PREDICTING STUDENT ACADEMIC PERFORMANCE IN BLENDED LEARNING USING ARTIFICIAL NEURAL NETWORKS

Nick Z. Zacharis

Department of Computer Systems Engineering, Technological Educational Institute of Piraeus, Athens, Greece

ABSTRACT

Along with the spreading of online education, the importance of active support of students involved in online learning processes has grown. The application of artificial intelligence in education allows instructors to analyze data extracted from university servers, identify patterns of student behavior and develop interventions for struggling students. This study used student data stored in a Moodle server and predicted student success in course, based on four learning activities - communication via emails, collaborative content creation with wiki, content interaction measured by files viewed and self-evaluation through online quizzes. Next, a model based on the Multi-Layer Perceptron Neural Network was trained to predict student performance on a blended learning course environment. The model predicted the performance of students with correct classification rate, CCR, of 98.3%.

KEYWORDS

Artificial Neural Networks, Blended Learning, Student Achievement, Learning Analytics, Moodle Data,

1. INTRODUCTION

Nowadays, the application of artificial intelligence in education has grown considerably worldwide, exploiting the fact that student related information have become available through the daily use of Learning Management Systems (LMSs). By examining logs and reports from such systems, instructors can evaluate the online activity of their students and see how many times they access the course, what is the number of pages or modules accessed and which assignments they completed. Online participation in chats and forums, and rates of students' progress can also be monitored and provide actionable insights to improve teaching. In order to improve the functionality provided by those unstructured, low-level information collected in server logs, researchers utilize many data mining methods to analyze raw data and extract high level information about students' behavior and learning strategies [1], [2], [3].

Analytics and data mining methodologies allow teachers to search through large datasets to discover patterns that reflect the students' behavior and learning [4]. An important research topic in Educational Data Mining is the modeling of student’s online activity in order to predict future academic performance [5]. Although students' performance holds an important role in the learning process, it itself is a complex phenomenon affected by many factors like the teaching environment and personal study habits. Different studies have used different indicators/variables to build models capable to predict academic performance. Research indicates that some variables are more efficient predictors than others, in terms of student success in LMS supported courses [1], [2].

One of the factors that influence the student's engagement is classroom environment. Research findings indicate that, teaching practices that encourage active learning strategies, collaborative group work and feedback, especially in blended learning environments, have increased potential to engage students in meaningful interactions with course materials, peers and instructors. In this study, four predictors/variables, drawn from the daily students' activity in a Moodle based blended learning course, were used as inputs in order to build an Artificial Neural Network (ANN) model capable to predict student success in terms of course grade. An ANN is a three-layer network, that uses a supervised learning algorithm to classify input data (e.g., number of messages, number of wiki postings, number of files viewed and number of quiz efforts) into specific output categories (e.g., *failure* or *success*). The proportion of students that the PNN made a correct diagnosis of success or failure was 98.3%.

2. RELATED WORK

Neural networks have been used by analysts to predict student learning outcomes. Lykourentzou et al. [6] used grades from a multiple-choice test as inputs to neural networks that predict students' final achievement. The results showed that neural network method was more accurate and efficient, compared to linear regression, in classifying students into two groups based on their learning outcom. In order to predict academic performance of business school graduates, Paliwal & Kumar [7] used data from a business school and compared neural networks and standard statistical techniques for prediction and classification problems. The findings demonstrated the superiority of ANN over the other algorithms and revealed that undergraduate academic results and test score were the most important variables in measuring the academic performance. Jayne, Lanitis & Christodoulou [8] predicted the grades of different courses, based on the grade pupils received either on mathematics or Physics course. Three neural network-based methods were investigated: multilayer perceptron, radial basis functions and mixture density networks. According to the results, multilayer perceptron and radial basis functions outperform mixture density networks.

Kanakana and Olanrewaju [9] used the average point scores of grade 12 students as inputs to a multilayer perception neural network and predicted first year college student achievement with high accuracy. In a literature review on data mining techniques used for predicting student performance, Shahiri, Husain & Rashid [10] found that cumulative grade point average (CGPA) is the most influence attribute because it determines future educational and career mobility. According to their findings, neural network has the highest prediction accuracy by (98%) followed by decision tree by (91%). Support vector machine and k-nearest neighbor had the same accuracy (83%), while naive Bayes gave lower prediction accuracy (76%).

