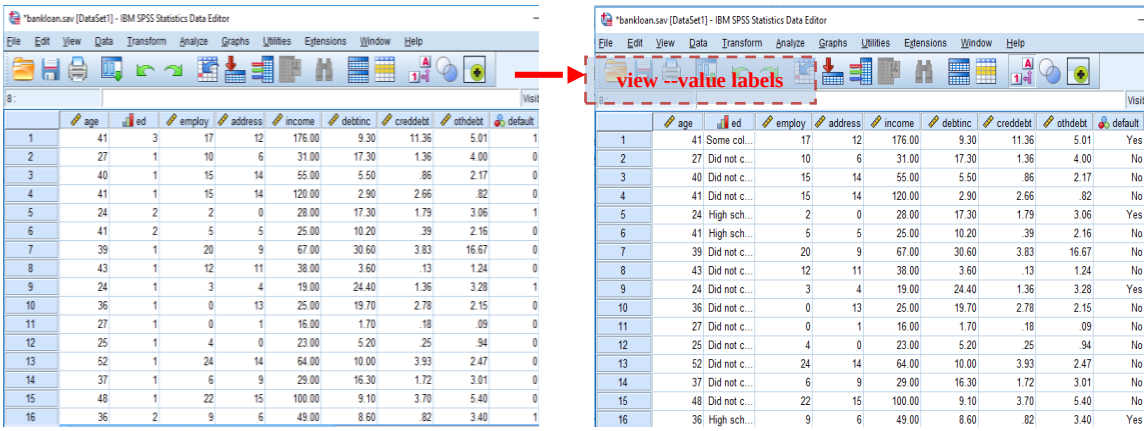
Data Mining with IBM SPSS 26.0
(IBM: Statistical Package for Social Sciences 26.0)

SPSS: Neural Networks

Case Study

Research team has collected data from loan applicants on level of education, *employ* 'years with current employer', *address* 'years at current address', *income* 'household income' (Rs.000), *debtinc* 'debt to income ratio', *creddebt* 'credit card debt', *othdebt* 'other debts' and *default* 'previous default' (0= No, 1 = Yes). SPSS data file named *bankloan*.sav contains the data from 850 respondents.



Research Questions

- To identify possible defaulters among a pool of loan applicants.
- To apply Multilayer Perceptron (MLP) neural network, which is function of the measurements that minimize the error in predict default.

The Multilayer Perceptron (MLP) procedure produces a predictive model for one or more dependent (target) variables (***Nominal, ordinal or scale***) based on the values of the predictor variables. In our case the target variable is ***Nominal*** *default* (0= No, 1 = Yes). To obtain the output of the above data, from the menus choose:

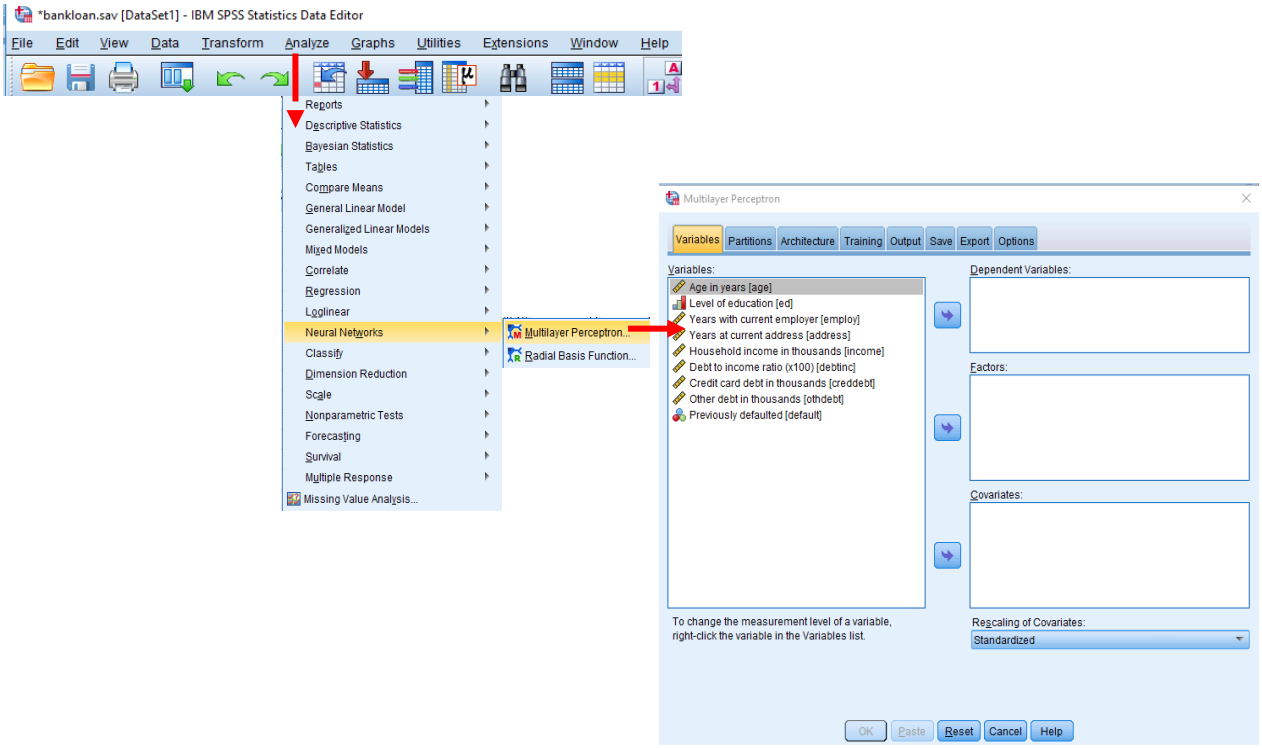
1. Analyze > Neural Networks > Multilayer Perceptron...
2. Select dependent variable (*default*).
3. Select covariates (*employ*, *debtinc*, *creddebt*, *othdebt*).

