Copyright Notice
Copyright © 2008-2015, Palisade Corporation

Trademark Acknowledgments
TopRank, BestFit, RISKview and Palisade are registered trademarks of Palisade Corporation
RISK is a trademark of Parker Brothers, Division of Tonka Corporation, and is used under license.
Microsoft, Excel and Windows are registered trademarks of Microsoft Corporation.
# Table of Contents

## Chapter 1: Welcome to NeuralTools

- Work Where You're Comfortable ...................................................... 1
- NeuralTools Analyses ......................................................................... 2
- NeuralTools Data Management ........................................................ 2
- NeuralTools Reporting........................................................................ 3
- Data Access and Sharing ..................................................................... 3

## Chapter 2: Getting Started

- About This Version .............................................................................. 5
- NeuralTools Professional and Industrial Versions ................................. 5
- Working with your Operating Environment ........................................... 5
- If You Need Help ................................................................................. 6
- NeuralTools System Requirements ..................................................... 8

## Installation Instructions ................................................................. 9

- General Installation Instructions ....................................................... 9
- The DecisionTools Suite .................................................................... 9
- Software Activation ............................................................................. 9

## Instructional Materials .................................................................. 10

- Example Spreadsheets ....................................................................... 10
- Quick Start Tutorials ........................................................................ 10
- Guided Tour Videos .......................................................................... 10
- Examples and Videos for the XDK .................................................... 10

## Chapter 3: An Overview of NeuralTools

- Why Neural Nets? .............................................................................. 11
- NeuralTools and Neural Nets .......................................................... 12
- NeuralTools Ribbon .......................................................................... 13
- Data Sets and the Data Set Manager .................................................. 13
- Training a Neural Net ........................................................................ 15
- Testing a Network ............................................................................ 19
- Prediction ............................................................................................. 20
- NeuralTools Reports and Charts ......................................................... 22
- NeuralTools Utilities ........................................................................... 24
- Using NeuralTools with StatTools and Evolver ................................... 24
# Chapter 4: NeuralTools Command Reference

**NeuralTools Icons**
- Icons in NeuralTools Dialogs

**NeuralTools Commands**
1. Data Group
2. Neural Nets Group
3. Help Group
4. Data Set Manager Command
5. Data Viewer Command
6. Train Command
7. Test Command
8. Predict Command
9. Application Settings Command
10. Neural Net Manager Command
11. Missing Data Utilities Command
12. Testing Sensitivity Command

# Chapter 5: More on Neural nets

**Neural Net Basics**
- Neural Nets versus Statistical Methods
- The Structure of a Neural Net
- Numeric and Category Prediction
- Training a Net
- Computer Processing of Neural Nets
- Types of Neural nets
- Input Transformation

**Multi-Layer Feedforward Nets**
- MLF Architecture
- MLF Net Training

**Generalized Regression Neural Nets and Probabilistic Neural Nets**
- Generalized Regression Neural Nets
- Probabilistic Neural Nets

**Comparison of MLF Nets to PN/GRN Nets**

**Recommended Readings**

**Index**
Chapter 1: Welcome to NeuralTools

NeuralTools gives Microsoft Excel, the industry-standard data analysis and modeling tool, a new and powerful modeling toolset. NeuralTools is a Microsoft Excel neural nets add-in, allowing you to analyze data in Excel worksheets and work in the familiar Microsoft Office environment. By combining a powerful data set manager, along with state-of-the-art neural net algorithms, NeuralTools brings you the best of two worlds: Microsoft Office ease-of-use and reporting, with robust, accurate predictions from neural nets.

Work Where You're Comfortable

If you know Excel, you'll know NeuralTools. NeuralTools works just like Excel, with ribbons, menus, and custom worksheet functions, all inside Excel. Unlike stand-alone neural net software, there is no steep learning curve and upfront training costs with NeuralTools, because you work just as you are used to working in Excel. Your data and variables are in Excel spreadsheets. You can utilize standard Excel formulas for calculations, along with Excel sorting, pivot tables, and other data analysis tools. Reports and charts from your analyses are in standard Excel format and can utilize all of Excel's built-in formatting capabilities.
NeuralTools Analyses

Neural nets are capable of learning complex relationships in data. By mimicking the functions of the brain, they can discern patterns in data and then extrapolate predictions when given new data. The problems neural nets are used for can be divided in two general groups:

- **Classification Problems:** These are problems in which you are trying to determine what type of category an unknown item falls into. Examples include medical diagnoses and prediction of credit repayment ability.

- **Numeric Problems:** These are problems where you need to predict a specific numeric outcome. Examples include stock price forecasting and predicting the level of sales during a future time period.

Neural nets are used in a broad range of applications including stock market prediction, credit and loan risk assignment, credit fraud detection, sales forecasting, general business forecasting, investment risk, medical diagnosis, control systems, and others.

NeuralTools includes the latest neural net algorithms to make the best predictions on both classification problems (called category prediction in NeuralTools) and numeric problems.

*Note: The more formal term “neural network” is often abbreviated to “neural net,” as will be done in this manual.*

NeuralTools Data Management

NeuralTools provides a comprehensive data set and variable manager right in Excel, similar to that provided with StatTools, Palisade's statistics add-in for Excel. You can define any number of data sets, each with the variables you want to analyze, directly from your data in Excel. NeuralTools intelligently assesses your blocks of data, suggesting variable names and types as well as data locations. Your data sets and variables can reside in different workbooks and worksheets, allowing you to organize your data as you see fit. Then, you train neural nets that refer to your variables, instead of re-selecting your data over and over again in Excel.
**NeuralTools Reporting**

Excel is great for reports and graphs, and NeuralTools makes the most of this. NeuralTools uses Excel-format graphs, which can be easily customized for new colors, fonts, and added text. Report titles, number formats, and text can be changed just as in any standard Excel worksheet. You can drag and drop tables and charts from NeuralTools reports straight into documents in other applications.

NeuralTools Industrial also includes Live Prediction, where predicted values are calculated as new data is entered into your Excel worksheet. This live calculation happens automatically, just like other Excel recalculations.

**Data Access and Sharing**

Excel has great data import features, so bringing your existing data into NeuralTools is easy. You can use standard Excel capabilities to import data from Microsoft SQL Server, Oracle, Microsoft Access, or any other ODBC compliant database. You can also load data from text files or other applications. If you can import it into Excel, you can use it with NeuralTools!

NeuralTools saves all of its results and data in Excel workbooks. Just like any other Excel file, you can send your NeuralTools results and networks to colleagues anywhere. Sharing couldn't be easier!
Chapter 2: Getting Started

This chapter describes the contents of the NeuralTools package and shows how to install NeuralTools and attach it to your copy of Microsoft Excel 2007 or higher.

About This Version

This version of NeuralTools can be used with Microsoft Excel 2007 or higher.

NeuralTools Professional and Industrial Versions

NeuralTools is available in two versions: Professional and Industrial. The differences are as follows:

• Data sets in NeuralTools Professional are limited to 1000 cases. Data sets in NeuralTools Industrial are limited only by available memory.

• Live Prediction, where predicted values are calculated as new data is entered into your Excel worksheet, is provided only in NeuralTools Industrial.

Working with your Operating Environment

This manual assumes that you have a general knowledge of the Windows operating system and Excel. In particular:

• You are familiar with your computer and using the mouse.

• You are familiar with terms such as icons, click, double-click, menu, ribbon, and window.

• You understand basic concepts such as directory (folder) structures and file naming.
If You Need Help

Technical support is provided free of charge for all registered users of NeuralTools with a current maintenance plan, or is available on a per incident charge. To ensure that you are a registered user of NeuralTools, please register online at


If you contact us by telephone, please have your serial number and User’s Guide ready. We can offer better technical support if you are in front of your computer and ready to work.

Before Calling

Before contacting technical support, please review the following checklist:

- Have you consulted the relevant sections of this online manual?
- Have you watched the online Quick Start videos available from the NeuralTools Welcome screen?
- Have you read the README.WRI file? It contains current information on NeuralTools that might not be included in the manual.
- Can you duplicate the problem consistently? Can you duplicate the problem on a different computer or with a different model?
- Have you consulted our Web site, http://www.palisade.com? This Web site contains the latest FAQ (a searchable database of tech support questions and answers). We recommend visiting our Web site regularly for all the latest information on NeuralTools and other Palisade software.
Palisade Corporation welcomes your questions, comments or suggestions regarding NeuralTools. Contact our technical support staff using any of the following methods:

- Email us at support@palisade.com.
- Telephone us at (607) 277-8000 any weekday from 9:00 AM to 5:00 PM, EST. Follow the prompt to reach technical support.
- Fax us at (607) 277-8001.
- Mail us a letter at:

  **Technical Support**  
  Palisade Corporation  
  798 Cascadilla St.  
  Ithaca, NY 14850 USA

If you want to contact Palisade Europe:

- Email us at support@palisade-europe.com.
- Telephone us at +44 1895 425050 (UK).
- Fax us at +44 1895 425051 (UK).
- Mail us a letter at:

  **Palisade Europe**  
  31 The Green  
  West Drayton  
  Middlesex  
  UB7 7PN  
  United Kingdom

If you want to contact Palisade Asia-Pacific:

- Email us at support@palisade.com.au.
- Telephone us at +61 2 9252 5922 (AU).
- Fax us at +61 2 9252 2820 (AU).
- Mail us a letter at:

  **Palisade Asia-Pacific Pty Limited**  
  Suite 404, Level 4  
  20 Loftus Street  
  Sydney NSW 2000  
  Australia

Regardless of how you contact us, please include the product name, version and serial number. The exact version can be found by selecting Help About from the Help menu on the NeuralTools ribbon.
Student Version

Telephone support is not available with the student version of NeuralTools. If you need help, we recommend the following alternatives:

- Consult with your professor or teaching assistant.
- Go to http://www.palisade.com for answers to frequently asked questions.
- Contact our technical support department via e-mail or fax.

NeuralTools System Requirements

System requirements for NeuralTools for Microsoft Excel for Windows include:

- Microsoft Windows XP or higher.
- Microsoft Excel 2007 or higher.
General Installation Instructions

The Setup program copies the NeuralTools system files into a folder you specify on your hard disk. To run the Setup program in Windows XP or higher:

1. Double-click the NeuralTools Setup.exe (or the DTSuite Setup.exe) from your download or installation CD.

2. Follow the Setup instructions on the screen.

If you encounter problems while installing NeuralTools, verify that there is adequate space on the drive to which you’re trying to install. After you have freed up adequate space, try rerunning the installation.

If you wish to remove NeuralTools from your computer, use the Control Panel’s Add/Remove Programs utility and select the entry for NeuralTools (or the DecisionTools Suite).

The DecisionTools Suite

NeuralTools for Excel is a member of the DecisionTools Suite, a set of products for risk and decision analysis. The default installation procedure of NeuralTools puts NeuralTools in a subfolder of a main Program Files\Palisade (or Program Files (x86)\Palisade) folder.

One subfolder of this Palisade folder will be the NeuralTools folder (by default called NeuralTools7). This folder contains the program file (NeuralTools.xla), plus example models and other files necessary for NeuralTools to run. Another subfolder of the Palisade folder is the SYSTEM folder. This contains files required by every add-in in the DecisionTools Suite, including common help files and program libraries.

Software Activation

Activation is a one-time license verification process that is required for your Palisade software to run as a fully licensed product. An activation ID such as DNA-6438907-651282-CDM is on your printed/emailed invoice. If you enter your activation ID during installation, your software is activated at the end of the installation process and no further user action is required. If you need to activate your software after installation, you should select the License Manager command on the NeuralTools Help menu.
Instructional Materials

The NeuralTools package includes a number of instructional materials to help you learn NeuralTools features and how to model with NeuralTools.

Example Spreadsheets

NeuralTools includes about 25 example spreadsheets. Some of these examples illustrate specific features of NeuralTools, whereas many others illustrate potential uses of NeuralTools in various fields. These examples not only help you learn how to use NeuralTools, but they illustrate how extensively NeuralTools can be applied.

You can find these example spreadsheets from the NeuralTools Help menu. When you click its Example Spreadsheets command, an “example file list” file opens in Excel. This file contains links to all of the example files.

Quick Start Tutorials

From the NeuralTools Welcome screen, which you see when you launch NeuralTools or you can access from the NeuralTools Help menu, you can click the Quick Start link to see a series of short videos that lead you through the basic features of NeuralTools. These Quick Start videos are intended for beginners, but they are sufficient to get you started creating your own NeuralTools analyses.