3. ARTIFICIAL NEURAL NETWORKS

Artificial Neural Networks are computing algorithms that can solve complex problems imitating animal brain processes in a simplified manner [11]. Perceptron-type neural networks consist of artificial neurons or nodes, which are information processing units arranged in layers and interconnected by synaptic weights (connections). Neurons can filter and transmit information in a supervised fashion in order to built a predictive model that classifies data stored in memory. The typical ANN model is a three-layered network of interconnected nodes: the input layer, the hidden layer, and the output layer. The nodes between input and output layers can form one or more hidden layers. Every neuron in one layer has a link to every other neuron in the next layer, but neurons belonging to the same layer have no connections between them (Figure 1). The input layer receives information from the outside world, the hidden layer perform the information processing and the output layer produces the class label or predicts continuous values. The values from the input layer entering a hidden node are multiplied by weights, a set of predetermined numbers, and the products are then added to produce a single number. This number is passed as an argument to a nonlinear mathematical function, the activation function, which returns a number between 0 and 1.

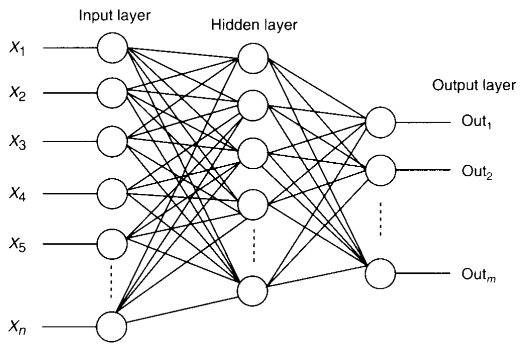


Figure 1. Neural network architecture.

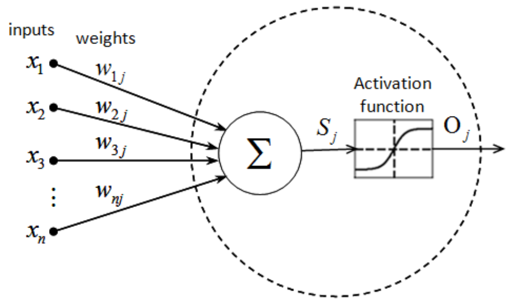


Figure 2. Neural network active node.

In Fig.2, the net sum of the weighted inputs entering a node *j* and the output activation function that converts a neuron's weighted input to its output activation (the most commonly used is the sigmoid function), are given by the equations $S_j = \sum_{i=1}^{n} x_i w_{ij}$ and $O_j = \frac{1}{1+e^{S_j}}$ respectively.

The neuron, and therefore the ANN, has two modes of operation, the training mode and the using mode. During the training phase, a data set with actual inputs and outputs will be used as examples to teach the system how to predict outputs. This supervised learning begins with random weights and, by using gradient descent search algorithms like Backpropagation, adjusts the weights to be applied to the task at hand. The difference between target output values and obtained values is used in the error function to drive learning [12]. The error function depends on the weights, which need to be modified in order to minimize the error. For a given training set $\{(x_1,t_1),(x_2,t_2),\cdots,(x_k,t_k)\}$ consisting of *k* ordered pairs of *n* inputs and *m* dimensional vectors (*n*-inputs, *m*-outputs), which are called the input and output patterns, the error for the output of each neuron can be defined by the equation: $E_j = \frac{1}{2}(O_j - t_j)^2$, while the error function of the network that must be minimized is given by: $E_j = \frac{1}{2}\sum_{j=1}^{k}(O_j - t_j)^2$, where O_j is the output produced when the input pattern x_j from the training set enters the network, and t_j is the target value [13]. During the training mode, each weight is changed adding to its previous value the quantity

$$\Delta w_{ij} = -\gamma \frac{\partial E}{\partial w_{ij}}$$

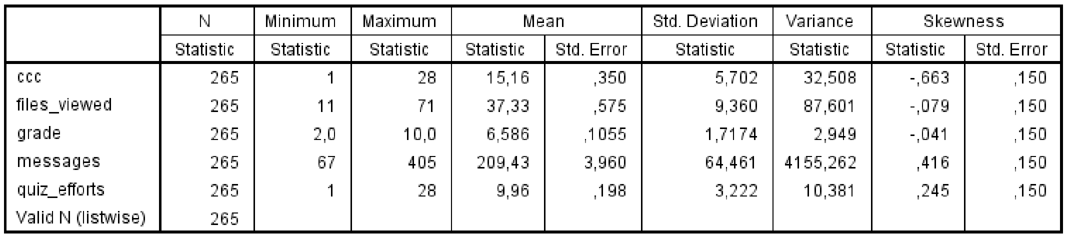
where γ is a constant that gives the learning rate. The higher the learning rate, the faster the convergent will be, but the searching path may trapped around the optimal solution and convergence become impossible. Once a set of good weights have been found, the neural network model can take another dataset with unknown output values and predict automatic the corresponding outputs.

