IMPROVING PERFORMANCE OF INDUCTIVE MODELS THROUGH AN ALGORITHM AND SAMPLE COMBINATION STRATEGY[1]

HALEH VAFAIE, PHD., DEAN ABBOTT[*], MARK HUTCHINS, I. PHILIP MATKOVSKY

Northrop Grumman Information Technology,4800 Hampden Lane, Bethesda, MD 20814

[*]*Abbott Consulting, San Diego, CA 92192*

Multiple approaches have been developed for improving predictive performance of a system by creating and combining various learned models. There are two main approaches to creating model ensembles. The first is to create a set of learned models by applying an algorithm repeatedly to different training sample data, the second applies various learning algorithms to the same sample data. The predictions of the models are then combined according to a voting scheme. This paper presents a method for combining models that were developed using numerous samples, modeling algorithms, and modelers and compares it with the alternate approaches. The presented results are based on findings from an ongoing operational data mining initiative with respect to selecting a model set that is best able to meet defined goals from among trained models. The operational goals to be attained in this initiative are to deploy data mining model(s) that maximizes specificity with minimal negative impact to sensitivity. The results of the model combination methods are evaluated with respect to sensitivity and false alarm rates and are then compared against other approaches.

Keywords:

1. Introduction

Combining models is not a new concept for the statistical pattern recognition, machine learning, or engineering communities, though in recent years there has been an explosion of research exploring creative new ways to combine models. Currently, there are two main approaches to model combination. The first approach generates a set of learned models by applying an algorithm repeatedly to different samples of the training data. The second approach applies different learning algorithms to the same sample of data to generate a set of learned models [1]. The predictions of the models are then combined according to a voting scheme. The diverse research has also produced a proliferation of terminology used to describe combining models developed using the same algorithm. Elder and Pregibon [2] used the term *Blending* to describe "the ancient statistical adage that 'in many counselors there is safety'". The same concept has been described as *Ensemble of Classifiers* by Dietterich [3], *Committee of Experts* by Steinberg [4], and *Perturb and Combine* (P&C) by Breiman [5]. Jain, Duin, and Mao [6] call the methods merely *Combiners*. However, the concept of combining models is actually quite simple:

[1] Presented at The Twelfth International Conference on Tools with Artificial Intelligence, Vancouver, British Columbia, November 13-15, 2000.

train several models from the same data set, or from samples of the same data set, and combine the output predictions, typically by voting for classification problems and averaging output values for estimation problems. The reductions in model bias (errors) and variance have been shown to be significant.

For decision trees, Breiman [7] demonstrates *bagging* and ARCing improve single CART models on 11 machine learning datasets in every case. Bagging merely creates multiple classifiers from bootstrap samples of the training data, and then combines the decisions of the individual classifiers via voting or averaging. Additionally, he documents that ARCing, using no special data preprocessing or classifier manipulation (just read the data and create the model), often achieves the performance of handcrafted classifiers that were tailored specifically for the data.

Boosting, like Bagging, changes the training datasets by emphasizing some records over others, though Boosting puts more weight (directly) on records that the classifier has misclassified. The final decision is produced by a weighted sum of the classifiers, where the weighting is based on the performance of the individual classifiers. In other research, Dietterich [8] shows that bagging, boosting, or another method called randomization always beats a C4.5 tree.

There is also a large body of literature describing model ensembles using neural networks, Tumer and Ghosh [9] showed combining neural network classifiers with identically distributed error variance reduces the error variance linearly with the number of classifiers. Perrone and Cooper [10] showed that when combining neural networks linearly, the combined model will lower mean squared error more than any of the individual models, although this is only true for very restrictive circumstance: if there is independence of the error correlation matrix. In both of these studies, however, restrictive assumptions reduce the applicability to real problems, yet they show the power of the combination process nevertheless. Other researchers have also shown the advantage of combining the outputs of many neural networks, including using Bagging and Boosting (e.g. [11], [12], [13], [14], and [15]).

Intuitively, it seems that producing relatively uncorrelated output predictions in the models to be combined is necessary to reduce error rates. If output predictions are highly correlated, little reduction in error is possible, as the "committee of experts" have no diversity from which to draw, and therefore no means to overcome erroneous predictions. Decision trees are very unstable in this regard as small perturbations in the training data set can produce large differences in the structure (and predictions) of a model [16], i.e., decision trees are unstable. Neural networks are sensitive to data used to train the models and to the many training parameters and random number seeds that need to be specified by the analyst. Indeed, many researchers merely train neural network models changing nothing but the random seed for weight initialization and produce significantly different models. Polynomial networks have considerable structural instability, as different data sets can produce significantly different models.

