Laura Maria BADEA, PhD Candidate
Department of Economic Cybernetics
E-mail: laura.maria.badea@gmail.com

USING NEURAL NETWORKS IN ESTIMATING DEFAULT PROBABILITY – A CASE STUDY ON RETAIL LENDING

Abstract. *Artificial neural networks have become very popular in solving financial problems and especially in developing credit scoring models. This article provides a detailed example of how neural networks can be used in building a model that predicts the probability of default for retail customers. It also shows that neural networks manage to bring more efficiency and robustness by developing intelligent systems. Using Multilayer Perceptron technique, the results are very good in terms of detecting individuals with high risk of default. Moreover, hybrid models such as neural networks using logistic activation functions are very efficient and lead to even better results.*

Key words: *artificial neural networks, probability of default, discriminant analysis, logistic regression, Multilayer Perceptron, retail lending.*

JEL Classification: C45, C53, C63

1. Introduction

Neural networks techniques have applicability in areas such as computer science (artificial intelligence and data mining), engineering (signal processing, adaptive control), mathematics (linear algebra, numerical analysis, and statistics), biology, and medicine. In economics neural networks have become very useful in risk analysis and prediction problems. The most important problems that can be solved using neural networks are the following[1]: monitoring of credit risk associated with mortgage evaluation, economic forecasting, financial default, and bankruptcy prediction, as well as identification of certain patterns in share prices.

Retail segment is still one of today's most profitable sources of income for banks and financial institutions. Still, as some studies have proven, such as Jacobson et al. (2005), and Dietsch and Petey (2004), loans granted to these clients provide higher default risk than those granted to companies. Thus, traditional methods of estimating this kind of risk have been completed or even replaced by advanced learning methods which have improved the predictability process.

According to IRBA (Internal Rating Based Approach) of Basel II, banks can develop their own internal rating systems (provided that certain minimum requirements are met), by deciding on the method of estimating the default

[1] Medsker, L., Turban, E. and R. Trippi (1996), "Neural Network Fundamentals for Financial Analysts", *Neural Networks in Finance and Investing*

probability (PD). Hayden and Porath (2006) propose the following methods, each having its specific advantages and limitations: linear regression, discriminant analysis, logit and probit models, hazard models, decision trees and neural networks.

2. Method description

Neural networks are non-parametric methods based on the existence of nodes. These nodes generate answers only if the information generated by other nodes is high enough to trigger an effect. Neural networks are much more flexible than previously mentioned methods and this is due to the fact that they don't make any assumptions about the data distribution.

Generally, neural networks are defined by the following elements:

- Processing units called "neurons" or "cells";
- An activation condition for each unit;
- Connections between units defined by the weights w_{ij}, revealing the effect that unit *i* has on unit *j*;
- A propagation rule defining the Y_i entry point against an outside stimulus;
- An activation function f_i, determining the new activation level based on input $Y_i(t)$ and current activation $a_i(t)$ (adjustment);
- An external input (deviation, bias) θ_j for each unit;
- A learning rule;
- A framework for the system to operate in, transmitting input signals and errors.

Neural Networks include three types of units, *input units*, receiving information from outside the network (they usually coincide with initial variables), *hidden units*, whose inputs and outputs remain inside the network, and *output units* which transmit the results outside the network. During the process, units can be adjusted either simultaneously or separately depending on the probability of adjustment a unit has at time-point *t*.

According to Rojas (1996) there are two methods of "learning":

- *Supervised* (also known as "learning with a teacher"); this method requires a priori knowledge of the output, and measurement of predictions' deviations. Further, supervised learning can be of two types: *consolidation ("reinforcement")* and *error correction*. The former method generates weight adjustments according to Boolean information that specifies whether or not the outputs are good but does not provide any expected value, whereas the latter takes into account the magnitude of errors, and the weights are adjusted in order to generate a response as close as possible to the realized values.
- *Unsupervised*; this method does not have at start any knowledge of the real output values, and in most situations, it also doesn't have information about the possible response classes.

Haykin (1994) and Bishop (1995) have described several neural network topologies within their studies:

- *Feed-forward*; this type of neural network is used in pattern recognition. The signal is induced in the input layer, towards the hidden layers, and finally to the output layer. Single Layer Perceptron (SLP) networks, Multilayer Perceptron (MLP) networks, and networks based on radial functions (RBF) are included in this category;
- *Feedback networks*, also known as recurrent. The difference between feedback networks and feed-forward networks is that the former have at least one response loop. More precisely, the outputs of some neurons from the hidden layers may be input for other neurons or even for themselves. This class contains: competitive networks, Kohonen- networks, the Hopfield models and ART (Adaptive Resonance Theory) models, where the units are connected in every way possible.

Of the above mentioned networks, the most common is the Multilayer Perceptron (MLP) that was first proposed by Rumelhart and McClelland (1986). This type of neural network implies the processing of inputs from the first layer by multiplying them with weights. These are afterwards summed in the first hidden layer and go through an activation function following the same course up to the output layer. The MLP consists of feed-forward units organized in layers. The total number of nodes and layers determines the complexity of the functions. However, the number of hidden layers and neurons within the neural networks is not predefined as there is no formula or rule to suggest the optimal values. G.K. Jha (2009) mentions several methods for determining the number of neurons in hidden layers:

- Number of neurons in the hidden layers should be between the number of input variables and the number of output variables;
- The number of neurons should equal two thirds of the number of input variables, plus the output layer size;
- The number of hidden neurons should be less than twice the number of input variables.

