-based classifier would represent a potential solution to the spam problem. Given the estimated fiscal impact of spam in 2005, new techniques, such as this one, are required to ensure email remains a productive tool for both home and business users.

It should be noted that filtering is but one solution to the spam problem. Other solutions include legislation (which has been adopted by the USA, the EU and Australia), protocol change (for example, to require sender authentication, or a per-email fee) and interactive filters (also known as 'challenge-response' systems). Non-interactive filters are able to classify emails without human intervention (although they are likely to offer the user some customisation options). Non-interactive filters represent the most likely short-term solution to the spam problem given their unsupervised nature and ease of adoption (when compared with legislation or a protocol change).

## 1.1 Research objectives

The goal of this research is to evaluate the effectiveness of the library-based ensemble classifier as described by Caruana et al. (2004), both as a stand-alone classifier in a variety of problem domains, as well as a classifier for filtering spam email messages. The evaluation of the classifier as a stand-alone learning method aims to validate the existing published results and to evaluate the classifier when exposed to multi-class problems (which is considered 'future work' by Caruana et al. (2004)).

---

<sup>1</sup>Loss is defined in section 5.1.2. Loss in accuracy is equivalent to error.

## 1.2 Report structure

Section 1.3 gives a brief overview of the ensemble classifier provided by Caruana et al. (2004); this is to give the reader some context for chapter 2, which covers the relevant background research in spam filtering and in machine learning. Chapter 3 describes the ensemble classifier in more detail, before the implementation is detailed in chapter 4. Chapter 5 discusses the evaluation performed on the ensemble classifier, and chapter 6 outlines the conclusions and further research that results from this study.

## 1.3 An overview of the library-based ensemble classifier

A brief introduction to the library-based ensemble classifier is provided here to put the background material presented in chapter 2 in context. Caruana et al.'s (2004) classifier has two distinct stages in its operation:

**Library construction:** Before the ensemble classifier can be formed, a library of models (hypotheses) must be assembled. The library is constructed by generating models of different types (e.g. artificial neural networks, decision trees, etc.) with different operating parameters. Caruana et al. (2004) generated libraries that contained approximately 2000 models over a period of 48 hours with a cluster of ten Linux machines. The model library is reusable: with the same dataset, the ensemble construction process can be run multiple times with different parameters on the same library.

**Ensemble construction:** The performance of the models will be highly variable; some will have excellent performance, while some will have mediocre (or even poor) performance. Instead of combining all models in an ensemble, forward stepwise selection is used to find a subset of models to form an ensemble with excellent performance. The basic ensemble algorithm is as follows:

1. Initialise the empty ensemble.
2. Add the model from the library that maximises the performance (as measured by an arbitrary metric) of the ensemble on the validation set.
3. Repeat step two for a fixed number of iterations, or until a stop condition is reached.
4. Return the subset of models that maximises performance on the validation set. This will not necessarily be the final ensemble; maximum performance may be achieved part way through the ensemble building process, and it is this subset that will be returned.

This simple algorithm is henceforth referred to as `SELECTIONWITHOUTREPLACEMENT`. When compared to the library building process, ensemble construction is relatively fast. The predictions of each model in the library are averaged with those of the ensemble and the model that improves ensemble performance the most is added. The ensemble can be optimised to any performance metric; it is likely that some subsets of models from the library will have excellent performance on the required metric.

Further details on the operation of the classifier (including optimising on different metrics and the use of alternative selection strategies) are detailed in chapter 3.

## Chapter 2

# Background and prior research

Applying Caruana et al.'s (2004) ensemble classifier to the spam classification problem builds on three areas of research, each of which will be reviewed in the chapter. Section 2.1 reviews the individual classifiers that will comprise the model library used for the experiments discussed in chapter 5, with an emphasis on the underlying theory and bias for each classifier, as well as details of the specific implementation of the algorithm used. Section 2.2 describes how ensemble classifiers, in general, leverage the individual biases of their component classifiers to perform more reliable and robust classification (than any individual component), and section 2.3 describes several ensemble construction approaches. Finally, section 2.4 reviews current research in the spam classification domain.

### 2.1 Component algorithms

The algorithms described in this section are used to generate the model library used in the evaluation (see chapter 5). The composition of the model library will affect its performance; however, the number of combinations of different classifiers and parameters is almost infinite and thus a thorough evaluation of this experimental factor is outside the scope of this report. Therefore, the classifiers that are used are similar to those used by Caruana et al. (2004), and represent a broad sample of machine learning approaches. More importantly, the collection of classifiers uses a variety of different biases to infer knowledge, increasing the diversity of the resulting model library (this is discussed further in section 2.2).

The decision to implement the system inside the WEKA machine learning environment (see chapter 4) prevented the use of the same implementations as those used by Caruana et al. (2004); similar implementations that were available in WEKA were used instead (Witten & Frank 2000).

#### 2.1.1 *k*-Nearest-Neighbour

The *k*-Nearest-Neighbour algorithm (*k*-NN) is a memory-based (or ‘instance-based’) machine learning method (Cover & Hart 1967). This approach does not seek to develop a model that describes the structure of the underlying data; rather, all training examples are stored, and test instances are classified by estimating their similarity to the stored examples. In its simplest form, each instance is a point in a multi-dimensional space, where its location is defined by its attributes. Test instances are assigned the majority class of the *k* closest instances. The underlying inductive bias of the algorithm assumes instances that are close in the attribute space are of the same class.

The IBk algorithm (based on the work of Aha, Kibler & Albert (1991)) implemented in the WEKA package was selected for use in this evaluation. It is based on traditional *k*-NN classification principles; however, it offers a range of distance weighting options similar to those used by Caruana et al. (2004). Three different schemes are available: no weighting (where the *k* closest instances determine the class), inverse weighting (where the inverse of the distance weights the impact of each instance on the final classification) and similarity weighting (where the vote of each neighbour in the classification decision is weighted



based on its attribute similarity to the test instance). The different weighting schemes allow the fine tuning of the inductive bias of the  $k$ -NN algorithm, as does the  $k$  value adopted.

### 2.1.2 Artificial neural networks

An artificial neural network (ANN) is an artificial model of the biological brain, where a group of interconnected artificial neurons use a mathematical model to process information (Mitchell 1997). A group of simple (in terms of their processing ability) processing nodes (neurons) are interconnected and the overall behaviour of the network is largely defined by the nature of the connections between the nodes. ANNs can approximate any function, given a sufficient number of neurons; however, practically sized ANNs are still capable of representing a rich space of non-linear functions.

The MultilayerPerceptron algorithm implemented in the WEKA package was selected for use in this evaluation. It uses the BACKPROPAGATION algorithm (as was used by Caruana et al. (2004)) to train the network (Rumelhart, Hinton & Williams 1986). Based on a fixed number of neurons and interconnections, it learns the connection weights for the network by using gradient descent to minimise the squared error between the actual and target outputs of the network (Mitchell 1997). The inductive bias of ANNs trained using the BACKPROPAGATION algorithm is difficult to quantify; however, it can be characterised as “smooth interpolation between data points” (Mitchell 1997). The algorithm also supports the variation of the learning rate and the momentum level in an attempt to avoid the local minima problem often encountered by ANNs.

### 2.1.3 Decision trees

Decision trees use tests on one or more attributes to classify a particular instance. A typical tree has several internal nodes, which represent tests, and several child nodes, which represent all potential classification outcomes; the tree can effectively be described as a series of if-else rules. Most tree generation algorithms identify an attribute which best differentiates between classes, creates a test (branch) on the value of that attribute, and then repeats the process to generate a tree. The process is stopped when the tree accurately predicts each instance, or when a stop condition has been reached.

Several decision trees from the WEKA package have been included.

- ID3 is a basic decision tree, which searches through the space of all possible decision trees using information gain to grow the tree from general to specific (Quinlan 1986, Mitchell 1997). Its inductive bias is to prefer shorter trees over larger trees.
- J48 is also used, which is an open implementation of the highly popular C4.5 package designed to resolve many of the issues surrounding the ID3 algorithm (Quinlan 1993). It provides a number of parameters that allow the user to control the growth of the tree and allows pruning to prevent over-fitting.
- ADTree is a generalisation of decision trees, voted decision trees and voted decision stumps (Freund & Mason 1999). It is designed as an alternative to boosted decision trees and avoids the associated large computational requirements and complexity.
- LMT or Logistic Model Trees, use logistic regression models in the leaves of the decision tree (Landwehr, Hall & Frank 2003).
- M5P is an adaption of the traditional decision tree to predict continuous values, rather than distinct classes, by using multivariate linear models (Quinlan 1992).
- NBTree is a hybrid algorithm that combines naive Bayes classification with decision tree classification (Kohavi 1996). Each inner tree node contains a uni-variate split, like standard decision trees, while leaf nodes contain naive Bayesian classifiers.
- REPTree (Reduced Error Pruning) has a fast pruning algorithm to correct the effects of spurious/noisy training examples on the decision tree (Frank & Witten 1998).

The selection of decision tree classifiers adopted by Caruana et al. (2004) are somewhat different to those used here; however, the underlying difference in bias and hypothesis search technique contributed by any standard decision tree is similar, and it is this that provides value to the resulting ensemble (rather than a specific implementation).

### 2.1.4 Support vector machines

Support vector machines (SVMs) map training instances into a higher dimensional feature space by some nonlinear function, and then calculate the optimal hyperplane which maximises the margin between the data points in the positive class and the data points in the negative class (Hearst, Dumais, Osman & Scholkopf 1998). The hyperplane can then be used to classify new instances (i.e. it is the decision boundary). In practice, it involves no computations in the feature space; kernel functions allow the construction of the hyperplane without mapping the training data into the higher-dimensional feature space.

The SMO algorithm implemented in WEKA is used to construct SVM models; this implementation is based on the work of Platt (1998). Sequential Minimal Optimisation (SMO) is a fast method of training SVMs. It breaks a large quadratic programming problem down into a series of the smallest possible sub-problems, minimising processor and memory demands. The algorithm implements both polynomial and radial basis function kernels, allowing the construction of similar SVM models to that seen in Caruana et al. (2004). The inductive bias of the algorithm is largely dependent on the kernel adopted; different kernels will lead to different models.

## 2.2 Ensemble theory

Ensembles of classifiers can often perform better than any individual classifier; this performance advantage can be attributed to three key factors (Dietterich 2000) (see figure 2.1):

**Statistical:** Machine learning algorithms attempt to construct a hypothesis that best approximates the unknown function that describes the data, based on the training examples provided. Insufficient training data can lead an algorithm to generate several hypotheses of equal performance. By taking all of the candidate hypotheses and combining them to form an ensemble, their votes are averaged and the risk of selecting an incorrect hypothesis is reduced.

**Computational:** Many machine learning approaches are not guaranteed to find the optimal hypothesis; rather, they perform some kind of local search which may find a local minima (rather than the global minima, or the optimal hypothesis). For example, decision trees and artificial neural networks can often produce sub-optimal solutions. By starting the local search in different locations, an ensemble can be formed by combining the resulting hypotheses; therefore, the resulting ensemble can provide a better approximation of the true underlying function.

**Representational:** The Statistical and Computational factors allow ensembles to locate better approximations of the true hypothesis that describes the data; however, this factor allows ensembles to expand the space of representable functions beyond that achievable by any individual classifier. It is possible that the true function may not be able to be represented by an individual machine learning algorithm, but a weighted sum of the hypotheses within the ensemble may extend the space of representable hypotheses to allow a more accurate representation.

These factors represent three key advantages that machine learning ensembles hold; they also represent three of the major limitations often recognised in specific machine learning algorithms. It is for this reason that ensembles have the potential to deliver better performance<sup>1</sup> than many individual learning algorithms.

In order for these advantages to eventuate, a necessary and sufficient condition is that the component classifiers are accurate and diverse (Hansen & Salamon 1990). Accurate classifiers have an error rate better than random guessing, and diverse classifiers make different errors on new data points. Locating accurate classifiers within the WEKA environment is not a difficult task; however, ensuring and measuring diversity

---

<sup>1</sup>In terms of accuracy, rather than in terms of their computational resource demands.

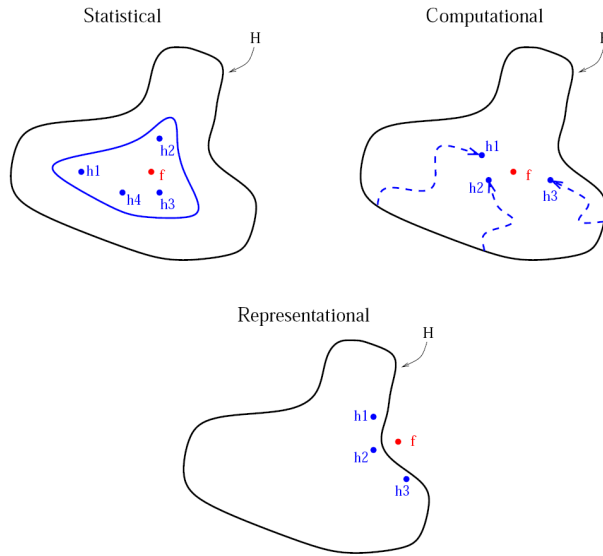


Figure 2.1: A graphical description of the three fundamental reasons why classifier ensembles can outperform individual classifiers (the outer curve denotes the hypothesis space searched by the learning algorithm) (Dietterich 2000).

is more difficult. The model library should be sufficiently diverse, given the range of models generated and their fundamental differences in search techniques and inductive bias. The diversity of the resulting ensemble needs to be monitored and the selection strategy adjusted as necessary. Both of these conditions must be filled for any ensemble to outperform its components.

## 2.3 Ensemble algorithms

Five general approaches exist for constructing ensembles (Dietterich 2000):

**Bayesian voting:** This approach primarily addresses the statistical component of ensemble theory. It combines several hypotheses using Bayesian conditional probabilities to arrive at a prediction for the instance being classified:

$$P(f(\mathbf{x}) = y | S, \mathbf{x}) = \sum_{h \in \mathcal{H}} h(\mathbf{x}) P(h | S)$$

where the left hand side indicates the probability that the instance  $\mathbf{x}$  is of class  $y$ , given the training sample  $S$ ; this probability is the combined votes of each hypothesis ( $h$ ) in the hypothesis space ( $\mathcal{H}$ ), each weighted by its posterior probability. This scheme is optimal if the true hypothesis that describes the data has been selected from the hypothesis space. In some problems, it is possible to enumerate each  $h \in \mathcal{H}$  to form the collection of models; otherwise, Bayesian voting can be approximated using random samples.

**Manipulating training examples:** This approach requires each individual learning algorithm to split or reuse training examples in order to generate a collection of hypotheses. Any individual learning algorithm is run multiple times, and a different subset of the training examples are used on each iteration; this process outputs multiple hypotheses, which are generally combined using a weighted vote. This approach is particularly well-suited to unstable machine learning algorithms (those whose

model is particularly sensitive to small changes in the training set); for example, decision trees, artificial neural networks and rule learning algorithms (linear regression, nearest neighbour, and linear threshold algorithms are generally more stable).

**Manipulating input features:** This approach requires each learning algorithm to generate multiple hypotheses by training models on subsets of the attributes of each instance. Positive results have been obtained by training models on related subsets of attributes; however, this technique only works when the input features are highly redundant.

**Manipulating output targets:** This approach requires each learning algorithm to construct an ensemble by manipulating the  $y$  (class) values that are provided to the learning algorithm. A large, multi-class dataset is divided into a binary problem by relabelling half the classes to 0, and the remainder to 1, and a classifier is generated from the relabelled dataset. This process is repeated  $n$  times to construct a set of hypotheses to construct the ensemble; new instances are classified on the votes generated by the component classifiers.

**Injecting randomness:** The use of randomness introduces an element of chance into the model building process; each time the model is built, a new hypothesis should be generated for the ensemble. For example, the BACKPROPAGATION algorithm used to train artificial neural networks can initialise the start weights to random values, or the C4.5 decision tree algorithm could introduce an element of randomness when selecting the next attribute to branch on.

The ensemble building process described by Caruana et al. (2004), and investigated in this report, does not fit cleanly in any of the described categories. Hypotheses are generated by a diverse set of algorithms with a range of parameters; the diverse set of classifiers required for a successful ensemble is generated through the use of different algorithms, rather than using the same algorithm and the previously described techniques.

Two ensemble building techniques are used to construct models for the library used in the evaluation (see chapter 5): bagging (see section 2.3.1) and boosting (see section 2.3.2). Other ensemble algorithms include stacking (Wolpert 1990), cross-validated committees and error-correcting output coding (Dietterich 2000).

### 2.3.1 Bagging

Bootstrap aggregation (or bagging, as it better known) manipulates training examples to generate a set of hypotheses to form the ensemble (Breiman 1996). On each iteration, the algorithm is presented with a randomly selected subset of the training dataset (an instance can be selected for the subset multiple times; i.e. selection with replacement). The WEKA Bagging algorithm is used on the decision tree algorithms described in section 2.1.3, along with the DecisionStump algorithm (a single level decision tree). No other machine learning methods are bagged; this is consistent with the approach taken by Caruana et al. (2004).

### 2.3.2 Boosting

Boosting also manipulates training examples to generate a set of hypotheses (Freund & Schapire 1996). Instead of dividing the training set, it attaches weights to each of the training instances, and on each iteration attempts to minimise the weighted error on the training set. Misclassified instances have their weight increased and correctly classified instances have their weight decreased; the learning problem effectively increases in difficulty with each iteration. The WEKA AdaBoostM1 algorithm is used on the same decision tree algorithms subjected to bagging (see section 2.3.1) (Freund & Schapire 1996); this is also consistent with the approach adopted by Caruana et al. (2004).

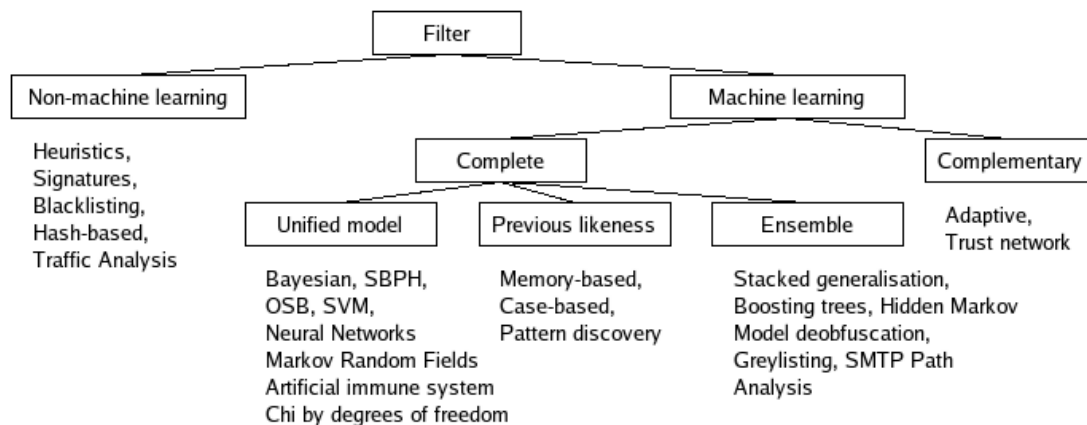


Figure 2.2: A classification of the various approaches to spam filtering discussed in section 2.4.

## 2.4 Spam filtering

Academic research into spam filtering approaches can be broadly separated into two primary categories: machine learning-based solutions, and static (or non-machine learning-based) solutions. The technologies seen in commercial and open source implementations can be similarly categorised. Figure 2.2 provides a brief taxonomy of the various approaches successfully evaluated in this domain. This review of spam filtering research primarily concentrates on filtering approaches, rather than providing a complete overview of the spam research field.

### 2.4.1 Machine learning approaches

Machine learning-based approaches can be broadly categorised (see figure 2.2) into complete and complementary solutions. Complementary solutions are designed to work as a component of a larger filtering system and only provide a partial solution to the spam problem. Complete solutions, as the name suggests, construct a comprehensive knowledge base to allow accurate classification of all incoming messages. Complete machine learning solutions take a variety of approaches: some aim to construct a unified model, some compare unseen email to previous instances (previous likeness) and others use a collaborative approach and combine different techniques (ensemble).

