On the Use of Ensemble Models for Credit Evaluation

By Albert Fensterstock, Jim Salters and Ryan Willging

OVERVIEW

Supervised learning algorithms (neural networks, random forests, decision trees, etc.) essentially search through a hypothesis space to find a suitable hypothesis that has a high accuracy rate when applied to a certain type of problem. Specifically, many machine learning algorithms rely on some kind of search procedure: given a set of observations and a space of all possible hypotheses that might be considered (the "hypothesis space"), they look in this space for those hypotheses that best fit the data (or are optimal with respect to some other quality criterion).

Even if the hypothesis space contains hypotheses that are very well-suited for a particular problem, it may be very difficult to find a good one within a single model. Ensembles combine multiple hypotheses to form (with a little art) a better hypothesis. Therefore, an ensemble is a technique for combining diverse models in an attempt to produce one that is stronger than any individual model.

Ensemble Theory

An ensemble is itself a supervised learning algorithm, because it can be trained and then used to make predictions. The trained ensemble, therefore, represents a single hypothesis. This hypothesis, however, is not necessarily contained within the hypothesis space of the models, from which it is built, (i.e., no single model represents it).

Some Common Types of Ensembles

These are many different approaches to improve the performance of a specific model by combining it with other models. Here, we shall describe only some of the most frequently used methods.

Bootstrap Aggregating (Bagging)

Bootstrap **agg**regating (bagging), is a technique for combining models into an ensemble with each model voting with an equal weight. If there were 3 models in the ensemble each would have 33.33% of the vote. So, if we were trying to predict which of two classes, A or B, a given applicant belonged to, an agreement of at least two of the model would be needed to arrive at a decision. To achieve the required model diversity in the ensemble, bagging trains each model in the ensemble using a randomly drawn subset of the training data so that none of the models have been trained on exactly the same data sample.

Boosting

Boosting is an incremental process for building an ensemble by training each new model on the instances that the previous model misclassified. This technique looks to build a new classifier as a composition of the unique patterns contained within the models included in the ensemble. This technique has been shown to yield better accuracy than bagging, but it also has a higher risk of over-fitting the training data.

Stacking

The crucial prior belief underlying the scientific method is that one can judge among a set of models by comparing them on data that was not used to create any of them. Just like its name implies, this technique involves stacking another model on top of already created models with the purpose of determining what models perform well (which models to use in the ensemble) given the raw input data.

This prior belief can be used to choose among a set of models based on a single data set. This is done by partitioning the data set into a *held-in* data set and a *held-out* data set; training the models on the held-in data; and then choosing whichever of those trained models performs best on the held-out data.

Stacking exploits this prior belief further. It does this by using performance on the held-out data to combine the models rather than choose among them, thereby typically getting performance better than any single one of the trained models.

Other methods that will not be discussed here due to their added complexity, but can be researched on the Internet are: **Bayes Optimal Classifier, Bayesian Model Averaging, Bayesian Model Combination, and Bucket of Models.**

As you can see, there are many different approaches for combining several models into a better one, and there is no single winner: everything depends upon your data and to what use you are going to put the resulting ensemble.

Some Background on Neural Networks

Each of the two ensemble models discussed in this paper is a combination of two types of neural networks. While we could have combined neural networks with other types of machine learning models, an ensemble consisting of only neural networks of different topologies worked very well for the type of problem we were addressing (credit evaluation), and so we decided to leave well enough alone, and just focus on pure neural network ensembles. This being the case, let's spend some time discussing what a neural network is, and the two types of networks we employed. The following neural network descriptions are primarily based on information contained in Palisade Corporation's NeuralTools Manual.

Neural Networks are capable of learning complex relationships in data. By mimicking the functions of the brain they can discern patterns in data, and then extrapolate predictions when given new data.

