Effective and Efficient Pruning of Meta-Classifiers in a Distributed Data Mining System *

Andreas L. Prodromidis †
Computer Science Department
Columbia University
New York, NY 10027
andreas@cs.columbia.edu

Salvatore J. Stolfo
Computer Science Department
Columbia University
New York, NY 10027
sal@cs.columbia.edu

Philip K. Chan
Computer Science Department
Florida Institute of Technology
Melbourne, FL 32901
pkc@cs.fit.edu

July 31, 1998

Abstract

Distributed data mining systems aim to discover and combine useful information that is distributed across multiple databases. One of the main challenges is the design of effective and efficient methods to combine multiple models computed over multiple distributed sources that scale well over many large distributed databases. We describe in detail several methods that evaluate, prune and combine large collections of imported models computed at remote sites into efficient and scalable meta-classifiers. We demonstrate and evaluate the pruning methods by detailing many experiments performed on actual credit card data sets supplied by collaborating financial institutions, where the target learning task is fraud detection. We show that pruned meta-classifiers can sustain or even improve predictive performance at a substantially higher throughput, compared to the unpruned meta-classifiers.

Keywords: distributed data mining, meta-learning, classifier evaluation, pruning, ensembles of classifiers, machine learning, credit card fraud detection.

Contact Author:

Andreas Prodromidis (Tel: 212-939-7078, Fax: 212-666-0140, e-mail: andreas@cs.columbia.edu)

*This research is supported by the Intrusion Detection Program (BAA9603) from DARPA (F30602-96-1-0311), NSF (IRI-96-32225 and CDA-96-25374) and NYSSTF (423115-445).

†Supported in part by an IBM fellowship

1 Introduction

Machine learning constitutes a significant part in the overall *Knowledge Discovery and Data Mining* (KDD) process, the process of extracting useful knowledge from large databases. One means of analyzing and mining useful information from large databases is to apply various machine learning algorithms to discover patterns exhibited in the data and compute descriptive representations (also called classifiers or models) that can be subsequently used for a variety of strategic and tactical purposes.

Machine learning or *inductive learning* (or *learning from examples* [23]) aims to identify regularities in a given set of training examples with little or no knowledge about the domain from which the examples are drawn. Given a set of training examples, i.e. $\{(\mathbf{x_1}, y_1), ..., (\mathbf{x_n}, y_n)\}$, for some unknown function $y = f(\mathbf{x})$, each interpreted as a set of attribute (feature) vectors $\mathbf{x}$ of the form $\{x_1, x_2, ..., x_m\}$ and a class label y associated with each vector, $y \in \{y^1, y^2, ..., y^k\}$, the task is to compute a *classifier* or *model* $\hat{f}$ that approximates f and correctly labels any feature vector drawn from the same source as the training set, for which the class label y is unknown. Some of the common representations used for the generated classifiers are decision trees, rules, version spaces, neural networks, distance functions, and probability distributions. In general, these representations are associated with different types of algorithms that extract different types of information from the database and provide alternative capabilities besides the common ability to classify unknown instances drawn from some domain.

However, one of the main challenges in knowledge discovery and data mining is the development of inductive learning techniques that **scale up** to large and (physically) distributed data sets. Many organizations seeking added value from their data are already dealing with overwhelming amounts of information that in time will likely grow in size faster than available improvements in machine resources. Many of the current generation of learning algorithms are computationally complex and require all the data to be resident in main memory which is clearly untenable for many realistic problems and databases. Furthermore, in certain cases, the data may be inherently distributed and cannot be localized on any one machine (even by a trusted third party) for a variety of practical reasons including physically dispersed mobile platforms like an armada of ships, security and fault tolerant distribution of data and services, competitive (business) reasons, as well as statutory constraints imposed by law. In such situations, it may not be possible, nor feasible, to inspect all of the data at one processing site to compute one primary "global" classifier or model at a

reasonable cost. We call the problem of learning useful new information from large and inherently distributed databases, the *scaling problem for machine learning*.

Meta-learning [8], a technique similar to *stacking* [40], was developed recently to deal with the scaling problem. The basic idea is to execute a number of machine learning processes on a number of data subsets in parallel, and then to combine their collective results (classifiers) through an additional phase of learning. Initially, each machine learning task, also called a *base learner*, computes a *base classifier*, i.e. a model of its underlying data subset or *training set*. Next, a separate machine learning task, called a *meta-learner*, integrates these independently computed base classifiers into a higher level classifier, called a *meta classifier*, by learning over a *meta-level training set*. This meta-level training set is composed from the predictions of the individual base classifiers when tested against a separate subset of the training data, also called a *validation set*. From these predictions, the meta-learner discovers the characteristics and performance of the base classifiers and computes a meta-classifier which is a model of the "global" data set. To classify an unlabeled instance, the base classifiers present their own predictions to the meta-classifier which then makes the final classification.

Meta-learning addresses the *scaling problem for machine learning*. It improves *efficiency* by executing *in parallel* the base-learning processes (implemented as a distinct serial program) on (possibly disjoint) subsets of the training data set (*a data reduction technique*). This approach has the advantage, of using serial code (without the time-consuming process of parallelizing it) and applying such code on small subsets of data that fit in main memory. Meta-learning, is *scalable* because meta-classifiers are classifiers themselves that can be further combined into higher level meta-classifiers in a bottom-up and distributed fashion. Furthermore, it improves *accuracy* by combining different learning systems each having different *inductive bias* (e.g representation, search heuristics, search space) [25]. By combining separately classifiers, meta-learning is expected to derive a higher level learned model that explains a large database more accurately than any of the individual learners.

The JAM system (Java Agents for Meta-learning) [38] is a distributed agent-based data mining system that implements meta-learning. JAM takes full advantage of the inherent parallelism and distributed nature of meta-learning by providing a set of learning agents that compute models (classifier agents) over data stored locally at a site, by supporting the remote dispatch and exchange of learning and classifier agents among the participating data base sites through a special distribution mechanism, and by providing a set of meta-learning agents that combine the computed models

that were learned (perhaps) at different sites.

There are several desiderata for a distributed data mining system. It must be: *extensible,* to allow use of past, present of future learning algorithm; *portable,* to be able to operate across different environments, e.g across the internet; *scalable,* to deal effectively with increasing number of data sites and database sizes; *efficient,* to utilize effectively the available system resources; *adaptive,* to be able to incorporate new information as it becomes available; *compatible,* to be able to integrate information from similar databases but of different schemas, and last but not least, *effective*, to generate accurate models of the underlying databases.

It will not be possible to cover all these issues in this paper. The extensibility and portability issues are described in detail in [38], and adaptivity is discussed in [29]. Compatibility is attained by several techniques that allow distributed data mining systems to cope with *incompatible* classifiers, i.e. classifiers that are trained from databases with similar but not identical schemas, presented in detail in [28]. Here, we concentrate on the scalability and efficiency of distributed data mining systems. The goal is dual:

1. to acquire and combine information from multiple databases in a timely manner and

2. to generate efficient meta-classifiers.

We address these two issues by employing meta-learning as a distributed learning method. Then we introduce distributed management protocols among the data sites that provide for the exchange of classifiers among remote sites. We then describe fast methods to evaluate these classifiers, filtering out (pruning) redundant models and deploying only the most essential classifiers to prevent the formation of complex, large and inefficient meta-classifiers. Through experiments performed on real credit card data provided by two different financial institutions, we evaluate the effectiveness of the proposed methods, we expose their limitations and dependencies to the particular data mining problem and we demonstrate that JAM, when provided with certain pruning methods, has the potential to improve efficiency without sacrificing predictive performance.

The remainder of this paper is organized as follows: Section 2 describes the problem in detail and outlines our approach. Section 3 and 4 describe two different methods on pruning meta-classifiers. In Section 5 we present the experiments performed and the performance results collected and Section 6 concludes the paper.

2 Performance of Distributed Data Mining Systems

Distributed data mining systems are compared with respect to their efficiency and scalability and with respect to the accuracy of their results. This section describes our approach from both perspectives.

2.1 Efficiency and Scalability

The main challenge in data mining and machine learning is to deal with large problems in a "reasonable" amount of time and at an acceptable cost. To our knowledge, JAM is the only data mining system to date that employs meta-learning as a means to mine distributed databases. However, the literature reports an extensive collection of methods that facilitate the use of inductive learning algorithms for mining very large databases [10]. Provost and Kolluri [32] categorized the available methods into three main groups; methods that rely on the design of fast algorithms, methods that reduce the problem size by partitioning the data and methods that employ a relational representation. Meta learning can be considered primarily as a method that reduces the size of the data, basically due to its data reduction technique and its parallel nature. On the other hand, it is also generic, meaning that it is algorithm and representation independent, hence it can benefit from fast algorithms and efficient relational representations.

Distributed systems have an additional level of complexity compared to serial, host-based systems. Distributed systems may need to deal with possibly heterogenous platforms, with several databases with (possibly) different schemas, with the design and implementation of scalable and effective protocols and with the selective and efficient use of the information gathered from the peer data sites. The scalability of a data mining system depends on the protocols that allow the collaboration and transfer of information among the data sites while efficiency relies on the appropriate evaluation and filtering of the total available set of models to minimize redundant use of system resources.

One of the most significant problems for the design of a distributed data mining system is to combine scalability and efficiency without sacrificing accuracy performance. To understand the issues and tackle the complexity of the problem, we examine scalability and efficiency at two levels, the system architecture level and the meta-learning level.

In the **first** case, we focus on the components of the system and its overall architecture. Assuming that the data mining system comprises several data sites, each with its own resources, databases,

machine learning agents and meta-learning capabilities, the problem is to design a protocol that would allow the data sites to collaborate efficiently without hindering their progress. Our goal is to allow the data sites to operate either independently or in collaboration without depending upon or being controlled by other "manager" sites. The protocol in a distributed data mining system should support dynamic reconfiguration of the system architecture (in case more data sites become available) and it should be scalable to allow data sites to participate in large numbers. Asynchronous protocols avoid the overheads of synchronization points introduced by synchronous protocols even though synchronous protocols are simpler and easier to design and implement. JAM has been designed around asynchronous protocols. Briefly, the JAM system can be viewed as a coarse-grain parallel application where the constituent sites function autonomously and (occasionally) exchange classifiers with each other. JAM is designed with asynchronous, distributed communication protocols that enable the participating database sites to operate independently and collaborate with other peer sites as necessary, thus eliminating centralized control and synchronization points.

Employing efficient distributed protocols addresses the scalability problem only partially. The analysis of the dependencies among the classifiers and the management of the agents within the data sites constitutes the other half of the scalability problem. If not controlled properly, the size and arrangement of the classifiers inside each data site may incur unnecessary and prohibitive overheads. Meta-classifiers are defined recursively as collections of classifiers structured in multi-level trees. Being hierarchically structured, meta-classifiers can promote scalability and efficiency in a simple and straight forward manner. It is possible, however, that certain conditions yield the opposite effects. As the number of data sites, the size of data subsets and the number of deployed learning algorithms increases, more base classifiers are made available to each site and meta-learning and meta-classifying are bound to strain system resources. To make things worse, meta-classifier trees can be re-built, and grown in breadth and depth as new information and new classifiers become available. Hence, in the **second** case, we delve inside the data sites to explore the types, characteristics and properties of the classifiers that participate in the distributed computation.