Guided Tour Videos

A series of Guided Tour videos is also available from the NeuralTools Welcome Screen. These are more in-depth videos, and they lead you through practically all of the NeuralTools features, from simple to more complex.

Examples andVideos for the XDK

You might want to learn how to use the macro language of Excel, Visual Basic for Applications (VBA), to automate NeuralTools procedures. NeuralTools provides an Excel Developer Kit (XDK) for doing this. You can learn about the XDK from the Developer Kit (XDK) command on the NeuralTools Help menu. There you will see links to example spreadsheet models of NeuralTools automation, as well as additional instructional videos on using the XDK.
NeuralTools provides you with powerful neural net capabilities in an environment that you are familiar with: Microsoft Excel. NeuralTools procedures, such as defining data sets, training and testing neural nets, and predicting values using trained networks, can be run on your data in Excel, and the reports and charts from your analyses are created in Excel.

**Why Neural Nets?**

Neural nets are capable of learning complex relationships in data. By mimicking the functions of the brain, they can discern patterns in data, and then extrapolate predictions when given new data. The problems neural nets are used for can be divided in two general groups:

- **Classification Problems:** These are problems in which you are trying to determine what type of category an unknown item falls into. Examples include medical diagnoses and prediction of credit repayment ability.

- **Numeric Problems:** These are problems where you need to predict a specific numeric outcome. Examples include stock price forecasting and predicting the level of sales during a future time period.

NeuralTools comes with example files that show how neural nets can be applied to different prediction problems. (You can find these under the NeuralTools Help menu.)
NeuralTools and Neural Nets

When using NeuralTools, neural nets are developed and used in four steps:

- **Data Preparation.** The data you use in NeuralTools is defined in data sets. A Data Set Manager is used to define data sets so they can be used over and over again with your neural nets.

- **Training.** During training, a neural net is generated from a data set comprised of cases with known output values. This data often consists of historical cases for which you know the values of the output (also called the dependent variable).

- **Testing.** During testing, a trained neural net is used to see how well it does at predicting known output values. The data used for testing is usually a subset of your historical data, but was not used in training the network. After testing, the performance of the network is measured by statistics such as the percentage of the known output values it correctly predicted.

- **Prediction.** In this last stage, a trained neural net is used to predict unknown output values. Once trained and tested, the network can be used as needed to predict outputs for new case data.

Training and testing are performed in an iterative, sometimes time-intensive process. Typically, you might need to train several different times with different settings to generate a neural net that tests best. Once you have your “best net,” you can use it for predicting.

Now, let's look at how NeuralTools works in Excel and how you perform the above four steps. This discussion uses the Abalone Age Prediction.xlsx example file, where the goal is to predict the numeric variable Age.
NeuralTools Ribbon

Once you have installed NeuralTools, its commands will appear on the NeuralTools ribbon.

Data Sets and the Data Set Manager

Data in NeuralTools is structured around cases and variables. You work with a data set, or a set of statistical variables, located in contiguous columns with variable names in the first row of the data set. Each row in the data set is a case (also called a record or an observation). The variables are classified as independent or dependent, depending on their role in the prediction process. The dependent variable is the variable to be predicted. The independent variables are the “explanatory” variables used to predict the dependent variable. Cases where the dependent variable values are known are used to train and test a neural net. Cases where the dependent variable values are unknown (missing) are used for prediction.

<table>
<thead>
<tr>
<th>Sex</th>
<th>Length</th>
<th>Diameter</th>
<th>Height</th>
<th>Whole Weight</th>
<th>Shucked Weight</th>
<th>Viscera Weight</th>
<th>Shell Weight</th>
<th>Rings</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>0.54</td>
<td>0.42</td>
<td>0.14</td>
<td>0.805</td>
<td>0.389</td>
<td>0.1725</td>
<td>0.14</td>
<td>11</td>
<td>12.5</td>
</tr>
<tr>
<td>I</td>
<td>0.445</td>
<td>0.39</td>
<td>0.11</td>
<td>0.4235</td>
<td>0.182</td>
<td>0.0765</td>
<td>0.14</td>
<td>9</td>
<td>10.5</td>
</tr>
<tr>
<td>I</td>
<td>0.535</td>
<td>0.4</td>
<td>0.125</td>
<td>0.775</td>
<td>0.388</td>
<td>0.208</td>
<td>0.2055</td>
<td>8</td>
<td>9.5</td>
</tr>
<tr>
<td>I</td>
<td>0.52</td>
<td>0.45</td>
<td>0.135</td>
<td>0.536</td>
<td>0.358</td>
<td>0.2565</td>
<td>0.2565</td>
<td>10</td>
<td>11.5</td>
</tr>
<tr>
<td>F</td>
<td>0.715</td>
<td>0.535</td>
<td>0.185</td>
<td>1.56</td>
<td>0.6655</td>
<td>0.383</td>
<td>0.405</td>
<td>11</td>
<td>12.5</td>
</tr>
<tr>
<td>F</td>
<td>0.595</td>
<td>0.44</td>
<td>0.135</td>
<td>0.964</td>
<td>0.5005</td>
<td>0.1715</td>
<td>0.2575</td>
<td>10</td>
<td>11.5</td>
</tr>
<tr>
<td>I</td>
<td>0.155</td>
<td>0.105</td>
<td>0.05</td>
<td>0.0175</td>
<td>0.005</td>
<td>0.0005</td>
<td>0.005</td>
<td>4</td>
<td>5.5</td>
</tr>
<tr>
<td>I</td>
<td>0.355</td>
<td>0.27</td>
<td>0.105</td>
<td>0.271</td>
<td>0.1425</td>
<td>0.0525</td>
<td>0.0735</td>
<td>9</td>
<td>10.5</td>
</tr>
<tr>
<td>I</td>
<td>0.455</td>
<td>0.355</td>
<td>0.11</td>
<td>0.474</td>
<td>0.023</td>
<td>0.1305</td>
<td>0.12</td>
<td>7</td>
<td>8.5</td>
</tr>
<tr>
<td>M</td>
<td>0.55</td>
<td>0.425</td>
<td>0.135</td>
<td>0.975</td>
<td>0.6775</td>
<td>0.243</td>
<td>0.335</td>
<td>13</td>
<td>34.5</td>
</tr>
<tr>
<td>M</td>
<td>0.55</td>
<td>0.4</td>
<td>0.18</td>
<td>0.6205</td>
<td>0.4405</td>
<td>0.159</td>
<td>0.225</td>
<td>10</td>
<td>11.5</td>
</tr>
<tr>
<td>M</td>
<td>0.33</td>
<td>0.235</td>
<td>0.695</td>
<td>0.1875</td>
<td>0.0735</td>
<td>0.045</td>
<td>0.06</td>
<td>7</td>
<td>8.5</td>
</tr>
</tbody>
</table>

The NeuralTools Data Set Manager allows you to define your data sets, variables, and cases. You can then use these predefined variables for training and testing neural nets, without having to re-select the data you want to analyze over and over. The data for training, testing, and prediction can be in a single data set, or they can be in different data sets. For example, one data set could contain the data for training and testing, with all dependent variable values known, and the data for prediction could be in another data set, with all dependent variable values missing. Alternatively, as in the Abalone example, the training data, the testing data, and the prediction data can all be in...
different data sets. (The above screenshot shows a subset of the training data.)

When you click the Data Set Manager icon on the NeuralTools ribbon, you see the Data Set Manager dialog. The following screenshot shows the results after defining the three data sets shown at the top. (In this example, each data set is in a separate worksheet.) Defining a data set is a very quick process. If a cell within the data range is already selected, NeuralTools will guess the data range, which is usually correct. Then you can give the data set a meaningful name and define the roles of the variables in the Variable Type column in the lower section. You can also decide whether you want to check the Apply Cell Formatting option. This applies blue formatting to the variable label row and other formatting to identify NeuralTools data sets.

Each variable in a data set corresponds to a range of Excel cells (a column), and NeuralTools supplies a range name to each variable’s range, the name that appears in the Variable Name column.
In NeuralTools, variables can be independent or dependent, and numeric or categorical (for example, Yes/No or Red/Green/Blue.) The Data Set Manager attempts to identify the type of each variable in your data set, but you can override this with your own selections. Note that the variable type can also be Unused, indicating that NeuralTools will ignore this variable in the analysis.

**Training a Neural Net**

After you have defined a data set that contains cases with known historical values, you can train a neural net using that data. There are different options that determine the type of network that will be generated by NeuralTools. Depending on the nature of your data, different network options will generate better performing trained networks—that is, networks that do a better job predicting values. The testing process, which is done following training, lets you see how well your trained network does at predicting output values.

To train a neural net, you must specify a data set that contains the data to be used during training. There are several ways to do this, depending on how your data sets are structured. If your data for training is in a separate data set, as in the Abalone example, you can use all of this data for training.
In contrast, if your training and testing data are in a single data set, NeuralTools allows you to withhold a certain percentage of the data for testing (20% is the default) and use the rest of the data for training. NeuralTools typically chooses which cases to use for training and which to use for testing through a random mechanism, but it provides other options that give you more control.

In addition, if prediction data (cases with missing values of the dependent variable) are in the same data set as the training and testing data, you can select to predict their values automatically. This means that when all of the data are in a single data set, you can train, test, and predict in a single analysis.

NeuralTools supports different neural net configurations to achieve the best possible predictions. For classification/category prediction (where the dependent variable is a category type), two types of networks are available: Probabilistic Neural Nets (PNN) and Multi-Layer Feedforward Networks (MLF). Numeric prediction can be performed using MLF networks, as well as Generalized Regression Neural Nets (GRNN), which are closely related to PNN networks. Finally, NeuralTools offers a Best Net Search option. This removes all of the guess work on your part, but it is slower because it tries several configurations simultaneously.
Once you have selected training and network configuration options, NeuralTools previews what it will perform during network training. Because training is the most time-intensive process in neural net modeling, it helps to review the training setup before proceeding. In particular, NeuralTools will try to identify any problems it has found in your data so you can correct them prior to training.
As NeuralTools proceeds with training a neural net on your data, it reports how well it is doing. Typically the net gets better and better as training proceeds. Graphs update to show NeuralTools’ progress during training.

Training stops when any of the stopping conditions you have set, such as maximum training time, are reached. If you have selected to automatically test the net or predict missing output values in your data set, this will be performed after training.
A summary training report shows how well your trained net performed, in this case, on the training data. Statistics such as % Bad Predictions show the percentage of cases in the training set for which the network predicted an output value that did not agree with the actual known value.

### Testing a Network

During testing, a trained neural net is tested to see how well it does at predicting known output values for cases not used in training. The testing data is usually a subset of your historical data with known output values. If the testing data and training data are in a single data sets, testing will occur automatically after the net has been trained. However, if the training and testing data are in separate data sets, as in the Abalone example, you must click Test on the NeuralTools ribbon to see the following dialog.
In this case, NeuralTools will match the variables in the testing data set with those in the training data as. As with training, NeuralTools will preview your testing setup prior to running.

Testing runs much faster than training. NeuralTools reports how well it did in predicting the known dependent variable values in the testing data. This helps you see whether the neural net is likely to be a good predictor when applied to cases with unknown output values.

### Testing Reports

<table>
<thead>
<tr>
<th>Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Net Information</strong></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Net Trained on Training Data</td>
</tr>
<tr>
<td>Configuration</td>
<td>GRNN Numeric Predictor</td>
</tr>
<tr>
<td>Location</td>
<td>This Workbook</td>
</tr>
<tr>
<td>Independent Category Variables</td>
<td>1 (Sex)</td>
</tr>
<tr>
<td>Independent Numeric Variables</td>
<td>7 (Length, Diameter, Height, Whole Weight, Shucked Weight, Viscera Weight, Shell Weight)</td>
</tr>
<tr>
<td>Dependent Variable</td>
<td>Numeric Var. (Age)</td>
</tr>
<tr>
<td><strong>Testing</strong></td>
<td></td>
</tr>
<tr>
<td>Number of Cases</td>
<td>500</td>
</tr>
<tr>
<td>% Bad Predictions (30% Tolerance)</td>
<td>10.0000%</td>
</tr>
<tr>
<td>Root Mean Square Error</td>
<td>2.247</td>
</tr>
<tr>
<td>Mean Absolute Error</td>
<td>1.560</td>
</tr>
<tr>
<td>Std. Deviation of Abs. Error</td>
<td>1.617</td>
</tr>
<tr>
<td><strong>Data Set</strong></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Testing Data</td>
</tr>
<tr>
<td>Number of Rows</td>
<td>500</td>
</tr>
<tr>
<td>Manual Case Tags</td>
<td>No</td>
</tr>
<tr>
<td>Variable Matching</td>
<td>Automatic</td>
</tr>
<tr>
<td>Indep. Category Variables Used</td>
<td>Names from training</td>
</tr>
<tr>
<td>Indep. Numeric Variables Used</td>
<td>Names from training</td>
</tr>
<tr>
<td>Dependent Variable</td>
<td>Numeric Var. (Age)</td>
</tr>
</tbody>
</table>

### Prediction

The final step is prediction, where you apply a trained neural net to new cases with missing values of the dependent variable. NeuralTools offers two methods for prediction: a **command-driven** method, and **Live Prediction** (Industrial version only), where new independent variable values for a case can be entered in your worksheet and NeuralTools will automatically calculate the predicted output value.
The following Prediction dialog, obtained by clicking the Predict icon on the NeuralTools ribbon, helps you set up the prediction process. You can predict just for cases with missing output values and optionally enable Live Prediction. Also, different trained nets can be used to see how predicted values differ.