The application of the naive Bayes algorithm to spam filtering was initially researched by Sahami et al. (1998) and Pantel & Lin (1998). Its use was later popularised by Graham (2002), and now forms a key part of most commercial and open source spam filters<sup>2</sup>. It was arguably this research, and its associated success, that prompted the substantial amount of research that followed that evaluated other machine learning techniques in the spam filtering domain. This technique has also established itself as the de facto standard for evaluating the success of new approaches. Further research by Androutsopoulos, Koutsias, Chandrinou & Spyropoulos (2000) showed that a static keyword filter was outperformed by a naive Bayesian filter (across a range of performance metrics). The naive Bayes filtering approach, and many other machine learning approaches, address the key shortcomings of a static (non-machine learning) filter by generating an unknown rule set (to prevent a spam message being crafted to pass through the filter) and by adapting the rule set in response to misclassifications (allowing continuous improvements in accuracy).

The implementations based on the naive Bayes algorithm tend to be inconsistent with Bayesian probability theory<sup>3</sup> and often disregard the raw probabilities generated in favour of an artificial scoring system (Vaughan-Nichols 2003); clearly, these issues have not affected its success. The general approach

<sup>2</sup>Including the PreciseMail AntiSpam System in use at the University of Canterbury.

<sup>3</sup>Naive Bayesian statistics assumes that the occurrence of events are independent; e.g. that the word 'offers' is no more likely to follow the word 'special' than any other word. Clearly, this is not the case.

adopted by naive Bayesian filtering has been extended to two new techniques: sparse binary polynomial hashing (SBPH) (Yerazunis 2003), and orthogonal sparse bigrams (OSB) (Siefkes, Assis, Chhabra & Yerazunis 2004), both of which outperform the traditional naive Bayes approach by considering the inter-word dependence of natural language.

The collection of algorithms selected by Caruana et al. (2004) for the evaluation of their ensemble classifier have also been shown to perform well in the spam filtering domain. For example:

- $k$ -NN classifiers (based on the TiMBL implementation) were shown to be slightly inferior to a naive Bayesian filter, in terms of overall accuracy, but superior in terms of spam recall (Androutopoulos, Paliouras, Karkaletsis, Sakkis, Spyropoulos & Stamatopoulos 2000, Sakkis, Androutopoulos, Paliouras, Karkaletsis, Spyropoulos & Stamatopoulos 2001a).
- Support vector machines and boosted C4.5 decision trees were evaluated by Drucker, Wu & Vapnik (Sep. 1999), and were found to outperform Ripper (a rule-based learner) and Rocchio algorithms (a TF-IDF based learner), with SVMs preferred for their lower training time. Support vector machines and random forests (RF) were shown to outperform the naive Bayes approach by Rios & Zha (2004).
- Boosted decision trees (using the AdaBoost algorithm) were shown to outperform naive Bayes,  $k$ -NN and stacking algorithms (Carreras & Márquez 2001).
- Artificial neural networks were shown to outperform naive Bayes,  $k$ -NN, stacking and boosted decision trees (Clark, Koprinska & Poon 2003).

Other notable applications not built upon in this research include:

- Markov random field models were used to evaluate the importance of inter-word dependence (i.e. the context of a word or phrase) when classifying email (Chhabra, Yerazunis & Siefkes 2004). It was found that considering phrases (as attributes) up to five tokens long improved accuracy when compared to a standard word-based naive Bayesian classifier.
- Genetic algorithms were used to develop a filter based on the human immune system, and provided similar accuracy to other filters; however, it required fewer computational resources, making it attractive when processing time is at a premium (Oda & White 2003a, Oda & White 2003b).
- Chi by degrees of freedom tests are typically used to identify the author of a document; however, O'Brien & Vogel (2003) built a classifier using the test based on the premise that tens of millions of spam messages may be the work of a group of 150 authors (Ludlow 2002). It was shown to provide equal or better results when compared to a naive Bayes filter.
- A case-based reasoning approach maintains system knowledge in a collection of previously classified cases, which are then used to classify unseen instances (through the use of similarity metrics). Cunningham, Nowlan, Delany & Haahr (2003) use this approach to develop a classifier that can accommodate concept drift; new concepts identified in emails can be learned and unlearned after a period of time. An initial evaluation shows the classifier outperforms a traditional naive Bayes classifier.
- Stacking, another ensemble method, is used to combine a memory-based ( $k$ -NN) and a naive Bayes classifier (Sakkis, Androutopoulos, Paliouras, Karkaletsis, Spyropoulos & Stamatopoulos 2001b). Given that the two component classifiers tend to make uncorrelated errors and have shown to be accurate in previous research, it is unsurprising<sup>4</sup> that the stacked classifier outperforms both of its component classifiers. The gains experienced from stacking through classifier diversity suggests that other ensemble approaches would be successful in this domain.

---

<sup>4</sup>Given that both the conditions for a successful ensemble have been achieved (see the necessary and sufficient condition described in section 2.2).

## 2.4.2 Non-machine learning approaches

Heuristic (or rule-based) analysis has been, and in some cases remains, the predominant filtering technique (Process Software 2004). It uses regular expression-style rules to identify phrases or characteristics that are common to spam messages, and classifies the message based on the frequency and severity of the features identified. It is simple, fast, requires no training period and has a consistent level of accuracy; however, it is unable to adapt to new spam characteristics (and therefore requires regular updates to the rule set) and can be penetrated by the determined spammer with careful message crafting. Graham (2002) believes heuristic filters cannot achieve 100% accuracy; rather, as they approach this level, an unacceptable level of false positives will result.

Other traditional filtering techniques include blacklisting and signatures. Blacklisting blocks specific senders, either on a user or DNS level, and typically has a notorious rate of false positives (Snyder 2004). Signature-based filtering generates a unique hash value for each message and distributes the hash to all users of the filter; messages that match the hash will be discarded. Such filters have a very low level of false positives; however, they are unable to detect spam until the signature has been propagated throughout the network and can be easily circumvented by spammers by inserting a string of random characters (this will change the hash value of the message). Yoshida, Adachi, Washio, Motoda, Homma, Nakashima, Fujikawa & Yamazaki (2004) use a combination of hashing and document space density to identify spam, with excellent results (98% recall and 100% precision on a corpus of 50 million emails). Damiani, di Vimercati, Paraboschi & Samarati (2004) examine different hashing techniques that are robust against attempts to disguise a message (through the insertion of random characters, for example).

## Chapter 3

# Library-based ensemble classifier

This chapter gives an overview of this implementation of the ensemble classifier proposed by Caruana et al. (2004); this approach will be referred to as a ‘library-based ensemble classifier’ (LBEC), for lack of a better name. A comprehensive search has revealed no other published works based on this research. Differences between this implementation and the original will be discussed where appropriate; however, there are three general areas where differences may exist: optimisation measures (section 3.3), selection strategies (section 3.2) and model generation. The models generated for the library in this evaluation are discussed in section 2.1 and 2.3, while the exact parameters used with these models are discussed in section 5.2.

### 3.1 Training process

Three datasets are required to construct and evaluate a library-based ensemble classifier. The ‘training’ set is used to construct the models that populate the library. The ‘validation’ set (or ‘hill climbing’ set) is used to construct the ensemble itself; this dataset is used to evaluate ensemble performance, and to determine which model to add. The ‘test’ set is used to evaluate the performance of the resulting ensemble. The ratio between the sets adopted by Caruana et al. (2004) is 4:1:20 (training : validation : test). The impact of changing this ratio on performance is beyond the scope of this evaluation, and will therefore remain the same as that used by Caruana et al. (2004). The training set itself is relatively small; most machine learning algorithms are evaluated with 10-fold cross validation, where 90% of the data set is used for training and the remainder for testing.

The library-based ensemble classifier does not require any more test sets than a standard classifier. If a stand-alone classifier was to be constructed, a validation set would still be required for parameter selection. In this research, the validation set is crucial for identifying robust stand-alone models to test against the ensemble classifier. The ensemble classifier uses the validation set for parameter and model selection; this is consistent with the approach adopted by Caruana et al. (2004).

### 3.2 Selection strategies

The model selection procedure detailed in section 1.3, while fast and effective, has the tendency to over-fit to the validation set (Caruana et al. 2004). Over-fitting results in an ensemble that is adjusted to specific features of the training data set which have no meaningful relationship to the underlying function describing the data. This reduces the ensemble’s ability to generalise to instances not seen during the training period. Sections 3.2.1 and 3.2.2 review other selection strategies proposed by Caruana et al. (2004).

#### 3.2.1 Deterministic selection

Deterministic selection algorithms do not contain any element of randomness and, given the same input data, will return the same ensemble each time.



The `SELECTIONWITHOUTREPLACEMENT` strategy was discussed in section 1.3. It adds models, from the library, that maximise the performance of the ensemble; if no model improves ensemble performance, then the model that decreases performance by the least is added. On each iteration, the performance of the ensemble is recorded and the ensemble from the best performance iteration is returned. Once a model is added to the ensemble, it is no longer available for selection. Caruana et al. (2004) found that performance rapidly improves as the best models are added to the ensemble; however, performance begins to decline quickly as the best models have been used and models that hurt the ensemble must now be added. The loss in performance can be significant if the correct stopping point is not recognised.

`SELECTIONWITHREPLACEMENT` is a variation on the original method and is shown to greatly reduce the risk of ensemble performance declining. As its name suggests, models can be added to the ensemble more than once. Models that already form part of the ensemble can be re-added. This results in performance remaining relatively constant once peak performance has been obtained. It also allows implicit model weighting; the classifications by models present in the ensemble multiple times will have a greater impact on the final decision.

A slight modification was made to the `SELECTIONWITHOUTREPLACEMENT` and `SELECTIONWITHREPLACEMENT` algorithms as proposed by Caruana et al. (2004) by introducing a classification similarity test. The test compares the classifications made by two models on the validation set; a high similarity value indicates that the models make similar classification decisions. An arbitrary threshold level determines how similar two models are required to be before a rejection occurs. The similarity test is made between the candidate classifier (the classifier that is being considered for inclusion in the ensemble) and the most recently added classifier; therefore, the similarity test does not entirely prevent models being added multiple times, or similar models being added. It simply provides a slight bias on the ensemble building process towards diversity, which is a necessary and sufficient condition for effective ensembles.

Forward selection methods have the potential to over-fit the ensemble to the validation set, as previously discussed. By initialising the ensemble with the best  $n$  models in the library, a strong initial ensemble is formed. Each of the  $n$  models performs well, and the combination of the  $n$  robust models makes it more difficult for the ensemble to over-fit once forward stepwise selection begins. The sorted ensemble initialisation can be applied to both the `SELECTIONWITHOUTREPLACEMENT` and `SELECTIONWITHREPLACEMENT` algorithms. The `BESTN` algorithm populates the ensemble with the best  $n$  models from the library and disregards any further stepwise selection.

### 3.2.2 Non-deterministic selection

Non-deterministic selection algorithms inject a controlled amount of randomness into the selection process; therefore, the ensembles generated are not guaranteed to be the same if generated multiple times. Randomness can introduce models into the ensemble that would not ordinarily be selected. This effectively increases the diversity of the ensemble, one of the necessary and sufficient conditions of an effective ensemble (see section 2.2).

`RANDOMSELECTIONWITHREPLACEMENT` operates in a similar manner to the `SELECTIONWITHREPLACEMENT` algorithm; however, instead of selecting a model to add from the entire model library, this algorithm selects from a random subset of the model library. The `RANDOMBESTNWITHOUTREPLACEMENT` algorithm brings a similar random element to the `BESTN` algorithm:  $n$  subsets of the model library are randomly selected, and the best model added from each to form the ensemble (no further stepwise selection occurs).

As the model library increases in size, the probability of generating an ensemble that over-fits to the validation set increases; as more models are generated, it is more likely that a subset of models are located that accurately describe the random characteristics of the validation set (rather than the underlying function of the data). The `BAGGING` approach aims to reduce this risk by generating a series of ensembles, and returning the average of all the candidate ensembles as the final result. The `RANDOMSELECTIONWITHREPLACEMENT` algorithm is used to generate a set of unique ensembles. If a combination of  $m$  models from the library is known to over-fit, the probability of those  $m$  models appearing in a random bag of models is less than  $(1 - p)^m$ , where  $p$  is the fraction of models in the bag.

### 3.3 Optimisation

The library-based ensemble classifier process makes it trivial to construct an ensemble classifier that provides excellent performance on any metric the user might require. Ensemble selection is capable of optimising to any measure, including less common measures such as SAR, which makes it useful in a variety of situations. Eight different measures can be used during the ensemble building process:

**Accuracy (ACC):** This measure is calculated as the number of correct classifications as a percentage of the total number of classifications. Accuracy ranges between 0 and 1 and should be maximised.

**Root-mean-squared-error (RMS):** RMS goes beyond the simplistic correct/incorrect evaluation provided by accuracy and takes into account the magnitude of any errors that occur (Witten & Frank 2000). Mean-squared-error is the most commonly used measure when evaluating numeric prediction; however, root-mean-squared-error gives the same dimensions of the predicted value itself. RMS can be calculated as:

$$RMS = \sqrt{\frac{\sum_{i=0}^n (p_i - a_i)^2}{n}}$$

where  $n$  is the number of instances. RMS should be minimised.

**F-score (FSC):** FSC is the harmonic mean of recall and precision, at a given threshold (Caruana & Niculescu-Mizil 2004b). It is calculated as:

$$FSC = \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}}$$

**Average precision (APR):** APR is a measure used to evaluate rankings in information retrieval and provides an overall evaluation of ranking quality (Caruana, Joachims & Backstrom 2004a). It can be thought of as the area under the precision/recall curve. A variety of methods exist in literature for calculating average precision; this implementation uses that adopted by Caruana et al. (2004a) which defines APR as the average of the precisions at each of the recall values for which precision is defined. APR should be maximised.

**Precision/recall break-even point (BEP):** BEP is considered to be the point at which recall equals precision.

**Receiver-operating-characteristics area (ROC):** An ROC plot shows the true positive rate vs. the false positive rate as the prediction threshold<sup>1</sup> varies through all possible values (Witten & Frank 2000, Caruana et al. 2004a); the ROC metric is the area under this curve. ROC should be maximised: a value of 1.0 indicates perfect predictions, a value of 0.5 indicates random predictions, and a value of 0.0 indicates perfectly incorrect predictions (i.e. the model is effectively backwards). An alternative calculation of the ROC metric is based on sorting the data by the predicted values, and counting the number of swaps needed to order the data by class:

$$ROC = 1.0 - \frac{\# \text{ swaps}}{(\# \text{ positives}) \times (\# \text{ negatives})}$$

**Probability calibration (CAL):** CAL is based on reliability diagrams (Caruana & Niculescu-Mizil 2004b). All cases are ordered by their predicted value and instances 1–100 are put in the same bin. The percentage of true positives is then measured, and then the mean prediction is calculated for these cases. The difference between the mean prediction and the observed frequency is the calibration error for the bin. The same process is then repeated for instances 2–102, 3–103 etc, and the error is computed in the same way for each bin. CAL is the mean of each bin's calibration error.

---

<sup>1</sup>The prediction threshold is a fixed value which determines the class of a prediction; for example, if the prediction threshold was 0.5, predictions above 0.5 would be considered class 1, and predictions below the threshold would be considered class 0.

**Squared error, Accuracy and ROC area (SAR):** SAR is a composite metric, calculated as (Caruana & Niculescu-Mizil 2004b):

$$SAR = \frac{(ACC + ROC + (1 - RMS))}{3}$$

SAR is devised as a metric that can be used when the correct training metric is not known and leads to robust ensembles in this situation (Caruana et al. 2004).

All nine performance metrics are evaluated as ensemble optimisation measures in chapter 5, as different metrics will be appropriate under different usage scenarios and different learning methods do not necessarily perform equally well on all metrics (Caruana et al. 2004).

# Chapter 4

## System overview

### 4.1 WEKA machine learning environment

The implementation of the library-based ensemble classifier evaluated in this study was built inside the WEKA machine learning toolkit (Witten & Frank 2000). WEKA offered an enormous amount of support during the development of the classifier, including dataset processing, statistical evaluation, easily extensible graphical user interfaces and a robust command-line interface for scripting. Perhaps most importantly, it offered a vast array of different machine learning algorithms, all of which could be accessed via a uniform programming interface. The implementation language of WEKA (Java) and its associated libraries also made advanced functionality straightforward, such as distributed computation (see section 4.4 for details). The resulting library-based ensemble classifier plugin can now be made available for other WEKA users to evaluate the concept.

Furthermore, the implementation of the library-based ensemble classifier within an alternative environment to that used by Caruana et al. (2004) allows for a more rigorous evaluation. The validity of the underlying approach can be evaluated through the use of different algorithms and implementations.

### 4.2 Corpus builder

Machine learning algorithms implemented in the WEKA environment read training and test data from files in the attribute-relation file format (ARFF) (see figure 4.1 for an example). It describes a finite number of attributes, and an unlimited number of instances. Attributes can have nominal or numeric values and each instance provides a value for each attribute. The adoption of the WEKA environment for implementation therefore mandates the use of the format to describe individual emails. This section describes the process of converting email messages from raw text to the ARFF format.

The attribute space (i.e. the number of attributes) for plain text documents, such as email messages, is huge. Given that performance diminishes with each additional attribute to consider, and that WEKA

```
@relation 'my-dataset'  
  
@attribute '>' {0,1}  
@attribute 'size' real  
@attribute 'class' {0,1}  
  
@data  
0,50.0,0  
1,12.0,1
```

Figure 4.1: An example of the ARFF format.

requires a finite number of attributes, a method of identifying the most valuable attributes must be adopted. After the attribute set is identified, each email must be described in terms of the attribute set. Two key approaches exist: manual and automated.

A manual approach defines metrics independently of the document collection. For example, a soil research problem may measure soil acidity, or a spam filtering problem may measure the number of recipients. This approach uses domain knowledge to identify and interpret key parts of each document; it has the ability to recognise that not all text within the document should be treated equally, and that more significant conclusions can be drawn from considering the context of the text. This approach is hard to evaluate in the spam filtering domain, as it is difficult to identify the key attributes that can identify spam. Furthermore, if such a set of identifying characteristics existed, there would not be such a great demand for better spam filtering technologies.

An automated approach uses some measure to identify useful attributes from the collection of text documents. Attributes are generally word-level tokens or phrases and can be identified with a measure such as information gain (IG):

$$IG(X, C) = \sum_{x \in \{0,1\}, c \in \{spam, legit\}} P(X = x, C = c) \cdot \log \frac{P(X = x, C = c)}{P(X = x) \cdot P(C = C)}$$

where the tokens with the top  $n$  IG values used as the attribute set (Sakkis et al. 2001a). A number of different preprocessing steps can be performed before calculating IG values. For example, a lemmatizer could convert all words to their base form (e.g. ‘was’ becomes ‘be’), a stop list could be used to remove immaterial words, or different tokenisation characters could be used (e.g. ‘ ’, ‘!’, ‘,’ etc.) (Androutsopoulos, Koutsias, Chandrinos & Spyropoulos 2000). Phrases consisting of multiple tokens can also be identified as attributes; however, the number of potential phrases that can be extracted is almost infinite and therefore would require some type of bound.

The impact of the various approaches described here is beyond the scope of this research; therefore, a simple approach has been adopted. Each email message in the training set is tokenised using the space character and tokens that occur less than ten times in the corpus are discarded. The information gain value for each token is then calculated as described above and the top  $n$  values identify the attributes for the data set. It is important to note that attribute identification should only use the training set, otherwise some implicit knowledge of the test set yet to be seen would be provided to the learning algorithm. Each email in the entire dataset is then described in terms of the attribute set. Each attribute is binary: either the token is present or not in the email and the frequency of the attribute in the message is not considered. This approach is consistent with most literature, including the initial paper in naive Bayesian filtering by Sahami et al. (1998).

The corpus builder was built in C# 2.0, using Microsoft Visual Studio 2005 Beta 2, and used ADO.NET for email extraction where appropriate.

### 4.3 WEKA classifier plugin

The library-based ensemble classifier plugin for WEKA allows the user to interact with it in the same manner as any other WEKA classifier. It can be manipulated from both the command line and from the GUI, and produces the same statistical evaluation of results as provided by all other classifiers.

