



NEURALWARE

## ***NeuralSight*® Product Overview**

**Release 2.0**

NeuralWare  
230 E Main Street Suite 200  
Carnegie PA 15106

Contact:  
Jack Copper  
412.278.6290  
[jack.copper@neuralware.com](mailto:jack.copper@neuralware.com)

This document is provided for informational purposes only. Information in this document is subject to change without notice.

© 2005 NeuralWare. All Rights Reserved.

November 2005

## Introducing NeuralSight

Neural networks comprise a collection of powerful adaptive artificial intelligence technologies that are applied to prediction, classification, and clustering problems across domains as diverse as bioinformatics, retail sales, credit risk and portfolio management, manufacturing, imaging, and robotics, as well as a host of others.

Since 1987, NeuralWare has been a leader in providing comprehensive neural network model design, development, and deployment capabilities for commercial, government, and academic users around the world.

**NeuralSight®** is the latest addition to the NeuralWare family of advanced neural network model development platforms. *NeuralSight* extends the power and flexibility of the *NeuralWorks Predict®* neural network engine by incorporating sophisticated yet easy-to-use facilities for building and evaluating hundreds, even thousands of neural network models with minimal effort and intervention. New features in *NeuralSight 2.0* include *Self-Organizing Map (SOM)* clustering and visualization capabilities, as well as usability enhancements and support for controlling *NeuralSight* model building through configuration script files.

### Key Benefits of the NeuralSight Platform

- No programming and no knowledge of the underlying mathematics of neural network technology is required to build and deploy robust, high-performance models for prediction and classification decision support systems and knowledge discovery through clustering.
- All aspects of model development can be logged to provide a rigorous audit trail of the data used in training and validation as well as the specific characteristics and internal structure of every model.
- Configuring the environment to build hundreds of models takes only minutes; NeuralSight can then run for hours unattended until the required number of models have been trained.
- NeuralSight automatically applies transformations of raw data to create a richer pool of potential model inputs, then a genetic algorithm optimization technique is employed to identify the best set of neural network inputs.
- After a collection of models is trained, with only a few mouse clicks all the models are ranked by standard performance metrics (e.g. classification accuracy, RMS error). *NeuralSight's* integrated Sensitivity Analysis facility can then be used to show the relative influence of each input on over-all model performance.
- Models produced by NeuralSight are fully compatible with *NeuralWorks Predict*. Models can be immediately used with data in Excel spreadsheets, or an existing or new custom enterprise application can be linked to the Predict Run-Time Engine so that models can be used by the application. In addition, FlashCode™ can be generated and compiled/linked for use in embedded applications which do not have native file system support.

## Compare NeuralSight Modeling with Traditional Approaches

NeuralSight	Traditional
<p>Obtain and clean up raw data</p> <p>Accept NeuralSight default model building options – or select/clear specific options by simple mouse clicks</p> <p>Specify how many models to build and/or how much time NeuralSight should spend building models then simply click a button to build all models</p> <p>Activate the <i>NeuralSight</i> Evaluation tab and click a button to specify the metric by which to rank models – immediately see all models ranked by the specified metric</p> <p>Select model(s) to deploy</p> <p>Click a button to generate source code if required, otherwise load the model in <i>Predict</i> (Excel) or make the model file available to an application linked to the Predict Run-Time Engine – confident that if the model must be updated a new model can be immediately loaded and used</p>	<p>Obtain and clean up raw data</p> <p>Identify and evaluate (trial and error) possible transformations of raw data</p> <p>Select (trial and error) model inputs</p> <p>Specify (trial and error) model structure/architecture</p> <p>Specify (trial and error) learning rules and learning rule coefficients/values</p> <p>Build one model, then repeat above steps</p> <p>Run models to compute performance metrics</p> <p>Load metrics from each model into a spreadsheet or similar program, and manually organize results to rank models and tabulate/plot results</p> <p>Select model(s) to deploy</p> <p>Extract model coefficients, structures, etc. from development environment to hand-craft model into a format suitable for deployment</p> <p>Hand-craft software interfaces for the model so that it can be used in the deployed application – and worry that if the model must be updated the new model structure may require writing another software interface</p>