We investigate the effects of pruning and discarding certain base-classifiers on the performance of the meta-classifier. From actual experiments on base- and meta-classifiers trained to detect credit card fraud, we measured, a decrease of 50%, 90% and 95% in throughput for meta-classifiers composed of 13, 60 and 100 base-classifiers respectively.[1] As more base-classifiers are combined in a

[1]The experiments were conducted on a Personal Computer with a 200MHz Pentium processor running Solaris 2.5.1. Throughput here means the rate at which a stream of data items can be piped through and labeled by a meta-

meta-classifier, additional time and resources are required to classify new instances and that translates to increased overheads and lower throughputs. Memory constraints are equally important. For the same problem, a single ID3 decision tree ([33]) may require more than 750KBytes of main memory, while a C4.5 decision tree ([34]) may need 80KBytes. Retaining a large number of base classifiers and meta-classifiers may not be practical nor feasible.

The objective of pruning is to build partially grown meta-classifiers (meta-classifiers with pruned subtrees) to achieve comparable or better performance (accuracy) results than fully grown meta-classifiers. *Pre-training pruning* refers to the filtering of the classifiers before they are used in the training of a meta-classifier. Instead of combining classifiers in a brute force manner, with pre-training pruning we introduce a preliminary stage for analyzing the available classifiers and qualifying them for inclusion in a meta-classifier. Only those classifiers that appear (according to one or more pre-defined metrics) to be most "promising" participate in the final meta-classifier. Conversely, *post-training pruning*, denotes the evaluation and pruning of constituent base classifiers after a complete meta-classifier has been constructed.

2.2 Effectiveness

We seek effective measures to compare local and imported classifiers that are fast to compute and that choose the "best" classifiers to combine so that other classifiers can be discarded. Ideally, the resulting combined ensemble of classifiers should produce a smaller, less expensive meta-classifier with the best possible performance (from the perspective of predictive accuracy and/or cost). For example, many combining techniques (that have been proposed) aim to maximize the accuracy of the ensemble. We consider alternative "optimality criteria". We seek to minimize the computational expense of the ensemble meta-classifier while maximizing its predictive efficacy. Predictive efficacy does not mean only overall accuracy (or minimal error). Other measures of interest include True Positive (TP) and False Positive (FP) rates for binary classification problems, ROC analysis and problem specific cost models are all different criteria relevant to different problems and learning tasks. In the credit card fraud domain, for example, overall predictive accuracy is inappropriate as the single measure of predictive performance. If 1% of the transactions are fraudulent, then a model that always predicts "non-fraud" will be 99% accurate. Hence, TP rate is more important. Of the 1% fraudulent transactions, we wish to compute models that predict 100% of these, yet

classifier, as for example, in real-time fraud detection of a stream of transactions.

produce no false alarms (i.e. predict no legitimate transactions to be fraudulent). Hence, maximizing the $TP - FP$ spread may be the right measure of a successful model. Yet, one may find a model with TP rate of 90%, i.e. it correctly predicts 90% of the fraudulent transactions, but here it may correctly predict the lowest cost transactions, being entirely wrong about the top 10% most expensive frauds. Therefore, a cost model criteria may be the best judge of success, i.e a classifier whose TP rate is 10% may be the best cost performer.

Each of these different criteria demands alternative measures for selecting base classifiers when forming the final ensemble meta-classifier. We define several "heuristic" measures that are fast to compute in the hope of choosing the "best" classifiers to combine.

One may posit the view that the combining technique can be an arbitrarily expensive off-line computation. In the context we consider here, classifiers can be computed at remote sites at arbitrary times (even in a continuous fashion) over very large data sets that change rapidly. We seek to compute a meta-classifier as fast as possible on commodity hardware. Techniques that are quadratic in the training set size of cubic in the number of models are prohibitively expensive.

We therefore approach the problem considering fast search techniques to compute a meta-classifier. A greedy search method is described whereby we iteratively apply measures to the set of classifies and a "best" classifier is chosen for inclusion until a termination condition is met. The resulting set of chosen classifiers are then combined into a meta-classifier. We consider two strategies, pre-training pruning and post-training pruning.

3 Pre-training Pruning

One can employ several different measures and methods to analyze, compare and manage the formation of ensembles of classifiers. Here, we focus on the *diversity* and *specialty* metrics. *Accuracy*, *correlation error* and *coverage* are other metrics explored in the literature. For example, Ali and Pazzani [1] define correlation error as the fraction of instances for which a pair of base classifiers make the same incorrect predictions and Brodley and Lane [5] measured coverage by computing the fraction of instances for which at least one of the base classifiers produces the correct prediction.

3.1 Diversity

Brodley [6] defines diversity by measuring the *classification overlap* of a pair of classifiers, i.e. the percentage of the instances classified the same way by two classifiers while Chan [7] associates it with the entropy in the set of predictions of the base classifiers. (When the predictions of the classifiers are distributed evenly across the possible classes, the entropy is higher and the set of classifiers is more diverse.) Krogh and Vedelsby [17] measure the diversity, called *ambiguity*, of an ensemble of base classifiers, by calculating the mean square difference between the prediction made by the ensemble and the base classifiers. They proved that increasing ambiguity/diversity decreases overall error. Kwok and Carter [18] showed that ensembles with decision trees that were more syntactically diverse achieved lower error rates than ensembles consisting of less diverse decision trees, while Ali and Pazzani [1] suggested that the larger the number of gain ties,[2] the greater the ensemble's syntactic diversity is, which may lead to less correlated errors among the classifiers and hence lower error rates. However, they also cautioned that syntactical diversity may not be enough and members of the ensemble should also be competent (accurate).

Here, we measure the diversity $\mathcal{D}$ within a set of classifiers $\mathcal{H}$ by calculating the average diversity of all possible pairs of classifiers in that set $\mathcal{H}$:

$$\mathcal{D} = \frac{\sum_{i=1}^{|\mathcal{H}|-1} \sum_{j=i+1}^{|\mathcal{H}|} \sum_{k=1}^{n} Dif\ (C_i(y_k),\ C_j(y_k))}{\frac{(|\mathcal{H}|-1)\cdot|\mathcal{H}|}{2} \cdot n}$$

where $C_j(y_i)$ denotes the classification of the y_i instance by the C_j classifier and $Dif(a, b)$ returns zero if a and b are equal, and one if they are different. Intuitively, the more diverse the classifiers are, the more room a meta classifier will have to improve performance.

3.2 Class specialty

The term *class specialty* defines a family of evaluation metrics that concentrate on the "bias" of a classifier towards certain classes. A classifier specializing in one class, should exhibit, for that class, both a high True Positive (TP) and a low False Positive (FP) rate. The TP rate is a measure of how "often" the classifier predicts the class correctly, while FP is a measure of how often the classifier predicts the wrong class.

[2]The information gain of an attribute captures the "ability" of that attribute to classify an arbitrary instance. The information gain measure favors the attribute whose addition as the next split-node in a decision tree (or as the next clause to the body of a rule) would result in a tree (rule) that would separate as many examples as possible into the different classes.

For concreteness, given a classifier C_j and a data set containing n examples, we construct a two dimensional contingency table T^j where each cell T^j_{kl} contains the number of examples $\mathbf{x}$ for which the true label $L(\mathbf{x}) = c_k$ and $C_j(\mathbf{x}) = c_l$. Thus, cell T^j_{kk} contains the number of examples classifier C_j classifies correctly as c_k. If the classifier C_j is capable of 100% accuracy on the given data set, then all non-zero counts appear along the diagonal. The sum of all the cells in T^j is n. Then, the TP and FP rates are defined as:

$$TP(C_j, c_k) = \frac{T^j_{kk}}{\sum_{i=1}^{c} T^j_{ki}} \qquad FP(C_j, c_k) = \frac{\sum_{i \neq k} T^j_{ik}}{\sum_{i \neq k} \sum_{l=1}^{c} T^j_{il}}$$

In essence, $FP(C_j, c_k)$ calculates the number of c_k examples that classifier C_j misclassified versus the total number of examples that belong in different classes (c is the number of classes). With the *class specialty* metric we attempt to quantify the bias of a classifier towards a certain class. In particular, a classifier C_j is highly biased/specialized for class c_k when its $TP(C_j, c_k)$ is high and its $FP(C_j, c_k)$ is low. With the class specialty defined, we can evaluate the candidate (base-) classifiers and combine those with the highest "bias" per class, or in other words, those with the most specialized and accurate view of each class, in the hope that a meta-classifier trained on these "expert" (base-) classifiers will be able to uncover and learn their bias and take advantage of their properties.

3.2.1 One-sided class specialty metric

Given the definitions of the TP and FP rates, this is the simplest and most straight forward metric of the class specialty family. We can use the one-sided class specialty metric to evaluate a classifier by inspecting the TP rate or the FP rate of the classifier on a given class over the validation data set. A simple pruning algorithm would integrate in a meta-classifier the (base-) classifiers that exhibit high TP rates with the classifiers that exhibit low FP rates.

3.2.2 Two-sided class specialty metrics

The problem with the one-sided class specialty metric is that it may qualify poor classifiers. Assume, for example, the extreme case of a classifier that always predicts the class c_k. This classifier is highly biased and the algorithm will select it. So, we define two new metrics that are more balanced, the *positive combined specialty* $PCS(C_j, c_k)$ and the *negative combined specialty* $NCS(C_j, c_k)$ metrics, that take into account both the TP and FP rates of a classifier for a particular class. The former is biased towards TP rates, while the later is biased towards FP rates:

$$PCS(C_j, c_k) = \frac{TP(C_j, c_k) - FP(C_j, c_k)}{1 - TP(C_j, c_k)} \qquad NCS(C_j, c_k) = \frac{TP(C_j, c_k) - FP(C_j, c_k)}{FP(C_j, c_k)}$$

With these metrics defined, we integrate in a meta-classifier the (base-) classifiers that exhibit high PCS rates with the classifiers that exhibit low NCS rates.

3.2.3 Combined class specialty metric

A third alternative is to define a metric that combines the TP and FP rates of a classifier for a particular class into a single formula. Such a metric has the advantage of distinguishing the single best classifier for each class with respect to some predefined criteria. The *combined class specialty* metric, or $CCS(C_j, c_k)$, is defined as:

$$CCS(C_j, c_k) = f_{TP}(c_k) \cdot TP(C_j, c_k) + f_{FP}(c_k) \cdot FP(C_j, c_k)$$

where $-1 \leq f_{TP}(c_k), f_{FP}(c_k) \leq 1, \forall c_k$. Coefficient functions f_{TP} and f_{FP} are single variable functions quantifying the importance of each class according to the needs of the learning task and the distribution of each class in the entire data set.[3] Note that, the total *accuracy* of a classifier C_j can be computed by $\sum_{k=1}^{c} CCS(C_j, c_k)$, with each $f_{TP}(c_k)$ set to the distribution of the class c_k in the testing set and each $f_{FP}(c_k)$ set to zero. In many real world problems, e.g. medical diagnosis, credit card fraud, etc, the plain $TP(C_j, c_k)$ and $FP(C_j, c_k)$ rates fail to capture the entire story. The distribution of the classes in the data set may not be uniform and maximizing the TP rate of one class may be more important than maximizing total accuracy. In the credit card fraud detection problem, for instance, catching expensive fraudulent transactions is more vital than eliminating false alarms. The *combined class specialty metric* provides the means to associate a cost model with the performance of each classifier and evaluate the classifiers from an aggregate cost perspective.

