IBM SPSS Neural Networks 21

IBM ®

Preface

IBM® SPSS® Statistics is a comprehensive system for analyzing data. The Neural Networks optional add-on module provides the additional analytic techniques described in this manual. The Neural Networks add-on module must be used with the SPSS Statistics Core system and is completely integrated into that system.

About IBM Business Analytics

IBM Business Analytics software delivers complete, consistent and accurate information that decision-makers trust to improve business performance. A comprehensive portfolio of business intelligence, predictive analytics, financial performance and strategy management, and analytic applications provides clear, immediate and actionable insights into current performance and the ability to predict future outcomes. Combined with rich industry solutions, proven practices and professional services, organizations of every size can drive the highest productivity, confidently automate decisions and deliver better results.

As part of this portfolio, IBM SPSS Predictive Analytics software helps organizations predict future events and proactively act upon that insight to drive better business outcomes. Commercial, government and academic customers worldwide rely on IBM SPSS technology as a competitive advantage in attracting, retaining and growing customers, while reducing fraud and mitigating risk. By incorporating IBM SPSS software into their daily operations, organizations become predictive enterprises – able to direct and automate decisions to meet business goals and achieve measurable competitive advantage. For further information or to reach a representative visit *http://www.ibm.com/spss*.

Technical support

Technical support is available to maintenance customers. Customers may contact Technical Support for assistance in using IBM Corp. products or for installation help for one of the supported hardware environments. To reach Technical Support, see the IBM Corp. web site at *http://www.ibm.com/support*. Be prepared to identify yourself, your organization, and your support agreement when requesting assistance.

Technical Support for Students

If you're a student using a student, academic or grad pack version of any IBM SPSS software product, please see our special online Solutions for Education (*http://www.ibm.com/spss/rd/students/*) pages for students. If you're a student using a university-supplied copy of the IBM SPSS software, please contact the IBM SPSS product coordinator at your university.

Customer Service

If you have any questions concerning your shipment or account, contact your local office. Please have your serial number ready for identification.

Training Seminars

IBM Corp. provides both public and onsite training seminars. All seminars feature hands-on workshops. Seminars will be offered in major cities on a regular basis. For more information on these seminars, go to *http://www.ibm.com/software/analytics/spss/training*.

Contents

Part I: User's Guide

1   Introduction to Neural Networks   *1*

2   Multilayer Perceptron   *4*

3   Radial Basis Function   *22*

Part II: Examples

4   Multilayer Perceptron   *36*

5 Radial Basis Function   72

Appendices

Part I:
User's Guide

Introduction to Neural Networks

Neural networks are the preferred tool for many predictive data mining applications because of their power, flexibility, and ease of use. Predictive neural networks are particularly useful in applications where the underlying process is complex, such as:

- Forecasting consumer demand to streamline production and delivery costs.
- Predicting the probability of response to direct mail marketing to determine which households on a mailing list should be sent an offer.
- Scoring an applicant to determine the risk of extending credit to the applicant.
- Detecting fraudulent transactions in an insurance claims database.

Neural networks used in predictive applications, such as the **multilayer perceptron (MLP)** and **radial basis function (RBF)** networks, are supervised in the sense that the model-predicted results can be compared against known values of the target variables. The Neural Networks option allows you to fit MLP and RBF networks and save the resulting models for scoring.

What Is a Neural Network?

The term **neural network** applies to a loosely related family of models, characterized by a large parameter space and flexible structure, descending from studies of brain functioning. As the family grew, most of the new models were designed for nonbiological applications, though much of the associated terminology reflects its origin.

Specific definitions of neural networks are as varied as the fields in which they are used. While no single definition properly covers the entire family of models, for now, consider the following description (Haykin, 1998):

A neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

- Knowledge is acquired by the network through a learning process.
- Interneuron connection strengths known as synaptic weights are used to store the knowledge.

For a discussion of why this definition is perhaps too restrictive, see (Ripley, 1996).

In order to differentiate neural networks from traditional statistical methods using this definition, what is *not* said is just as significant as the actual text of the definition. For example, the traditional linear regression model can acquire knowledge through the least-squares method and store that knowledge in the regression coefficients. In this sense, it is a neural network. In fact, you can argue that linear regression is a special case of certain neural networks. However, linear regression has a rigid model structure and set of assumptions that are imposed before learning from the data.

By contrast, the definition above makes minimal demands on model structure and assumptions. Thus, a neural network can approximate a wide range of statistical models without requiring that you hypothesize in advance certain relationships between the dependent and independent

1

variables. Instead, the form of the relationships is determined during the learning process. If a linear relationship between the dependent and independent variables is appropriate, the results of the neural network should closely approximate those of the linear regression model. If a nonlinear relationship is more appropriate, the neural network will automatically approximate the "correct" model structure.

The trade-off for this flexibility is that the synaptic weights of a neural network are not easily interpretable. Thus, if you are trying to explain an underlying process that produces the relationships between the dependent and independent variables, it would be better to use a more traditional statistical model. However, if model interpretability is not important, you can often obtain good model results more quickly using a neural network.

Neural Network Structure

Although neural networks impose minimal demands on model structure and assumptions, it is useful to understand the general **network architecture**. The multilayer perceptron (MLP) or radial basis function (RBF) network is a function of predictors (also called inputs or independent variables) that minimize the prediction error of target variables (also called outputs).

Consider the *bankloan.sav* dataset that ships with the product, in which you want to be able to identify possible defaulters among a pool of loan applicants. An MLP or RBF network applied to this problem is a function of the measurements that minimize the error in predicting default. The following figure is useful for relating the form of this function.

Figure 1-1
Feedforward architecture with one hidden layer
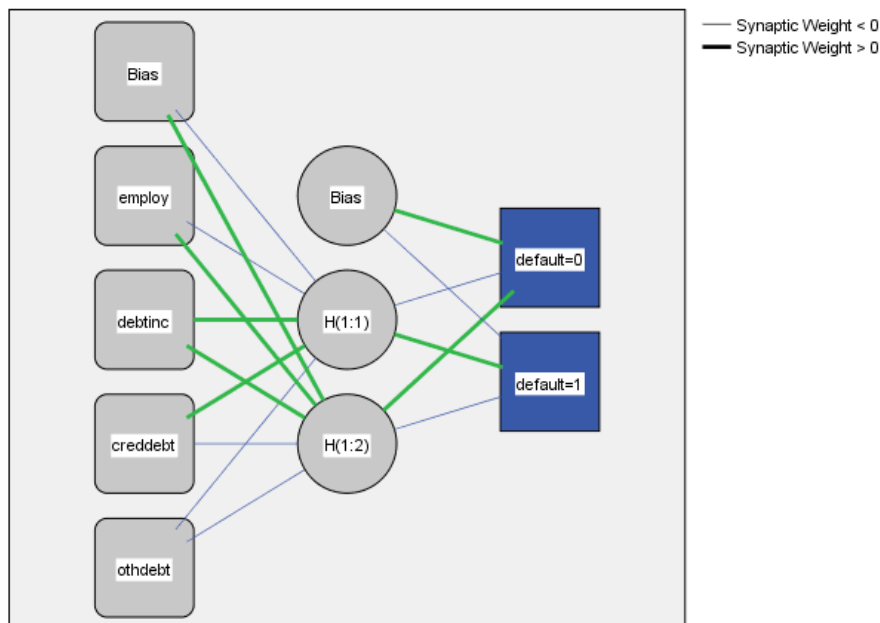
Hidden layer activation function: Hyperbolic tangent

Output layer activation function: Softmax

This structure is known as a **feedforward architecture** because the connections in the network flow forward from the input layer to the output layer without any feedback loops. In this figure:

■ 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 depends in part upon the network type and in part upon 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 depends in part on the network type and in part on 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.

Multilayer Perceptron

The Multilayer Perceptron (MLP) procedure produces a predictive model for one or more dependent (target) variables based on the values of the predictor variables.

Examples. Following are two scenarios using the MLP procedure:

A loan officer at a bank needs to be able to identify characteristics that are indicative of people who are likely to default on loans and use those characteristics to identify good and bad credit risks. Using a sample of past customers, she can train a multilayer perceptron, validate the analysis using a holdout sample of past customers, and then use the network to classify prospective customers as good or bad credit risks.

A hospital system is interested in tracking costs and lengths of stay for patients admitted for treatment of myocardial infarction (MI, or "heart attack"). Obtaining accurate estimates of these measures allows the administration to properly manage the available bed space as patients are treated. Using the treatment records of a sample of patients who received treatment for MI, the administrator can train a network to predict both cost and length of stay.

Dependent variables. The dependent variables can be:

- **Nominal.** A variable can be treated as nominal when its values represent categories with no intrinsic ranking (for example, the department of the company in which an employee works). Examples of nominal variables include region, zip code, and religious affiliation.

- **Ordinal.** A variable can be treated as ordinal when its values represent categories with some intrinsic ranking (for example, levels of service satisfaction from highly dissatisfied to highly satisfied). Examples of ordinal variables include attitude scores representing degree of satisfaction or confidence and preference rating scores.

- **Scale.** A variable can be treated as scale (continuous) when its values represent ordered categories with a meaningful metric, so that distance comparisons between values are appropriate. Examples of scale variables include age in years and income in thousands of dollars.

  The procedure assumes that the appropriate measurement level has been assigned to all dependent variables; however, you can temporarily change the measurement level for a variable by right-clicking the variable in the source variable list and selecting a measurement level from the context menu.

An icon next to each variable in the variable list identifies the measurement level and data type:

	Numeric	**String**	**Date**	**Time**
Scale (Continuous)		n/a		

	Numeric	**String**	**Date**	**Time**
Ordinal				
Nominal				

Predictor variables. Predictors can be specified as factors (categorical) or covariates (scale).

Categorical variable coding. The procedure temporarily recodes categorical predictors and dependent variables using one-of-*c* coding for the duration of the procedure. If there are *c* categories of a variable, then the variable is stored as c vectors, with the first category denoted (1,0,...,0), the next category (0,1,0,...,0), ..., and the final category (0,0,...,0,1).

This coding scheme increases the number of synaptic weights and can result in slower training; however, more "compact" coding methods usually lead to poorly fit neural networks. If your network training is proceeding very slowly, try reducing the number of categories in your categorical predictors by combining similar categories or dropping cases that have extremely rare categories.

All one-of-*c* coding is based on the training data, even if a testing or holdout sample is defined (see Partitions on p. 8). Thus, if the testing or holdout samples contain cases with predictor categories that are not present in the training data, then those cases are not used by the procedure or in scoring. If the testing or holdout samples contain cases with dependent variable categories that are not present in the training data, then those cases are not used by the procedure, but they may be scored.

Rescaling. Scale-dependent variables and covariates are rescaled by default to improve network training. All rescaling is performed based on the training data, even if a testing or holdout sample is defined (see Partitions on p. 8). That is, depending on the type of rescaling, the mean, standard deviation, minimum value, or maximum value of a covariate or dependent variable is computed using only the training data. If you specify a variable to define partitions, it is important that these covariates or dependent variables have similar distributions across the training, testing, and holdout samples.

Frequency weights. Frequency weights are ignored by this procedure.

Replicating results. If you want to replicate your results exactly, use the same initialization value for the random number generator, the same data order, and the same variable order, in addition to using the same procedure settings. More details on this issue follow:

- **Random number generation.** The procedure uses random number generation during random assignment of partitions, random subsampling for initialization of synaptic weights, random subsampling for automatic architecture selection, and the simulated annealing algorithm used in weight initialization and automatic architecture selection. To reproduce the same randomized results in the future, use the same initialization value for the random number generator before each run of the Multilayer Perceptron procedure. See Preparing the Data for Analysis on p. 36 for step-by-step instructions.

- **Case order.** The Online and Mini-batch training methods (see Training on p. 12) are explicitly dependent upon case order; however, even Batch training is dependent upon case order because initialization of synaptic weights involves subsampling from the dataset.

To minimize order effects, randomly order the cases. To verify the stability of a given solution, you may want to obtain several different solutions with cases sorted in different random orders. In situations with extremely large file sizes, multiple runs can be performed with a sample of cases sorted in different random orders.

■ **Variable order.** Results may be influenced by the order of variables in the factor and covariate lists due to the different pattern of initial values assigned when the variable order is changed. As with case order effects, you might try different variable orders (simply drag and drop within the factor and covariate lists) to assess the stability of a given solution.
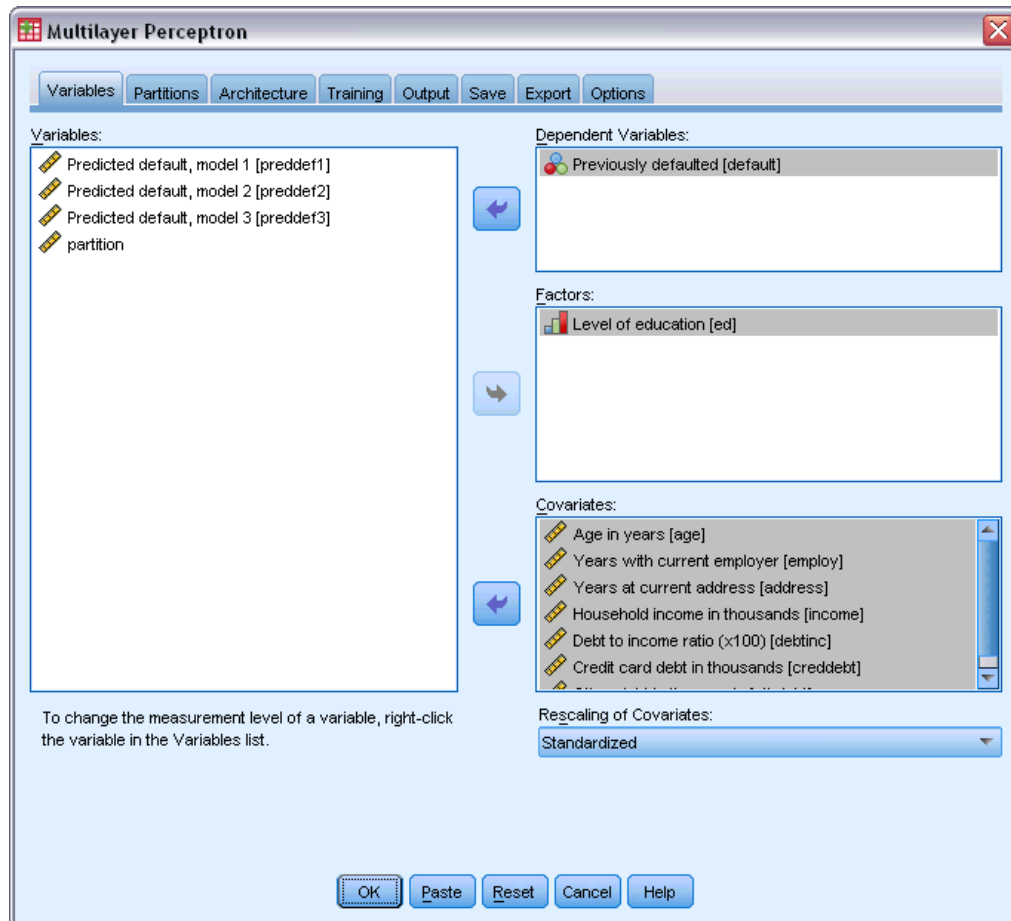
Creating a Multilayer Perceptron Network

From the menus choose:
Analyze > Neural Networks > Multilayer Perceptron...

Figure 2-1
Multilayer Perceptron: Variables tab

▶ Select at least one dependent variable.

▶ Select at least one factor or covariate.

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

- **Standardized.** Subtract the mean and divide by the standard deviation, $(x-\text{mean})/s$.

- **Normalized.** Subtract the minimum and divide by the range, $(x-\text{min})/(\text{max}-\text{min})$. Normalized values fall between 0 and 1.

- **Adjusted Normalized.** Adjusted version of subtracting the minimum and dividing by the range, $[2*(x-\text{min})/(\text{max}-\text{min})]-1$. Adjusted normalized values fall between -1 and 1.

- **None.** No rescaling of covariates.

Fields with Unknown Measurement Level

The Measurement Level alert is displayed when the measurement level for one or more variables (fields) in the dataset is unknown. Since measurement level affects the computation of results for this procedure, all variables must have a defined measurement level.

Figure 2-2
Measurement level alert
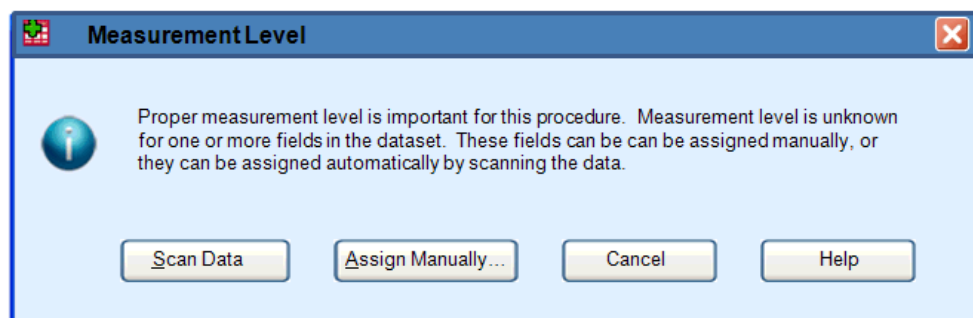
- **Scan Data.** Reads the data in the active dataset and assigns default measurement level to any fields with a currently unknown measurement level. If the dataset is large, that may take some time.

- **Assign Manually.** Opens a dialog that lists all fields with an unknown measurement level. You can use this dialog to assign measurement level to those fields. You can also assign measurement level in Variable View of the Data Editor.

Since measurement level is important for this procedure, you cannot access the dialog to run this procedure until all fields have a defined measurement level.

Partitions

Figure 2-3
Multilayer Perceptron: Partitions tab

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.

■ **Use partitioning variable to assign cases.** Specify a numeric variable that assigns each case in the active dataset to the training, testing, or holdout sample. Cases with a positive value on the variable are assigned to the training sample, cases with a value of 0, to the testing sample, and cases with a negative value, to the holdout sample. Cases with a system-missing value are excluded from the analysis. Any user-missing values for the partition variable are always treated as valid.

Note: Using a partitioning variable will not guarantee identical results in successive runs of the procedure. See "Replicating results" in the main Multilayer Perceptron topic.

Architecture

Figure 2-4
Multilayer Perceptron: Architecture tab

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.** This function has the form: $\gamma(c) = \tanh(c) = (e^c - e^{-c})/(e^c + e^{-c})$. 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.** This function has the form: $\gamma(c) = 1/(1 + e^{-c})$. 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 Function. The activation function "links" the weighted sums of units in a layer to the values of units in the succeeding layer.

- **Identity.** This function has the form: $\gamma(c) = c$. 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: $\gamma(c_k) = \exp(c_k)/\Sigma_j \exp(c_j)$. 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.** This function has the form: $\gamma(c) = \tanh(c) = (e^c - e^{-c})/(e^c + e^{-c})$. It takes real-valued arguments and transforms them to the range $(-1, 1)$.

- **Sigmoid.** This function has the form: $\gamma(c) = 1/(1 + e^{-c})$. It takes real-valued arguments and transforms them to the range $(0, 1)$.

Rescaling of Scale Dependent Variables. These controls are available only if at least one scale-dependent variable has been selected.

- **Standardized.** Subtract the mean and divide by the standard deviation, $(x - \text{mean})/s$.

- **Normalized.** Subtract the minimum and divide by the range, $(x - \text{min})/(\text{max} - \text{min})$. Normalized values fall between 0 and 1. This is the required rescaling method for scale-dependent variables if the output layer uses the sigmoid activation function. The correction option specifies a small number ε that is applied as a correction to the rescaling formula; this correction ensures that all rescaled dependent variable values will be within the range of the activation function. In particular, the values 0 and 1, which occur in the uncorrected formula when x takes its minimum and maximum value, define the limits of the range of the sigmoid function but are not within that range. The corrected formula is $[x - (\text{min} - \varepsilon)]/[(\text{max} + \varepsilon) - (\text{min} - \varepsilon)]$. Specify a number greater than or equal to 0.

- **Adjusted Normalized.** Adjusted version of subtracting the minimum and dividing by the range, $[2*(x - \text{min})/(\text{max} - \text{min})] - 1$. Adjusted normalized values fall between -1 and 1. This is the required rescaling method for scale-dependent variables if the output layer uses the hyperbolic tangent activation function. The correction option specifies a small number ε that is applied as a correction to the rescaling formula; this correction ensures that all rescaled dependent variable values will be within the range of the activation function. In particular, the values -1 and 1, which occur in the uncorrected formula when x takes its minimum and maximum value, define the limits of the range of the hyperbolic tangent function but are not within that range. The corrected formula is $\{2*[(x - (\text{min} - \varepsilon))/((\text{max} + \varepsilon) - (\text{min} - \varepsilon))]\} - 1$. Specify a number greater than or equal to 0.

- **None.** No rescaling of scale-dependent variables.

Training

Figure 2-5
Multilayer Perceptron: Training tab
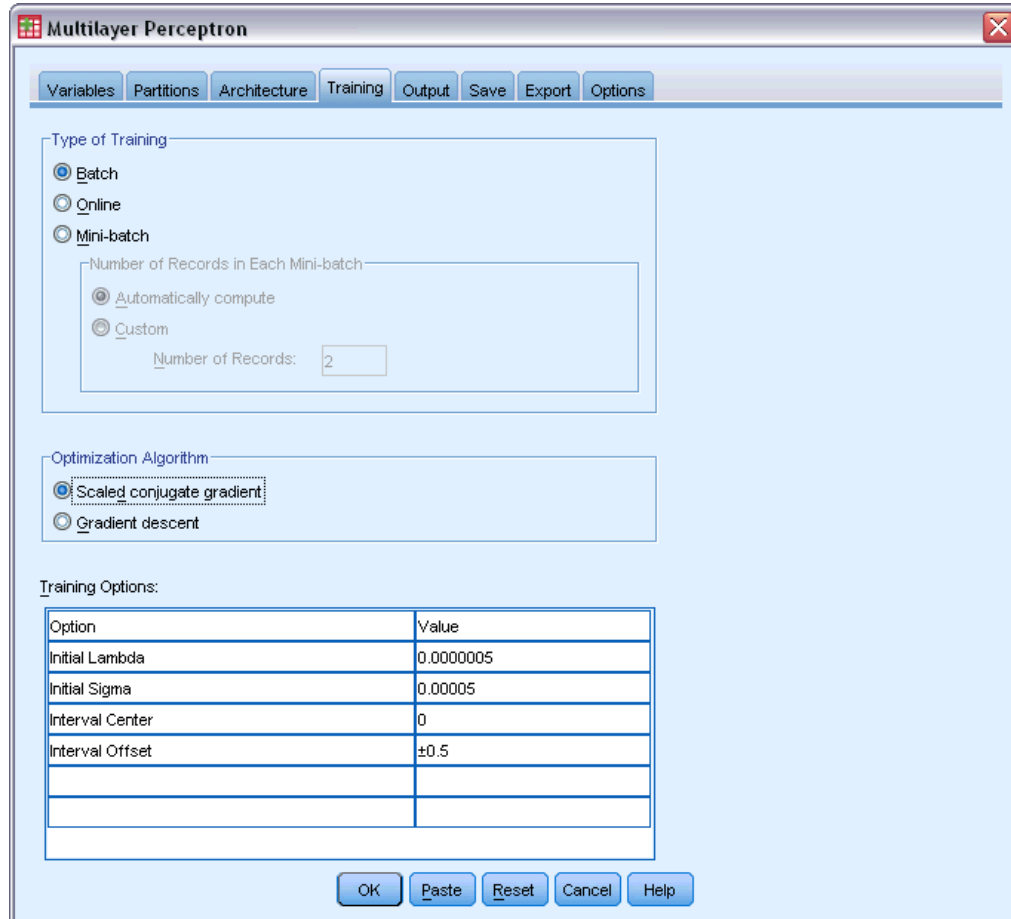
The Training tab is used to specify how the network should be trained. The type of training and the optimization algorithm determine which training options are available.