As with training and testing, NeuralTools first previews the data and setup it will use for prediction. Then, predictions are reported to your worksheet in Excel.
Assuming you check the **Place Predicted Values Directly in Data Set** option, the predicted output values are shown next to the cases for which prediction is performed. In the screen here, predicted values are in pink.

<table>
<thead>
<tr>
<th></th>
<th>Sex</th>
<th>Length</th>
<th>Diameter</th>
<th>Height</th>
<th>Whole Weight</th>
<th>Shucked Weight</th>
<th>Viscera Weight</th>
<th>Shell Weight</th>
<th>Rings</th>
<th>Predicted Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>0.385</td>
<td>0.28</td>
<td>0.095</td>
<td>0.257</td>
<td>0.119</td>
<td>0.059</td>
<td>0.07</td>
<td>8.746968</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>0.40</td>
<td>0.365</td>
<td>0.125</td>
<td>0.4785</td>
<td>0.206</td>
<td>0.104</td>
<td>0.141</td>
<td>10.70997</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0.34</td>
<td>0.25</td>
<td>0.09</td>
<td>0.179</td>
<td>0.0795</td>
<td>0.033</td>
<td>0.055</td>
<td>8.204008</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When Live Prediction is enabled, NeuralTools automatically adds an Excel formula to the cell where the predicted value is shown. This formula *generates* the predicted value, so if you change independent variable values for a case, the predicted value will automatically be recalculated. Using Live Prediction you can simply type data for new cases directly in Excel and automatically generate a new prediction, without going through the Prediction dialog. For example, if the independent variable values for the first case are changed as shown, the predicted value automatically updates. As with any worksheet cell, you can reference a LivePrediction cell in any Excel formula.

Again, Live Prediction is available in the Industrial version only.

**NeuralTools Reports and Charts**

NeuralTools creates both Summary and Detailed Reports from training, testing, and prediction. **Summary Reports** are shown on their own worksheet and have overall information on Testing or Training. (Two of summary reports were shown above.) A **Detailed Report** provides information on a case-by-case basis and is shown next to the data being reported on.

<table>
<thead>
<tr>
<th>Shell Weight</th>
<th>Rings</th>
<th>Age</th>
<th>Prediction</th>
<th>Good/Bad</th>
<th>Residual</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.07</td>
<td>8</td>
<td>9.5</td>
<td>8.43</td>
<td>Good</td>
<td>1.02</td>
</tr>
<tr>
<td>0.065</td>
<td>11</td>
<td>12.5</td>
<td>8.28</td>
<td>Bad</td>
<td>4.22</td>
</tr>
<tr>
<td>0.13</td>
<td>6</td>
<td>7.5</td>
<td>9.82</td>
<td>Bad</td>
<td>-2.32</td>
</tr>
<tr>
<td>0.355</td>
<td>9</td>
<td>10.5</td>
<td>12.22</td>
<td>Good</td>
<td>-1.72</td>
</tr>
<tr>
<td>0.3285</td>
<td>11</td>
<td>12.5</td>
<td>11.64</td>
<td>Good</td>
<td>0.86</td>
</tr>
<tr>
<td>0.1125</td>
<td>11</td>
<td>12.5</td>
<td>9.67</td>
<td>Good</td>
<td>2.83</td>
</tr>
<tr>
<td>0.085</td>
<td>8</td>
<td>9.5</td>
<td>9.62</td>
<td>Good</td>
<td>-0.12</td>
</tr>
<tr>
<td>0.285</td>
<td>8</td>
<td>9.5</td>
<td>11.58</td>
<td>Good</td>
<td>-2.08</td>
</tr>
<tr>
<td>0.3</td>
<td>10</td>
<td>11.5</td>
<td>11.76</td>
<td>Good</td>
<td>-0.26</td>
</tr>
<tr>
<td>0.325</td>
<td>10</td>
<td>11.5</td>
<td>12.00</td>
<td>Good</td>
<td>-0.50</td>
</tr>
</tbody>
</table>
In addition, most of the Summary Report information can be found inside the Detailed Report as a comment added to the title cell; that version of the Summary Report is referred to as the **Quick Summary**.

<table>
<thead>
<tr>
<th>NeuralTools Quick Summary (Testing)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Net Information</strong></td>
</tr>
<tr>
<td>Name: Net Trained on Training Data</td>
</tr>
<tr>
<td>Configuration: GRNN Numeric Predictor</td>
</tr>
<tr>
<td>Location: This Workbook</td>
</tr>
<tr>
<td>Independent Category Variables: 1 (Sex)</td>
</tr>
<tr>
<td>Independent Numeric Variables: 7 (Length, Diameter, Height, Whole Weight, Shucked Weight, Visc...</td>
</tr>
<tr>
<td>Dependent Variable: Numeric Var. (Age)</td>
</tr>
<tr>
<td><strong>Testing</strong></td>
</tr>
<tr>
<td>Number of Cases: 500</td>
</tr>
<tr>
<td>% Bad Predictions (30% Tolerance): 10.0000%</td>
</tr>
<tr>
<td>Root Mean Square Error: 2.247</td>
</tr>
<tr>
<td>Mean Absolute Error: 1.560</td>
</tr>
<tr>
<td>Std. Deviation of Abs. Error: 1.617</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Name: Testing Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Rows: 500</td>
<td></td>
</tr>
<tr>
<td>Manual Case Tags: NO</td>
<td></td>
</tr>
<tr>
<td>Variable Matching: Automatic</td>
<td></td>
</tr>
<tr>
<td>Indep. Category Variables Used: Names from training</td>
<td></td>
</tr>
<tr>
<td>Indep. Numeric Variables Used: Names from training</td>
<td></td>
</tr>
<tr>
<td>Dependent Variable: Numeric Var. (Age)</td>
<td></td>
</tr>
</tbody>
</table>

You can control which reports are generated from the NeuralTools Application Settings dialog, accessed through the Utilities menu.
Whenever NeuralTools creates one or more charts, it places them with the reports. Charts are created in Excel format and can be customized using standard Excel chart commands.

**NeuralTools Utilities**

Three utilities are provided to help you to manage neural net modeling in NeuralTools. These are all accessed from the Utilities menu on the NeuralTools ribbon. A Neural Net Manager allows you to copy or move trained neural nets between workbooks and files. A Missing Data utility helps identify and correct cases with missing data in your data sets. A Testing Sensitivity utility helps to determine if the testing results are stable under different random selections of testing cases.

**Using NeuralTools with StatTools and Evolver**

NeuralTools is designed to be used with StatTools, the statistics add-in for Excel from Palisade. Both products share the same Data Set Manager; data sets defined in NeuralTools can be analyzed in StatTools and vice versa. Using StatTools, you can calculate statistics on variables in data sets defined in NeuralTools, along with statistics on predictions generated by NeuralTools.

Detailed Reports generated in NeuralTools are immediately available for analysis in StatTools; they automatically show on the list of data sets in StatTools Data Set Manager. This facilitates the use of StatTools to obtain statistical results beyond those contained in NeuralTools’ Summary Reports. For example, a testing Summary Report includes a histogram of residuals (defined as differences between actual and predicted values). Based on the histogram, the residuals might appear to be approximately normally distributed. To test the hypothesis of normality, one of StatTools’ normality tests can be applied to the Residuals variable in the Detailed Report.

NeuralTools’ Live Prediction feature makes it easy to see how changes to independent values affect the prediction. With Live Prediction, other tools available in Excel can be used to explore the relationship between independent variables and the dependent variable.

Palisades Evolver optimizer can be used with NeuralTools' Live Prediction capability to calculate optimal values for predictions made in NeuralTools. The file Auto Loans 4 - Using Evolver.xlsx file provides an example of allocating funds to several loan applicants to minimize the probability of a default occurring.
Chapter 4: NeuralTools
Command Reference

NeuralTools Icons

The following icons are shown on the NeuralTools ribbon in Excel.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Function Performed</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="data-set-manager.png" alt="Data Set Manager" /></td>
<td>Define a data set, or edit or delete an existing data set</td>
</tr>
<tr>
<td><img src="data-viewer.png" alt="Data Viewer" /></td>
<td>View distributions of variables and correlations between variables</td>
</tr>
<tr>
<td><img src="train.png" alt="Train" /></td>
<td>Train a neural net</td>
</tr>
<tr>
<td><img src="test.png" alt="Test" /></td>
<td>Test a neural net</td>
</tr>
<tr>
<td><img src="predict.png" alt="Predict" /></td>
<td>Predict values using a trained neural net</td>
</tr>
<tr>
<td><img src="utilities.png" alt="Utilities" /></td>
<td>Run neural net utilities</td>
</tr>
<tr>
<td><img src="help.png" alt="Help" /></td>
<td>Display NeuralTools help files</td>
</tr>
</tbody>
</table>
Icons in NeuralTools Dialogs

NeuralTools dialogs often have two icons at the bottom left, such as the ones shown here. The Help icon allows you to quickly access help on the relevant dialog. The Application Settings (double check) icon displays the Application Settings dialog where you can enter or edit settings for NeuralTools reports, as well as default settings for Training, Prediction, and Runtime.
NeuralTools Commands

The NeuralTools commands discussed in depth in this section fall into three groups, as indicated on the NeuralTools ribbon.

Data Group

The two commands in this group let you manage data sets and then view the data with standard graphs and statistics.

Neural Nets Group

Once you define one or more data sets and possibly use the Data Viewer to learn more about your data, you turn to the Neural Nets group to create and use neural nets. In some cases you can perform “one stop shopping” with the Train command, and in other cases you will need all three of the commands in this group.

Help Group

The Utilities and Help menus in this group provide several useful utilities for managing neural nets, plus various help files for becoming more familiar with NeuralTools.
Data Set Manager Command

Defines a new NeuralTools data set, or edits or deletes an existing data set

The Data Set Manager command allows you to define your data sets with cases and variables. Once data sets are defined, they can be used for neural net training, testing, and prediction. The Data Set Manager dialog allows you to add or remove data sets, name a data set, and specify the roles of the variables for analysis.

NeuralTools is structured around variables and cases. You work with a data set, or a set of statistical variables, located in contiguous columns in an Excel worksheet with variable names in the first row of the data set. Each row in the data set is a case. Each case has a set of independent variable values and either a known or missing value for the dependent output variable.

Each variable in a data set has an associated range of Excel cells, along with a corresponding range name. When you are defining a data set, NeuralTools attempts to identify the variables in a block of cells surrounding the current selection in Excel. This makes it quick and easy to set up a data set with variable names in the top row and variables laid out by column.
The options in the Data Set Manager dialog include:

- **New, Delete.** Adds a new data set, or deletes an existing one.
- **Name.** Specifies the name of the data set. You can accept the default generic name or enter a more meaningful name.
- **Excel Range.** Specifies the Excel range associated with a data set.
- **Apply Cell Formatting.** Adds a grid and colors that identify your data set.
Variable Options

The Variables table in the lower section of the Data Set Manager dialog provides information about each variable in a data set, including the variable’s range, its name, and its type.

Neural Tools guesses at the variable types, but you can (and often should) modify them. The Variable Type options include:

- **Dependent Category.** A dependent or output variable whose possible values are taken from a set of possible categories, such as Yes/No or Red/Green/Blue.

- **Dependent Numeric.** A dependent or output variable whose possible values are numeric.

- **Independent Category.** An independent variable whose possible values are taken from a set of possible categories, such as Yes/No or Red/Green/Blue.

- **Independent Numeric.** An independent variable whose possible values are numeric.

- **Tag.** A variable that takes the possible values "train", "test", or "predict". This type of variable is used to identify cases in a data set that will be used for training, testing, and prediction. It lets you specify the role of each case in the data set.