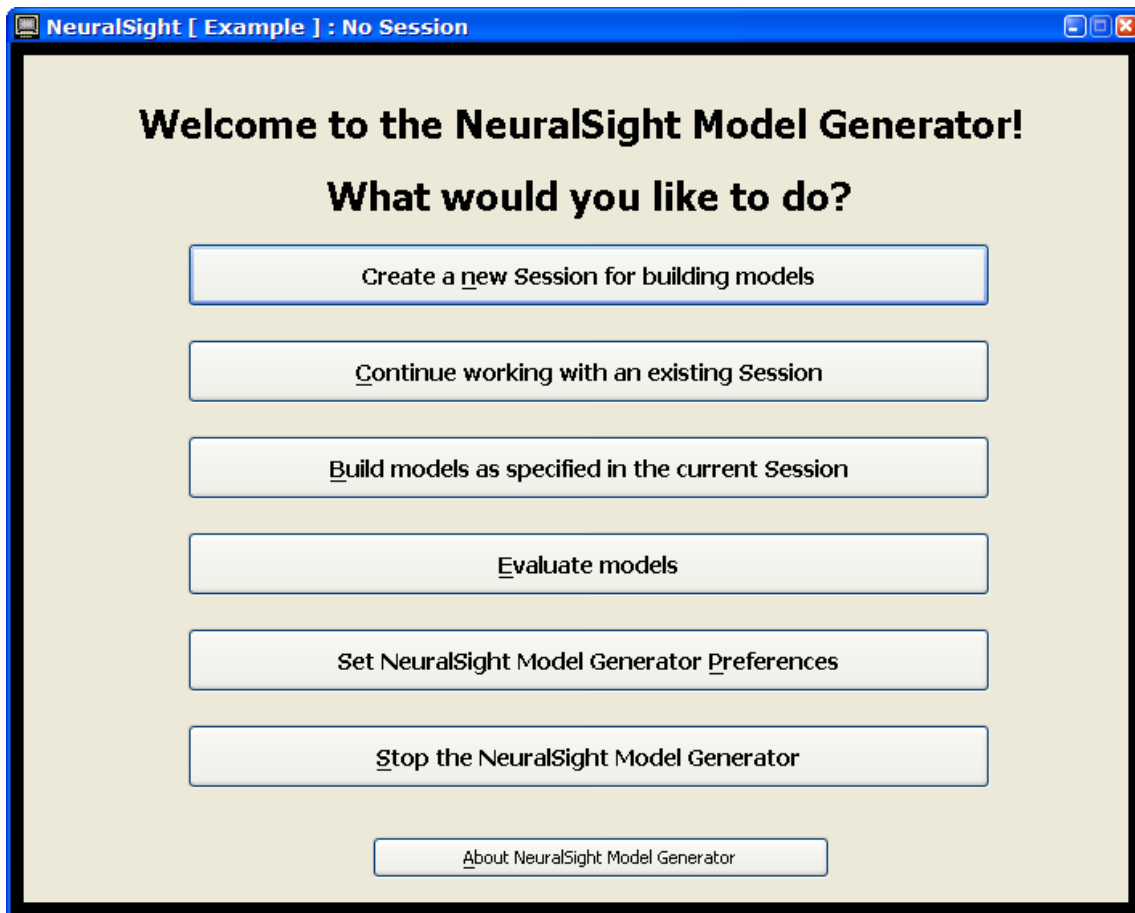
Note that with the traditional approach almost every step requires analysts to make decisions about model development that cannot typically be made in real-world circumstances without considerable trial and error effort – while with *NeuralSight* an analyst simply clicks a small number of buttons or enters file names or easy-to-define quantities like the number of models to build. *NeuralSight* handles all the technical details of constructing and training neural network models in a highly automated yet principled manner!

*In short, NeuralSight can shave weeks, possibly months off initial model development! And even greater savings accrue when the effort required to update models built using traditional approaches is factored into application modeling life-cycle considerations.*

## Using NeuralSight

The *NeuralSight* Graphical User Interface (GUI) is implemented in Java™ – it runs on a wide variety of computer systems. *NeuralSight* is currently available for Microsoft® Windows® and Red Hat® Linux operating systems for Intel® x86-architecture computers (please contact NeuralWare to discuss NeuralSight availability for other operating systems and architectures). The screen shots which follow illustrate how some of the key features of *NeuralSight* are implemented.

All ***NeuralSight*** operations are started through the main "Welcome" window. Each "button" in the window provides access to dialog boxes and other GUI elements which implement the operation described by the button label.



In ***NeuralSight***, models are logically organized in *Sessions*. A *Session* is a collection of models that all model the same phenomenon. However, each model in a *Session* represents a different set of modeling options (for example, the number of hidden units in the model) or modeling data records.