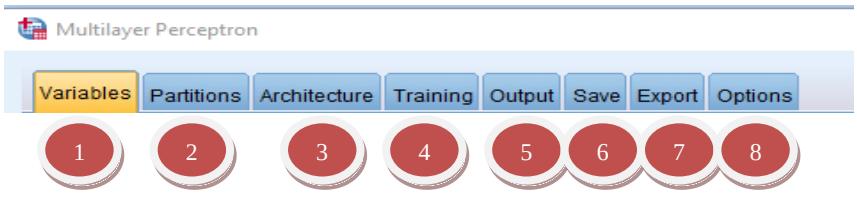
Optionally, on the Variables tab you can change the method for rescaling covariates. The choices are:

- **Standardized** $\frac{(x-mean)}{s}$
- **Normalized** $\frac{(x-min)}{(max-min)}$
- **Normalized** [0,1] Adjusted Normal $\frac{2*(x-min)}{(max-min)} - 1$, [-1,1]
- **None** No rescaling of covariates

Syntax

```
*Multilayer Perceptron Network.
MLP default (MLEVEL=N) WITH employ debtinc creddebt othdebt
 /RESCALE COVARIATE=NONE
  /PARTITION  TRAINING=7  TESTING=3  HOLDOUT=0
  /ARCHITECTURE   AUTOMATIC=YES (MINUNITS=1 MAXUNITS=50)
  /CRITERIA TRAINING=BATCH OPTIMIZATION=SCALEDCONJUGATE
LAMBDAINITIAL=0.0000005
    SIGMAINITIAL=0.00005 INTERVALCENTER=0 INTERVALOFFSET=0.5
MEMSIZE=1000
  /PRINT CPS NETWORKINFO SUMMARY CLASSIFICATION
  /PLOT NETWORK ROC GAIN LIFT PREDICTED
  /SAVE PREDVAL
  /STOPPINGRULES ERRORSTEPS= 1 (DATA=AUTO) TRAININGTIMER=ON
(MAXTIME=15) MAXEPOCHS=AUTO
    ERRORCHANGE=1.0E-4 ERRORRATIO=0.001
 /MISSING USERMISSING=EXCLUDE .
```





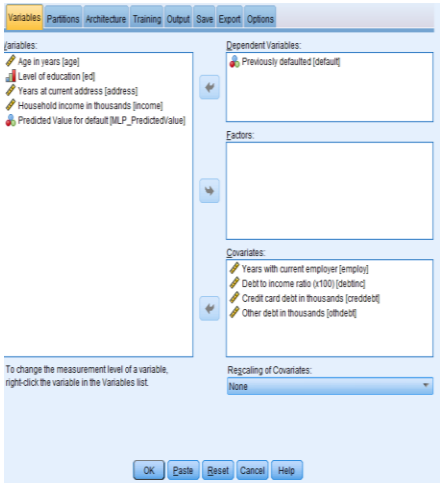
1.

Select dependent variable (*default*).

Select covariates (*employ*, *debtinc*, *creddebt*, *othdebt*).