- **Unused.** A variable that will not be used in a neural net analysis.
The **Import** button allows variable types to be copied to this data set from another data set or trained neural net. The Import Variable Types dialog allows you to select the location and net to use for variable definitions.

![NeuralTools - Import Variable Types](image)

**More on Tag Variables**

A **tag variable** is a special type of variable in a NeuralTools data set that are used to identify cases in a data set that will be used for training, testing, and prediction. They are especially useful when you want to include all data (to be used for network training, testing, and prediction) in a single dataset. When you have a tag variable, NeuralTools selects the cases to use for training, testing, or prediction based on the tag variable values. By changing tag variable values, you can retrain a network using different cases and see how network performance changes. You can also add new cases with unknown dependent variable values to a dataset and assign them to be predicted using the "Predict" tag. A Tag variable can have only three possible values:

- **Train** specifies that the case will be used for training.
- **Test** specifies that the case will be used for testing.
- **Predict** specifies that the case will be used for prediction.

**Note:** If you have a tag variable in your data set, options in the Training dialog change. See the Train command for more information.
In a single session, NeuralTools allows:

- Up to 256 data sets, located in a single workbook.

- Up to 16384 variables per data set (in Excel 2007 and later). All the data for a single data set must be located in the same workbook.

- Number of data points per variable and cases per data set limited only by available memory (in Excel 2007 and later).

Actual data capacities may be lower, depending on the system configuration and version of Excel in use. Memory limitations of Excel itself may also affect data capacities.

**Note:** The Data Set Manager dialog lists all data sets and variables in the active workbook (this is the workbook listed in the caption of the Data Set Manager dialog). To list data sets in other workbooks, activate the desired workbook in Excel and display the Data Set Manager dialog.
Data Viewer Command

Allows you to view statistics and graphs for any data set

The NeuralTools Data Viewer is a tool that allows you to easily review statistics and graphs for any data set defined in NeuralTools. This can be useful when you are deciding which variables to include when training a neural net. For example, variables that show some degree of correlation with the dependent variable should probably be included. On the other hand, if a set of independent variables are very highly linearly correlated with each other (Pearson correlation close to 1), it might be worth trying to train a neural net with just one of those variables included. While neural nets can handle linearly correlated independent variables, the training process is more likely to be successful with fewer variables.

Clicking the Data Viewer icon on the NeuralTools ribbon brings up the Data Viewer Options dialog, where you choose a data set (previously defined in NeuralTools Data Set Manager) and select variables from that data set.
The following options are available:

- **Ignore Entire Row/Column if Any Cell Has a Missing or Non-Numeric Value.** Setting this option will discard an entire row or column if any of the values in that row or column is missing or non-numeric.

- **Correlation Calculations.** The data viewer displays correlation coefficients between your variables. This option controls how these coefficients are calculated.

### Multivariate Data Viewer Window

After you specify the Data Viewer options and click OK (and if your data contains more than one variable), you see the Multivariate Data Viewer window. This window has several different “views,” each giving a different perspective of your data. You can change the view by clicking the view buttons at the bottom of the window.

### Summary View

The summary view provides a quick overview of your data set in tabular format. Each row corresponds to one variable, and shows the variable’s name, a thumbnail graph, and summary statistics.
Options in the **Summary View** include:

- You can control which columns are displayed in the table by clicking the **Configure Columns** button (fourth from left) at the bottom of the window.

- You can click and drag an individual thumbnail off the window to get a full-sized graph.

- You can export this table to Excel by clicking the **Export** button (third from left) at the bottom at the bottom of the window.

**Correlation View**

The correlation view provides visual and numeric information about the correlation patterns between variables using a two-dimensional array of graphs. Histograms of individual variables appear on the diagonals, scatter plots of pairs of variables appear on either side of the diagonal. (Note: the graphs below the diagonals are always reflections of the graphs above the diagonal.) The correlation coefficients associated with each pair are shown at the top of each scatter plot.
Options in the Correlation View include:

- You can click and drag an individual graph off the window to get a full-sized graph.
- You can export the correlation coefficients to Excel by clicking the Export button at the bottom on the window.

**Overlay View**

The overlay view shows all the variables simultaneously in a single graph.

Options in the Overlay View include:

- You can control which inputs are displayed (up to a maximum of 10) in the overlay graph by clicking the Graph Options button at the bottom of the window, and choosing the Select Variables to Graph menu item. By default, NeuralTools will show only variables with the same formatting as the first variable in the data set. For example, if you have data set with some numeric values, some date values, and some percentage values, NeuralTools will, by default, show only the numeric values together on the graph.
- You can export this graph to Excel by clicking the Export button at the bottom of the window.
- You can change the formatting of the graph by clicking the Graph Options button at the bottom of the window.
The Trend and Boxplot views show summary statistics of all the variables in a compact, graphical format.
Options in the Trend and Boxplot Views include:

- You can control which inputs are displayed in the graphs (up to a maximum of 500) by clicking the Graph Options button at the bottom of the window and choosing the Select Variables to Graph menu item. By default, NeuralTools will show only variables with the same formatting as the first variable in the data set. For example, if you have data set with some numeric values, some date values, and some percentage values, NeuralTools will, by default, show only the numeric values together on the graph.

- You can export the graphs to Excel by clicking the Export button at the bottom of the window.

- You can change the formatting of the graphs by clicking the Graph Options button at the bottom of the window.
Train Command

Specifies settings for training a neural net and runs the training

The Train command allows you to specify settings for training a neural net in NeuralTools and to start training a neural net.

The Train tab in the Training dialog specifies general options for training a neural net. It includes the following:

- **Data Set.** Shows the data set to be used when training the neural net. This data set needs to be defined using the Data Set Manager, and it must be in the active sheet.

- **Save Net As.** Specifies the name and location for the neural net to be trained. Neural nets can be saved to an Excel workbook or to a file on disk. Click **Browse...** to change the name or location shown.
You can also enter a name and description for the neural net to be saved in this Save Net dialog.

The **When Training is Completed** options in the Training dialog allow you to automatically test and predict using the trained net following training. This can be done when the data to test and predict is located in the same data set with the training data.

- **Automatically Test on Randomly Selected Cases**  If this option is *not* checked, all of the cases in the selected data set will be used for training. If it is checked, then the following two options indicate how cases will be selected for testing.
  - **% Selected Cases.** Specifies the percentage of randomly selected cases that will be “held out” for testing, with the rest used for training.
  - **Select Same Cases as Long as This Number Is the Same.** If checked, this allows you to fix the random seed for choosing test cases. Note that Best Net Search uses a fixed selection of testing cases to compare multiple nets, regardless of whether this option is checked.

- **Automatically Predict Missing Dependent Values.** If this option is *not* checked, no automatic predictions for cases with missing dependent variables will be performed. If it is checked, then the following two options for prediction are relevant.
- **Enable Live Prediction.** Specifies that NeuralTools will place formulas in the cells where the predicted dependent variable values are shown to calculate the predicted values. For more on Live Prediction, see the Predict command in this chapter.

- **Calculate Variable Impacts.** Specifies that NeuralTools will calculate the relative impact of each independent variable in the training data set in determining the predictions calculated by the net. See below for more on this option.

**Train Tab with Tag Variable**

If the data set includes a “tag” variable with values “train,” “test,” and “predict,” the Train tab options are slightly different. Essentially, the tag variable values indicate the cases for testing and prediction.

**What is Variable Impact Analysis?**

The purpose of Variable Impact analysis is to measure the sensitivity of net predictions to changes in independent variables. This analysis is performed only on training data. As a result of the analysis, every independent variable is assigned a "Relative Variable Impact" value; these are percent values and add to 100%. The lower the percent value for a given variable, the less that variable affects the predictions. The results of the analysis can help in the selection of a new set of independent variables, one that will allow more accurate predictions.
For example, a variable with a low impact value can be eliminated in favor of some new variable.

However, you need to keep in mind that the results of the Impact Analysis are relative to a given neural net. The fact that one net “learned” to disregard a given variable makes it likely that another net will also “learn” to disregard it. But it is still possible that another training session with a different type of net might “discover” how the variable can make a significant contribution to accurate predictions. In data sets with smaller numbers of cases and/or larger numbers of variables, the differences in the relative impact of the variables between trained nets can be more pronounced. Also, it is important to remember that these values are “relative.” For example, suppose that with two independent variables one is assigned 99%, and the other 1%. This means that the latter is much less important than the former, but it does not mean that it is unimportant, particularly if high accuracy of predictions is desired.

Some additional points to note about Variable Impact Analysis include:

- Only the training data set is included in the analysis. If Auto-Testing or Auto-Prediction are used, those cases are not included. The reason is that they might have numeric values outside the training range, which could make analysis results less meaningful.

- For a given category independent variable, for every case, the analysis steps through all the valid categories for that variable, and measures the change to the predicted value. (With category prediction there is no numeric predicted value, but there are raw numeric net outputs on which the category prediction is based; those numeric outputs are used by the analysis.)

- For a given numeric independent variable, for every case, the analysis steps through the range from the minimum to the maximum training value for that variable, measuring the change to the predicted value (or, in the case of category prediction, change to the raw numeric outputs).
The purpose of the Variable Impact Analysis is not meant to support firm conclusions, like stating with high confidence that a variable is irrelevant. Instead, it is meant to help in a search for the best set of independent variables. Specifically, the results of the analysis might indicate that a given variable appears irrelevant, so that it’s worth trying to train a net without this variable.

The results of a Variable Impact analysis are displayed in the bottom section of the Training Summary report:

<table>
<thead>
<tr>
<th>Summary</th>
<th>Net Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Net Trained on Training Data (2)</td>
</tr>
<tr>
<td>Configuration</td>
<td>GRNN Numeric Predictor</td>
</tr>
<tr>
<td>Location</td>
<td>This Workbook</td>
</tr>
<tr>
<td>Independent Category Variables</td>
<td>1 (Sex)</td>
</tr>
<tr>
<td>Independent Numeric Variables</td>
<td>7 (Length, Diameter, Height, Whole Weight, Shucked Weight, Viscera Weight, Shell Weight)</td>
</tr>
<tr>
<td>Dependent Variable</td>
<td>Numeric Var. (Age)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Training</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Cases</td>
<td>1000</td>
</tr>
<tr>
<td>Training Time</td>
<td>0:00:06</td>
</tr>
<tr>
<td>Number of Trials</td>
<td>101</td>
</tr>
<tr>
<td>Reason Stopped</td>
<td>Auto-Stopped</td>
</tr>
<tr>
<td>% Bad Predictions (30% Tolerance)</td>
<td>5.0000%</td>
</tr>
<tr>
<td>Root Mean Square Error</td>
<td>1.858</td>
</tr>
<tr>
<td>Mean Absolute Error</td>
<td>1.290</td>
</tr>
<tr>
<td>Std. Deviation of Abs. Error</td>
<td>1.337</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Set</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Training Data</td>
</tr>
<tr>
<td>Number of Rows</td>
<td>1000</td>
</tr>
<tr>
<td>Manual Case Tags</td>
<td>NO</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable Impact Analysis</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Shell Weight</td>
<td>42.6663%</td>
</tr>
<tr>
<td>Whole Weight</td>
<td>28.8449%</td>
</tr>
<tr>
<td>Shucked Weight</td>
<td>20.4734%</td>
</tr>
<tr>
<td>Height</td>
<td>3.8846%</td>
</tr>
<tr>
<td>Sex</td>
<td>2.8059%</td>
</tr>
<tr>
<td>Diameter</td>
<td>0.5025%</td>
</tr>
<tr>
<td>Viscera Weight</td>
<td>0.4546%</td>
</tr>
<tr>
<td>Length</td>
<td>0.3679%</td>
</tr>
</tbody>
</table>
The Net Configuration tab in the Training dialog allows you to select the type of neural net that will be trained on your data. You can select a specific net configuration or select a Best Net Search, where NeuralTools will test a variety of possible configurations to identify the one that performs best.

NeuralTools supports different neural net configurations. For classification/category prediction, two types of networks are available: **Probabilistic Neural Nets (PNN)** and **Multi-Layer Feedforward Nets (MLF)**. Numeric prediction can be performed using MLF or **Generalized Regression Neural Nets (GRNN)**, which are closely related to PNN. For more information on the technical aspects of the available network configurations, see the More on Neural nets section of this manual.
The **Net Configuration tab** includes the following options:

- **Type of Net.** Selects the type of net to be used in training, or alternatively, selects a Best Net Search. The Net Configuration tab **Options** change depending on the type of net selected. Available net types are:
  
  - **PN/GRN Net.** These net types require no additional options to be selected for training. For this reason, this setting is the default when NeuralTools is installed. The PNN configuration is used for categorical dependent variables; the GRNN is used for numeric dependent variables.
  