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

- **Batch.** Updates the synaptic weights 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.** The assumptions that justify the use of 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. The 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.** The initial value of the lambda parameter for the scaled conjugate gradient algorithm. Specify a number greater than 0 and less than 0.000001.
- **Initial Sigma.** The initial value of the sigma parameter for the scaled conjugate gradient algorithm. Specify a number greater than 0 and less than 0.0001.
- **Interval Center and Interval Offset.** The 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 $\beta = (1/pK)*\ln(\eta_0/\eta_{low})$, 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.

Output

Figure 2-6
Multilayer Perceptron: Output tab

Network Structure. Displays summary information about the neural network.

- **Description.** Displays 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.** Displays 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.

- **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.

- **Classification results.** Displays 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.

- **ROC curve.** Displays an ROC (Receiver Operating Characteristic) curve 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.

- **Cumulative gains 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.

- **Lift chart.** Displays a lift chart for each categorical dependent variable. The display of one curve for each dependent variable category is the same as for ROC curves.

- **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 are displayed for each response category, with the observed response category as the cluster variable. For scale-dependent variables, a scatterplot is displayed.

- **Residual by predicted chart.** Displays a residual-by-predicted-value chart for each scale-dependent variable. There should be no visible patterns between residuals and predicted values. This chart is produced only for scale-dependent variables.

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.

Independent variable importance analysis. Performs a sensitivity analysis, which computes the importance of each predictor in determining the neural network. The analysis is based on the combined training and testing samples or only on the training sample if there is no testing sample. This creates a table and a chart displaying importance and normalized importance for each predictor. Note that sensitivity analysis is computationally expensive and time-consuming if there are large numbers of predictors or cases.

Save

Figure 2-7
Multilayer Perceptron: Save tab
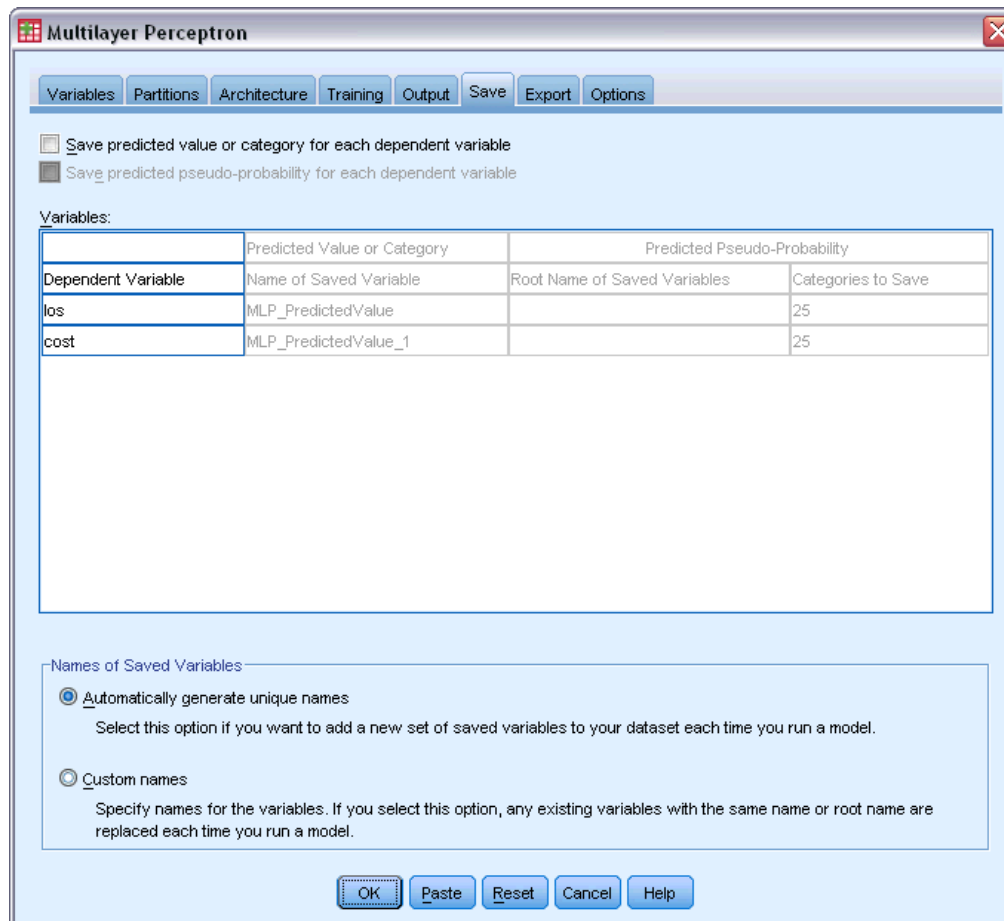
The Save tab is used to save predictions as variables in the dataset.

- **Save predicted value or category for each dependent variable.** This 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.** This 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.

Names of Saved Variables. Automatic name generation 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.

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 (see Output on p. 14) 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.

Export

Figure 2-8
Multilayer Perceptron: Export tab
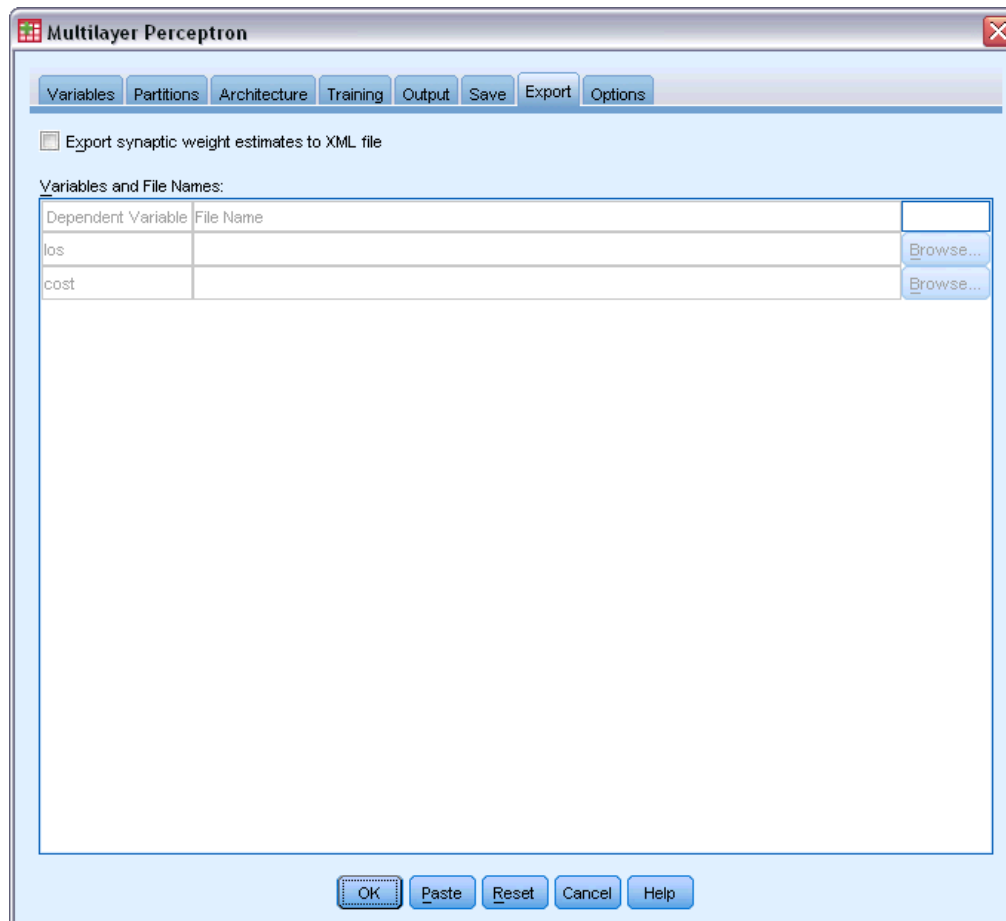
The Export tab is used to save the synaptic weight estimates for each dependent variable to an XML (PMML) file. You can use this model file to apply the model information to other data files for scoring purposes. This option is not available if split files have been defined.

Options

Figure 2-9
Multilayer Perceptron: Options tab

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 without a decrease in error.** 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. You can also specify which data sample is used to compute the 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.** Choose whether to specify a maximum number of minutes for the algorithm to run. Specify a number greater than 0.

- **Maximum Training Epochs.** The maximum number of epochs (data passes) allowed. If the maximum number of epochs is exceeded, then training stops. Specify an integer greater than 0.

- **Minimum relative change in training error.** Training stops if the relative change in the training error compared to the previous step is less than the criterion value. 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.

- **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.

Radial Basis Function

The Radial Basis Function (RBF) procedure produces a predictive model for one or more dependent (target) variables based on values of predictor variables.

Example. A telecommunications provider has segmented its customer base by service usage patterns, categorizing the customers into four groups. An RBF network using demographic data to predict group membership allows the company to customize offers for individual prospective customers.

Dependent variables. The dependent variables can be:

- **Nominal.** A variable can be treated as nominal when its values represent categories with no intrinsic ranking (for example, the department of the company in which an employee works). Examples of nominal variables include region, zip code, and religious affiliation.

- **Ordinal.** A variable can be treated as ordinal when its values represent categories with some intrinsic ranking (for example, levels of service satisfaction from highly dissatisfied to highly satisfied). Examples of ordinal variables include attitude scores representing degree of satisfaction or confidence and preference rating scores.

- **Scale.** A variable can be treated as scale (continuous) when its values represent ordered categories with a meaningful metric, so that distance comparisons between values are appropriate. Examples of scale variables include age in years and income in thousands of dollars.

    The procedure assumes that the appropriate measurement level has been assigned to all dependent variables, although you can temporarily change the measurement level for a variable by right-clicking the variable in the source variable list and selecting a measurement level from the context menu.

An icon next to each variable in the variable list identifies the measurement level and data type:

	Numeric	**String**	**Date**	**Time**
Scale (Continuous)		n/a		
Ordinal				
Nominal				

Predictor variables. Predictors can be specified as factors (categorical) or covariates (scale).

22

Categorical variable coding. The procedure temporarily recodes categorical predictors and dependent variables using one-of-*c* coding for the duration of the procedure. If there are *c* categories of a variable, then the variable is stored as c vectors, with the first category denoted $(1,0,...,0)$, the next category $(0,1,0,...,0)$, ..., and the final category $(0,0,...,0,1)$.

This coding scheme increases the number of synaptic weights and can result in slower training, but more "compact" coding methods usually lead to poorly fit neural networks. If your network training is proceeding very slowly, try reducing the number of categories in your categorical predictors by combining similar categories or dropping cases that have extremely rare categories.

All one-of-*c* coding is based on the training data, even if a testing or holdout sample is defined (see Partitions on p. 26). Thus, if the testing or holdout samples contain cases with predictor categories that are not present in the training data, then those cases are not used by the procedure or in scoring. If the testing or holdout samples contain cases with dependent variable categories that are not present in the training data, then those cases are not used by the procedure but they may be scored.

Rescaling. Scale dependent variables and covariates are rescaled by default to improve network training. All rescaling is performed based on the training data, even if a testing or holdout sample is defined (see Partitions on p. 26). That is, depending on the type of rescaling, the mean, standard deviation, minimum value, or maximum value of a covariate or dependent variable are computed using only the training data. If you specify a variable to define partitions, it is important that these covariates or dependent variables have similar distributions across the training, testing, and holdout samples.

Frequency weights. Frequency weights are ignored by this procedure.

Replicating results. If you want to exactly replicate your results, use the same initialization value for the random number generator and the same data order, in addition to using the same procedure settings. More details on this issue follow:

- **Random number generation.** The procedure uses random number generation during random assignment of partitions. To reproduce the same randomized results in the future, use the same initialization value for the random number generator before each run of the Radial Basis Function procedure. See Preparing the Data for Analysis on p. 72 for step-by-step instructions.

- **Case order.** Results are also dependent on data order because the two-step cluster algorithm is used to determine the radial basis functions.

   To minimize order effects, randomly order the cases. To verify the stability of a given solution, you may want to obtain several different solutions with cases sorted in different random orders. In situations with extremely large file sizes, multiple runs can be performed with a sample of cases sorted in different random orders.

Creating a Radial Basis Function Network

From the menus choose:
Analyze > Neural Networks > Radial Basis Function...

Figure 3-1
Radial Basis Function: Variables tab

▶ Select at least one dependent variable.

▶ Select at least one factor or covariate.

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

- **Standardized.** Subtract the mean and divide by the standard deviation, $(x-\text{mean})/s$.

- **Normalized.** Subtract the minimum and divide by the range, $(x-\text{min})/(\text{max}-\text{min})$. Normalized values fall between 0 and 1.

- **Adjusted Normalized.** Adjusted version of subtracting the minimum and dividing by the range, $[2*(x-\text{min})/(\text{max}-\text{min})]-1$. Adjusted normalized values fall between −1 and 1.

- **None.** No rescaling of covariates.

Fields with Unknown Measurement Level

The Measurement Level alert is displayed when the measurement level for one or more variables (fields) in the dataset is unknown. Since measurement level affects the computation of results for this procedure, all variables must have a defined measurement level.

Figure 3-2
Measurement level alert
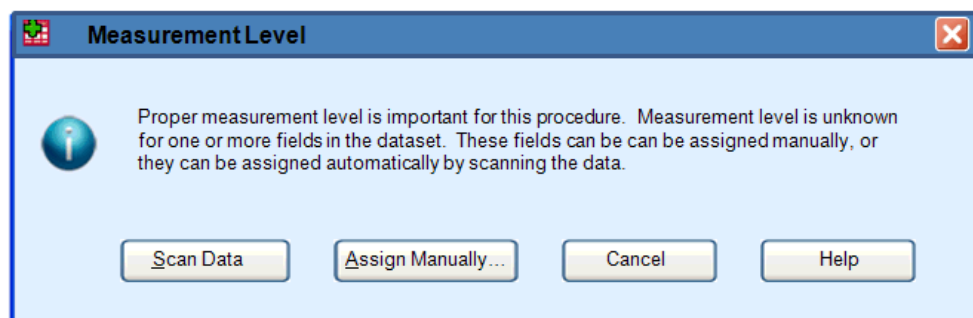
- **Scan Data.** Reads the data in the active dataset and assigns default measurement level to any fields with a currently unknown measurement level. If the dataset is large, that may take some time.
- **Assign Manually.** Opens a dialog that lists all fields with an unknown measurement level. You can use this dialog to assign measurement level to those fields. You can also assign measurement level in Variable View of the Data Editor.

Since measurement level is important for this procedure, you cannot access the dialog to run this procedure until all fields have a defined measurement level.

Partitions

Figure 3-3
Radial Basis Function: Partitions tab

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.

■ **Use partitioning variable to assign cases.** Specify a numeric variable that assigns each case in the active dataset to the training, testing, or holdout sample. Cases with a positive value on the variable are assigned to the training sample, cases with a value of 0, to the testing sample, and cases with a negative value, to the holdout sample. Cases with a system-missing value are excluded from the analysis. Any user-missing values for the partition variable are always treated as valid.

Architecture

Figure 3-4
Radial Basis Function: Architecture tab

The Architecture tab is used to specify the structure of the network. The procedure creates a neural network with one hidden "radial basis function" layer; in general, it will not be necessary to change these settings.

Number of Units in Hidden Layer. There are three ways of choosing the number of hidden units.

1. **Find the best number of units within an automatically computed range.** The procedure automatically computes the minimum and maximum values of the range and finds the best number of hidden units within the range.

    If a testing sample is defined, then the procedure uses the testing data criterion: The best number of hidden units is the one that yields the smallest error in the testing data. If a testing sample is not defined, then the procedure uses the Bayesian information criterion (BIC): The best number of hidden units is the one that yields the smallest BIC based on the training data.

2. **Find the best number of units within a specified range.** You can provide your own range, and the procedure will find the "best" number of hidden units within that range. As before, the best number of hidden units from the range is determined using the testing data criterion or the BIC.

3. **Use a specified number of units.** You can override the use of a range and specify a particular number of units directly.

    Activation Function for Hidden Layer. The activation function for the hidden layer is the radial basis function, which "links" the units in a layer to the values of units in the succeeding layer. For the output layer, the activation function is the identity function; thus, the output units are simply weighted sums of the hidden units.

    - **Normalized radial basis function.** Uses the softmax activation function so the activations of all hidden units are normalized to sum to 1.
    - **Ordinary radial basis function.** Uses the exponential activation function so the activation of the hidden unit is a Gaussian "bump" as a function of the inputs.

    Overlap Among Hidden Units. The overlapping factor is a multiplier applied to the width of the radial basis functions. The automatically computed value of the overlapping factor is $1+0.1d$, where d is the number of input units (the sum of the number of categories across all factors and the number of covariates).

Output

Figure 3-5
Radial Basis Function: Output tab

Network Structure. Displays summary information about the neural network.

- **Description.** Displays 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.** Displays 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 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 the training sample if there is no testing sample.

■ **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, and the training time.

The error is the sum-of-squares error. In addition, 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.

■ **Classification results.** Displays a classification table for each categorical dependent variable. 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.

■ **ROC curve.** Displays an ROC (Receiver Operating Characteristic) curve 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.

■ **Cumulative gains 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.

■ **Lift chart.** Displays a lift chart for each categorical dependent variable. The display of one curve for each dependent variable category is the same as for ROC curves.

■ **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 are displayed for each response category, with the observed response category as the cluster variable. For scale dependent variables, a scatterplot is displayed.

■ **Residual by predicted chart.** Displays a residual-by-predicted-value chart for each scale dependent variable. There should be no visible patterns between residuals and predicted values. This chart is produced only for scale dependent variables.

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.

Independent variable importance analysis. Performs a sensitivity analysis, which computes the importance of each predictor in determining the neural network. The analysis is based on the combined training and testing samples or only the training sample if there is no testing sample. This creates a table and a chart displaying importance and normalized importance for each

predictor. Note that sensitivity analysis is computationally expensive and time-consuming if there is a large number of predictors or cases.

Save

Figure 3-6
Radial Basis Function: Save tab
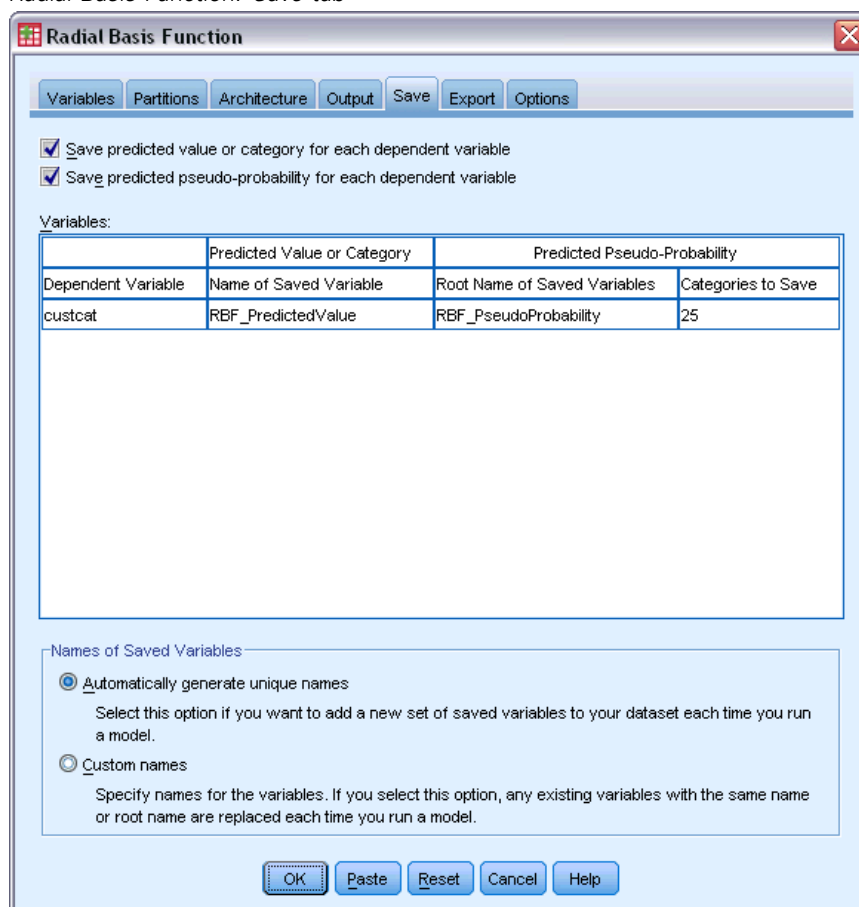
The Save tab is used to save predictions as variables in the dataset.

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

- **Save predicted pseudo-probability for each dependent variable.** This 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.

Names of Saved Variables. Automatic name generation ensures that you keep all of your work. Custom names allow you to discard or replace results from previous runs without first deleting the saved variables in the Data Editor.

Probabilities and Pseudo-Probabilities

Predicted pseudo-probabilities cannot be interpreted as probabilities because the Radial Basis Function procedure uses the sum-of-squares error and identity activation function for the output layer. 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 (see Output on p. 29) 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, .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.

Export

Figure 3-7
Radial Basis Function: Export tab
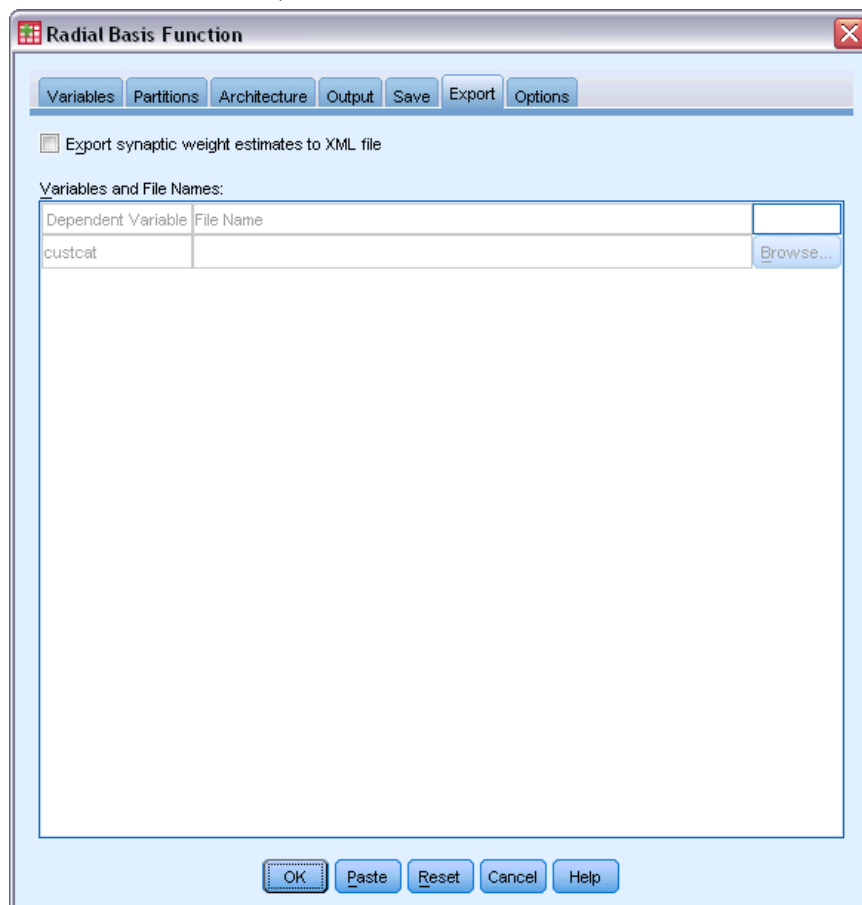
The Export tab is used to save the synaptic weight estimates for each dependent variable to an XML (PMML) file. You can use this model file to apply the model information to other data files for scoring purposes. This option is not available if split files have been defined.