## Configuring a Session

Configuration is a straightforward process of specifying the location of modeling (and optionally validation) data files, establishing stopping criteria (hours or number of models adopted), specifying which records in the files should be used for modeling, specifying model adoption criteria (for example, minimum acceptable

Average Classification Rate, or minimum acceptable *R Correlation*), and specifying a small number of neural network training options, from which a specific subset will be used to build each individual model.

The steps in Session configuration are organized as a series of tabbed property pages. As a *Session* is configured, each successive tab is accessible (enabled) only after valid parameters have been entered on the preceding tab. The General tab is enabled when the *Session Configuration* dialog box first appears.

Logical model and folder naming conventions

Support for multiple types of models

The screenshot shows the 'Session Configuration' dialog box for a 'Self-Organizing Map'. It features several sections:
 

- Names:** Session Name (Example), Model Name (Model).
- Model Type:** Radio buttons for Prediction, Classification, and Self-Organizing Map (selected).
- Model and Data Files:** Fields for Folder for Session, Folder for Adopted Models, and Modeling Data File, each with a 'Select...' button. Includes checkboxes for 'Data Row Numbers from File' and 'Ignore Last Field'.
- Other Files:** Checkboxes for 'Create Predict Log File', 'Keep Predict Command Files', and 'Keep Intermediate Modeling Data Files'.
- Stopping Criteria:** Input fields for 'Maximum Model Building Time' (1) and 'Maximum Number of Adopted Models' (1).
- Categories:** A table with columns 'Category' and 'Upper Bound'. The table contains 10 rows with values from 0.1 to 0.9. Below the table is a 'Number of Categories' field set to 10 and an 'Update Table' button.

 Navigation buttons at the bottom include '< Back', 'Next >', 'OK', 'Apply', and 'Cancel'. Red arrows point from the text boxes above to specific elements in the dialog.

Time or performance-based stopping criteria

Data-driven (Classification) or user-defined categories

All Model-building options are on one page – NeuralSight defaults are suitable for most modeling efforts.

The screenshot shows the 'Session Configuration' dialog box for 'Classification', specifically the 'Set Build Options' tab. It includes:
 

- Do Not Execute Advanced Options:** A checkbox that is currently unchecked.
- Standard Options:**
  - Cascaded Variable Selection:** A checkbox that is unchecked.
  - Data Noise:** Checkboxes for Moderate, Noisy, and Very Noisy (all checked).
  - Data Transformation:** Checkboxes for Superficial, Moderate, and Comprehensive (all checked).
  - Variable Selection:** Checkboxes for Off and Comprehensive (both checked).
  - Network Search:** Checkboxes for Comprehensive and Exhaustive (both checked).
- Advanced Options:**
  - Hidden Layer Size:** Checkboxes for 15, 1, 2; 25, 1, 2; and 40, 5, 5 (all checked).
  - Weight Decay:** Checkboxes for .00025, .00250, and .02500 (all checked).
  - Hidden Noise:** Checkboxes for 5, 15, 25, and 40 (all checked).
  - Quantization:** Checkboxes for 50, 100, 200, and 500 (all checked).
  - Population Factor:** Checkboxes for 0.5, 2.0, and 4.0 (2.0 is checked).
  - Hidden Layer Transfer Function:** Checkboxes for Sigmoid, Sine, and Gauss (Sine and Gauss are checked).
  - Output Layer Transfer Function:** Checkboxes for Sigmoid and Linear (both unchecked).
  - Evaluation Function:** Checkboxes for Root Mean Power, RMS, and Common Mean Correlation (all unchecked).
  - Accuracy:** Radio buttons for 20%, 10%, and 5% (20% is selected).

 Navigation buttons at the bottom include '< Back', 'Next >', 'OK', 'Apply', and 'Cancel'. A red arrow points from the text above to the 'Set Build Options' tab.

After a *Session* is configured, model building can commence. Models are trained and evaluated with respect to performance criteria specified by the modeler (for prediction and classification models) or after a specific number of models have been built (SOM models). Models can be built on multiple threads if appropriate hardware is available, and the build process can be interrupted and restarted.

*NeuralSight* efficiently processes very large files with many fields. It supports filtering records based on descriptive information contained in each record. Modeling and validation datasets can be partitioned to include precise distributions of data values to assist in fine-tuning model performance. *NeuralSight* also maintains a detailed account of the specific records used in model building. This allows examining characteristics of the modeling and/or validation records that produced specific prediction or classification results, or that produced the associative relationships discovered by the SOM algorithm. It also provides a convenient audit trail if *NeuralSight* models are used in applications that require rigorous documentation of the modeling process.