  - **MLF Net.** This configuration has one or two hidden layers of nodes.

![Net Configuration Tab](image)

By selecting zero nodes for the second layer, the second layer will be eliminated. The most reliable way to find the best configuration of an MLFN net is to use the Best Net Search option instead of the option to train a single MLF Net. If there is not enough time for Best Net Search, it is recommended that the “Number of Nodes” values be left as **Automatic.**
- **Best Net Search.** In a Best Net Search, NeuralTools tests all checked net configurations, including PNN/GRNN and MLFN configurations with node counts in the entered minimum-maximum range shown below. The best performing configuration for your data is identified, based on the error obtained on the testing data. If **Store All Trial Nets in New Workbook** is selected, you will be able to individually load each tested net (regardless if it was the best performing network) from a workbook and use it for prediction after training is finished; a full testing Summary Report for each net will also be available. Just keep in mind that the Best Net Search performs a lot of training, so it can require significant computer time.
The **Runtime tab** in the Training dialog allows you to enter stopping conditions for training. If no stopping conditions are selected, training will stop eventually. The time required will be relatively short for PNN/GRNN nets, and much longer with MLF nets. One possible approach is to select no stopping conditions and click the Stop button in the training progress dialog when no more time is available for training. With Best Net Search, a time limit for training a single net must be defined, to ensure that the search algorithm does not devote too much time to one particular configuration. The three available stopping conditions can be combined, specifying that NeuralTools will stop when any of the conditions are reached.

![NeuralTools - Training dialog](image)

The Training Runtime options include:

- **Time.** Specifies a fixed time limit for training a single network. Training may stop before the specified period elapses, as soon as the algorithm determines that it is unlikely that further progress will be made. If a Best Net search is used, each tested net configuration will train for the entered time.

- **Trials.** Specifies that NeuralTools will execute no more than the specified number of trials before stopping. With MLFN, a “trial” is a single assignment of “weights” to connections between neurons; training consists of an intelligent search for weights that will generate best predictions. With PNN/GRNN, a trial is an assignment of “smoothing factors” to variables. Training consists of a search for the best smoothing factors.
• **Progress.** Specifies that NeuralTools will stop if it cannot improve an error statistic at least the entered % with the specified number of minutes.

Once you click Next in the Training dialog, you will see the Training Preview dialog. However, if you have already trained a net on this data set, you will first see the following prompt. This gives you the choice of deleting a previously trained net or creating a second trained net in addition to the previous one.

The Training Preview dialog then shows the setup of the current network training along with any errors detected in your data, prior to starting training. By examining the contents of this dialog, you can see all your selected training assumptions as reported by NeuralTools. The **Errors and Warnings** section gives a description of any problems NeuralTools has detected in your data or settings, and you can correct these if necessary prior to spending time training.
The Training Progress window reports on the status of network training as it runs. The graphs show how NeuralTools is improving the network and reducing the reported error.

The Training Progress Window reports the “prediction” error on the training data. Observing changes in this value should not lead to any direct conclusions about the quality of predictions the net will make for cases not used in training. Such conclusions should be based on the error obtained on the testing data. Also note that with numeric prediction, the error reported in the Progress Window is the Root Mean Square error based on scaled data (see information about scaling in the Inputs Transformation section of this manual). For category prediction, the reported error is based on numeric representation of category data.

Both summary and detailed reports can be created after training. These reports show the performance of the trained neural net. The actual contents of the generated reports are specified in the Reports to Generate and Columns in Detailed Reports sections of the Application Settings dialog (from the Utilities menu). Summary Reports are placed on their own worksheet, while Detailed Reports are placed in columns to the right of a data set.
The Training Summary report gives statistics and graphs (not shown here) on the performance of the trained neural net.

<table>
<thead>
<tr>
<th>Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Net Information</strong></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Net Trained on Training Data</td>
</tr>
<tr>
<td>Configuration</td>
<td>GRNN Numeric Predictor</td>
</tr>
<tr>
<td>Location</td>
<td>This Workbook</td>
</tr>
<tr>
<td>Independent Category Vari</td>
<td>1 (Sex)</td>
</tr>
<tr>
<td>Independent Numeric Vari</td>
<td>7 (Length, Diameter, Height, Whole Weight, Shucked Weight, Viscera Weight, Shell Weight)</td>
</tr>
<tr>
<td>Dependent Variable</td>
<td>Numeric Var. (Age)</td>
</tr>
<tr>
<td><strong>Training</strong></td>
<td></td>
</tr>
<tr>
<td>Number of Cases</td>
<td>1000</td>
</tr>
<tr>
<td>Training Time</td>
<td>0:00:06</td>
</tr>
<tr>
<td>Number of Trials</td>
<td>101</td>
</tr>
<tr>
<td>Reason Stopped</td>
<td>Auto-Stopped</td>
</tr>
<tr>
<td>% Bad Predictions</td>
<td>5.0000%</td>
</tr>
<tr>
<td>Root Mean Square Error</td>
<td>1.856</td>
</tr>
<tr>
<td>Mean Absolute Error</td>
<td>1.290</td>
</tr>
<tr>
<td>Std. Deviation of Abs. Error</td>
<td>1.337</td>
</tr>
<tr>
<td><strong>Data Set</strong></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Training Data</td>
</tr>
<tr>
<td>Number of Rows</td>
<td>1000</td>
</tr>
<tr>
<td>Manual Case Tags</td>
<td>NO</td>
</tr>
</tbody>
</table>

Histogram of Residuals (Training)
For Category Prediction/Classification, key statistics and graphs in the training summary report include:

- **% Bad Predictions.** The percentage of cases for which the predicted category does not agree with the actual category.

- **Mean Incorrect Probability (available with PNN only).** For every case, NeuralTools computes Probability of Incorrect Categories, which is the sum of probabilities assigned by PNN to incorrect categories. For example, if for a given case a net assigns 30% probability to red, 20% to yellow, and 50% to green, and the correct category is red, then the value for that case is 20% + 50% = 70%. This value provides a case-by-case error measure for category prediction, corresponding to the Residual Error for numeric prediction. Then **Mean Incorrect Probability** is the average error value for all the cases.

- Detailed Reports show Incorrect Probability on a case-by-case basis. To better understand the concept, it might be helpful to change Detailed Report settings to show the probabilities assigned by a PNN to every possible category for the dependent variable. To do this, click the dropdown menu for the **Columns in Detailed Reports** row in the Application Settings dialog to see the **NeuralTools - Columns to Display in Detailed Reports** dialog. In that dialog, select **Probabilities of All Categories (for PNN) for Testing.** Then train a PNN on a data set with at least three categories in the dependent variable (the Auto Loans 1a example file can be used) with **Automatically Test** selected. In the resulting Detailed Report, you can see how the values in the **Incorrect%** column relate to the probabilities assigned to each possible category — the Incorrect% is the sum of the probabilities for all incorrect categories.

- **Classification Matrix.** Compares actual to predicted categories on category-by-category basis. For example, the classification matrix might reveal that a net correctly detects a medical condition in patients that have the condition, but has some tendency to raise false alarms for healthy patients.

- **Variable Impacts.** If selected, displays the relative impact of independent variables on predictions.

- **Histogram of Probability of Incorrect Categories (available with PNN nets only).** See Mean Incorrect Probability above for an explanation of Probability of Incorrect Categories.
For Numeric Prediction, key statistics and graphs in the training summary report include:

- **% Bad Predictions.** A prediction counts as "bad" if it falls outside the defined margin around the actual value; the width of the margin is defined as Good/Bad Tolerance (Training) setting in the Application Settings dialog.

- **Root Mean Square Error.** A measure of deviation of predictions from actual value (calculated as square root of average squared deviations).

- **Mean Absolute Error.** Average deviation of predictions from actual values.

- **Variable Impacts.** If selected, displays the relative impact of independent variables on predictions.

- **Histogram of Residuals.** “Residual” is the difference between the actual and predicted values.

- **Scatter plots.** Displays relationships between actual values, predicted values, and residuals.

You can request a Detailed Report for the training data from the Application Settings dialog, but the default is to not create such a report. The reason is that you are usually not interested in seeing how well the trained net predicts individual training cases; you are more interested in seeing how well it performs on testing cases. However, if you do decide to request a Detailed report, new columns will be placed next to the training data set.
Test Command

Specifies settings for testing a trained neural net and runs the testing

The Test command allows you to specify settings to be used for testing a trained neural net and then start the testing.

Testing data usually is data with known output values that were not used in training the net. The options in the Testing dialog include:

- **Data Set.** Specifies the data set to be used when testing the trained neural net. This data set needs to be defined using the Data Set Manager, and it must be in the active worksheet.

- **Net to Use.** Specifies the name and location for the neural net to be tested. Neural nets can be saved to an Excel workbook or to a file on disk. Click Browse... to change the name or location shown.
**Variable Matching** specifies how variables in the data set to be tested will be matched with variables in the data set that was used to train the net.

Two options are possible for variable matching:

- **Automatic Matching.** Variable Names in the testing data set are matched by name with those in the trained net’s data set, and variable types are set based on this matching.

- **Custom Matching.** Custom matching allows you to individually assign the matching of variables in the testing data set with those in the trained net’s data set. This is done when variable names are different in the two data sets or different assignments are desired.

The Variable Matching dialog lists the names of variables in each data set so they can be matched. Only variables with the same type can be matched. Each time you do a matching, the assignments made are stored with the data set. If you matched previously, you can click **Load Prior Matching** to cycle through previous matchings to access a set of previous matchings for the data set.
The Testing Preview dialog shows the setup of the current network testing along with any errors detected in your data, prior to starting testing. By examining the contents of this dialog, you can see all your selected testing assumptions as reported by NeuralTools. The **Errors and Warnings** section gives a description of any problems NeuralTools has detected in your data, and you can correct these if necessary prior to testing.
Testing Reports

Both summary and detailed reports can be created after testing. These reports show the performance of the trained neural net on the testing data. The actual contents of the generated reports are specified in the **Reports to Generate** and **Columns in Detailed Reports** sections of the Application Settings dialog (from the Utilities menu). Summary Reports are placed on their own worksheet, while Detailed Reports are placed in columns to the right of a data set.

The Testing Summary report provides statistics and graphs (not shown here) on the performance of the trained neural net on the test data set.
For Category Prediction/Classification, key statistics and graphs in the testing summary report include:

- **% Bad Predictions.** The percentage of cases for which the predicted category does not agree with the actual category.

- **Mean Incorrect Probability (available with PNN only).** For every case, NeuralTools computes Probability of Incorrect Categories, which is the sum of probabilities assigned by PNN to incorrect categories. For example, if for a given case a net assigns 30% probability to red, 20% to yellow, and 50% to green, and the correct category is red, then the value for that case is 20% + 50% = 70%. This value provides a case-by-case error measure for category prediction, corresponding to the Residual Error for numeric prediction. Then **Mean Incorrect Probability** is the average error value for all the testing cases.

- Detailed Reports show Incorrect Probability on a case-by-case basis. To better understand the concept, it might be helpful to change Detailed Report settings to show the probabilities assigned by a PNN to every possible category for the dependent variable. To do this, click the dropdown menu for the **Columns in Detailed Reports** row in the Application Settings dialog to see the **NeuralTools - Columns to Display in Detailed Reports** dialog. In that dialog, select **Probabilities of All Categories (for PNN) for Testing.** Then train a PNN on a data set with at least three categories in the dependent variable (the Auto Loans 1a example file can be used) with **Automatically Test** selected. In the resulting Detailed Report, you can see how the values in the Incorrect% column relate to the probabilities assigned to each possible category — the Incorrect% is the sum of the probabilities for all incorrect categories.

- **Classification Matrix.** Compares actual to predicted categories on category-by-category basis. For example, the classification matrix might reveal that a net correctly detects a medical condition in patients that have the condition, but has some tendency to raise false alarms for healthy patients.

- **Histogram of Probability of Incorrect Categories (available with PNN nets only).** See "Mean Incorrect Probability" above for an explanation of "Probability of Incorrect Categories".
For Numeric Prediction, key statistics and graphs in the testing summary report include:

- **% Bad Predictions.** A prediction counts as “bad” if it falls outside the defined margin around the actual value; the width of the margin is defined as **Good/Bad Tolerance (Training)** setting in the Application Settings dialog.

- **Root Mean Square Error.** A measure of deviation of predictions from actual value (calculated as square root of average squared deviations).