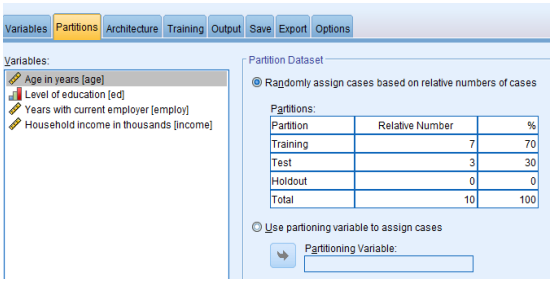
Rescaling covariates [Optional]

- **Standardized** $\frac{(x-mean)}{s}$
- **Normalized** $\frac{(x-min)}{(max-min)}$
- **Normalized** [0,1] Adjusted Normal $\frac{2*(x-min)}{(max-min)} - 1$, [-1,1]
- **None** No rescaling of covariates



2.

Partition Dataset. This group specifies the method of partitioning the active dataset into training, testing, and holdout samples.



The **training sample** comprises the data records used to train the neural network; some percentage of cases in the dataset must be assigned to the training sample in order to obtain a model.

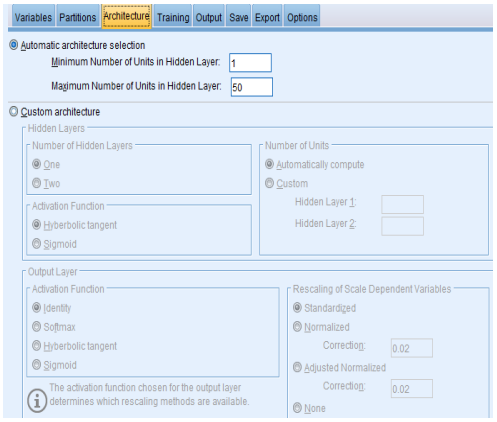
The **testing sample** is an independent set of data records used to track errors during training in order to prevent overtraining. It is highly recommended that you create a training sample, and *network training will generally be most efficient if the testing sample is smaller than the training sample*.

The **holdout sample** is another independent set of data records used to assess the final neural network; the error for the holdout sample gives an "honest" estimate of the predictive ability of the model because the holdout cases were not used to build the model.

Randomly assign cases based on relative number of cases. Specify the relative number (ratio) of cases randomly assigned to each sample (training, testing, and holdout). The % column reports the percentage of cases that will be assigned to each sample based on the relative numbers you have specified. For example, specifying 7, 3, 0 as the relative numbers for training, testing, and holdout samples corresponds to 70%, 30%, and 0%. Specifying 2, 1, 1 as the relative numbers corresponds to 50%, 25%, and 25%; 1, 1, 1 corresponds to dividing the dataset into equal thirds among training, testing, and holdout.

3.

The Architecture tab is used to specify the structure of the network. The procedure can select the "best" architecture automatically, or you can specify a custom architecture.



Automatic architecture selection builds a network with one hidden layer. Specify the *minimum* and *maximum* number of units allowed in the hidden layer, and the automatic architecture selection computes the "best" number of units in the hidden layer. Automatic architecture selection uses the default activation functions for the hidden and output layers.

Custom architecture selection gives you expert control over the hidden and output layers and can be most useful when you know in advance what architecture you want or when you need to tweak the results of the Automatic architecture selection.

Hidden Layers: The hidden layer contains unobservable network nodes (units). Each hidden unit is a function of the weighted sum of the inputs. The function is the activation function, and the values of the weights are determined by the estimation algorithm. If the network contains a second hidden layer, each hidden unit in the second layer is a function of the weighted sum of the units in the first hidden layer. The same activation function is used in both layers.

Number of Hidden Layers. A multilayer perceptron can have one or two hidden layers