When prediction or classification model training ends an error distribution for modeling and validation datasets is generated

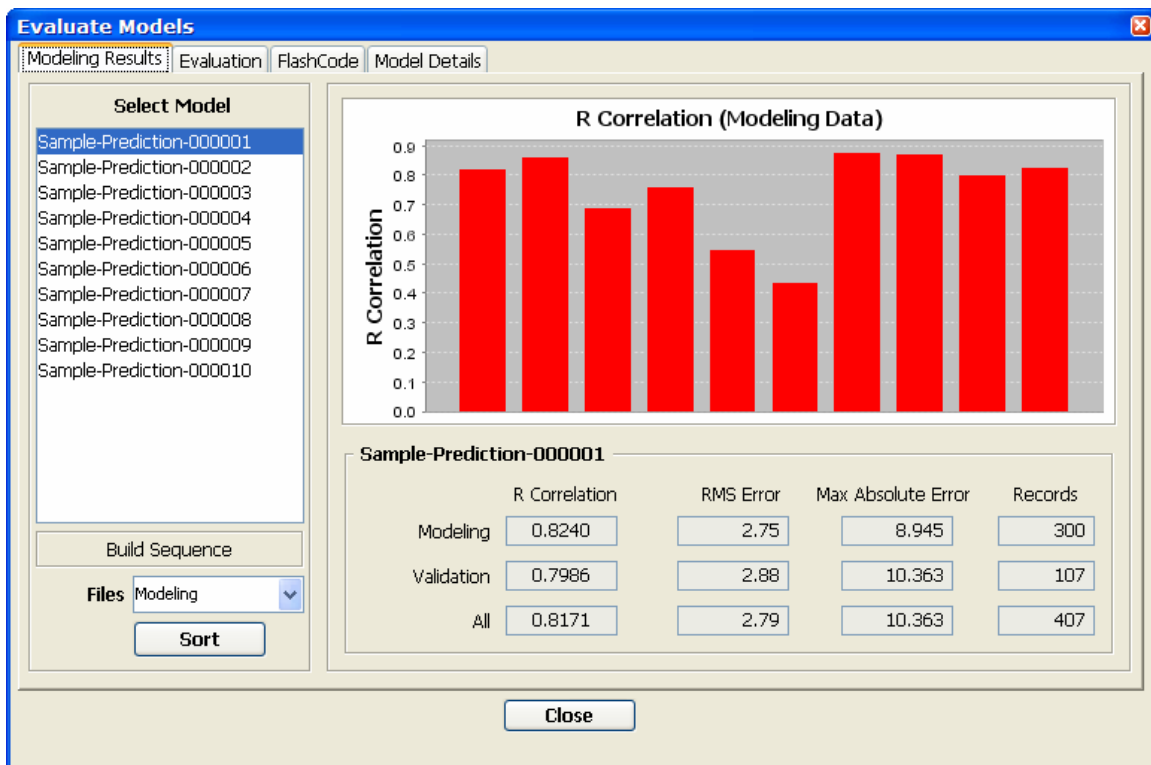
The screenshot shows the 'Evaluate Models' window with the following components:

- Model List:** A list of models under the 'Select Model' header, including 'Example-Model-000008' through 'Example-Model-000003'. Below the list is an 'Accuracy Average' field and a 'Sort' button.
- Error Distribution by Class Table:**

Class	Accuracy	SEVERE	MODERATE	ALARM	NORMAL	Total
SEVERE	0.950	19	1	0	0	20
MODERATE	0.760	6	38	6	0	50
ALARM	0.700	2	7	21	0	30
NORMAL	1.000	0	0	0	6	6
Total	0.792	27	46	27	6	106
Average	0.853					
- Buttons:** 'Close' button at the bottom center.