There has been some success in combining models across algorithm types. Bauer and Kohavi [17] discuss the possibility of combining models across algorithm types, although they did not have success. Elder [18] called this approach *Bundling* and showed that it out-performed individual models on average in a direct marketing application. This approach also improved performance over a single model when applied to several other data sets from the Machine Learning Repository at UC/Irvine [19]. The same conclusion was found by Jain *et al* [6] on a digit classification problem. The work by Tresp and Tanguchi [20] and Merz [1] are other examples of this approach.

It has been shown that a good model ensemble is one where the individual models are both accurate and make their errors on different parts of the input [15]. So a case can be made for combining models across algorithm families as a means of providing *uncorrelated* output estimates because of the difference in basis functions used to build the model. In fact, the manner in which the combination of models are assembled (voting, averaging, and two types of advisor perceptrons) is not as important as performing *some* kind of combination [21].

Regarding the bias – variance tradeoff, ensemble methods like Bagging and Bundling are primarily variance reduction methods, that is, they reduce the likelihood of deploying a poor model by reducing the error variance when evaluating models on out-of-sample data. This does not mean that they *cannot* reduce model bias, and they do indeed, on average, reduce model bias, but they do not *necessarily* reduce bias. Methods that operate more directly on the modeling errors themselves, such as Boosting and ARCing, are more effective at reducing model bias.

The previous work show that by introducing diversity, either in terms of the various algorithms used or the sample training data sets, the overall performance is improved. Our hypothesis is that by combining the two criteria, i.e., generating model ensemble form models that were generated using different algorithms and training data set, the maximum diversity is allowed in the ensemble and therefore, the results should further be improved. The following section presents the real world problem domain, fraud detection, we selected for our initial studies. The fraud detection problem has several common issues which will be described in detail.

2. Problem Description

The detection of illegal, improper, or unusual transactions is a constant challenge for any business. Both the private and public sectors have worked extensively to detect fraud in their systems. The extent of fraud, while varying from application to application, can be quite large. Cellular phone fraud alone costs the industry hundreds of millions of dollars per year [22], and fraudulent transfers of illegally obtained funds (money laundering) are estimated to be as much as $300 billion annually [23].

For the current research, the problem included establishing a data set of known fraudulent payments, a target population of over one million non-fraudulent payments, and a method by which to leverage the known fraud cases in the training of detection models. As is typical in fraud detection, the set of known cases was very small relative to the number of non-fraud examples. Thus, the authors had to devise methods to reduce false alarms without drastically compromising the sensitivity of trained models.

Rather than using a single fraud/not-fraud binary label for the output variable, four fraudulent payment types, called types A, B, C, and D, was identified as comprising the different styles of payments in the known fraud data. Although separating the known fraudulent payments into types was not essential to the modeling process, the authors believed that the additional information would aid classifiers in identifying suspicious payments, and reduce false alarms. The primary reason for this belief was that the types of known fraudulent payments had significantly different behaviors, which would cause a classification algorithm to try to segment the fraudulent payment data into groups anyway. By creating the payment types beforehand, simpler models were more likely to be formed.

3. Creating Training, Testing, and Validation Subsets

A procedure of using training, testing, and validation data sets was used throughout the model building process to reduce the likelihood of overfitting models. Overfitting has several harmful side effects, first of which is poorer performance on unseen data than is expected from the training error rates [24]. Additionally, it also has the effect of selecting a model that includes extra variables or weights that are spurious. Therefore, not only is the model less efficient, but it also identifies the **wrong** patterns in the data. If our objective is to find the best model that truly represents the patterns in the data, great care should be taken to avoid overfit.

The common practice of using both a training data set to build a model and a testing data set to assess the model accuracy reduces the likelihood of overfit. However, the data mining process is usually iterative, including the training/testing cycle. The testing data set provides an indication of how well the model will perform on unseen data. But after several iterations of training and testing, the testing data set ultimately guides the selection of candidate inputs, algorithm parameters, and "good" models. Therefore, the testing data set ceases to be independent and becomes an integral part of the model itself. To guard against this effect, a third (*validation)* data set is used to assess model quality (only once) at the end of the training/testing cycles. Before dividing the data into training, testing, and validation sets, there were three problems to overcome specific to the data set used in this application.

Data Problem 1. The first difficulty, common in fraud detection problems, was the small number of labeled fraud payments or transactions available for modeling. The difficulty was particularly acute here with our desire to have three data subsets for each model (training, testing, and validation). However, with precious few known fraudulent payments available, keeping any of them out of training data meant that patterns of fraudulent payment behavior *may* have been withheld from the models, and therefore missed once the models were deployed. To overcome this data problem, the authors used cross-validation. In cross-validation, data are split into training, testing, and sometimes validation data sets multiple times, so that each fraud payment is included in each of the data sets. The number of payments used in the training set can include all the payments except one, keeping the unused payment for testing, and repeat the splitting until each payment has been excluded from the training set (included in the testing set) one time. However, because thousands of payments were used in training our models, this was deemed too computationally expensive. Most of the benefit of cross-validation can be achieved from a much smaller number of cross-validation folds, and for this project, it was decided that 11 subsets would be sufficient.