The above mentioned rules are only a starting point in selecting the number of hidden neurons. Ramlall (2010) points out the importance of selecting the number of hidden neurons empirically. If the network contains too few units then the data will not be well modeled, whereas if too many neurons are included there is a high risk of overfitting. Most often the optimal hidden neurons rate is determined by changing the number and running tests, then choosing the model which generated the lowest error.

The neurons from the hidden layers and the output layer are adjusting units, each having its activation function. The most common activation functions used in neural networks are: *linear function* ($f(x) = x$), *sigmoid function*: ($f(x) = \frac{1}{1+e^{-x}}$), and *hyperbolic tangent function* ($\mathrm{f(x)} = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$).

Input for neuron *j* is described by the following equation: $Y_j = \sum_i w_{ij} x_i + \theta_j$, where x_i represents the output of the preceding layer, w_{ij} represents the weight of the connection between *i* and *j* units, and θ_j is the bias of unit *j*.

Activation value (output) for neuron j is (exemplified for sigmoid function): $a_j = f(Y_j) = \frac{1}{1+e^{-Y_j}}$.

By differentiating, we obtain the following expression: $f'(Y_j) = \frac{e^{-Y_j}}{(1+e^{-Y_j})^2} = f(Y_j)\left(1 - f(Y_j)\right) = a_j(1 - a_j)$.

After the number of units and type of activation function are established, the weights need to be set up to a beginning level. "Learning" algorithms are involved in this process. The most frequently used algorithm in MLP is *backpropagation*. This algorithm involves adjusting the weights and thresholds in order to minimize the prediction error. Newly proposed algorithms are faster on some issues. These include *conjugate gradient descent*, used when the activation function is not linear but sigmoid, and *Levenberg-Marquardt algorithm*, which was proposed especially for minimizing sum of squared errors. Still, backpropagation algorithm is preferred for being easier to understand, and also because it involves less memory resources and achieves an acceptable level of error very quickly.

Backpropagation algorithm progresses iteratively, executing several cycles. Each time, estimated outputs are compared with the real ones and errors are recalculated for subsequent weight adjustments. Initial run values are random, and therefore errors will be high, but the following iterations' errors will decrease to an acceptable level. This is one of the possible exit conditions. The iterative loop can also be executed until a minimal value is reached or until the desired number of iterations is executed. The stop condition for the algorithm can be one of these three exit criteria.

Error function used by "backpropagation" algorithm is the sum of squared errors recorded in the network outputs for all the observations in the training set: $E_p = \sum_{j=1}^{N} (d_{pj} - a_{pj})^2$, where d_{pj} represents the realized value (actual) for observation p, a_{pj} represents the calculated output value of node j for the same observation p, and N is the total number of output values.

The total error is calculated using the following formula, with T representing the total number of observation from the training set: $E = \sum_{p=1}^{T} \frac{1}{2} E_p = \sum_{p=1}^{T} \frac{1}{2} \sum_{j=1}^{N} (d_{pj} - a_{pj})^2$.

The method used to minimize the total error uses the descending gradient, and each change in weights is proportional with the result of deriving the error by its corresponding weight: $\Delta_p w_{ij} = -\eta \frac{dE_p}{dw_{ij}}$, where η represents the step size, also known as learning rate. Derivation of the error by the weight leads to: $\frac{dE_k}{dw_{ij}} = -\delta_{pj} a_{pj}$, where for an output unit and hidden unit respectively we have

the following relations: $\delta_{pj} = f'(Y_{pj})(d_{pj} - a_{pj})$, and $\delta_{pj} = f'(Y_{pj}) \sum_k \delta_{pk} w_{jk}$ (where δ_{ks} are the errors in the layer following the layer of unit j).

3. Data set and variables

The data set used for the analysis contains the anonymous information of 10.360 private individuals. They have applied for a loan in January 2007 - August 2008 at one of the financial institutions with a considerable share in the banking sector from Romania. Selected variables include socio-demographic information, and data about the customers' financial situation. As Thanh and Kleimeier (2006) state, financial information for retail segment is very limited as compared to the corporate sector. In the dataset used for the current analysis, qualitative characteristics are available in greater number, whereas financial information is limited to variables such as: monthly income, loan amount, interest rate and term. Furthermore, Kočenda and Vojtek (2009) show how socio-demographic features provide more stable information during financial instability than standard financial data.

Because the purpose of this paper is to develop a PD estimation model for retail applicants, the information used will cover only those aspects that are known a priori to the lending process. Therefore, data regarding the payment behavior of those customers will only be used in assessing the status of "good" (non-defaulter) or "bad" (defaulter).

Under Basel II[2], default occurs when the bank considers that it is possible that the debtor will not fulfill all the obligations without having to resort to proceedings for the liquidation of collateral (if any), or when "the debtor has delayed to pay due amounts by more than 90 days". In some cases, banks may use a threshold representing the amount of money below which, over 90 days past dues will not be considered defaults.