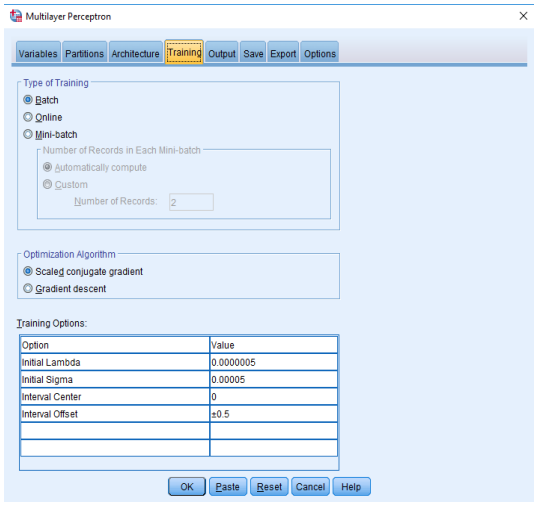
Activation Function. The activation function "links" the weighted sums of units in a layer to the values of units in the succeeding layer. **Hyperbolic tangent fu**nction: $f(x) = \frac{e^{2x}-1}{e^{2x}+1}$ it takes real-valued arguments and transforms them to the range (–1, 1). When automatic architecture selection is used, this is the activation function for all units in the hidden layers **Sigmoid function**. $S(x) = \frac{e^{x}}{1+e^{x}}$ It takes real-valued arguments and transforms them to the range (0, 1).

Number of Units. The number of units in each hidden layer can be specified explicitly or determined automatically by the estimation algorithm.

Output Layer The output layer contains the target (dependent) variables. *Activation Fun*ction. The activation function "links" the weighted sums of units in a layer to the values of units in the succeeding layer. I**dentity function**: It takes real-valued arguments and returns them unchanged. When automatic architecture selection is used, this is the activation function for units in the output layer if there are any scale-dependent variables. **Softmax:** This function has the form: $\gamma(x_k) = \exp(x_k)/\Sigma_j \exp(x_j)$. It takes a vector of real-valued arguments and transforms it to a vector whose elements fall in the range (0, 1) and sum to 1. *Softmax is available only if all dependent variables are categorical*. When automatic architecture selection is used, this is the activation function for units in the output layer if all dependent variables are categorical. **Hyperbolic tangent function** *or* **Sigmoid function**.

4.

Training tab is used to specify how the network should be trained. The type of training and the optimization algorithm determine which training options are available



Type of Training determines how the network processes the records. Select one of the following training types:

Batch updates the synaptic weights (strength of a connection) only after passing all training data records; that is, batch training uses information from all records in the training dataset. Batch training is often preferred because it directly minimizes the total error; however, batch training may need to update the weights many times until one of the stopping rules is met and hence may need many data passes. It is most useful for "smaller" datasets.

Online updates the synaptic weights after every single training data record; that is, online training uses information from one record at a time. Online training continuously gets a record and updates the weights until one of the stopping rules is met. If all the records are used once and none of the stopping rules is met, then the process continues by recycling the data records. Online training is superior to batch for "larger" datasets with associated predictors; that is, if there are many records and many inputs, and their values are not independent of each other, then online training can more quickly obtain a reasonable answer than batch training.

Mini-batch divides the training data records into groups of approximately equal size, then updates the synaptic weights after passing one group; that is, mini-batch training uses information from a group of records. Then the process recycles the data group if necessary. Mini-batch training offers a compromise between batch and online training, and it may be best for "medium-size" datasets. The procedure can automatically determine the number of training records per mini-batch, or you can specify an integer

greater than 1 and less than or equal to the maximum number of cases to store in memory. You can set the maximum number of cases to store in memory on the Options tab.

Optimization Algorithm. This is the method used to estimate the synaptic weights.

Scaled conjugate gradient uses conjugate gradient methods apply only to batch training types, so this method is not available for online or mini-batch training.

Gradient descent. This method must be used with online or mini-batch training; it can also be used with batch training.

Training Options allow you to fine-tune the optimization algorithm. You generally will not need to change these settings unless the network runs into problems with estimation. Training options for the scaled **conjugate gradient algorithm** include:

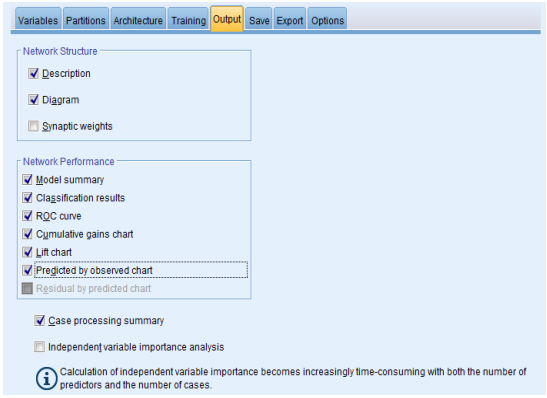
- **Initial Lambda** to specify a number in the interval (0, 0.000001).
- **Initial Sigma** to specify a number in the interval (0, 0.0001).
- **Interval Center (a_0) and Interval Offset (a)** define the interval [a_0−a, a_0+a], in which weight vectors are randomly generated when simulated annealing is used. Simulated annealing is used to break out of a local minimum, with the goal of finding the global minimum, during application of the optimization algorithm. This approach is used in weight initialization and automatic architecture selection. Specify a number for the interval center and a number greater than 0 for the interval offset.