4. METHODOLOGY

4.1. Data

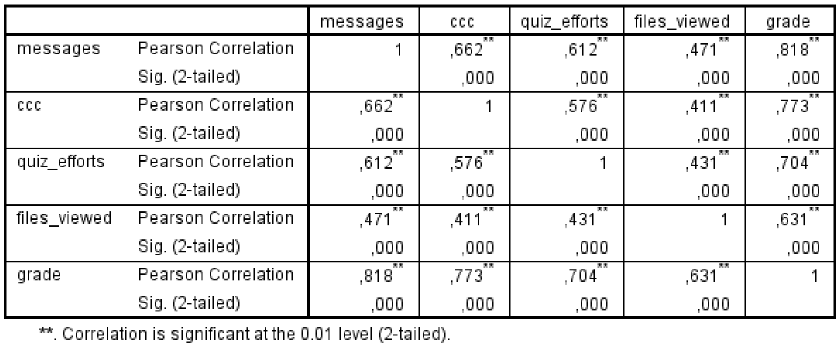
The data for this study was collected during the 2015 - 2016 academic year at a large technological university. 265 students majoring in computer systems engineering and mechanical engineering were randomly selected from two blended learning courses on programming languages and object oriented programming. Both courses were hosted in Moodle, a LMS that keeps records of various user activities. All lecture notes, assignments, quizzes and other collaborative communication tools were available through course web page. The Moodle logging API keeps track of what materials students have accessed and when, and stores all log information in the tables of its relational data base. (mdl_log). A set of queries in SQL format, was used to extract information from the log file. Table 1 gives the descriptive statistics of the used data. Variable correlations are presented in Table 2.

Table 1. Descriptive Statistics.



	N	Minimum	Maximum	Mean		Std. Deviation	Variance	Skewness	
	Statistic	Statistic	Statistic	Statistic	Std. Error	Statistic	Statistic	Statistic	Std. Error
ccc	265	1	28	15,16	,350	5,702	32,508	-,663	,150
files_viewed	265	11	71	37,33	,575	9,360	87,601	-,079	,150
grade	265	2,0	10,0	6,586	,1055	1,7174	2,949	-,041	,150
messages	265	67	405	209,43	3,960	64,461	4155,262	,416	,150
quiz_efforts	265	1	28	9,96	,198	3,222	10,381	,245	,150
Valid N (listwise)	265								

Table 2. Correlations.



		messages	ccc	quiz_efforts	files_viewed	grade
messages	Pearson Correlation	1	,662**	,612**	,471**	,818**
	Sig. (2-tailed)		,000	,000	,000	,000
ccc	Pearson Correlation	,662**	1	,576**	,411**	,773**
	Sig. (2-tailed)	,000		,000	,000	,000
quiz_efforts	Pearson Correlation	,612**	,576**	1	,431**	,704**
	Sig. (2-tailed)	,000	,000		,000	,000
files_viewed	Pearson Correlation	,471**	,411**	,431**	1	,631**
	Sig. (2-tailed)	,000	,000	,000		,000
grade	Pearson Correlation	,818**	,773**	,704**	,631**	1
	Sig. (2-tailed)	,000	,000	,000	,000	

**. Correlation is significant at the 0.01 level (2-tailed).

4.2. Variables used to build the ANN

Independent variables

1) Number of messages viewed and/or posted by the student.
2) Number of content creation contributions of each student.
3) Number of files viewed by the student.
4) Number of quiz efforts taken by the student.

Dependent variable

Course outcome, which is a dichotomous variable with two values: 0 and 1. The value 0 indicates failure to pass the course, while the value 1 shows that student was able to pass the course. Course outcome variable results directly from the degree each student obtained at the end of the course. If the degree was equal or greater than six, course outcome variable takes the value 1 (success), otherwise the value 0 (failure).

4.3. ANN design and setup

The Multilayer Perceptron (MLP) Module of IBM SPSS Statistics 21 was used to build the neural network model and test its accuracy. MLP neural networks, trained with a back-propagation learning algorithm which uses the gradient descent to update the weights towards minimizing the error function.