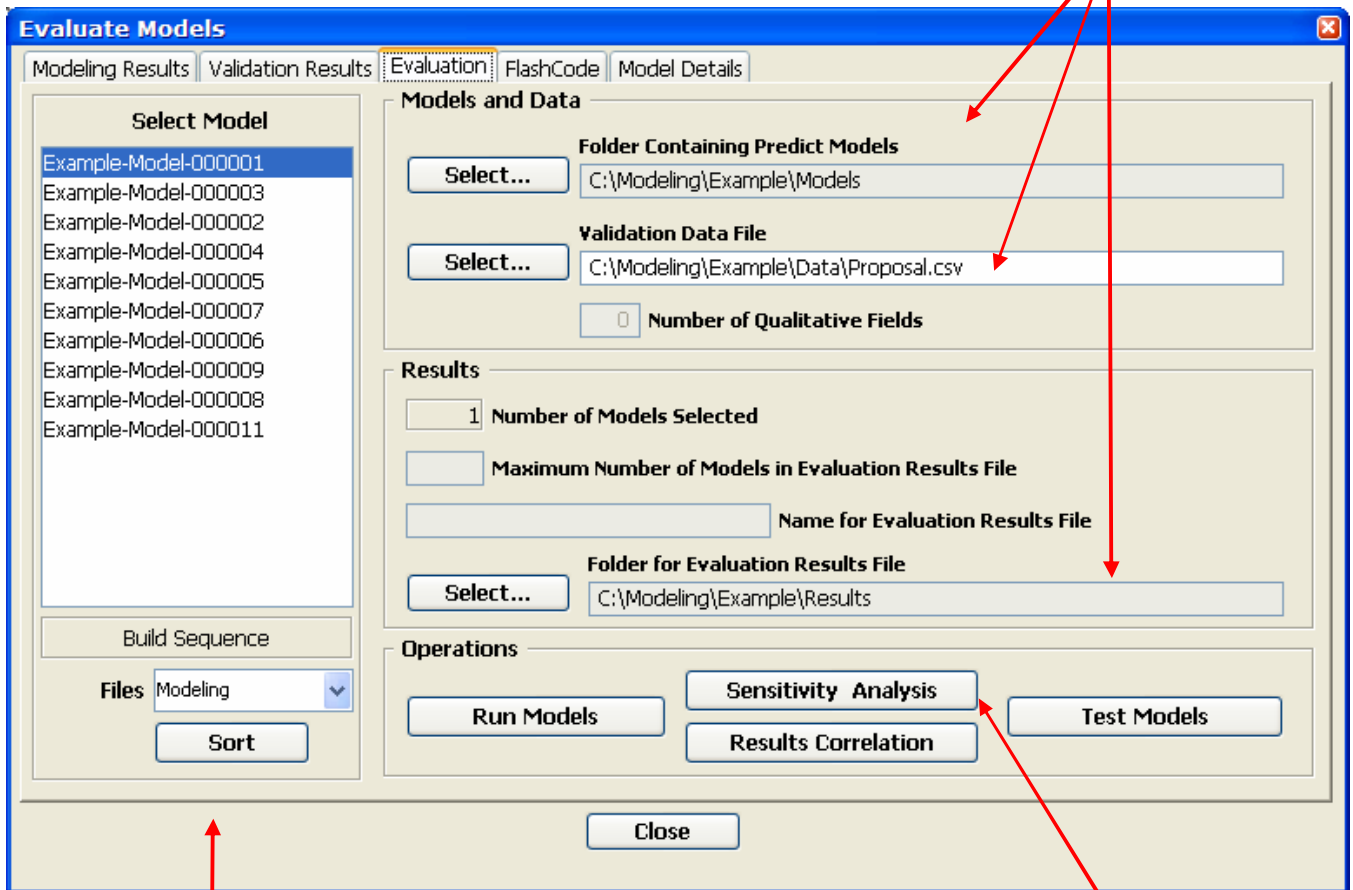
Annotations in the image include a red box at the top containing the text 'When prediction or classification model training ends an error distribution for modeling and validation datasets is generated' with arrows pointing to the 'Modeling Results' tab and the error distribution table. Another red box at the bottom contains the text 'The list of models can be ranked by performance based on multiple criteria' with an arrow pointing to the 'Sort' button.

Similarly, a Session that builds Prediction models produces plots for R Correlation or RMS Error, as illustrated below.



*NeuralSight* provides a simple interface for specifying performance standards that prediction or classification models must meet to be adopted – that is, retained as a candidate neural network model in a collection of models built during an automated model building *Session*. After a model building *Session* ends (either due to elapsed time or to a sufficient number of adopted models), all prediction or classification models that were adopted can be easily ranked to help decide which model or combination of models is most suitable to deploy. SOM visualization capabilities permit drilling down into SOM nodes to gain insight into characteristics of the records assigned to each node, which in turn guides selecting the SOM model or models to deploy.