Training options for the **gradient descent algorithm** include:

- **Initial Learning Rate**. The initial value of the learning rate for the gradient descent algorithm. A higher learning rate means that the network will train faster, possibly at the cost of becoming unstable. Specify a number greater than 0.
- **Lower Boundary of Learning Rate**. The lower boundary on the learning rate for the gradient descent algorithm. This setting applies only to online and mini-batch training. Specify a number greater than 0 and less than the initial learning rate.
- **Momentum**. The initial momentum parameter for the gradient descent algorithm. The momentum term helps to prevent instabilities caused by a too-high learning rate. Specify a number greater than 0.
- **Learning rate reduction, in Epochs**. The number of epochs (p), or data passes of the training sample, required to reduce the initial learning rate to the lower boundary of the learning rate when gradient descent is used with online or mini-batch training. This gives you control of the learning rate decay factor $\beta = (1/p\ K)*\ln(\eta_0/\eta_{low})$, where η_0 is the initial learning rate, η_{low} is the lower bound on the learning rate, and K is the total number of mini-batches (or the number of training records for online training) in the training dataset. Specify an integer greater than 0.

5.

Output tab



Network Structure displays summary information about the neural network including the dependent variables, number of input and output units, number of hidden layers and units, and activation functions.

*Diagram d*isplays the network diagram as a non-editable chart. *Note that as the number of covariates and factor levels increases, the diagram becomes more difficult to interpret*.

Synaptic weights displays the coefficient estimates that show the relationship between the units in a given layer to the units in the following layer. The synaptic weights are based on the training sample even if the active dataset is partitioned into training, testing, and holdout data. Note that the number of synaptic weights can become rather large and that these weights are generally not used for interpreting network results. Network Performance. Displays results used to determine whether the model is "good". Note: Charts in this group are based on the combined training and testing samples or only on the training sample if there is no testing sample.

Case processing summary displays the case processing summary table, which summarizes the number of cases included and excluded in the analysis, in total and by training, testing, and holdout samples.

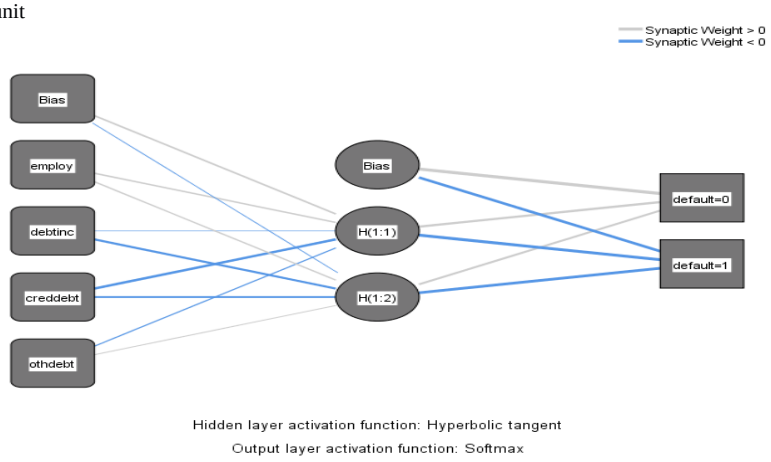
Case Processing Summary

		N	Percent
Sample	Training	485	69.3%
	Testing	215	30.7%
Valid		700	100.0%
Excluded		150	
Total		850	

Network Information

Input Layer	Covariates	1	Years with current employer
		2	Debt to income ratio (x100)
		3	Credit card debt in thousands
		4	Other debt in thousands
	Number of Units[a]		4
	Rescaling Method for Covariates		None
Hidden Layer(s)	Number of Hidden Layers		1
	Number of Units in Hidden Layer 1[a]		2
	Activation Function		Hyperbolic tangent
Output Layer	Dependent Variables	1	Previously defaulted
	Number of Units		2
	Activation Function		Softmax
	Error Function		Cross-entropy

a. Excluding the bias unit



This structure is known as a ***feed-forward architecture*** because the connections in the network flow forward from the input layer to the output layer. ***The input layer*** contains the predictors. ***The hidden layer*** contains unobservable nodes, or units. The value of each hidden unit is some function of the predictors; the exact form of the function is user-controllable specifications. ***The output layer*** contains the responses. Since the history of default is a categorical variable with two categories, it is recoded as two indicator variables. Each output unit is some function of the hidden units. Again, the exact form of the function is user-controllable specifications.