The data were randomly assigned to training (60%), testing (20%), and holdout (20%) subsets. The training dataset is used to find the weights and build the model. The testing data is used to find errors and prevent overtraining during the training mode. The holdout data is used to validate the model [15]. Before training, all covariates were normalized using the formula (x−min)/(max−min), which returns values between 0 and 1, and data only from the training set. The basic MLP configurations are summarized below:

```
MLP Success  (MLEVEL=N) WITH Messages Ccc Quiz_Efforts Files_Viewed /RESCALE COVARIATE=NORMALIZED
/PARTITION TRAINING=6 TESTING=2 HOLDOUT=2 /ARCHITECTURE  AUTOMATIC=NO HIDDENLAYERS=1
(NUMUNITS=AUTO) HIDDEN FUNCTION=TANH OUTPUTFUNCTION=SOFTMAX /CRITERIA TRAINING=BATCH
OPTIMIZATION= SCALED CONJUGATE LAMBDAINITIAL=0.0000005 SIGMAINITIAL=0.00005 INTERVALCENTER=0
INTERVALOFFSET=0.5 MEMSIZE=1000 /PRINT CPS NETWORKINFO SUMMARY CLASSIFICATION SOLUTION
IMPORTANCE /PLOT NETWORK ROC GAIN LIFT PREDIC TED/OUTFILE MODEL='C:\Users\Nikzach\Hubic\Artificial
Prediction - Nikos\6-8-2016\Synaptic_Weights.Xml' /STOPPINGRULES ERRORS TEPS= 10 (DATA=AUTO) TRAININGTIMER=
ON (MAXTIME=15) MAXEPOCHS =AUTO ERRORCHANGE=1.0E-4 ERRORRATIO= 0.0010/MISSING USERMISSING=
EXCLUDE .
```

For the hidden layer, hyperbolic tangent (or tanh) was used as activation function. The activation of the *j*th output neuron is $O_j = \tanh(S_j) = \frac{e^{S_j} - e^{-S_j}}{e^{S_j} + e^{-S_j}}$, takes real numbers as arguments and returns real values between -1 and 1. For the output layer, softmax function was used as activation function. The activation of the *j*th output neuron is $O_j = \sigma(S_j) = \frac{e^{S_j}}{\sum_{k=1}^{m} e^{S_k}}$, where *m* is the number of output neurons. The softmax function takes real numbers as arguments and maps them into real values between 0 and 1, that have sum equal to 1. Since the sum of the output activations is 1, the softmax layer can be thought of as a probability distribution and the O_j value can be interpreted as the network's estimated probability (or pseudo-probability) of the classification of input x.

Gradient descent optimization algorithm can achieve the best solution with two different implementations: batch mode and online (or incremental) mode. The batch algorithm, which uses all records in the training dataset to update the synaptic weights [15], was used for the training mode since it converges to the same minimum as online learning algorithm, but is slightly more efficient in terms of number of computations.

The scaled conjugate gradient method, the most widely used iterative method for solving linear equations, was used for the batch training of the ANN. It has been found that when this algorithm is used to train a multi-layer perceptron network, is more computationally efficiently than the gradient-descent and conjugate gradient methods in solving the optimization problems encountered [14]. Before each iteration, all training dataset is fetched and the synaptic weights are updated. Therefore, the algorithm finds the global minimum on the error surface by minimizing the total error made in the previous iteration [15], [16].

Four parameters - initial lambda, initial sigma and interval center and interval offset - determine the way the scaled conjugate gradient algorithm builds the model. The parameter lambda controls if the Hessian matrix is negative definite [16]. The parameter sigma controls the size of weight change that affects the estimation of Hessian through the first order derivatives of error function [13]. The parameters interval center a_0 and a force the simulated annealing algorithm to generates random weights between $a_0 - a$ and $a_0 + a$ that updated repeatedly minimizing the error function [15]. Initial lambda was set to 0.0000005, initial sigma to 0.00005. Interval center was defined as 0 and interval offset was set to ±0.5.

Stopping Rules

1) Maximum steps without a decrease in error: 10
2) Maximum training time: 15 min
3) Maximum training epochs: auto
4) Minimum relative change in training error: 1.0e-4
5) Minimum relative change in training error ratio: 1.0e-3

When softmax activation function is applied to the output layer, SPSS uses the cross-entropy error function instead of the squared error function that uses for the other activation functions. The cross entropy error function for one training example is given by the formula

$$E = -\sum_{j=1}^{m} t_j \ln O_j$$

where m is the number of output nodes/classes, t_j is the target value of output node j and O_j is the actual output value of output node j. The backpropagation algorithm, in each iteration (or epoch), calculates the gradient of the training error as $\frac{\partial E}{\partial w_{hj}} = (O_j - t_j)x_h$ for the weights linking nodes in the hidden and nodes in the output layer of the network, and

$\frac{\partial E}{\partial w_{hj}} = \sum_{j=1}^{m} \left(O_j - t_j\right) x_h w_{hj} \left(1 - x_h\right) x_i$ for the weights of links of the hidden and the input layer.

For each training example, every weight w_{ih} is updated by adding to it $\Delta w_{ih} = -\gamma \frac{\partial E}{\partial w_{ih}}$, thus taking the new value: $\Delta w_{ih} \leftarrow w_{ih} + \Delta w_{ih}$.