In this paper the convention is that the 90 days past dues marks the split between "default" and "non-default" customers. All those clients that defaulted at least once during the time interval September 2008 - August 2009 were classified as "bad". All other observations were considered "good"[3]. The splitting process

[2] Basel II Commettee (2004), „International Convergence of Capital Measurement and Capital Standards"

[3] The database does not include information on the applications that were not accepted. According to Thomas, Edelman and Crook (2002) there are five techniques for estimating the behavior of applicants rejected in the past. All these methods are based on assumptions regarding the performance of rejected applicants, but as shown by Banasik, Crook and Thomas (2003), classification accuracy of applicants is not significantly different if the analysis does not include those people.

resulted into 8,932 "good" customers and 1,428 "bad" customers (13.78% defaulters).

Available variables are binary, nominal (divided into several classes) and numeric. Initially there were 23 available characteristics. Variables related to the loan are: *product type* (includes three categories: personal needs loans, goods loans and mortgage loans), *loan amount (RON)*[4], *interest rate (%), co-debtor (Y/N)*[5], and *loan term.* Information regarding the applicants' particularities is captured in the following variables: *net income, age, sex, highest level of education, marital status, number of children, landline phone number (Y/N), mobile phone number (Y/N), work phone number (Y/N), residential status, city*[6]*, region, industry* (the operating industry of the applicant's employer), *profession, type of company where the applicant works (legal form of company), seniority at current address, seniority at current work place, and seniority at previous work place.*

In order to have a data set as clean as possible, some operations were performed: outlier removals, and missing values replacements using different techniques (frequencies per different types of applicants or mean values for numerical variables).

4. Model development

The main purpose of this section is to develop PD estimation models applicable to private individuals. The used methods are: discriminant analysis, logistic regression and neural networks. For neural networks two distinct models were developed, depending on the activation functions chosen in the hidden layers. The model assessment is based on the best detection rate in terms of customer classification.

Data partitioning is a very important step in developing a model. Therefore, the database was divided into three distinct sets, each with different functionality:

- *Training*; which forms the basis for learning;
- *Validation*; which is used for evaluation, adjustment and acceptance of the model learned in training stage in order to minimize the error on the validation set;
- *Test* data set; that will act as out-of-sample information.

Partitioning was done randomly, choosing a seed (starting number for the sequence generated random numbers) equal to 15,000. From the available data (10,360 records), 70% was assigned to the training set, 20% to the validation set, and 10% to the test set.

4.1. Discriminant analysis

Discriminant analysis is one of the oldest methods used in credit scoring. This method derives from linear probabilistic models and was first introduced by

[4] Of all the 10,360 loans, 3,634 were granted in EUR, and 34 in USD. These have been converted into RON using the NBR exchange rate corresponding to the loan approval date.
[5] This indicates whether or not a codebtor exists for the loan.
[6] This was split into four categories: Bucharest, county residence, city and rural place.

Fisher (1936). Table1 presents the value of the parameters corresponding to the two functions that define the profiles of "good" and "bad" classes. Variable selection was done using "stepwise forward" technique[7] which eliminates redundant information.

Table 1 – Discriminant functions for both customer classes

	P:0 (non-default)	P:1 (default)
Product type	3.3174	-0.0003
Loan amount (RON)	-0.00003	-0.00002
Interest	-0.1779	-0.0474
Term	0.0583	0.0911
Net Income	-0.0003	-0.0003
Codebtor	10.2604	12.0120
Age	2.7935	2.7597
Sex	6.6164	6.3620
Marital status	0.6001	0.6766
Number of children	5.7376	5.9733
Work phone no.	5.7915	5.9278
Residential status	2.5554	2.4616
City	2.4137	2.5908
Industry	0.3754	0.3450
Profession	0.9811	0.8359
Type of company	2.9333	3.1345
Intercept	-44.9835	-49.8807

[7] Thanh and Kleimeier (2006) used "Stepwise forward" selection on credit scoring model applied on data from Vietnam. A "forward" selection means that each independent variable will be evaluated in relation to the growth that brings to R-squared. The variable that provides the greatest increase will be included in the model if it meets the statistical entry criterion. This refers to the level of significance of R-square increased by adding a new variable. When no other feature meets this criterion anymore, the selection process stops. In addition to "forward" technique, "stepwise forward" means that with each new step, existing variables in the model are evaluated for removal. To the extent that the statistical condition is met, the feature whose exclusion would affect the least the value of R-squared will be removed. The fact that a feature could be significant at some point, and thereafter may not be significant anymore might seem strange. However, it is possible that by adding a new predictor, some variables become redundant. In order to apply this selection method, nominal variables were transformed into numbers using a simple coding. At first there was done an analysis on the whole data list (using all 23 available variables). SLSTAY and SLENTRY variables (both taking into account the threshold of 0.15) define the cutoffs that allow features to enter and remain in the model. This statistical analysis was performed in SAS version 9.1. Using this method, 16 variables resulted as significant for the model. This selection method is very important before performing discriminant analysis or logistic regression because only relevant variables are included in the models (see Appendix).