3.3 Combining metrics

Instead of relying just on one criterion to choose the "best" (base-) classifiers the pruning algorithms can employ several metrics simultaneously. Different metrics capture different properties and qualify different classifiers as "best". By combining the various "best" classifiers into a meta-classifier we can presumably form meta-classifiers of higher accuracy and efficiency, without searching exhaustively the entire space of the possible meta-classifiers. For example, one possible

[3] A more general and elaborate specialty metric may take into account the individual instances as well:

$$CCS(C_j, c_k, (\mathbf{x_i}, y_i)) = f_{TP}(c_k, (\mathbf{x_i}, y_i)) \cdot TP(C_j, c_k, (\mathbf{x_i}, y_i)) + f_{FP}(c_k, (\mathbf{x_i}, y_i)) \cdot FP(C_j, c_k, (\mathbf{x_i}, y_i))$$

approach would be to combine the (base-) classifiers with high *coverage* and low *correlation error*. In another study [36] concerning credit card fraud detection we employ evaluation formulas for selecting classifiers that are based on characteristics such as *diversity*, *coverage* and *correlated error* or their combinations, e.g. *True Positive rate* and *diversity*. Next, we introduce two algorithms that we use in the rest of this paper.

3.4 Algorithms

Pruning refers to the evaluation and elimination of classifiers before they are used for the training of the meta-classifier. A pruning algorithm is provided with a set of pre-computed classifiers $\mathcal{H}$ (obtained from one or more databases by one or more machine learning algorithms) and a validation set $\mathcal{V}$ (a separate subset of data, disjoint from the training and test sets). The result is a set of classifiers $\mathcal{C} \subseteq \mathcal{H}$ to be combined in a meta-classifier. Forming accurate combinations, however, can be considered as "black art" [40] and determining the optimal meta-classifier is a combinatorial problem. Thus, we employ the accuracy, diversity, coverage and specialty metrics to guide a greedy search. More specifically, we implemented a diversity-based pruning algorithm and three instances of a combined coverage/specialty-based pruning algorithm, described in Table 1 in the Appendix.

3.4.1 Diversity-Based Pruning Algorithm

The diversity-based algorithm works iteratively selecting one classifier each time starting with the most accurate (base-) classifier. Initially it computes the diversity matrix d where each cell d_{ij} contains the ratio of the instances of the validation set for which classifiers C_i and C_j give different predictions. In each round, the algorithm adds to the list of selected classifiers $\mathcal{C}$ the classifier C_k that is most diverse to the classifiers chosen so far, i.e. the C_k that maximizes $\mathcal{D}$ over $\mathcal{C} \cup C_k, \forall k \in \{1, 2, \ldots |\mathcal{H}|\}$. The selection process ends when the N most diverse classifiers are selected. N is a parameter that depends on factors such as minimum system throughput, memory constraints or diversity thresholds.[4] The algorithm is independent of the number of attributes of the data set and its complexity is bounded by $O(n \cdot |\mathcal{H}|^2)$ (where n denotes the number of examples) due to the computation of the diversity matrix. For many practical problems, however, $|\mathcal{H}|$ is much smaller

[4]The diversity $\mathcal{D}$ of a set of classifiers $\mathcal{C}$ decreases as the size of the set increases. By introducing a threshold, one can avoid including redundant classifiers in the final outcome.

than n and the overheads are minor.

3.4.2 Coverage/Specialty-Based Pruning Algorithm

This algorithm combines the *coverage* metric and one of the instances of the *specialty metric*. Initially, the algorithm starts by choosing the (base-) classifier with the best performance with respect to the specialty metric for a particular target class on the validation set. Then, it iteratively selects classifiers based on their performance on the examples the previously chosen classifiers failed to cover. The cycle ends when there are no other examples to cover. The algorithm repeats the selection process for a different target class. The complexity of the algorithm is bounded by $O(n \cdot c \cdot |\mathcal{H}|^2)$. For each target class (c is the total number) it considers at most $|\mathcal{H}|$ classifiers. Each time it compares each classifier with all remaining classifiers (bounded by $|\mathcal{H}|$)) on all misclassified examples (bounded by n).

Even though the algorithm performs a greedy search, it combines classifiers that are diverse (they classify correctly different subsets of data), accurate (with the best performance on the data set used for evaluation with respect to the class specialty) and with high coverage. The three instances of the pruning algorithm that we studied combine coverage with $PCS(C_j, c_k)$, coverage with a specific $CCS(C_j, c_k)$ and coverage with a combined class specialty metric that associates a specific cost model tailored to the credit card fraud detection problem.

3.5 Related Work

There are several other methods reported in the Machine Learning and KDD literature that compute, evaluate and combine ensembles of classifiers [10]. Most of these algorithms, however, generate their classifiers by applying the learning algorithms on variants of the same data set or use different approaches to combine the derived classifiers [2, 11, 15, 16, 17, 26, 27, 35, 39]. Breiman [3] and LeBlanc and Tibshirani [19], for example, acknowledge the value of using multiple predictive models to increase accuracy, but they rely on cross-validation data and analytical methods, (e.g. least squares regression), to compute the best linear combination of the available hypothesis (models). All these methods can be considered as weighted voting among hypotheses. Instead, our methods employ meta-learning to combine the available models, i.e. we apply arbitrary learning algorithms to discover the correlations among the available models. Meta-learning has the advantage of employing an arbitrary learning algorithm for computing non-linear relations

among the classifiers (at the expense, perhaps, of generating less intuitive representations).

Merz and Pazzani's PCR^* [22] and Merz's $SCANN$ [21] algorithms are also algorithms that combine multiple models, the first for improving regression estimates, the latter for improving classification performance. The PCR^* algorithm maps the estimates of the models into a new representation using principal components analysis upon which a regression model is built. $SCANN$ [21] employs correspondence analysis [13] (similar to Principal Component Analysis) to map the predictions of the classifiers onto a new scaled space that clusters similar prediction behaviors and then uses the nearest neighbor algorithm to meta-learn the transformed predictions of the individual classifiers. PCR^* and $SCANN$ are sophisticated and effective combining algorithms, but they are appropriate for different contexts. PCR^* specifically avoids eliminating any of the candidate learned models, and $SCANN$ is too computationally expensive to combine classifiers and generate predictions within domains with many classifiers and large data sets. On the other hand, the pre-training pruning methods presented in this paper preceed the meta-learning phase and, as such, can be used in conjunction with $SCANN$ or any other combining algorithm if computational constraints pose no problem.

Provost and Fawcett [30, 31] introduced the ROC convex hull method for its intuitiveness and flexibility. The method evaluates and selects classification models by mapping them onto a *True Positive/False Positive* plane and by allowing comparisons under different metrics (TP/FP rates, accuracy, cost, etc.). The intent of these algorithms is to discover and prune away the sub-optimal models, while in this work, we focus on evaluation metrics that provide information about the interdependencies among the (base) classifiers and their potential when forming ensembles of classifiers [10, 14]. In fact, the performance of sub-optimal yet diverse models can be substantially improved when combined together and even surpass that of the best single model.

In other related work, Margineantu and Dietterich [20] studied the problem of pruning the ensemble of classifiers (i.e. the set of hypothesis (classifiers)) obtained by the boosting algorithm ADABOOST [12]. According to their findings, by examining the diversity and accuracy of the available classifiers, it is possible for a subset of classifiers to achieve similar levels of performance as the entire set. This research however, was restricted to derive all classifiers by applying the same learning algorithm on many different subsets of the same training sets. In this paper, we are considering the same problem, but with several additional dimensions. We consider the general setting where ensembles of classifiers can be obtained by applying, possibly, different learning algorithms over (possibly) distinct databases. Furthermore, instead of voting (ADABOOST) over the predic-

tions of classifiers for the final classification, we adopt meta-learning to combine predictions of the individual classifiers. The best pruning method described in [20] (reduced error pruning with back-fitting) is a computationally expensive algorithm that performs an extensive search for the subset of classifiers that gives the best voted performance on a separate pruning (validation) data set. In meta-learning, this algorithm would require the computation of a large (and possibly) prohibitive number of intermediate meta-classifiers before it could converge to its final set of classifiers. By relying on the heuristic metrics described in this paper, our pre-training pruning methods avoid computing such intermediate meta-classifiers.

4 Post-training pruning

Post-training pruning refers to the pruning of a meta-classifier after it is constructed. In contrast with pre-training pruning which uses greedy forward-search methods to choose classifiers, post-training pruning is considered a backwards selection method. It starts with all available classifiers (or with the classifiers pre-training pruning selected) and iteratively removes one at a time. Again, the objective is to perform a search on the (possibly reduced) space of meta-classifiers and further prune the meta-classifier without degrading predictive performance.

The deployment and effectiveness of post-training pruning methods depends highly upon the availability of unseen labeled (validation) data. Post-training pruning can be facilitated if there is an abundance of data, since a separate labeled subset can be used to estimate the effects of discarding specific base classifiers and thus guide the backwards elimination process. A hill climbing pruning algorithm, for example, would employ the separate validation data to evaluate and select the best (out of $|\mathcal{H}|$ possible) meta-classifier with $|\mathcal{H}| - 1$ base classifiers, then evaluate and select the best meta-classifier with $|\mathcal{H}| - 2$ (out of $|\mathcal{H}| - 1$ possible) and so on. In the event that additional data is not available, standard cross validation techniques can be used, instead, to estimate the performance of the pruned meta-classifier, at the expense of increased complexity.

Another disadvantage of such a method, is the overhead of constructing and evaluating all the intermediate meta-classifiers ($O(|\mathcal{H}|^2)$ in the example), which, depending on the learning algorithm and the data set size, can be prohibitive. Next, we describe an efficient post-training pruning algorithm that does not require intermediate meta-classifiers or separate validation data. Instead it extracts information from the ensemble meta-classifier and employs the meta-classifier training set.

4.1 Cost complexity post-training pruning

The algorithm is inspired by the *minimal cost complexity pruning* method of the CART decision tree learning algorithm [4] that associates a cost complexity parameter to the number of terminal nodes of decision trees. The method prunes decision trees by minimizing a metric that combines two factors, the cost complexity (size) of the tree and the misclassification cost estimate (error rate). The degree of pruning decision trees is controlled by adjusting the cost complexity parameter (which can be regarded as the weight of the cost complexity factor), i.e. an increase of the complexity cost parameter results in heavier pruning. The *minimal cost complexity pruning* method guarantees the "best" pruned decision tree (according to the misclassification cost) of a specific size (as dictated by the cost complexity parameter) will be found.

The post-training pruning algorithm operates in three phases. First, it computes a decision tree model of the meta-classifier by applying a decision tree algorithm to the data set composed by the the meta-classifier's training set and the meta-classifier's predictions on the same set, i.e. the target class is the predictions of the meta-classifier on its own training set. This decision tree reveals the base classifiers that do not participate in the splitting criteria and hence are irrelevant to the meta-classifier. Those base classifiers that are deemed irrelevant are pruned in order to meet the performance objective. The next phase aims to further reduce the number of selected base classifiers, if necessary, according to the restrictions imposed by the available system resources and/or the runtime constraints. The post-training pruning algorithm applies the minimal cost complexity pruning method to reduce the size of the decision tree and thus prune away additional base classifiers. The degree of pruning can be controlled by the cost complexity parameter and since minimal cost complexity pruning eliminates first the branches of the tree with the least contribution to the decision tree's performance, the base classifiers pruned during this phase will also be the least "important" base classifiers. Finally, the post-training pruning algorithm re-computes a new final ensemble meta-classifier over the remaining base-classifiers (those that were not discarded during the first two phases).