The Structure of a Neural Net

The structure of a neural net consists of connected units referred to as **"nodes"** or **"neurons"**. Each neuron performs a portion of the computations inside the net: a neuron takes some numbers as inputs, performs a relatively simple computation on these inputs, and returns an output. The output value of a neuron is passed on as one of the inputs for another neuron, except for neurons that generate the final output values of the entire system. Neurons are arranged in **layers**. The input layer neurons receive the inputs (training data) for the computations, for example, the credit rating, years in business, and dollar value of outstanding loans of an individual credit applicant. These values are passed to the neurons in the first hidden layer, which perform computations on their inputs and pass their outputs to the next layer. This next layer could be another hidden layer, if there is one. The outputs from the neurons in the last hidden layer are passed to the neuron or neurons that generate the final outputs of the net. For example, in our application there are two outputs: a “0” (most likely to default), or a “1” (most likely to pay in full).

Types of Neural Networks

There are various types of neural networks, differing in structure, kinds of computations performed inside neurons, and training algorithms. One type is the **Multi-Layer Feedforward Network (MLFN)**. With MLFN nets, a user can specify if there should be one or two layers of hidden neurons, and how many neurons the hidden layers should contain (the software usually provides help with making appropriate selections). Another type is **Generalized Regression Neural Nets (GRNN)** and **Probabilistic Neural Nets (PNN)**; these are closely related, with the former used for numeric prediction, and the latter for category prediction/classification. With GRNN/PNN nets there is no need for the user to make decisions about the structure of a net. These nets always have two hidden layers of neurons, with one neuron per training case in the first hidden layer, and the size of the second layer determined by some facts about the training data. A detailed description of the mathematical and topological differences between the aforementioned networks is beyond the scope of this paper; however, the Internet contains a wealth of information for those interested in pursuing it.

Numeric and Category Prediction

The problems Neural Networks are used for can be divided in two general groups:

Classification Problems: Problems in which you are trying to determine what type of category an unknown item falls into. Examples include medical diagnoses and in our instance, the prediction of credit repayment ability i.e., whether or not an applicant for credit will or will not default.

An example of predicting a category would be trying to determine the winery that produced a bottle of wine from the chemical ingredients in a given bottle. The inputs include Alcohol, Ash, Magnesium, Total Phenols, Flavanoids and others and the output is the winery. The neural net takes the list of inputs and outputs and tries to find the best hypothesis that will result in the correct identification of the specific winery. The hypothesis then is a classification algorithm and the resulting neural model is a trained classification algorithm.

Numeric Problems: Situations where you need to predict a specific numeric outcome. Examples include stock price forecasting and predicting the level of sales during a future time period.

When neural nets are used to predict numeric values, they typically have just one output. This is because single-output nets are more reliable than multiple-output nets, and almost any prediction problem can be addressed using single-output nets. For example, instead of constructing a single net to predict the volume and the price for a stock on the following day, it is better to build one net for price predictions, and one for volume predictions. On the other hand, neural nets can have multiple outputs when used for classification/category prediction. For example, suppose that we want to predict whether or not a company will pay an invoice within 30, 60, 90, or after 90 days. Then the net will have four numeric outputs, and the greatest output will indicate the category selected by the net.

Training a Net

Training a net is the process of fine-tuning the parameters of the computation, i.e., the purpose of training is to make the net output approximately correct values (predictions) for given inputs. This process is guided by training data on the one hand and the training algorithm on the other. The training algorithm selects various sets of computation parameters, and evaluates each set by applying the net to each training case to determine how good the answers given by the net are. Each set of parameters is a "trial"; the training algorithm selects new sets of parameters based on the results of previous trials.