Table2 represents the classification matrix at a 0.5 PD level. The total detection rate is 88.30%. However, the default detection is quite low, only 41.29%. This is explained by the share of "bad" customers which is much lower than the "good" share.

Table 2 – Classification matrix for discriminant analysis

		Training				Test			
		Classification rate	Predicted		Total	Classification rate	Predicted		Total
			Good	Bad			Good	Bad	
Real	Good	**95.83%**	7703	335	8038	**96.09%**	859	35	894
	Bad	**41.29%**	755	531	1286	**47.18%**	75	67	142
Total		**88.30%**			9324	**89.38%**			1036

4.2. Logistic regression

Logistic regression is another popular method used in estimating the PD. This is mainly because it generates results that are easy to interpret.

Both, discriminant analysis and logistic regression methods are sensitive to correlations between variables, so it is necessary to eliminate highly correlated features before making any prediction. That is why, before applying logistic regression, variables were chosen using stepwise forward selection.

A simple logistic regression was applied, and the overall detection rate at a 0.5 PD was better, compared with discriminant analysis. However, "bad" class detection is worse in this case on the "out-of-sample" dataset (see Table3).

Table 3 – Classification matrix for logistic regression

		Training				Test			
		Classification rate	Predicted		Total	Classification rate	Predicted		Total
			Good	Bad			Good	Bad	
Real	Good	**97.15%**	7795	229	8024	**97.47%**	885	23	908
	Bad	**43.46%**	735	565	1300	**42.19%**	74	54	128
Total		**89.66%**			9324	**90.64%**			1036

4.3. Neural networks

Neural network models are based on interconnections between the input layer, hidden layers, and output layer. This makes a prior selection of variables not stringently required because neural networks are not as sensitive as discriminant analysis or logistic regression to possible correlations between variables. On the contrary, as shown by Baesens (2003), the predictive power of neural networks can

become higher particularly due these correlations present in the characteristics of individual applicants.

Very popular in PD estimations, neural networks proved to be more robust and efficient than traditional statistical methods, such as discriminant analysis or logistic regression. Still, using neural networks in predicting the PD for applicants is not an easy process because extensive testing is needed to determine the best architecture and tune the model.

For developing the model using neural networks the MLP architecture was chosen. The first layer of the neural network includes all 23 variables. Combination functions[8] are linear in the hidden layer as well as in the output layer. Activation functions in the hidden layers are hyperbolic tangent and logistic. The learning algorithm used in MLP is backprobpagation which uses the error between the actual output (real) and calculated output to adjust the weights.

Since there is no "recipe" suggesting the optimal number of layers, and hidden neurons in a neural network, it is necessary to make several trials, compare the errors and decide on that scenario that generated the lowest error on the validation set.

Initially a model with a 3 neuron hidden layer was tested. The error on the validation set was 0.2799. Subsequently, the number of neurons in the hidden layer was increased to 5, aiming to see how the error evolves. This dropped to 0.2777 which suggests that increasing the number of neurons in the hidden layer increases the accuracy of the model. For the next two versions (with 10 and 17 neurons in hidden layer) the error on the validation set continued to decrease to 0.2733 and 0.2747 respectively. Afterwards, the number of hidden layers was increased to two and a model was run with 3 hidden neurons and 17 neurons respectively in the hidden layers. Error rose to 0.2814. The next step was increasing the number of hidden layers to 3 and run a model with 3, 10, and 3 neurons. The validation error decreased to 0.2743. The increase in the number of neurons in the second hidden layer to 15, led to an increase in the error level to 0.2789.

Therefore, the two best results are obtained using one hidden layer with 10 neurons, followed by the model with three hidden layers made of 3, 10 and 3 neurons. Komorad (2002) makes the selection of the model by the classification error on the validation set. Sorting ascending by the above mentioned indicator (see Table4), the best results are obtained on the 3-10-3 NN architecture model, where the lowest misclassification error was reached (0.09700).

[8] The functions describing how outputs from multiple nodes are combined in order to form the input value for another unit

Table 4 – Error level vs. neural network architectures

Name	Target E	Root ASE	Valid:Root ASE	Test:Root ASE	Misclassification Rate	Valid:Misclassification	Test:Misclassification
NN 3 10 3	1	0.243796556	0.274390691	0.2715252368	0.0791505792	0.097007722	0.0936293436
NN 10	1	0.2526407277	0.2733945439	0.2620660837	0.0879757308	0.097972973	0.0965250965
NN 3 15 3	1	0.2356726793	0.2789636819	0.2725348961	0.0740485383	0.0994208494	0.0955598456
NN 3	1	0.252137612	0.2799742563	0.2720338568	0.0857694429	0.1008687259	0.0984555985
NN 17	1	0.2523744236	0.2747227929	0.2667227605	0.0852178709	0.1018339768	0.1003861004
NN 3 17	1	0.2659126656	0.2814067031	0.2722951522	0.0968008825	0.1027992278	0.0936293436
NN 5	1	0.26759737	0.2777465811	0.2670706454	0.0965250965	0.1032818533	0.0859073359

The estimation process starts from the input layer towards the output and implies changing each node's weight by reverse propagation algorithm (backpropagation) in order to reduce the output error. This stops when the average error on the set of validation is minimal. After executing a neural network model with 3, 10, and 3 neurons in the three hidden layers results, at 0.5 PD level, reveal better detection rates (see Table 5) than previously used methods (disciminant analysis and logistic regression).