Options

Figure 3-8
Radial Basis Function: Options tab
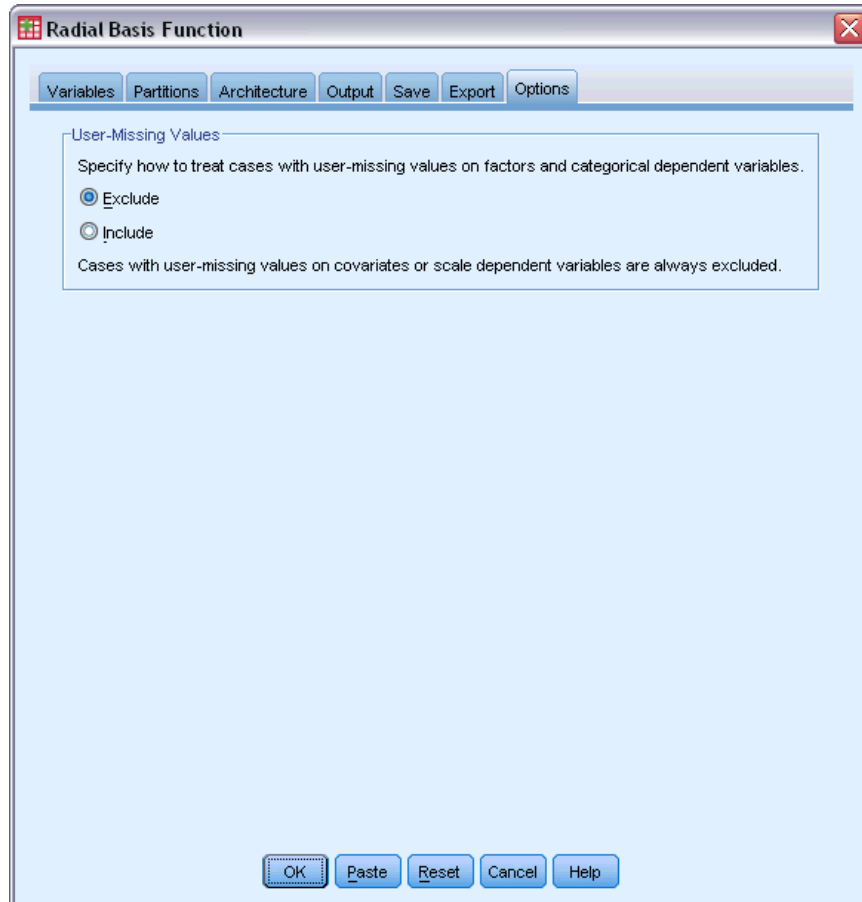
User-Missing Values. Factors must have valid values for a case to be included in the analysis. These controls allow you to decide whether user-missing values are treated as valid among factors and categorical dependent variables.

Part II:
Examples

Multilayer Perceptron

The Multilayer Perceptron (MLP) procedure produces a predictive model for one or more dependent (target) variables based on values of the predictor variables.

Using a Multilayer Perceptron to Assess Credit Risk

A loan officer at a bank needs to be able to identify characteristics that are indicative of people who are likely to default on loans and use those characteristics to identify good and bad credit risks.

   Suppose that information on 850 past and prospective customers is contained in *bankloan.sav*. For more information, see the topic Sample Files in Appendix A on p. 84. The first 700 cases are customers who were previously given loans. Use a random sample of these 700 customers to create a multilayer perceptron, setting the remaining customers aside to validate the analysis. Then use the model to classify the 150 prospective customers as good or bad credit risks.

   Additionally, the loan officer has previously analyzed the data using logistic regression (in the Regression option) and wonders how the multilayer perceptron compares as a classification tool.

Preparing the Data for Analysis

Setting the random seed allows you to replicate the analysis exactly.

▶ To set the random seed, from the menus choose:
Transform > Random Number Generators...

Figure 4-1
Random Number Generators dialog box
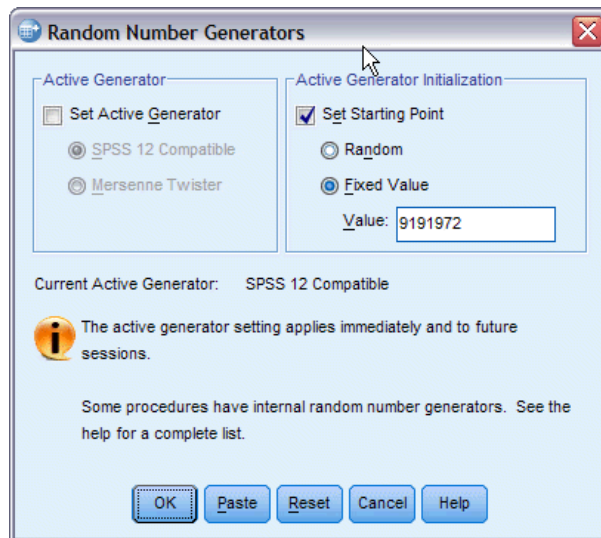
▶ Select Set Starting Point.

▶ Select Fixed Value, and type **9191972** as the value.

▶ Click OK.

In the previous logistic regression analysis, approximately 70% of the past customers were assigned to the training sample and 30% to a holdout sample. A partition variable will be necessary to exactly recreate the samples used in those analyses.

▶ To create the partition variable, from the menus choose:
Transform > Compute Variable...

Figure 4-2
Compute Variable dialog box
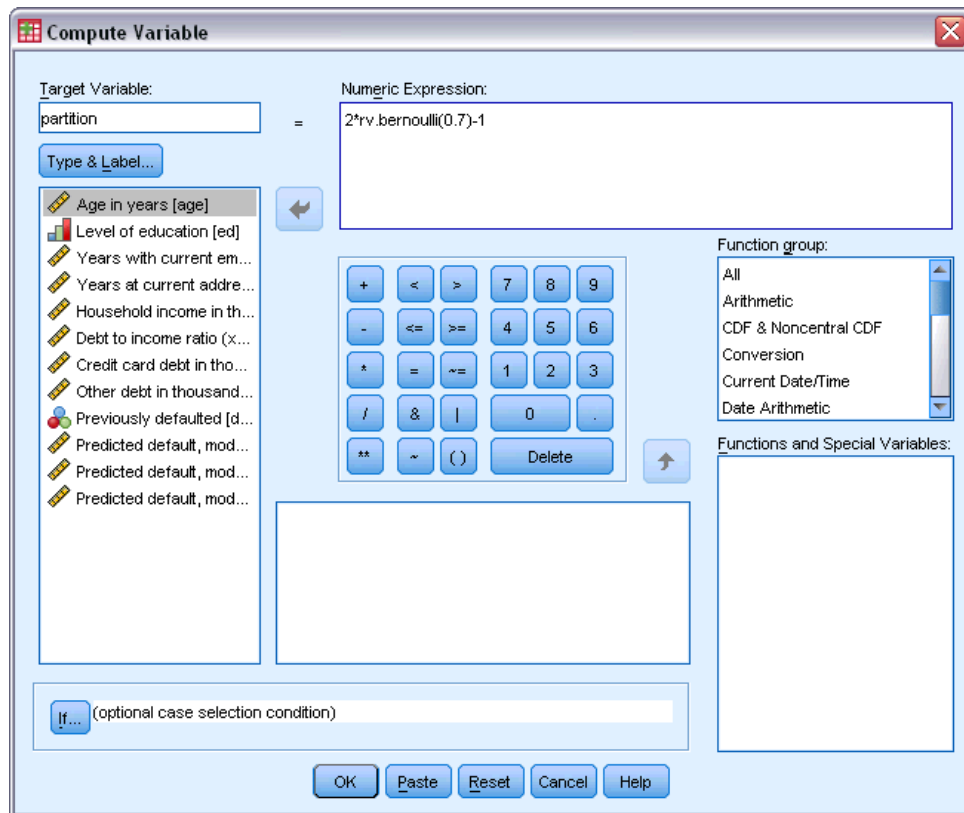
▶ Type partition in the Target Variable text box.

▶ Type 2*rv.bernoulli(0.7)-1 in the Numeric Expression text box.

This sets the values of *partition* to be randomly generated **Bernoulli** variates with a probability parameter of 0.7, modified so that it takes values 1 or −1, instead of 1 or 0. Recall that cases with positive values on the partition variable are assigned to the training sample, cases with negative values are assigned to the holdout sample, and cases with a value of 0 are assigned to the testing sample. For now, we won't specify a testing sample.

▶ Click OK in the Compute Variable dialog box.

Approximately 70% of the customers previously given loans will have a *partition* value of 1. These customers will be used to create the model. The remaining customers who were previously given loans will have a *partition* value of −1 and will be used to validate the model results.

Running the Analysis

▶ To run a Multilayer Perceptron analysis, from the menus choose:
Analyze > Neural Networks > Multilayer Perceptron...

Figure 4-3
Multilayer Perceptron: Variables tab
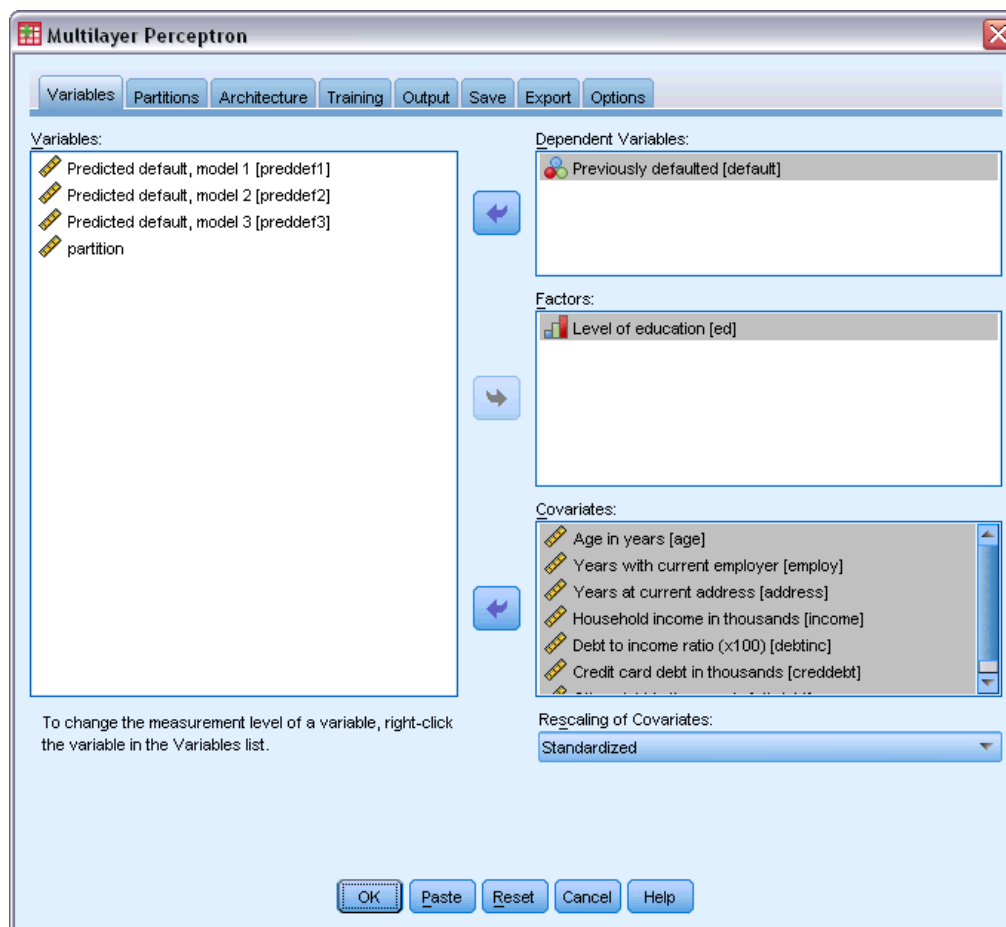
▶ Select *Previously defaulted [default]* as a dependent variable.

▶ Select *Level of education [ed]* as a factor.

▶ Select *Age in years [age]* through *Other debt in thousands [othdebt]* as covariates.

▶ Click the Partitions tab.

Figure 4-4
Multilayer Perceptron: Partitions tab
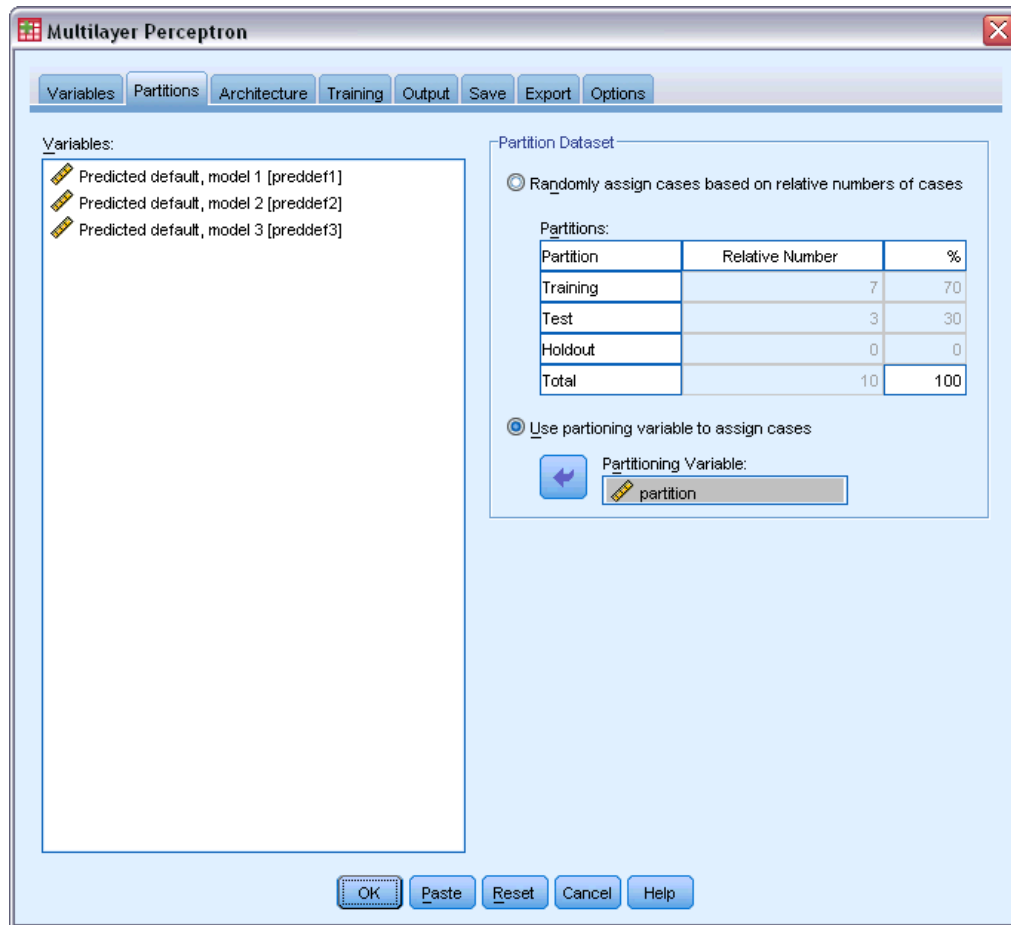
▶ Select Use partitioning variable to assign cases.

▶ Select *partition* as the partitioning variable.

▶ Click the Output tab.

Figure 4-5
Multilayer Perceptron: Output tab
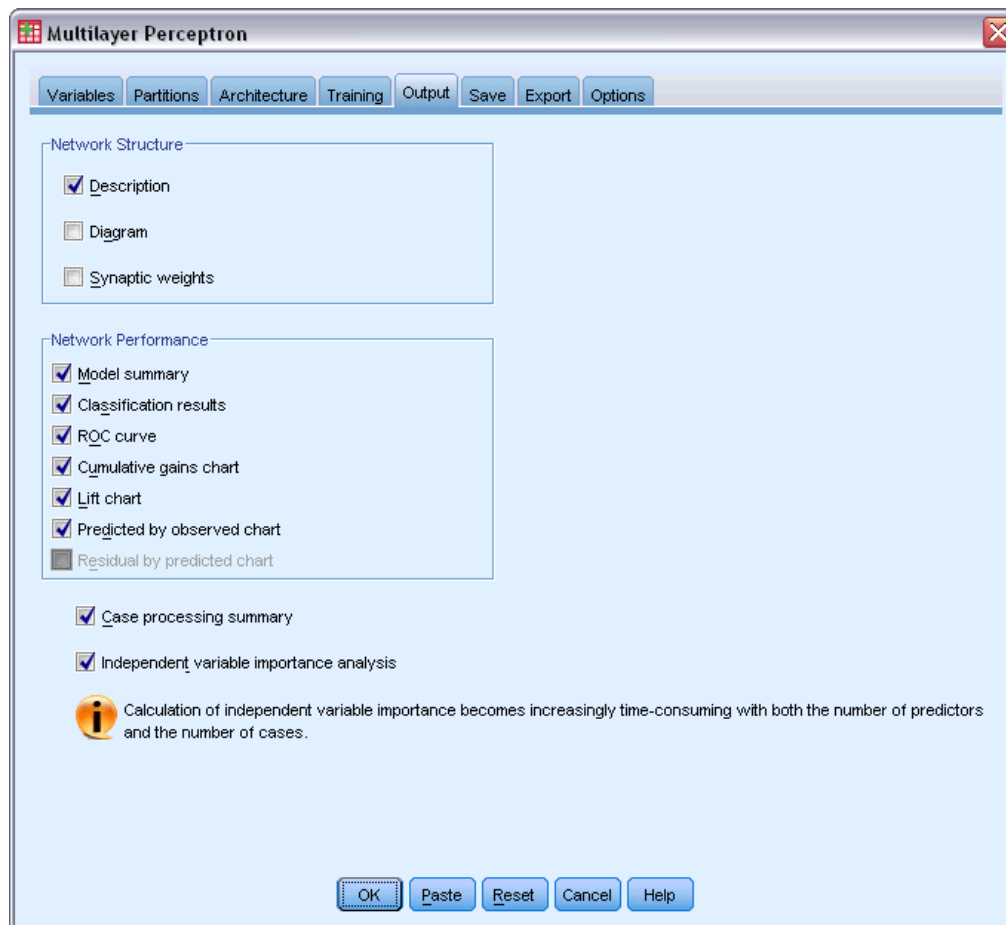
▶ Deselect Diagram in the Network Structure group.

▶ Select ROC curve, Cumulative gains chart, Lift chart, and Predicted by observed chart in the Network Performance group. The Residual by predicted chart is unavailable because the dependent variable is not scale.

▶ Select Independent variable importance analysis.

▶ Click OK.

Case Processing Summary

Figure 4-6
Case processing summary

		N	Percent
Sample	Training	499	71.3%
	Holdout	201	28.7%
Valid		700	100.0%
Excluded		150	
Total		850	

The case processing summary shows that 499 cases were assigned to the training sample and 201 to the holdout sample. The 150 cases excluded from the analysis are the prospective customers.

Network Information

Figure 4-7
Network information

Input Layer	Factors	1	Level of education	
	Covariates	1	Age in years	
		2	Years with current employer	
		3	Years at current address	
		4	Household income in …	
		5	Debt to income ratio (x100)	
		6	Credit card debt in thousands	
		7	Other debt in thousands	
	Number of Units[a]			12
	Rescaling Method for Covariates		Standardized	
Hidden Layer(s)	Number of Hidden Layers			1
	Number of Units in Hidden Layer 1[a]			4
	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

The network information table displays information about the neural network and is useful for ensuring that the specifications are correct. Here, note in particular that:

■ The number of units in the input layer is the number of covariates plus the total number of factor levels; a separate unit is created for each category of *Level of education* and none of the categories are considered "redundant" units as is typical in many modeling procedures.

■ Likewise, a separate output unit is created for each category of *Previously defaulted*, for a total of two units in the output layer.

■ Automatic architecture selection has chosen four units in the hidden layer.

■ All other network information is default for the procedure.

Model Summary

Figure 4-8
Model summary

Training	Cross Entropy Error	156.606
	Percent Incorrect Predictions	15.6%
	Stopping Rule Used	Maximum number of epochs (100) exceeded
	Training Time	00:00:00.081
Holdout	Percent Incorrect Predictions	25.4%

Dependent Variable: Previously defaulted

The model summary displays information about the results of training and applying the final network to the holdout sample.

■ Cross entropy error is displayed because the output layer uses the softmax activation function. This is the error function that the network tries to minimize during training.

■ The percentage of incorrect predictions is taken from the classification table and will be discussed further in that topic.

■ The estimation algorithm stopped because the maximum number of epochs was reached. Ideally, training should stop because the error has converged. This raises questions about whether something went wrong during training and is something to keep in mind while further inspecting the output.

Classification

Figure 4-9
Classification

		Predicted		
Sample	Observed	No	Yes	Percent Correct
Training	No	347	28	92.5%
	Yes	50	74	59.7%
	Overall Percent	79.6%	20.4%	84.4%
Holdout	No	123	19	86.6%
	Yes	32	27	45.8%
	Overall Percent	77.1%	22.9%	74.6%

Dependent Variable: Previously defaulted

The classification table shows the practical results of using the network. For each case, the predicted response is *Yes* if that cases's predicted pseudo-probability is greater than 0.5. For each sample:

■ Cells on the diagonal of the cross-classification of cases are correct predictions.

■ Cells off the diagonal of the cross-classification of cases are incorrect predictions.

Of the cases used to create the model, 74 of the 124 people who previously defaulted are classified correctly. 347 of the 375 non-defaulters are classified correctly. Overall, 84.4% of the training cases are classified correctly, corresponding to the 15.6% incorrect shown in the model summary table. A better model should correctly identify a higher percentage of the cases.

Classifications based upon the cases used to create the model tend to be too "optimistic" in the sense that their classification rate is inflated. The holdout sample helps to validate the model; here 74.6% of these cases were correctly classified by the model. This suggests that, overall, your model is in fact correct about three out of four times.

Correcting Overtraining

Thinking back on the logistic regression analysis previously performed, the loan officer recalls that the training and holdout samples correctly predicted a similar percentage of cases, about 80%. By contrast, the neural network had a higher percentage of correct cases in the training sample, with the holdout sample doing a considerably worse job of predicting customers that actually defaulted (45.8% correct for the holdout sample versus 59.7% for the training sample). Combined with the stopping rule reported in the model summary table, this makes you suspect that the network may be **overtraining**; that is, it is chasing spurious patterns that appear in the training data by random variation.

Fortunately, the solution is relatively simple: specify a testing sample to help keep the network "on track." We created the partition variable so that it would exactly recreate the training and holdout samples used in the logistic regression analysis; however, logistic regression has no concept of a "testing" sample. Let's take a portion of the training sample and reassign it to a testing sample.