THE MODELING PROCESS

Our final ensemble model will be developed in two steps. First, we will utilize a large number of the variables which we will select from the available data, based upon our experience with this type of problem (credit evaluation) together with a certain amount of trial and error in a “Kitchensink Model”, and then based on an analysis of the impact each of the model variables has on the model’s decision (Variable Impact Analysis); a Correlation Analysis; and a Means Test, we will pare the model down, thereby eliminating low impact and/or duplicate variables. Additionally, if it is desired to keep the ongoing operational data cost at a minimum, we may eliminate variables that cost additional money to acquire (e.g., bureau data) assuming, of course, that we do not loose a significant amount of predictability in the process.

Gathering the Data

As noted above, the process of creating a credit evaluation model consists of evaluating historical data and developing a model that has the ability to make predictions about applicants that were not included in the model data, i.e., hold-outs and future applicants. The representative data shown in the following table are typical of the information available to develop a model. As far as how much data is necessary for model development, we can usually get good results with about 1,000 cases, loans in this example, where the final results of the loan are known i.e., paid in full or defaulted and if so, how much was paid prior to default.

REPRESENTATIVE MODEL DATA AND ITS SOURCE

INTERNAL	COMMERCIAL BUREAU	CONSUMER BUREAU	FINANCIAL STATEMENTS
Account Tenure	Various Bureau Predictive Indicators – Paydex; CCS; FSS; Intelliscore, etc.	Various Bureau Predictive Indicators – FICO, etc.	Leverage
Collection Effort			Working Capital
Credit Balance		Age of Newest Trade	Net Liquid Balance
Current Aging	Company History	Average Trade Balance	Net Worth
Date of Last Payment	Industry/Geography	Charge-Offs	Solvency Ratios
Historical Aging	Negative Payment	Collection Inquiries	Cash Position
Late Fees	Experiences	Credit Limit	Profit Returns
NSF Checks	Previous Bankruptcy	Current Balance	Industry Norm Information
Days Beyond Terms	Secured Financing	Delinquent Trade Lines	Total Liabilities
Payment Amounts	Size of Company	Inquiries	Gross Profit Margin
Write-Off Amounts	Suits/Liens/Judgments	Public Records	
Application Date	UCC Filings	Time On File	
Application Decision	Years in Business	Total Trades	
Funding Date	Trade Interchange Data		

Some Initial Data Tests

Let's assume that we are working with a dataset that consists of about 1,000 loans where the final results are known, randomly selected from a larger dataset. Each loan has data attributes consisting of information similar to that described in the above table. Our first step will be to do some data analysis which in this instance will consist of a Means Test and a Correlation Analysis.

The Means Test

This is an analysis that can help determine which variables to keep and which to discard. Essentially, we will compare the mean of a variable when the result was GOOD (loan paid off) to the mean value when the result was BAD (borrower defaulted on some portion of the loan). When a significant difference between the means occurs, that variable may have predictive value. To accomplish this:

1. We will split the dataset into two segments – GOODs and BADs, and then,
2. Determine the mean, standard deviation, minimum, maximum, and range of every numeric non-categorical variable for the GOOD loans, and

3. Then do the same for the BAD loans, and finally
4. Compare the mean of each BAD variable to the mean of each GOOD variable (BAD%GOOD). Any variable where the BAD%GOOD is <70% or >130% may indicate a variable that is possibly a strong predictor of BAD or GOOD.

A sample result is shown below:

REPRESENTATIVE MEANS TEST

One Variable Summary	Years Owned APPLICANT GOOD DATA	Average OD's/month APPLICANT GOOD DATA	Average Cash Balance APPLICANT GOOD DATA	Average NSFs/month APPLICANT GOOD DATA
Mean	9.592	3.934	9680.25	0.865
Std. Dev.	8.347	8.071	16009.88	2.454
Minimum	0.000	0.000	-25024.00	0.000
Maximum	54.000	71.500	147138.00	26.500
Range	54.000	71.500	172162.00	26.500
One Variable Summary	**APPLICANT BAD DATA**	**APPLICANT BAD DATA**	**APPLICANT BAD DATA**	**APPLICANT BAD DATA**
Mean	8.457	7.66	7449.36	2.556
Std. Dev.	7.736	13.71	14038.88	6.064
Minimum	0.600	0.00	-58249.00	0.000
Maximum	40.000	114.00	115520.00	43.000
Range	39.400	114.00	173769.00	43.000
BAD%GOOD	88.17%	**194.80%**	76.95%	**295.34%**

As can be seen from the above, two of the variables may be predictive (OD's and NSF's) and two may not (Years Owned and Average Cash Balance).