An advantage of computing a decision tree model of a meta-classifier is its increased comprehensibility comparing to that of the meta-classifier. The decision tree models, combined with certain *correlation* measurements (described next) can be used as a tool for studying and understanding the various pruning methods and for interpreting meta-classifiers.

4.2 Correlation metric and pruning

Contrary to the diversity metric of Section 3.1, the correlation metric measures the degree of "similarity" between a pair of classifiers. Given $|\mathcal{H}| + 1$ classifiers C_1, C_2, ... $C_{|\mathcal{H}|}$ and C' and a data set of n examples mapped onto c classes, we can construct a two dimensional $c \times |\mathcal{H}|$ contingency matrix where each cell $M_{ij}^{C'}$ contains the number of examples classified as i by both classifiers C' and C_j. This means that if C' and C_j agree only when predicting class i, the $M_{ij}^{C'}$ cell would be the only non-zero cell of the j^{th} column of the matrix. If two classifiers C_i and C_j generate identical predictions on the data set, the i^{th} and j^{th} columns would be identical. And for the same reason, the cells of the j^{th} column would sum up to n only if C' and C_j produce identical predictions.

We call matrix $M_{ij}^{C'}$ the *correlation matrix* of C' since it captures the correlation information of the C' classifier with all the other classifiers $C_1 ... C_{|\mathcal{H}|}$. In particular, we define:

$$Corr(C', C_j) = \frac{\sum_{i=1}^{c} M_{ij}^{C'}}{n} \tag{1}$$

as the *correlation* between the two classifiers C' and C_j. *Correlation* measures the ratio of the instances in which the two classifiers agree, i.e. yield the same predictions.

In the meta-learning context, the correlation matrix M_{ij}^{MC} of the meta-classifier MC and the base-classifiers can provide valuable information on the internals of the meta-classifier. If, n_i denotes the total number of examples predicted as i by the meta-classifier, then the ratio $\frac{M_{ij}^{MC}}{n_i}$ can be considered as a measure of the degree the meta-classifier agrees with the base classifier C_j about the i^{th} class. Furthermore, by comparing these ratios to the specialties of the base classifiers (calculated during the pre-training pruning phase), a human expert may discover that the meta-classifier has underrated a certain base classifier in favor of another and may decide to remove the latter and re-combine the rest. The correlations $Corr(MC, C_j)$ between the meta-classifier MC and the base classifier C_j, $1 \leq j \leq |\mathcal{H}|$, can also be used as an indication of the degree the meta-classifier relies on that base classifier for its predictions. For example, it may reveal that a meta-classifier depends almost exclusively on one base classifier, hence the meta-classifier can be at least replaced by that single base classifier, or it may reveal that one or more of the base classifiers are trusted very little and hence the meta-classifier is at least inefficient. In fact, the correlation metric can be used as an additional metric for post-training pruning and for searching the meta-classifier space backwards.

A simple correlation-based post-training pruning algorithm that we employed in of our exper-

iments, removes the base classifiers that are least correlated to the meta-classifier. It begins by identifying and removing the base classifier that is least correlated to the initial meta-classifier, and continues iteratively, by computing a new reduced meta-classifier (with the remaining base classifiers), then it searches for the next least correlated base classifier and removes it. The method stops when enough base-classifiers are pruned. The advantage of this method is that it employs a different metric to search the space of possible meta-classifiers, so it evaluates different combinations of base classifiers not considered during the pre-training pruning phase. On the other hand, the least correlated base classifiers are not always the least important base classifiers and pruning them away may force the search process from considering "good" combinations.

5 Empirical Evaluation

In this section we present several experiments performed to evaluate the classification models and the algorithms that search for effective and efficient meta-classifiers.

5.1 Experimental setting

Learning algorithms Five inductive learning algorithms are used in our experiments, Bayes, C4.5, ID3, CART and Ripper. ID3, its successor C4.5 [34], and CART are decision tree based algorithms, Bayes, described in [24], is a naive Bayesian classifier, and Ripper [9] is a rule induction algorithm.

Learning tasks Two data sets of real credit card transactions were used in our experiments. The credit card data sets were provided by the Chase and First Union Banks, members of FSTC (Financial Services Technology Consortium).

The two data sets contained credit card transactions labeled as fraudulent or legitimate. Each bank supplied .5 million records spanning one year. Chase bank data consisted, on average, of 42,000 sampled credit card transactions records per month with a distribution of 20% fraud and 80% non-fraud distribution, whereas First Union data were sampled in a non-uniform manner (many records from some months, very few from others) with a 15% versus 85% distribution. The schemas of the databases was developed over years of experience and continuous analysis by bank personnel to capture important information for fraud detection. We cannot reveal the details of the schema beyond what is described in [37]. The records have a fixed length of 137 bytes each and about 30 numeric attributes including the binary class label (fraud/legitimate transaction). Some of

the fields are numeric and the rest categorical, i.e. numbers were used to represent a few discrete categories.

To evaluate and compare the meta-classifiers constructed, we adopted three metrics: the overall accuracy, the $(TP - FP)$ spread and a cost model fit to the credit card fraud detection problem. Overall accuracy expresses the ability of a classifier to provide correct predictions, $(TP - FP)$[5] denotes the ability of a classifier to catch fraudulent transactions while minimizing false alarms, and finally, the cost model captures the performance of a classifier with respect to the goal of the target application (stop loss due to fraud).

Credit card companies have a fixed overhead that serves as a threshold value for challenging the legitimacy of a credit card transaction. If the transaction amount *amt*, is below this threshold, they choose to authorize the transaction automatically. Each transaction predicted as fraudulent requires an "overhead" referral fee for authorization personnel to decide the final disposition. This "overhead" cost is typically a "fixed fee" that we call $\$Y$. Therefore, even if we could accurately predict and identify all fraudulent transactions, those whose *amt* is less than $\$Y$ would produce $\$(Y - amt)$ in losses anyway. With this overhead cost taken into account, we seek to produce fraud detectors that generate the maximum savings $S(C_i, Y)$:

$$S(C_j, Y) = \sum_{i=1}^{n} [F(C_j, \mathbf{x_i}) \cdot (amt(\mathbf{x_i}) - Y) - L(C_j, \mathbf{x_i}) \cdot Y] \cdot I(\mathbf{x_i}, Y)$$

where $F(C_j, \mathbf{x_i})$ returns one when classifier C_j classifies correctly a fraudulent transaction $\mathbf{x_i}$ and $L(C_j, \mathbf{x_i})$ returns one when classifier C_j misclassifies a legitimate transaction $\mathbf{x_i}$. $I(\mathbf{x_i}, Y)$ inspects the transaction amount *amt* of transaction x_i, and returns one if it greater than \$Y and zero otherwise, while n denotes that number of examples in the data set used in the evaluation.

First we distributed the data sets across six different data sites (each site storing two months of data) and we prepared the set of candidate base classifiers, i.e. the original set of base classifiers the pruning algorithm is called to evaluate. We computed these classifiers by applying the 5 learning algorithms on each month of data, therefore creating 60 base classifiers (10 classifiers per data site). Next, we had each data site import the "remote" base classifiers (50 in total) that were subsequently used in the pruning and meta-learning phases, thus ensuring that each classifier would not be tested unfairly on known data. Specifically, we had each site use half of its local data (one month) to test,

[5]The $(TP - FP)$ spread is an ad-hoc, yet informative and simple metric characterizing the performance of the classifiers. In comparing the classifiers, one can replace the $(TP - FP)$ spread, which defines a certain family of curves in the ROC plot, with a different metric or even with a complete analysis [30] in the ROC space.

prune and meta-learn the base-classifiers and the other half to evaluate the overall performance of the pruned meta-classifier. The above "month-dependent" data partitioning scheme was used only on the Chase bank data set. The very skewed nature of the First Union data forced us to equi-partition the entire data set randomly into 12 subsets and assign 2 subsets in each data site.

5.2 Performance of base classifiers

In Figures 1 and 2, we present the accuracy, the $(TP-FP)$ spread and the savings of the individual base classifiers on the Chase and First Union credit card data, respectively. The x-axis corresponds to the month of data used to train the base classifiers, starting in October 1995 and ending in September 1996 (the one year span is repeated for the five learning algorithms). Each vertical bar represents a specific base classifier, e.g. the first bar of the left plot of Figure 1 represents the accuracy (83%) of the Bayesian base classifier that was trained on the December 1995 data. The maximum achievable savings a perfect classifier, with respect to the cost model, is \$1,470K for the Chase and \$1,085K for the First Union data sets. The test set consists of October 1995 - November 1995 data, hence, the performance results of the classifiers corresponding to these months are not included in these plots as these classifiers had been trained on this data. According to the figures, some learning algorithms can be more suitable for one problem (e.g naive Bayes on Chase data) than for another (e.g. naive Bayes on First Union data), even though the two sets are very similar in nature. Overall, it appears that all learning algorithms performed better on the First Union data set than on the Chase data set. On the other hand, note that there are fewer fraudulent transactions in the First Union data and this causes a higher baseline accuracy.

5.3 Pre-Training Pruning Experiments

With 50 candidate base classifiers per data site, there are $(2^{50} - 50)$ distinct combinations of base classifiers from which the pruning algorithm has to choose one. In the meta-learning stage, we employed all five learning algorithms to combine the selected base classifiers. The next section, reports the performance results of pruning and meta-learning averaged over all six data sites. In essence, this experiment is a 6-fold cross validation where each fold is executed in parallel.

Results The results from these experiment are displayed in Figures 3, and 4. Figure 3 plots the overall accuracy, the $(TP-FP)$ spread and the savings (in dollars) for the Chase bank credit card

data, and Figure 4 for the First Union data. Each figure contrasts two specialty/coverage based, one diversity-based, a metric-specific pruning method and an additional classifier selection method denoted here as *arbitrary*. As the name indicates, *arbitrary* uses no particular strategy to evaluate base classifiers; instead it combines them in a "random" order, i.e. when they become available. The metric-specific pruning methods (which correspond to three different instances of the CCS specialty metric of Section 3.2, namely the accuracy, the $(TP-FP)$,[6] and the cost model) evaluate, rank and select the base-classifiers according to their performance with respect to that metric. For brevity, we plotted only the best performing meta-learning algorithms, the accuracy of the Ripper meta-classifiers and the $(TP - FP)$ rates and savings of the Bayesian meta-classifiers.

The vertical lines in the figures denote the number of base classifiers integrated in the final meta-classifier as determined by the specialty/coverage algorithms. The final Chase meta-classifier for the $(TP - FP)$/coverage algorithm, for example, combines 33 base classifiers (denoted by the $(TP - FP)$ vertical line), while the final First Union meta-classifier for the accuracy/coverage algorithm consists of 26 base classifiers (denoted by the accuracy vertical line). In these graphs we have included the intermediate performance results (i.e. the accuracy, $(TP - FP)$ rates and savings of the partially built meta-classifiers) as well as the performance results of the redundant meta-classifiers would have had, had we used more base-classifiers or not introduced the pruning phase. Vertical lines for the diversity-based and metric-specific pruning algorithms are not shown in these figures as they depend on real-time constraints and available resources as discussed in Section 3.4.