Creating the Testing Sample

Figure 4-10
Compute Variable dialog box
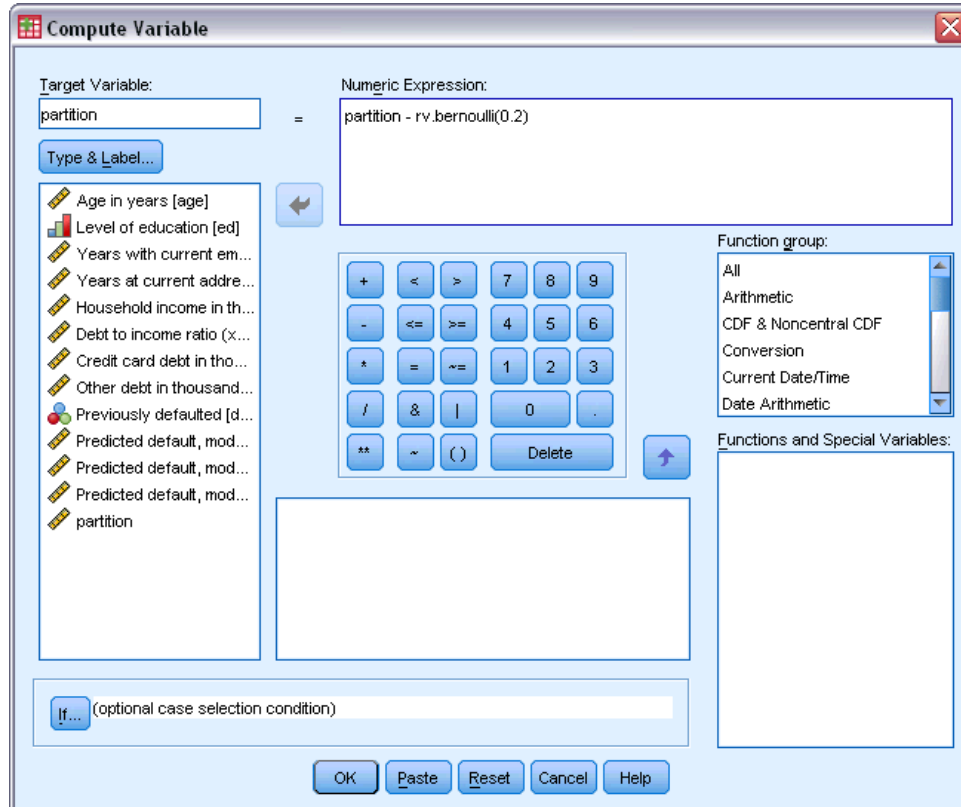
▶ Recall the Compute Variable dialog box.

▶ Type partition - rv.bernoulli(0.2) in the Numeric Expression text box.

▶ Click If.

Figure 4-11
Compute Variable: If Cases dialog box
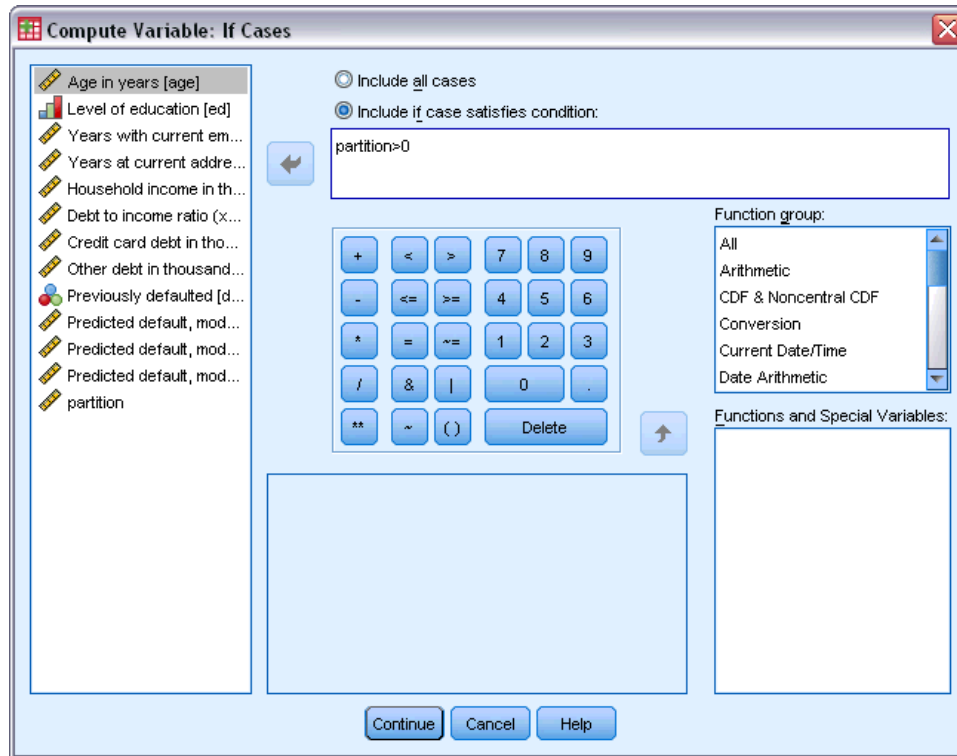
▶ Select Include if case satisfies condition.

▶ Type partition>0 in the text box.

▶ Click Continue.

▶ Click OK in the Compute Variable dialog box.

This resets the values of *partition* that were greater than 0 so that approximately 20% take the value 0, and 80% remain valued at 1. Overall, approximately $100*(0.7*0.8)=56\%$ of the customers previously given loans will be in the training sample, and 14% will be in the testing sample. Customers who were originally assigned to the holdout sample remain there.

Running the Analysis

▶ Recall the Multilayer Perceptron dialog box and click the Save tab.

▶ Select Save predicted pseudo-probability for each dependent variable.

▶ Click OK.

Case Processing Summary

Figure 4-12
Case processing summary for model with testing sample

		N	Percent
Sample	Training	398	56.9%
	Testing	101	14.4%
	Holdout	201	28.7%
Valid		700	100.0%
Excluded		150	
Total		850	

Of the 499 cases originally assigned to the training sample, 101 have been reassigned to the testing sample.

Network Information

Figure 4-13
Network information

Input Layer	Factors	1	Level of education
	Covariates	1	Age in years
		2	Years with current employer
		3	Years at current address
		4	Household income in …
		5	Debt to income ratio (x100)
		6	Credit card debt in thousands
		7	Other debt in thousands
	Number of Units[a]		12
	Rescaling Method for Covariates		Standardized
Hidden Layer(s)	Number of Hidden Layers		1
	Number of Units in Hidden Layer 1[a]		7
	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

The only change to the network information table is that the automatic architecture selection has chosen seven units in the hidden layer.

Model Summary

Figure 4-14
Model summary

Training	Cross Entropy Error	159.870
	Percent Incorrect Predictions	20.1%
	Stopping Rule Used	1 consecutive step (s) with no decrease in error[a]
	Training Time	00:00:01.013
Testing	Cross Entropy Error	40.068
	Percent Incorrect Predictions	17.8%
Holdout	Percent Incorrect Predictions	20.4%

Dependent Variable: Previously defaulted

a.Error computations are based on the testing sample.

The model summary shows a couple of positive signs:

- The percentage of incorrect predictions is roughly equal across training, testing, and holdout samples.

- The estimation algorithm stopped because the error did not decrease after a step in the algorithm.

This further suggests that the original model may have, in fact, overtrained and that the problem was solved by adding a testing sample. Of course, the sample sizes are relatively small, and perhaps we should not read too much into the swing of a few percentage points.

Classification

Figure 4-15
Classification

		Predicted		
Sample	Observed	No	Yes	Percent Correct
Training	No	263	34	88.6%
	Yes	46	55	54.5%
	Overall Percent	77.6%	22.4%	79.9%
Testing	No	73	5	93.6%
	Yes	13	10	43.5%
	Overall Percent	85.1%	14.9%	82.2%
Holdout	No	124	18	87.3%
	Yes	23	36	61.0%
	Overall Percent	73.1%	26.9%	79.6%

Dependent Variable: Previously defaulted

The classification table shows that, using 0.5 as the pseudo-probability cutoff for classification, the network does considerably better at predicting non-defaulters than defaulters. Unfortunately, the single cutoff value gives you a very limited view of the predictive ability of the network, so it is not necessarily very useful for comparing competing networks. Instead, look at the ROC curve.

ROC Curve

Figure 4-16
ROC curve
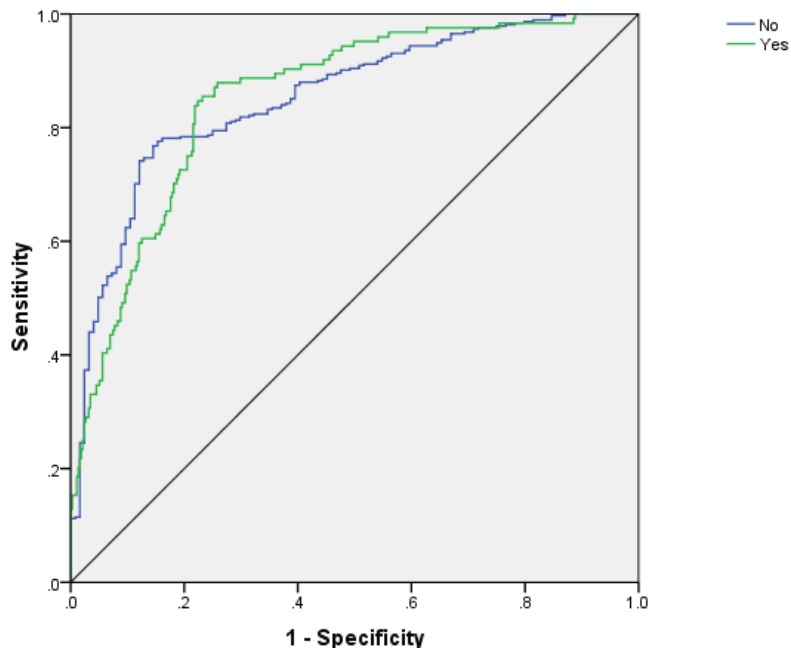
Dependent Variable: Previously defaulted

The ROC curve gives you a visual display of the **sensitivity** and **specificity** for all possible cutoffs in a single plot, which is much cleaner and more powerful than a series of tables. The chart shown here displays two curves, one for the category *No* and one for the category *Yes*. Since there are only two categories, the curves are symmetrical about a 45-degree line (not displayed) from the upper left corner of the chart to the lower right.

   Note that this chart is based on the combined training and testing samples. To produce an ROC chart for the holdout sample, split the file on the partition variable and run the ROC Curve procedure on the saved predicted pseudo-probabilities.

Figure 4-17
Area under the curve

		Area
Previously defaulted	No	.853
	Yes	.853

The area under the curve is a numerical summary of the ROC curve, and the values in the table represent, for each category, the probability that the predicted pseudo-probability of being in that category is higher for a randomly chosen case in that category than for a randomly chosen case not in that category. For example, for a randomly selected defaulter and randomly selected non-defaulter, there is a 0.853 probability that the model-predicted pseudo-probability of default will be higher for the defaulter than for the non-defaulter.

While the area under the curve is a useful one-statistic summary of the accuracy of the network, you need to be able to choose a specific criterion by which customers are classified. The predicted-by-observed chart provides a visual start on this process.

Predicted-by-Observed Chart

Figure 4-18
Predicted-by-observed chart
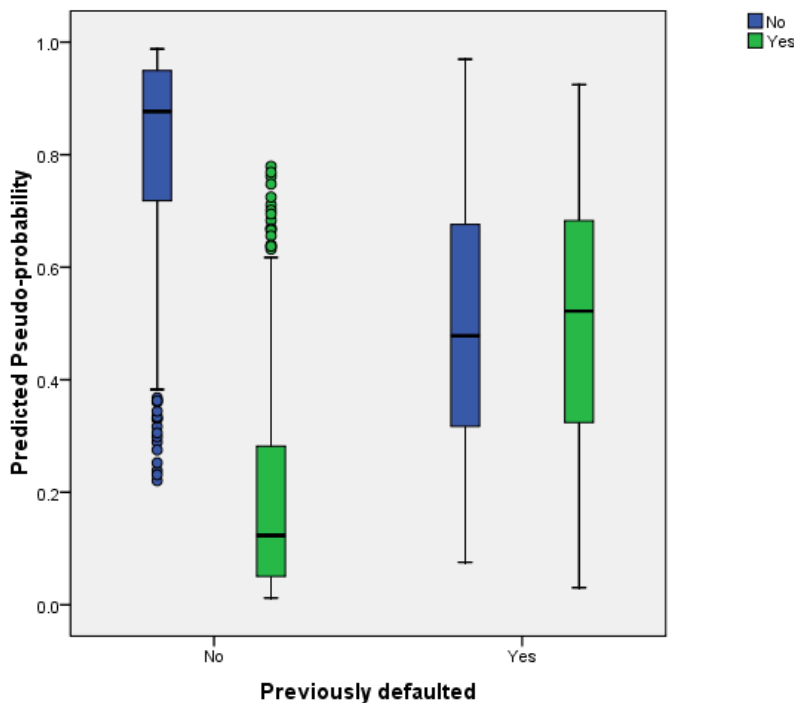
For categorical dependent variables, the predicted-by-observed chart displays clustered boxplots of predicted pseudo-probabilities for the combined training and testing samples. The *x* axis corresponds to the observed response categories, and the legend corresponds to predicted categories.

- The leftmost boxplot shows, for cases that have observed category No, the predicted pseudo-probability of category No. The portion of the boxplot above the 0.5 mark on the *y* axis represents correct predictions shown in the classification table. The portion below the 0.5 mark represents incorrect predictions. Remember from the classification table that the network is very good at predicting cases with the No category using the 0.5 cutoff, so only a portion of the lower whisker and some outlying cases are misclassified.

- The next boxplot to the right shows, for cases that have observed category No, the predicted pseudo-probability of category Yes. Since there are only two categories in the target variable, the first two boxplots are symmetrical about the horizontal line at 0.5.

- The third boxplot shows, for cases that have observed category Yes, the predicted pseudo-probability of category No. It and the last boxplot are symmetrical about the horizontal line at 0.5.

- The last boxplot shows, for cases that have observed category Yes, the predicted pseudo-probability of category Yes. The portion of the boxplot above the 0.5 mark on the *y* axis represents correct predictions shown in the classification table. The portion below the 0.5 mark represents incorrect predictions. Remember from the classification table that the network predicts slightly more than half of the cases with the Yes category using the 0.5 cutoff, so a good portion of the box is misclassified.

Looking at the plot, it appears that by lowering the cutoff for classifying a case as Yes from 0.5 to approximately 0.3—this is roughly the value where the top of the second box and the bottom of the fourth box are—you can increase the chance of correctly catching prospective defaulters without losing many potential good customers. That is, moving from 0.5 to 0.3 along the second box incorrectly reclassifies relatively few non-defaulting customers along the whisker as predicted defaulters, while along the fourth box, this move correctly reclassifies many defaulting customers within the box as predicted defaulters.

Cumulative Gains and Lift Charts

Figure 4-19
Cumulative gains chart
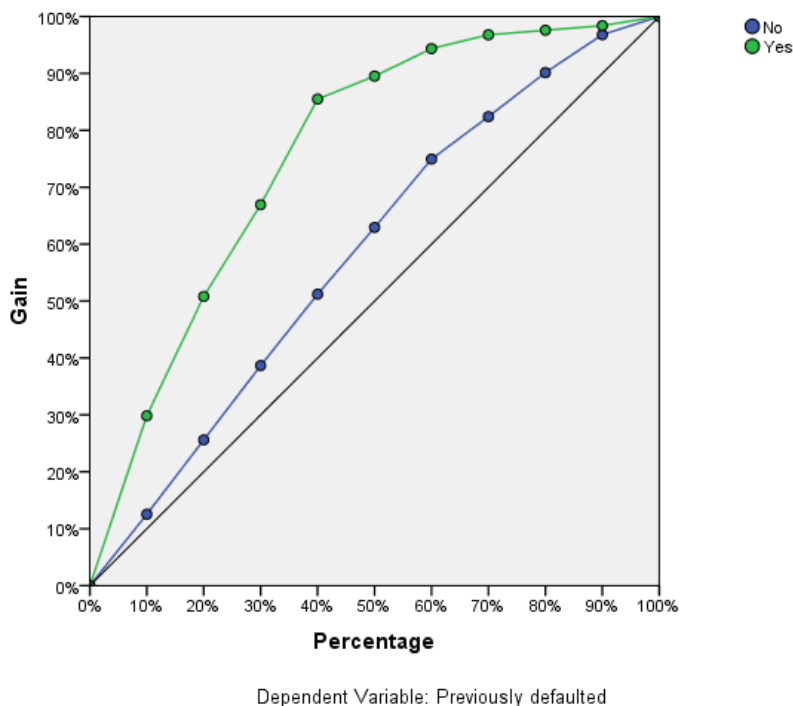
Dependent Variable: Previously defaulted

The cumulative gains chart shows the percentage of the overall number of cases in a given category "gained" by targeting a percentage of the total number of cases. For example, the first point on the curve for the *Yes* category is at (10%, 30%), meaning that if you score a dataset with the network and sort all of the cases by predicted pseudo-probability of *Yes*, you would expect the top 10% to contain approximately 30% of all of the cases that actually take the category *Yes* (defaulters). Likewise, the top 20% would contain approximately 50% of the defaulters, the top 30% of cases would contain 70% of defaulters, and so on. If you select 100% of the scored dataset, you obtain all of the defaulters in the dataset.

The diagonal line is the "baseline" curve; if you select 10% of the cases from the scored dataset at random, you would expect to "gain" approximately 10% of all of the cases that actually take the category Yes. The farther above the baseline a curve lies, the greater the gain. You can use the cumulative gains chart to help choose a classification cutoff by choosing a percentage that corresponds to a desirable gain, and then mapping that percentage to the appropriate cutoff value.

What constitutes a "desirable" gain depends on the cost of Type I and Type II errors. That is, what is the cost of classifying a defaulter as a non-defaulter (Type I)? What is the cost of classifying a non-defaulter as a defaulter (Type II)? If bad debt is the primary concern, then you want to lower your Type I error; on the cumulative gains chart, this might correspond to rejecting loans to applicants in the top 40% of pseudo-predicted probability of *Yes*, which captures nearly 90% of the possible defaulters but removes nearly half of your applicant pool. If growing your customer base is the priority, then you want to lower your Type II error. On the chart, this might correspond to rejecting the top 10%, which captures 30% of the defaulters and leaves most of your applicant pool intact. Usually, both are major concerns, so you have to choose a decision rule for classifying customers that gives the best mix of sensitivity and specificity.

Figure 4-20
Lift chart
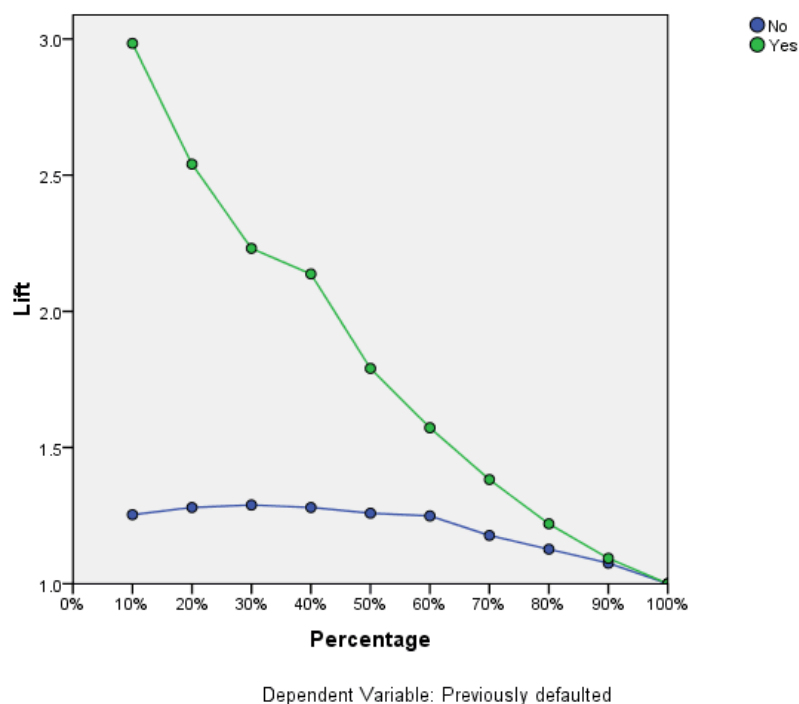
Dependent Variable: Previously defaulted

The lift chart is derived from the cumulative gains chart; the values on the *y* axis correspond to the ratio of the cumulative gain for each curve to the baseline. Thus, the lift at 10% for the category Yes is 30%/10% = 3.0. It provides another way of looking at the information in the cumulative gains chart.

Note: The cumulative gains and lift charts are based on the combined training and testing samples.

Independent Variable Importance

Figure 4-21
Independent variable importance

	Importance	Normalized Importance
Level of education	.032	11.9%
Age in years	.075	27.9%
Years with current employer	.268	100.0%
Years at current address	.166	61.8%
Household income in thousands	.033	12.2%
Debt to income ratio (x100)	.125	46.5%
Credit card debt in thousands	.213	79.3%
Other debt in thousands	.090	33.6%

The importance of an independent variable is a measure of how much the network's model-predicted value changes for different values of the independent variable. Normalized importance is simply the importance values divided by the largest importance values and expressed as percentages.

Figure 4-22
Independent variable importance chart
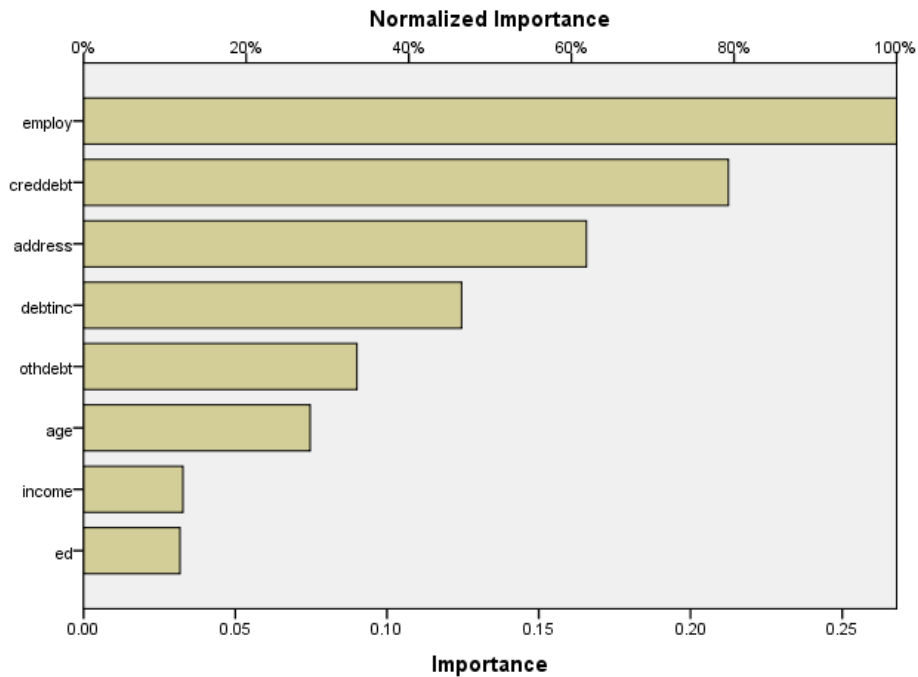
The importance chart is simply a bar chart of the values in the importance table, sorted in descending value of importance. It appears that variables related to a customer's stability (*employ*, *address*) and debt (*creddebt*, *debtinc*) have the greatest effect on how the network classifies customers; what you cannot tell is the "direction" of the relationship between these variables and the predicted probability of default. You would guess that a larger amount of debt indicates a greater likelihood of default, but to be sure, you would need to use a model with more easily interpretable parameters.

Summary

Using the Multilayer Perceptron procedure, you have constructed a network for predicting the probability that a given customer will default on a loan. The model results are comparable to those obtained using Logistic Regression or Discriminant Analysis, so you can be reasonably confident that the data do not contain relationships that cannot be captured by those models; thus, you can use them to further explore the nature of the relationship between the dependent and independent variables.

Using a Multilayer Perceptron to Estimate Healthcare Costs and Lengths of Stay

A hospital system is interested in tracking costs and lengths of stay for patients admitted for treatment of myocardial infarction (MI, or "heart attack"). Obtaining accurate estimates of these measures allow the administration to properly manage the available bed space as patients are treated.

The data file *patient_los.sav* contains the treatment records of a sample of patients who received treatment for MI. For more information, see the topic Sample Files in Appendix A on p. 84. Use the Multilayer Perceptron procedure to build a network for predicting costs and length of stay.

Preparing the Data for Analysis

Setting the random seed allows you to replicate the analysis exactly.

▶ To set the random seed, from the menus choose:
Transform > Random Number Generators...