- **Mean Absolute Error.** Average deviation of predictions from actual values.

- **Histogram of Residuals.** “Residual” is the difference between the actual and predicted values.

- **Scatter plots.** Displays relationships between actual values, predicted values, and residuals.

The Testing Detailed report is placed next to the testing data set and shows how well the trained net predicted individual output values in the test data set.

<table>
<thead>
<tr>
<th>Shell Weight</th>
<th>Rings</th>
<th>Age</th>
<th>Tag Used</th>
<th>Prediction</th>
<th>Good/Bad</th>
<th>Residual</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.07</td>
<td>8</td>
<td>9.5</td>
<td>test</td>
<td>8.48</td>
<td>Good</td>
<td>1.02</td>
</tr>
<tr>
<td>0.065</td>
<td>11</td>
<td>12.5</td>
<td>test</td>
<td>8.28</td>
<td>Bad</td>
<td>4.22</td>
</tr>
<tr>
<td>0.13</td>
<td>6</td>
<td>7.5</td>
<td>test</td>
<td>9.82</td>
<td>Bad</td>
<td>-2.32</td>
</tr>
<tr>
<td>0.325</td>
<td>9</td>
<td>10.5</td>
<td>test</td>
<td>12.22</td>
<td>Good</td>
<td>-1.72</td>
</tr>
<tr>
<td>0.325</td>
<td>11</td>
<td>12.5</td>
<td>test</td>
<td>11.64</td>
<td>Good</td>
<td>0.86</td>
</tr>
<tr>
<td>0.1125</td>
<td>11</td>
<td>12.5</td>
<td>test</td>
<td>9.67</td>
<td>Good</td>
<td>2.83</td>
</tr>
<tr>
<td>0.085</td>
<td>8</td>
<td>9.5</td>
<td>test</td>
<td>9.62</td>
<td>Good</td>
<td>-0.12</td>
</tr>
<tr>
<td>0.285</td>
<td>8</td>
<td>9.5</td>
<td>test</td>
<td>11.58</td>
<td>Good</td>
<td>-2.08</td>
</tr>
<tr>
<td>0.3</td>
<td>10</td>
<td>11.5</td>
<td>test</td>
<td>11.76</td>
<td>Good</td>
<td>-0.26</td>
</tr>
<tr>
<td>0.325</td>
<td>10</td>
<td>11.5</td>
<td>test</td>
<td>12.00</td>
<td>Good</td>
<td>-0.50</td>
</tr>
</tbody>
</table>

In the Testing Detailed Report, predictions are marked as **Good** or **Bad** based on the tolerance level set in the Application Settings dialog. If you run multiple tests, multiple Detailed Reports can be added in new columns to the right of the test data set, so that you can see how predictions change for individual cases as new trained nets are tested.
A pop-up comment in Excel provides quick access to Summary Report information while examining a Detailed Report. Just hover the mouse over the cell to see this comment. (Of course, Excel comments must be enabled in the Excel Options dialog.)

<table>
<thead>
<tr>
<th>NeuralTools Quick Summary (Testing)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Net Information</strong></td>
</tr>
<tr>
<td>Name: Not Trained on Training Data</td>
</tr>
<tr>
<td>Configuration: GRNN Numeric Predictor</td>
</tr>
<tr>
<td>Location: This Workbook</td>
</tr>
<tr>
<td>Independent Category Variables: 1 (Sex)</td>
</tr>
<tr>
<td>Independent Numeric Variables: 7 (Length, Diameter, Height, Whole Weight, Shucked Weight, Visc...)</td>
</tr>
<tr>
<td>Dependent Variable: Numeric Var. (Age)</td>
</tr>
<tr>
<td><strong>Testing</strong></td>
</tr>
<tr>
<td>Number of Cases: 500</td>
</tr>
<tr>
<td>% Bad Predictions (30% Tolerance): 10.0000%</td>
</tr>
<tr>
<td>Root Mean Square Error: 2.247</td>
</tr>
<tr>
<td>Mean Absolute Error: 1.560</td>
</tr>
<tr>
<td>Std. Deviation of Abs. Error: 1.617</td>
</tr>
<tr>
<td><strong>Data Set</strong></td>
</tr>
<tr>
<td>Name: Testing Data</td>
</tr>
<tr>
<td>Number of Rows: 500</td>
</tr>
<tr>
<td>Manual Case Tags: NO</td>
</tr>
<tr>
<td>Variable Matching: Automatic</td>
</tr>
<tr>
<td>Indep. Category Variables Used: Names from training</td>
</tr>
<tr>
<td>Indep. Numeric Variables Used: Names from training</td>
</tr>
<tr>
<td>Dependent Variable: Numeric Var. (Age)</td>
</tr>
</tbody>
</table>
Predict Command

Specifies settings for predicting values using a trained neural net and runs the prediction

The Predict command allows you to specify settings to be used for predicting values using a trained neural net and then calculating the predictions.

Data to predict are typically cases with unknown dependent variable values. Options in the Prediction dialog include:

- **Data Set.** Specifies the data set to be used for prediction. This data set needs to be defined using the Data Set Manager, and it must be in the active worksheet.

- **Net to Use.** Specifies the name and location for the neural net to be used for prediction. Neural nets can be saved to an Excel workbook or to a file on disk. Click **Browse...** to change the name or location shown.

- **Variable Matching.** Specifies how variables in the data set with the prediction data will be matched with variables in the data set that was used to train the net. Click **Edit...** to change variable matching. For more information on **Variable Matching**, see the Test command in this chapter.
• **Predict For.** Selects the cases for which predictions will be made. Typically you will select to predict cases with Missing Dependent Variable Values, but you can make prediction for All Cases (even those where the dependent variable value is known) if desired. If you have a Tag variable in the data set, dependent variable values will be predicted only for cases marked with the tag “predict”.

• **Options.** Sets predicted value location and Live Prediction options.
  - **Place Predicted Values Directly in Data Set.** This option specifies that predicted values will be placed directly in the dependent variable location in the data set for each predicted case, possibly in addition to being placed in the Detailed Report (depending on whether Detailed Reports are selected in the Reports to Generate setting in Application Settings). This overwrites any current contents of the cell, so it should be used with caution. You will be able to identify predicted values by color in the data set.
  - **Enable Live Prediction.** Specifies that NeuralTools will place formulas in the cells where the predicted dependent variable values are shown. These formulas allow NeuralTools to calculate the predicted values as independent values change.
  - **Exclude Live Prediction for Cases with Missing or Invalid Values.** Specifies that a live prediction formula will not be added where input variable values for a case are missing. Missing input values cause live prediction formulas to return an error value. However, it can be useful to allow NeuralTools to enter formulas in cases where independent values are missing, because as soon as missing values are filled, the prediction will automatically show.
**Live Prediction** is a powerful capability of NeuralTools (Industrial version only) that allows you to perform predictions automatically in Excel without going through a specific Predict operation. With Live Prediction, NeuralTools places formulas in the cells where the predicted dependent variable values are shown. These formulas use a custom NeuralTools function to calculate the predicted values, such as:

```
=NetOutputPrediction(_PALDS_DG25B8C82B!$A$140,
"DG25B8C82B", "VG1DD83AF2", 'Prediction Data'!$A$6:$I$6, A7:I7)
```

The actual formula is added to your worksheet by NeuralTools and does not need to be entered by you. The arguments let NeuralTools identify the trained network in use, along with the location of the independent values in the worksheet. When the input independent variable values for a case are added or changed, NeuralTools will automatically return a new predicted value. This makes it simple to add and generate predictions for new cases using an existing trained net.

Note: if the prediction will be based on cell values that are not expected to change, then de-selecting Live Prediction in training or prediction dialog is recommended. This decreases the time it takes for Excel to recalculate the workbook.
The Prediction Preview dialog shows the prediction setup for the selected data set along with any errors detected in your data or settings, prior to starting prediction. By examining the contents of this dialog, you can see all your selected prediction assumptions as reported by NeuralTools. The **Errors and Warnings** section gives a description of any problems NeuralTools has detected in your data, and you can correct these if necessary prior to prediction.
Both summary and detailed reports can be created after prediction. You can control whether these reports are generated, and what they will contain, in the **Reports to Generate** and **Columns in Detailed Reports** sections of the Application Settings dialog.

By default, the Summary Report for predictions is not created, although you can request it from the Application Settings dialog. The more useful report for predictions is the Detailed Report. This report is placed in columns to the right of the prediction data set. This provides a location for predictions when you do not want to place them inside the dependent variable in the data set itself. If the dependent variable contains historical data for some cases, it might be safer not to mix historical cases with predictions.

If you are running multiple predictions, several Detailed Reports can be added in new columns to the right of the data set, so that you can see how predictions change for individual cases as new trained nets are used.
Application Settings Command

Specifies default settings for training, testing and prediction

The Application Settings dialog (from the Utilities menu) allows you to specify a variety of settings for all of your NeuralTools analyses. These include: (1) which reports to generate for training, testing, and prediction, (2) which training defaults to use, (3) which runtime defaults to use, and others. Many Application Settings are defaults to be used in the Training, Testing, and Prediction dialogs. These dialog are described in detail in previous sections of this chapter. Other settings are covered here.
Reports settings include:

- **Reports to Generate.** Each operation in NeuralTools can produce both Summary and Detailed reports. However, you will typically want to use the default report settings, as certain reports add little value to certain operations. For example, the detailed report is the standard report from prediction, and a summary report in this case adds little value.

  ![NeuralTools - Reports to Generate](image.png)

Summary Reports are placed on their own worksheet, while Detailed Reports are placed in columns to the right of a data set, in the same worksheet as the data set.

- **Place Summary Reports In** options include:
  - **Active Workbook.** A new worksheet in the same workbook as the data is created for each report.
  - **New Workbook.** Reports are placed in a new workbook.

- **Detailed Report Location** options include:
  - **Overwrite Existing Reports.** Columns with data from prior detailed reports are overwritten with new detailed reports. (To delete a detailed report manually, simply delete their columns from the worksheet.)
  - **Right of the Data Set.** New columns are inserted to the right of the data set to hold the new detailed reports.
  - **Right of Existing Reports.** Columns to the right of the data set and existing reports are used to hold the new detailed reports.
• **Columns in Detailed Reports.** For every selected row, a column will be displayed in the detailed report to the right of the data set that will show the information for every case.

![Image of NeuralTools - Columns to Display in Detailed Reports]

The following columns can be displayed:

- **Tag Used.** "train", "test" and "predict" tags show for every case if it was used as part of the training or testing set, or if a prediction was made for a given case.

- **Prediction Obtained Using Net.** Number or category predicted by net.

- **Probability of Predicted Category (for PNN).** PNN not only predicts an unknown category, but also assigns a probability of that category. Not available when categories are predicted using MLFN. Does not apply to numeric prediction.

- **Probability of Incorrect Categories (for PNN).** Sum of probabilities assigned by a PNN net to incorrect categories. For example, if for a given case a net assigns 30% probability to red, 20% to yellow, and 50% to green, and the correct category is red, then the value for that case is 20% + 50% = 70%. This column provides a case-by-case error measure for category prediction, corresponding to the "Residual Error" column for numeric prediction.

- **Residual Error.** The difference between the actual and predicted dependent values. Does not apply to category prediction.

- **Good/Bad Evaluation.** For numeric prediction, the column indicates whether the prediction for a given case falls within the margin around the actual value defined in the Tolerance for Good/Bad Evaluation setting. For
category prediction, the column indicates whether the predicted category agrees with the actual category.

- **Probabilities of All Categories (for PNN).** If this option is selected and a PNN is trained, one column will be inserted for every dependent category. For example, if the net is used to predict a color, there could be Red%, Yellow%, and Green% columns, containing probabilities assigned to each color.

- **Tolerance for Good/ Bad Evaluation.** For testing and training, if a numeric prediction is within the entered % of the actual dependent variable value, it will be labeled Good.
Neural Net Manager Command

Allows the copying, moving, and deletion of trained neural nets

The Neural Net Manager (from the Utilities menu) allows you to manage trained neural nets, where you can move them between workbooks and files, as well as add descriptive information about them.

Neural nets can be stored in an Excel workbook or in a file on disk. Any number of networks can be placed in a single Excel workbook. By using the Neural Net Manager, you can move networks to new workbooks or files, or delete or replace them. This allows you to easily analyze data sets in other workbooks using existing trained networks, without the workbook with the training data present.
Neural Net Manager options include:

- **Copy.** This copies a trained neural net to a different location. Simply select the workbook or file where you want to place the network.

  ![NeuralTools - Select New Net Location](image)

- **Remove.** Deletes a trained neural net.