The benefits from the pruning methods are clear. Although the algorithms are based on greedy search techniques and heuristically determined termination conditions and there is no guarantee that they will always produce the best ensemble meta-classifier, these experiments demonstrate these algorithms to be successful in computing "good" combinations of base classifiers, at least, with respect, to the three evaluation metrics. Furthermore, these results establish that not all base classifiers are necessary in forming effective meta-classifiers. In all these cases the pruned meta-classifiers are more efficient (fewer base classifiers are retained) and at least as effective (accuracy, (TP-FP), savings) as the complete meta-classifiers and certainly superior to those of the "arbitrary"

[6]The $CCS/coverage$ pruning algorithm defined in Section 3.4 iterates through all classes c_k, and selects the classifiers that maximize the $CCS(c_k)$. The fraud detection problem, however, is a binary classification problem, hence the CCS algorithm is, initially, reduced to select the classifiers that maximize CCS(fraud)/coverage, (i.e f_{TP}(fraud) - f_{FP}(fraud)), and furthermore, reduced to $(TP - FP)$ to match the evaluation metric.

pruning method. Using more base-classifiers than selected (denoted by the vertical lines) has no positive impact on the performance of the meta-classifiers. Overall, the pruning methods composed meta-classifiers with 0.7% higher accuracy, 6.2% higher $(TP - FP)$ spread and \$180K/month additional savings over the best single classifier for the Chase data and 1.65% higher accuracy, 10% higher $(TP - FP)$ spread and \$140K/month additional savings over the best single classifier for the First Union data. These pruned meta-classifiers also achieve 60% better throughput, 1% higher $(TP - FP)$ spread and \$100K/month additional savings than the unpruned meta-classifier for the Chase data and 100% better throughput, 2% higher $(TP - FP)$ spread and \$10K/month additional savings for the First Union data. Finally, the meta-classifiers succeeded in improving the performance even over the single classifier trained over the entire data sets (10 months of data for training, 2 months for testing in a 6-fold cross validation fashion). The meta-classifiers increased the accuracy by 0.4%, the $(TP - FP)$ rate by 4.7% and the savings by \$144K/month for the Chase data set and 1.2%, 6.5% and \$108K/month for the First Union data set, respectively.

Analysis A head to head comparison between the various pruning algorithms seems to point to a contradiction. The simple metric-specific pruning methods choose better combinations of classifiers for the Chase data, while the specialty/coverage-based and diversity-based pruning methods perform better on classifiers for the First Union data. In fact, this observation is more pronounced in most of the plots (not shown here) generated from the meta-classifiers computed by the other learning algorithms.

The performance of a meta-classifier, however, is directly related to the properties and characteristics of its constituent base-classifiers. As we have already noted, the more diverse the set of base-classifiers is, the more room for improvement the meta-classifier has. For example, to obtain diverse classifiers from a single learning program Freund and Schapire [12] introduced the boosting algorithm that resamples the data set to artificially generate diverse training subsets. In our experiments, the diversity of the base classifiers is attributed, first, on the use of disparate learning algorithms, and second, on the degree the training sets are different. Although the first factor is the same for both Chase and First Union data sets, we postulate that this is not the case with the second. Recall that the First Union classifiers were trained on subsets of data of equal size and class distribution while the Chase base-classifiers were trained on subsets of data defined according to the date of the credit card transaction, which led to variations in the size of the training sets and the class distributions. As a result, in the Chase bank case, the simple metric-specific pruning

algorithm combines the best base-classifiers that are already diverse and hence achieves superior results while the specialty/coverage pruning algorithms combine diverse base-classifiers that are not necessarily the best. On the other hand, in the First Union case, the specialty/coverage pruning algorithms are more effective, since the best base-classifiers are not as diverse.

Figures 1 and 2 also support this observation. The performance of the Chase base classifiers varies significantly, even when comparing classifiers computed by the same learning algorithm, indicating higher diversity. Conversely, the First Union classifiers, with the exception of the significantly inferior Bayesian classifiers, lack this diversity especially when comparing classifiers computed by the same learning algorithm. To substantiate this conjecture,[7] we plotted Figure 5 as a means to visualize the diversity among the base classifiers. Each cell in the plot displays the diversity of a pair of base classifiers, with bright cells denoting a high degree of similarity and dark cells high degree of diversity. The bottom right half is allocated to Chase base classifiers while the top left half represents the First Union base classifiers. Each base-classifier is identical to itself as shown by the white diagonal cells. The Bayesian First Union base classifiers, for example, are very similar to each other (light color cells at the bottom right corner of the plot and above the diagonal) and fairly different from the rest (10 dark columns at the left side of the figure), an observation that confirms the results from Figure 2. The diversity plot clearly demonstrates that Chase base classifiers are more diverse than First Union base classifiers.

Furthermore, observe that in the First Union data the various pruning algorithms are comparably successful and their performance plots are less distinguishable. A closer inspection on the classifiers composing the pruned sets (C) revealed that the sets of selected classifiers were more "similar" (there were more common members) for First Union than for Chase. Moreover, the inspection showed that for the First Union meta-classifiers, the specialty/coverage based and diversity-based pruning algorithms tended to select mainly the ID3 base-classifiers for being more specialized/diverse[8] thus substantiating the conjecture that the primary source of diversity for First Union meta-classifiers is the use of different learning algorithms. If the training sets for First Union were more diverse, there would have been more diversity among the other base-classifiers and presumably more variety in the pruned set. In any event, all pruning methods tend to converge after a certain point. After all, as they add classifiers, their pool of selected classifiers is bound to

[7] After all, similarity in performance does not necessarily imply high correlation between classifiers, e.g. it is possible for two classifiers to have different predictive behavior and exhibit similar accuracy.

[8] The ID3 learning algorithm is known to overfit its training sets. In fact, small changes in the learning sets can force ID3 to compute significantly different classifiers.

converge to the same set.

Note that training classifiers to distinguish fraudulent transactions is not a direct approach to maximizing savings (or the $TP - FP$ spread). In this case, the learning task is ill-defined. The base-classifiers are unaware of the adopted cost model and the actual value (in dollars) of the fraud/legitimate label. The effects of this shortcoming are best demonstrated in Figure 1. The most accurate classifiers are not necessarily the most cost effective. Although the Bayesian base classifiers are less accurate than the Ripper and C4.5 base classifiers, they are by far the best under the cost model. Similarly, the meta-classifiers are trained to maximize the overall accuracy not by examining the savings in dollars but by relying on the predictions of the base-classifiers. In fact, Figure 1 reveals that with only a few exceptions, the Chase base-classifiers are inclined towards catching "cheap" fraudulent transactions and for this they exhibit low savings scores. Naturally, the meta-classifiers are trained to trust the wrong base-classifiers for the wrong reasons, i.e. they trust the base-classifiers that are most accurate instead of the classifiers that accrue highest savings.

The superior performance of the simple cost-specific pruning method confirms this hypothesis. The cost-specific pruning method evaluates the base classifiers with respect to the cost model. The algorithm forms meta-classifiers by selecting base-classifiers based on cost savings performance. The performance of this algorithm is displayed in the right plot of Figure 3 under the name "cost". While the ensemble consists of base classifiers with good cost model performance the meta-classifier exhibits substantially improved performance as well. The curves for the First Union data set were not as distinct since the majority of the First Union base-classifiers happened to catch the "expensive" fraudulent transactions anyway (see Figure 2), so the pruning algorithms were able to form meta-classifiers with the appropriate base classifiers. The same, but to a lesser degree, holds for the $(TP-FP)$ spread. In general, unless the learning algorithm's target function is aligned with the evaluation metric, the resulting base- and meta-classifiers will not be able to solve the classification problem as best as possible except perhaps by chance. In such cases, it is often preferable to discard from the ensemble the base classifiers that do not exhibit the desired property.

One way to deal with this situation in to use cost-sensitive algorithms, i.e. algorithms that employ cost models to guide the learning strategy. On the other hand, this approach has the disadvantage of requiring significant change to generic algorithms. An alternative, but (probably) less effective technique is to tune the learning problem according to the adopted cost model. In the credit card fraud domain, for example, we can transform the binary classification problem into a

multi-class problem by multiplexing the binary class and the continuous amt attribute (properly quantized into several "bins"). The classifiers derived from the modified problem would perhaps fit better to the specification of the cost model and ultimately achieve better results.

5.4 Post-Training Pruning experiments

As with the pre-training pruning experiments, the post-training pruning algorithms are evaluated on their ability to choose the best set of base classifiers among the 50 that are available per data site. As before, the final meta-classifiers correspond to the higher level models derived by the same five learning algorithms. The difference in these tests is that the post-training pruning algorithms start with pre-computed (complete or pruned) meta-classifiers and search for less expensive but effective combinations of base classifiers. The experiments examine the effectiveness of the cost-complexity pruning algorithm and compare it against the correlation-based back pruning method.

The performance results of the pruned meta-classifiers for the Chase and First Union data sets are presented in Figures 6 and 7, respectively. As with the previous experiments, we display the accuracy (left plots), the $(TP - FP)$ spreads (middle plots) and the savings (right plots) achieved. According to these figures, in most of the cases, the cost-complexity post-training pruning algorithm computes superior meta-classifiers compared to the correlation-based pruning method. In general, the cost-complexity algorithm is successful in pruning the majority of the base-classifiers without any performance penalties. In contrast, the correlation-based pruned meta-classifiers perform better only with respect to the cost model and only for the Chase data set. This can be attributed to the manner in which the specific meta-classifiers correlate to their base-classifiers and not to the search heuristics of the method. The pruning method favors (retains) the base classifiers that are more correlated to the Bayesian meta-classifier which in this case happens to be the Bayesian base classifiers with the higher cost savings. As a result, the pruned meta-classifiers demonstrate increased savings even compared to the complete meta-classifier. Conversely, the cost-complexity pruning algorithm generates pruned meta-classifiers that best match the behavior and performance of the complete meta-classifiers. Overall, the correlation-based pruning cannot be considered a reliable pruning method unless only a few base classifiers need to be removed, while the cost-complexity algorithm is more robust and appropriate when many base classifiers need to be discarded.