Data Problem 2. A second problem related to the assumption of independence of database rows—individual payments—to one another. Typically, the splitting of data into training, testing, and validation subsets, is done randomly. However, this procedure assumes independence between rows (payments) in the data. Upon further examination, it became clear that the modeling variables that were a part of the payments tended to be correlated within each payee, because payees tend to invoice for work that is similar from payment to payment. For example, a telephone company may have a similarly sized invoice for long distance service each month. If some payments from a payee are included in both the training and testing subsets, testing results will be unrealistically optimistic. This occurs because the same pattern or a very similar pattern will exist for the same payee in both the training data (which is used to create a model) and the testing data (which is used to assess the model). When the model is deployed, most of the payees will not have been seen during training; the purpose of the model is to generalize payment patterns, not profile individual payees.

The solution was not only to select payees randomly and to include each in only one of the data sets (training, testing, *or* validation), but also to keep together all the fraudulent payments associated with each payee during the split. For example, if XYZ Corporation was chosen to be included in the testing data, all other payments made to XYZ Corporation were also included in the testing data.

Data Problem 3. A third problem resulted from the lack of adjudicated non-fraudulent payments in the general population. These payments may have been improper or non-fraudulent, and this label was unknown when collected in the database. It was assumed for the purposes of modeling, however, that they were non-fraudulent because this was far more likely to be true. Additionally, for small data sets, it was unlikely that any of the payments would have been labeled incorrectly as non-fraudulent when they were actually fraudulent. However, for larger data sets, perhaps over 100,000 payments, the likelihood of having mislabeled payments included in the training or testing sets was much higher. The modeling classifiers could become confused if too many mislabeled payments are included. If the mislabeled payments happened to correspond to patterns of fraud, these payments may be dismissed erroneously in an effort to reduce false alarms.

To compensate for this effect, training data sets of approximately 4,000 payments each (compared to a population of 1.4 million) would keep the likelihood of mislabeled payments sufficiently small so that training would not be negatively effected. The testing data sets were selected to have 2,000 payments from the population. This split of training and testing (2/3, 1/3) is common among machine learning community. By keeping the data sets small, the time needed to train and test models was also reduced significantly. A much larger 125,000-payment data set was used for model validation. The general (non-fraudulent payment) population was split into the 11 training and testing sets randomly based on the above mention criteria. For validation data, a single set of 125,000 payments was randomly selected from the population. Additional considerations were given to the non-fraudulent payment population for generating diversity, but will not be discussed in depth here. For a more detailed description of the process, see Abbott, *et al* [25].

Known fraud data, with the four type labels (A, B, C, and D), were split following a different procedure. For types A, B, and C, all payments associated with one payee were selected randomly for testing data, all payments associated with a random payee were selected for validation data, and all the payments associated with the remaining payees were put into the training data. In this way, we included as many patterns in the training data as possible for each split. For type D, because so few payees were available, the payments were split randomly between training and testing data sets, and another payee was held out for validation. Then known fraud splits were combined with the non-fraudulent payment data in each of the 11 splits. We must note that the validation data set was the same for each of the 11 splits, while the fraudulent payment data sets were not and contained fraudulent payments not included in the respective training or testing sets.

4. Criteria for Scoring Models

A superset of over 100 models was developed using SPSS, Inc. Clementine as the data mining toolset. The algorithms available for modeling in Clementine are neural networks, decision trees, rule induction, and association rules. Multiple algorithms were used on all 11 data sets to develop the models. Models were trained on the training data set and tested on the corresponding testing data set. All of the models were scored with respect to sensitivity and false alarm rate on the training and testing data. To rank the models percentages for the following three areas were calculated:

1) Fraudulent payment Sensitivity: the percentage of fraudulent payments correctly called fraud from the known fraud population. Sensitivity predicts the likelihood that we will find fraudulent payments in the general population.
2) False Alarm Rate: the percentage of non-fraudulent payments incorrectly called fraudulent from the general population.
3) True Alarm Rate: the percentage of fraudulent payments predicted as fraudulent from the entire population of payments predicted as fraudulent. True alarm rate predicts the density of fraudulent payments that will populate the list of fraudulent payment predictions.

5. Experiments and Results

In this section, we describe a series of experiments that were performed to empirically validate our hypotheses. These experiments involved generation of four model ensemble. The first combination was generated to represent the ensemble generated from models that used the same algorithm on many sample data. In this case the selected models were the best neural network models that were produced using 10 sample sets. The second model combination is a representative of the ensemble generated from models that used the same sample data with various algorithms. Eleven best models generated for a sample were selected for this model ensemble. The third and fourth model ensembles represent variations of our approach which many sample data and algorithms are used to generate the candidate models. The results of these four model ensemble are contrasted with respect to target class sensitivity and false alarm rate.