- **Replace.** Overwrites a trained neural net with a new one. This feature is available with nets that are used for Live Prediction. After replacement, live predictions that were previously made using the old net will be made using the new one. However, this does not apply to detailed reports. If a detailed report contains live prediction cells where the net to be replaced is used, then after replacement those cells will contain fixed values.

- **Net Information.** Allows descriptive information to be added to a network. This helps identify the trained network and the conditions it was trained under.
Chapter 4: NeuralTools Command Reference

Missing Data Utilities Command

Allows the replacement of missing data and error values in a data set with artificial values

The Missing Data Utilities command (from the Utilities menu) allows you to replace missing or other unwanted data in your data set with artificial values. Cases with missing variable values are disregarded by NeuralTools during training, testing and prediction, so it is often useful to correct these before processing.

The Training Preview dialog displays a warning when you have cases with missing values in a data set. If this happens, these cases can be fixed using the Missing Data Utilities command.

The Missing Data Utilities dialog has the following options:

- **Variables to Modify.** Provides a list of the variables used in the data set in the current worksheet and displays the number missing, error, and (for numeric variables) non-numeric data. Checking a variable selects it to have missing or other unwanted data replaced.

  The variable list provides a right-click menu with commands for selecting and deselecting groups of variables.
• **Values to Replace.** Selects the types of values in the selected variables that will be replaced. **Specific Value** allows you to replace all instances of a specific value for a variable with a new value.

• **What to Use For Replacement.** Specifies the values to be placed in the data set instead of the missing or other unwanted data. Different values are specified for Category and Numeric variables:

  The options for Category Variables are:

  - **Most or Least Frequent Category.** The category value that occurs most or least frequently in the data set.

  - **Neighboring Category.** The category value that occurs in the case next to the case with the missing value.

  - **Randomly Selected Category.** A category value randomly selected from those in the data set.

  - **Specific Category.** Sets all missing or unwanted values to a specific value of your choice.

  The options for Numeric Variables are:

  - **Variable Average Value.** The average value for the variable across all cases in the data set.

  - **Variable Median Value.** The median value for the variable across all cases in the data set.

  - **Interpolation from Neighboring Values.** The value calculated by interpolating between the variable values in the cases in the data set next to the case with the missing value.

  - **Random Val. (between Min. and Max.).** A random value selected between the variable minimum and maximum for all cases in the data set.

  - **Specific Value.** Sets all missing or unwanted values to a specific value of your choice.

  For both variable types, **Clear Cells** clears the selected values for the variable.
More on Missing Values

The Missing Data Utilities dialog provides one possible approach to missing data: It generates artificial data where actual data are missing. However, it is sometimes better to simply leave missing data as blank cells, and let NeuralTools disregard cases with missing data. Note that NeuralTools will not recognize special symbols like "?" as missing data; question marks need to be cleared, and this can be accomplished with the Missing Data Utilities, by selecting Specific Value in Values to Replace section, and Clear Cells in What to Use for Replacement section.

It may also be possible to use NeuralTools to predict (and thereby fill in) missing values in one independent variable from other independent variables that have little or no missing data. Testing results will indicate whether a net trained to predict missing values is reliable.
Testing Sensitivity Command

Determines if the testing results are stable under different random selections of testing cases

The Testing Sensitivity command (from the Utilities menu) can be used to check the possibility that good testing results were obtained by luck. The values of a testing measure will differ, at least to some extent, from one training session to another if the subset of cases used for testing is selected randomly. The smaller the data set is, the larger the variation is likely to be. Therefore, with a small data set, the testing results from a single training session might not be reliable. The Testing Sensitivity analysis helps to determine if the results are stable under different selections of testing cases. It also helps answer what percent of cases should be set aside for testing.
The Testing Sensitivity dialog includes the following options:

- **% Cases to Set Aside for Testing.** Multiple percent values can be entered. Each value will be included in the analysis, by setting aside the specified percent of cases for testing during multiple training sessions.

- **Number to Train for Each % Value.** The number of training sessions for each percent value listed above.

- **Net Configuration settings.** The multiple training sessions included in the analysis will all share the same net configuration. For details on specifying the net configuration, see the **Training Command** section of this chapter.

The bottom section of the dialog provides an estimate of how long it will take for the analysis to complete. This can be based on a completed training session, or on the Runtime settings specified in the Training dialog.

The analysis generates a report showing the ranges between the minimum and maximum values of a testing measure obtained in the multiple training sessions.
Chapter 5: More on Neural nets

Neural Net Basics

A neural net is a system that takes numeric inputs, performs computations on these inputs, and outputs one or more numeric values. When a neural net is designed and trained for a specific application, it outputs approximately correct values for given inputs. For example, a net could have inputs representing some easily measured characteristics of an abalone (a sea animal), such as length, diameter, and weight. The computations performed inside the net would result in a single number, which is generally close to the age of the animal. (The age of an abalone is difficult to determine.)

The inspiration for neural nets comes from the structure of the brain. A brain consists of a large number of cells, referred to as “neurons.” A neuron receives impulses from other neurons through a number of “dendrites.” Depending on the impulses received, a neuron may send a signal to other neurons, through its single “axon,” which connects to dendrites of other neurons. Like the brain, artificial neural nets consist of elements, each of which receives a number of inputs, and generates a single output, where the output is a relatively simple function of the inputs.

Neural Nets versus Statistical Methods

Neural nets provide an alternative to more traditional statistical methods. Like Linear Regression, they are used for function approximation. Like Discriminant Analysis and Logistic Regression, they are used for classification. The advantage of neural nets is that they are capable of modeling extremely complex relationships. This stands in contrast with the traditional linear techniques (Linear Regression and Linear Discriminant Analysis). Techniques for optimizing linear models were well known before artificial neural nets were invented in middle of the 20th century. Effective algorithms for training neural nets took many years to develop. However, there are now a range of sophisticated algorithms for neural net training, making them an attractive alternative to the more traditional methods.
The Structure of a Neural Net

The structure of a neural net consists of connected units referred to as nodes, also called neurons. Each node performs a portion of the computations inside the net: a node takes some numbers as inputs, performs a relatively simple computation on these inputs, and returns an output. The output value of a node is passed on as one of the inputs to another node, except for nodes that generate the final output values of the entire system.

Nodes are arranged in layers. The input layer nodes receive the inputs for the computations, like the length, diameter, and weight of an individual abalone. These values are passed to the nodes in the first hidden layer, which perform computations on their inputs and pass their outputs to the next layer. This next layer could be another hidden layer, if there is one. The outputs from the nodes in the last hidden layer are passed to the node or nodes that generate the final outputs of the net, like the age of the abalone.

Numeric and Category Prediction

When neural nets are used to predict numeric values, they typically have just one output. This is because single-output neural nets are more reliable than multiple-output nets, and almost any prediction problem can be addressed using single-output nets. For example, instead of constructing a single net to predict the volume and the price for a stock on the following day, it is better to build one net for price predictions, and another for volume predictions. On the other hand, neural nets have multiple outputs when they are used for classification/category prediction. For example, suppose you want to predict whether the price of a stock the following day will “rise more that 1%,” “fall more than 1%”, or “not change more than 1%.” In this case, the net will have three numeric outputs, and the output with the largest value will indicate the category selected by the net.

Training a Net

Training a net is the process of fine-tuning the parameters of the computation, where the purpose is to make the net output approximately correct values for given input values. This process is guided by training data on the one hand, and the training algorithm on the other. The training algorithm selects various sets of computation parameters, and evaluates each set by applying the net to each training case to determine how good the output values from the net are. Each set of parameters is a “trial”; the training algorithm selects new sets of parameters based on the results of previous trials.
Computer Processing of Neural Nets

A neural net is a model of computations that can be implemented in various types of computer hardware. A neural net could be built from small processing elements, with each performing the work of a single node. However, neural nets are typically implemented on a computer with a single powerful processor, like most computers currently in use. With single-processor computers the program, like NeuralTools, uses the same processor to perform each node’s computations; in this case the concept of a node describes part of the computations needed to obtain a prediction, as opposed to a physical processing element.

Types of Neural nets

There are various types of neural nets, differing in structure, kinds of computations performed inside nodes, and training algorithms. One type offered in NeuralTools is the Multi-Layer Feedforward Network. With MLF nets, you can specify whether there should be one or two layers of hidden nodes, and how many nodes the hidden layers should contain. (NeuralTools provides help with making appropriate selections, as described in the section on MLF nets.) NeuralTools also offers Generalized Regression Neural Nets and Probabilistic Neural Nets; these are closely related, with the former used for numeric prediction, and the latter for category prediction/classification. With GRN/PN nets there is no need for you to make decisions about the structure of a net. These nets always have two hidden layers of nodes, with one node per training case in the first hidden layer, and the size of the second layer determined by some facts about training data.

The remaining sections of this chapter discuss in more detail each type of neural net offered in NeuralTools.

Input Transformation

NeuralTools scales numeric variables before training, so that the values of each variable are approximately in the same range. This is done to equalize the effect variables have on net output during initial stages of training. When a variable is not significant for making correct predictions, this will be reflected during training by reducing the weights of connections leading from an input to first-hidden-layer nodes. However, if that insignificant variable happens to have a larger order of magnitude than other variables, the weights need to be reduced so much more to compensate for the greater values.
The scaling uses the mean and the standard deviation for each variable, computed on the training set. The mean is subtracted from each value, and the result is divided by the standard deviation. The same scaling parameters are used when testing the trained net or using it to make predictions.

Category/symbolic data cannot be used directly with a neural net, which takes numbers as inputs. Consequently, every independent category variable is represented by a number of numeric net inputs, one for every possible category. The “one-of-n” conversion method is used. For example, consider the following set of training cases:

<table>
<thead>
<tr>
<th>Age</th>
<th>State</th>
<th>Loan Amount</th>
<th>Dependent: Loan Payment</th>
</tr>
</thead>
<tbody>
<tr>
<td>41</td>
<td>NY</td>
<td>4000</td>
<td>timely</td>
</tr>
<tr>
<td>32</td>
<td>CT</td>
<td>7000</td>
<td>late</td>
</tr>
<tr>
<td>54</td>
<td>NJ</td>
<td>6000</td>
<td>timely</td>
</tr>
<tr>
<td>37</td>
<td>NY</td>
<td>5000</td>
<td>default</td>
</tr>
</tbody>
</table>

They are presented to the net as:

<table>
<thead>
<tr>
<th>Age</th>
<th>State=CT</th>
<th>State=NJ</th>
<th>State=NY</th>
<th>Loan Amount</th>
<th>Dependent: Loan Payment</th>
</tr>
</thead>
<tbody>
<tr>
<td>41</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4000</td>
<td>timely</td>
</tr>
<tr>
<td>32</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>7000</td>
<td>late</td>
</tr>
<tr>
<td>54</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>6000</td>
<td>timely</td>
</tr>
<tr>
<td>37</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>5000</td>
<td>default</td>
</tr>
</tbody>
</table>
Multi-Layer Feedforward Nets

Multi-Layer Feedforward Networks (also referred to as Multi-Layer Perceptron Networks) are systems capable of approximating complex relationships between independent variables and a dependent variable.

**MLF Architecture**

The following diagram shows an MLF net for numeric prediction with three independent numeric variables. This net is configured to have two nodes in the first hidden layer and three nodes in the second hidden layer.

The behavior of the net is determined by:

- Its topology (the number of hidden layers and the numbers of nodes in those layers)
- The “weight” of each connection (a parameter assigned to each connection) and bias terms (a parameter assigned to each node)
- An activation/transfer function, used to convert the inputs of each node into its output
Specifically, a hidden neuron with $n$ inputs first computes a weighted sum of its inputs:

$$Sum = in_0 \cdot w_0 + in_1 \cdot w_1 + \ldots + in_n \cdot w_n + bias$$

Here, $in_0$ to $in_n$ are outputs of nodes in the previous layer, while $w_0$ to $w_n$ are connection weights; each node has its own bias value. Then the activation function is applied to $Sum$ to generate the output of the node.

A sigmoid (S-shaped) function is used as the activation function in hidden layer nodes. Specifically, NeuralTools uses the hyperbolic tangent function. In NeuralTools the output node uses identity as the activation function; that is, it simply returns the weighted sum of its inputs. Neural nets are sometimes constructed with sigmoid activation functions in output nodes. However, that is not needed for a neural net to be able to approximate complex functions. Moreover, sigmoid functions have restricted output range (-1 to 1 for the hyperbolic tangent function), and there will typically be dependent values outside the range. Thus using a sigmoid function in the output node would force an additional transformation of output values before passing training data to the net.