Figure 4-23
Random Number Generators dialog box
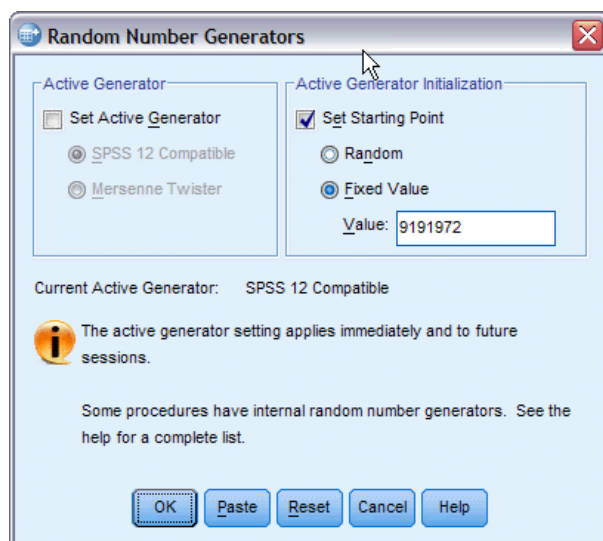
▶ Select Set Starting Point.

▶ Select Fixed Value, and type 9191972 as the value.

▶ Click OK.

Running the Analysis

▶ To run a Multilayer Perceptron analysis, from the menus choose:
Analyze > Neural Networks > Multilayer Perceptron...

Figure 4-24
Multilayer Perceptron: Variables tab and context menu for Length of stay
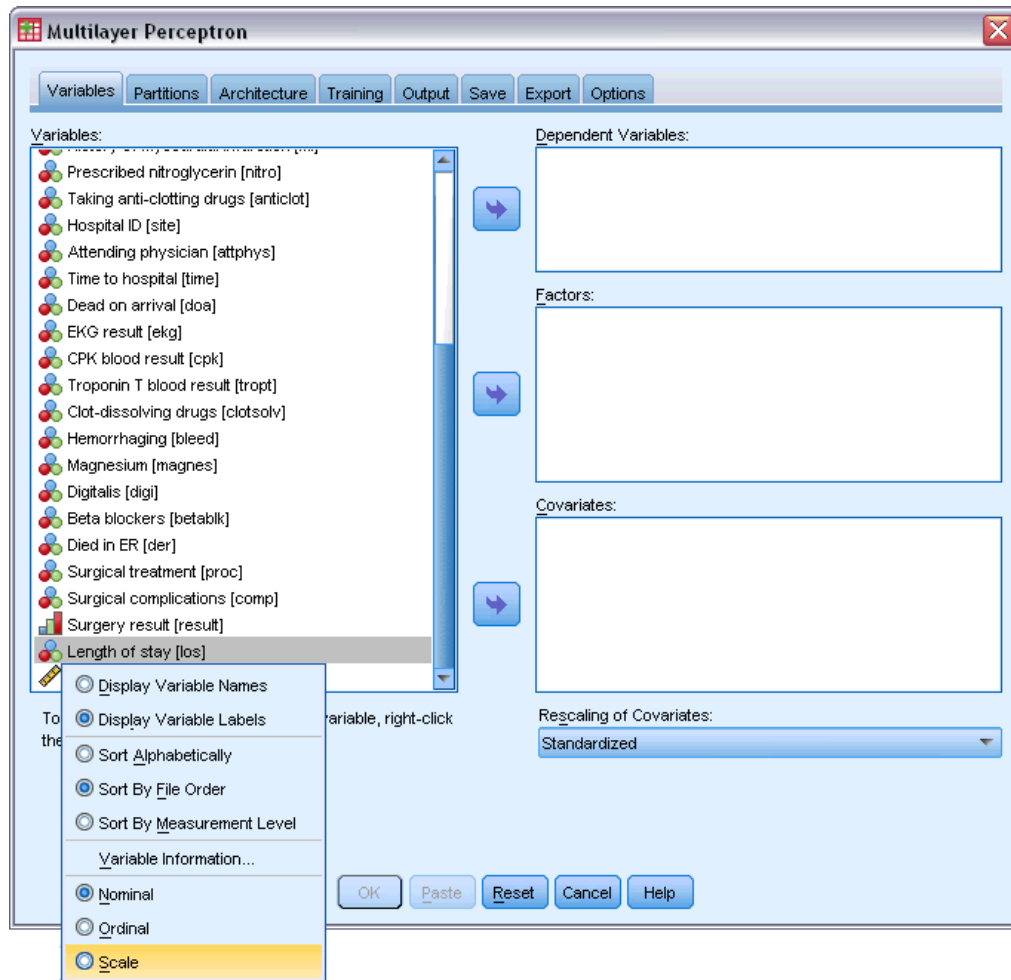
Length of stay [los] has an ordinal measurement level, but you want the network to treat it as scale.

▶ Right-click on *Length of stay [los]* and select Scale on the context menu.

Figure 4-25
Multilayer Perceptron: Variables tab with dependent variables and factors selected
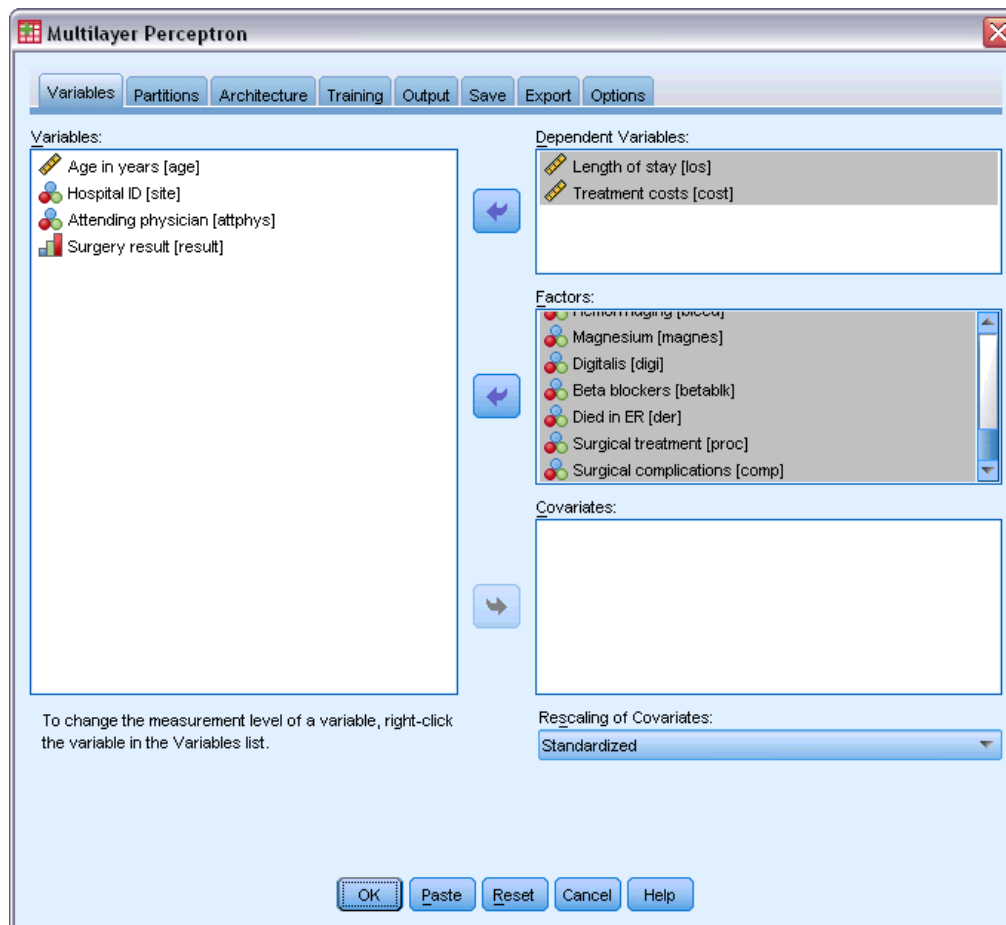
▶ Select *Length of stay [los]* and *Treatment costs [cost]* as dependent variables.

▶ Select *Age category [agecat]* through *Taking anti-clotting drugs [anticlot]* and *Time to hospital [time]* through *Surgical complications [comp]* as factors. To ensure exact replication of the model results below, be sure to maintain the order of the variables in the factor list. To this end, you may find it helpful to select each set of predictors and use the button to move them to the factor list, rather than use drap-and-drop. Alternatively, changing the order of variables helps you to assess the stability of the solution.

▶ Click the Partitions tab.

Figure 4-26
Multilayer Perceptron: Partitions tab
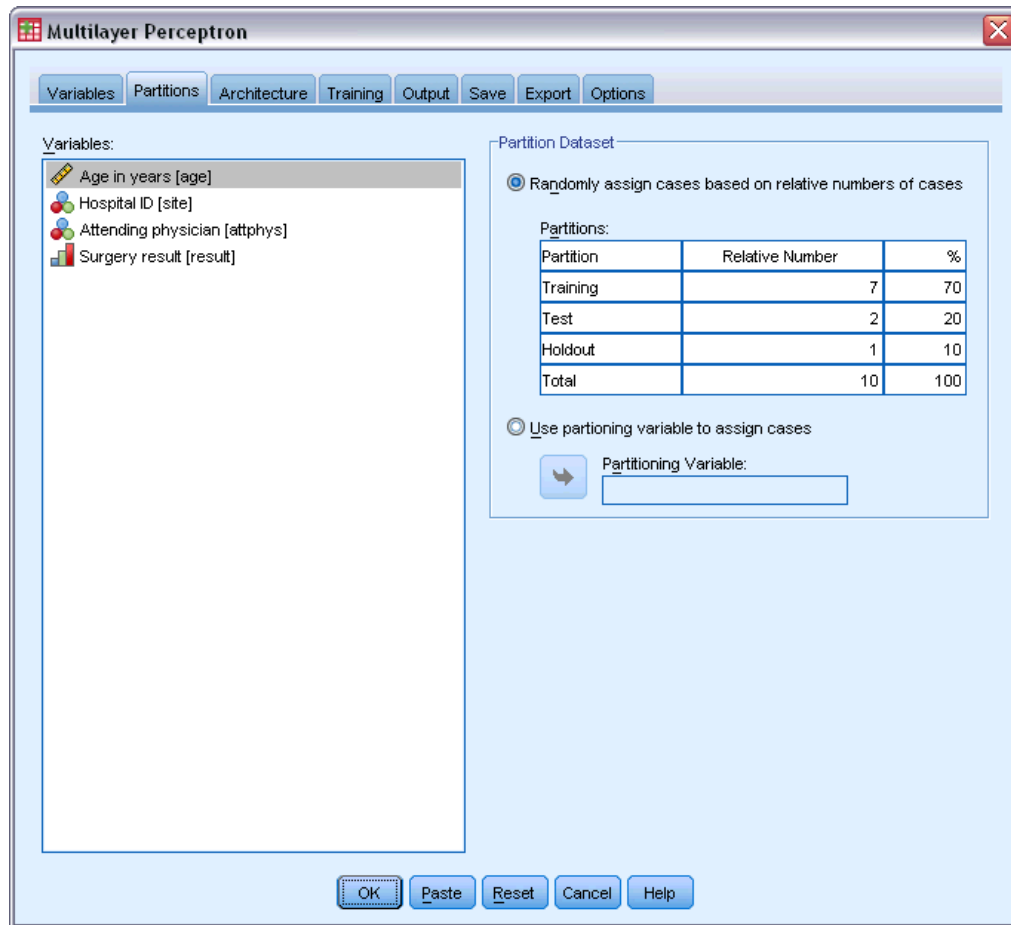
▶ Type 2 as the relative number of cases to assign to the testing sample.

▶ Type 1 as the relative number of cases to assign to the holdout sample.

▶ Click the Architecture tab.

Figure 4-27
Multilayer Perceptron: Architecture tab
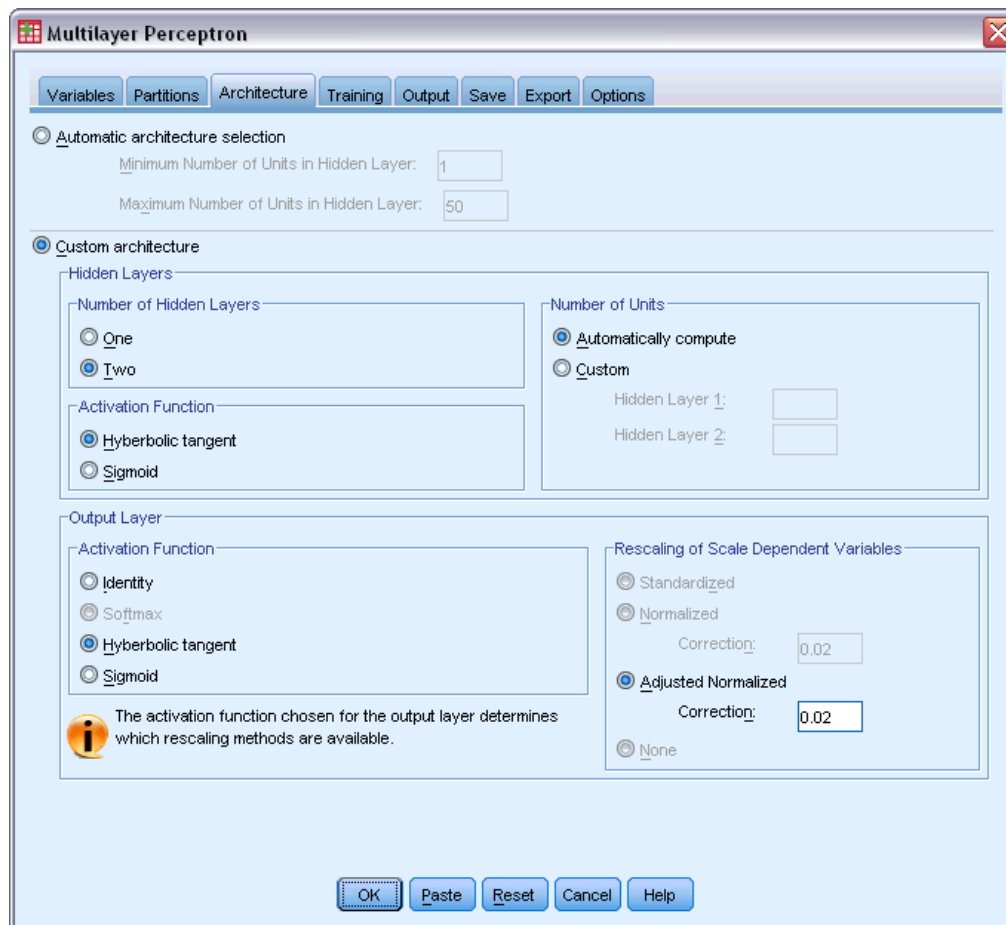
▶ Select Custom architecture.

▶ Select Two as the number of hidden layers.

▶ Select Hyperbolic tangent as the output layer activation function. Note that this automatically sets the rescaling method for the dependent variables to Adjusted Normalized.

▶ Click the Training tab.

Figure 4-28
Multilayer Perceptron: Training tab
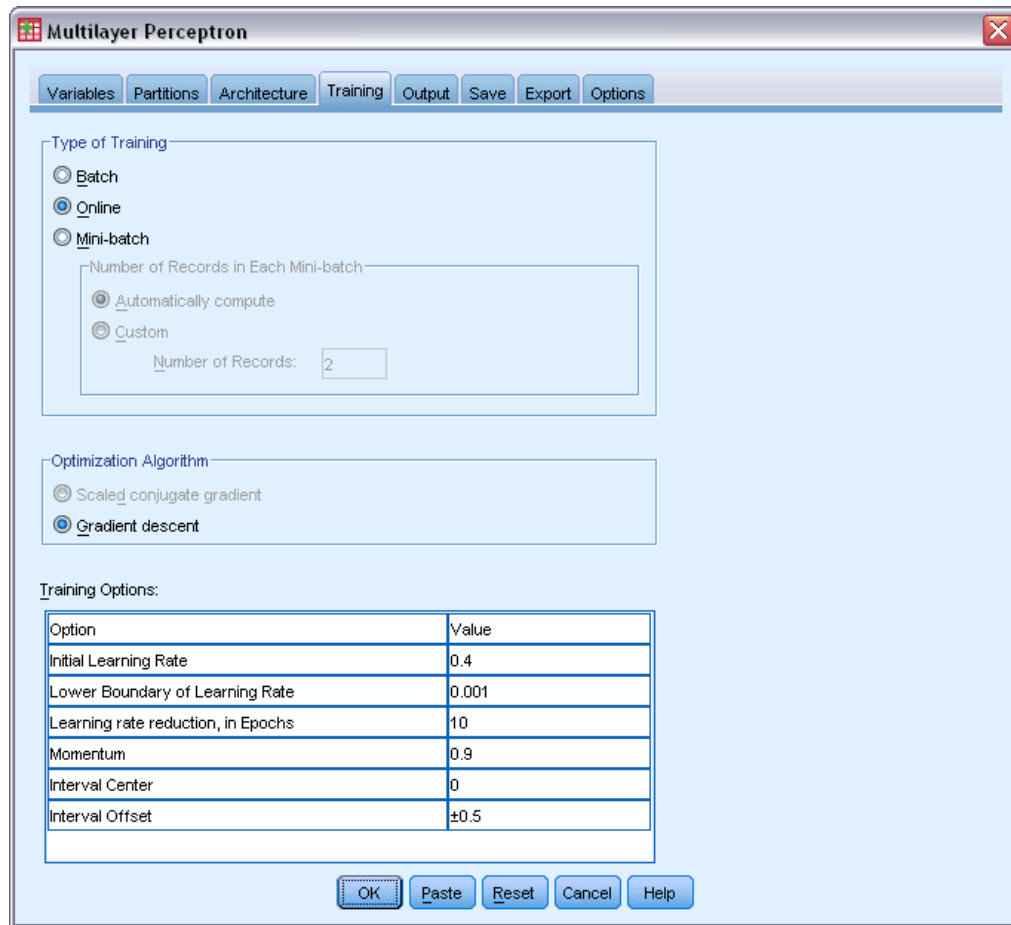
▶ Select Online as the type of training. Online training is supposed to perform well on "larger" datasets with correlated predictors. Note that this automatically sets Gradient descent as the optimization algorithm with the corresponding default options.

▶ Click the Output tab.

Figure 4-29
Multilayer Perceptron: Output tab
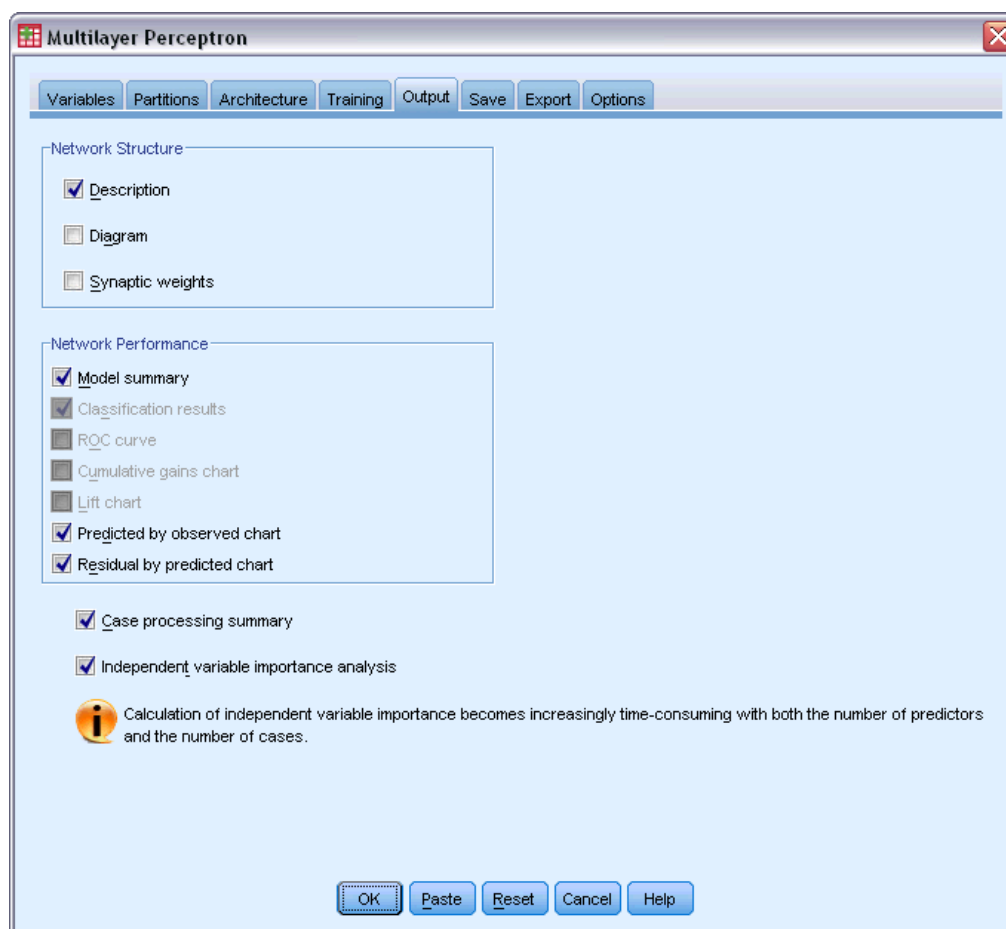
▶ Deselect Diagram; there are a lot of inputs, and the resulting diagram will be unwieldy.

▶ Select Predicted by observed chart and Residual by predicted chart in the Network Performance group. Classification results, the ROC curve, cumulative gains chart, and lift chart are not available because neither dependent variable is treated as categorical (nominal or ordinal).

▶ Select Independent variable importance analysis.

▶ Click the Options tab.

Figure 4-30
Options tab

► Choose to Include user-missing variables. Patients who did not have a surgical procedure have user-missing values on the *Surgical complications* variable. This ensures that those patients are included in the analysis.

► Click OK.

Warnings

Figure 4-31
Warnings

The warnings table notes that the variables *doa* and *der* are constant in the training sample. Patients who were dead on arrival or died in the emergency room have user-missing values on *Length of stay*. Since we are treating *Length of stay* as a scale variable for this analysis and

cases with user-missing values on scale variables are excluded, only patients who lived past the emergency room are included.

Case Processing Summary

Figure 4-32
Case processing summary

		N	Percent
Sample	Training	5647	70.6%
	Testing	1570	19.6%
	Holdout	781	9.8%
Valid		7998	100.0%
Excluded		2002	
Total		10000	

The case processing summary shows that 5647 cases were assigned to the training sample, 1570 to the testing sample, and 781 to the holdout sample. The 2002 cases excluded from the analysis are patients who died in transit to the hospital or in the emergency room.

Network Information

Figure 4-33
Network information

Input Layer	Factors	1	Age category	
		2	Gender	
		3	History of diabetes	
		4	Blood pressure	
		5	Smoker	
		6	Cholesterol	
		7	Physically active	
		8	Obesity	
		9	History of angina	
		10	History of myocardial infarction	
		11	Prescribed nitroglycerin	
		12	Taking anti-clotting drugs	
		13	Time to hospital	
		14	EKG result	
		15	CPK blood result	
		16	Troponin T blood result	
		17	Clot-dissolving drugs	
		18	Hemorrhaging	
		19	Magnesium	
		20	Digitalis	
		21	Beta blockers	
		22	Surgical treatment	
		23	Surgical complications	
	Number of Units[a]			63
Hidden Layer(s)	Number of Hidden Layers			2
	Number of Units in Hidden Layer 1[a]			12
	Number of Units in Hidden Layer 2[a]			9
	Activation Function		Hyperbolic tangent	
Output Layer	Dependent Variables	1	Length of stay	
		2	Treatment costs	
	Number of Units			2
	Rescaling Method for Scale Dependents		Adjusted Normalized	
	Activation Function		Hyperbolic tangent	
	Error Function		Sum of Squares	

a. Excluding the bias unit

The network information table displays information about the neural network and is useful for ensuring that the specifications are correct. Here, note in particular that:

■ The number of units in the input layer is the total number of factor levels (there are no covariates).