Table 5 – Classification matrix for neural network with hyperbolic tangent activation function

		Training				**Test**			
		Classification rate	**Predicted**		**Total**	**Classification rate**	**Predicted**		**Total**
			Good	**Bad**			**Good**	**Bad**	
Real	**Good**	**97.63%**	7834	190	8024	**98.13%**	891	17	908
	Bad	**49.54%**	656	644	1300	**45.31%**	70	58	128
Total		**90.93%**			9324	**91.60%**			1036

Comparing the results of the three techniques[9] used above one can conclude that the best detection rate is obtained in case of neural networks. Not only the overall classification rate has the best value (90.93% for neural networks, 88.30% for discriminant analysis, and 89.66% for logistic regression), but also for defaulters category, neural networks provide the highest detection rate, when compared with other methods (49.54% vs. 41.29% for discriminant analysis, and 43.46% for logistic regression).

When using a logistic type activation function in the hidden layers, the detection rate is visibly improved. „Bad" applicants' detection, at a 0.5 PD is 59%, whereas the overall detection rate reaches 92.02% (see Table6).

[9] Discriminant analysis, logistic regression, and neural networks with hyperbolic tangent activation function

Table 6 – Classification matrix for neural networks with logistic activation function

		Training				Test			
		Classification rate	Predicted		Total	Classification rate	Predicted		Total
			Good	Bad			Good	Bad	
Real	Good	**97.37%**	7813	211	8024	**98.24%**	892	16	908
	Bad	**59.00%**	533	767	1300	**56.25%**	56	72	128
Total		**92.02%**			9324	**93.05%**			1036

5. Performance and model calibrations

To evaluate the model proposed in the previous section, more analysis and graphs will be performed. Some of these relate to "Receiver Operator Characteristic" curve, Lorenz curve, Gini coefficient, "Area under the ROC Curve" (AUC), and Kolmogorov-Smirnov statistic.

5.1. Receiver Operator Characteristic "Curve (ROC)

As defined by Webb (2002), ROC curve describes the evolution of a classifier without considering the class distribution or cost of error. This allows to the ROC curve to disconnect the classification performance of these two factors. ROC curve is a two-dimensional graph of *sensitivity*[10] on the X axis against *1 – specificity*[11] on Y axis. The closer the curve gets to (0.1) point, the better the model discriminates.

Figure1 outlines the differences between three of the methods used to estimate PD (logistic regression, neural networks with hyperbolic tangent activation function and neural networks with logistic activation function). The best results, on the training dataset, are achieved by neural networks that use logistic functions for activation in the hidden layers. The lowest detection is recorded in case of logistic regression model.

[10] Detection rate of "good" customers for this paper's case
[11] Detection rate of "bad" customers for this paper's case

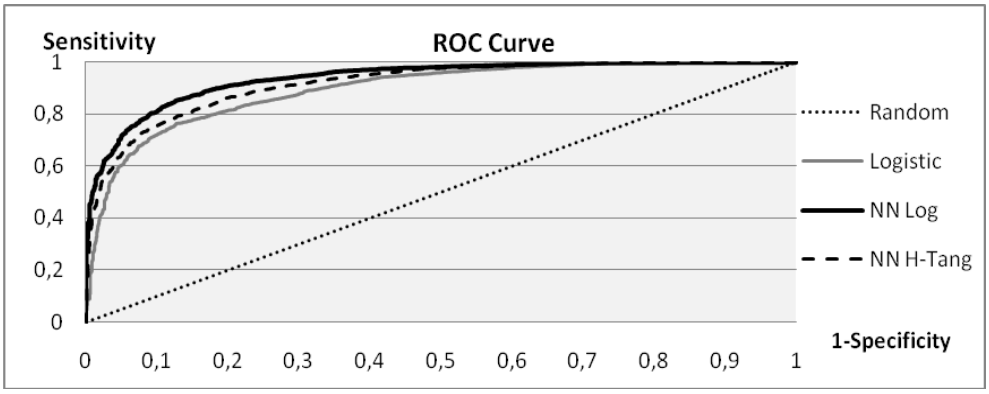

Figure 1 – ROC Curve for training dataset

On test dataset, the ROC curves (see Figure 2) also prove that the best separation is achieved by neural networks with logistic activation functions. They have a more fragmented contour because test data has limited number of observations (1036 records), and also because of its "out-of-sample" character.