Figures 4.2, 4.3, 4.4 and 4.5 provide an indication of the graphical interface that allows users to interact with the classifier. Figure 4.2 lets users load and visualise the validation dataset, while figure 4.3 allows the user to load test set, configure the classifier and start an evaluation. Figure 4.5 shows the configuration window for the library-based ensemble classifier. It allows the user to configure the selection strategy, as well as the location of the model library (which needs to be constructed before an evaluation can take place). Figure 4.4 provides an example of the configuration window of a typical selection strategy. It allows the user to alter the parameters associated with the strategy, and choose an optimising metric to use during the selection process.

The user can select a number of alternative selection strategies, each of which is in line with those discussed in section 3.2. The architecture of the system allows additional selection strategies to be easily integrated into the library-based ensemble classifier. The currently implemented strategies are:

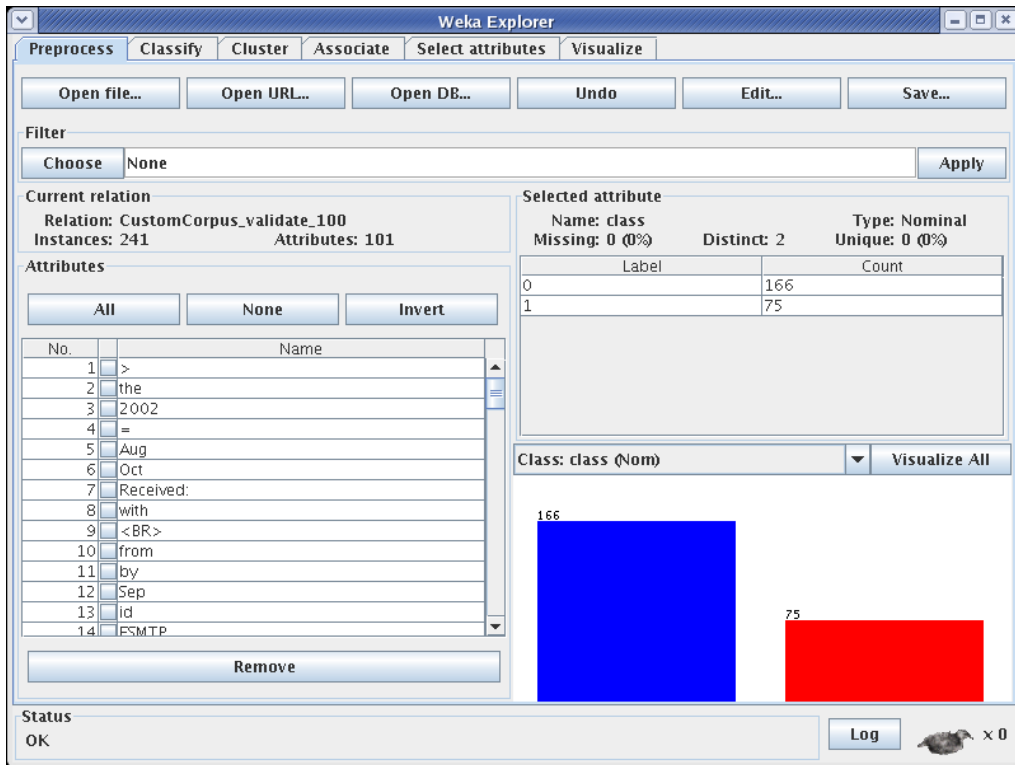


Figure 4.2: A screenshot of the WEKA dataset preprocessing screen, where the user loads the validation dataset.

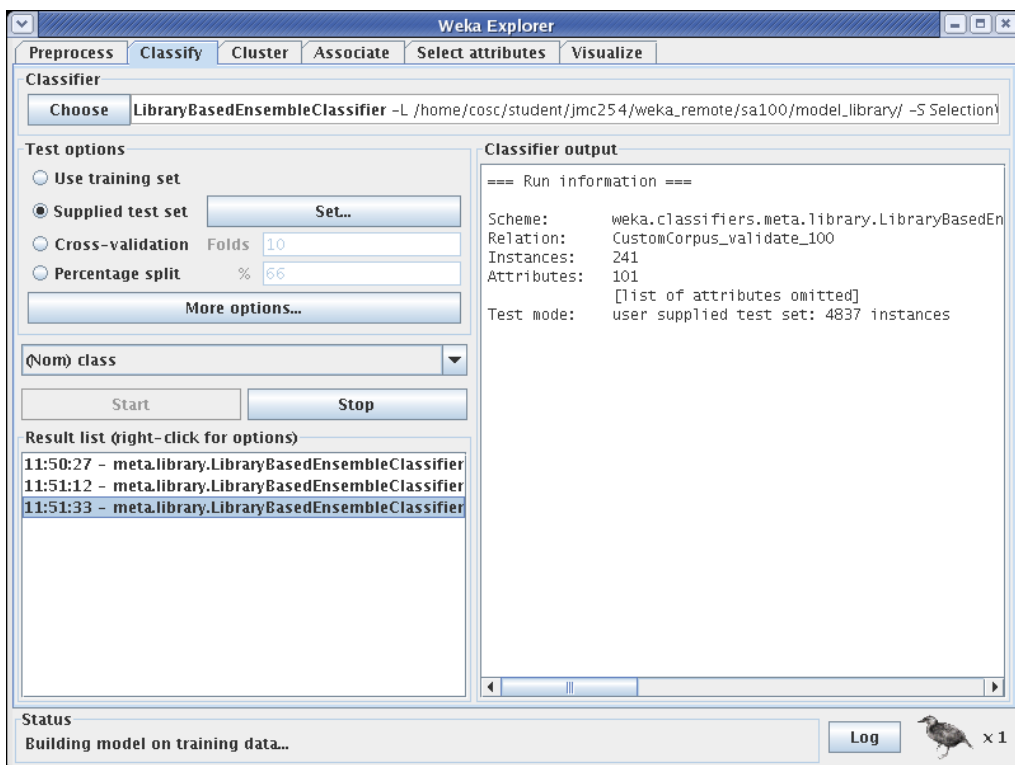


Figure 4.3: A screenshot of the WEKA dataset classifier screen, where the user can evaluate the classifier.

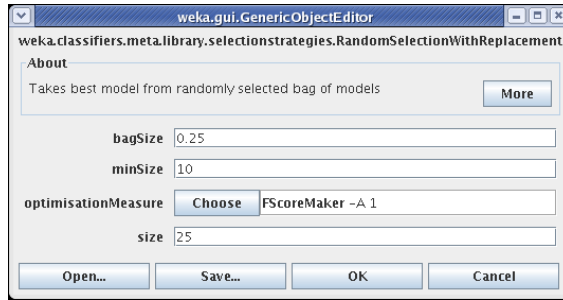


Figure 4.4: A screenshot of a WEKA configuration screen, where the user can configure the selection strategy they have selected.

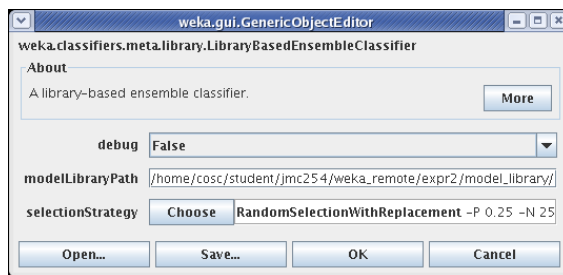


Figure 4.5: A screenshot of a WEKA configuration screen, where the user can configure the library-based ensemble classifier.

**SELECTIONWITHOUTREPLACEMENT and SELECTIONWITHREPLACEMENT:** Three parameters are defined for these algorithms: a maximum value describes how many iterations the algorithm should continue adding models before returning the best performing ensemble, a minimum value allows the ensemble to be seeded with the best  $n$  models from the library (as defined by their performance on the validation set) and a similarity value determines the threshold for model rejection.

**BESTN:** One parameter is defined for this algorithm and sets the  $n$  value for the evaluation.

**RANDOMSELECTIONWITHREPLACEMENT and RANDOMBESTNWITHOUTREPLACEMENT:** Three parameters are defined for these algorithms: a bag size describes how large a subset of the library should be used for selection, a maximum value describes how many iterations the algorithm should continue adding models before returning the best performing ensemble and a minimum value describes the minimum size of the resulting ensemble.

**BAGGING:** This algorithm uses the **RANDOMSELECTIONWITHREPLACEMENT** strategy to generate a set of ensembles and it therefore takes the bag size, maximum value and minimum value parameters as described for this strategy. It also takes a ‘repeats’ parameter, which describes how many candidate ensembles to construct before returning the average ensemble. To find the average ensemble, the frequency of each model present in the generated ensembles is calculated and divided by the total number of models in all of the ensembles. These numbers are then rounded to the nearest integer; this number indicates the number of models that should be included in the final ensemble. In the case where no model is present, 0.05 is added to the mean frequency; this process continues until a suitable number of classifiers are inserted into the final ensemble.

All strategies also take an additional parameter that determines which optimisation metric to use.

All of the metrics discussed in section 3.3 are available to optimise the performance of the ensemble under any selection strategy. Basic metrics, such as accuracy, f-score and root-mean-squared-error, are implemented in the Java environment. The remaining metrics are implemented in the **PERF** package.

The PERF system was written by Rich Caruana for KDD2004 and is now publicly available. It calculates the ROC, AVP, BEP, CAL and SAR metrics for the WEKA plugin. The package itself was designed as a stand-alone C application; therefore, it was altered to accept library calls and accessed via the Java Native Interface. The complexity of the code and the benefits that would have been realised by implementing the metrics in Java did not mandate a reimplementaion in Java; hence the use of JNI. The C code for the PERF package does not depend on any unique Linux functionality; therefore, it should be easily ported to other platforms and therefore not restrict the platform-independent nature of WEKA.

The classification of an instance is determined by a majority vote of the classifiers in the ensemble. In the event of a tie, the classifier that was most recently added is removed; this process continues until a classification decision is reached.

The classification decisions of each instance in the validation set for each classifier are permanently recorded. Each time a model is added to the library-based ensemble classifier, the decisions of each of the component classifiers are used to determine which classifier to add and are also used to determine the performance of the ensemble on the validation set. The performance of each classifier on the test set is not similarly persisted; only the classifiers within the ensemble will be required to classify the test set. This caching strategy significantly improved performance, as the component classifiers are no longer deserialised to classify the validation set, as the results are already available; only the classifiers that form part of the ensemble will be deserialised.

## 4.4 WEKA model builder

The library builder is similarly integrated into the WEKA toolkit; however, users interact with this tool from the command line, rather than through a graphical user interface. Given the scope of the model building task<sup>1</sup>, a distributed generation scheme was required to build the models in a realistic time frame. Given that WEKA already used the Java Remote Method Invocation (RMI) package for distributed classifier evaluation, RMI was also used to co-ordinate the remote generation of classifier models.

A Master-Slave architecture was designed to orchestrate the construction of the library. An initial script initialised Java on the required remote machines (nodes), which was followed by the initialisation of the master node. The master node would contact each remote node in turn and require each remote node to associate itself to it. Once association was complete, each remote node would request a model build task; this would continue until all the necessary models were constructed. At this point, the system would terminate.

To generate the required quantity of different models, the parameters for each machine learning algorithm are varied and exposed to the training data set. Every classifier within the WEKA toolkit (including the library-based ensemble classifier plug-in) can be configured through the use of command line options; therefore, a system was developed to describe how command line options could vary for each classifier type. Various parameter structures can be described using the available Java classes: parameters with a range of options (e.g. `-F 0, 1, 2`), flag parameters (e.g. `-R`), empty parameters, mutually exclusive parameters (e.g. `-F` or `-R 1, 2, 4`) and independent parameters (e.g. `-R -F -T 1, 2, 5`). These classes could be then combined to describe any sequence of parameters (e.g. `-F` or `(-R -G (-T 1, 2 or -G 2, ))`). A tree is then constructed to represent the arrangement of the parameters, and all permissible permutations of the described parameters are generated and used to determine which models are to be generated, and with which parameters. In an ideal implementation, a simple language could be used to describe these relationships.

---

<sup>1</sup>Caruana et al. (2004) used a cluster of ten Linux boxes to generate 2000 models over 48 hours.



# Chapter 5

## Evaluation

The evaluation of the library-based ensemble classifier can be divided into three key steps:

**Classifier validation:** (section 5.2) Before the classifier can be extended, or applied to spam, validation and verification of the implementation must occur. This evaluation takes the datasets used by Caruana et al. (2004) and compares the results in an effort to gauge the strength of the implementation.

**Classifier extension:** (section 5.3) The classifier implemented and evaluated by Caruana et al. (2004) was only capable of learning datasets with a binary class (i.e. each instance belonged to either class 0 or 1) and used only large datasets. This implementation is capable of learning multi-class data; this is an important development over the original classifier and is evaluated to determine its performance. Smaller datasets will also be evaluated.

**Application to spam:** (section 5.4) After the implementation has been validated, the classifier will be evaluated in the spam filtering domain.

The first section (5.1) provides information common to all three evaluations.

### 5.1 Overview

#### 5.1.1 Experimental setup

The following algorithms and parameters are used to generate between 1800–2000 models for each experimental dataset. Not all algorithms are applicable to all datasets used, as some cannot be applied to particular attribute or class types (e.g. numeric or nominal).

**AdaBoostM1:** all the decision trees outlined here are boosted, with a range of iterations between 1 and 128. The percentage of weight mass for training is varied between 80 and 100%. The default parameters for each tree are used.

**ADTree:** the number of boosting iterations ranges from 5 to 20 and all four node expansion schemes (all, weight, z\_pure and random walk) are used.

**Bagging:** all of the decision trees outlined here are bagged, with a range of iterations between 5 and 25. The bag size is varied from 25 to 100% of the training set size. The default parameters for each tree are used.

**IBk:**  $k$  values ranged from 1 to the number of instances in the training set. Three types of distance weighting were also used: no weighting (standard), inverse weighting and 1– distance weighting. Window sizes of 0, 50, 100, 150 and 200 were also used.

**Id3:** no options are available for this tree.

**J48:** pruning confidence levels were varied between 0.05 and 0.4 and the minimum number of instances per leaf was varied between 2 and 7. Unpruned trees were also evaluated, along with no subtree raising, reduced error pruning and Laplace smoothing for predicted probabilities.

**LMT:** the minimum number of instances for splitting was varied between 5 and 25. Other options evaluated include splitting on residuals rather than class values, cross-validation for boosting at all nodes and the use of errors on probabilities instead of misclassification errors for the stopping criteria.

**M5P:** the minimum number of instances per leaf was varied between 2 and 8. Other options evaluated include the use of unsmoothed predictions, unpruned trees and the use of a regression tree (rather than a model tree).

**MultilayerPerceptron:** the learning rate was varied between 0.2 and 0.4 and the momentum between 0.0 and 0.9. The number of hidden units was varied between 1 and 128 and included some two level networks. No weight decay or early stopping is used; rather, the nets are stopped at epochs ranging from 50 to 500 to ensure some nets under-fit or over-fit.

**REPTree:** the number of instances per leaf was varied between 2 and 4, the minimum splitting variance between  $1.0e-5$  and  $1.0e-1$ , the maximum tree depth between 20 and unlimited, and the number of folds for reduced error pruning between 2 and 4. Trees with pruning disabled were also produced.

**SMO:** linear and polynomial (order two and three) support vector machines have complexity constants between  $1.0e-8$  and  $1.0e2$ . The polynomial kernels are evaluated with and without feature space normalisation and the use of lower order terms. Radial basis function kernels are evaluated with the same range of complexity constants and have a gamma value between 0.001 and 2.0.

The models, and their parameters, were selected to mimic the model libraries constructed by Caruana et al. (2004); as was previously mentioned, the effect of the model library composition is beyond the scope of this research.

Each dataset will be split into the training, validation and test sets in the ratio set out by Caruana et al. (2004) (4 : 1 : 20). The dataset splitting process used proportionate stratified sampling, which ensured the class ratio seen in the full data set would be replicated in each individual dataset. The instances contained in each individual dataset were mutually exclusive.

All selection strategies will be evaluated and the validation set used to tune strategy parameters. All optimisation measures will also be reviewed, in combination with each selection strategy. The following strategies and parameters are evaluated:

**SELECTIONWITHOUTREPLACEMENT:** Similarity values of 0.85, 0.90, 0.95 and 1.0 are tested; a value of 1.0 disables the test (as no two models will ever be more than 100% similar).

**SELECTIONWITHREPLACEMENT:** The same similarity values as were used by the previous algorithm are also applied to this one, in conjunction with a range of seed values (where the best  $n$  models seed the ensemble) of 0, 3, 6, 9 and 12.

**BESTN:** This algorithm is evaluated by adding the top 5, 10, 15, 20 and 25 models to construct the ensemble.

**RANDOMBESTNWITHOUTREPLACEMENT:** Bag sizes of 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 and 0.9 are evaluated.

**RANDOMSELECTIONWITHREPLACEMENT:** The same bag sizes as tested by **RANDOMBESTNWITHOUTREPLACEMENT** are evaluated with this algorithm.

**BAGGING:** Bag sizes of 0.2, 0.4, 0.6 and 0.8 are evaluated, in conjunction with 5, 10, 15, and 20 repeats (i.e. number of candidate ensembles to average to find the final ensemble).

Where appropriate, the minimum ensemble size was set to zero, and the maximum ensemble size was set to fifty. This provided a minimum of constraints on growth and ensured the best performance ensemble would be returned.

The validation set is used for parameter selection, in addition to ensemble construction. A number of different ensembles will be generated, using the various strategies outlined here. Performance on the validation set will be used to identify the ‘best’ ensemble and stand-alone models. These models will then be applied to the test set, and their results will be used in the experimental analysis. These models may not be the best performers on the test set, but the validation set represents the only strategy available to the end user to identify the ensemble they wish to use for prediction.

## 5.1.2 Reporting

The complete results for each data set have been placed in appendix A, with summary statistics presented here. Performance is characterised both in terms of both percent reduction in loss and normalised performance. To calculate percent reduction in loss, each metric is first converted, if necessary, to loss where 0 indicates perfect performance and 1 indicates terrible performance. Perfect prediction results in zero loss, and occurs when  $ACC = BEP = FSC = ROC = APR = SAR = 1$  and  $RMS = CAL = 0$ . For example, if FSC increases from 0.7 to 0.75, the losses are 0.3 and 0.25 respectively, and the percentage reduction in loss is 7.14%. Percentage reduction in loss does introduce some bias, as it gives more emphasis to reductions at a relatively low level of loss.

Absolute scores are difficult to use for comparisons, as they do not allow comparisons of measures with different behaviours (e.g. perfect  $ACC = 1$  while perfect  $RMS = 0$ ) and the comparison of the results of different datasets (e.g. excellent performance on one dataset maybe 98%, but on another 85%). Normalised scores allow averaging across metrics and problems, as all performance figures are converted to a [0,1] scale. Baseline performance (0.0) is generally calculated for each metric by predicting  $p$  for every case in the test set, where  $p$  is the percentage of positives in the training set (i.e. classify every instance as the most common class) (Caruana & Niculescu-Mizil 2004b); however, ROC baseline is independent of the data set and is set at 0.5. The top value in the range represents the best recorded performance of a library-based ensemble on the test set, regardless of its performance on the validation set.

The reported values for the library-based ensemble classifier are recorded as LBEC, bagged trees as BAG, support vector machines as SVM, artificial neural networks as ANN, boosted decision trees as BST-DT, nearest neighbour as KNN and decision trees as DT. LBEC-T refers to the classifier’s performance on the test set. LBEC-V is also in places and records the classifier’s performance on the validation set. Where both LBEC-T and LBEC-V are reported, they refer to the same ensemble.

## 5.2 Classifier validation

### 5.2.1 Aim

The aim of this experiment is to verify and validate the implementation of the library-based ensemble classifier plugin for WEKA, as well as the effectiveness of the various optimisation metrics and selection strategies. This experiment will also attempt to determine the optimal operating parameters for the library-based ensemble classifier.