■ Two hidden layers were requested, and the procedure has chosen 12 units in the first hidden layer and 9 in the second.

■ A separate output unit is created for each of the scale-dependent variables. They are rescaled by the adjusted normalized method, which requires the use of the hyperbolic tangent activation function for the output layer.

■ Sum-of-squares error is reported because the dependent variables are scale.

Model Summary

Figure 4-34
Model summary

Training	Sum of Squares Error		91.812
	Average Overall Relative Error		.083
	Relative Error for Scale Dependents	Length of stay	.131
		Treatment costs	.033
	Stopping Rule Used		1 consecutive step (s) with no decrease in error[a]
	Training Time		00:00:18.055
Testing	Sum of Squares Error		26.798
	Average Overall Relative Error		.088
	Relative Error for Scale Dependents	Length of stay	.141
		Treatment costs	.033
Holdout	Average Overall Relative Error		.099
	Relative Error for Scale Dependents	Length of stay	.154
		Treatment costs	.041

a.Error computations are based on the testing sample.

The model summary displays information about the results of training and applying the final network to the holdout sample.

■ Sum-of-squares error is displayed because the output layer has scale-dependent variables. This is the error function that the network tries to minimize during training. Note that the sums of squares and all following error values are computed for the rescaled values of the dependent variables.

■ The relative error for each scale-dependent variable is the ratio of the sum-of-squares error for the dependent variable to the sum-of-squares error for the "null" model, in which the mean value of the dependent variable is used as the predicted value for each case. There appears to be more error in the predictions of *Length of stay* than in *Treatment costs*.

■ The average overall error is the ratio of the sum-of-squares error for all dependent variables to the sum-of-squares error for the "null" model, in which the mean values of the dependent variables are used as the predicted values for each case. In this example, the average overall error happens to be close to the average of the relative errors, but this will not always be the case.

The average overall relative error and relative errors are fairly constant across the training, testing, and holdout samples, which gives you some confidence that the model is not overtrained and that the error in future cases scored by the network will be close to the error reported in this table.

■ The estimation algorithm stopped because the error did not decrease after a step in the algorithm.

Predicted-by-Observed Charts

Figure 4-35
Predicted-by-observed chart for Length of stay
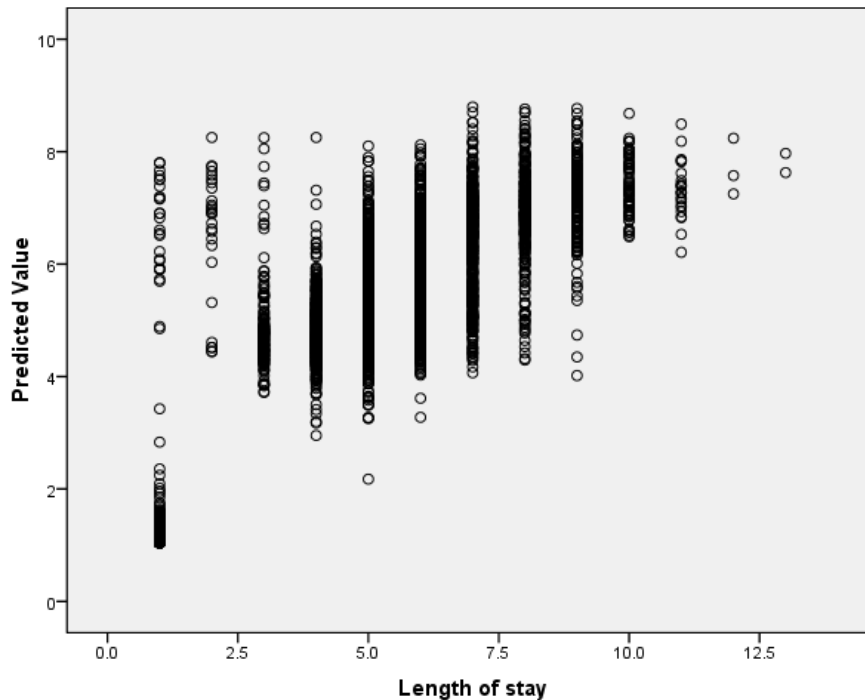
For scale-dependent variables, the predicted-by-observed chart displays a scatterplot of predicted values on the *y* axis by observed values on the *x* axis for the combined training and testing samples. Ideally, values should lie roughly along a 45-degree line starting at the origin. The points in this plot form vertical lines at each observed number of days of *Length of stay*.

Looking at the plot, it appears that the network does a reasonably good job of predicting *Length of stay*. The general trend of the plot is off the ideal 45-degree line in the sense that predictions for observed lengths of stay under five days tend to overestimate the length of stay, while predictions for observed lengths of stay beyond six days tend to underestimate the length of stay.

The cluster of patients in the lower left part of the plot are likely to be patients who did not undergo surgery. There is also a cluster of patients in the upper left part of the plot, where the observed length of stay is one to three days and the predicted values are much greater. It is likely that these cases are patients who died in the hospital post-surgery.

Figure 4-36
Predicted-by-observed chart for Treatment costs
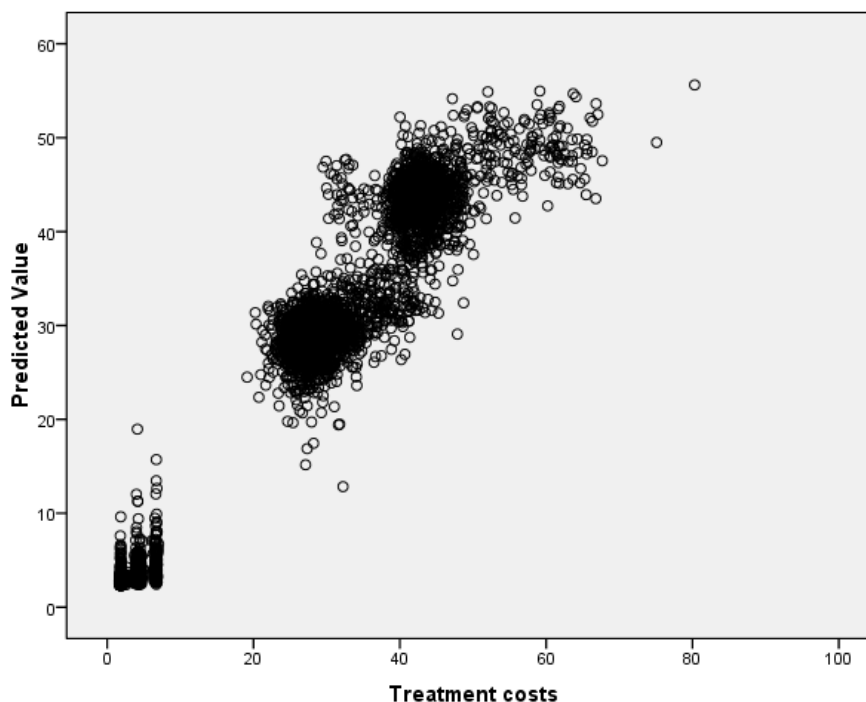
The network also appears to be reasonably good at predicting *Treatment costs*. There appear to be three primary clusters of patients:

■ In the lower left are primarily patients who did not have surgery. Their costs are relatively low and are differentiated by the type of *Clot-dissolving drugs [clotsolv]* administered in the emergency room.

■ The next cluster of patients have treatment costs of about $30,000. These are patients who had percutaneous transluminal coronary angioplasty (PTCA).

■ The final cluster have treatment costs in excess of $40,000. These are patients who had coronary artery bypass surgery (CABG). This surgery is somewhat more expensive than PTCA, and patients have a longer hospital recovery time, which increases costs further.

There are also a number of cases with costs in excess of $50,000 that the network does not predict very well. These are patients who experienced complications during surgery, which can increase the cost of surgery and length of stay.

Residual-by-Predicted Charts

Figure 4-37
Residual-by-predicted chart for Length of stay

Dependent Variable: Length of stay
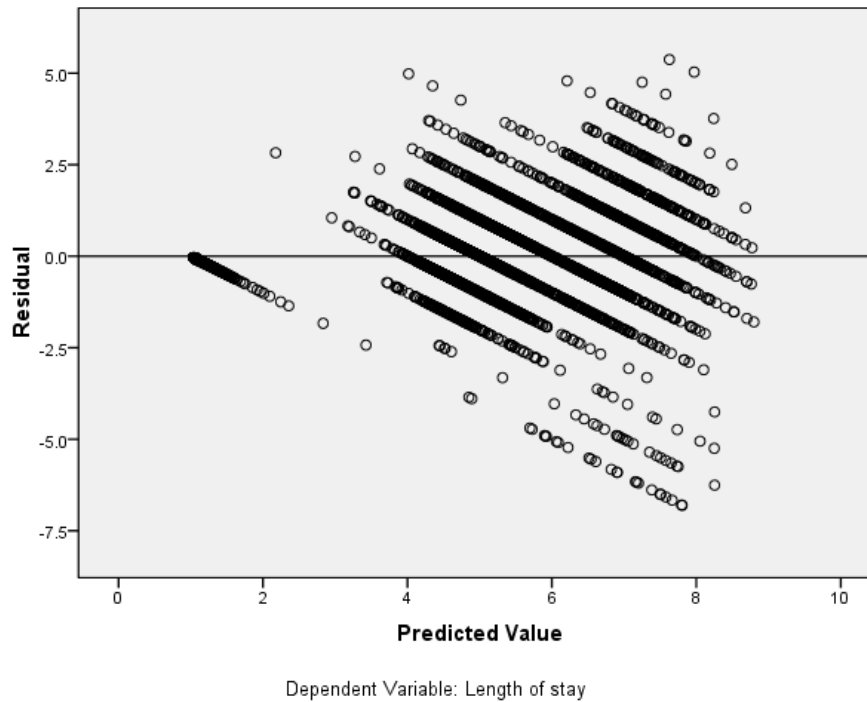
The residual-by-predicted chart displays a scatterplot of the residual (observed value minus predicted value) on the *y* axis by the predicted value on the *x* axis. Each diagonal line in this plot corresponds to a vertical line in the predicted-by-observed chart, and you can more clearly see the progression from over-prediction to under-prediction of the length of stay as the observed length of stay increases.

Figure 4-38
Residual-by-predicted chart for Treatment costs

Dependent Variable: Treatment costs
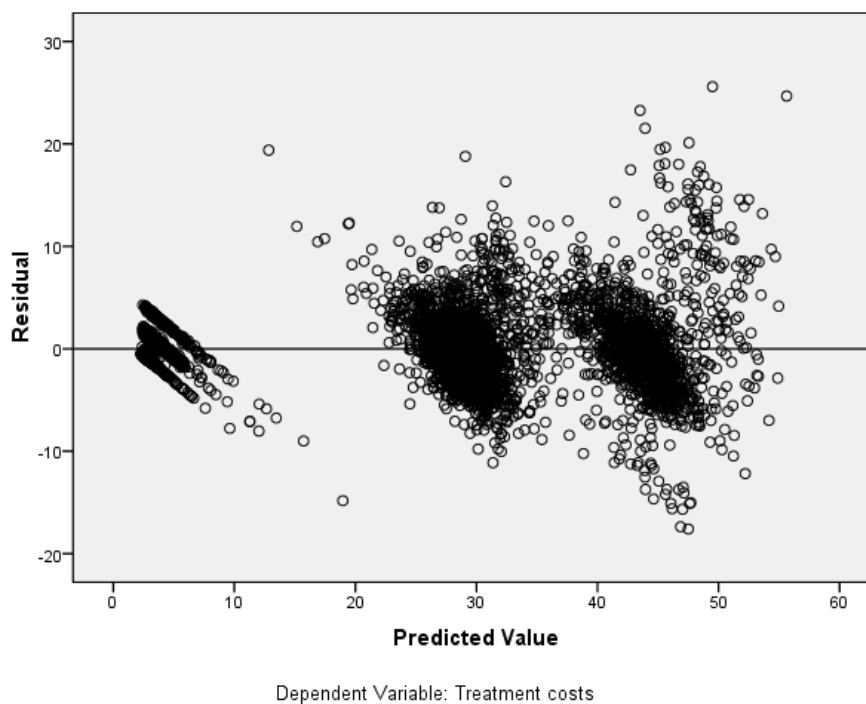
Likewise, for each of the three clusters of patients observed in the predicted-by-observed plot
for *Treatment costs*, the residual-by-predicted chart shows a progression from over-prediction
to under-prediction of costs as the observed costs increase. The patients with complications
during CABG are still clearly visible, but it is also easier to see the patients who experienced
complications during PTCA; they appear as a subcluster slightly to the right and above the main
group of PTCA patients around the $30,000 mark on the *x* axis.

Independent Variable Importance

Figure 4-39
Independent variable importance chart
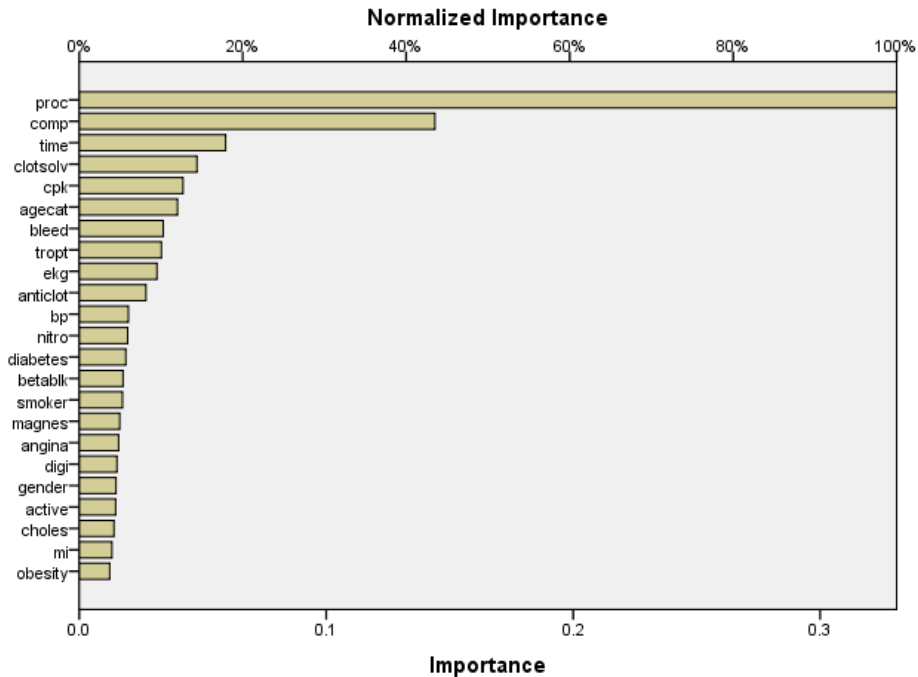
The importance chart shows that the results are dominated by the surgical procedure performed, followed by whether complications were encountered, followed distantly by other predictors. The importance of the surgical procedure is clearly visible in the plots for *Treatment costs* and is somewhat less visible for *Length of stay*, although the effect of complications on *Length of stay* appears to be visible in the patients with the largest observed lengths of stay.

Summary

The network appears to perform well when predicting values for "typical" patients but does not capture patients who died post-surgery. One possible way to handle this would be to create multiple networks. One network would predict the patient result, perhaps simply whether the patient survived or not, and then separate networks would predict *Treatment costs* and *Length of stay* conditional on whether the patient survived. You could then combine the network results to likely obtain better predictions. You could take a similar approach to the problem of under-prediction of costs and lengths of stay for patients who experienced complications during surgery.

Recommended Readings

See the following texts for more information on neural networks and multilayer perceptrons:

Bishop, C. M. 1995. *Neural Networks for Pattern Recognition*, 3rd ed. Oxford: Oxford University Press.

Fine, T. L. 1999. *Feedforward Neural Network Methodology*, 3rd ed. New York: Springer-Verlag.

Haykin, S. 1998. *Neural Networks: A Comprehensive Foundation*, 2nd ed. New York: Macmillan College Publishing.

Ripley, B. D. 1996. *Pattern Recognition and Neural Networks*. Cambridge: Cambridge University Press.

Radial Basis Function

The Radial Basis Function (RBF) procedure produces a predictive model for one or more dependent (target) variables based on values of predictor variables.

Using Radial Basis Function to Classify Telecommunications Customers

A telecommunications provider has segmented its customer base by service usage patterns, categorizing the customers into four groups. If demographic data can be used to predict group membership, you can customize offers for individual prospective customers.

Suppose information on current customers is contained in *telco.sav*. For more information, see the topic Sample Files in Appendix A on p. 84. Use the Radial Basis Function procedure to classify customers.

Preparing the Data for Analysis

Setting the random seed allows you to replicate the analysis exactly.

▶ To set the random seed, from the menus choose:
Transform > Random Number Generators...

Figure 5-1
Random Number Generators dialog box
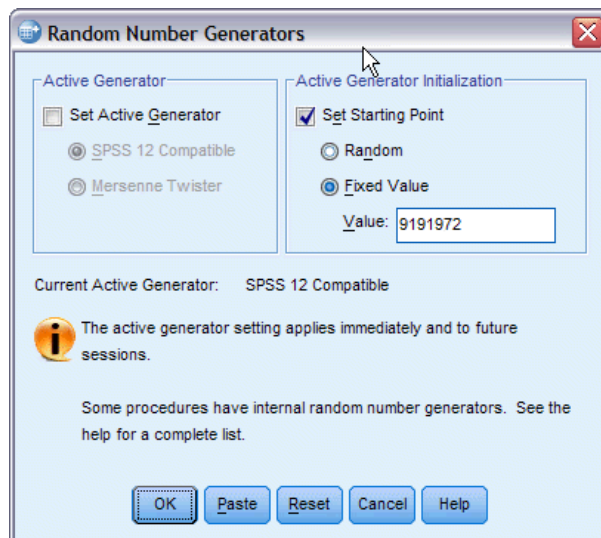
▶ Select Set Starting Point.

▶ Select Fixed Value, and type **9191972** as the value.

▶ Click OK.

Running the Analysis

▶ To run a Radial Basis Function analysis, from the menus choose:
Analyze > Neural Networks > Radial Basis Function...

Figure 5-2
Radial Basis Function: Variables tab
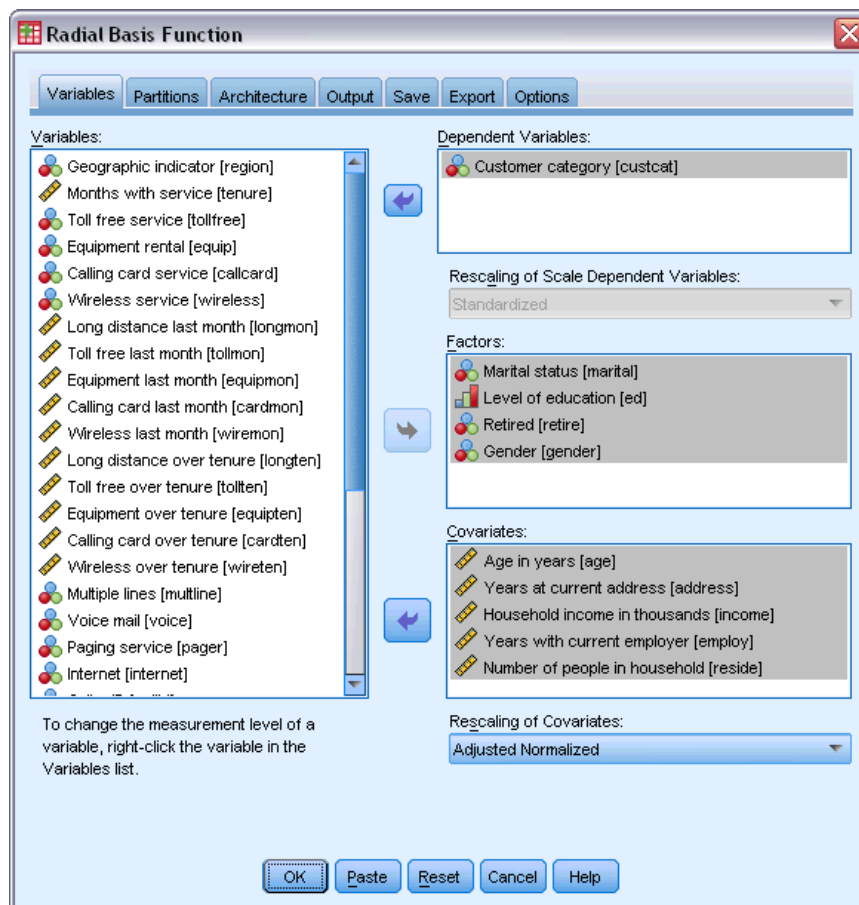
▶ Select *Customer category [custcat]* as the dependent variable.

▶ Select *Marital status [marital]*, *Level of education [ed]*, *Retired [retire]*, and *Gender [gender]* as factors.

▶ Select *Age in years [age]* through *Number of people in household [reside]* as covariates.

▶ Select Adjusted Normalized as the method for rescaling covariates.

▶ Click the Partitions tab.

Figure 5-3
Radial Basis Function: Partitions tab
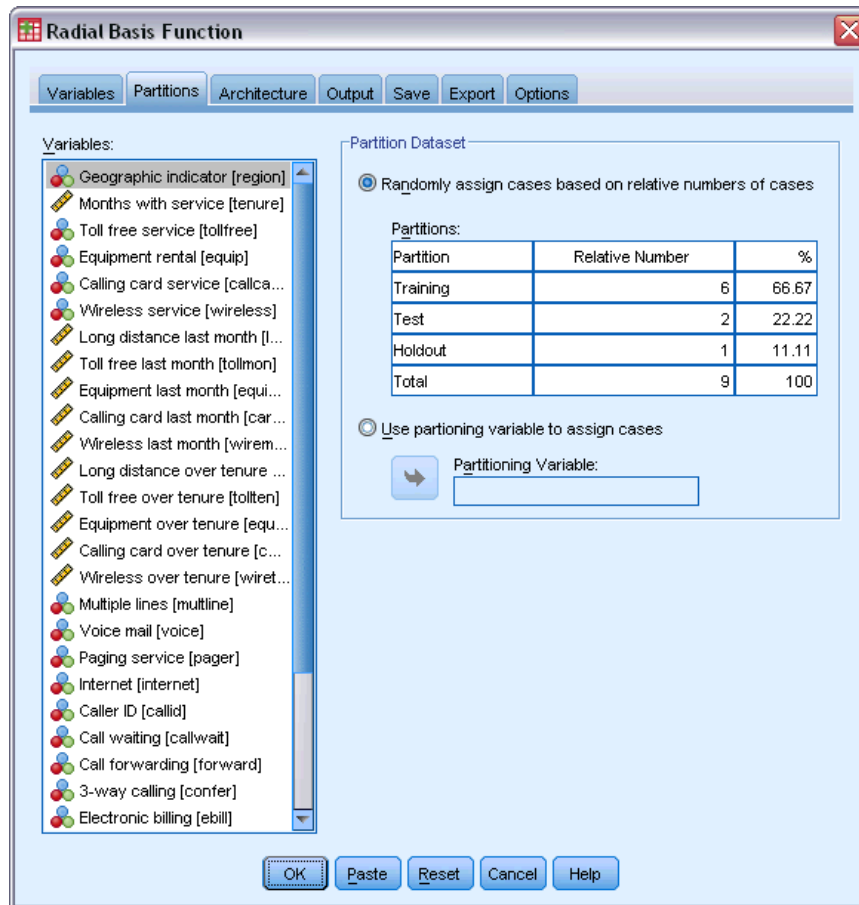
By specifying relative numbers of cases, it's easy to create fractional partitions that would be difficult to specify percentages for. Say you want to assign 2/3 of the dataset to the training sample and 2/3 of the remaining cases to testing.