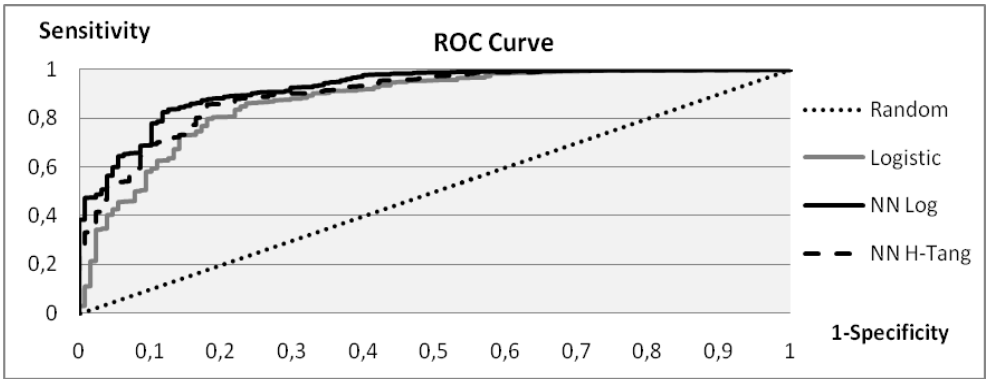

Figure 2 - ROC Curve for test dataset

5.2. Lorenz Curve

Like the ROC curve, Lorenz curve accounts "bad" clients on deciles. Lorenz curve captures how well the selected model approaches the ideal situation where after about 14%[12] of all customer data, all „bad" customers are detected. However, neural network model with logistic activation function estimate well. After 30% of the data 91% of the defaulters have been detected.

For neural networks using hyperbolic tangent activation function, the first 30% of the data results in detecting 87% of all „bad" customers; with logistic regression, the first 39% of the data results in detecting 83% of all „bad" customers being detected. Using this analysis also leads to the conclusion that the best results are attained by the neural networks technique using logistic activation function.

[12] "Bad" customers share.

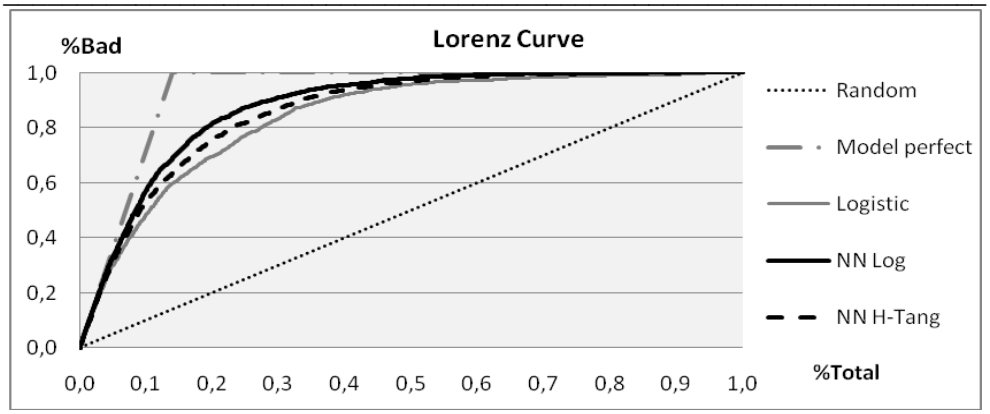

Figure 3 - Lorenz Curve

5.3. Gini coefficient and AUC

"Accuracy Ratio" (AR) value, derived from Lorenz curve, and Gini coefficient are two important ratios in assessing a model's performance.

Gini coefficient measures the performance of two models, in this case, the developed one and the random model. Gini coefficient measurement uses the area between the Lorenz curve, the diagonal line (that does not have any prediction information) and 0.5. The bigger this value (closer to 1), the more accurate the predictions made by the model are.

The formula for calculating Gini coefficient is: $G = 1 - \sum_{i=0}^{N}(Y_{i-1} - Y_i)(X_{i-1} + X_i)$, where *Yi* represents the cumulative probability that observations are "good" (non-defaulters), and *Xi* represents the difference between 1 and the cumulative probability that the observations are "bad" (defaulters). The Area Under the ROC Curve is calculated by $AUC = \frac{Gini+1}{2}$.

Table7 contains the values for Gini coefficients, and AUC for three prediction methods, for both training and test datasets. The best performance is achieved when using neural networks with logistic activation function.

Table 7 - Gini coefficient and „Area under the ROC curve" for training and test datasets

	Training			Test		
	NN Log	**NN H-Tang**	**Logistic**	**NN Log**	**NN H-Tang**	**Logistic**
c (Gini)	0.877	0.834	0.789	0.847	0.799	0.747
AUC	0.939	0.917	0.895	0.924	0.899	0.873

5.4. Kolmogorov-Smirnov distance

Kolmogorov-Smirnov distance represents a measure for the separation between „good" and „bad" groups. It is used to identify the maximum distance between cumulative distributions of customers who have defaulted and those who have not, based on scores. In Figure4, there are visible both the distribution of maximum scores as well as the point separation (0.72637). Maximum separation point is achieved for 0.85 cutoff value, where the cutoff value is calculated as the difference between 1 and PD value. The most effective model is again neural networks with logistic activation function in the hidden layers (where *P(B)* - Distribution of „bad" customers and *P(G)* - Distribution of „good" customers).