When MLF nets are used for classification, they have multiple output nodes, one corresponding to each possible dependent category. A net classifies a case by computing its numeric outputs; the selected category is the one corresponding to the node that outputs the largest value.
MLF Net Training

Training an MLF net consists of finding a set of connection weights and bias terms that will cause the net to give approximately correct output values when presented with new cases (for simplicity the bias term will be omitted in the presentation below). Training starts by assigning a set of randomly selected connection weights. A prediction is made for each training case (by presenting independent values as inputs to obtain the output). The output will most likely be different from the known dependent value. The difference is an error value. These error values lead to an error measure for the entire training set; it indicates how well the net does given the initial weights.

The net will probably not do very well with the random initial assignment of weights, so the process performs subsequent trials with other assignments of weights. However, the assignments of weights are no longer random, but rather are determined by the training algorithm, that is, the method for selecting connection weights based on results of previous trials. This becomes an optimization problem, where the goal is to minimize the error measure by changing connection weights.

The first successful algorithm for training connection weights in MLF nets was called “back-propagation.” Researchers now tend to favor other algorithms that are faster and more likely to find the global optimum. NeuralTools uses the “Conjugate Gradient Descent” method, belonging to the category of “second-order” optimization methods. These optimization methods are designed to find the local minimum of a function: they proceed efficiently down the slope of the error function. To reduce the risk of finding the local rather than the global minimum, NeuralTools combines “deterministic” with “stochastic” optimization methods. Specifically, the stochastic Simulated Annealing method is used along with the Conjugate Gradient Descent method. The algorithm decides which method to use at a particular point, based on the results of previous trials. For more information on the Conjugate Gradient Descent method, see Bishop (1995) and Masters (1995). For more information on Simulated Annealing, see Masters (1995).
The error measure used when training numeric prediction nets is the Mean Squared Error over all training cases, that is, the mean squared difference between the actual output values and the output values given by the net.

With classification, there are multiple outputs for each training case (with one output corresponding to each dependent category). In this case, the Mean Squared Error is computed, over all the outputs for all the training cases, by reference to the desired output values. That is, for each training case, the output value should be close to 1 for the output corresponding to the correct category, and the remaining output values should be close to 0.

The NeuralTools MLF training algorithms restarts itself multiple times from different initial starting weights. Therefore, the longer a net is trained, the better. The more times it is allowed to restart itself, the more likely it is that the global minimum of the error function will be found.

The selection of the number of layers and the numbers of nodes in the layers determines whether the net is capable of learning the relationship between the independent variables and the dependent variable. Typically, a net with a single hidden layer and two hidden nodes will not train to a satisfactory error level. However, increasing the number of layers and nodes comes at a price that is often not worth paying. A single hidden layer is sufficient for almost any problem; using two layers will typically result in unnecessarily long training times. Moreover, a few nodes in a single hidden layer are typically sufficient.

NeuralTools can auto-configure the net topology based on training data. However, the Best Net Search feature offers a more reliable approach. As part of the Best Net Search, a range of single-hidden-layer nets with different numbers of nodes will be trained. By default, five MLF nets, with 2 to 6 hidden nodes will be included. If sufficient time is available, the range can be broadened; but it is recommended that you start with a 2-node net, for reasons related to preventing over-training.
The term “over-training” refers to the situation where the net learns not only the general characteristics of the relationship between independent variables and the dependent variable, but instead starts learning facts about training cases that will not apply in general—that is, they will not apply to cases not included in training. To address this problem, the testing set is sometimes divided into testing-while-training set, and the “proper” testing set, to be used after training. The error on the testing-while-training set is periodically computed during training. When it starts to increase, this is taken as evidence that the net is beginning to over-train, and training is stopped.

NeuralTools takes a different approach to preventing over-training. The approach with two distinct testing sets is often unrealistic, with not enough data to split into a training set and two testing sets. Also, the increase of error on a testing-while-training set is not necessarily a reliable indicator of over-training; the increase could be local, and the error might continue to decrease with more training. NeuralTools’ Best Net Search is designed to prevent over-training. With default settings, Best Net Search will start with a net with 2 nodes, which is typically too small to get over-trained. With default settings, Best Net Search will train nets with up to 6 neurons. If the nets with 5 and 6 neurons over-train, this will show in the results from the single testing set; one of the nets with 2, 3 or 4 neurons will have the lowest testing error.
Generalized Regression Neural Nets and Probabilistic Neural Nets

Generalized Regression Neural Nets and Probabilistic Neural Nets are based on similar ideas. GRN nets are used for numeric prediction/function approximation, while PN nets are used for category prediction/classification. Both types of nets were put forward by Donald Specht ("Probabilistic Neural Networks", Neural Networks, 3, 1990, pp. 109-118; "A General Regression Neural Network", IEEE Transactions on Neural Networks, 2, 1991, pp. 568-576). They are covered in Masters (1995), whose presentation is summarized below. Please consult these sources for additional details.

**Generalized Regression Neural Nets**

By way of example, consider the training data set presented in the graph, with one independent numeric variable, and one dependent numeric variable.

A human observer can discern a pattern in the data. If asked about the unknown dependent value for the independent value 6, you can estimate it as greater than 200 and smaller than 400. Note that this estimate is not based on the two closest known cases, which would indicate a value below 200; you look at cases beyond the closest ones.
However, you do not pay much attention to cases with independent values around -10. The closer a known case is to the unknown one, the more weight it is given when estimating the unknown dependent value. The Generalized Regression Neural Net is built on these intuitive ideas. Every training case is represented in the net. When presented with a case, the net computes the predicted dependent value using the dependent values of every training case, with closer training cases contributing more significantly to the value of the output.

A Generalized Regression Neural Net for two independent numeric variables is structured as shown in the graph (assuming there are just three training cases):

The Pattern Layer contains one node for each training case. Presenting a training case to the net consists here of presenting two independent numeric values. Each node in the pattern layer computes its distance from the presented case. The values passed to the Numerator and Denominator Nodes are functions of the distance and the dependent value. The two nodes in the Summation Layer sum its inputs, while the Output Node divides them to generate the prediction.

The distance function computed in the Pattern Layer nodes uses “smoothing factors”; every input has its own smoothing factor value. With a single input, the greater the value of the smoothing factor, the more significant distant training cases become for the predicted value. With two inputs, the smoothing factor relates to the distance along one axis on a plane, and in general, with multiple inputs, to one dimension in multi-dimensional space.
Training a GRN net consists of optimizing smoothing factors to minimize the error on the training set, and the Conjugate Gradient Descent optimization method is used to accomplish this. The error measure used during training to evaluate different sets of smoothing factors is the Mean Squared Error. However, when computing the Squared Error for a training case, that case is temporarily excluded from the Pattern Layer. This is because the excluded node would compute a zero distance, making other node insignificant in the computation of the prediction.

Probabilistic Neural Nets

Turning to Probabilistic Neural Nets, consider the following training data set with two independent numeric variables, and a dependent variable with two categories:

The circles represent training cases in one category, while the squares designate those belonging to the other category. We want to predict the category of the case shown as the question mark. A human observer will decide that the case is more likely in the circle category than the square category. However, many classification methods will not be able to reach the same conclusions. Methods that require linear separability of categories will fail. Nearest neighbor methods will assign the unknown case into the square category. So will methods that focus on central tendencies, since the unknown case is closer to the centroid of the square category than to the centroid of the circle category.

On the other hand, a PN net will make the correct prediction. It will consider the distance of the new case to every training case, giving greater weight to closer cases. The effect of the neighboring square will be outweighed by the circles in the immediate vicinity.
A Probabilistic Neural Net is structured as shown in the following graph, which assumes there are two independent numeric variables, two dependent categories, and five training cases (three in one category and two in the other):

When a case is presented to the net, each node in the Pattern Layer computes the distance between the training case represented by the node, and the input case. The value passed to Summation Layer nodes is a function of the distance and smoothing factors. As with GRN nets, each input has its own smoothing factor; these factors determine how rapidly the significance of training cases decreases with distance. In the Summation Layer there is one node per dependent category; each node sums the output values for the nodes corresponding to the training cases in that category. The output values of the Summation Layer nodes can be interpreted as probability density function estimates for each class. The output node selects the category with the highest probability density function value as the predicted category.

Like with GRN nets, training a PN net consists of optimizing smoothing factors to minimize the error on the training set, and the Conjugate Gradient Descent optimization method is used. The error measure used during training to evaluate different sets of smoothing factors is computed based on all the values returned by nodes in the Summation Layer for all the training cases. The measure takes into account not only the probability assigned to the correct category, but also distribution of probabilities assigned to incorrect categories (approximately uniform distribution of probabilities among incorrect categories is better than some incorrect category having a large probability). Note that when computing the error for a training case, that case is temporarily excluded from the Pattern Layer. This is because the excluded node would compute a zero distance, making other nodes insignificant in the computation.
Comparison of MLF Nets to PN/GRN Nets

Each of the types of neural nets available in NeuralTools has advantages and disadvantages, as described here:

Advantages of GRN/PN nets:

- They train fast.
- They do not require topology specification (numbers of hidden layers and nodes).
- PN nets not only classify, but also return the probabilities that the case falls in different possible dependent categories.

Advantages of MLF nets:

- They are smaller in size, thus faster to make predictions.
- They are more reliable outside the range of training data (for example, when the value of some independent variable falls outside the range of values for that variable in the training data). However, prediction outside the range of training data is always risky, including with MLF nets.
- They are capable of generalizing from very small training sets.
Recommended Readings

The following texts provide additional background on the neural nets used in NeuralTools:


Index

A
Application Settings Command. See NeuralTools Commands

B
Best Net Search, 16, 44

C
Case. See NeuralTools
Classification Matrix, 51, 57
Classification Problems, 11

D
Data Set. See NeuralTools
Data Set Manager. See NeuralTools
Data Set Manager Command. See NeuralTools Commands
Data Viewer, 33
Data Viewer Command. See NeuralTools Commands
DecisionTools Suite. See Palisade Corporation
Developer Kit (XDK). See NeuralTools

G
Generalized Regression Neural Nets (GRNN), 16, 44, 79, 87

M
Missing Data Utilities Command. See NeuralTools Commands
Multi-Layer Feedforward Networks (MLF), 16, 44, 79, 81
Versus PN/GRN Nets, 91

N
Neural Net, 77
Prediction, 20
References, 92
Structure, 78
Testing, 19
Training, 15, 78
Versus Statistical Methods, 77
Neural Net Manager, 69
Neural Net Manager Command. See NeuralTools Commands
NeuralTools
Cases, 28
Data Set, 13
Data Set Limits, 32
Data Set Manager, 12, 13, 28
Detailed Reports, 22
Developer Kit (XDK), 10
Example Spreadsheets, 10
Guided Tour Videos, 10
Icons, 25
Input Transformation, 79
Installation, 9
Instructional Materials, 10
License Manager, 9
Live Prediction, 22, 62
Net Configurations, 16, 44
Prediction, 12, 20
Prediction Reports, 64
Professional and Industrial
Versions, 5
Quick Start Videos, 10
Quick Summary Reports, 22, 59
Report Settings, 66
Ribbon, 13
Runtime Settings, 47
Software Activation, 9
Student Version, 8
Summary Reports, 22
System Requirements, 8
Tag Variable, 31, 41
Testing, 12, 19
Testing Reports, 20, 56
Training, 12, 15
Training Progress Window, 18, 49
Training Reports, 19, 49
Uninstalling, 9
Using with Evolver, 24
Using with StatTools, 24
Variable Impact Analysis, 41
Variable Matching, 54
Variable Types, 15, 30
Variables, 28
NeuralTools Commands, 26
Application Settings, 65
Data Set Manager, 28
Data Viewer, 33
Missing Data Utilities, 71
Neural Net Manager, 69
Predict, 60
Test, 53
Testing Sensitivity, 74
Train, 39
Numeric Problems, 11

Palisade Corporation
Contacting, 7
DecisionTools Suite, 9
Getting Help, 6
Predict Command. See NeuralTools
Commands
Prediction. See NeuralTools
Probabilistic Neural Nets (PNN), 16,
44, 79, 89
Versus MLF Nets, 91

RMSE. See Root Mean Square Error
Root Mean Square Error, 52, 58

Tag Variable. See NeuralTools
Test Command. See NeuralTools
Commands
Testing. See NeuralTools
Testing Sensitivity Command. See
NeuralTools Commands
Train Command. See NeuralTools
Commands
Training. See NeuralTools

Variable. See NeuralTools
Variable Impact Analysis. See
NeuralTools