▶ Type 6 as the relative number for the training sample.

▶ Type 2 as the relative number for the testing sample.

▶ Type 1 as the relative number for the holdout sample.

A total of 9 relative cases have been specified. 6/9 = 2/3, or about 66.67%, are assigned to the training sample; 2/9, or about 22.22%, are assigned to testing; 1/9, or about 11.11% are assigned to the holdout sample.

▶ Click the Output tab.

Figure 5-4
Radial Basis Function: Output tab
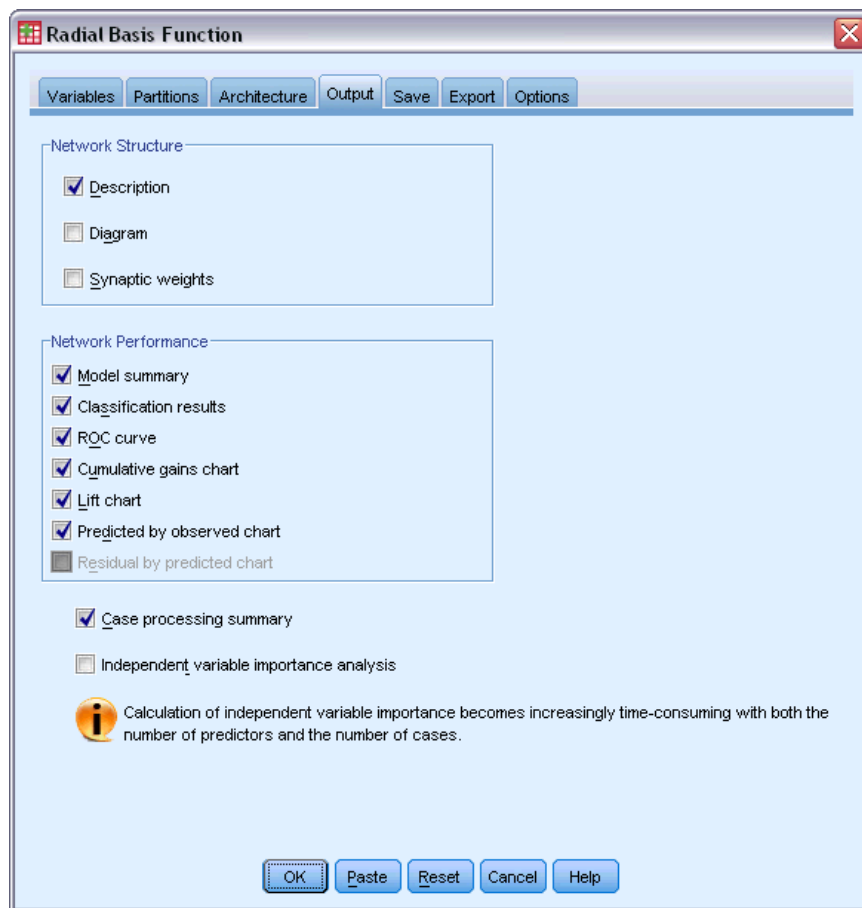
▶ Deselect Diagram in the Network Structure group.

▶ Select ROC curve, Cumulative gains chart, Lift chart, and Predicted by observed chart in the Network Performance group.

▶ Click the Save tab.

Figure 5-5
Radial Basis Function: Save tab

▶ Select Save predicted value or category for each dependent variable and Save predicted pseudo-probability for each dependent variable.

▶ Click OK.

Case Processing Summary

Figure 5-6
Case processing summary

		N	Percent
Sample	Training	665	66.5%
	Testing	224	22.4%
	Holdout	111	11.1%
Valid		1000	100.0%
Excluded		0	
Total		1000	

The case processing summary shows that 665 cases were assigned to the training sample, 224 to the testing sample, and 111 to the holdout sample. No cases were excluded from the analysis.

Network Information

Figure 5-7
Network information

Input Layer	Factors	1	Marital status
		2	Level of education
		3	Retired
		4	Gender
	Covariates	1	Age in years
		2	Years at current address
		3	Household income in thousands
		4	Years with current employer
		5	Number of people in household
	Number of Units		16
	Rescaling Method for Covariates		Adjusted Normalized
Hidden Layer	Number of Units		9[a]
	Activation Function		Softmax
Output Layer	Dependent Variables	1	Customer category
	Number of Units		4
	Activation Function		Identity
	Error Function		Sum of Squares

a. Determined by the testing data criterion: The "best" number of hidden units is the one that yields the smallest error in the testing data.

The network information table displays information about the neural network and is useful for ensuring that the specifications are correct. Here, note in particular that:

■ The number of units in the input layer is the number of covariates plus the total number of factor levels; a separate unit is created for each category of *Marital status*, *Level of education*, *Retired*, and *Gender*, and none of the categories are considered "redundant" units as is typical in many modeling procedures.

■ Likewise, a separate output unit is created for each category of *Customer category*, for a total of 4 units in the output layer.

■ Covariates are rescaled using the adjusted normalized method.

■ Automatic architecture selection has chosen 9 units in the hidden layer.

■ All other network information is default for the procedure.

Model Summary

Figure 5-8
Model summary

Training	Sum of Squares Error	235.969
	Percent Incorrect Predictions	61.8%
	Training Time	2.72
Testing	Sum of Squares Error	80.851ᵃ
	Percent Incorrect Predictions	62.9%
Holdout	Percent Incorrect Predictions	59.5%

Dependent Variable: Customer category

a. The number of hidden units is determined by the testing data criterion: The "best" number of hidden units is the one that yields the smallest error in the testing data.

The model summary displays information about the results of training, testing, and applying the final network to the holdout sample.

■ Sum of squares error is displayed because that is always used for RBF networks. This is the error function that the network tries to minimize during training and testing.

■ The percentage of incorrect predictions is taken from the classification table and will be discussed further in that topic.

Classification

Figure 5-9
Classification

Sample	Observed	Predicted				
		Basic service	E-service	Plus service	Total service	Percent Correct
Training	Basic service	64	0	66	45	36.6%
	E-service	22	1	57	61	.7%
	Plus service	47	0	104	34	56.2%
	Total service	29	1	49	85	51.8%
	Overall Percent	24.4%	.3%	41.5%	33.8%	38.2%
Testing	Basic service	18	0	26	15	30.5%
	E-service	15	0	16	22	.0%
	Plus service	11	0	39	15	60.0%
	Total service	4	0	17	26	55.3%
	Overall Percent	21.4%	.0%	43.8%	34.8%	37.1%
Holdout	Basic service	11	0	11	10	34.4%
	E-service	4	0	9	10	.0%
	Plus service	10	0	19	2	61.3%
	Total service	5	0	5	15	60.0%
	Overall Percent	27.0%	.0%	39.6%	33.3%	40.5%

Dependent Variable: Customer category

The classification table shows the practical results of using the network. For each case, the predicted response is the category with the highest predicted pseudo-probability.

■ Cells on the diagonal are correct predictions.

■ Cells off the diagonal are incorrect predictions.

Given the observed data, the "null" model (that is, one without predictors) would classify all customers into the modal group, *Plus service*. Thus, the null model would be correct 281/1000 = 28.1% of the time. The RBF network gets 10.1% more, or 38.2% of the customers. In particular, your model excels at identifying *Plus service* and *Total service* customers. However, it does an exceptionally poor job of classifying *E-service* customers. You may need to find another predictor in order to separate these customers; alternatively, given that these customers are most often misclassified as *Plus service* and *Total service* customers, the company could simply try to upsell potential customers who would normally fall into the *E-service* category.

Classifications based on the cases used to create the model tend to be too "optimistic" in the sense that their classification rate is inflated. The holdout sample helps to validate the model; here, 40.2% of these cases were correctly classified by the model. Although the holdout sample is rather small, this suggests that your model is in fact correct about two out of five times.

Predicted-by-Observed Chart

Figure 5-10
Predicted-by-observed chart
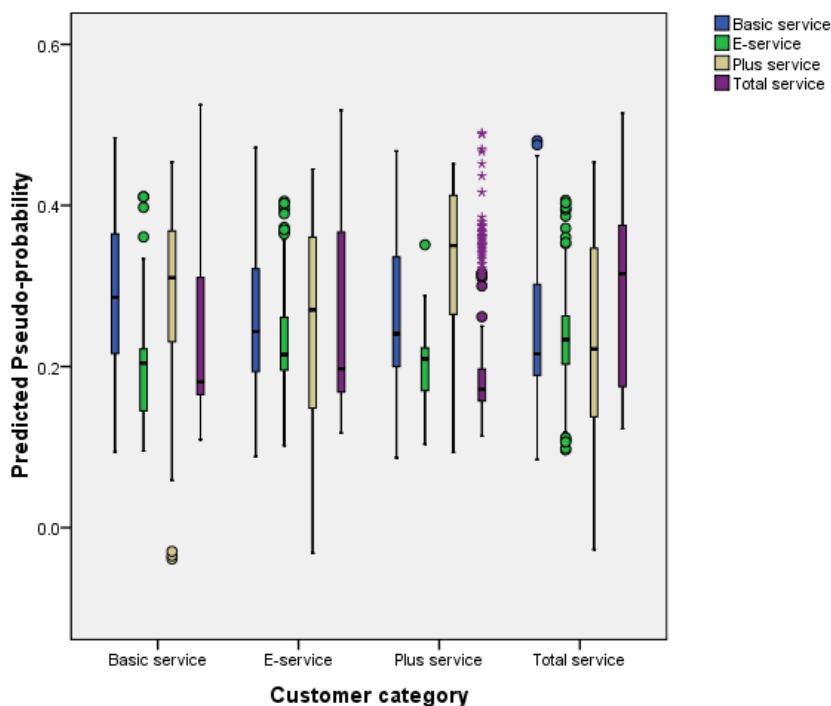
For categorical dependent variables, the predicted-by-observed chart displays clustered boxplots of predicted pseudo-probabilities for the combined training and testing samples. The *x* axis corresponds to the observed response categories, and the legend corresponds to predicted categories. Thus:

■ The leftmost boxplot shows, for cases that have observed category *Basic service*, the predicted pseudo-probability of category *Basic service*.

- The next boxplot to the right shows, for cases that have observed category *Basic service*, the predicted pseudo-probability of category *E-service*.

- The third boxplot shows, for cases that have observed category *Basic service*, the predicted pseudo-probability of category *Plus service*. Remember from the classification table that roughly as many *Basic service* customers were misclassified as *Plus service* as correctly classified as *Basic service*; thus, this boxplot is roughly equivalent to the leftmost.

- The fourth boxplot shows, for cases that have observed category *Basic service*, the predicted pseudo-probability of category *Total service*

Since there are more than two categories in the target variable, the first four boxplots are not symmetrical about the horizontal line at 0.5 nor in any other way. As a result, interpreting this plot for targets with more than two categories can be difficult because it is impossible to determine, from looking at a portion of cases in one boxplot, the corresponding location of those cases in another boxplot.

ROC Curve

Figure 5-11
ROC curve

Dependent Variable: Customer category
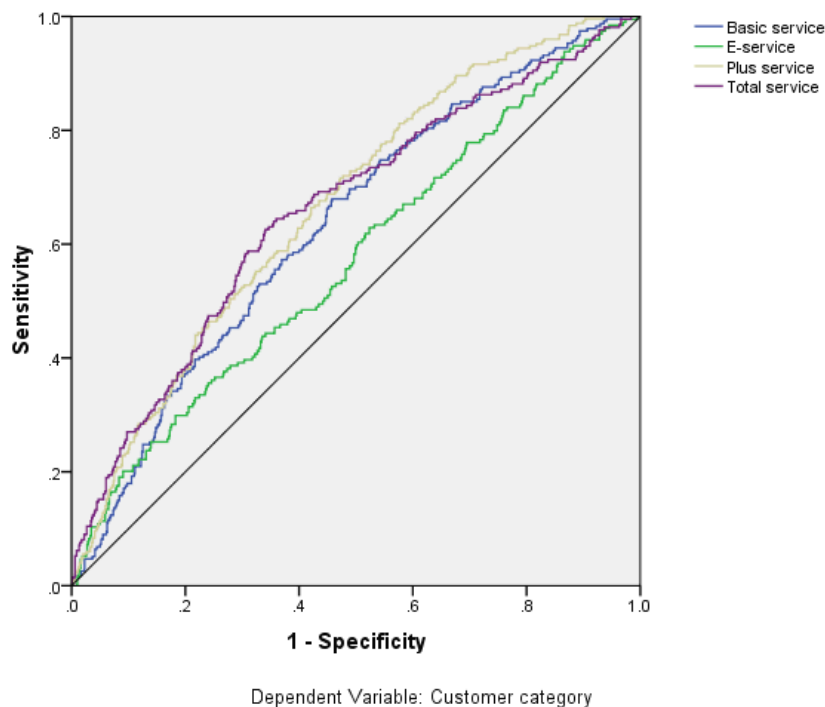
An ROC curve gives you a visual display of the **sensitivity** by **specificity** for all possible classification cutoffs. The chart shown here displays four curves, one for each category of the target variable.

Note that this chart is based on the combined training and testing samples. To produce an ROC chart for the holdout sample, split the file on the partition variable and run the ROC Curve procedure on the predicted pseudo-probabilities.

Figure 5-12
Area under the curve

		Area
Customer category	Basic service	.635
	E-service	.573
	Plus service	.668
	Total service	.659

The area under the curve is a numerical summary of the ROC curve, and the values in the table represent, for each category, the probability that the predicted pseudo-probability of being in that category is higher for a randomly chosen case in that category than for a randomly chosen case not in that category. For example, for a randomly selected customer in *Plus service* and a randomly selected customer in *Basic service*, *E-Service*, or *Total service*, there is a 0.668 probability that the model-predicted pseudo-probability of default will be higher for the customer in *Plus service*.

Cumulative Gains and Lift Charts

Figure 5-13
Cumulative gains chart
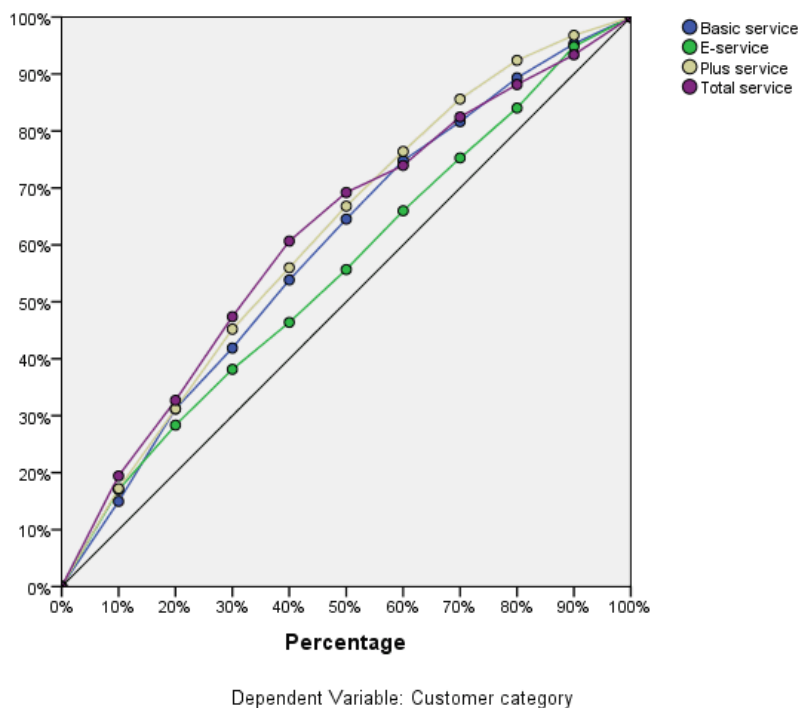
Dependent Variable: Customer category

The cumulative gains chart shows the percentage of the overall number of cases in a given category "gained" by targeting a percentage of the total number of cases. For example, the first point on the curve for the *Total service* category is approximately at (10%, 20%), meaning that if you score a dataset with the network and sort all of the cases by predicted pseudo-probability of *Total service*, you would expect the top 10% to contain approximately 20% of all of the cases that actually take the category *Total service*. Likewise, the top 20% would contain approximately 30% of the defaulters, the top 30% of cases, 50% of defaulters, and so on. If you select 100% of the scored dataset, you obtain all of the defaulters in the dataset.

The diagonal line is the "baseline" curve; if you select 10% of the cases from the scored dataset at random, you would expect to "gain" approximately 10% of all of the cases that actually take any given category. The farther above the baseline a curve lies, the greater the gain.

Figure 5-14
Lift chart
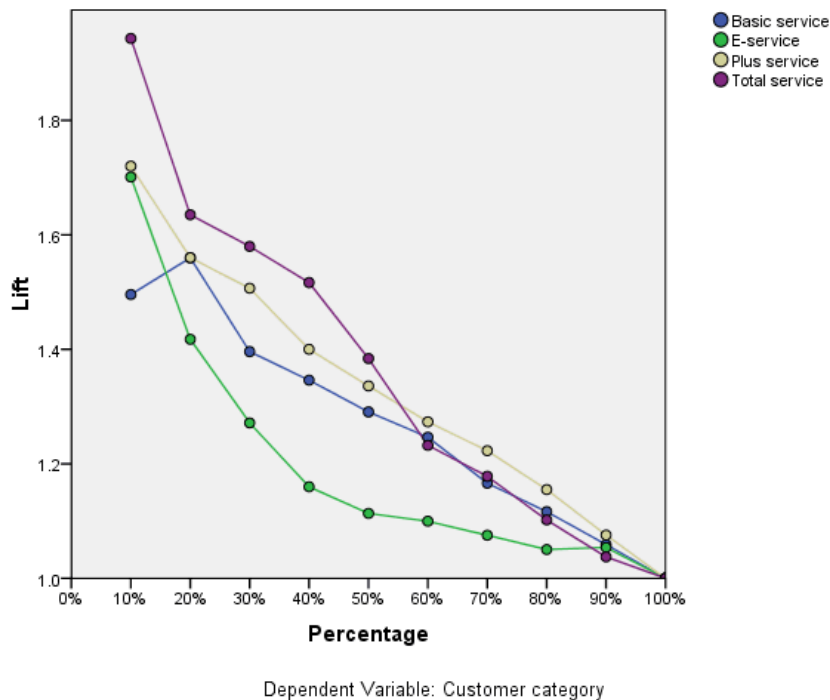
Dependent Variable: Customer category

The lift chart is derived from the cumulative gains chart; the values on the *y* axis correspond to the ratio of the cumulative gain for each curve to the baseline. Thus, the lift at 10% for the category *Total service* is approximately 20%/10% = 2.0. It provides another way of looking at the information in the cumulative gains chart.

Note: The cumulative gains and lift charts are based on the combined training and testing samples.

Recommended Readings

See the following texts for more information on Radial Basis Function:

Bishop, C. M. 1995. *Neural Networks for Pattern Recognition*, 3rd ed. Oxford: Oxford University Press.

Fine, T. L. 1999. *Feedforward Neural Network Methodology*, 3rd ed. New York: Springer-Verlag.

Haykin, S. 1998. *Neural Networks: A Comprehensive Foundation*, 2nd ed. New York: Macmillan College Publishing.

Ripley, B. D. 1996. *Pattern Recognition and Neural Networks*. Cambridge: Cambridge University Press.

Tao, K. K. 1993. A closer look at the radial basis function (RBF) networks. In: *Conference Record of the Twenty-Seventh Asilomar Conference on Signals, Systems, and Computers,* A. Singh, ed. Los Alamitos, Calif.: IEEE Comput. Soc. Press, 401–405.

Uykan, Z., C. Guzelis, M. E. Celebi, and H. N. Koivo. 2000. Analysis of input-output clustering for determining centers of RBFN. *IEEE Transactions on Neural Networks*, 11, 851–858.

Sample Files

The sample files installed with the product can be found in the *Samples* subdirectory of the installation directory. There is a separate folder within the Samples subdirectory for each of the following languages: English, French, German, Italian, Japanese, Korean, Polish, Russian, Simplified Chinese, Spanish, and Traditional Chinese.

Not all sample files are available in all languages. If a sample file is not available in a language, that language folder contains an English version of the sample file.

Descriptions

Following are brief descriptions of the sample files used in various examples throughout the documentation.

■ **accidents.sav.** This is a hypothetical data file that concerns an insurance company that is studying age and gender risk factors for automobile accidents in a given region. Each case corresponds to a cross-classification of age category and gender.

■ **adl.sav.** This is a hypothetical data file that concerns efforts to determine the benefits of a proposed type of therapy for stroke patients. Physicians randomly assigned female stroke patients to one of two groups. The first received the standard physical therapy, and the second received an additional emotional therapy. Three months following the treatments, each patient's abilities to perform common activities of daily life were scored as ordinal variables.

■ **advert.sav.** This is a hypothetical data file that concerns a retailer's efforts to examine the relationship between money spent on advertising and the resulting sales. To this end, they have collected past sales figures and the associated advertising costs..

■ **aflatoxin.sav.** This is a hypothetical data file that concerns the testing of corn crops for aflatoxin, a poison whose concentration varies widely between and within crop yields. A grain processor has received 16 samples from each of 8 crop yields and measured the alfatoxin levels in parts per billion (PPB).

■ **anorectic.sav.** While working toward a standardized symptomatology of anorectic/bulimic behavior, researchers (Van der Ham, Meulman, Van Strien, and Van Engeland, 1997) made a study of 55 adolescents with known eating disorders. Each patient was seen four times over four years, for a total of 220 observations. At each observation, the patients were scored for each of 16 symptoms. Symptom scores are missing for patient 71 at time 2, patient 76 at time 2, and patient 47 at time 3, leaving 217 valid observations.

■ **bankloan.sav.** This is a hypothetical data file that concerns a bank's efforts to reduce the rate of loan defaults. The file contains financial and demographic information on 850 past and prospective customers. The first 700 cases are customers who were previously given loans. The last 150 cases are prospective customers that the bank needs to classify as good or bad credit risks.

■ **bankloan_binning.sav.** This is a hypothetical data file containing financial and demographic information on 5,000 past customers.

- **behavior.sav.** In a classic example (Price and Bouffard, 1974), 52 students were asked to rate the combinations of 15 situations and 15 behaviors on a 10-point scale ranging from 0="extremely appropriate" to 9="extremely inappropriate." Averaged over individuals, the values are taken as dissimilarities.

- **behavior_ini.sav.** This data file contains an initial configuration for a two-dimensional solution for *behavior.sav*.

- **brakes.sav.** This is a hypothetical data file that concerns quality control at a factory that produces disc brakes for high-performance automobiles. The data file contains diameter measurements of 16 discs from each of 8 production machines. The target diameter for the brakes is 322 millimeters.

- **breakfast.sav.** In a classic study (Green and Rao, 1972), 21 Wharton School MBA students and their spouses were asked to rank 15 breakfast items in order of preference with 1="most preferred" to 15="least preferred." Their preferences were recorded under six different scenarios, from "Overall preference" to "Snack, with beverage only."

- **breakfast-overall.sav.** This data file contains the breakfast item preferences for the first scenario, "Overall preference," only.

- **broadband_1.sav.** This is a hypothetical data file containing the number of subscribers, by region, to a national broadband service. The data file contains monthly subscriber numbers for 85 regions over a four-year period.

- **broadband_2.sav.** This data file is identical to *broadband_1.sav* but contains data for three additional months.

- **car_insurance_claims.sav.** A dataset presented and analyzed elsewhere (McCullagh and Nelder, 1989) concerns damage claims for cars. The average claim amount can be modeled as having a gamma distribution, using an inverse link function to relate the mean of the dependent variable to a linear combination of the policyholder age, vehicle type, and vehicle age. The number of claims filed can be used as a scaling weight.

- **car_sales.sav.** This data file contains hypothetical sales estimates, list prices, and physical specifications for various makes and models of vehicles. The list prices and physical specifications were obtained alternately from *edmunds.com* and manufacturer sites.