### 5.2.2 Hypothesis

The strength of the library-based ensemble classifier is based on the diversity in the inductive bias used by each of the component learning algorithms. While the learning algorithms and their implementations are different in this implementation to that used by Caruana et al. (2004), the underlying diversity in inductive bias remains. It is therefore hypothesised that this implementation will perform similarly. It is difficult to estimate the performance of individual selection algorithms; if forward stepwise selection forms ensembles that over-fit to the validation set, then it is likely that more advanced algorithms (such as seeded initiali-

	ACC	FSC	ROC	AVP	BEP	RMS	CAL	SAR	MEAN
ADULT	1.37	5.26	-11.99	27.51	-1.64	1.23	27.15	-0.43	6.06
COVTYPE	-0.12	-13.67	0.05	-6.95	-3.48	0.11	0.05	-0.42	-3.05
LETTER.p1	-33.85	-27.29	-22.40	-10.12	-3.28	-17.83	71.21	-18.69	-7.78
LETTER.p2	2.48	-6.66	-12.87	-26.61	-17.12	-5.23	-7.98	-14.33	-11.04
MEAN	-7.53	-10.59	-11.80	-4.04	-6.38	-5.43	22.61	-8.47	-3.95

Table 5.1: Percentage reduction in loss provided by the library-based ensemble classifier when compared with the best individual classifier (as determined by the validation set) for the general datasets.

sation, random selection with replacement and bagging) will perform best. If no over fitting occurs, then forward stepwise selection is likely to perform best.

### 5.2.3 Method

Caruana et al. (2004) use several publicly available datasets in their initial evaluation of their library-based ensemble classifier. A partial validation of this implementation will be achieved if this classifier produces similar results as the original implementation. Six different datasets were used to validate the original implementation; however, only three could be found in the public domain. All three datasets can be found as part of the UCI machine learning repository (Newman, Hettich, Blake & Merz 1998). They are:

**ADULT:** is a binary class dataset, which predicts whether income exceeds \$50,000 per year based on census data. It contains 48,842 instances and 14 attributes (6 continuous and 8 nominal). It includes missing attribute values.

**COVTYPE:** is a multi-class dataset which predicts the forest cover type from cartographic variables. It has been converted into a binary dataset by treating the largest class as class 1, and the remainder as class 0. The full dataset contains 581,012 instances and 54 attributes. A 10% subset is used for the purposes of this evaluation to bring the dataset in line with sizes of the ADULT and LETTER sets.

**LETTER:** is a multi-class dataset, which predicts which English character is present in a black-and-white image. It has been converted into a binary dataset in two ways: firstly, the letter ‘O’ is treated as class 1 (and the remainder class 0), and secondly, letters 1–13 are treated as class 0 and letters 14–26 are treated as class 1. The resulting datasets are respectively known as LETTER.p1 and LETTER.p2. The datasets contain 20,000 instances and 16 attributes.

### 5.2.4 Results

#### Overview

This section gives an overview of the best performance that can be expected from the library-based ensemble classifier in a general topic domain.

Tables A.1, A.2, A.3 and A.4 show the performance of the best classifiers (as identified by the validation set) when applied to the test set. The result of the best model of a particular type, under the given metric, is reported; therefore, each data point is potentially from a different model. The validation set allows the identification of models that are likely to provide the best performance, in terms of the given metric, on the test set. Tables A.5, A.6, A.7 and A.8 provide normalised performance scores for each dataset.

Table 5.1 compares the performance of the best library-based ensemble classifier (as determined by performance on the validation set) on the test set against the performance of the best individual model (as determined by performance on the validation set) in terms of percent reduction in loss. Table 5.2 gives the average performance of the library-based ensemble classifier and the best individual models (as determined by performance on the validation set) over all four datasets. Table 5.3 provides the mean composition of each ensemble generated for this evaluation, over all four data sets. Each score is the average proportion that each algorithm represents in an ensemble optimised for a particular measure.

	ACC	FSC	ROC	AVP	BEP	RMS	SAR	MEAN
LBEC-T	0.983	0.984	0.986	0.910	0.968	0.977	0.986	0.971
BAG	0.896	0.906	0.858	0.890	0.869	0.849	0.880	0.878
SVM	0.857	0.924	0.906	0.790	0.871	0.808	0.893	0.864
ANN	0.825	0.903	0.885	0.722	0.861	0.764	0.837	0.828
BST-DT	0.950	0.955	0.920	0.837	0.945	0.928	0.940	0.925
KNN	0.813	0.831	0.887	0.864	0.894	0.761	0.874	0.846
DT	0.842	0.892	0.825	0.841	0.858	0.780	0.858	0.842

Table 5.2: Normalised scores for the best models of each type and for ensemble selection, averaged over the four problem datasets.

	ACC	FSC	ROC	AVP	BEP	RMS	CAL	SAR	MEAN
BAG	0.052	0.048	0.047	0.037	0.050	0.049	0.125	0.046	0.057
SVM	0.143	0.175	0.242	0.206	0.184	0.169	0.179	0.220	0.190
ANN	0.170	0.187	0.178	0.215	0.195	0.185	0.279	0.185	0.199
BST-DT	0.357	0.278	0.265	0.332	0.269	0.309	0.247	0.273	0.291
KNN	0.138	0.176	0.151	0.170	0.164	0.152	0.125	0.160	0.155
DT	0.139	0.135	0.118	0.041	0.138	0.136	0.045	0.116	0.109

Table 5.3: Average model composition using the LETTER, ADULT and COVTYPE datasets.

### Selection strategies

Figure 5.1 shows the performance of the selection with and without replacement strategies when applied to the LETTER.p1 and LETTER.p2 data sets. The performance of the ensemble is recorded as additional models are added; both strategies would return the subset with the lowest level of error.

The similarity threshold introduced to the selection with replacement algorithm has been evaluated in table 5.4. The scores reported in this table are the test results of the best performing ensembles (as determined by the validation set) as generated by the selection with replacement strategy.

The selection with replacement algorithm was also evaluated with varying levels of seeded initialisation. Seeded initialisation starts the ensemble with the best N models (as determined by their performance on the validation set) and then uses forward stepwise selection to maximise performance. Caruana et al. (2004) recommends between five and twenty-five models be used to seed an ensemble before forward stepwise selection begins; therefore, the number of models used to seed the ensemble was varied between two and twenty-four in increments of two and tested on the LETTER.p1 and LETTER.p2 datasets. No accurate and significant relationship was found, either through linear regression or visual observation; however, the overall trend was increasing (i.e. performance generally increased with a larger seed).

Figure 5.2(a) shows the accuracy provided by the BestN algorithm as N is varied. The performance value reported for each N value is determined by the performance of the top BestN ensemble as determined by the validation set. Normalised accuracy figures are used.

An analysis was also performed to identify the relationship between the bag size (i.e. percentage of the model library) used in the random best N without replacement and the random selection with replacement algorithms and average performance recorded. Given the random nature of the output, multiple ensembles

	LETTER.p1	LETTER.p2	ADULT	COVTYPE
0.85	94.826	98.624	95.044	98.466
0.9	90.035	98.142	97.828	98.466
0.95	98.219	99.547	95.514	98.466
1	97.221	98.734	95.514	98.466

Table 5.4: Effects of the similarity value using the selection with replacement strategy and optimising to accuracy.

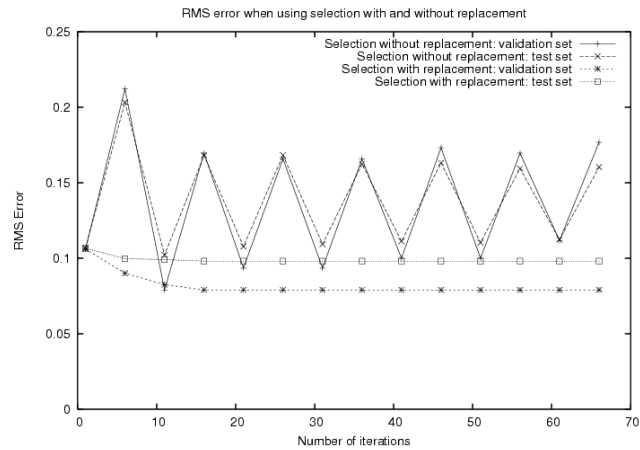
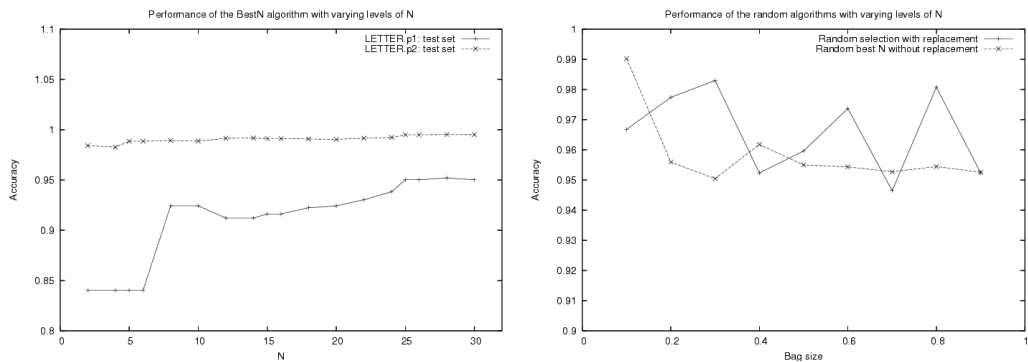


Figure 5.1: The performance of the selection with and without replacement algorithms as additional models are added. The LETTER.p1 and LETTER.p2 data sets are used.



(a) The normalised accuracy of the BestN selection strategy as the value of  $N$  is varied, on the LETTER.p1 and LETTER.p2 test sets. (b) Average normalised accuracy for the four general data sets under the random selection strategies as the bag size is varied.

Figure 5.2: Evaluation of the BestN and random selection strategies.

	L.p1	L.p2	COVTYPE	ADULT	MEAN	
Selection without replacement	1	1	3	1	1.5	1st
Selection with replacement	3	2	1	3	2.25	2nd =
BestN	6	6	6	5	5.75	6th
Random selection with replacement	2	3	2	2	2.25	2nd =
Random best-n without replacement	5	4	5	4	4.5	4th
Bagging	4	5	4	5	4.75	5th

Table 5.5: Average validation set accuracy rank for generated ensembles, by selection strategy (L.p1: LETTER.p1, L.p2: LETTER.p2).

	L.p1	L.p2	COVTYPE	ADULT	MEAN	
Selection without replacement	3	1	3	2	2.25	1st
Selection with replacement	4	4	4	3	3.75	4th
BestN	2	3	2	5	3	2nd
Random selection with replacement	1	6	6	4	4.25	5th
Random best-n without replacement	6	5	1	1	3.25	3rd
Bagging	5	2	5	6	4.5	6th

Table 5.6: Average test set performance rank for generated ensembles, by selection strategy (L.p1: LETTER.p1, L.p2: LETTER.p2).

were generated at each level, and the averages used. Normalised accuracy figures were exposed to linear regression and visual inspection, but no identifiable relationship could be determined. Figure 5.2(b) gives the normalised accuracy scores with a varying bag size, averaged over all four general datasets.

The relationship between normalised accuracy and the number of bagging iterations was similarly investigated. The number of iterations was varied between two and thirty, in increments of two over the two LETTER datasets. Neither the individual accuracies or the average trend showed an identifiable relationship (either through linear regression or visual inspection).

Tables 5.5 and 5.6 show the average performance of each ensemble building strategy. Each ensemble built is ranked in terms of its performance on the validation and on the test set, giving an average rank for each strategy. These average ranks are then used to rank the performance of each strategy (i.e. the rank of 1 for selection without replacement means that this strategy had the highest average rank, not that the average rank of the ensembles built with this method were, on average, ranked 1st).

### 5.2.5 Discussion

A review of the percentage reduction in error for each problem reveals results that are broadly similar to those reported by Caruana et al. (2004). Both the LETTER.p1 and LETTER.p2 data sets record excellent performance under the library-based ensemble classifier, with their respective average percentage reductions in loss (-7.78% and -11.04%) comparable to that previously reported (-22.96% and -6.15%). COVTYPE also performs similarly, with an average percentage reduction in loss (-3.05%) similar to that seen by Caruana et al. (2004) (-2.17%). Note that the figures reported from the Caruana et al. (2004) paper have been adjusted to reflect the difference in measures used (the original paper used a range of ten different metrics).

The performance of the ADULT dataset is not consistent with that seen in the previously reported results. This implementation of the library-based ensemble classifier saw an average of a 6.06% increase in loss, compared with the previous result of a 3.34% decrease in loss. A closer examination of the absolute results for this data set reveal significant decreases from the performance of the ensemble in the validation set to the performance of the ensemble in the test set (see table A.3). For example, FSC in the validation set is 0.9263, and drops to 0.6616 in the test set (a 29% decrease); validation/test set differentials of this magnitude are not observed in the other four datasets and can likely be attributed to the ensemble strategy over-fitting to the validation set.

A comparison of the average normalised scores to those previously reported by Caruana et al. (2004)

shows that the scores reported in this study are higher for both the ensembles and the individual classifiers. It should be noted that the averaged normalised scores previously reported were averaged over seven, rather than four, datasets; however, it seems reasonable to conclude a difference may exist. Two possibilities exist: either the classifiers provided by WEKA are more accurate than those used by Caruana et al. (2004) or the performance of their ensemble classifier is better than this implementation (this would increase the top end of the normalised performance scale). It is difficult to determine the actual cause as insufficient data is provided by the original publication.

The average composition recorded in table 5.3 shows a balanced selection of models is included in each ensemble. This suggests that the generated ensembles are suitably diverse in the range of inductive biases they include and fulfill the necessary and sufficient condition of diversity for good ensembles. Boosted decision trees, on average, represent a substantially higher proportion than any other model type. This is understandable given their generally excellent performance; this model type typically ranks well amongst the other component algorithms. Model composition details are published by Caruana et al. (2004) for the ADULT and COVTYPE datasets only. When compared to the compositions recorded in these experiments, some similarities emerge: ADULT has a preference<sup>1</sup> for SVMs, ANNs and trees in both implementations, and COVTYPE has a preference for boosted decision trees and  $k$ -NN models.

The performance and composition results from this evaluation suggest that this implementation of the library-based ensemble classifier is robust. In most cases, the performance gains recorded by Caruana et al. (2004) have been realised; however, some concerns exist surrounding the ADULT data set and the range of normalised scores. Given the different implementation environment, and the subtly different inductive biases contributed by each algorithm, the results seen here are broadly consistent with expectations. It is therefore reasonable to conclude that this system is a valid implementation of the library-based ensemble classifier concept.

The similarity value introduced in this implementation that was used by the selection with replacement strategy in an attempt to increase diversity seems to be unnecessary and ineffective. Table 5.4 shows the effects of the similarity value on classification accuracy: the results are unpredictable, and no observable relationship exists. It is therefore reasonable to conclude that the diversity obtained through simple forward stepwise selection is sufficient, and no safeguards are required to prohibit the selection strategy including similar models.

Forward stepwise selection, through the selection without or the selection with replacement algorithms, adds models to the ensemble in an effort to maximise performance. It returns the ensemble that performs best on the validation set, which may be a subset of the resulting ensemble (for example, if the best recorded performance occurs when 5 models have been added, it is this ensemble that will be returned, but the stepwise selection continues until 50 models have been added). Therefore, the strategy of forward stepwise selection suffers from a potential problem: how many steps/iterations should occur before returning the best ensemble found? An additional test set could be formed to determine when to stop adding models to the ensemble; however, as can be seen in figure 5.1, the selection with replacement algorithm flattens the curve to such an extent that this is no longer necessary. The error measured by selection without replacement varies significantly, and it is difficult to estimate an appropriate stopping point; however, this is not the case with selection with replacement, as the optimal level of error is quickly reached and does not subsequently deviate from this level. It is for this reason that the selection with replacement is in some ways more reliable than selection without replacement, as the optimal level is almost guaranteed to be found within a reasonable time span.

The performance of the BestN selection strategy is evaluated in figure 5.2(a). It is reasonable to assume that the ensemble performance will increase as additional top models are added; however, it is not entirely clear to which extent this relationship holds. Caruana et al. (2004) suggest adding between 5 and 25 of the top models for seeded initialisation; this figure seems entirely reasonable, as it appears there are a sufficiently high number of accurate library classifiers that can be added without negatively affecting performance. A similar investigation was conducted to evaluate the effect of seeded ensemble initialisation with the selection with replacement strategy; again, no distinct relationship could be detected, but the overall trends was upwards. This is consistent with Caruana et al.'s (2004) recommended number of models to seed (5–25).

---

<sup>1</sup>i.e. those models that have the most weight in the average ensemble composition.



The investigation into the relationship between the bag size and accuracy recorded (see figure 5.2(b)) conflicts with the results reported by Caruana et al. (2004). Their results indicate a far stronger relationship, where the recorded accuracy improves and is relatively stable between a bag size of 0.2 and 0.9; clearly, no such stable relationship exists in figure 5.2(b). In some respects, it is unsurprising that a random process produces such results; however, the consistency of results would depend on the number of accurate models within the library. If it is likely that an ensemble can be randomly selected without including some highly accurate models, or with a large subset of models with average performance, the overall performance could appear to be quite random. If the model library is somewhat more consistent and accurate, it is less likely that an ensemble could be formed with poorly performing models. This potentially explains the difference: if the variance in accuracy is higher in the libraries used by this system than what is used by Caruana et al.'s (2004), then the random selection strategies are likely to be similarly more variable in their performance.

The Bagging selection strategy was evaluated in terms of the number of bagging iterations; this yielded no identifiable relationship. This suggests the averaging process returns a similar ensemble, in terms of classification ability, regardless of the number of iterations; this implies that each ensemble generated after each iteration is also likely to be similar.

The average ranks of the selection strategies are displayed in tables 5.5 and 5.6 for the validation and test sets respectively. It appears that the simplest strategies perform the best: the more complex, non-deterministic strategies are generally beaten by simple forward stepwise selection and BestN. The selection with replacement algorithm (with and without seeded initialisation) tends to over-fit to the validation set, leading to relatively poor performance on the test set. The BestN strategy forms robust ensembles by simply taking a set of well-performing models; this leads to relatively poor performance on the validation set, but the robustness and strength of the individual models lead the ensemble to perform relatively well on the test set. The concerns regarding the variability of the selection without replacement algorithm previously discussed seem to be unfounded; the upper limit of 50 forward stepwise selections appears to be sufficient for the strategy to identify a suitable ensemble. The poor performance of the non-deterministic algorithms can be partially attributed to their random nature, as it introduces two possibilities: the potential for a better than average ensemble, and the potential for a worse than average ensemble. The variability of the random algorithms is likely to impact their ranking, as they are somewhat less reliable than the typical deterministic systems.

Somewhat surprisingly, the rankings of each strategy should be of little interest to the user. Given that validation set can be used for parameter selection and ensemble construction, it takes a relatively short amount of time to generate a series of ensembles using different strategies. The best ensemble on the validation set can then be applied to the test set. The rankings of the selection strategies bear no resemblance to the models that perform best on the validation set or test set; the ability of non-deterministic algorithms to produce potentially better ensembles (as discussed in the previous paragraph) allows these strategies to be used to generate the 'best' ensemble with a similar frequency to the deterministic algorithms.

Two major conclusions can be drawn from this evaluation: firstly, the implementation is valid and performs appropriately, and secondly, no one selection strategy can be reliably used to generate the 'best' ensemble possible.

## **5.3 Classifier extension**

### **5.3.1 Aim**

The aim of this experiment is to evaluate the performance of the library-based ensemble classifier on multi-class data and on smaller data sets. Neither of these aspects were evaluated by the original investigation (Caruana et al. 2004).

### **5.3.2 Hypothesis**

As was referred to in section 5.2.2, the strength of the library-based ensemble classifier is derived from the diversity in inductive bias present in the model library. Each of the classifiers used to populate the model library have been shown to accurately classify multi-class data; therefore, it is hypothesised that the

library-based ensemble classifier will show similar levels of performance on multi-class data to that shown on binary-class datasets.