The degree of pruning of a meta-classifier within a data site is dictated by the throughput re-

quirements of the particular problem. There is trade-off between the throughput and the predictive performance of classification systems. Higher throughput requirements necessitate higher pruning and that implies that only the most important available models can be combined. Fortunately, the cost-complexity pruning algorithm is fairly intelligent in discerning the set of classifiers that is most "important". Figure 8 demonstrates the algorithm's effectiveness on the Chase (left) and First Union (right) classifiers by displaying the predictive performance and throughput of the pruned meta-classifiers as a function of the degree of pruning. To normalize the different evaluation metrics and better quantify the effects of pruning, we measured the ratio of the performance improvement of the pruned meta-classifier over the performance improvement of the complete (original) meta-classifiers. In other words, we measured the performance *gain* ratio $G = \frac{P_{PRUNED} - P_{BASE}}{P_{COMPLETE} - P_{BASE}}$, where P_{PRUNED}, $P_{COMPLETE}$ and P_{BASE} denote the performance (accuracy, $(TP - FP)$ spread and savings) of the pruned meta-classifier, the complete meta-classifier and the best base classifier, respectively. Values of $G \simeq 1$ indicate pruned meta-classifiers that sustain the performance levels to that of the complete meta-classifier while values of $G < 1$ indicate performance losses. When only the best base classifier is used, there is no performance improvement and $G = 0$. In this figure, the black colored bars represent the accuracy gain ratios, the dark gray colored bars represent the $(TP - FP)$ gain ratios and the light gray bars represent the savings gain ratios of the pruned meta-classifier. The very light gray bars correspond to the relative throughput of the pruned meta-classifier T_P to the throughput of the complete meta-classifier T_C. To estimate the throughput of the meta-classifiers, we measured the time needed for a meta-classifier to generate a prediction. This time includes the time required to obtain the predictions of the constituents base classifiers sequentially on an unseen credit card transaction, the time required to assemble these predictions into a single meta-level "prediction" vector and the time required for the meta-classifier to examine the vector and generate the final prediction. The measurements were performed on a Personal Computer with a 200MHz Pentium processor running Solaris 2.5.1. These measurements show that cost-complexity pruning is successful in finding Chase meta-classifiers that retain their performance levels to 100% of the original even with as much as 60% of the base classifiers pruned or within 60% of the original with 90% pruning. At the same time, the pruned classifiers exhibit 230% and 638% higher throughput. For the First Union base classifiers, the results are even better. With 80% pruning, the pruned meta-classifiers have gain ratios $G \simeq 1$ and with 90% pruning they are within 80% of the original performance. The throughput improvement in this case is 5.08 and 9.92 times better, respectively.

5.5 Combining Pre-Training and Post-Training Pruning

Pre-training pruning and post-training pruning are complementary methods that employ different metrics to discard base classifiers. It is possible to visualize and contrast the manner the different pruning methods select or discard the base classifiers by computing the meta-classifier correlation matrices and by mapping them onto a modified *density* plot. Figure 9, for instance, displays three plots for the Chase data set that represent the correlation between the meta-classifiers and the base classifiers as selected by the specialty/coverage (left), the correlation-based (middle) and the cost-complexity (right) pruning algorithms, respectively. In these plots, each column corresponds to a single base classifier and each row corresponds to one meta-classifier, with the complete (unpruned) meta-classifier mapped on the bottom row, and the single base classifier mapped to the top. Dark gray cells represent meta-classifiers and base-classifiers that are highly correlated, while white cells identify the pruned base classifiers. The 3-column density plots accompanying the correlation-density plot represent the performance of the individual meta-classifiers, namely their accuracy (left column), their $(TP - FP)$ spread (middle column) and their savings (right column), with dark colors signifying "better" meta-classifiers.

Apart from exposing the relationships between meta classifiers and base classifiers, these plots provide information regarding the order the various base classifiers were selected or discarded. They can be read top-down for the pre-training pruning algorithms and bottom-up for post-training pruning. The specialty/coverage method, for example, begins by selecting a Ripper classifier (which corresponds to the January'96 data) and continues by adding mostly ID3 and CART classifiers.

The pruning algorithms examined in this paper can be briefly described as iterative search methods that greedily (without back-tracking) add (in pre-training) or remove (in post-training) one classifier per iteration. Assuming that the data sets and the base classifiers are common to all pruning algorithms, the meta-classifiers enumerated by these methods (recall that the pruning methods discussed do not require intermediate meta-classifiers), depends on the starting point (e.g the first base-classifier selected, the initial meta-classifier, etc.) on the search heuristic (metrics) evaluating the classifiers at each step, and on the portion of the validation data used for the evaluation. As a result, different pruning algorithms compute different combinations of classifiers and search distinct parts of the space of possible meta-classifiers. The correlation plots introduced above can serve as tools for studying the properties and limitations of the various pruning algorithms. Figure 9, for example, demonstrates that the three pruning algorithms consider very

different combinations of base-classifiers. The first (Specialty/Coverage) tends to select diverse classifiers (c.f. to Figure 5), the second (correlation-based) clusters the classifiers by the algorithm type, while the third (cost-complexity) forms more complex relationships and thus appears more balanced and robust.

By allowing pre-training pruning to compute both the intermediate and the larger meta-classifiers, (in a manner similar to that of Figures 3 and 4), and by applying post-training pruning to the best meta-classifier, it is possible to combine two different parts of the meta-classifier space and produce a final pruned meta-classifier that is even more effective and more efficient. The rationale parallels the strategy used in decision tree algorithms: first grow a large tree to avoid stopping too soon and then prune the tree back to compute subtrees with lower misclassification rate. In this case, post-training pruning is applied to the best intermediate meta-classifier which may not necessarily be the largest (complete) meta-classifier. This is particularly the case when the learning algorithm's target function is not aligned with the evaluation metric and only a small number of classifiers might be appropriate.

Table 2 for the Chase data and Table 3 for the First Union data compare the effectiveness and efficiency of the complete meta-classifiers (first rows) to the most effective meta-classifiers computed by the pre-training pruning algorithms (second rows), the most effective meta-classifiers computed by the combination of the pre-training and the cost-complexity post-training pruning algorithms (third rows) and the most effective meta-classifiers computer by the pre-training and the correlation-based post training pruning algorithms (last rows). Effectiveness is measured by the meta-classifier's accuracy (first columns) the meta-classifier's $(TP-FP)$ spread (second columns) and the meta-classifier's savings (third columns) each accompanied by its degree of pruning (compared to the complete meta-classifier) and efficiency (relative throughput, denoted as T_P/T_C). As expected, there is no single best meta-classifier; depending on the evaluation criteria and the target learning task, different meta-classifiers should be constructed and deployed.

According to these experiments, pre-training pruning is able to compute more effective and more efficient meta-classifiers than those obtained by combining all available base classifiers, and post-training pruning and in particular the cost-complexity algorithm is successful in further reducing the size of these meta-classifiers. In fact, the larger the pre-trained pruned meta-classifier is, the larger the degree of additional pruning the cost-complexity algorithm exhibits. The post-training pruning algorithm that is based on the correlation metric, however, provides either limited or, is some cases, no further improvement, e.g. see the $(TP - FP)$ column of Table 3. Another obser-

vation to note, is that the throughput of a meta-classifier is also a function of its constituent base classifiers. Some classifiers are faster in producing predictions than others (due to size, different representations, implementation details, etc) and this may have a significant effect on the speed of the entire meta-classifier. As an example, observe the two post-trained pruned First Union meta-classifiers with the best $(TP - FP)$ spreads. They both have the same size, yet they are composed of different base classifiers and for that they exhibit different throughputs.

6 Conclusion

In this paper we studied efficient and effective methods that combine multiple models computed over distributed databases into scalable and efficient meta-classifiers. We addressed scalability and efficiency, first, by employing distributed and asynchronous protocols at the architectural level for managing the learning agents across the data sites of the JAM system, and second by introducing pruning algorithms at the data site level to evaluate, retain and use only the most essential classifiers in an ensemble meta-classifier. We introduced several evaluation metrics and pre-training pruning algorithms for selecting classifiers before they are combined into a meta-classifier, and a post-training pruning algorithm based on the cost-complexity pruning of decision tree for discarding classifiers after being combined into a meta-classifier.

On real credit card transaction data sets from two separate institutions we demonstrated that pruning base classifiers in meta-learning can achieve similar or better performance results than the brute-force assembled meta-classifiers but at a substantially reduced computational cost and higher throughput. With the help of correlation-based visualization tools we analyzed base classifiers and meta-classifiers and we established that the suitability of a pre-training pruning method depends on the particular set of candidate base classifiers and the target learning task. Furthermore, we determined cost-complexity post-training pruning methods to be an efficient, intuitive and robust method for discarding the base classifiers and we illustrated that combining cost-complexity pruning and pre-training pruning can yield meta-classifiers with further improved effectiveness and efficiency.

In these experiments, we considered only simple one-level meta-classifiers, i.e. meta-classifiers integrating only base classifiers. Our future plans include the extension of these methods to meta-classifier hierarchies. As more databases become available, we expect that the pruning of larger meta-classifier hierarchies will be of even greater importance.

A Appendix

Table 1: The Diversity-based (left) and Coverage/Specialty-based (right) pruning algorithms

Let $\mathcal{C} := \emptyset$, $N :=$ maximum number of classifiers
For i := 1, 2, ..., $|\mathcal{H}|$ - 1 **do**
  For j := i, i+1, ..., $|\mathcal{H}|$ **do**
    Let d_{ij} := the ratio of the instances the C_i and C_j give different predictions
Let C' := the classifier with the highest accuracy
$\mathcal{C} := \mathcal{C} \cup C'$, $\mathcal{H} := \mathcal{H} - C'$
For i := 1, 2, ..., N **do**
  For j := 1, 2, ..., $|\mathcal{H}|$ **do**
    Let $D_j := 2 \cdot (\sum_{k=1}^{|\mathcal{C}|} d_{jk})/|\mathcal{C}| \cdot (|\mathcal{C}| + 1)$
  Let C' := the classifier from $\mathcal{H}$ with the highest D_j
  $\mathcal{C} := \mathcal{C} \cup C'$, $\mathcal{H} := \mathcal{H} - C'$

Let $\mathcal{C} := \emptyset$
For all target classes c_k, k = 1, 2, ..., c, **do**
  Let $\mathcal{E} := \mathcal{V}$
  Let $\mathcal{C}_k := \emptyset$, $\mathcal{H}_k := \emptyset$
  Until no other examples in $\mathcal{E}$ can be covered or $\mathcal{H}_k = \emptyset$
    Let C' := the classifier with the best *class specialty* on target class c_k for $\mathcal{E}$
    $\mathcal{C}_k := \mathcal{C}_k \cup C'$, $\mathcal{H}_k := \mathcal{H}_k - C'$
    $\mathcal{E} :=$ $\mathcal{E}$ - examples covered by C'
  $\mathcal{C} := \mathcal{C} \cup \mathcal{C}_k$

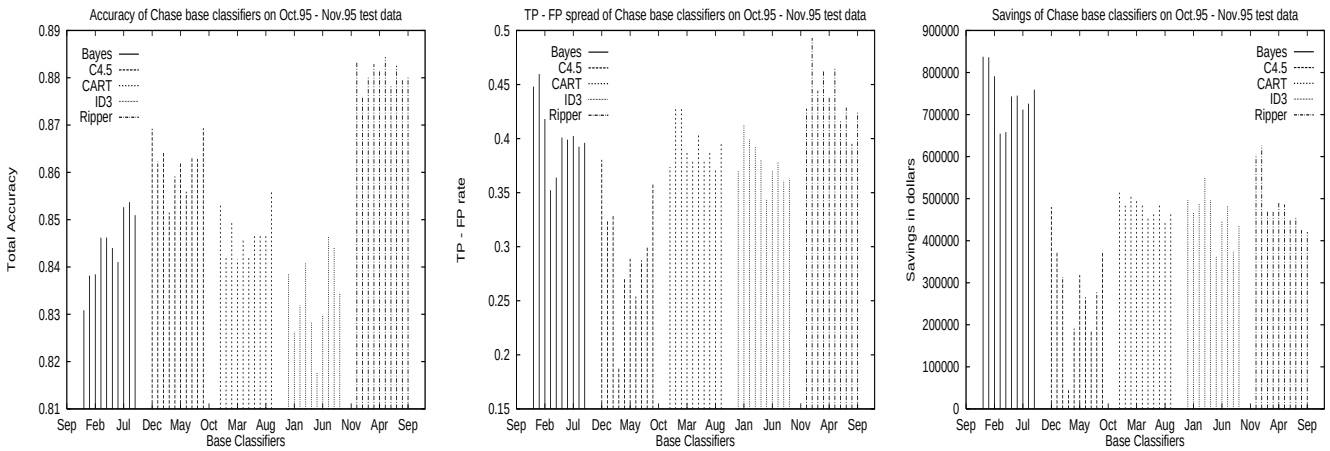

Figure 1: Base Classifiers: Total accuracy (left), (TP-FP) spread (middle) and savings (right) on Chase credit card data.