The following section reviews the criteria for evaluating the generated models and creating the above mentioned model ensembles. Finally the results from each of the model selection approaches are compared against each other.

5.1 Evaluating Models to Be Included in the Ensembles

Evaluation of models to be included in the ensembles used a weighting system to combine the testing and validation results. The weights applied to testing scores and validation scores were 30% for testing scores and 70% for validation scores. Scores for testing and validation results were then added together to produce an overall score and the overall score of a model was then ranked. The models selected for a given ensemble were selected based on performance and other criteria described below. In order to perform a fair comparison the final result of all the ensembles were generated using the same criterion, a simple vote of the outputs.

Any number of votes could be required to call a payment suspicious. After a brief investigation, it was determined that a majority vote provided the best tradeoff. Our ultimate goal was to minimize the false alarms (non-fraudulent payments that were flagged by the models) and maximize sensitivity (known fraudulent payments that were flagged by the models). The final decision was based on the majority vote; if a majority of the models in the ensemble classified a payment as "suspicious," that payment was labeled as "suspicious" and a candidate for further investigation.

Model Ensemble 1: Using the same algorithm

For generating this model ensemble the selected models were the best neural network models that were produced using 10 sample sets. These models were compared against other neural network models generated for the same sample and were selected based on their combined best score of both sensitivity and false alarm rate. With 10 models in this model ensemble, a vote of five or more meant the payment is suspicious.

Figure 1 shows the model sensitivity of all 10 models and the combination on the validation data set. Note that the combination was much better than average, and the best model overall. This behavior is typical in voting combinations [19].

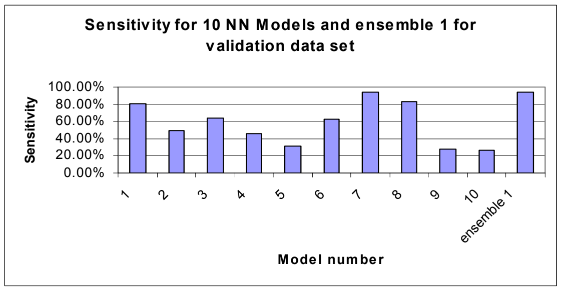

Figure 1: Model Sensitivity Comparison

In Figure 2, the false alarm performance is shown. As with sensitivity, the combination model had a much lower score than the average model, though model 2 had the overall lowest false alarm rate. It is clear, however, that *any* positive, non-zero weighting of sensitivity and false alarms results in the combination having the best score overall. For example, Figure 3 shows an equal weighting between sensitivity and false alarms.

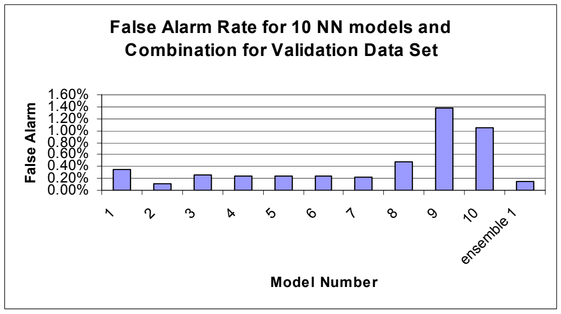

Figure 2: False Alarm Rate Comparison

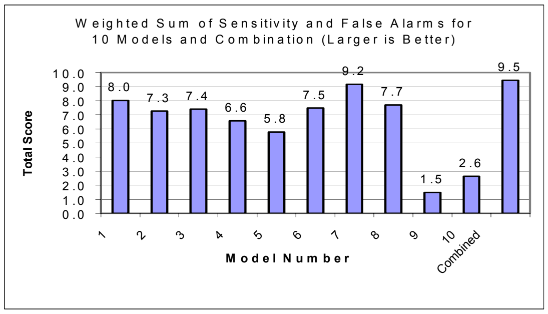

Figure 3: Total Weighted Score Comparison

Model Ensemble 2: Using the same sample

For generating this model ensemble one of the sample sets was randomly selected. Then eleven best models generated for that sample were selected to be included in the model ensemble. These models were selected based on their combined best score of both sensitivity and false alarm rate compared to other models generated using this sample. Table 1 lists the algorithms used in this model ensemble.

Tab1e 1: Algorithms Used to Create Models Included in Model Ensemble 2.

Model Number	Algorithm Type
1	Decision tree
2	Neural Network
3	Decision tree
4	Neural Network
5	Neural Network
6	Neural Network
7	Decision tree
8	Decision tree
9	Rule based
10	Rule based
11	Decision tree

Figure 4 shows the model sensitivity of all 11 models and the combination on the validation data set. Note that the combination was much better than average, and the best model overall. As mentioned before this behavior is typical in voting combinations. In Figure 5, the false alarm performance is shown. Although four of the models produced lower false alarms, the combination model had a much lower score than the average and *any* positive, non-zero weighting of sensitivity and false alarms results in the combination having the best score overall. For example, Figure 6 shows an equal weighting between sensitivity and false alarms.