The performance of the classifier on smaller data sets is less clear. Smaller datasets are likely to result in a number of models with incomplete hypotheses due to insufficient training data; in this situation, ensembles often provide good performance for the reasons discussed in section 2.2. However, the small validation set is likely to not fully represent the true function of the data and encourage over-fitting. Therefore, the performance of the classifier on smaller data sets is likely to be highly dependent on the data set used: if the validation set does not lead to over fitting, the performance should be reasonable or performance is likely to be poor.

### 5.3.3 Method

The initial evaluation of the classifier (Caruana et al. 2004) did not investigate performance on multi-class data or small data sets; therefore, there is no reference available for comparison. Four datasets are used to validate the performance of the classifier; these too are available from the UCI machine learning repository (Newman et al. 1998). They are:

**LETTER:** is used in the first evaluation (section 5.2) and converted to a binary problem; in this evaluation, it is used in its unmodified state. This dataset will allow an evaluation of the performance of the library-based ensemble classifier in a multi-class setting. It has 26 different classes.

**LETTER.p3:** is a subset of the LETTER.p1 dataset used in the previous experiment. It has been reduced to 5% of its original size to evaluate the suitability of the library-based ensemble classifier on small data sets.

**PIMA:** is a binary dataset and describes whether the patient shows signs of diabetes. All patients are female, at least 21 years of age and of Pima Indian heritage. This dataset contains 768 instances and 8 attributes; it is significantly smaller than the other datasets previously evaluated, and should therefore give some additional insight on where it is appropriate to deploy the library-based ensemble classifier.

**ADULT4:** is a subset of the ADULT binary dataset used in the previous experiment. It has been reduced to 4% of its original size to evaluate the suitability of the library-based ensemble classifier on small datasets.

**WAVEFORM-5000:** is a multi-class dataset, with a number of attributes describing three classes of waves. Each wave class is a combination of two base waves. Two different versions of the dataset are used: one contains 21 attributes (WAVEFORM-21) and the other 40 attributes (WAVEFORM-40). In both cases, the dataset contains 5000 instances with noisy data present throughout. In WAVEFORM-21, all attributes are relevant; in WAVEFORM-40, the additional 19 attributes are irrelevant and their values effectively random. These datasets allow the evaluation of the ensemble's ability to deal with noise, and with irrelevant attributes.

### 5.3.4 Results

Tables A.9, A.10, A.11, A.12, and A.13 show the performance of the best classifiers (as identified by the validation set) when applied to the test set. The result of the best model of a particular type, under the given metric, is reported; therefore, each data point is potentially from a different model. The validation set allows the identification of models that are likely to provide the best performance, in terms of the given metric, on the test set. Tables A.14, A.15, A.16, A.17 and A.18 provide normalised performance scores for each dataset.

Table 5.7 compares the performance of the best library-based ensemble classifier on the test set against the performance of the best individual model (as determined by performance on the validation set) over the three smaller data sets. Table 5.8 gives the average performance of the library-based ensemble classifier and the best individual models (as determined by performance on the validation set) over the same three datasets. Table 5.9 compares the performance of the best library-based ensemble classifier on the test set

	ACC	FSC	ROC	AVP	BEP	RMS	CAL	SAR	MEAN
ADULT4	10.93	-4.70	-3.93	0.61	-0.30	5.58	23.63	3.81	4.46
LETTER.p3	-3.69	-2.20	-10.38	-1.55	-5.50	-3.78	0.00	-0.67	-3.47
PIMA	8.89	0.00	9.79	10.61	19.57	4.35	5.27	9.85	8.54
MEAN	5.38	-2.30	-1.51	3.22	4.59	2.05	9.63	4.33	3.18

Table 5.7: Percentage reduction in loss provided by the library-based ensemble classifier when compared with the best individual classifier (as determined by the validation set) for the small datasets.

	ACC	FSC	ROC	AVP	BEP	RMS	SAR	MEAN
LBEC-T	0.968	0.992	0.980	0.940	0.884	0.966	0.976	0.958
BAG	0.662	0.640	0.578	0.510	0.589	0.672	0.612	0.609
SVM	0.797	0.791	0.778	0.888	0.776	0.773	0.803	0.801
ANN	0.052	0.836	0.927	0.796	0.751	0.144	0.826	0.619
BST-DT	0.493	0.876	0.713	0.688	0.729	0.464	0.663	0.661
KNN	-0.475	0.734	0.753	0.674	0.551	-0.118	0.636	0.393
DT	0.707	0.759	0.818	0.775	0.788	0.557	0.711	0.731

Table 5.8: Normalised scores for the best models of each type and for ensemble selection, averaged over the three small datasets.

against the performance of the best individual model (as determined by performance on the validation set) over the three multi-class data sets. Table 5.10 gives the average performance of the library-based ensemble classifier and the best individual models (as determined by performance on the validation set) over the same three datasets. Note that the performance of the multi-class data sets are reported only in terms of accuracy: the remaining metrics cannot necessarily be applied in a multi-class setting.

### 5.3.5 Discussion

The evaluation of the library-based ensemble classifier on smaller datasets yielded mixed results. The performance of the data subsets (ADULT4 and LETTER.p3) were broadly comparable to the fully-sized datasets: ADULT4 saw an average increase in loss over the best model, while LETTER.p3 saw a reduction. The changes in loss were relatively smaller than the those observed on the full-sized data sets. The PIMA dataset also experiences an increase in loss when exposed to the ensemble classifier (see table 5.7).

A comparison of the absolute scores recorded by the best ensemble on the validation set shows significant drops in performance when the same ensemble is exposed to the test set. This significant drop is present across all eight metrics, and all three data sets. It is therefore reasonable to conclude that overfitting is occurring. The relatively small validation set size is unlikely to fully represent the underlying function that describes the data, forcing the ensemble to fit the function that is present in the validation set (which may or may not be the true function). This similarly affects the individual learning algorithms selected based on their performance of the validation set: they too over-fit, and perform poorly on the test set. However, some library-based ensembles do perform well on the test set. Therefore, the library-based ensemble classifier is much like all other classifiers on small datasets: performance on the validation set is not a good indicator of future performance.

	ACC
LETTER	2.97
WAVEFORM-21	-1.97
WAVEFORM-40	-1.62
MEAN	-0.20

Table 5.9: Percentage reduction in loss provided by the library-based ensemble classifier when compared with the best individual classifier (as determined by the validation set) for the multi-class datasets.

	ACC
LBEC-T	0.994
BAG	0.970
SVM	0.988
ANN	0.958
BST-DT	0.978
KNN	0.974
DT	0.967

Table 5.10: Normalised scores for the best models of each type and for ensemble selection, averaged over the multi-class datasets.

Similarly mixed results were seen in the multi-class data sets (see tables 5.9). The library-based ensemble classifier improves performance on both WAVEFORM datasets; the performance gains recorded by WAVEFORM-21 are, on average, greater than those recorded by WAVEFORM-40. This is unsurprising, as the library models have a similar performance imbalance; this can be attributed to the introduction of the 19 irrelevant attributes (as discussed in section 5.3.3). From this, it can be concluded that the library-based ensemble classifier does not possess the ability to cope with irrelevant attributes any better than its component algorithms. The WAVEFORM datasets are also noisy: it could therefore be theorised that the noise hindered the construction of a robust ensemble by over-fitting the ensemble to the function that underlies the validation set<sup>2</sup>, and lead to the smaller reductions in loss displayed by these data sets (when compared against those seen in section 5.2).

An initial review of the performance of the LETTER dataset reveals disappointing performance, with the library-based ensemble classifier increasing loss by 2.97% when compared to the best support vector machine (as determined by performance on the validation set); however, a closer inspection of the generated ensembles reveals the top 17 (as determined by performance on the validation set), from a total of 63 ensembles generated, either consist of only one model, or consist of several models of the same type. Therefore, it is completely understandable that the library-based ensemble classifier reduces performance, as the diversity requirement for an effective ensemble is not present. The top diverse classifier, as determined by validation set performance, records an accuracy of 91.8875% on the test set, outperforming all of the top individual algorithms (as determined by validation set performance). Therefore, the library-based ensemble classifier reduces loss by approximately 1%, which is in line with the gains seen in the WAVEFORM multi-class data sets.

From extending the library-based ensemble classifier beyond the conditions considered by Caruana et al. (2004), two main conclusions can be drawn. Firstly, building highly accurate ensembles for small datasets is hard as it is difficult to avoid over-fitting the ensemble to the validation set. The validation set in a small data set scenario may not be large enough to adequately reflect the true function of the underlying data; this leads to over-fitting. Secondly, the library-based ensemble classifier does appear to offer performance benefits to multi-class data sets as well; however, care must be taken to ensure the resulting ‘ensembles’ are suitably diverse for the expected benefits to be fully realised.

## 5.4 Application to spam

### 5.4.1 Aim

The aim of this experiment is to evaluate the performance and suitability of the library-based ensemble classifier in the spam filtering domain.

<sup>2</sup>This function would not be the true function that described the data, given the relatively small size of the set and the noise that is present.

## 5.4.2 Hypothesis

Given the success of machine learning algorithms in the spam classification domain and the success of the library-based ensemble classifier in a variety of domains, it is reasonable to expect that the library-based ensemble classifier will perform similarly in this domain. It is therefore hypothesised that the library-based ensemble classifier will equal or out-perform the top models present in the library and the naive Bayesian approach currently favoured by many commercial systems.

## 5.4.3 Method

Research into the application of machine learning algorithms to the spam classification task has confirmed the suitability of this approach for this task and identified a number of candidate algorithms that perform adequately. However, it is not possible to isolate the most promising approaches as each author uses a different collection of email (corpus) to evaluate their new solution. This lack of comparability also prohibits others from repeating and confirming published results. Therefore, it is difficult to leverage existing literature to construct the most robust system possible.

In an ideal setting, a standard benchmark corpus, composed of both legitimate and spam email, would be made available as a common basis for evaluation; however, this is far from being a straightforward task. There exist two key barriers to the development of any benchmark corpus:

- Legitimate email is difficult to make publicly available. Several online repositories of spam email exist, but no similar corpus of legitimate email is available, presumably due to privacy concerns. The recent release of internal Enron email as part of the Federal Energy Regulatory Commission investigation may prove to solve this particular issue; however, it consists of approximately 500,000 messages, of which an unknown quantity are spam emails.
- Spam email is constantly changing due to consumer trends and technological sophistication (Hunt & Cournane 2004). Any static spam email corpus would become less representative of current spam as time progressed. Graham-Cumming (2005) has reported eleven new HTML and encoding techniques designed to disguise a message in the last 12 months (this brings the total number of documented techniques to 42).

Given the difficulties associated with building a standard corpus, and the obvious need for one, several alternatives have been proposed. Legitimate email has been imitated through the use of mailing list postings. For example, in the LingSpam corpus (Androutsopoulos, Koutsias, Chandrinos, Paliouras & Spyropoulos 2000), postings from the moderated ‘Linguist’ mailing list are used in place of legitimate email. While the topic of such email may be slightly more specialised than that received by the normal user, it is still surprisingly diverse, with job postings, software availability announcements and flame-like responses. SpamAssassin, an open-source filtering project, maintains a collection of legitimate and spam emails generated in a similar manner to that of the LingSpam corpus. Both corpora are now over two years old, limiting their usefulness. Despite this, the standard, freely available nature of the corpora makes them an attractive option for preliminary evaluations of new spam filtering solutions.

Therefore, a two-pronged approach has been adopted to evaluate the spam filtering capability of the library-based ensemble classifier:

**Publicly available corpora:** Despite the age and legitimate email issues surrounding the use of publicly available corpora, they allow results to be compared and replicated, both of which are critical in scientific research. Two publicly available corpora have been evaluated:

**SpamAssassin corpus:** A test corpus is maintained by the SpamAssassin open-source project. It consists of 1897 spam and 4150 legitimate messages (from public mailing list posts). The corpus is converted into ARFF format via the process described in section 4.2. Each SpamAssassin message is complete with full email headers.

**SPAMBASE:** An internal Hewlett-Packard email dataset was released to the UCI machine learning repository in 1999 (Newman et al. 1998). This is not a corpus; it is already composed in an ARFF-like format. The 57 attributes are not identified through an information-gain-based

method such as that described in section 4.2. Instead, each attribute is manually identified, and the resulting set of attributes includes word and character frequencies and lengths of all-capital character strings. While the dataset is now over five years old, it is based on real user email and uses a markedly different approach to identifying attributes.

**Private corpus:** The privately constructed corpus represents a more realistic test of the classifier's ability to distinguish between legitimate and spam email. All of the author's email was collected over a period of one year; this formed the legitimate portion of the private corpus. A total of 5066 messages were retrieved, and had a diverse range of topics (personal, university and work-related correspondence, as well as a number of automated emails). The legitimate email was exported from a file in Microsoft Outlook PST format; regrettably, the export options did not allow the extraction of full email headers. Spam email for the corpus was sourced from user contributions to the SpamArchive (SpamArchive 2005) over the same time period; a total of 4965 messages were added to the private corpus. Given that the legitimate email did not include email headers, the headers were stripped from the SpamArchive emails.

The SpamAssassin and private corpus both use an information-gain-based approach to identify the most promising attributes; both will be evaluated using 50, 100, 200, 300, 400 and 500 attributes.

Cost-sensitive evaluation recognises the distorted balance between the misclassification cost of a false positive and the misclassification cost of a false negative. In other words, most users would consider a legitimate piece of email that is classified as spam to be worse than a spam message classified as legitimate. This form of evaluation has been used by a number of authors (for example, Sakkis et al. (2001a)) in this domain and typically considers the misclassification of a legitimate email message to be 1, 9, and 999 times more expensive than the misclassification of a spam message. While it is reasonable to say that a cost imbalance exists between the two types of misclassification error, there exists no research that suggests the imbalance factor is 1, 9 or 999; therefore, the conclusions drawn by such an analysis are limited. Furthermore, the misclassification cost imbalance is proportional to the accuracy of the filter itself; as a filter becomes more accurate, a user will become more complacent about checking filtering messages. Given the difficulty of estimating the misclassification factor, and the limited conclusions that it allows, cost sensitive evaluation has not been used here; however, it should be noted that WEKA and the library-based ensemble classifier does support cost-sensitive evaluation.

#### 5.4.4 Results

Tables A.19, A.20, A.21, A.22, A.23, and A.24 show the performance of the best classifiers (as identified by the validation set) when applied to the SpamAssassin test set. The result of the best model of a particular type, under the given metric, is reported; therefore, each data point is potentially from a different model. The validation set allows the identification of models that are likely to provide the best performance, in terms of the given metric, on the test set. Tables A.32, A.33, A.34, A.35, A.36, and A.37 provide normalised scores for the same datasets. Tables A.25, A.26, A.27, A.28, A.29, and A.30 show the performance of the best classifiers (as identified by the validation set) when applied to the private corpus test set. Tables A.38, A.39, A.40, A.41, A.42, and A.43 provide normalised scores for the same datasets. Tables A.31 and A.44 respectively show the absolute and normalised performance of the best classifiers (as identified by the validation set) when applied to the SPAMBASE dataset

Table 5.11 compares the performance of the best library-based ensemble classifier on the test set against the performance of the best individual model (as determined by performance on the validation set) over the spam data sets. Table 5.12 gives the average performance of the library-based ensemble classifier and the best individual models (as determined by performance on the validation set) over the same three datasets. Tables 5.13, 5.14 and 5.15 compare the accuracy of the best library-based ensemble classifier (as determined by the validation set) against the accuracy of a naive Bayes classifier.

Figure 5.3 shows the effect of different numbers of attributes on the performance of the library-based ensemble classifier. The accuracy of the library-based ensemble classifier at each attribute level is the best ensemble performance on the test set, where the best ensemble is identified by its performance on the validation set.

	ACC	FSC	ROC	AVP	BEP	RMS	CAL	SAR	MEAN
SA50	-5.50	-12.86	-8.72	-7.46	11.17	-3.79	-9.93	-11.66	-6.10
SA100	-7.79	-2.73	10.94	2.24	-1.92	-2.41	8.36	-1.90	0.60
SA200	-5.17	0.23	-7.08	-3.70	-14.16	-4.11	-20.14	-3.44	-7.20
SA300	16.93	6.23	5.90	-11.81	-4.38	10.72	-9.97	2.28	1.99
SA400	6.49	-1.84	9.68	14.60	13.84	0.00	19.54	6.76	8.63
SA500	-8.97	-0.99	32.87	-1.24	34.60	5.15	3.45	2.64	8.44
CC50	-5.09	-3.88	-3.39	-4.61	-5.24	-2.31	-0.67	-3.95	-3.64
CC100	-9.68	-11.42	-7.39	-11.84	-5.25	-3.06	-9.77	-2.78	-7.65
CC200	-3.36	-4.21	2.00	-29.21	3.63	-0.26	10.22	1.84	-2.42
CC300	-11.87	-3.16	-16.98	-26.19	-17.65	-5.99	-9.46	-7.74	-12.38
CC400	-8.67	-5.42	-10.34	-2.42	-20.26	-2.51	-7.00	-5.93	-7.82
CC500	-3.89	-20.39	-23.91	-16.68	-16.65	-4.16	-25.33	6.70	-13.04
SPAMBASE	-8.59	-15.03	-16.28	-10.23	2.14	-4.39	-4.42	-8.81	-8.20
MEAN	-4.24	-5.81	-2.52	-8.35	-1.55	-1.32	-4.24	-2.00	-3.75

Table 5.11: Percentage reduction in loss provided by the library-based ensemble classifier when compared with the best individual classifier (as determined by the validation set) for the spam datasets.

	ACC	FSC	ROC	AVP	BEP	RMS	SAR	MEAN
LBEC-T	0.990	0.993	0.990	0.983	0.982	0.978	0.987	0.986
BAG	0.955	0.959	0.954	0.919	0.922	0.919	0.942	0.938
SVM	0.965	0.970	0.971	0.946	0.960	0.938	0.953	0.958
ANN	0.954	0.964	0.965	0.938	0.962	0.919	0.949	0.950
BST-DT	0.965	0.966	0.968	0.939	0.952	0.938	0.959	0.955
KNN	0.891	0.893	0.910	0.868	0.885	0.823	0.878	0.878
DT	0.957	0.961	0.961	0.936	0.948	0.918	0.946	0.947

Table 5.12: Normalised scores for the best models of each type and for ensemble selection, averaged over the spam datasets.

	50	100	200	300	400	500
LBEC	92.91	95.60	96.59	97.00	97.29	97.69
NB	85.78	88.20	90.24	91.63	91.88	92.35
PRIL	-50.13	-62.68	-65.05	-64.19	-66.67	-69.74

Table 5.13: Accuracy recorded by the library-based ensemble classifier and a naive Bayes (NB) classifier over the SpamAssassin data set, with different numbers of attributes. Percentage reduction in loss (PRIL) is also reported.

	50	100	200	300	400	500
LBEC	87.15	91.83	94.62	94.99	95.76	96.44
NB	79.28	82.09	84.51	86.20	87.23	86.99
PRIL	-37.98	-54.37	-65.24	-63.69	-66.82	-72.63

Table 5.14: Accuracy recorded by the library-based ensemble classifier and a naive Bayes (NB) classifier over the private corpus data set, with different numbers of attributes. Percentage reduction in loss (PRIL) is also reported.

	ACC
LBEC	94.21
NB	80.68
PRIL	-70.04

Table 5.15: Accuracy recorded by the library-based ensemble classifier and a naive Bayes (NB) classifier over the SPAMBASE data set, with different numbers of attributes. Percentage reduction in loss (PRIL) is also reported.

	LBEC	NB	PMAS-C	PMAS-B	PMAS-H
SA	96.44	92.35	96.39	94.71	96.20

Table 5.16: The accuracy of the library-based ensemble classifier when compared against the WEKA naive Bayes algorithm, PMAS combined filter, PMAS Bayesian filter and PMAS heuristic filter.