Likewise, for numeric categorical variables you can perform a histogram procedure for the GOOD and BAD datasets. For example, our dataset contains a NAICS code (North American Industry Classification System) which is a numeric code used by business and government to classify business establishments according to type of economic activity.

Analyzing the numeric code you can compare the relative frequency percentages (BAD%GOOD) on the histograms to see if the distributions by Bin # (e.g., Bin #3 = NAICS codes from 20 through 29.9) are significantly different. Here, as shown below, NAICS codes from 20 through 39 maybe predictive.

REPRESENTATIVE HISTOGRAM PROCEDERE

			Primary NAICS/APPLICANT GOOD DATA				Primary NAICS/BAD DATA		Rel. Freq.
Histogram	Bin Min	Bin Max	Bin Midpoint	Freq.	Rel. Freq.	Prb. Density	Rel. Freq.	Prb. Density	BAD%GOOD
Bin #1	0.00	10.00	5.00	0	0.0000	0.000	0.0000	0.000	-
Bin #2	10.00	20.00	15.00	1	0.0017	0.000	0.0000	0.000	-
Bin #3	20.00	30.00	25.00	18	0.0306	0.003	0.0717	0.007	**234.78%**
Bin #4	30.00	40.00	35.00	20	0.0340	0.003	0.0493	0.005	**145.27%**
Bin #5	40.00	50.00	45.00	192	0.3260	0.033	0.3722	0.037	114.18%
Bin #6	50.00	60.00	55.00	60	0.1019	0.010	0.0807	0.008	79.24%
Bin #7	60.00	70.00	65.00	59	0.1002	0.010	0.0717	0.007	71.63%
Bin #8	70.00	80.00	75.00	114	0.1935	0.019	0.1794	0.018	92.68%
Bin #9	80.00	90.00	85.00	125	0.2122	0.021	0.1749	0.017	82.41%
Bin #10	90.00	100.00	95.00	0	0.0000	0.000	0.0000	0.000	-

Correlation Analysis

This analysis produces a matrix in which every variable is correlated to every other variable, and can tell you which variables are correlated (say value >0.5 or <-0.5). The purpose of this is twofold:

1. If two variables are highly correlated (value >0.7 or <-0.7) and you desire to reduce the number of variables in the model you may want to see if one of them can be eliminated without any loss of predictiveness, and
2. If you are stress testing the model (sensitivity analysis) and want to try a different value for a highly correlated variable you need to be sure that you change the value of each variable it is highly correlated to. This can be accomplished in one of two ways: (1) either one variable is some function of the other and by changing one the other automatically changes, or (2) you can perform a linear regression on the two variables and use the relationship V1 = aV2 + b to determine the value of V1 when you change V.

REPRESENTATIVE CORRELATION ANALYSIS

Linear Correlation Table	Factor/month APPLICANT DATA ALL	Term APPLICANT DATA ALL	Cash APPLICANT DATA ALL	Receivables APPLICANT DATA ALL	Average NSFs/month APPLICANT DATA ALL	Average OD's/month APPLICANT DATA ALL	Years Owned APPLICANT DATA ALL	Consumer Score APPLICANT DATA ALL
Factor/month	1.000	**-0.883**	-0.124	-0.147	0.070	0.013	-0.005	-0.065
Term	**-0.883**	1.000	0.104	0.145	-0.063	-0.021	-0.022	0.048
Cash	-0.124	0.104	1.000	**0.994**	0.044	0.095	0.047	0.176
Receivables	-0.147	0.145	**0.994**	1.000	0.053	0.100	0.047	0.165
Average NSFs/month	0.070	-0.063	0.044	0.053	1.000	0.358	0.034	-0.146
Average OD's/month	0.013	-0.021	0.095	0.100	0.358	1.000	0.109	-0.091
Years Owned	-0.005	-0.022	0.047	0.047	0.034	0.109	1.000	0.121
Consumer Score	-0.065	0.048	0.176	0.165	-0.146	-0.091	0.121	1.000