The Evaluation tab provides additional facilities for evaluating model performance – models can be executed using other datasets, with results written to Excel-compatible .csv files

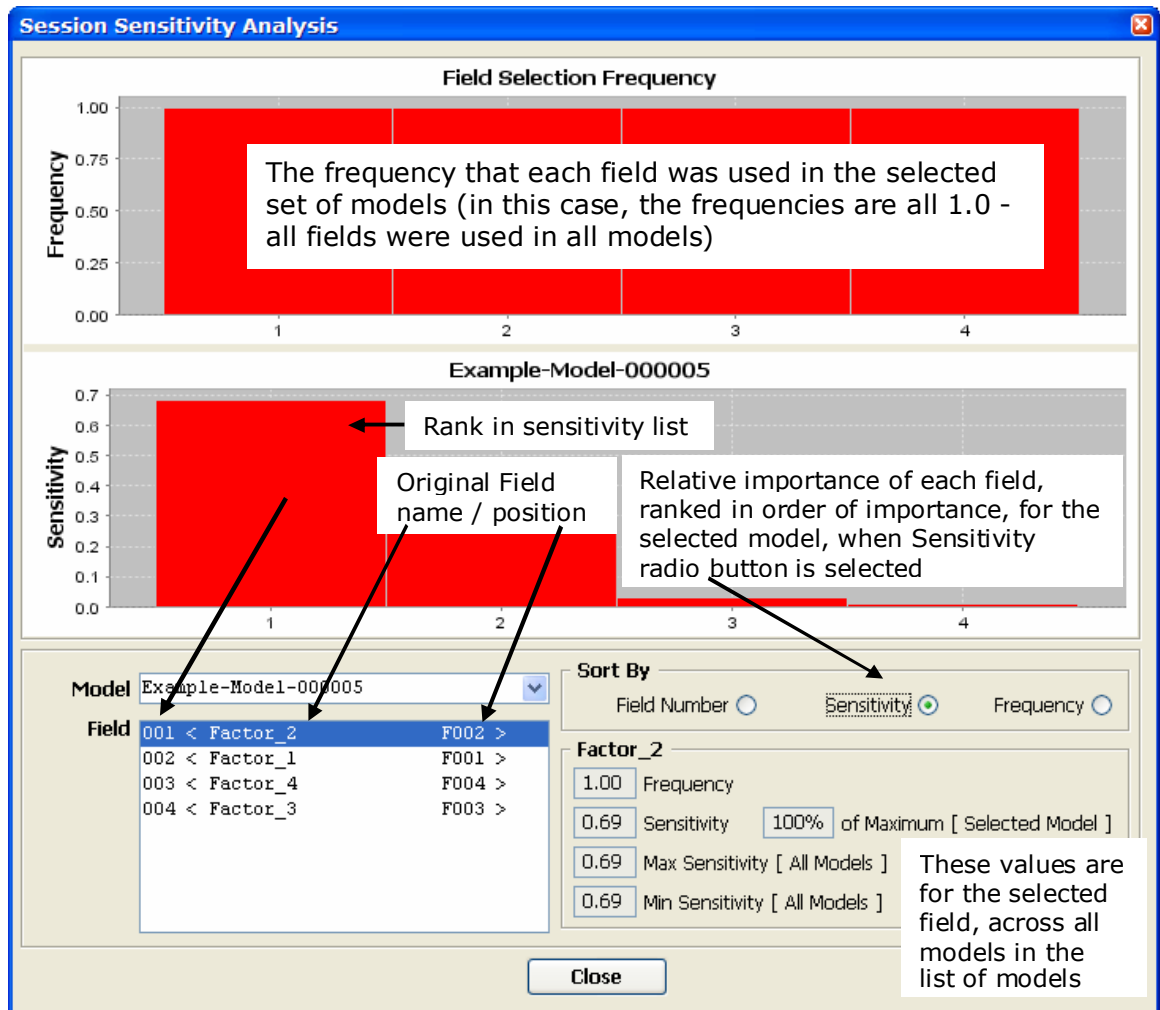


Models can be evaluated with respect to their performance on Modeling data only, Validation data only, or All available data

Sensitivity Analysis is a powerful technique to identify which input fields have the most influence on model performance (refer to the image on the next page)

## Sensitivity Analysis

*Sensitivity Analysis* charts show the frequency that input fields were used in the models trained in a *Session*, and the impact of input fields on model performance for the selected model



At the same time the *Sensitivity Analysis* chart is generated, a .csv file is created that maps sensitivity values to variables used in each model, so that additional comparisons of model behavior can easily be performed in Excel or other applications.