The MLP network allows a second hidden layer; in that case, each unit of the second hidden layer is a function of the units in the first hidden layer, and each response is a function of the units in the second hidden layer.

Bias Unit: In a typical artificial neural network each neuron/activity in one "layer" is connected - via a weight - to each neuron in the next activity. Each of these activities stores some sort of computation, normally a composite of the weighted activities in previous layers. A bias unit is an "extra" neuron added to each pre-output layer that stores the value of 1. Bias units aren't connected to any previous layer and in this sense don't represent a true "activity".

Model summary displays a summary of the neural network results by partition and overall, including the error, the relative error or percentage of incorrect predictions, the stopping rule used to stop training, and the training time. The error is the sum-of-squares error when the identity, sigmoid, or hyperbolic tangent activation function is applied to the output layer. It is the cross-entropy error when the softmax activation function is applied to the output layer. Relative errors or percentages of incorrect predictions are displayed depending on the dependent variable measurement levels. If any dependent variable has scale measurement level, then the average overall relative error (relative to the mean model) is displayed. If all dependent variables are categorical, then the average percentage of incorrect predictions is displayed. Relative errors or percentages of incorrect predictions are also displayed for individual dependent variables.

Model Summary

Training	Cross Entropy Error	202.282
	Percent Incorrect Predictions	20.0%
	Stopping Rule Used	1 consecutive step(s) with no decrease in error[a]
	Training Time	0:00:00.19
Testing	Cross Entropy Error	99.585
	Percent Incorrect Predictions	22.3%

Dependent Variable: Previously defaulted

a. Error computations are based on the testing sample.

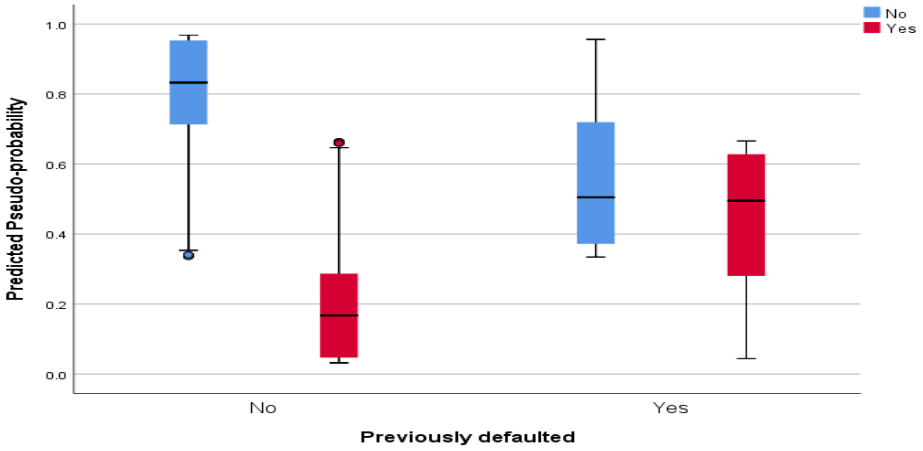
*Classification results d*isplays a classification table for each categorical dependent variable by partition and overall. Each table gives the number of cases classified correctly and incorrectly for each dependent variable category. The percentage of the total cases that were correctly classified is also reported.

Classification

		Predicted		
Sample	Observed	No	Yes	Percent Correct
Training	No	322	38	89.4%
	Yes	59	66	52.8%
	Overall Percent	78.6%	21.4%	80.0%
Testing	No	143	14	91.1%
	Yes	34	24	41.4%
	Overall Percent	82.3%	17.7%	77.7%

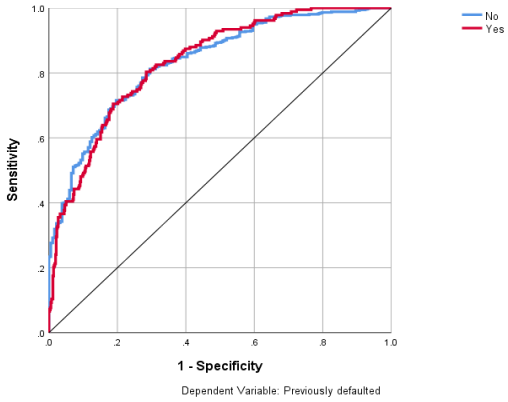
Dependent Variable: Previously defaulted