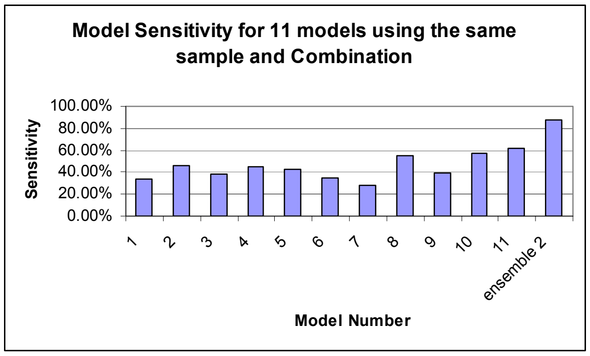

Figure 4: Model Sensitivity Comparison

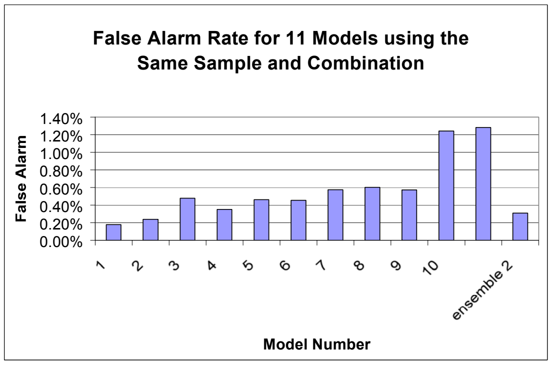

Figure 5: False Alarm Rate Comparison

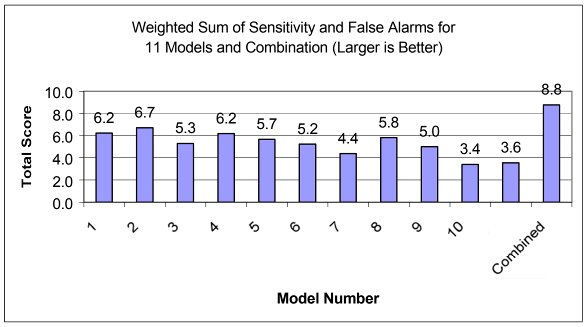

Figure 6: Total Weighted Score Comparison

Model Ensemble 3:

In building this ensemble and for the purposes of majority voting, we choose eleven best models among all the generated models (over 100). The models were chosen based on the following criteria:

1) Overall performance of model: Determined by a model's weighted and ranked score.

2) Algorithm Diversity: For example, we did not want all the final models to be neural networks.
3) Sample Split Representation (Modeler Diversity): To decrease variation and bias of results, we made sure that as many of the samples and modelers were represented as possible.

With 11 models in the final ensemble, a majority vote meant that six or more models must flag the payment as suspicious to label it suspicious. Table 2 lists the algorithms used in this model ensemble.

Tab1e 2: Algorithms Used to Create Models Included in Model Ensemble 3.

Model Number	Algorithm Type
1	Neural Network
2	Decision tree
3	Neural Network
4	Decision tree
5	Decision tree
6	Decision tree
7	Rule based
8	Neural Network
9	Neural Network
10	Neural Network
11	Rule based

Figure 7 shows the model sensitivity of all 11 models and the combination on the validation data set. Note that the combination was again much better than average, and nearly the best model overall (model 10 had slightly higher sensitivity).

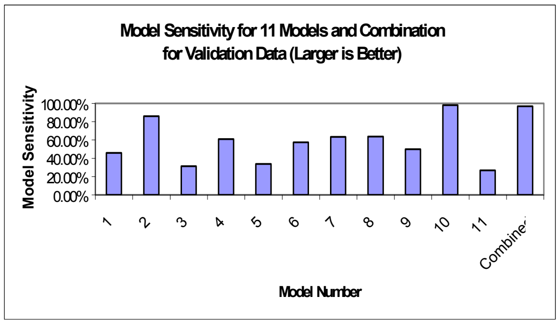

Figure 7: Model Sensitivity Comparison

Figure 8 shows the false alarm performance. As with sensitivity, the combination model had a much lower score than the average model, though model 6 had the overall lowest false alarm rate. Again it can be seen that any positive, non-zero weighting of sensitivity and false alarms results in the combination having the best score overall. Figure 9 shows an equal weighting between sensitivity and false alarms.