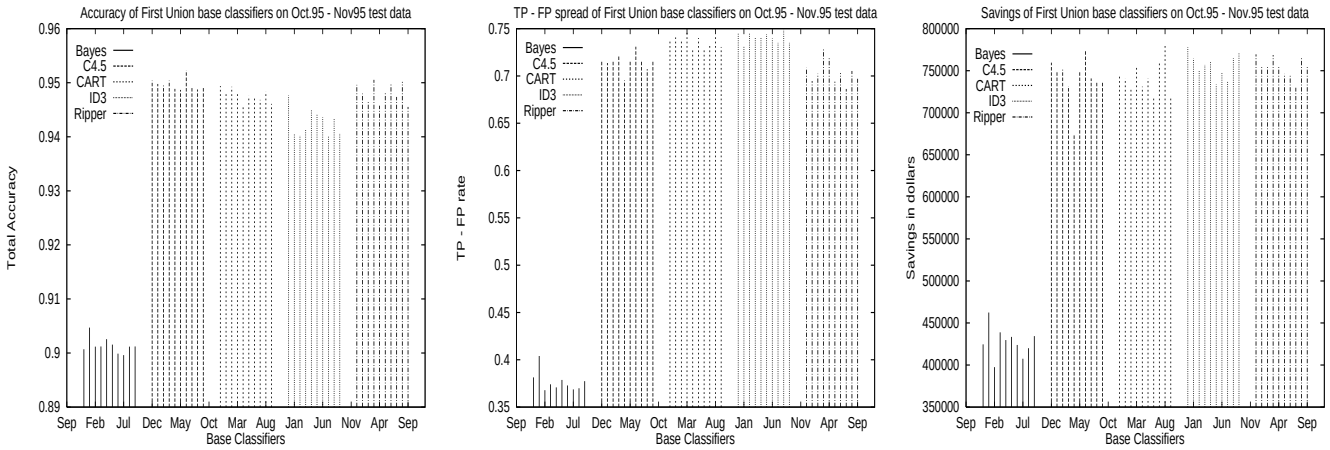

Figure 2: Base Classifiers: Total accuracy (left), (TP-FP) spread (middle) and savings (right) on First Union credit card data.

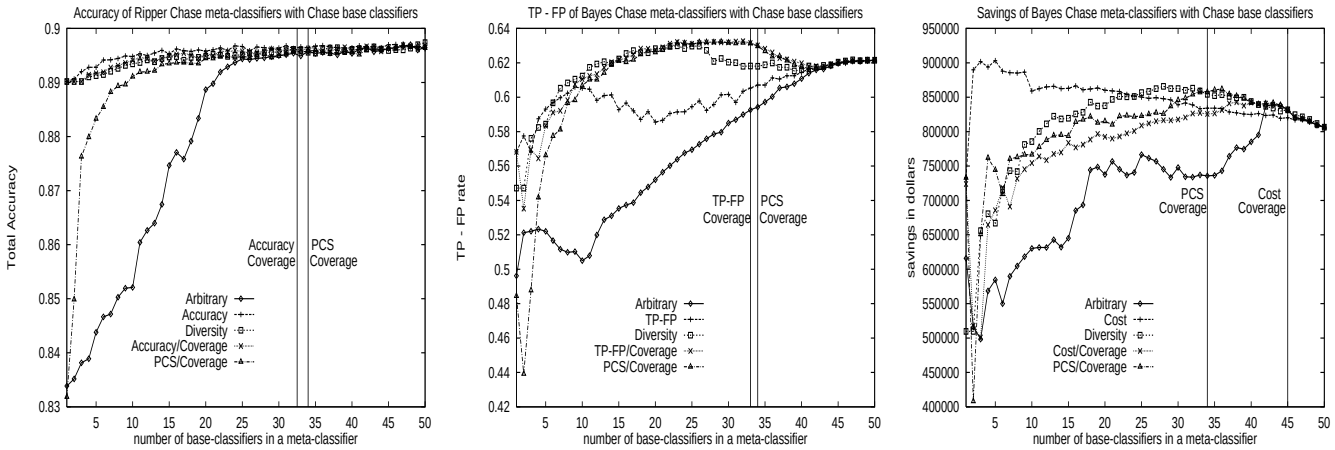

Figure 3: Pruning algorithms: Total accuracy (left), (TP-FP) spread (middle) and savings (right) on CHASE credit card data.

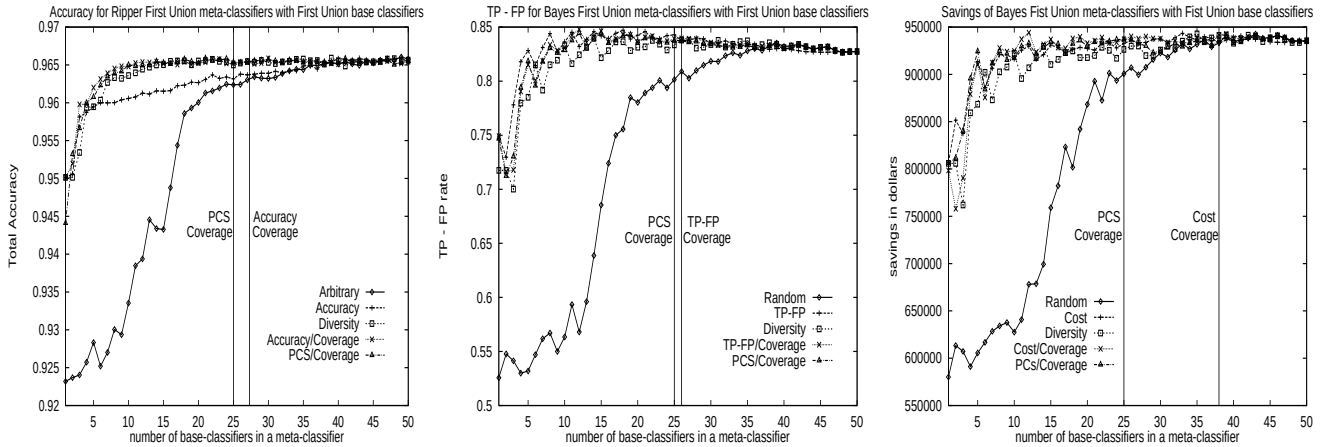

Figure 4: Pruning algorithms: Total accuracy (left), (TP-FP) spread (middle) and savings (right) on First Union credit card data.

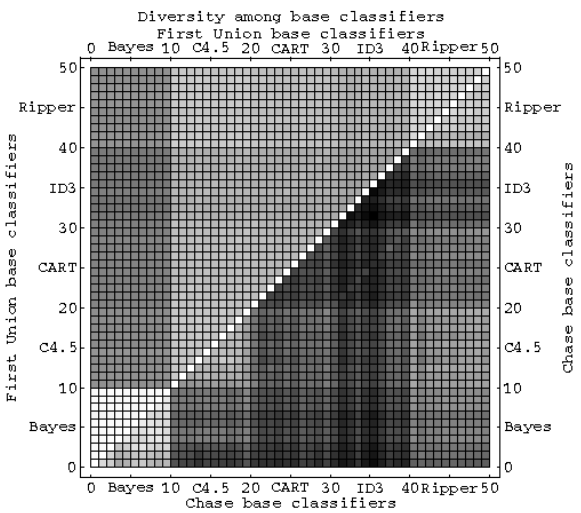

Figure 5: Base classifier diversity plot between all pairs of Chase base classifiers (bottom right half) and all pairs of First Union base classifiers (top left half).

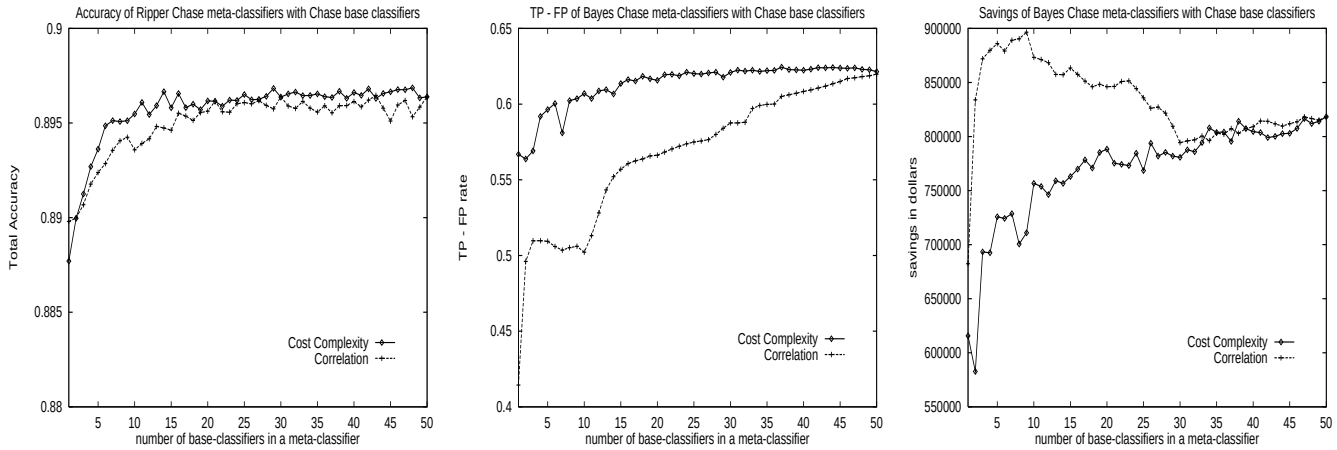

Figure 6: Post-Pruning algorithms on Chase data: Total accuracy (left), (TP-FP) spread (middle) and savings (right).