Predicted by observed chart displays a predicted-by-observed-value chart for each dependent variable. For categorical dependent variables, clustered boxplots of predicted pseudo-probabilities (an approximation of joint probability distribution) are displayed for each response category, with the observed response category as the cluster variable. *For scale-dependent variables, a scatterplot is displayed.*



The ROC curve (Receiver Operating Characteristic) is created by plotting the *Sensitivity*-true positive rate (TPR) also called probability of detection against the (1-specificity)-false positive rate (FPR) probability of false alarm at various threshold settings. It can also be thought of as a plot of the power as a function of the Type I Error of the decision rule. The ROC curve is thus the sensitivity as a function of fall-out.

ROC curve displayed for each categorical dependent variable. It also displays a table giving the area under each curve. For a given dependent variable, the ROC chart displays one curve for each category. *If the dependent variable has two categories, then each curve treats the category at issue as the positive state versus the other category*. If the dependent variable has more than two categories, then each curve treats the category at issue as the positive state versus the aggregate of all other categories.



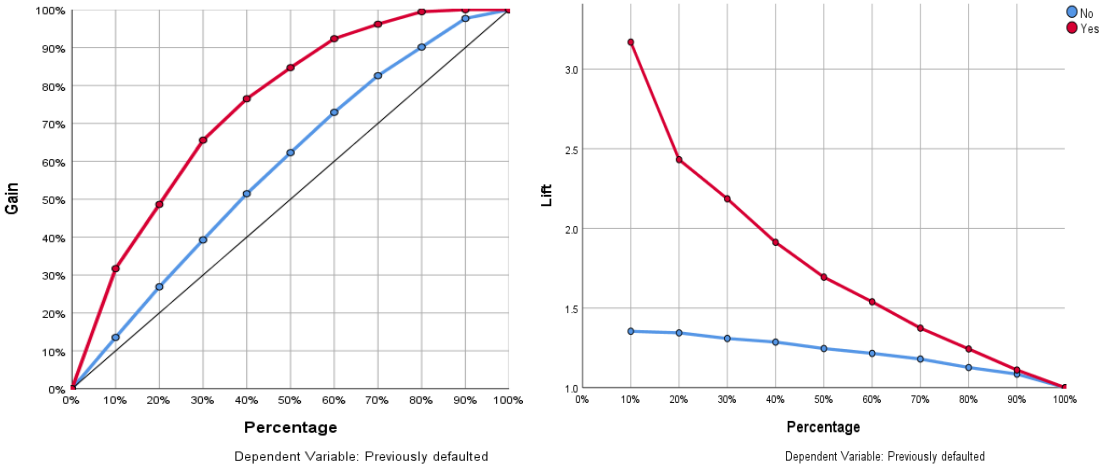
Area Under the Curve

		Area
Previously defaulted	No	.834
	Yes	.834

That is, AUC measures the entire two-dimensional area underneath the entire ROC curve from (0,0) to (1,1). A model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0. AUC is desirable for the following two reasons:

- AUC is scale-invariant. It measures how well predictions are ranked, rather than their absolute values.
- AUC is classification-threshold-invariant. It measures the quality of the model's predictions irrespective of what classification threshold is chosen.

Cumulative gains & Lift chart displays a cumulative gains chart for each categorical dependent variable. The display of one curve for each dependent variable category is the same as for ROC curves.



Cumulative Gains Chart:

- The *y*-axis shows the percentage of % default (yes and no).
- The *x*-axis shows the percentage of loan applicants (% total).
- **Baseline (overall response rate):** If we contact *X*% of applicants then receive *X*% of the responses.
- **Lift Curve:** Using the predictions of the response model, calculate the percentage of yes and no defaults for the percent of loan applicants and map these points to create the lift curve.

Lift Chart: Shows the actual lift. Calculate the points on the lift curve by determining the ratio between the result predicted by our model and the result using no model. For contacting 20% of loan applicant, Gain % 20% (Baseline), 28% (No) & 50% (Yes). The *y*-value of the lift curve at 20% is 50 / 20 = 2.5 (Yes), 28/20 1.4 (No).

Probabilities and Pseudo-Probabilities
Categorical dependent variables with softmax activation and cross-entropy error will have a predicted value for each category, where each predicted value is the probability that the case belongs to the category.
Categorical dependent variables with sum-of-squares error will have a predicted value for each category, but the predicted values cannot be interpreted as probabilities. The procedure saves these predicted pseudo-probabilities even if any are less than 0 or greater than 1, or the sum for a given dependent variable is not 1.