As can be seen above, only Factor/month and Term (-0.883) and Cash and Receivables (0.994) are highly correlated.

BUILDING THE FIRST MODEL

The first model will be a Kitchensink Model which we will use to ascertain which of the variables are most predictive. It is our intention to develop two models; a sparse model that can be used as an initial evaluator of an applicant's credit worthiness, and thereby keep the cost of data at a minimum for new applicants, i.e., the model will not contain any data that must be purchased from an outside source, such as bureau data. And then, if the applicant is deemed potentially credit worthy additional data can be acquired and a second model applied which will aid in the final decision as to whether or not to grant a loan, and if so for how much. With respect to the model building process, we will build the second model first (the final decision maker), and then determine from that model which of the variables should be used for initial screening purposes (the sparse model).

Sample Model Output

Some of the typical output from a neural net development process is the following:

Training and Testing Results: As can be seen, during training the % Bad Predictions was 4.7692% while the % Bad Predictions when the model was applied to loans not in the training set was substantially higher at 27.6074%.

Training	
Number of Cases	650
Training Time	0:01:16
Number of Trials	280
Reason Stopped	Auto-Stopped
% Bad Predictions	4.7692%
Mean Incorrect Probability	10.8215%
Std. Deviation of Incorrect Prob.	14.9326%
Testing	
Number of Cases	163
% Bad Predictions	27.6074%
Mean Incorrect Probability	34.7740%
Std. Deviation of Incorrect Prob.	31.7563%

Source: NeuralTools Neural Net Training Output

Variable Impact Analysis: The Variable Impact Analysis measures the sensitivity of net predictions to changes in independent variables. In this analysis, every independent variable is assigned a relative value; these are percent values and add to 100%. The lower the percent value for a given variable, the less that variable affects the predictions.

Classification Matrices: These further analyze the prediction errors shown above by distributing the prediction into BAD (0) and GOOD (1) categories. In our instance the most important value, for the purpose of determining which models will be contained in

our ensemble is the prediction error rate Bad (%) for the loans that went bad, in this instance, as shown below, 70.2128%.

Classification Matrix (for training cases)			
	0	**1**	**Bad (%)**
0	147	30	16.9492%
1	1	472	0.2114%

Classification Matrix (for testing cases)			
	0	**1**	**Bad (%)**
0	14	33	70.2128%
1	12	104	10.3448%

Source: NeuralTools Neural Net Training Output

The Kitchensink Model

A substantial amount of analysis was performed (over 100 neural nets were created and evaluated) before we settled on the variables to use for the Kitchensink ensemble.

Then, to determine the Kitchensink Model, we created 3 PNN nets, and 13 MLFN nets that varied with respect to the number of nodes in the first layer.

Comparison of Individual Model Results (Kitchensink Model)

For the Kitchensink ensemble model, we used the best PNN net and the four best MLFN nets as measured by their Bad (%). These are the shaded results below.