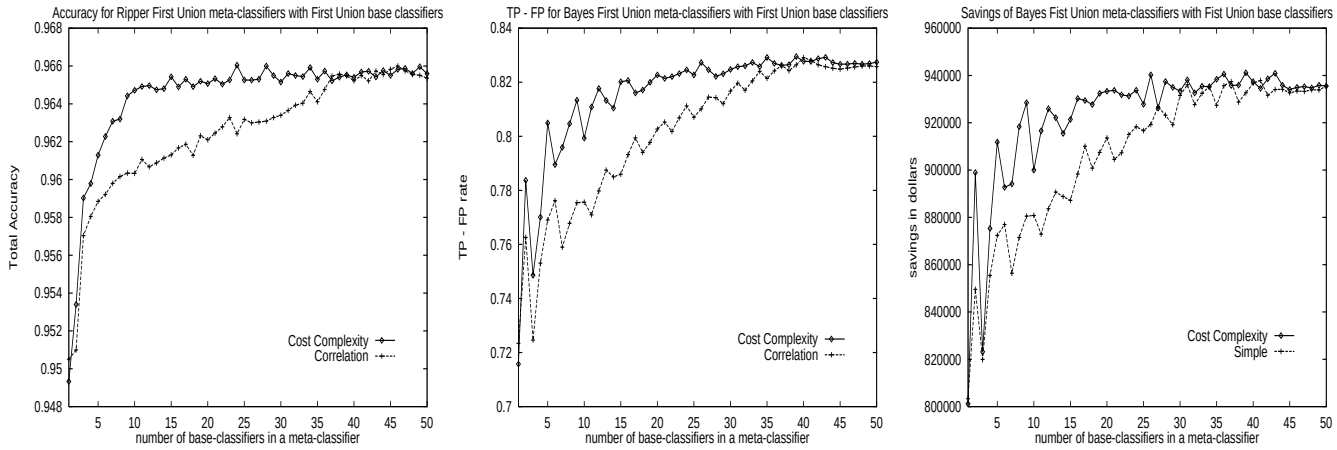

Figure 7: Post-Pruning algorithms on First Union: Total accuracy (left), (TP-FP) spread (middle) and savings (right).

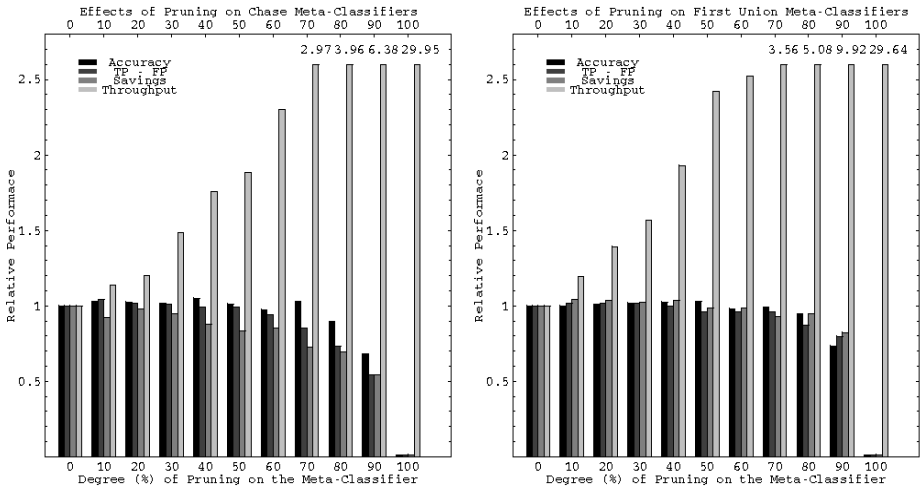

Figure 8: Bar charts of the accuracy (black bars), TP - FP (dark gray bars), savings (light gray bars) and throughput (very light gray bars) of the Chase (right) and First Union (left) meta-classifiers as a function of the degree of pruning.

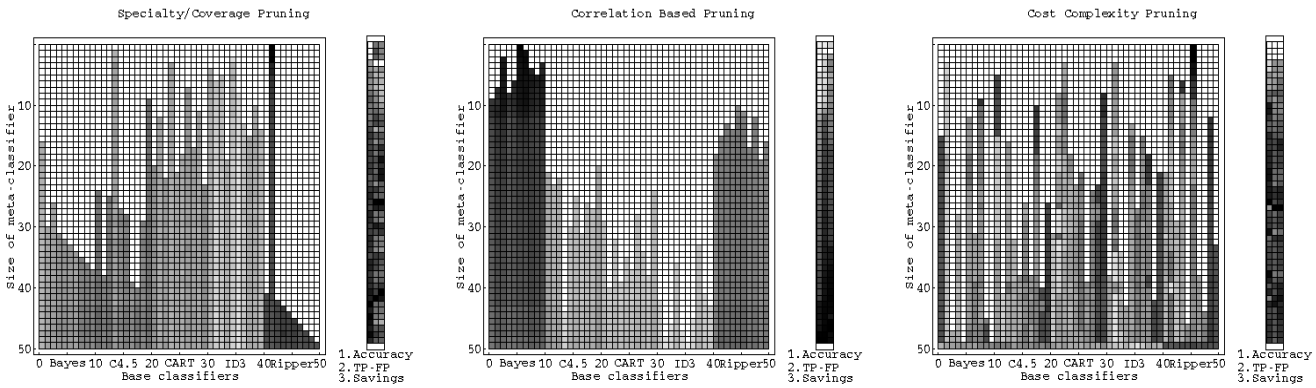

Figure 9: Selection of classifiers on Chase data: Specialty/Coverage Pruning (left) Correlation Pruning (middle) and Cost-Complexity Pruning (right).

Table 2: Performance and Relative Throughput of the Best Chase Meta-Classifiers

	Accuracy	Pruning	T_P/T_C	TP - FP	Pruning	T_P/T_C	Savings	Pruning	T_P/T_C
Complete	89.57	0 %	1.0	62.05	0 %	1.0	818K	0 %	1.0
Pre Training	89.66	30 %	1.43	63.13	54 %	2.01	935K	90 %	6.5
+ Post (CC)	89.66	72 %	3.18	63.1	64 %	2.57	923K	92 %	8.25
+ Post (LC)	89.62	38 %	1.6	63.13	54 %	2.01	931K	94 %	10.11

Table 3: Performance and Relative Throughput of the Best First Union Meta-Classifiers

	Accuracy	Pruning	T_P/T_C	TP - FP	Pruning	T_P/T_C	Savings	Pruning	T_P/T_C
Complete	96.47	0 %	1.0	82.53	0 %	1.0	935K	0 %	1.0
Pre Training	96.53	54 %	2.48	84.61	72 %	4.33	940K	8%	1.16
+ Post (CC)	95.53	72 %	3.87	84.52	78 %	4.78	940K	48%	2.32
+ Post (LC)	96.53	54 %	2.48	84.69	78 %	5.25	940K	12%	1.25

References

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

[2] L. Breiman. Heuristics of instability in model selection. Technical report, Department of Statistics, University of California at Berkeley, 1994.

[3] L. Breiman. Stacked regressions. *Machine Learning*, 24:41–48, 1996.

[4] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, CA, 1984.

[5] C. Brodley and T. Lane. Creating and exploiting coverage and diversity. In *Work. Notes AAAI-96 Workshop Integrating Multiple Learned Models*, pages 8–14, 1996.

[6] C.Brodley. Addressing the selective superiority problem: Automatic algorithm/model class selection. In *Proc. 10th Intl. Conf. Machine Learning*, pages 17–24. Morgan Kaufmann, 1993.

[7] P. Chan. *An Extensible Meta-Learning Approach for Scalable and Accurate Inductive Learning*. PhD thesis, Department of Computer Science, Columbia University, New York, NY, 1996.

[8] P. Chan and S. Stolfo. Meta-learning for multistrategy and parallel learning. In *Proc. Second Intl. Work. Multistrategy Learning*, pages 150–165, 1993.

[9] W. Cohen. Fast effective rule induction. In *Proc. 12th Intl. Conf. Machine Learning*, pages 115–123, 1995.

[10] T.G. Dietterich. Machine learning research: Four current directions. *AI Magazine*, 18(4):97–136, 1997.

[11] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, pages 23–37. Springer-Verlag, 1995.

[12] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proc. Thirteenth Conf. Machine Learning*, pages 148–156, 1996.

[13] Michael J. Greenacre. *Theory and Application of Correspondence Analysis*. Academic Press, London, 1984.

[14] L. Hansen and P. Salamon. Neural network ensembles. *IEEE Trans. Pattern Analysis and Mach. Itell.*, 12:993–1001, 1990.

[15] R.A. Jacobs, M.I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixture of local experts. *Neural Computation*, 3(1):79–87, 1991.

[16] Michael I Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6:181–214, 1994.

[17] A. Krogh and J. Vedelsby. Neural network ensembles, cross validation, and active learning. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Info. Proc. Sys. 7*, pages 231–238. MIT Press, 1995.

[18] S. Kwok and C. Carter. Multiple decision trees. In *Uncertainty in Aritificial Intelligence 4*, pages 327–335, 1990.

[19] M. LeBlanc and R. Tibshirani. Combining estimates in regression and classification. Technical Report 9318, Department of Statistics, University of Toronto, Toronto, ON, 1993.

[20] D. Margineantu and T. Dietterich. Pruning adaptive boosting. In *Proc. Fourteenth Intl. Conf. Machine Learning*, pages 211–218, 1997.

[21] C. Merz. Using correspondence analysis to combine classifiers. *Machine Learning*, 1998. to appear.

[22] C. Merz and M. Pazzani. A principal components approach to combining regression estimates. *Machine Learning*, 1998. to appear.

[23] R. Michalski. A theory and methodology of inductive learning. In R. Michalski, J. Carbonell, and T. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 83–134. Morgan Kaufmann, 1983.

[24] M. Minksy and S. Papert. *Perceptrons: An Introduction to Computation Geometry*. MIT Press, Cambridge, MA, 1969. (Expanded edition, 1988).

[25] T. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.

[26] David W. Opitz and J. Jude W. Shavlik. Generating accurate and diverse members of a neural-network ensemble. *Advances in Neural Information Processing Systems*, 8:535–541, 1996.

[27] M. P. Perrone and L. N. Cooper. When networks disagree: Ensemble methods for hydrid neural networks. *Artificial Neural Networks for Speech and Vision*, pages 126–142, 1993.

[28] A. Prodromidis and S. Stolfo. Mining databases with different schemas: Integrating incompatible classifiers. In *Proc. 4rd Intl. Conf. Knowledge Discovery and Data Mining*, 1998.

[29] A. L. Prodromidis. On the management of distributed learning agents. Technical Report CUCS-032-97 (PhD Thesis proposal), Department of Computer Science, Columbia University, New York, NY, 1997.

[30] F. Provost and T. Fawcett. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *Proc. Third Intl. Conf. Knowledge Discovery and Data Mining*, pages 43–48, 1997.

[31] F. Provost and T. Fawcett. Robust classification systems for imprecise environments. In *Proc. AAAI-98*. AAAI Press, 1998.

[32] F. Provost and V. Kolluri. Scaling up inductive algorithms: An overview. In *Proc. Third Intl. Conf. Knowledge Discovery and Data Mining*, pages 239–242, 1997.

[33] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[34] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[35] R. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–226, 1990.

[36] S. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. Chan. Credit card fraud detection using meta-learning: Issues and initial results. Working notes of AAAI Workshop on AI Approaches to Fraud Detection and Risk Management, 1997.

[37] S. Stolfo, W.D. Fan, A. Prodromidis W.Lee, S. Tselepis, and P. K. Chan. Agent-based fraud and intrusion detection in financial information systems. Submitted to IEEE Symposium on Security and Privacy, 1998.

[38] S. Stolfo, A. Prodromidis, S. Tselepis, W. Lee, W. Fan, and P. Chan. JAM: Java agents for meta-learning over distributed databases. In *Proc. 3rd Intl. Conf. Knowledge Discovery and Data Mining*, pages 74–81, 1997.

[39] Volker Tresp and Michiaki Taniguchi. Combining estimators using non-constant weighting functions. *Advances in Neural Information Processing Systems*, 7:419–426, 1995.

[40] D. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.