5. Results of Multilayer Perceptron Neural Network

The aim of this study was to examine whether a MLP neural network can help instructors to correctly predict students' course outcome (*failure* or *success*), by analyzing data obtained from their online activity and engagement with course modules. Table 3 gives information about the datasets used to build the ANN model.

Table 3. Case Processing Summary.

		N	Percent
Sample	Training	153	57,7%
	Testing	53	20,0%
	Holdout	59	22,3%
Valid		265	100,0%
Excluded		0	
Total		265	

The Table 4, shows the number of neurons in every layer and the four independent variables *(# messages*, *# ccc*, *# quiz efforts*, *# files viewed*). Automatic architecture selection chose 3 nodes for the hidden layer, while the output layer had 2 nodes to code the depended variable *course outcome*. For the hidden layer the activation function was the hyperbolic tangent, while for the output layer used the softmax function. Cross entropy was used as error function because of the use of softmax function.

Table 4. Network Information.

Input Layer	Covariates	1	messages
		2	ccc
		3	quiz_efforts
		4	files_viewed
	Number of Units[a]		4
	Rescaling Method for Covariates		Normalized
Hidden Layer(s)	Number of Hidden Layers		1
	Number of Units in Hidden Layer 1[a]		3
	Activation Function		Hyperbolic tangent
Output Layer	Dependent Variables	1	course outcome
	Number of Units		2
	Activation Function		Softmax
	Error Function		Cross-entropy

a. Excluding the bias unit

The network diagram that SPSS used to predict course outcome (success=0, success=1) from 4 online student activities, is shown in Figure 3. The diagram shows the 4 input nodes, the 3 hidden nodes and the two output nodes representing failure and success categories.

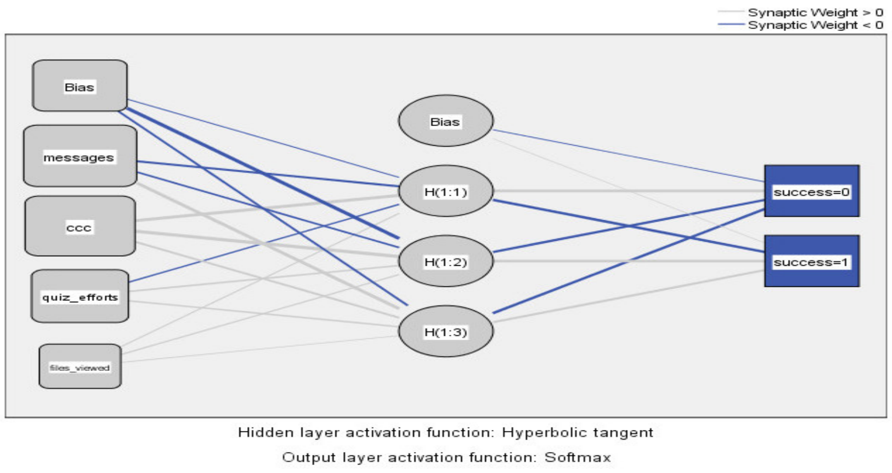


Figure 3. Network Diagram

The model summary, shown in Table 5, provides information related to the results of training (and testing) and holdout sample. Cross entropy error is given for both training and testing sample since is the error function that network minimizes during the training phase. The small value (=17.973) of this error indicates the power of the model to predict course outcome. The cross entropy error is less for the holdout sample compared with the training and testing data set, meaning that the network model has not been overfitted to the training data and has learn to generalize from trend. The result justifies the role of testing sample which is to prevent overtraining. According to the table, the percentage of incorrect predictions based on training and testing sample respectively are 2% and 1.9%, while the rate of incorrect predictions in holdout data set drops to 1.7%. The learning procedure was performed until 10 consecutive steps with no decrease in error function was attained from the testing sample.

Table 5. Model Summary.

Training	Cross Entropy Error	17,973
	Percent Incorrect Predictions	2,0%
	Stopping Rule Used	10 consecutive step (s) with no decrease in error[a]
	Training Time	0:00:00,02
Testing	Cross Entropy Error	3,081
	Percent Incorrect Predictions	1,9%
Holdout	Percent Incorrect Predictions	1,7%

Dependent Variable: success

a. Error computations are based on the testing sample.

Table 6 displays the synaptic weights between which have been calculated using only the data of the training dataset [15].

Table 6. Parameter Estimates.