## Model Details

The **Model Details** tab indicates the transforms that the Predict Engine applied, as well as the results of the Variable Selection process. When combined with the information generated by Sensitivity Analysis, a complete picture of the structure and performance of every model is available.

The Architecture frame shows the high-level effects of data transformations, variable selection, and cascade correlation construction of the neural network.

The Fields frame shows all the Transforms that were applied to each field in the selected model, and the status of each Transform (Active Transforms are used in the final model).

**Evaluate Models**

Modeling Results | Validation Results | Evaluation | FlashCode | **Model Details**

**Architecture [ Example-Model-000005 ]**

5 Fields -> 4 Inputs -> 2 Hidden -> 4 Outputs

**Fields**

Number	Type	Name	Transforms	Status	Reason
1	I	Factor_1		Active	
2	I	Factor_2		Active	
3	I	Factor_3		Active	
4	I	Factor_4		Active	
5	O	DIAGNOSIS			
			alarm	Active	
			moderate	Active	
			normal	Active	
			severe	Active	

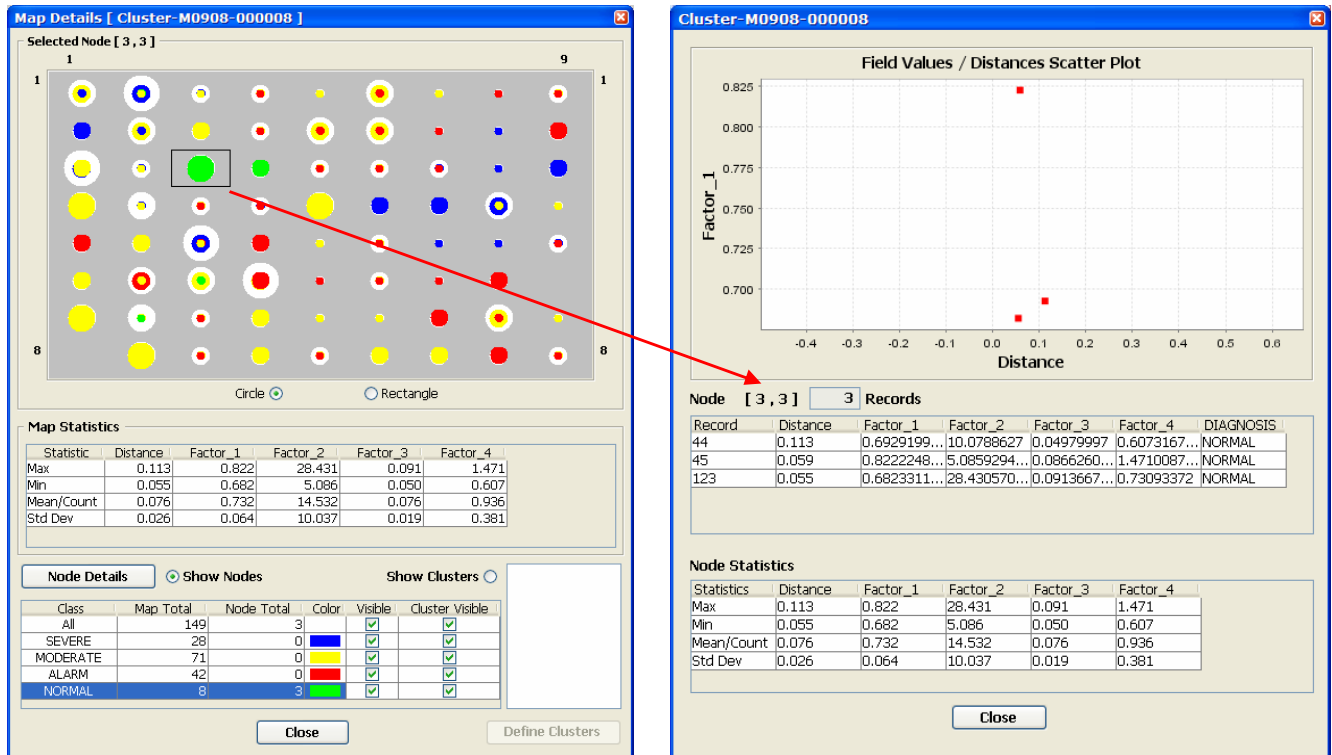
Show Active Transforms  Show All Transforms

Close

Models in the **Select Model** list can be sorted using the same options available on other tabs. You can choose whether the items in the **Fields** list represent only Fields/Transforms that are *Active*, as illustrated above, or you can view all Transforms that Predict tried.