COMPARISON OF MODEL RESULTS - KITCHENSINK MODEL

	% Bad Predictions		**Testing Clasification Matrix**	
	Training	**Testing**	**0 - Bad (%)**	**1 - Bad (%)**
PNN MODELS:				
PNN 1	**5.385%**	**22.086%**	**57.143%**	**7.018%**
PNN 2	**4.154%**	**23.125%**	**64.444%**	**6.957%**
PNN 3	**7.539%**	**27.778%**	**66.000%**	**10.714%**
MLFN MODELS:				
MLFN 1	**6.769%**	**33.333%**	**51.351%**	**28.000%**
MLFN 2	**1.539%**	**37.267%**	**65.217%**	**26.087%**
MLFN 3	**0.000%**	**33.540%**	**46.154%**	**29.508%**
MLFN 4	**0.000%**	**34.969%**	**51.282%**	**29.839%**
MLFN 5	**0.000%**	**34.356%**	**60.000%**	**27.344%**
MLFN 6	**0.000%**	**34.568%**	**47.619%**	**30.000%**
MLFN 7	**0.000%**	**29.193%**	**34.884%**	**27.119%**
MLFN 8	**0.000%**	**36.196%**	**60.000%**	**27.119%**

Next, we evaluated the ensemble model by running a prediction utility for each of the individual models. For the ensemble, each of the five individual models' output (0 or 1) was initially multiplied by a weight that was based on the model's Bad (%) such that the sum of the weights equaled 1. The weights were then modified based upon a heuristic trial and error procedure. The weighted results of each model were added together to arrive at the ensemble's prediction. Any output equal to 0.5 or less was considered BAD and an output greater than 0.5 was considered GOOD.

The final Kitchensink Model consisted of 25 variables of which 7 were categorical and 18 were numeric. Its prediction error rate was 0.49%. The fact that makes this so interesting and the technique so powerful is that the individual prediction error rates for the five models in the ensemble ranged from 22.09% to 34.97%. In other words, the prediction error rates of the individual models that make up the ensemble were from 45 to 71 times higher than the ensemble's prediction error rate. This supported our initial hypothesis that an ensemble model can produce a prediction error rate that is superior to the prediction error rate of any individual model in the ensemble.

The Sparse (Best Variables) Model

As previously noted, it was our desire to produce a sparse model that did not require any exogenous data and that could be used for initial screening. As such, we did some further analysis and utilized the Means Test, Correlation Analysis and Variable Impact Analysis as the basis for reducing the number of variables in the model, a so called "Best Variables Model".

To determine which models to use in our Best Variables ensemble we created 5 PNN nets, and 16 MLFN nets that varied with respect to the number of nodes in the first layer.

Comparison of Individual Model Results (Best Variables Model)

COMPARISON OF MODEL RESULTS - BEST VARIABLES

	% Bad Predictions		Testing Clasification Matrix	
	Training	Testing	0 - Bad (%)	1 - Bad (%)
PNN MODELS:				
PNN 1	9.8462%	33.7500%	81.6327%	12.6126%
PNN 2	7.3846%	24.5399%	76.1905%	6.6116%
PNN 3	15.5385%	28.8344%	84.3137%	3.5714%
PNN 4	9.8462%	24.5399%	68.1818%	8.4034%
PNN 5	16.1538%	29.4479%	86.0000%	4.4248%
MLFN MODELS:				
MLFN 1	3.6923%	34.9693%	66.6667%	21.7391%
MLFN 2	0.9231%	34.9693%	51.2821%	29.8387%
MLFN 3	0.1538%	35.5828%	63.8298%	24.1379%
MLFN 4	0.0000%	44.1718%	62.2222%	37.2881%
MLFN 5	0.1538%	36.4198%	51.0204%	30.0885%
MLFN 6	0.0000%	38.6503%	52.2727%	33.6134%
MLFN 7	0.0000%	35.5828%	47.3684%	29.2453%
MLFN 8	0.0000%	38.8889%	61.1111%	32.5397%
MLFN 9	0.0000%	34.9693%	43.7500%	32.8244%
MLFN 10	0.0000%	36.4198%	57.8947%	29.8387%
MLFN 11	0.0000%	41.1043%	52.0000%	36.2832%
MLFN 12	0.0000%	34.1615%	48.5714%	30.1587%
MLFN 13	0.0000%	46.0123%	56.8182%	42.0168%
MLFN 14	0.0000%	30.6250%	45.0000%	25.8333%
MLFN 15	0.0000%	36.1963%	49.0196%	30.3571%
MLFN 16	0.0000%	39.8773%	57.4468%	32.7586%

For the Best Variables ensemble, as we did for the Kitchensink ensemble, we used the best PNN net and the four best MLFN nets as measured by their Bad (%). These are the shaded results above.