Predictor		Predicted				
		Hidden Layer 1			Output Layer	
		H(1:1)	H(1:2)	H(1:3)	[success=0]	[success=1]
Input Layer	(Bias)	-,521	-5,055	-2,105		
	messages	-2,899	-1,818	3,838		
	ccc	4,938	8,924	2,611		
	quiz_efforts	-,773	,949	,848		
	files_viewed	,578	,639	,349		
Hidden Layer 1	(Bias)				-,362	,158
	H(1:1)				3,779	-3,367
	H(1:2)				-3,210	3,481
	H(1:3)				-3,230	3,142

Table 7 displays a classification table (i.e. confusion matrix) for categorical dependent variable course outcome, by partition and overall. For each case, the predicted outcome is defined as *success* if the predicted probability is greater than 0.5. As can be seen, the MLP network correctly classified 150 students, out of 153, in the training sample and 52 out of 53 in testing sample. Overall 98% of the training cases were correctly classified. In the holdout sample, the sensitivity (or recall or true positive rate), given by the formula $\frac{TP}{TP+FN}$ 100%, was found to be 100%, the specificity (or true negative rate), $\frac{TN}{TN+FP}$ 100%, was 96.6% and the accuracy of the model, $\frac{TN+TP}{TN+FP+TP+FN}$ 100%, was 98.3%. The MLP network model misclassified only 1 student (1.7%) as false positive. This extremely small Type II error rate is of great importance, since the possibility to predict success for a student who is going to fail should be minimum.

Table 7. Confusion matrix.

Sample	Observed	Predicted		
		failure	success	Percent Correct
Training	failure	51	2	96,2%
	success	1	99	99,0%
	Overall Percent	34,0%	66,0%	98,0%
Testing	failure	20	0	100,0%
	success	1	32	97,0%
	Overall Percent	39,6%	60,4%	98,1%
Holdout	failure	28	1	96,6%
	success	0	30	100,0%
	Overall Percent	47,5%	52,5%	98,3%

Dependent Variable: course outcome

Figure 4 shows box plots of predicted pseudo-probabilities. For the dependent variable *course outcome*, the chart displays box plots that classify the predicted pseudo-probabilities based on the whole dataset [15]. For each box plot, the values above 0.5 show correct predictions.

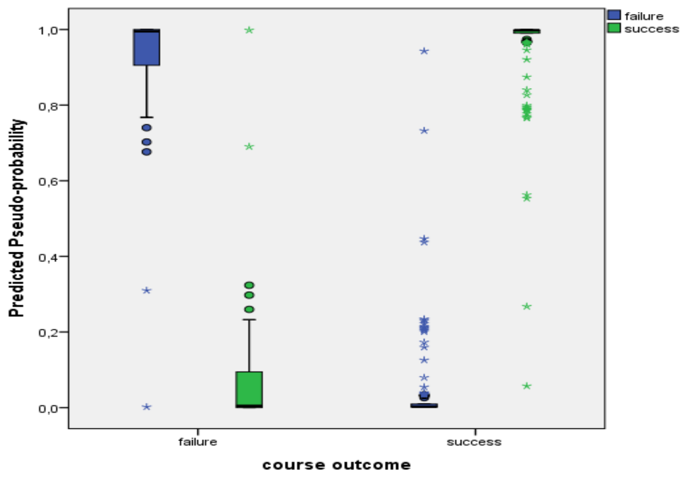


Figure 4. Predicted-by-observed chart.

The first, from the left, bloxplot shows the predicted probability of the observed failed students to be in the *failure* category. The second boxplot shows, the probability for a student to be classified in *failure* category although he really was in *success* category. The third boxplot shows, for outcomes that have observed category *success*, the predicted probability of category *failure*. The right boxplot shows, the probability a student who really succeeded to be classified in the *success* category.

The ROC curve is a diagram of sensitivity versus specificity that shows the classification performance for all possible cutoffs. Figure 5 gives the sensitivity and specificity (= 1 − false positive rate) chart, based on the combined training and testing samples. The 45-degree line from the upper right corner of the chart to the lower left represents the scenario of randomly guessing the class. The more the curve moves away the 45-degree baseline, the more accurate is the classification.

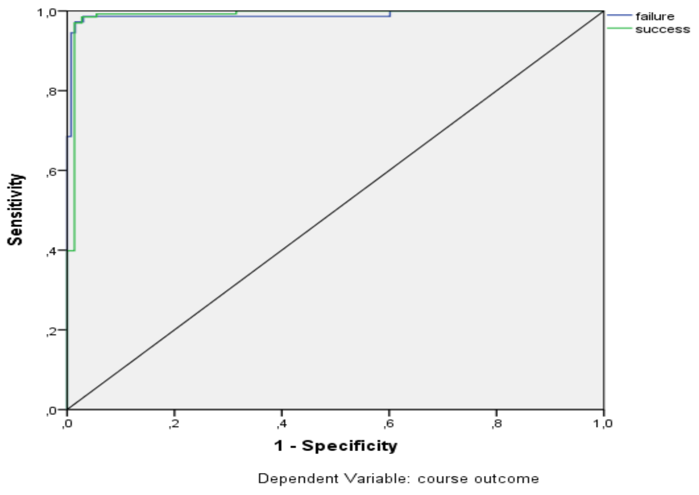


Figure 5. ROC curve.