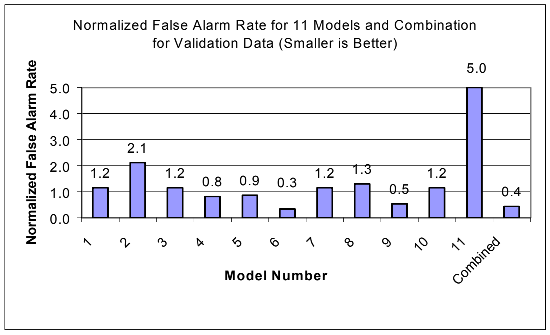

Figure 8: False Alarm Rate Comparison

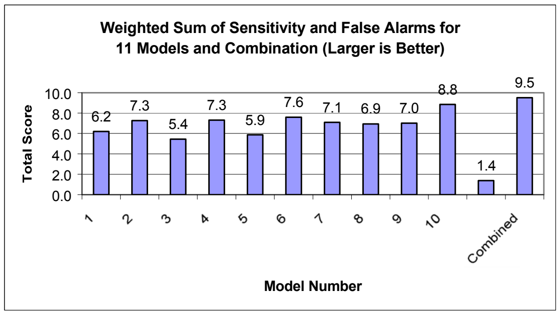

Figure 9: Total Weighted Score Comparison

Model Ensemble 4

In this strategy the models were selected among the entire generated model set according to how well they performed on the validation data set. The selection was done based on the following criteria:

(i) The two models that produced the lowest false alarm rate for classifying fraud type A.
(ii) The two models that produced the lowest false alarm rate for classifying fraud type B.
(iii) The two models that produced the lowest false alarm rate for classifying fraud type C.
(iv) The two models that produced the lowest false alarm rate for classifying fraud type D.
(v) The two models that produced the lowest overall false alarm rate.
(vi) The two models that produced the highest fraud sensitivity rate.

In this selection process, a model can be chosen for up to two criteria. This selection process resulted in ten models (two of the models were among the best for two of the criteria). In this model ensemble, the selected model only votes for those cases on which it performs best. For example, a model that was selected based on its performance for type C will only vote for classification of a payment as type C. However, the models that produced the lowest false alarm and the highest fraud sensitivity were involved in the selection of all types of fraudulent classes. Thus, at any time six models are used to decide on a class of a payment. Again as before, the majority voting was used to label a payment as suspicious. Since at any given time only six models were used to vote on the labeling of a payment, a payment's label would be improper if four or more models flag it as improper.

Table 3 lists the algorithms used for developing the selected models for the given criteria. The final ensemble included four neural networks, five decision trees, and one rule set.

Tab1e 3: Algorithms Used to Create Models Included in Model Ensemble 4.

Model number	Criterion Used	Algorithm Type
1	1 and 4	Rule induction
2	1 and 2	Decision tree
3	2	Decision tree
4	3	Neural Network
5	3	Neural Network
6	4	Decision tree
7	5	Decision tree
8	5	Neural Network
9	6	Decision tree
10	6	Neural Network

Figure 10, shows the false alarm performance of all 10 models and ensemble 4 on the validation data set. Again, the combination had a much lower score than the average model, and nearly the best overall (model 2 had slightly lower false alarm rate). In Figure 11, the model sensitivity is shown. Although the combination was much better than average, its performance was not comparable to many of the models. This could be the cause of selecting models that perform well only for a given case. However, once more it could be seen that *any* positive, non-zero weighting of sensitivity and false alarms results in the combination having the best score overall. For example, Figure 12 shows an equal weighting between sensitivity and false alarms. It can be seen that the combination had the highest weighted score.

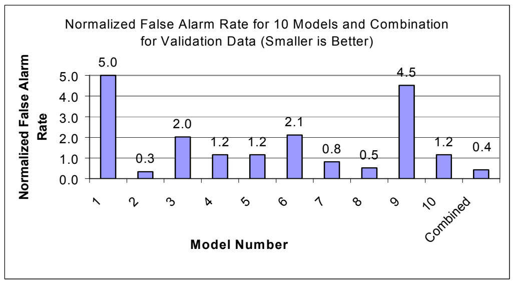

Figure 10: False Alarm Rate Comparison

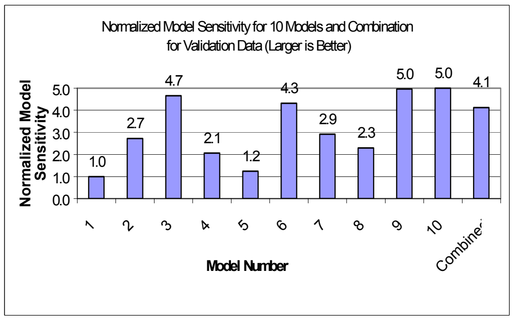

Figure 121: Model Sensitivity Comparison

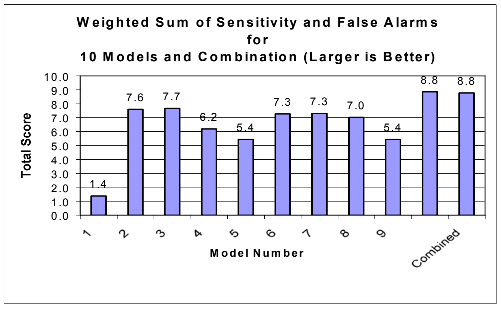

Figure 12: Total Weighted Score Comparison

5.2 Comparison of Model Ensemble Approaches

In this section, the various model selection approaches are compared with respect to fraud sensitivity and false alarm rates. Fraud sensitivity and false alarm rate form two factors for evaluation of the models. As reviewed earlier, in the fraud analysis case researched by the authors, the challenge is to minimize false alarms while not sacrificing model sensitivity. Ensembles that maximize one factor at the expense of the other are sub optimal to ensembles that are capable of improving on two factors simultaneously.