Again, we evaluated the ensemble model by running a prediction utility for each of the individual models. Likewise for this model, each of the five individual models' output (0 or 1) was multiplied by a weight that was based on the Bad (%) such that the sum of the weights equaled 1, and then the weights were modified based upon a heuristic trial and error procedure. The weighted results of each model were added together to arrive at the ensemble's prediction. Any output equal to 0.5 or less was considered BAD and an output greater than 0.5 was considered GOOD.

The final model consisted of 13 variables of which 2 were categorical and 11 were numeric. Its prediction error rate was 0.74% (a 33% higher prediction error rate for about 50% less variables compared to the Kitchensink Model). The individual prediction

error rates for the five models in the ensemble ranged from 24.54% to 46.01%, i.e., the prediction error rates of the individual models that make up the ensemble were from 33 to 62 times higher than the ensemble's prediction error rate, again supporting our hypothesis that an ensemble model can produce a prediction error rate that is superior to the prediction error rate of any individual model in the ensemble.

SOME OTHER CABABILITIES OF ENSEMBLE MODELS

Using the Ensemble to Determine a Credit Line

An ensemble model is created through a batch process; however, individual applicant evaluation is performed in real-time. This allows a great deal of flexibility in the decisioning process, in particular with respect to determining a credit line. As noted, the model's output is either a GOOD or BAD decision, but all GOODs and all BADs are not created equal. Let's explore two ensemble decisions, one GOOD and one BAD.

Model Output is GOOD: In any model that evaluates a request for a loan or line of credit, the amount requested is an input variable. Because the model's decision is in real-time it is possible to replace the original amount requested with another amount and, instantaneously, get a new model evaluation. In the case of a GOOD decision, one can increment the requested amount until the model's decision changes from GOOD to BAD, thereby arriving at the maximum amount of risk (credit line) the model feels is applicable to a particular applicant.

Model Output is BAD: In this instance, we want to determine if there is any amount of risk that the model would accept. Here, you would decrement the amount requested until the model's decision changed from BAD to GOOD. This amount would represent the amount of risk that the model deems acceptable, and provides the possibility for additional profit from an otherwise unacceptable applicant.

Using the Ensemble to Determine an Early Warning System

After a model has been developed, it is important to determine which of the model's variables are sensitive to change such that had the variable(s) had a different value at the time of the applicant's evaluation, the model would have produced a different result, i.e., predicted BAD instead of GOOD. This implies that once a loan is granted it is important to continue to track the debtor to be sure that their circumstances have not changed to the extent that their current implied risk of default is not significantly greater than it was at the time the loan was granted.

To accomplish this task, one needs to decide which of the model's variables, with respect to a given debtor, need to be reevaluated on a periodic basis. In our instance, it was determined that four specific variables were the main variables of interest, and that a material change in any one of them, or in a combination of them might yield a significant change in the inherent risk of default.

Next, a stratified random sample was taken from the entire dataset that was used to develop the models using Score Class as the stratification boundary. Our sample was taken only from the GOOD population and three Score Classes were used for stratification, specifically: Class 0 - was very unlikely to default (none of the models in the ensemble initially predicted default); Class 1 - was more likely to default (one of the models in the ensemble initially predicted default); and Class 2 - was much more likely to default (two of the models in the ensemble initially predicted default). Or, in other words, initially, a majority of the ensemble, three, four or five of the individual models, forecasted that the applicant would not default. For statistical purposes, our sample was large enough to result in a very small standard error of estimate: about ± 5% at the 95% confidence level. Stated another way, if our test indicated that a given change in a variable or set of variables produced a possible default rate of 32.0%, in a specific score class, then we could say that we are 95% confident that the true population default rate will be between 30.4% and 33.6%.