The ROC, cumulative gains, and lift charts are created based on pseudo-probabilities. In the event that any of the pseudo-probabilities are less than 0 or greater than 1, or the sum for a given variable is not 1, they are first rescaled to be between 0 and 1 and to sum to 1. Pseudo-probabilities are rescaled by dividing by their sum. For example, if a case has predicted pseudo-probabilities of 0.50, 0.60, and 0.40 for a three-category dependent variable, then each pseudo-probability is divided by the sum 1.50 to get 0.33, 0.40, and 0.27. If any of the pseudo-probabilities are negative, then the absolute value of the lowest is added to all pseudo-probabilities before the above rescaling. For example, if the pseudo-probabilities are -0.30, 0.50, and 1.30, then first add 0.30 to each value to get 0.00, 0.80, and 1.60. Next, divide each new value by the sum 2.40 to get 0.00, 0.33, and 0.67.

6.

Save tab is used to save predictions as variables in the dataset.

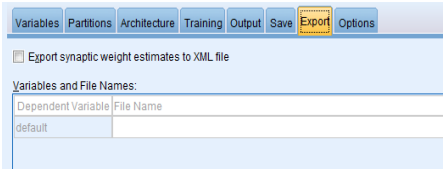
Save predicted value or category for each dependent variable saves the predicted value for scale-dependent variables and the predicted category for categorical dependent variables.

Save predicted pseudo-probability or category for each dependent variable saves the predicted pseudo-probabilities for categorical dependent variables. A separate variable is saved for each of the first n categories, where n is specified in the Categories to Save column.

Name of Saved Variables generates automatic name and ensures that you keep all of your work.

Custom names allow you to discard/replace results from previous runs without first deleting the saved variables in the Data Editor.

7.



Export tab is used to save the synaptic weight estimates for each dependent variable to an Extensible Markup Language XML (Predictive Model Markup Language PMML) file. You can use this model file to apply the model information to other data files for scoring purposes.

8.

Options
User-Missing Values. Factors must have valid values for a case to be included in the analysis. These controls allow you to decide whether user-missing values are treated as valid among factors and categorical dependent variables.

Stopping Rules. These are the rules that determine when to stop training the neural network. Training proceeds through at least one data pass. Training can then be stopped according to the following criteria, which are checked in the listed order. In the stopping rule definitions that follow, a step corresponds to a data pass for the online and mini-batch methods and an iteration for the batch method.

Maximum steps The number of steps to allow before checking for a decrease in error. If there is no decrease in error after the specified number of steps, then training stops. Specify an integer greater than 0.
Data to use for computing prediction error. Choose automatically uses the testing sample if it exists and uses the training sample otherwise. *Note* that batch training guarantees a decrease in the training sample error after each data pass; thus, this option applies only to batch training if a testing sample exists. Both training and test data checks the error for each of these samples; this option applies only if a testing sample exits.

Note: After each complete data pass, online and mini-batch training require an extra data pass in order to compute the training error. This extra data pass can slow training considerably, so it is generally recommended that you supply a testing sample and select Choose automatically in any case.

Maximum training time (minutes) for the algorithm to run (>0).

Maximum Training Epochs (data passes) if exceeds than training stops (>0).

Minimum relative change in training error if the relative change in the training error compared to the previous step is less than the criterion value then training stops (>0). For online and mini-batch training, this criterion is ignored if only testing data is used to compute the error.

Minimum relative change in training error ratio. Training stops if the ratio of the training error to the error of the null model is less than the criterion value. The null model predicts the average value for all dependent variables. Specify a number greater than 0. For online and mini-batch training, this criterion is ignored if only testing data is used to compute the error.

Maximum cases to store in memory. This controls the following settings within the multilayer perceptron algorithms. Specify an integer greater than 1

- In automatic architecture selection, the size of the sample used to determine the network archicteture is min(1000, memsize), where memsize is the maximum number of cases to store in memory.
- In mini-batch training with automatic computation of the number of mini-batches, the number of mini-batches is *min*(max(M/10,2), memsize), where M is the number of cases in the training sample.

Limitations of Neural Networks

While Neural Networks has many advantages like handling of nominal/categorical data with non-linear functionality, it has also certain limitations:

- Do not know about distribution of the outcomes
- Cannot propose the test of equality ‘t’, ‘z’ …etc
- Can only cont the number of prediction with real and results in different output with different samples.