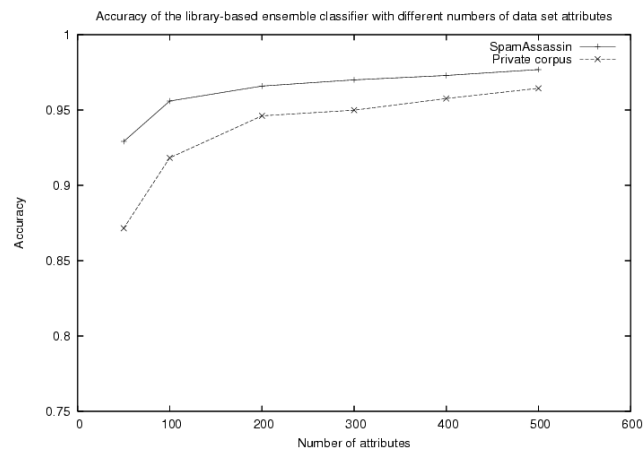


Figure 5.3: The effects of different numbers of attributes in the data set on the SpamAssassin and private corpora.



Carpinter & Hunt (2005) investigated the effectiveness of the PreciseMail Anti-Spam filtering system used by the University of Canterbury. The SpamAssassin corpus was passed through the University filter, and the classification results were recorded for the combined filter (using Bayesian and heuristic classification), the Bayesian filter and the heuristic filter. Table 5.16 compares the performance of the library-based ensemble classifier against the WEKA naive Bayes implementation, the combined PMAS filter, the Bayesian PMAS filter and the heuristic PMAS filter. It is critical to note that these results are not necessarily directly comparable, as the experimental method differs significantly. The PMAS filter was set to auto train, and developed its Bayesian model from the spam received by the University on a regular basis, and no part of the SpamAssassin corpus was used for training. However, these results provide a useful context for the results recorded by the library-based ensemble classifier.

### 5.4.5 Discussion

The performance of the library-based ensemble classifier when exposed to the SpamAssassin corpus is highly variable; reductions in loss are, on average, observed at low numbers of attributes (50–200), while increases in loss are recorded at higher numbers of attributes. It is difficult to explain this variable performance, as the range of models within the ensembles constructed are suitably diverse; however, the data set itself may provide some clues. Given that the legitimate email is retrieved from a mailing list, it is likely to be somewhat less diverse than an average users email; it is likely to centre upon one topic, and perhaps contains references to the mailing list itself. Therefore, it is reasonable to conclude that the data set is perhaps less diverse than a real user's collection of email; this lack of diversity reduces the difficulty of the learning task. Ensembles typically improve performance by providing a better estimation of the underlying hypothesis than any individual model; however, if the task is not difficult to learn, then individual models may be capable of finding a highly accurate hypothesis. In such a case, the hypothesis generated by the ensemble would be unlikely to be more accurate.

This explanation of the SpamAssassin performance is merely a theory; however, there exists some support. The performance of the library-based ensemble classifier on the SpamAssassin data set is consistently and significantly better than that seen on the private corpus dataset. Furthermore, it is reasonable to assume that additional attributes would make the problem easier and reduce the effectiveness of the ensemble classifier; this pattern is observed in the performance of the library-based ensemble classifier. Therefore, it is possible that the data set itself may not be an accurate reflection of user email. Despite this, the library-based ensemble classifier does noticeably improve performance at the lower attribute levels. This improvement is valuable; any increase in the number of attributes will result in substantial increases in the processing time required.

When applied to the private corpus, the library-based classifier consistently outperforms the best models of all types (as identified by validation set performance). The average reduction in loss is 7.86% and is in line with expectations, as Caruana et al. (2004) found an average reduction in loss of 8.76%. The average reduction in loss when applied to the SPAMBASE data set was 8.20%, which is also in line with expectations.

When put against the industry-standard machine learning approach for spam filtering, naive Bayesian learning, the library-based classifier reduced error (accuracy loss) by 62.25% on average. However, the naive Bayes approach implemented by WEKA is somewhat different to that used in industry. All WEKA classifiers are required to work from a finite number of attributes. In real-world naive Bayes filtering solutions, information regarding an infinite<sup>3</sup> number of tokens can be maintained and used to classify incoming email; a similar unlimited attribute system could not realistically be applied to many of the machine learning algorithms that comprise the model library. Therefore, it is difficult to determine whether the library-based ensemble classifier would outperform the naive Bayesian classifier present in many commercial implementations; however, the scale of the performance improvement over the WEKA classifier certainly merits further investigation, as it is reasonable to say that the library-based ensemble classifier typically outperforms a host of alternative algorithms.

Given the success recorded on the private corpus and the SPAMBASE data set, as well as the partial success on the SpamAssassin data set, some conclusions can be drawn about the differing attribute selection

---

<sup>3</sup>Limited only by computer resources.

approaches. Both the SpamAssassin and private corpus data sets were generated using an information gain approach, while the SPAMBASE data set used a manual attribute identification process. It has been shown here that identifying attributes through information gain can lead to effective classification, as can manual identification. The accuracy of the library-based ensemble classifier on the SPAMBASE data set is approximately equal to the accuracy recorded on the CC200 data set. Both data sets are based on similar corpora, which represent real user legitimate email. While the accuracies of each are not directly comparable given they represent different data sets, it is fair to say both approaches have merit, and benefit from the library-based ensemble classifier.

The comparison of the results observed from the library-based ensemble classifier and the PreciseMail Anti-Spam System give an indication of the current capabilities of commercial systems. It is important to reiterate that care should be taken when making comparisons; however, it appears that the library-based ensemble classifier classifies more accurately than the naive Bayesian filtering component within PMAS. The library-based ensemble classifier also outperforms the combined heuristic and naive Bayesian filter, but only slightly. It is likely that a combination of a heuristic rule filter and the library-based ensemble classifier would provide even better performance than either alone, as is observed when the heuristic and naive Bayesian components are combined.

While the private corpus does represent the most realistic collection of email available, the reported results are likely to be negatively affected by the exclusion of email headers (i.e. real-world performance is potentially higher). Graham (2003) concluded that email headers could substantially improve the classification accuracy of a naive Bayes filter; it is reasonable to assume a similar performance enhancement could be expected with the library-based ensemble classifier. Performance could only improve with the addition of more relevant information: if header tokens had a poor predictive ability, then the information gain ratio would not identify them as attributes and they would be ignored. The magnitude of any such performance improvement is beyond the scope of this work.

From this experiment, it is reasonable to conclude that the library-based ensemble classifier acts as a highly accurate spam filter for real user email and represents a more attractive solution, in terms of accuracy, when compared to other machine learning alternatives. Furthermore, better performance may be possible by incorporating a heuristic filter into the ensemble and by using full email headers when constructing the corpus.

## 5.5 Experimental observations

Overall, the results of the library-based ensemble classifier are impressive when compared against a range of benchmarks. It is effective; however, the question of its efficiency remains open and beyond the scope of this research. Despite this, it is important to approximately characterise performance in order to better judge the merits of this approach. For example, each model library generated for a full-sized dataset took between 24 and 48 hours using a collection of forty brand new machines (with AMD Athlon 3500 64-bit processors and 1GB of RAM). From each model library, typically 66 ensembles were generated and evaluated under a variety of metrics; this took at least 24 hours depending on the size of the ensemble and the complexity of the data<sup>4</sup>. To put this in perspective, a WEKA naive Bayes model can be generated in less than 0.2 seconds. It is therefore unlikely that library-based ensemble classifiers will be present in spam filtering solutions in the immediate or short-term future.

A visual inspection of the attributes located by the information gain ratio from the private corpus yield some interesting results. It identified 'James', 'Carpinter' and 'Christchurch', which are likely to be good indicators that a message is legitimate. It also identified a hyper link present in the 'Daily Dilbert' and CNET mailings that the author subscribes to. The string 'Buy', ',,,,,' and several illegal characters were identified as indicators of spam. The ability to identify user-specific characteristics is a key strength of a machine learning approach to spam and is why such filters are best located at the user, rather than the server, level (Garcia, Hoepman & van Nieuwenhuizen 2004).

---

<sup>4</sup>The ensemble construction process is typically reasonably quick; however, evaluating the ensemble on a large dataset can take a reasonable amount of time. Also, some metrics are particularly time-intensive in their calculation (e.g. average precision).

## Chapter 6

# Conclusions and further work

In this research, Caruana et al.'s (2004) ensemble classifier has been implemented within the WEKA machine learning environment and evaluated, extended and applied to the spam filtering domain. Despite the use of different classifiers and implementations, the underlying benefit derived from a diverse collection of inductive biases remains. It has been confirmed that this approach provides the necessary and sufficient accuracy and diversity of candidate hypotheses within an ensemble for good performance across a range of different metrics.

The initial experimental evaluation used the same datasets as used by Caruana et al. (2004) in an attempt to validate the implementation of the classifier. Overall, the ensemble classifier out-performed the component algorithms that comprised the model library; therefore, the implementation was considered to be a faithful implementation of the concept proposed by Caruana et al. (2004). An investigation into the performance of each selection strategy yielded few conclusions, with no individual strategy shown to consistently outperform any other, or to consistently avoid over-fitting to the validation set. Experiments with smaller datasets found that over-fitting to the validation was a significant issue which restricted performance. An extension to the classifier to allow it to classify multi-class data yielded similar benefits to those seen with binary datasets.

An evaluation of the classifier in the spam filtering problem domain yielded similarly positive results. A number of barriers prevent a realistic evaluation of a spam filtering algorithm; therefore, three differently constructed data sets are used in an attempt to draw robust conclusions. Overall, the classifier exceeded the performance of the models present in the library and also out-performed the industry-standard naive Bayesian filtering technique. Despite its classification performance, the computational requirements of the ensemble classifier are likely to prevent it from being applied to this problem in a real-world setting.

Some would argue that the substantial investment in computational resources required by the classifier outweigh any performance benefits. It could also be argued that there exists little point evaluating such a technology to implement in a domain that cannot support such an investment. Both arguments have merit, but fail to understand the potential application of the algorithm and ongoing research into spam filtering. In most applications of machine learning, the investment required for this classifier may not be worthwhile; however, there will exist critical problems where the expense is justified (for example, medical screening or the detection of financial fraud). Furthermore, the reusable nature of the model library minimises the ongoing cost associated with this classifier. In its present form, it is clearly not suited for spam filtering; however, it is critical to identify the upper limits achievable with machine learning algorithms in order to identify promising new techniques with lesser performance requirements and to determine whether machine learning can in fact offer a complete solution to this problem.

Caruana et al.'s (2004) ensemble classifier has been validated and extended to multi-class and spam data sets with positive results. The improvement provided by the ensemble classifier is not dramatic; however, consistently improving performance over the very best models is impressive. Furthermore, the ensemble can be easily optimised to whatever performance metric is required.

## 6.1 Further research and work

The current state of the WEKA plugin supporting the library-based ensemble classifier and distributed model generation is not yet ready for general use. The core components are present, and are responsible for the results presented in this study; however, certain aspects could be refined to increase usability. The architecture of the plugin provides an easily extensible platform for the evaluation of new selection strategies and metrics; given the over-fitting problem and other issues identified in this report, this is an area that could be actively researched to improve performance. Other further research with the library-based classifier could evaluate the effect of the training : validation : testing ratios, or investigate the effect of retraining the models selected to be part of the ensemble on the training and validation sets.

The biggest barrier that currently exists to the adoption of this high performance classifier is its significant computational resource demands. An interesting future research project may attempt to reduce performance demands by achieving diversity in inductive bias by using faster learning algorithms or implementations (for example, many of the decision trees used are particularly fast to train and evaluate when compared to a large artificial neural network) or a smaller library.

For machine learning to continue to improve its spam classification ability, and to find a resolution to this massive problem, it is critical to recognise that spam email filtering is not a text classification problem. Significant amounts of information are missed by machine learning techniques, as the context and meaning of much of the text is beyond the comprehension of most approaches. Considering the context of tokens is likely to result in improvements (e.g. the word ‘special’ when followed by ‘offers’ is a more likely indicator of spam than considering each token individually); however, this is merely one step that needs to be taken. Machine learning algorithms also need to be provided with more sophisticated data that can only be derived from interpreting the contents of the email (e.g. the number of recipients and the number of received headers may be valuable indicators of spam email). The information gain approach identified in this research is simple and reasonably effective; however, next generation spam filtering solutions will require additional information to better formulate an accurate hypothesis of the classification boundary. Further research could attempt to interpret email text, in context, to identify attributes that would otherwise be missed.

The degree to which the model library could be shared or reused would also be of interest. The computational cost of developing thousands of machine learning models may be acceptable if an email provider has tens of thousands of users. It is unlikely such a setup would provide classification accuracies like those reported here for the private corpus, as a filter operating at the server level would need to use a more generic set of attributes. Despite the limitations of server-level machine learning filters, naive Bayes filtering is widely used in such a position with positive results; therefore, a shared, generic model library at the server level could potentially form high performance ensembles.

Another issue facing machine learning in the spam filtering domain centres on concept drift. The structure and topics of legitimate and spam email change over time and this often renders machine learning models progressively less effective. Concept drift can be separated into two categories: a change in the underlying function of the data and a change in the attribute set. It would be interesting to characterise the type and extent of changes present in legitimate and spam email messages over time. The library-based ensemble filter is well suited to changes in the underlying function, as models that get progressively less accurate will simply not be selected and new models can be easily added. Changes to the attribute set are more difficult to manage and often require a regeneration of the model; clearly this is an expensive process, and it would be valuable to investigate machine learning approaches that could avoid this or minimise the cost of doing so.

## 6.2 Acknowledgements

I would like to thank my supervisor, Dr. Brent Martin, for his support, advice, proof-reading and answers (to all of my questions without fail). Thanks should also go to Assoc. Prof. Ray Hunt who originally sparked my interest in this area, and to Warwick Irwin who allowed me to present my implementation as part of my COSC427 project. Thanks also to my family, and in particular my Mum who proof-read all forty pages. The honours room crew should also be acknowledged for keeping me mostly sane over the

past year. And finally, thanks to the users of the undergraduate computer science labs, who unwittingly lent their machines for the computationally expensive process of library generation and ensemble evaluation.

# Bibliography

- Aha, D. W., Kibler, D. & Albert, M. K. (1991), 'Instance-based learning algorithms', *Mach. Learn.* **6**(1), 37–66.
- Androutsopoulos, I., Koutsias, J., Chandrinou, K., Paliouras, G. & Spyropoulos, C. (2000), An evaluation of naive bayesian anti-spam filtering, in 'Proc. of the workshop on Machine Learning in the New Information Age'.
- Androutsopoulos, I., Koutsias, J., Chandrinou, K. V. & Spyropoulos, C. D. (2000), An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages, in 'SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval', ACM Press, pp. 160–167.
- Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Sakkis, G., Spyropoulos, C. & Stamatopoulos, P. (2000), Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach, in 'Workshop on Machine Learning and Textual Information Access, 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)'.
- Breiman, L. (1996), 'Bagging predictors', *Machine Learning* **24**(2), 123–140.
- Carpinter, J. & Hunt, R. (2005), Tightening the net: a review of current and next generation spam filtering tools. University of Canterbury Department of Computer Science Summer Project (presently unpublished).
- Carreras, X. & Márquez, L. (2001), Boosting trees for anti-spam email filtering, in 'Proceedings of RANLP-01, 4th International Conference on Recent Advances in Natural Language Processing', Tzigras Chark, BG.
- Caruana, R., Joachims, T. & Backstrom, L. (2004a), 'Kdd-cup 2004: results and analysis', *SIGKDD Explor. Newsl.* **6**(2), 95–108.
- Caruana, R. & Niculescu-Mizil, A. (2004b), Data mining in metric space: an empirical analysis of supervised learning performance criteria, in 'Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining'.
- Caruana, R., Niculescu-Mizil, A., Crew, G. & Ksikes, A. (2004), Ensemble selection from libraries of models, in 'Proceedings of the Twenty-First International Conference on Machine Learning', pp. 137–144.
- Chhabra, S., Yezounis, W. & Siefkes, C. (2004), Spam filtering using a markov random field model with variable weighting schemas, in 'Data Mining, Fourth IEEE International Conference on', pp. 347–350.
- Clark, J., Koprinska, I. & Poon, J. (2003), A neural network based approach to automated e-mail classification, in 'Web Intelligence, 2003. WI 2003. Proceedings. IEEE/WIC International Conference on', pp. 702–705.
- Cover, T. M. & Hart, P. E. (1967), 'Nearest neighbor pattern classification', *IEEE Trans. Inform. Theory* **13**, 21–27.

- Cunningham, P., Nowlan, N., Delany, S. & Haahr, M. (2003), A case-based approach to spam filtering that can track concept drift, *in* 'ICCB'03 Workshop on Long-Lived CBR Systems'.
- Damiani, E., di Vimercati, S. D. C., Paraboschi, S. & Samarati, P. (2004), P2p-based collaborative spam detection and filtering, *in* 'P2P '04: Proceedings of the Fourth International Conference on Peer-to-Peer Computing (P2P'04)', IEEE Computer Society, pp. 176–183.
- Dietterich, T. G. (2000), 'Ensemble methods in machine learning', *Lecture Notes in Computer Science* **1857**, 1–15.
- Drucker, H., Wu, D. & Vapnik, V. (Sep. 1999), 'Support vector machines for spam categorization', *Neural Networks, IEEE Transactions on* **10**(5), 1048–1054.
- Frank, E. & Witten, I. H. (1998), 'Reduced-error pruning with significance tests'.
- Freund, Y. & Mason, L. (1999), The alternating decision tree learning algorithm, *in* 'Proceedings of the Sixteenth International Conference on Machine Learning', pp. 124–133.
- Freund, Y. & Schapire, R. E. (1996), Experiments with a new boosting algorithm, *in* 'International Conference on Machine Learning', pp. 148–156.
- Garcia, F., Hoepman, J.-H. & van Nieuwenhuizen, J. (2004), Spam filter analysis, *in* 'Proceedings of 19th IFIP International Information Security Conference, WCC2004-SEC', Kluwer Academic Publishers, Toulouse, France.
- Graham-Cumming, J. (2005), 'The spammers' compendium', [www.jgc.org/tsc/index.htm](http://www.jgc.org/tsc/index.htm).
- Graham, P. (2002), A plan for spam. <http://paulgraham.com/spam.html>.
- Graham, P. (2003), Better bayesian filtering, *in* 'Proc. of the 2003 Spam Conference'.
- Hansen, L. & Salamon, P. (1990), 'Neural network ensembles', *IEEE Trans. Pattern Analysis and Machine Intell.* **12**, 993–1001.
- Hearst, M., Dumais, S., Osman, E. & Scholkopf, J. P. B. (1998), 'Support vector machines', *Intelligent Systems, IEEE* **13**(4), 18–28.
- Hunt, R. & Cournane, A. (2004), 'An analysis of the tools used for the generation and prevention of spam', *Computers & Security* **23**(2), 154–166.
- Kohavi, R. (1996), Scaling up the accuracy of naive-bayes classifiers: a decision tree hybrid, *in* 'Proceedings of the Second International Conference on Knowledge Discovery and Data Mining'.
- Landwehr, N., Hall, M. & Frank, E. (2003), 'Logistic model trees'.
- Ludlow, M. (2002), 'Just 150 'spammers' blamed for e-mail woe', *The Sunday Times*.
- Mail Abuse Prevention Systems (2004), 'Definition of spam', [www.mail-abuse.com/spam\\_def.html](http://www.mail-abuse.com/spam_def.html).
- Mitchell, T. (1997), *Machine Learning*, McGraw–Hill.
- Newman, D., Hettich, S., Blake, C. & Merz, C. (1998), 'UCI repository of machine learning databases'.
- O'Brien, C. & Vogel, C. (2003), Spam filters: bayes vs. chi-squared; letters vs. words, *in* 'ISICT '03: Proceedings of the 1st international symposium on Information and communication technologies', Trinity College Dublin.
- Oda, T. & White, T. (2003a), Developing an immunity to spam, *in* 'Genetic and Evolutionary Computation Conference'.
- Oda, T. & White, T. (2003b), Increasing the accuracy of a spam-detecting artificial immune system, *in* 'Evolutionary Computation, CEC '03. The 2003 Congress on', Vol. 1, pp. 390–396.