- **car_sales_uprepared.sav.** This is a modified version of *car_sales.sav* that does not include any transformed versions of the fields.

- **carpet.sav.** In a popular example (Green and Wind, 1973), a company interested in marketing a new carpet cleaner wants to examine the influence of five factors on consumer preference—package design, brand name, price, a *Good Housekeeping* seal, and a money-back guarantee. There are three factor levels for package design, each one differing in the location of the applicator brush; three brand names (*K2R*, *Glory*, and *Bissell*); three price levels; and two levels (either no or yes) for each of the last two factors. Ten consumers rank 22 profiles defined by these factors. The variable *Preference* contains the rank of the average rankings for each profile. Low rankings correspond to high preference. This variable reflects an overall measure of preference for each profile.

- **carpet_prefs.sav.** This data file is based on the same example as described for *carpet.sav*, but it contains the actual rankings collected from each of the 10 consumers. The consumers were asked to rank the 22 product profiles from the most to the least preferred. The variables *PREF1* through *PREF22* contain the identifiers of the associated profiles, as defined in *carpet_plan.sav*.

- **catalog.sav.** This data file contains hypothetical monthly sales figures for three products sold by a catalog company. Data for five possible predictor variables are also included.

- **catalog_seasfac.sav.** This data file is the same as *catalog.sav* except for the addition of a set of seasonal factors calculated from the Seasonal Decomposition procedure along with the accompanying date variables.

- **cellular.sav.** This is a hypothetical data file that concerns a cellular phone company's efforts to reduce churn. Churn propensity scores are applied to accounts, ranging from 0 to 100. Accounts scoring 50 or above may be looking to change providers.

- **ceramics.sav.** This is a hypothetical data file that concerns a manufacturer's efforts to determine whether a new premium alloy has a greater heat resistance than a standard alloy. Each case represents a separate test of one of the alloys; the heat at which the bearing failed is recorded.

- **cereal.sav.** This is a hypothetical data file that concerns a poll of 880 people about their breakfast preferences, also noting their age, gender, marital status, and whether or not they have an active lifestyle (based on whether they exercise at least twice a week). Each case represents a separate respondent.

- **clothing_defects.sav.** This is a hypothetical data file that concerns the quality control process at a clothing factory. From each lot produced at the factory, the inspectors take a sample of clothes and count the number of clothes that are unacceptable.

- **coffee.sav.** This data file pertains to perceived images of six iced-coffee brands (Kennedy, Riquier, and Sharp, 1996) . For each of 23 iced-coffee image attributes, people selected all brands that were described by the attribute. The six brands are denoted AA, BB, CC, DD, EE, and FF to preserve confidentiality.

- **contacts.sav.** This is a hypothetical data file that concerns the contact lists for a group of corporate computer sales representatives. Each contact is categorized by the department of the company in which they work and their company ranks. Also recorded are the amount of the last sale made, the time since the last sale, and the size of the contact's company.

- **creditpromo.sav.** This is a hypothetical data file that concerns a department store's efforts to evaluate the effectiveness of a recent credit card promotion. To this end, 500 cardholders were randomly selected. Half received an ad promoting a reduced interest rate on purchases made over the next three months. Half received a standard seasonal ad.

- **customer_dbase.sav.** This is a hypothetical data file that concerns a company's efforts to use the information in its data warehouse to make special offers to customers who are most likely to reply. A subset of the customer base was selected at random and given the special offers, and their responses were recorded.

- **customer_information.sav.** A hypothetical data file containing customer mailing information, such as name and address.

- **customer_subset.sav.** A subset of 80 cases from *customer_dbase.sav*.

- **debate.sav.** This is a hypothetical data file that concerns paired responses to a survey from attendees of a political debate before and after the debate. Each case corresponds to a separate respondent.

- **debate_aggregate.sav.** This is a hypothetical data file that aggregates the responses in *debate.sav*. Each case corresponds to a cross-classification of preference before and after the debate.

- **demo.sav.** This is a hypothetical data file that concerns a purchased customer database, for the purpose of mailing monthly offers. Whether or not the customer responded to the offer is recorded, along with various demographic information.

- **demo_cs_1.sav.** This is a hypothetical data file that concerns the first step of a company's efforts to compile a database of survey information. Each case corresponds to a different city, and the region, province, district, and city identification are recorded.

- **demo_cs_2.sav.** This is a hypothetical data file that concerns the second step of a company's efforts to compile a database of survey information. Each case corresponds to a different household unit from cities selected in the first step, and the region, province, district, city, subdivision, and unit identification are recorded. The sampling information from the first two stages of the design is also included.

- **demo_cs.sav.** This is a hypothetical data file that contains survey information collected using a complex sampling design. Each case corresponds to a different household unit, and various demographic and sampling information is recorded.

- **dmdata.sav.** This is a hypothetical data file that contains demographic and purchasing information for a direct marketing company. *dmdata2.sav* contains information for a subset of contacts that received a test mailing, and *dmdata3.sav* contains information on the remaining contacts who did not receive the test mailing.

- **dietstudy.sav.** This hypothetical data file contains the results of a study of the "Stillman diet" (Rickman, Mitchell, Dingman, and Dalen, 1974). Each case corresponds to a separate subject and records his or her pre- and post-diet weights in pounds and triglyceride levels in mg/100 ml.

- **dvdplayer.sav.** This is a hypothetical data file that concerns the development of a new DVD player. Using a prototype, the marketing team has collected focus group data. Each case corresponds to a separate surveyed user and records some demographic information about them and their responses to questions about the prototype.

- **german_credit.sav.** This data file is taken from the "German credit" dataset in the Repository of Machine Learning Databases (Blake and Merz, 1998) at the University of California, Irvine.

- **grocery_1month.sav.** This hypothetical data file is the *grocery_coupons.sav* data file with the weekly purchases "rolled-up" so that each case corresponds to a separate customer. Some of the variables that changed weekly disappear as a result, and the amount spent recorded is now the sum of the amounts spent during the four weeks of the study.

- **grocery_coupons.sav.** This is a hypothetical data file that contains survey data collected by a grocery store chain interested in the purchasing habits of their customers. Each customer is followed for four weeks, and each case corresponds to a separate customer-week and records information about where and how the customer shops, including how much was spent on groceries during that week.

- **guttman.sav.** Bell (Bell, 1961) presented a table to illustrate possible social groups. Guttman (Guttman, 1968) used a portion of this table, in which five variables describing such things as social interaction, feelings of belonging to a group, physical proximity of members, and formality of the relationship were crossed with seven theoretical social groups, including crowds (for example, people at a football game), audiences (for example, people at a theater or classroom lecture), public (for example, newspaper or television audiences), mobs (like a crowd but with much more intense interaction), primary groups (intimate), secondary groups

(voluntary), and the modern community (loose confederation resulting from close physical proximity and a need for specialized services).

- **health_funding.sav.** This is a hypothetical data file that contains data on health care funding (amount per 100 population), disease rates (rate per 10,000 population), and visits to health care providers (rate per 10,000 population). Each case represents a different city.

- **hivassay.sav.** This is a hypothetical data file that concerns the efforts of a pharmaceutical lab to develop a rapid assay for detecting HIV infection. The results of the assay are eight deepening shades of red, with deeper shades indicating greater likelihood of infection. A laboratory trial was conducted on 2,000 blood samples, half of which were infected with HIV and half of which were clean.

- **hourlywagedata.sav.** This is a hypothetical data file that concerns the hourly wages of nurses from office and hospital positions and with varying levels of experience.

- **insurance_claims.sav.** This is a hypothetical data file that concerns an insurance company that wants to build a model for flagging suspicious, potentially fraudulent claims. Each case represents a separate claim.

- **insure.sav.** This is a hypothetical data file that concerns an insurance company that is studying the risk factors that indicate whether a client will have to make a claim on a 10-year term life insurance contract. Each case in the data file represents a pair of contracts, one of which recorded a claim and the other didn't, matched on age and gender.

- **judges.sav.** This is a hypothetical data file that concerns the scores given by trained judges (plus one enthusiast) to 300 gymnastics performances. Each row represents a separate performance; the judges viewed the same performances.

- **kinship_dat.sav.** Rosenberg and Kim (Rosenberg and Kim, 1975) set out to analyze 15 kinship terms (aunt, brother, cousin, daughter, father, granddaughter, grandfather, grandmother, grandson, mother, nephew, niece, sister, son, uncle). They asked four groups of college students (two female, two male) to sort these terms on the basis of similarities. Two groups (one female, one male) were asked to sort twice, with the second sorting based on a different criterion from the first sort. Thus, a total of six "sources" were obtained. Each source corresponds to a 15×15 proximity matrix, whose cells are equal to the number of people in a source minus the number of times the objects were partitioned together in that source.

- **kinship_ini.sav.** This data file contains an initial configuration for a three-dimensional solution for *kinship_dat.sav*.

- **kinship_var.sav.** This data file contains independent variables *gender*, *gener*(ation), and *degree* (of separation) that can be used to interpret the dimensions of a solution for *kinship_dat.sav*. Specifically, they can be used to restrict the space of the solution to a linear combination of these variables.

- **marketvalues.sav.** This data file concerns home sales in a new housing development in Algonquin, Ill., during the years from 1999–2000. These sales are a matter of public record.

- **nhis2000_subset.sav.** The National Health Interview Survey (NHIS) is a large, population-based survey of the U.S. civilian population. Interviews are carried out face-to-face in a nationally representative sample of households. Demographic information and observations about health behaviors and status are obtained for members of each household. This data file contains a subset of information from the 2000 survey. National Center for Health Statistics. National Health Interview Survey, 2000. Public-use data file and documentation. *ftp://ftp.cdc.gov/pub/Health_Statistics/NCHS/Datasets/NHIS/2000/.* Accessed 2003.

- **ozone.sav.** The data include 330 observations on six meteorological variables for predicting ozone concentration from the remaining variables. Previous researchers (Breiman and Friedman, 1985), (Hastie and Tibshirani, 1990), among others found nonlinearities among these variables, which hinder standard regression approaches.

- **pain_medication.sav.** This hypothetical data file contains the results of a clinical trial for anti-inflammatory medication for treating chronic arthritic pain. Of particular interest is the time it takes for the drug to take effect and how it compares to an existing medication.

- **patient_los.sav.** This hypothetical data file contains the treatment records of patients who were admitted to the hospital for suspected myocardial infarction (MI, or "heart attack"). Each case corresponds to a separate patient and records many variables related to their hospital stay.

- **patlos_sample.sav.** This hypothetical data file contains the treatment records of a sample of patients who received thrombolytics during treatment for myocardial infarction (MI, or "heart attack"). Each case corresponds to a separate patient and records many variables related to their hospital stay.

- **poll_cs.sav.** This is a hypothetical data file that concerns pollsters' efforts to determine the level of public support for a bill before the legislature. The cases correspond to registered voters. Each case records the county, township, and neighborhood in which the voter lives.

- **poll_cs_sample.sav.** This hypothetical data file contains a sample of the voters listed in *poll_cs.sav*. The sample was taken according to the design specified in the *poll.csplan* plan file, and this data file records the inclusion probabilities and sample weights. Note, however, that because the sampling plan makes use of a probability-proportional-to-size (PPS) method, there is also a file containing the joint selection probabilities (*poll_jointprob.sav*). The additional variables corresponding to voter demographics and their opinion on the proposed bill were collected and added the data file after the sample as taken.

- **property_assess.sav.** This is a hypothetical data file that concerns a county assessor's efforts to keep property value assessments up to date on limited resources. The cases correspond to properties sold in the county in the past year. Each case in the data file records the township in which the property lies, the assessor who last visited the property, the time since that assessment, the valuation made at that time, and the sale value of the property.

- **property_assess_cs.sav.** This is a hypothetical data file that concerns a state assessor's efforts to keep property value assessments up to date on limited resources. The cases correspond to properties in the state. Each case in the data file records the county, township, and neighborhood in which the property lies, the time since the last assessment, and the valuation made at that time.

- **property_assess_cs_sample.sav.** This hypothetical data file contains a sample of the properties listed in *property_assess_cs.sav*. The sample was taken according to the design specified in the *property_assess.csplan* plan file, and this data file records the inclusion probabilities and sample weights. The additional variable *Current value* was collected and added to the data file after the sample was taken.

- **recidivism.sav.** This is a hypothetical data file that concerns a government law enforcement agency's efforts to understand recidivism rates in their area of jurisdiction. Each case corresponds to a previous offender and records their demographic information, some details of their first crime, and then the time until their second arrest, if it occurred within two years of the first arrest.

- **recidivism_cs_sample.sav.** This is a hypothetical data file that concerns a government law enforcement agency's efforts to understand recidivism rates in their area of jurisdiction. Each case corresponds to a previous offender, released from their first arrest during the month of June, 2003, and records their demographic information, some details of their first crime, and the data of their second arrest, if it occurred by the end of June, 2006. Offenders were selected from sampled departments according to the sampling plan specified in *recidivism_cs.csplan*; because it makes use of a probability-proportional-to-size (PPS) method, there is also a file containing the joint selection probabilities (*recidivism_cs_jointprob.sav*).

- **rfm_transactions.sav.** A hypothetical data file containing purchase transaction data, including date of purchase, item(s) purchased, and monetary amount of each transaction.

- **salesperformance.sav.** This is a hypothetical data file that concerns the evaluation of two new sales training courses. Sixty employees, divided into three groups, all receive standard training. In addition, group 2 gets technical training; group 3, a hands-on tutorial. Each employee was tested at the end of the training course and their score recorded. Each case in the data file represents a separate trainee and records the group to which they were assigned and the score they received on the exam.

- **satisf.sav.** This is a hypothetical data file that concerns a satisfaction survey conducted by a retail company at 4 store locations. 582 customers were surveyed in all, and each case represents the responses from a single customer.

- **screws.sav.** This data file contains information on the characteristics of screws, bolts, nuts, and tacks (Hartigan, 1975).

- **shampoo_ph.sav.** This is a hypothetical data file that concerns the quality control at a factory for hair products. At regular time intervals, six separate output batches are measured and their pH recorded. The target range is 4.5–5.5.

- **ships.sav.** A dataset presented and analyzed elsewhere (McCullagh et al., 1989) that concerns damage to cargo ships caused by waves. The incident counts can be modeled as occurring at a Poisson rate given the ship type, construction period, and service period. The aggregate months of service for each cell of the table formed by the cross-classification of factors provides values for the exposure to risk.

- **site.sav.** This is a hypothetical data file that concerns a company's efforts to choose new sites for their expanding business. They have hired two consultants to separately evaluate the sites, who, in addition to an extended report, summarized each site as a "good," "fair," or "poor" prospect.

- **smokers.sav.** This data file is abstracted from the 1998 National Household Survey of Drug Abuse and is a probability sample of American households. (*http://dx.doi.org/10.3886/ICPSR02934*) Thus, the first step in an analysis of this data file should be to weight the data to reflect population trends.

- **stocks.sav** This hypothetical data file contains stocks prices and volume for one year.

- **stroke_clean.sav.** This hypothetical data file contains the state of a medical database after it has been cleaned using procedures in the Data Preparation option.

- **stroke_invalid.sav.** This hypothetical data file contains the initial state of a medical database and contains several data entry errors.

- **stroke_survival.** This hypothetical data file concerns survival times for patients exiting a rehabilitation program post-ischemic stroke face a number of challenges. Post-stroke, the occurrence of myocardial infarction, ischemic stroke, or hemorrhagic stroke is noted and the time of the event recorded. The sample is left-truncated because it only includes patients who survived through the end of the rehabilitation program administered post-stroke.

- **stroke_valid.sav.** This hypothetical data file contains the state of a medical database after the values have been checked using the Validate Data procedure. It still contains potentially anomalous cases.

- **survey_sample.sav.** This data file contains survey data, including demographic data and various attitude measures. It is based on a subset of variables from the 1998 NORC General Social Survey, although some data values have been modified and additional fictitious variables have been added for demonstration purposes.

- **telco.sav.** This is a hypothetical data file that concerns a telecommunications company's efforts to reduce churn in their customer base. Each case corresponds to a separate customer and records various demographic and service usage information.

- **telco_extra.sav.** This data file is similar to the *telco.sav* data file, but the "tenure" and log-transformed customer spending variables have been removed and replaced by standardized log-transformed customer spending variables.

- **telco_missing.sav.** This data file is a subset of the *telco.sav* data file, but some of the demographic data values have been replaced with missing values.

- **testmarket.sav.** This hypothetical data file concerns a fast food chain's plans to add a new item to its menu. There are three possible campaigns for promoting the new product, so the new item is introduced at locations in several randomly selected markets. A different promotion is used at each location, and the weekly sales of the new item are recorded for the first four weeks. Each case corresponds to a separate location-week.

- **testmarket_1month.sav.** This hypothetical data file is the *testmarket.sav* data file with the weekly sales "rolled-up" so that each case corresponds to a separate location. Some of the variables that changed weekly disappear as a result, and the sales recorded is now the sum of the sales during the four weeks of the study.

- **tree_car.sav.** This is a hypothetical data file containing demographic and vehicle purchase price data.

- **tree_credit.sav.** This is a hypothetical data file containing demographic and bank loan history data.

- **tree_missing_data.sav** This is a hypothetical data file containing demographic and bank loan history data with a large number of missing values.

- **tree_score_car.sav.** This is a hypothetical data file containing demographic and vehicle purchase price data.

- **tree_textdata.sav.** A simple data file with only two variables intended primarily to show the default state of variables prior to assignment of measurement level and value labels.

- **tv-survey.sav.** This is a hypothetical data file that concerns a survey conducted by a TV studio that is considering whether to extend the run of a successful program. 906 respondents were asked whether they would watch the program under various conditions. Each row represents a separate respondent; each column is a separate condition.

- **ulcer_recurrence.sav.** This file contains partial information from a study designed to compare the efficacy of two therapies for preventing the recurrence of ulcers. It provides a good example of interval-censored data and has been presented and analyzed elsewhere (Collett, 2003).

- **ulcer_recurrence_recoded.sav.** This file reorganizes the information in *ulcer_recurrence.sav* to allow you model the event probability for each interval of the study rather than simply the end-of-study event probability. It has been presented and analyzed elsewhere (Collett et al., 2003).

- **verd1985.sav.** This data file concerns a survey (Verdegaal, 1985). The responses of 15 subjects to 8 variables were recorded. The variables of interest are divided into three sets. Set 1 includes *age* and *marital*, set 2 includes *pet* and *news*, and set 3 includes *music* and *live*. *Pet* is scaled as multiple nominal and *age* is scaled as ordinal; all of the other variables are scaled as single nominal.

- **virus.sav.** This is a hypothetical data file that concerns the efforts of an Internet service provider (ISP) to determine the effects of a virus on its networks. They have tracked the (approximate) percentage of infected e-mail traffic on its networks over time, from the moment of discovery until the threat was contained.

- **wheeze_steubenville.sav.** This is a subset from a longitudinal study of the health effects of air pollution on children (Ware, Dockery, Spiro III, Speizer, and Ferris Jr., 1984). The data contain repeated binary measures of the wheezing status for children from Steubenville, Ohio, at ages 7, 8, 9 and 10 years, along with a fixed recording of whether or not the mother was a smoker during the first year of the study.

- **workprog.sav.** This is a hypothetical data file that concerns a government works program that tries to place disadvantaged people into better jobs. A sample of potential program participants were followed, some of whom were randomly selected for enrollment in the program, while others were not. Each case represents a separate program participant.

- **worldsales.sav** This hypothetical data file contains sales revenue by continent and product.

Notices

This information was developed for products and services offered worldwide.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785, U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing, Legal and Intellectual Property Law, IBM Japan Ltd., 1623-14, Shimotsuruma, Yamato-shi, Kanagawa 242-8502 Japan.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Software Group, Attention: Licensing, 233 S. Wacker Dr., Chicago, IL 60606, USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, ibm.com, and SPSS are trademarks of IBM Corporation, registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at *http://www.ibm.com/legal/copytrade.shtml*.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

This product uses WinWrap Basic, Copyright 1993-2007, Polar Engineering and Consulting, *http://www.winwrap.com*.

Other product and service names might be trademarks of IBM or other companies.

Adobe product screenshot(s) reprinted with permission from Adobe Systems Incorporated.

Microsoft product screenshot(s) reprinted with permission from Microsoft Corporation.

Bibliography

Bell, E. H. 1961. *Social foundations of human behavior: Introduction to the study of sociology*. New York: Harper & Row.

Bishop, C. M. 1995. *Neural Networks for Pattern Recognition*, 3rd ed. Oxford: Oxford University Press.

Blake, C. L., and C. J. Merz. 1998. "UCI Repository of machine learning databases." Available at http://www.ics.uci.edu/~mlearn/MLRepository.html.

Breiman, L., and J. H. Friedman. 1985. Estimating optimal transformations for multiple regression and correlation. *Journal of the American Statistical Association*, 80, 580–598.

Collett, D. 2003. *Modelling survival data in medical research*, 2 ed. Boca Raton: Chapman & Hall/CRC.

Fine, T. L. 1999. *Feedforward Neural Network Methodology*, 3rd ed. New York: Springer-Verlag.

Green, P. E., and V. Rao. 1972. *Applied multidimensional scaling*. Hinsdale, Ill.: Dryden Press.

Green, P. E., and Y. Wind. 1973. *Multiattribute decisions in marketing: A measurement approach*. Hinsdale, Ill.: Dryden Press.

Guttman, L. 1968. A general nonmetric technique for finding the smallest coordinate space for configurations of points. *Psychometrika*, 33, 469–506.

Hartigan, J. A. 1975. *Clustering algorithms*. New York: John Wiley and Sons.

Hastie, T., and R. Tibshirani. 1990. *Generalized additive models*. London: Chapman and Hall.

Haykin, S. 1998. *Neural Networks: A Comprehensive Foundation*, 2nd ed. New York: Macmillan College Publishing.

Kennedy, R., C. Riquier, and B. Sharp. 1996. Practical applications of correspondence analysis to categorical data in market research. *Journal of Targeting, Measurement, and Analysis for Marketing*, 5, 56–70.

McCullagh, P., and J. A. Nelder. 1989. *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

Price, R. H., and D. L. Bouffard. 1974. Behavioral appropriateness and situational constraints as dimensions of social behavior. *Journal of Personality and Social Psychology*, 30, 579–586.

Rickman, R., N. Mitchell, J. Dingman, and J. E. Dalen. 1974. Changes in serum cholesterol during the Stillman Diet. *Journal of the American Medical Association*, 228, 54–58.

Ripley, B. D. 1996. *Pattern Recognition and Neural Networks*. Cambridge: Cambridge University Press.

Rosenberg, S., and M. P. Kim. 1975. The method of sorting as a data-gathering procedure in multivariate research. *Multivariate Behavioral Research*, 10, 489–502.

Tao, K. K. 1993. A closer look at the radial basis function (RBF) networks. In: *Conference Record of the Twenty-Seventh Asilomar Conference on Signals, Systems, and Computers,* A. Singh, ed. Los Alamitos, Calif.: IEEE Comput. Soc. Press, 401–405.

Uykan, Z., C. Guzelis, M. E. Celebi, and H. N. Koivo. 2000. Analysis of input-output clustering for determining centers of RBFN. *IEEE Transactions on Neural Networks*, 11, 851–858.

Van der Ham, T., J. J. Meulman, D. C. Van Strien, and H. Van Engeland. 1997. Empirically based subgrouping of eating disorders in adolescents: A longitudinal perspective. *British Journal of Psychiatry*, 170, 363–368.

Verdegaal, R. 1985. *Meer sets analyse voor kwalitatieve gegevens (in Dutch)*. Leiden: Department of Data Theory, University of Leiden.

Ware, J. H., D. W. Dockery, A. Spiro III, F. E. Speizer, and B. G. Ferris Jr.. 1984. Passive smoking, gas cooking, and respiratory health of children living in six cities. *American Review of Respiratory Diseases*, 129, 366–374.

Index