Figure 6 gives the area under the ROC curve. The area value shows that, if a student from the *success* category and a student from the *failure* category are randomly selected, there is 0.989 probability that the model-predicted pseudo-probability for the first student of being in the *success* category, is higher than the model-predicted pseudo-probability for the second student of being in the *success* category.

		Area
course outcome	failure	,989
	success	,989

Figure 6. Area under the curve.

The chart in Figure 7 gives the cumulative gains that is the presentence of correct classifications obtained by the ANN model against the correct classifications that could result by chance (i.e. without using the model). For example, the third point on the curve for the *failure* category is at (30%, 84%), meaning that if the network score a dataset and sort all of the cases by predicted pseudo-probability of *failure*, it would be expected the top 30% to contain approximately 84% of all of the cases that actually take the category *failure*. The selection of 100% of the scored dataset, obtains all of the observed *failure* cases in the dataset. Gain is a measure of the effectiveness of a classification model calculated as the percentage of correct predictions obtained with the model, versus the percentage of correct predictions obtained without a model (baseline). The farther above the basline a curve lies, the greater the gain. A higher overall gain indicates better performance.

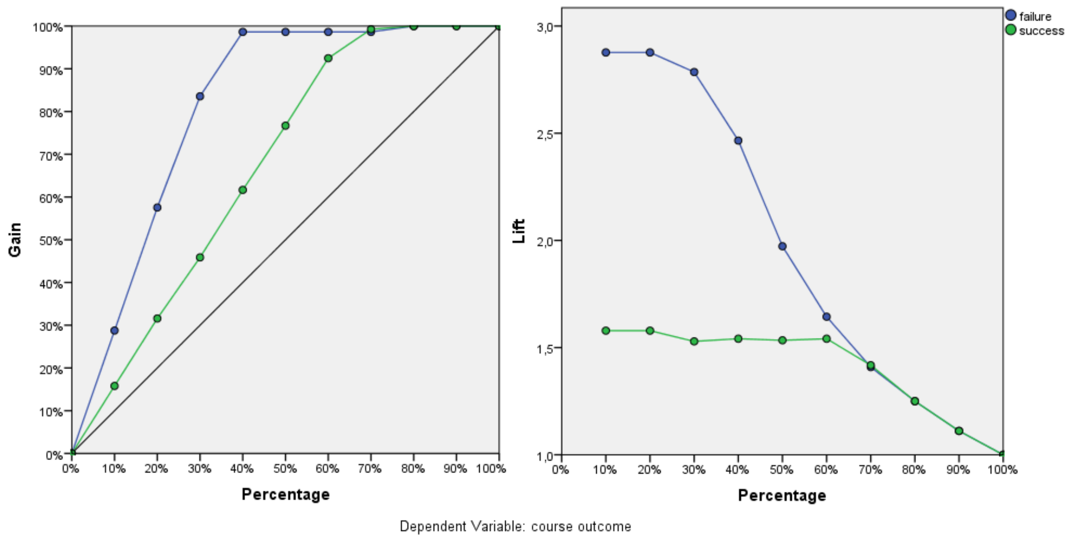


Figure 7.Cumulative Gains and Lift Charts.

Lift charts, as well as gain charts, are visual aids for evaluating performance of classification models. However, in contrast to the confusion matrix that evaluates models on the whole population, gain or lift chart evaluates model performance in a portion of the population. A lift chart uses a part of the dataset to give a clear view of the benefit to use a model in contrast to not using a model. The values from the gains diagram are used to calculate the lift factor (i.e. the benefit): the lift at 84% for the category failure is 84%/30% = 2.8.

Figure 8 gives the impact of each independent variable in the ANN model in terms of relative and normalized importance. Chart in Figure 9 also depicts the importance of the variables, i.e how sensitive is the model is the change of each input variable.

	Importance	Normalized Importance
messages	,376	100,0%
ccc	,347	92,1%
quiz_efforts	,216	57,5%
files_viewed	,061	16,1%

Figure 8. Independent variable importance.