- Pantel, P. & Lin, D. (1998), Spamcop—a spam classification & organisation program, in ‘Learning for Text Categorization: Papers from the 1998 Workshop’, AAAI Technical Report WS-98-05, Madison, Wisconsin.
- Platt, J. (1998), ‘Sequential minimal optimization: A fast algorithm for training support vector machines’.
- Postini Inc. (2005), ‘Email security annual review & threat report 2005’, <http://www.postini.com>.
- Process Software (2004), ‘Explanation of common spam filtering techniques (white paper)’, <http://www.process.com/>.
- Quinlan, J. R. (1986), ‘Induction of decision trees’, *Machine Learning* **1**(1), 81–106.
- Quinlan, J. R. (1992), Learning with Continuous Classes, in ‘5th Australian Joint Conference on Artificial Intelligence’, pp. 343–348.
- Quinlan, J. R. (1993), *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo.
- Rios, G. & Zha, H. (2004), Exploring support vector machines and random forests for spam detection, in ‘Conference on Email and Anti-Spam’.
- Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986), *Parallel Distributed Processing: Explorations of the Microstructure of Cognition*, Vol. 1, MA: The MIT Press, chapter Learning internal representations by error propagation, pp. 318–362.
- Sahami, M., Dumais, S., Heckerman, D. & Horvitz, E. (1998), A bayesian approach to filtering junk E-mail, in ‘Learning for Text Categorization: Papers from the 1998 Workshop’, AAAI Technical Report WS-98-05, Madison, Wisconsin.
- Sakkis, G., Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Spyropoulos, C. & Stamatopoulos, P. (2001a), A memory-based approach to anti-spam filtering, Technical report, Tech Report DEMO 2001.
- Sakkis, G., Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Spyropoulos, C. & Stamatopoulos, P. (2001b), Stacking classifiers for anti-spam filtering of e-mail, in ‘Empirical Methods in Natural Language Processing’, pp. 44–50.
- Siefkes, C., Assis, F., Chhabra, S. & Yerazunis, W. (2004), Combining winnow and orthogonal sparse bigrams for incremental spam filtering, in ‘Proceedings of ECML/PKDD 2004, LNCS’, Springer Verlag.
- Snyder, J. (2004), ‘Spam in the wild, the sequel’, <http://www.nwfusion.com/reviews/2004/122004spampkg.html>.
- SpamArchive (2005), <http://www.spamarchive.org/>.
- Vaughan-Nichols, S. (2003), ‘Saving private e-mail’, *Spectrum, IEEE* pp. 40–44.
- Wagner, M. (2002), ‘Study: E-mail viruses up, spam down’, Internetweek.com. <http://www.internetweek.com/story/INW20021109S0002>.
- Witten, I. & Frank, E. (2000), *Data Mining: Practical machine learning tools with Java implementations*, Morgan Kaufmann, San Francisco.
- Wolpert, D. H. (1990), Stacked generalization, Technical Report LA-UR-90-3460, Los Alamos, NM.
- Yerazunis, W. (2003), Sparse binary polynomial hashing and the crml14 discriminator, in ‘MIT Spam Conference’.
- Yoshida, K., Adachi, F., Washio, T., Motoda, H., Homma, T., Nakashima, A., Fujikawa, H. & Yamazaki, K. (2004), Density-based spam detector, in ‘KDD ’04: Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining’, ACM Press, pp. 486–493.
- Zeller Jr., T. (2005), ‘Law barring junk e-mail allows a flood instead’, The New York Times.



# Appendix A

## Detailed results

### A.1 Classifier validation experiments

	ACC	FSC	ROC	AVP	BEP	RMS	CAL	SAR
LBEC-T	0.99363	0.90108	0.96934	0.79502	0.82818	0.08062	0.00339	0.95216
LBEC-V	0.99500	0.93333	0.99610	0.86494	0.93333	0.07071	0.00101	0.96733
BAG	0.98581	0.78261	0.84195	0.65865	0.68955	0.11911	0.00339	0.90256
SVM	0.98394	0.79522	0.96049	0.65412	0.76534	0.12674	0.00340	0.93724
ANN	0.98781	0.84487	0.93512	0.73023	0.81193	0.11040	0.00198	0.93615
BST-DT	0.99037	0.86396	0.90417	0.77193	0.81185	0.09811	0.00198	0.93215
KNN	0.98862	0.85510	0.94151	0.74649	0.82236	0.10665	0.00198	0.94116
DT	0.98444	0.79198	0.86395	0.65213	0.72053	0.12475	0.00198	0.91628

Table A.1: Absolute results for the LETTER.p1 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	CAL	SAR
LBEC-T	0.94844	0.95265	0.95022	0.94697	0.95267	0.21257	0.00496	0.89912
LBEC-V	0.97750	0.97651	0.97628	0.97627	0.97230	0.15000	0.00262	0.93279
BAG	0.91450	0.91527	0.91448	0.89202	0.91245	0.29240	0.00900	0.84552
SVM	0.93581	0.93608	0.93582	0.92034	0.93463	0.25335	0.00676	0.87276
ANN	0.90906	0.90940	0.90907	0.88835	0.90774	0.30156	0.00957	0.83886
BST-DT	0.94969	0.94927	0.94287	0.92774	0.94289	0.22430	0.00539	0.88225
KNN	0.93837	0.93825	0.93842	0.92703	0.93189	0.24824	0.00648	0.87618
DT	0.88506	0.88635	0.88503	0.85537	0.88206	0.33902	0.01209	0.81035

Table A.2: Absolute results for the LETTER.p2 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	CAL	SAR
LBEC-T	0.85542	0.66164	0.81598	0.48804	0.67966	0.38230	0.01611	0.75220
LBEC-V	0.88334	0.92629	0.85632	0.94517	0.91405	0.33929	0.01197	0.78711
BAG	0.85738	0.65839	0.77626	0.59849	0.64044	0.37765	0.01267	0.75087
SVM	0.84805	0.65060	0.75963	0.46423	0.62079	0.38981	0.01267	0.72982
ANN	0.83523	0.61472	0.79090	0.39553	0.63733	0.40592	0.03497	0.71841
BST-DT	0.85189	0.64078	0.78113	0.42859	0.66529	0.38485	0.01900	0.74498
KNN	0.81599	0.48268	0.75566	0.52594	0.62858	0.42896	0.02293	0.71841
DT	0.85454	0.67854	0.76459	0.59444	0.67433	0.38155	0.01973	0.75114

Table A.3: Absolute results for the ADULT dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	CAL	SAR
LBEC-T	0.82491	0.82298	0.82473	0.79563	0.82103	0.41915	0.01847	0.74468
LBEC-V	0.85125	0.84822	0.84761	0.83469	0.83559	0.38569	0.01565	0.77227
BAG	0.81059	0.78273	0.81065	0.76532	0.80120	0.43522	0.01995	0.72867
SVM	0.79389	0.78004	0.79442	0.74175	0.77368	0.45399	0.02163	0.71144
ANN	0.77100	0.78418	0.77180	0.71089	0.74862	0.47854	0.02396	0.68808
BST-DT	0.82470	0.79496	0.82481	0.78037	0.81458	0.41869	0.01846	0.74361
KNN	0.78477	0.61754	0.78470	0.73898	0.77792	0.46393	0.02267	0.70184
DT	0.78559	0.78610	0.78586	0.73590	0.77098	0.46305	0.02256	0.70280

Table A.4: Absolute results for the COVTYPE dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	SAR	MEAN
LBEC-T	1.00032	0.98784	0.97651	0.94256	0.90543	1.00000	0.97830	0.97014
BAG	0.75056	0.85797	0.71146	0.77271	0.74665	0.66091	0.73086	0.74730
SVM	0.69083	0.87179	0.95810	0.76707	0.83346	0.59369	0.90387	0.80269
ANN	0.81444	0.92622	0.90531	0.86186	0.88682	0.73764	0.89843	0.86153
BST-DT	0.89620	0.94715	0.84092	0.91380	0.88673	0.84592	0.87847	0.88703
KNN	0.84031	0.93743	0.91861	0.88211	0.89877	0.77068	0.92342	0.88162
DT	0.70680	0.86824	0.75724	0.76459	0.78213	0.61122	0.79930	0.75565

Table A.5: Normalised results for the LETTER.p1 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	SAR	MEAN
LBEC-T	0.98143	0.97987	0.98473	0.99652	0.98946	0.98654	0.99478	0.98762
BAG	0.90665	0.85059	0.90656	0.87310	0.90096	0.82660	0.88046	0.87785
SVM	0.95360	0.92256	0.95324	0.93671	0.94976	0.90483	0.93856	0.93704
ANN	0.89466	0.83029	0.89473	0.86486	0.89060	0.80825	0.86626	0.86423
BST-DT	0.98418	0.96818	0.96866	0.95333	0.96794	0.96304	0.95880	0.96630
KNN	0.95924	0.93007	0.95892	0.95173	0.94374	0.91507	0.94585	0.94352
DT	0.84178	0.75057	0.84215	0.79078	0.83409	0.73320	0.80546	0.79972

Table A.6: Normalised results for the LETTER.p2 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	SAR	MEAN
LBEC-T	0.96525	0.97466	0.99528	0.70050	0.98351	0.94656	0.98415	0.93570
BAG	0.98371	0.96988	0.87017	1.01378	0.89561	0.98716	0.97611	0.95663
SVM	0.89586	0.95840	0.81778	0.63297	0.85158	0.88098	0.84878	0.84091
ANN	0.77514	0.90554	0.91628	0.43811	0.88864	0.74031	0.77976	0.77768
BST-DT	0.93202	0.94393	0.88550	0.53188	0.95130	0.92429	0.94048	0.87277
KNN	0.59397	0.71104	0.80528	0.80800	0.86904	0.53912	0.77976	0.72946
DT	0.95697	0.99956	0.83341	1.00230	0.97156	0.95311	0.97774	0.95638

Table A.7: Normalised results for the ADULT dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	SAR	MEAN
LBEC-T	0.98655	0.99526	0.98600	1.00000	0.99179	0.97595	0.98582	0.98877
BAG	0.94135	0.94658	0.94325	0.90153	0.93280	0.91976	0.93435	0.93138
SVM	0.88863	0.94333	0.89397	0.82496	0.85094	0.85414	0.87896	0.87642
ANN	0.81637	0.94834	0.82529	0.72470	0.77640	0.76830	0.80386	0.80904
BST-DT	0.98589	0.96137	0.98625	0.95042	0.97260	0.97755	0.98238	0.97378
KNN	0.85984	0.74681	0.86446	0.81596	0.86356	0.81938	0.84810	0.83116
DT	0.86243	0.95066	0.86798	0.80595	0.84291	0.82246	0.85119	0.85765

Table A.8: Normalised results for the COVTYPE dataset.

## A.2 Classifier extension experiments

	ACC	FSC	ROC	AVP	BEP	RMS	CAL	SAR
LBEC-T	0.76059	0.62844	0.71030	0.57393	0.54804	0.48930	0.02736	0.65487
LBEC-V	0.90323	0.86957	0.92500	0.89351	0.83333	0.31109	0.00877	0.83223
BAG	0.72801	0.62843	0.66957	0.54264	0.61479	0.52152	0.03134	0.62535
SVM	0.72801	0.47839	0.63143	0.54634	0.49094	0.52152	0.02649	0.59790
ANN	0.77036	0.61425	0.73613	0.61481	0.62201	0.47921	0.29890	0.68583
BST-DT	0.67752	0.57511	0.66558	0.49388	0.53175	0.56787	0.03628	0.59175
KNN	0.72313	0.50000	0.64735	0.53779	0.50590	0.52619	0.02649	0.61476
DT	0.78013	0.62434	0.71820	0.60657	0.60125	0.46890	0.02599	0.66867

Table A.9: Absolute results for the PIMA dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	CAL	SAR
LBEC-T	0.80518	0.58174	0.75021	0.47862	0.56835	0.44247	0.02925	0.69395
LBEC-V	0.96226	0.97561	0.97500	0.99329	0.97500	0.13736	0.00251	0.90568
BAG	0.82438	0.54920	0.74000	0.42767	0.52580	0.41907	0.02565	0.70519
SVM	0.79655	0.37158	0.70271	0.47286	0.53667	0.45106	0.03190	0.68408
ANN	0.76488	0.53210	0.72200	0.45830	0.50980	0.48490	0.02750	0.66531
BST-DT	0.78983	0.56112	0.69658	0.45291	0.55944	0.45845	0.03028	0.68067
KNN	0.80806	0.53917	0.69135	0.48180	0.50840	0.43811	0.02795	0.68710
DT	0.80710	0.52482	0.72339	0.47683	0.56705	0.43920	0.02366	0.68693

Table A.10: Absolute results for the ADULT4 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	CAL	SAR
LBEC-T	0.97969	0.70330	0.83766	0.54393	0.66721	0.13975	0.00197	0.88496
LBEC-V	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000	0.00000	1.00000
BAG	0.96250	0.00000	0.50000	0.04257	0.03750	0.19365	0.00197	0.75628
SVM	0.97891	0.69663	0.81886	0.53673	0.64783	0.14524	0.01356	0.88418
ANN	0.94453	0.44961	0.78098	0.26770	0.35802	0.23552	0.00197	0.83000
BST-DT	0.97344	0.54054	0.70590	0.38089	0.42690	0.16298	0.01332	0.83879
KNN	0.91016	0.35028	0.78314	0.19844	0.24031	0.29974	0.02459	0.79785
DT	0.96250	0.28346	0.69453	0.21460	0.37037	0.22009	0.00197	0.77501

Table A.11: Absolute results for the LETTER.p3 dataset.

	ACC
LBEC-T	0.91563
LBEC-V	0.93125
BAG	0.87913
SVM	0.91806
ANN	0.86763
BST-DT	0.91044
KNN	0.88719
DT	0.84819

Table A.12: Absolute results for the LETTER dataset.

	WAVEFORM-21	WAVEFORM-40
LBEC-T	0.86300	0.84775
LBEC-V	0.90500	0.89000
BAG	0.84100	0.84175
SVM	0.84900	0.84350
ANN	0.84425	0.81925
BST-DT	0.84800	0.82625
KNN	0.85525	0.83000
DT	0.86025	0.84525

Table A.13: Absolute results for the WAVEFORM-5000 datasets (in terms of accuracy).

	ACC	FSC	ROC	AVP	BEP	RMS	SAR	MEAN
LBEC-T	1.00000	0.97707	0.95984	0.93486	0.70441	1.00000	0.99689	0.93901
BAG	0.70143	0.97705	0.77394	0.80131	0.94008	0.68121	0.77809	0.80759
SVM	0.70143	0.74378	0.59986	0.81710	0.50281	0.68121	0.57464	0.66012
ANN	1.08953	0.95501	1.07773	1.10936	0.96558	1.09983	1.22636	1.07477
BST-DT	0.23873	0.89415	0.75573	0.59318	0.64689	0.22262	0.52905	0.55434
KNN	0.65671	0.77738	0.67252	0.78060	0.55563	0.63501	0.69960	0.68249
DT	1.17907	0.97069	0.99589	1.07418	0.89228	1.20184	1.09917	1.05902

Table A.14: Normalised results for the PIMA dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	SAR	MEAN
LBEC-T	0.90562	1.00000	1.00000	0.93080	0.94924	0.89849	0.95530	0.94850
BAG	1.28313	0.94406	0.95919	0.72725	0.82590	1.33352	1.05804	1.01873
SVM	0.73594	0.63874	0.81016	0.90779	0.85741	0.73880	0.86508	0.79342
ANN	0.11325	0.91467	0.88725	0.84962	0.77952	0.10969	0.69351	0.62107
BST-DT	0.60381	0.96455	0.78566	0.82809	0.92342	0.60141	0.83391	0.79155
KNN	0.96225	0.92682	0.76476	0.94351	0.77547	0.97955	0.89269	0.89215
DT	0.94337	0.90216	0.89281	0.92365	0.94548	0.95929	0.89113	0.92256

Table A.15: Normalised results for the ADULT4 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	SAR	MEAN
LBEC-T	0.99985	1.00000	0.97921	0.95419	0.99788	1.00000	0.97596	0.98673
BAG	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
SVM	0.95462	0.99052	0.92469	0.94049	0.96717	0.89814	0.97004	0.94938
ANN	-1.04538	0.63929	0.81484	0.42847	0.50792	-0.77681	0.55912	0.16106
BST-DT	0.63642	0.76858	0.59711	0.64389	0.61707	0.56902	0.62579	0.63684
KNN	-3.04479	0.49805	0.82110	0.29665	0.32138	-1.96827	0.31528	-0.39437
DT	0.00000	0.40304	0.56413	0.32741	0.52749	-0.49054	0.14206	0.21051

Table A.16: Normalised results for the LETTER.p3 dataset.

	ACC
LBEC-T	0.98403
BAG	0.94860
SVM	0.99265
ANN	0.93559
BST-DT	0.98402
KNN	0.95772
DT	0.91360

Table A.17: Normalised results for the LETTER dataset.

	WAVEFORM-21	WAVEFORM-40
LBEC-T	0.99884	0.99384
BAG	0.97337	0.98681
SVM	0.98263	0.98886
ANN	0.97713	0.96042
BST-DT	0.98148	0.96863
KNN	0.98987	0.97303
DT	0.99566	0.99091

Table A.18: Normalised results for the WAVEFORM-5000 datasets (in terms of accuracy).

### A.3 Application to spam experiments

	ACC	FSC	ROC	AVP	BEP	RMS	CAL	SAR
LBEC-T	0.92908	0.89460	0.91775	0.84973	0.85632	0.26356	0.00807	0.87070
LBEC-V	0.96266	0.94194	0.96257	0.90649	0.93333	0.19325	0.00490	0.91166
BAG	0.92227	0.87483	0.90704	0.83167	0.86738	0.27881	0.00931	0.85016
SVM	0.92495	0.87904	0.90989	0.83761	0.87076	0.27395	0.00896	0.85363
ANN	0.91834	0.87179	0.90937	0.83436	0.86391	0.28577	0.00916	0.84731
BST-DT	0.91813	0.86676	0.89937	0.82607	0.85181	0.28613	0.00960	0.84379
KNN	0.91420	0.86041	0.89490	0.81771	0.84594	0.29291	0.01010	0.83873
DT	0.91462	0.86201	0.89717	0.81673	0.86367	0.29220	0.01010	0.83986

Table A.19: Absolute results for the SA50 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	CAL	SAR
LBEC-T	0.95596	0.92813	0.94116	0.89734	0.92440	0.21327	0.00583	0.89692
LBEC-V	0.98340	0.98039	0.98795	0.96351	0.96104	0.11157	0.00184	0.95565
BAG	0.95224	0.91730	0.93816	0.89959	0.90668	0.21853	0.00538	0.88660
SVM	0.95183	0.92323	0.94415	0.89135	0.92292	0.21948	0.00590	0.89217
ANN	0.94997	0.92081	0.94440	0.88486	0.91423	0.22368	0.00627	0.89005
BST-DT	0.95018	0.92611	0.94696	0.87215	0.91965	0.22321	0.00697	0.89492
KNN	0.92826	0.89914	0.92653	0.84670	0.87023	0.26784	0.00841	0.87058
DT	0.94129	0.89851	0.92548	0.87080	0.90100	0.24231	0.00705	0.87003

Table A.20: Absolute results for the SA100 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	CAL	SAR
LBEC-T	0.96589	0.94275	0.96218	0.91858	0.94338	0.18187	0.00349	0.91427
LBEC-V	0.99585	0.99338	0.99699	0.99371	0.98675	0.06442	0.00124	0.97614
BAG	0.95369	0.92468	0.94839	0.89192	0.91839	0.21520	0.00577	0.89311
SVM	0.95514	0.92546	0.95617	0.91545	0.93404	0.21181	0.00437	0.89341
ANN	0.95121	0.92283	0.94548	0.90620	0.93034	0.22089	0.00507	0.89194
BST-DT	0.96403	0.94288	0.95930	0.90359	0.91538	0.18966	0.00518	0.91122
KNN	0.94811	0.91442	0.93552	0.88714	0.91031	0.22780	0.00614	0.88396
DT	0.95204	0.92343	0.94394	0.88836	0.91885	0.21901	0.00607	0.89232

Table A.21: Absolute results for the SA200 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	CAL	SAR
LBEC-T	0.97002	0.95635	0.96696	0.94337	0.95609	0.17727	0.00298	0.92603
LBEC-V	0.99585	0.98039	0.99096	1.00000	0.98684	0.06442	0.00000	0.96486
BAG	0.96858	0.94903	0.95885	0.91806	0.93380	0.17727	0.00437	0.91672
SVM	0.97436	0.95891	0.96880	0.93579	0.95408	0.16011	0.00331	0.92768
ANN	0.96382	0.94585	0.96155	0.91628	0.93914	0.19021	0.00451	0.91425
BST-DT	0.95245	0.92292	0.94030	0.89913	0.90908	0.21806	0.00542	0.89156
KNN	0.95762	0.93237	0.95051	0.90447	0.93151	0.20587	0.00516	0.90075
DT	0.95969	0.93437	0.94754	0.91714	0.91655	0.20078	0.00440	0.90215

Table A.22: Absolute results for the SA300 dataset.