## Evaluating SOM Models

The screen shots below illustrate how SOM results are presented. The image on the left is an overview of the distribution of records across the entire map. The relative size of circles indicates the relative number of records assigned to nodes. The image on the right provides detailed statistics about specific records assigned to the selected node.



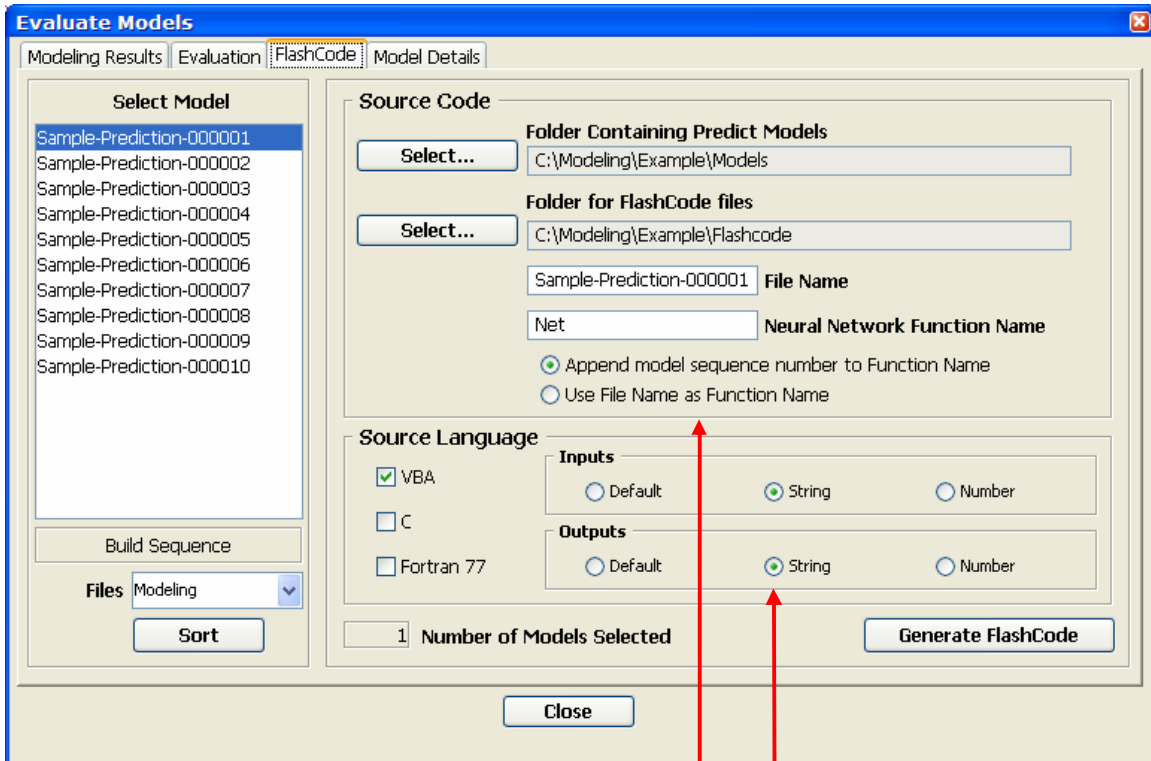
The tables present basic statistics for the entire map and each individual node in the map; the Node table also lists each record assigned to the selected node. The scatter plot of distance against field values shows at a glance if there are any noticeable potential outlier values in a node (the scatter plot changes when a different column label in the Node table is clicked).

## Deploying Models

Models created by NeuralSight are fully compatible with the Predict Run-Time Engine. Prediction and classification models can be directly used in decision support systems, either by linking the decision support application to the Predict Run-Time Engine and simply loading the selected model, or by using the NeuralSight FlashCode feature to create source code that can be compiled and embedded in a microcontroller.

Associations and relationships discovered by SOM clustering can be directly used in classification or search-by-association applications linked to the Run-Time Engine, or they can be used as a pre-processing filter for a standard *NeuralSight* classification model.

*NeuralSight* supports generating C, Visual Basic, or Fortran source code representations of trained networks. Licenses for use of source code in deployed applications are negotiated on a case-by-case basis.



The name of the function that represents the network can be specified, and a sequence number automatically appended, so that multiple models can be used in the same application without causing naming conflicts.

Depending on the data acquisition environment expected at execution time, the format of input and output values can be specified to be Strings to provide maximum run-time flexibility.