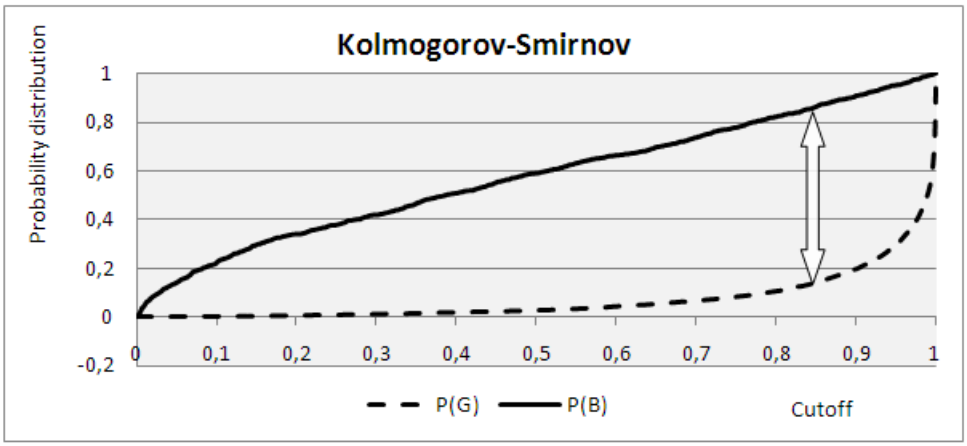

Figure 4 - Kolmogorov-Smirnov distance

Table 8 - Classification matrix for neural networks with logistic activation function, at a 0.85 cutoff

		Training				Test			
		Classification rate	Predicted		Total	Classification rate	Predicted		Total
			Good	Bad			Good	Bad	
Real	Good	**81.75%**	6560	1464	8024	**83.15%**	755	153	908
	Bad	**84.46%**	202	1098	1300	**82.81%**	22	106	128
Total		**82.13%**			9324	**83.11%**			1036

Theoretically, the cutoff value of 0.85 generates the maximum separation between the two customer sets. Table8 shows the detection rates at a 0.85 cutoff value. Detection rate for „bad" applicants is, at this point, 84.46%, and total detection is 82.13%. Although overall correct classification is less than the one recorded at a 0.5 cutoff value (92.02%), default detection increased from 59% to 82.13%. This gain in „bad" customer detection rate brings more advantages as

compared with the drop in „good" customer detection rate, from 97.37% to 81.75%.

In their paper, Tsai, Lin et al. (2009) use classification rates („good" and „bad") to detect the point of maximum separation. Figure5 shows the point where the detection rates on the two groups are equal. This confirms the choice for the cutoff level considered optimal for neural network model using logistic activation function in the hidden layers.

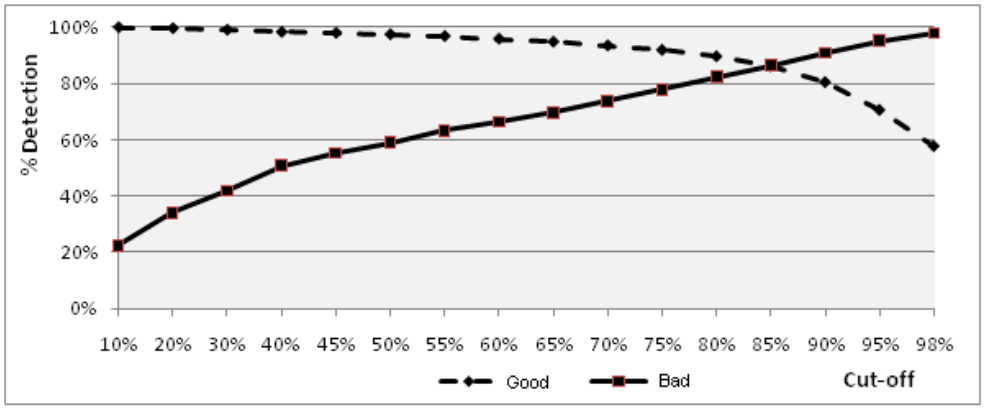

Figure 5 - Class detection at different cutoff values

Model calibration is of high importance for financial institutions and banks. Given that the lending decision is based mainly on the cutoff value, all risks and assumptions should be analyzed. Risks appear both when setting a too low or a too high cutoff value. For a too low cutoff value, risky applicants would receive a loan approval. On the other hand, when a too high cutoff value is chosen, the opportunity cost raises, as valid applicants are rejected, and thus potential interest payments are lost. However, in severe economic conditions, a higher cutoff value is preferable, as the defaulting risk gets higher for many applicants. In the end, the choice for the proper cutoff value belongs to bank's decision makers.

6. Conclusions

In recent years neural networks have become increasingly popular in estimating default probability. This article tries to show that these techniques are more efficient and more robust when compared with other conventional methods such as logistic regression or discriminant analysis. Moreover, using hybrid neural networks that have logistic activation functions in the hidden layers, the results improve significantly.

Neural networks, however, involve certain costs such as difficulties in determining the architecture and the need for many tests to determine the optimal number of layers and hidden neurons. So the model configuring decision is rather

based on empirical studies, since there is no theoretical support suggesting a global "recipe" for neural networks architecture. Due to their "black box" character results can also be difficult to interpret. However, the advantages and benefits offered by these modeling techniques worth assuming these costs because estimating the probability of default is very important for banks and financial institutions.