	ACC	FSC	ROC	AVP	BEP	RMS	CAL	SAR
LBEC-T	0.97292	0.95947	0.96894	0.94270	0.95220	0.15946	0.00312	0.92194
LBEC-V	0.99170	0.99338	0.99096	0.98738	0.98684	0.09110	0.00124	0.96486
BAG	0.96175	0.93815	0.95173	0.91899	0.92915	0.19557	0.00432	0.90597
SVM	0.96858	0.94899	0.97168	0.93606	0.95801	0.17727	0.00376	0.91666
ANN	0.96960	0.95182	0.96487	0.92825	0.94655	0.17433	0.00305	0.92050
BST-DT	0.97457	0.95871	0.96555	0.95000	0.94229	0.15946	0.00261	0.92688
KNN	0.95865	0.93404	0.95180	0.90661	0.93346	0.20334	0.00504	0.90237
DT	0.96485	0.94276	0.95346	0.92904	0.92433	0.18747	0.00374	0.91028

Table A.23: Absolute results for the SA400 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	CAL	SAR
LBEC-T	0.97685	0.95912	0.96237	0.95062	0.94348	0.16768	0.00270	0.92495
LBEC-V	0.99585	0.99338	0.99699	0.99371	0.98684	0.06442	0.00102	0.97614
BAG	0.96175	0.93815	0.95173	0.91899	0.92915	0.19557	0.00432	0.90597
SVM	0.96858	0.94899	0.97168	0.93606	0.95801	0.17727	0.00376	0.91666
ANN	0.96961	0.95182	0.96487	0.92825	0.94655	0.17433	0.00305	0.92050
BST-DT	0.97457	0.95871	0.96555	0.95000	0.94229	0.15946	0.00261	0.92688
KNN	0.95865	0.93404	0.95180	0.90661	0.93346	0.20334	0.00504	0.90237
DT	0.96485	0.94276	0.95346	0.92904	0.92433	0.18747	0.00374	0.91028

Table A.24: Absolute results for the SA500 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	CAL	SAR
LBEC-T	0.87148	0.89188	0.85924	0.84671	0.87387	0.35949	0.01190	0.78880
LBEC-V	0.88496	0.90698	0.88411	0.88532	0.88466	0.33037	0.00987	0.81486
BAG	0.85210	0.88356	0.83050	0.82476	0.83274	0.38458	0.01206	0.76601
SVM	0.86028	0.88631	0.85181	0.83930	0.85374	0.37379	0.01252	0.77643
ANN	0.86086	0.88752	0.84421	0.81844	0.86690	0.37302	0.01198	0.77735
BST-DT	0.86086	0.88577	0.84888	0.83013	0.85268	0.37302	0.01211	0.77891
KNN	0.84219	0.85173	0.83498	0.83004	0.84724	0.39726	0.01517	0.75743
DT	0.86459	0.87971	0.85430	0.83861	0.85517	0.36798	0.01373	0.78011

Table A.25: Absolute results for the CC50 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	CAL	SAR
LBEC-T	0.91829	0.93188	0.91202	0.90322	0.90848	0.29156	0.00794	0.84417
LBEC-V	0.95575	0.96040	0.95041	0.95113	0.95522	0.21725	0.00465	0.89944
BAG	0.90767	0.91854	0.89858	0.87870	0.87912	0.30386	0.00887	0.83413
SVM	0.90810	0.92103	0.90297	0.88931	0.90341	0.30315	0.00956	0.83597
ANN	0.90264	0.91711	0.90122	0.88998	0.89369	0.31202	0.00994	0.82882
BST-DT	0.90810	0.92099	0.90500	0.89022	0.89936	0.30315	0.00960	0.83971
KNN	0.88067	0.90155	0.87877	0.86336	0.87444	0.34544	0.01167	0.80974
DT	0.90953	0.92310	0.90262	0.88720	0.89618	0.30077	0.00880	0.83713

Table A.26: Absolute results for the CC100 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	CAL	SAR
LBEC-T	0.94615	0.95454	0.93717	0.94558	0.92694	0.23543	0.00593	0.88005
LBEC-V	0.96755	0.97500	0.96916	0.97607	0.96538	0.16294	0.00253	0.91852
BAG	0.93208	0.94302	0.92337	0.90165	0.90434	0.26062	0.00594	0.86494
SVM	0.94428	0.95254	0.93840	0.92312	0.92950	0.23604	0.00538	0.88222
ANN	0.93481	0.94381	0.93216	0.92068	0.92925	0.25533	0.00672	0.87311
BST-DT	0.93251	0.94440	0.92767	0.91526	0.92297	0.25979	0.00661	0.86680
KNN	0.89977	0.91568	0.89088	0.87231	0.88037	0.31659	0.00992	0.82469
DT	0.94113	0.93882	0.93596	0.92213	0.92773	0.26929	0.00693	0.87815

Table A.27: Absolute results for the CC200 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	CAL	SAR
LBEC-T	0.94989	0.95274	0.94895	0.95006	0.95022	0.22418	0.00517	0.89027
LBEC-V	0.97935	0.98246	0.97469	0.97174	0.97512	0.14370	0.00207	0.93762
BAG	0.92476	0.937	0.91554	0.89347	0.89689	0.27431	0.00695	0.85533
SVM	0.94213	0.95013	0.93817	0.92743	0.93452	0.24056	0.00592	0.87991
ANN	0.9394	0.94713	0.93743	0.93234	0.93955	0.24617	0.00615	0.87689
BST-DT	0.93754	0.94647	0.93253	0.91934	0.92585	0.24993	0.00628	0.87338
KNN	0.90767	0.92224	0.89912	0.88037	0.88757	0.30386	0.01022	0.83431
DT	0.94314	0.9512	0.93851	0.92595	0.9319	0.23846	0.00571	0.88106

Table A.28: Absolute results for the CC300 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	CAL	SAR
LBEC-T	0.95764	0.96228	0.95474	0.95158	0.95879	0.20996	0.00425	0.90209
LBEC-V	0.98525	0.98753	0.98227	0.98141	0.97980	0.12145	0.00130	0.94869
BAG	0.94989	0.95766	0.94344	0.92656	0.92776	0.22386	0.00457	0.88982
SVM	0.95261	0.95913	0.94901	0.94028	0.94710	0.21768	0.00480	0.89465
ANN	0.94917	0.95621	0.94769	0.95038	0.94832	0.22546	0.00514	0.89155
BST-DT	0.94816	0.95511	0.94281	0.93275	0.93945	0.22768	0.00537	0.88594
KNN	0.91844	0.93087	0.91120	0.89421	0.90125	0.28559	0.00808	0.84802
DT	0.95362	0.96012	0.94952	0.93776	0.94232	0.21536	0.00461	0.89592

Table A.29: Absolute results for the CC400 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	CAL	SAR
LBEC-T	0.96439	0.96935	0.96665	0.96219	0.96275	0.18448	0.00339	0.90413
LBEC-V	0.97640	0.98020	0.97265	0.97431	0.97487	0.14370	0.00229	0.93283
BAG	0.94400	0.95267	0.93679	0.91147	0.91473	0.23665	0.00516	0.88138
SVM	0.95994	0.96150	0.95617	0.94503	0.94874	0.20016	0.00454	0.90532
ANN	0.94902	0.95728	0.94698	0.95462	0.95531	0.22578	0.00512	0.89007
BST-DT	0.96295	0.95338	0.95457	0.94637	0.95204	0.19248	0.00545	0.91015
KNN	0.91054	0.92511	0.90102	0.87989	0.88541	0.29910	0.00854	0.83749
DT	0.95233	0.95921	0.94735	0.94254	0.94866	0.21834	0.00462	0.89378

Table A.30: Absolute results for the CC500 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	CAL	SAR
LBEC-T	0.94212	0.92982	0.94206	0.90766	0.92017	0.24058	0.00648	0.88323
LBEC-V	0.97838	0.97436	0.97783	0.96619	0.96189	0.14704	0.00262	0.93639
BAG	0.93668	0.91741	0.93079	0.89714	0.90598	0.25163	0.00678	0.87195
SVM	0.90109	0.84911	0.87549	0.83688	0.83744	0.31450	0.01082	0.80695
ANN	0.91005	0.88200	0.90277	0.83575	0.88159	0.29991	0.00982	0.83668
BST-DT	0.92935	0.90960	0.92657	0.87670	0.92184	0.26580	0.00789	0.86337
KNN	0.86332	0.82119	0.85310	0.80126	0.81124	0.36971	0.01370	0.78223
DT	0.92636	0.90589	0.92247	0.87395	0.90500	0.27137	0.00815	0.86006

Table A.31: Absolute results for the SPAMBASE dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	SAR	MEAN
LBEC-T	0.97423	0.99521	0.98022	0.98432	0.94302	0.97642	1.00000	0.97906
BAG	0.94689	0.97321	0.95509	0.95109	0.96224	0.92619	0.93749	0.95032
SVM	0.95765	0.97790	0.96178	0.96202	0.96811	0.94220	0.94805	0.95967
ANN	0.93112	0.96983	0.96056	0.95604	0.95621	0.90327	0.92881	0.94369
BST-DT	0.93027	0.96423	0.93709	0.94079	0.93519	0.90208	0.91810	0.93254
KNN	0.91450	0.95717	0.92660	0.92541	0.92499	0.87975	0.90270	0.91873
DT	0.91618	0.95895	0.93193	0.92360	0.95579	0.88209	0.90614	0.92496

Table A.32: Normalised results for the SA50 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	SAR	MEAN
LBEC-T	1.00000	0.99890	0.97380	0.98861	0.99993	0.98882	0.99963	0.99281
BAG	0.98620	0.98725	0.96718	0.99243	0.97092	0.97382	0.97056	0.97834
SVM	0.98468	0.99363	0.98040	0.97845	0.99751	0.97111	0.98625	0.98458
ANN	0.97778	0.99102	0.98095	0.96743	0.98328	0.95914	0.98028	0.97713
BST-DT	0.97856	0.99673	0.98660	0.94587	0.99216	0.96048	0.99400	0.97920
KNN	0.89725	0.96770	0.94150	0.90268	0.91125	0.83321	0.92542	0.91129
DT	0.94558	0.96702	0.93919	0.94358	0.96163	0.90601	0.92387	0.94098

Table A.33: Normalised results for the SA100 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	SAR	MEAN
LBEC-T	0.98326	0.98872	0.99236	0.97381	0.98955	0.97441	0.98048	0.98323
BAG	0.94028	0.96976	0.96275	0.93081	0.95028	0.88853	0.92473	0.93816
SVM	0.94539	0.97058	0.97945	0.96876	0.97487	0.89726	0.92552	0.95169
ANN	0.93154	0.96782	0.95650	0.95384	0.96906	0.87387	0.92165	0.93918
BST-DT	0.97671	0.98885	0.98617	0.94963	0.94555	0.95434	0.97244	0.96767
KNN	0.92062	0.95900	0.93511	0.92310	0.93759	0.85606	0.90062	0.91887
DT	0.93446	0.96845	0.95319	0.92507	0.95101	0.87871	0.92265	0.93336

Table A.34: Normalised results for the SA200 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	SAR	MEAN
LBEC-T	0.98705	1.00000	0.99599	0.99194	0.99597	0.96326	0.99802	0.99032
BAG	0.98204	0.99235	0.97869	0.95200	0.96142	0.96326	0.97382	0.97194
SVM	1.00216	1.00268	0.99991	0.97998	0.99285	1.00644	1.00231	0.99805
ANN	0.96548	0.98902	0.98445	0.94919	0.96969	0.93069	0.96740	0.96513
BST-DT	0.92591	0.96504	0.93913	0.92213	0.92309	0.86060	0.90842	0.92062
KNN	0.94390	0.97493	0.96090	0.93055	0.95787	0.89128	0.93231	0.94168
DT	0.95111	0.97702	0.95457	0.95055	0.93467	0.90409	0.93595	0.94399

Table A.35: Normalised results for the SA300 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	SAR	MEAN
LBEC-T	0.98160	0.99500	0.99125	0.98305	0.98419	0.98049	0.96970	0.98361
BAG	0.94334	0.97289	0.95487	0.94593	0.94866	0.89210	0.92892	0.94096
SVM	0.96674	0.98413	0.99704	0.97265	0.99314	0.93690	0.95622	0.97240
ANN	0.97023	0.98707	0.98265	0.96042	0.97548	0.94409	0.96602	0.96942
BST-DT	0.98726	0.99421	0.98408	0.99447	0.96891	0.98049	0.98231	0.98453
KNN	0.93272	0.96863	0.95502	0.92655	0.95530	0.87308	0.91973	0.93300
DT	0.95396	0.97767	0.95853	0.96166	0.94123	0.91193	0.93993	0.94927

Table A.36: Normalised results for the SA400 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	SAR	MEAN
LBEC-T	0.99364	0.99344	0.97777	0.99249	0.97450	0.94095	0.97462	0.97820
BAG	0.94198	0.97172	0.95527	0.94312	0.95233	0.87406	0.92630	0.93783
SVM	0.96535	0.98295	0.99746	0.96977	0.99698	0.91795	0.95351	0.96914
ANN	0.96887	0.98588	0.98306	0.95758	0.97925	0.92500	0.96329	0.96613
BST-DT	0.98584	0.99302	0.98450	0.99152	0.97266	0.96067	0.97953	0.98111
KNN	0.93138	0.96747	0.95542	0.92380	0.95900	0.85543	0.91713	0.92995
DT	0.95259	0.97650	0.95893	0.95881	0.94487	0.89349	0.93727	0.94607

Table A.37: Normalised results for the SA500 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	SAR	MEAN
LBEC-T	1.00000	0.99891	0.99430	0.90729	0.98337	0.99865	0.99708	0.98280
BAG	0.93561	0.94865	0.91475	0.83505	0.85004	0.91396	0.92541	0.90335
SVM	0.96279	0.96527	0.97373	0.88290	0.91812	0.95038	0.95818	0.94448
ANN	0.96471	0.97257	0.95270	0.81425	0.96078	0.95298	0.96107	0.93987
BST-DT	0.96471	0.96200	0.96562	0.85272	0.91468	0.95298	0.96598	0.93981
KNN	0.90268	0.75637	0.92715	0.85243	0.89705	0.87116	0.89843	0.87218
DT	0.97711	0.92540	0.98063	0.88063	0.92275	0.96999	0.96975	0.94661

Table A.38: Normalised results for the CC50 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	SAR	MEAN
LBEC-T	1.00000	0.99820	1.00000	0.99826	1.00000	0.98655	0.98995	0.99614
BAG	0.96946	0.93336	0.96738	0.92457	0.91313	0.95319	0.96327	0.94634
SVM	0.97070	0.94546	0.97804	0.95646	0.98500	0.95512	0.96816	0.96556
ANN	0.95500	0.92641	0.97379	0.95847	0.95624	0.93107	0.94915	0.95002
BST-DT	0.97070	0.94527	0.98296	0.95919	0.97302	0.95512	0.97810	0.96634
KNN	0.89183	0.85078	0.91930	0.87847	0.89928	0.84044	0.89844	0.88265
DT	0.97481	0.95552	0.97719	0.95012	0.96361	0.96157	0.97124	0.96487

Table A.39: Normalised results for the CC100 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	SAR	MEAN
LBEC-T	0.98941	0.99126	0.98168	0.99572	0.93834	0.97463	0.97719	0.97832
BAG	0.95235	0.94118	0.95069	0.87893	0.87885	0.91617	0.94103	0.92274
SVM	0.98449	0.98257	0.98444	0.93601	0.94508	0.97322	0.98239	0.96974
ANN	0.95954	0.94462	0.97043	0.92952	0.94443	0.92845	0.96058	0.94822
BST-DT	0.95348	0.94718	0.96034	0.91512	0.92789	0.91809	0.94548	0.93823
KNN	0.86725	0.82233	0.87773	0.80094	0.81574	0.78626	0.84471	0.83071
DT	0.97619	0.92292	0.97896	0.93338	0.94042	0.89605	0.97265	0.94580

Table A.40: Normalised results for the CC200 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	SAR	MEAN
LBEC-T	0.99473	0.97215	0.98916	0.99108	0.98867	0.98441	0.98911	0.98704
BAG	0.92884	0.90451	0.91555	0.84311	0.84982	0.86996	0.90654	0.88833
SVM	0.97438	0.96094	0.96541	0.93191	0.94779	0.94701	0.96462	0.95601
ANN	0.96723	0.94804	0.96378	0.94475	0.96089	0.93420	0.95749	0.95377
BST-DT	0.96235	0.94521	0.95298	0.91076	0.92522	0.92562	0.94919	0.93876
KNN	0.88403	0.84108	0.87937	0.80886	0.82555	0.80249	0.85686	0.84261
DT	0.97703	0.96554	0.96616	0.92804	0.94097	0.95180	0.96734	0.95670

Table A.41: Normalised results for the CC300 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	SAR	MEAN
LBEC-T	0.99704	0.99009	0.99512	0.98275	0.99157	0.98464	0.99457	0.99082
BAG	0.97708	0.97069	0.97039	0.91814	0.91233	0.95391	0.96621	0.95268
SVM	0.98408	0.97686	0.98258	0.95357	0.96172	0.96757	0.97738	0.97197
ANN	0.97522	0.96460	0.97969	0.97965	0.96483	0.95037	0.97021	0.96923
BST-DT	0.97262	0.95998	0.96901	0.93412	0.94218	0.94546	0.95725	0.95438
KNN	0.89608	0.85818	0.89984	0.83459	0.84463	0.81744	0.86962	0.86005
DT	0.98668	0.98102	0.98370	0.94706	0.94951	0.97270	0.98031	0.97157

Table A.42: Normalised results for the CC400 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	SAR	MEAN
LBEC-T	0.99169	0.99009	1.00000	0.99431	0.99720	0.98671	0.97175	0.99025
BAG	0.94035	0.92208	0.93601	0.86538	0.87512	0.87739	0.92063	0.90528
SVM	0.98049	0.95809	0.97754	0.95069	0.96159	0.95386	0.97443	0.96524
ANN	0.95299	0.94088	0.95785	0.97506	0.97829	0.90017	0.94015	0.94934
BST-DT	0.98807	0.92498	0.97411	0.95409	0.96998	0.96995	0.98528	0.96664
KNN	0.85611	0.80971	0.85936	0.78510	0.80058	0.74653	0.82199	0.81134
DT	0.96133	0.94875	0.95864	0.94436	0.96138	0.91576	0.94849	0.94839

Table A.43: Normalised results for the CC500 dataset.

	ACC	FSC	ROC	AVP	BEP	RMS	SAR	MEAN
LBEC-T	0.98225	0.99654	0.99243	0.99161	0.98398	0.96776	0.98691	0.98593
BAG	0.96611	0.98324	0.96713	0.97145	0.95770	0.93984	0.95817	0.96338
SVM	0.86049	0.91004	0.84298	0.85598	0.83077	0.78098	0.79257	0.83912
ANN	0.88708	0.94529	0.90423	0.85382	0.91253	0.81785	0.86831	0.88416
BST-DT	0.94436	0.97487	0.95766	0.93228	0.98707	0.90404	0.93631	0.94808
KNN	0.74840	0.88011	0.79272	0.78773	0.78224	0.64148	0.72959	0.76604
DT	0.93548	0.97089	0.94845	0.92702	0.95589	0.88996	0.92788	0.93651

Table A.44: Normalised results for the SPAMBASE dataset.