All scores for the various selection approaches were generated by running the models against the same validation data set. Figure 13 displays a comparison of the normalized false alarm scores across the model selection approaches. In Figure 13, it can be seen that Ensemble 2 (using multiple algorithms on the same sample) generates the highest false alarm rate (roughly 0.30%) from among the model ensembles. This relatively poor showing of ensemble 2 may be attributable to the inclusion of two models (10 and 11 in Figure 5) that may have been left out of the ensemble, after all the target of 11 models is an arbitrary target (NOTE: However, this did not improve the performance). In fact, when model 10 and 11 are excluded from Ensemble 2, the resultant false alarm rate and sensitivity are 0.42% and 87.6% respectively. This act of revisiting model inclusions in a model ensemble represents after-the-fact tuning of an ensemble, which, is a luxury in a mission critical analysis of fraud. Ensembles 3 and 4 represent the lowest false alarm rates (below 0.10% for both ensembles).

Figure 14 displays a comparison of model sensitivities. The higher the sensitivity the better the ensemble performance. Figure 14 displays that Ensemble 4 is not capable of maximizing its performance on two factors simultaneously, an ensemble trait required by the authors to satisfactorily handle the fraud analysis problem. Ensemble 3 has the highest sensitivity rate of the ensembles. This trait demonstrates that Ensemble 3 can maximize two factors simultaneously.

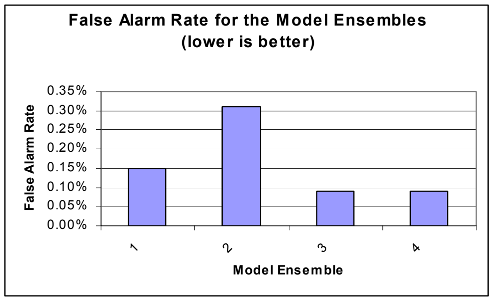

Figure 13: False Alarm Rate Comparison

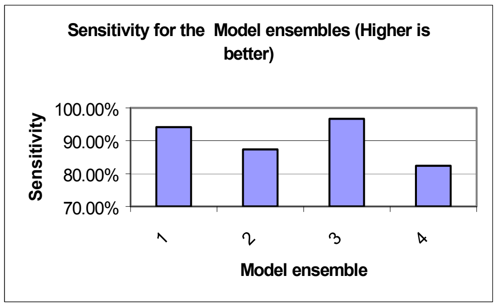

Figure 14: Model Sensitivity Comparison

Figure 15 displays the weighted combination of sensitivity and false alarm scores. Ensemble 3 and 4 have the highest weighted combined scores. This result supports the notion that increased diversity of the models improves the quality of the decisions of the committee.

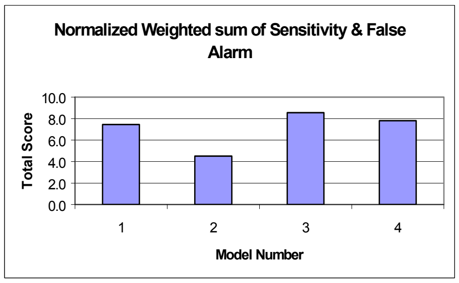

Figure 15: Total Weighted Score Comparison

Ensemble 3 can be seen as an approach that is designed to maximize the diversity of the models, both based on the type of algorithm and the training data sets. Further, Ensemble 3, with little handcrafting, generated results that maximize both factors simultaneously.

6. Discussion

There is strength in diversity. This is an adage that has found support in the literature reviewed in this paper and in the results presented in this paper. However, there are multiple sources of diversity. In the author's case, ensemble diversity has two main sources, namely sample diversity and algorithm diversity.

It has been demonstrated in prior research that two strategies for improving classification and prediction decisions are to combine multiple algorithms in a single sample or to combine multiple samples with a single algorithm type. The authors' research has extended the existing research by both combining models and samples.

This approach is demonstrated in the results listed here to provide measurable improvements over the existing combination approaches. The combination of algorithms and samples adds a further benefit in that it minimizes the risks of poor sample selection. The selection of a poor sample is a real problem that can be encountered especially given small sample sizes or low levels of representation of target classes. Data paucity is a challenge that faces fraud analysts; therefore the potential for generating a poor sample is not negligible.