REFERENCES

[1] **Baesens, B. (2003),** ***Developing Intelligent Systems for Credit Scoring Using Machine Learning Techniques; Ph.D***. thesis, K.U.Leuven;
[2] **Banasik, J. & Crook, J. & Thomas, L. (2003),** ***Sample Selection Bias in Credit Scoring Models;*** *Journal of the Operational Research Society*, Vol. 54 No.8, 822-832;
[3] **Basel II Committee (2004),** ***„International Convergence of Capital Measurement and Capital Standards"***, Bank for International Settlements, basel, Switzerland, 92-93
[4] **Bishop, C.M. (1995),** ***Neural Networks for Pattern Recognition;*** Clarendon Press, Oxford, 77-191;
[5] **Dietsch, M. & Petey, J. (2004),** ***Should SME Exposures Be Treated as Retail or Corporate Exposures? A Comparative Analysis of Default Probabilities and Asset Correlations in French and German SMEs;*** Journal of Banking and Finance, Volume 28, Issue 4, 773-788;
[6] **Fisher, R.A. (1936),** ***The Use of Multiple Measurements in Taxonomic Problems; Annals*** of Eugenics, 7, 179-188;
[7] **Hayden, E. & Porath, D. (2006),** ***Statistical Methods to Develop Rating Models; In "Statistical Methods to Develop Rating Models – The Basel II Risk Parameters, Estimation, Validation and Stress Testing"***, Springer, Berlin, 1-11;
[8] **Haykin, S. (1994),** ***Neural Networks***, Macmillan College Publishing Company, Inc, New York;
[9] **Jacobson, T. & Linde, J. & Roszbach, K. (2005),** ***Exploring Interactions between Real Activity and the Financial Stance.*** Journal of Financial Stability, Elsevier, Volume 1, Issue 3, 308-341;
[10] **Jha, G.K. (2009),** ***Artificial Neural Networks;*** Indian Agricultural Research Institute, PUSA, New Delhi, 7-8;
[11] **Kočenda, E & Vojtek, M. (2009),** ***Default Predictors and Credit Scoring Models for Retail Banking;*** CESifo Working Paper Series 2862, CESifo Group Munich, Prague;
[12] **Medsker, L. & Turban, E. & Trippi R.R. (1996),** ***Neural Network Fundamentals for Financial Analysts*** **.**Neural Networks in Finance and Investing, Revised Edition, Irwin Professional Publishing, Chicago, IL., 3-43;
[13] **Ramlall, I. (2010)**, ***Artificial Intelligence: Neural networks Simplified.*** International Research Journal of Finance and Economics, Issue 39, 105-120;
[14] **Rojas, R. (1996),** ***Neural Networks, A Systematic Introduction;*** Springer Publ., Berlin, 78-79;

[15] **Rumelhart, D.E. & McClelland, J.L. & the PDP research group. (1986),** ***Parallel Distributed Processing: Explorations in the Microstructure of Cognition;*** Volume I. Cambridge, MIT Press, 216-271;
[16] **Thanh, D. & Thi, H. & Kleimeier, S. (2006),** ***A Credit Scoring Model for Vietnam's Retail Banking Market*** **.** International Review of Financial Analysis, Elsevier, Volume 16, Issue 5, 471-495;
[17] **Thomas, L. & Edelman, D.B. and J.N. Crook (2002),** ***Credit Scoring and Its Applications;*** SIAM Monograms on Mathematical Modeling and Computation, 141-144;
[18] **Tsai, M.C. & Lin, S.P. & Cheng, C.C. & Lin, Y.P. (2009),** ***The Consumer Loan Default Predicting Model – An Application of DEA–DA and Neural Network ;*** Journal Expert Systems with Applications: An International Journal, Volume 36, issue 9, NY, USA, 11682-11690;
[19] **Web, A.R. (2002),** ***Statistical Pattern Recognition*** **,** 2nd Edition, John Wiley & Sons, Malvern, UK, 15-17 and 260-266.

Appendix

Appendix 1 – Summary of Stepise forward Selection

Summary of Stepwise Selection								
Step	Variable Entered	Variable Removed	Number Vars In	Partial R-Square	Model R-Square	C(p)	F Value	Pr > F
1	Product type		1	0.0791	0.0791	2614.29	890.15	<.0001
2	Term		2	0.0725	0.1517	1594.69	885.51	<.0001
3	Interest		3	0.0591	0.2108	764.052	775.71	<.0001
4	Codebtor		4	0.0264	0.2372	393.572	359.01	<.0001
5	Profession		5	0.011	0.2482	241.088	151.05	<.0001
6	Type of employer		6	0.0083	0.2565	126.234	115.52	<.0001
7	City		7	0.0032	0.2597	82.982	44.93	<.0001
8	Loan amount (RON)		8	0.0018	0.2615	59.6046	25.25	<.0001
9	Industry		9	0.0012	0.2627	44.4599	17.09	<.0001
10	Number of children		10	0.0011	0.2638	31.1243	15.31	<.0001

11	**Sex**		11	0.0008	0.2646	22.4707	10.64	0.0011
12	**Marital status**		12	0.0004	0.265	18.499	5.97	0.0146
13	**Work phone**		13	0.0003	0.2653	16.6675	3.83	0.0504
14	**Income**		14	0.0002	0.2655	15.9244	2.74	0.0977
15	**Residential status**		15	0.0002	0.2656	15.3073	2.62	0.1057
16	**Age**		16	0.0002	0.2658	14.8287	2.48	0.1154