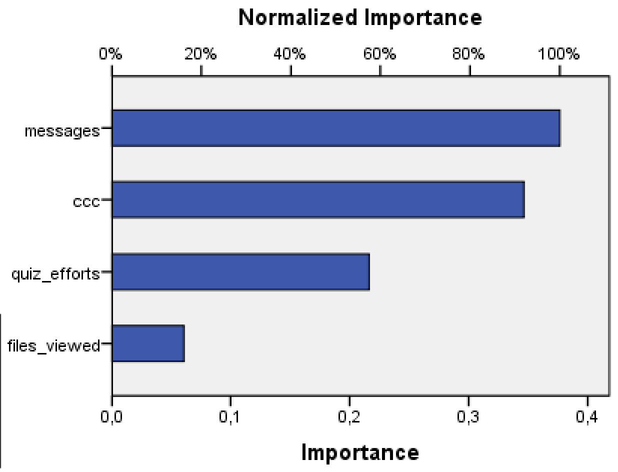


Figure 9. Independent variable importance chart.

From the chart is apparent that variables related to student's engagement with peers and instructor (*messages*) and collaborative creation of new content (*ccc*), have the greatest effect on how the network classifies students, in terms of course outcomes. Supporting self directed learning by means such as online quizzes (*quiz efforts*) is also a major determinant of model predictive power, by far more important than the engagement with the online course modules, as expressed by *files viewed*.

6. CONCLUSION

The aim of this research was to determine the effectiveness of artificial neural networks in predicting student achievement, based on data collected from students' online activities in Web-based blended learning courses. The literature review indicated that neural networks outperform all other classifiers, regarding prediction accuracy. A multilayer perceptron neural network was trained by back-propagation algorithm, to predict students ability to successfully pass the course. The classification accuracy rate was very high, with 98.3% accuracy in classifying the students into the predicted success and failure categories. The results also showed that the most powerful predictors of course outcome were the numbers of messages posted by the students and the contributions they made in team content creation projects. Although future work will need to validate these findings in larger and more diverse samples, there is strong evidence that the proposed model can be used effectively to predict student course achievement and help instructor to design timely interventions that increase the possibility of success.

REFERENCES

[1] Macfadyen, L. P., & Dawson, S. (2010). Mining LMS data to develop an "early warning system" for educators: A proof of concept. *Computers & Education, 54*(2), 588–599.

[2] Zacharis, N. Z. (2015). A multivariate approach to predicting student outcomes in web-enabled blended learning courses. *Internet and Higher Education, 27*, 44–53.

[3] Strang, D. K. (2016). Can online student performance be forecasted by learning analytics? *International Journal of Technology Enhanced Learning, 8*(1), 26-47.

[4] Sabourin, J., Rowe, J., Mott, B., Lester, J. (2011). When Off-Task in On-Task: The Affective Role of Off-Task Behavior in Narrative-Centered Learning Environments. *Proceedings of the 15th International Conference on Artificial Intelligence in Education*, 534-536.

[5] Baker, R.S.J.d., Yacef, K. (2009). The State of Educational Data Mining in 2009: A Review and Future Visions. *Journal of Educational Data Mining, 1*(1), 3-17.

[6] Lykourentzou, I., Giannoukos, I., Mpardis, G., Nikolopoulos, V. and Loumos, V. (2009), Early and dynamic student achievement prediction in e-learning courses using neural networks. *J. Am. Soc. Inf. Sci., 60*: 372–380. doi: 10.1002/asi.20970

[7] Paliwal, M., & Kumar, U. A. (2009). A study of academic performance of business school graduates using neural network and statistical techniques. *Expert Systems with Applications, 36*(4), 7865–7872.

[8] Jayne C, Lanitis A, Christodoulou C (2011). Neural network methods for one-to-many multi-valued mapping problems. *Neural Comput Appl 20*(6):775–785

[9] Kanakana, G.M., Olanrewaju, A.O. (2011). Predicting student performance in engineering education using an artificial neural network at Tshwane university of technology. *Proceedings of the International Conference on Industrial Engineering, Systems Engineering and Engineering Management for Sustainable Global Development, Stellenbosch, South Africa*, pp. 1–7.

[10] Shahiri, A.M., Husain, W., Rashid, A.N. (2015). A review on predicting student's performance using data mining techniques. *Procedia Computer Science, 72*, 414-422.

[11] McClelland, J.L., Rumelhart, D.E., and Hinton, G.E. (1986). The appeal of parallel distributed processing, in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition - Foundations, Vol.1, MIT Press, Cambridge*, pp.3-44.

[12] Leverington, D. (2009). A Basic Introduction to Feedforward Backpropagation Neural Networks. http://www.webpages.ttu.edu/dleverin/neural_network/neural_networks.html

[13] Rojas Raúl (1996). *Neural Networks: A Systematic Introduction*, Springer-Verlag, Berlin, New-York.

[14] Marwala, T. (2010). *Finite Element Model Updating Using Computational Intelligence Techniques: Applications to Structural Dynamics*, Springer Publishing Company, Inc .

[15] IBM (2016). Knowledge Center. http://goo.gl/SuuMHu

[16] Møller, M.F., 1993. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks, 6* (4),525–533.