The combination of algorithms and samples further avoids the necessity to continuously "handcraft" results by iteratively adding and dropping models until a result is adequate. With a fixed set of models, trained over a fixed set of samples, it is expected that the combination of models and samples is an optimal approach to improve classification decisions precisely because it is the combination approach best able to maximize the diversity of the individual models selected.

The authors note that further research needs be pursued in two areas. First the data set of trained models needs to be increased. This would allow for the collection of multiple ensembles within each type of ensemble explored in this research. Second, the authors believe that the voting mechanisms themselves could be manipulated to test the optimal decision rule for improving classification decisions.

7. References

[1] Merz, C. J. (1999) "Using Correspondence Analysis to Combine Classifiers" Machine Learning Journal.

[2] Elder, J.F., and Pregibon, D. (1995). A Statistical Perspective on Knowledge Discovery in Databases. *Advances in Knowledge Discovery and Data Mining*. U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Editors. AAAI/MIT Press.

[3] Dietterich, T. (1997). Machine-Learning Research: Four Current Directions. *AI Magazine*. 18(4): 97-136.

[4] Steinberg D. (1997), *CART Users Manual*, Salford Systems.

[5] Breiman, L. (1996), Bagging predictors. *Machine Learning* **24**: 123-140.

[6] Jain, A.K., R.P.W. Duin, and J. Mao (2000), Statistical Pattern Recognition: A Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**(1): 4-37.

[7] Breiman, L. (1996). Arcing Classifiers. *Technical Report,* July.

[8] Dietterich, T., (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning,* **40** (2) 139-158.

[9] Tumer, K. and J. Ghosh (1996), Analysis of Decision Boundaries in Linearly Combined Neural Classifiers. *Pattern Recognition*, **28**: 341-348.

[10] Perrone, M.P. and L.N. Cooper (1993), When Networks Disagree: Ensemble Methods for Hybrid Neural Networks. *Neural Networks for Speech and Image Processing*, R.J. Mammone, ed., Chapman-Hall.

[11] Jacob, R.A., M. I. Jordan, S. L. Nowlan, G.E. Hinton (1991), Adaptive mixture of local experts, Neural Computation, vol. 3, No. 1.

[12] Maclin, R. and J. W. Shavlik(1995), Combining the predictions of multiple classifiers: Using competitive learning to initialize neural networks. In Proceedings of the 14th International Joint Conference on Artificial Intelligence,.

[13] Krogh, A, and J. Vedelsby (1995), Neural network Ensembles, cross validation, and active learning. In G. Tesauro, D. Touretzky, and T. Leen, editors, Advances in Neural Information Processing Systems, volume 7, The MIT Press.

[14] Wolpert, D.H. (1992), Stacked generalization, *Neural Networks* **5**: 241-259.

[15] Optiz, D. W., and R. Maclin (1999), Popular Ensemble Methods: An Empirical Study. Journal of Artificial Intelligence Research, **11**, pp. 169-198.

[16] Bala, J., K. De Jong, J. Huang, H. Vafaie, and H. Wechsler (1995), Hybrid Learning Using Genetic Algorithms and Decision Trees for Pattern Classification. Proceedings of the International Joint Conference on Artificial Intelligence, Montreal, P.Q., Canada.

[17] Bauer, E., Kohavi R. (1998), An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Machine Learning,* 36(1/2) July/August 1999:105-139.

[18] Elder, J. F. IV, D.W. Abbott (1997), Fusing Diverse Algorithms. *29th Symposium on the Interface,* Houston, TX, May 14-17.

[19] Abbott, D.W. (1999), Combining Models to Improve Classifier Accuracy and Robustness. *2nd International Conference on Information Fusion —Fusion99*, San Jose, CA, July 6.

[20] Tresp, V., and M. Tanguchi (1995), Combining estimators using non-constant weighting functions. In G. Tesauro, D. Touretzky, and T. Leen, editors, Advances in Neural Information Processing Systems, volume 7, The MIT Press.

[21] Lee, S. S. and J.F. Elder (1997), Bundling Heterogeneous Classifiers with Advisor Perceptrons, *White Paper*, University of Idaho and Elder Research.

[22] Fawcett, T., and F. Provost (1997). Adaptive Fraud Detection. *Data Mining and Knowledge Discovery*, Kluwer Academic Publishers, **1**(3), 291-31.

[23] Jensen, D., (1995). Prospective Assessment of AI Technologies for Fraud Detection: A Case Study. *Working Papers of the AAAI-97 Workshop on Artificial Intelligence Approaches to Fraud Detection and Risk Management*, July.

[24] Jensen, D., and P.R. Cohen (2000). Multiple Comparisons in Induction Algorithms. *Machine Learning Journal*, **38**(3), 1-30.

[25] Abbott, D.W., Vafaie H., Hutchins, M. and Riney, D. (2000), Improper Payment Detection in Department of Defense Financial Transactions, *Federal Data Mining Symposium*, Washington, DC, March 28-29.