Incrementing and Decrementing the Variables of Interest

Initially, each variable of interest was incremented or decremented by itself with the other variables of interest held constant. Next, pairs and then triplets of the variables were incremented or decremented with the other variable(s) of interest held constant. And, finally, all of the variables were incremented or decremented simultaneously.

Test Results

With respect to the Best Variables Model we found the following: Certain changes in the variables of interest increased the possibility of default: in Class 2 by up to 60.0%; in Class 1 by up to 40.0%; and in Class 0 by up to 32.3%. Thus, the probability of default can change significantly, over time, based on small changes in certain variables.

Operationally, therefore, to minimize the possible loss inherent in a defaulted loan, it is necessary that a debtor's circumstances be reevaluated on an ongoing basis to be sure that the inherent risk in an account has not increased to the point that were they evaluated currently they would be denied funding.

CONCLUSION

As described in this paper, Ensemble Models represent a very powerful method for determining credit risk. And, given the fact that they can be built on a PC based platform, at a reasonable cost, makes them a very attractive alternative to other methods of credit evaluation, and in particular to judgment-based models. As shown, in this paper, this class of models can aid in providing an easy to implement solution to the associated problems of initial credit evaluation, credit line determination and ongoing debt tracking.

Albert Fensterstock is managing director of Albert Fensterstock Associates. He has more than 40 years experience in financial and operations management and analysis, and currently specializes in the development and implementation of decision support systems based on various types of predictive analytics including statistical analysis, neural networks and simulation technology. His solutions are frequently used for improving risk analysis capability and collection department efficiency. He founded CreditRiskMonitor.com Inc., the first Internet-based financial information analysis and news service created specifically for the corporate credit function. Credit Today has named him one of the 50 most influential people in credit. He can be reached at 516-313-1020 or via e-mail at albie_f@yahoo.com.

Jim Salters is CEO of The Business Backer, an industry leader connecting small businesses with the capital and expertise they need to survive and thrive. Throughout his career, Jim's mission-driven leadership and innovative strategies have repeatedly generated 10x growth and profitability for early stage companies. Under Jim's direction, The Business Backer has secured $130 million in funding to more than 4,000 small businesses across the United States. Jim graduated with honors from Princeton University. He received Ernst & Young's Entrepreneur of the Year award and was named one of the Cincinnati Business Courier's Forty under 40. He can be reached at 513-746-2000 or via email at jsalters@businessbacker.com.

Ryan Willging is senior operations analyst of The Business Backer. Ryan identifies process improvement opportunities and implements technology solutions to improve operational efficiency. His successes allow The Business Backer to provide a lower cost of capital to a wider range of small businesses. He most recently built technology enabling a 94% increase in underwriter capacity, while reducing the time to decision by 53%. Ryan holds a Finance degree from Miami University. He can be reached at 513-746-2000 or via e-mail at rwillging@businessbacker.com

References

Dietterich, Thomas G. (2000) *Ensemble Methods in Machine Learning*
Schwenk, Holger and Benggio, Yoshua (2000) Neural Computation
Kuncheva, L. and Whitaker, C. (2003) *Measures of Diversity in Classifier Ensembles*
Mitchell, Tom M. (1997) *Machine Learning*
Palisade Corporation (2012) *NeuralTools*
Specht, Donald F. (1989) *Probabilistic Neural Networks*
Sollich, P. and Krogh, A. (1996) *Learning With ensembles: How overfitting Can be Useful*.
Zenko, Bernard (2004) *Is Combining Classifiers Better Than Selecting the Best One?*