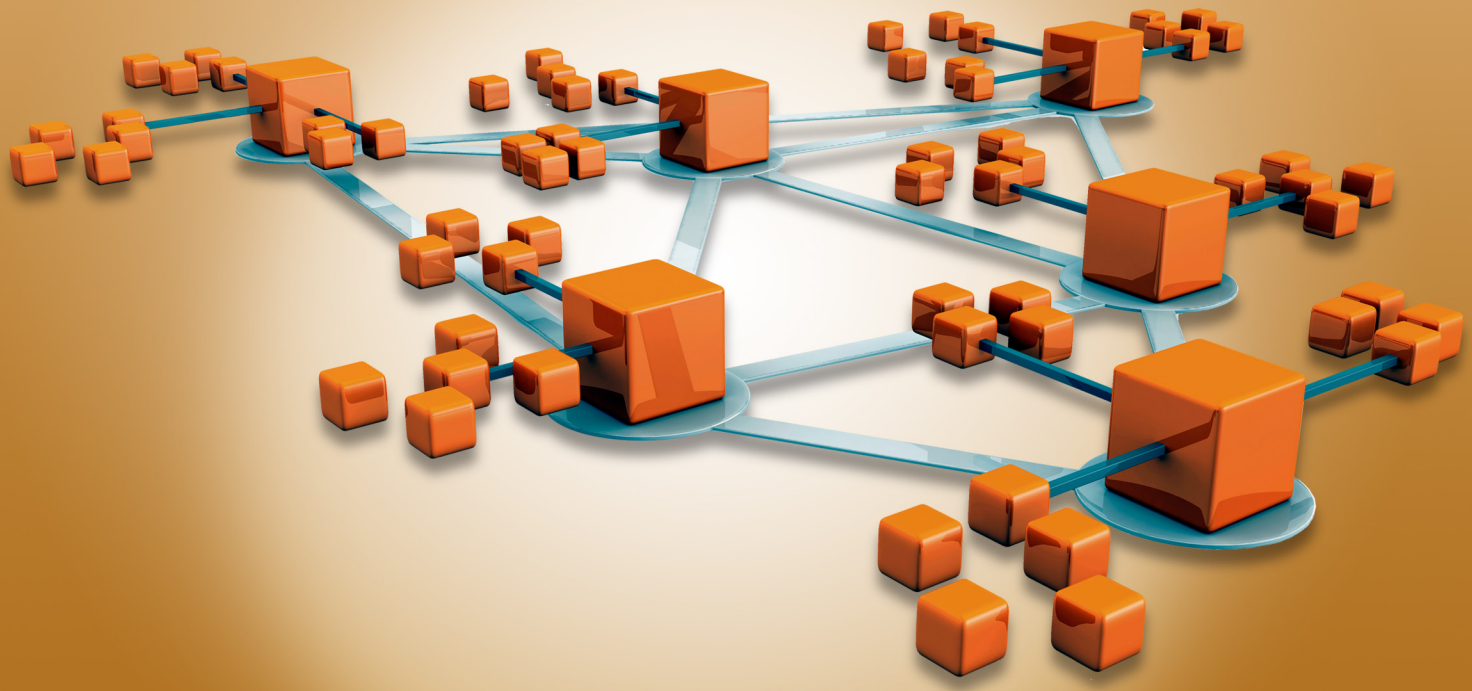
sas

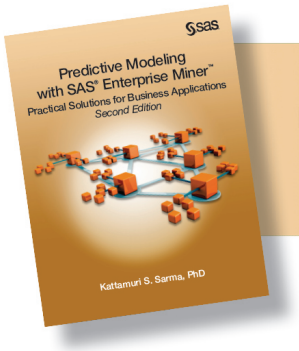
Predictive Modeling with SAS® Enterprise Miner™

Practical Solutions for Business Applications

Second Edition

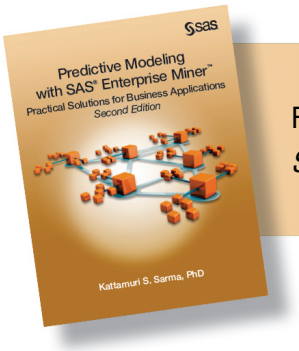


Kattamuri S. Sarma, PhD



From *Predictive Modeling with SAS® Enterprise Miner™, Second Edition.* Full book available for purchase here.

Contents



Note: This short excerpt was taken from a chapter in the published book.

5.4 A Neural Network Model to Predict Response

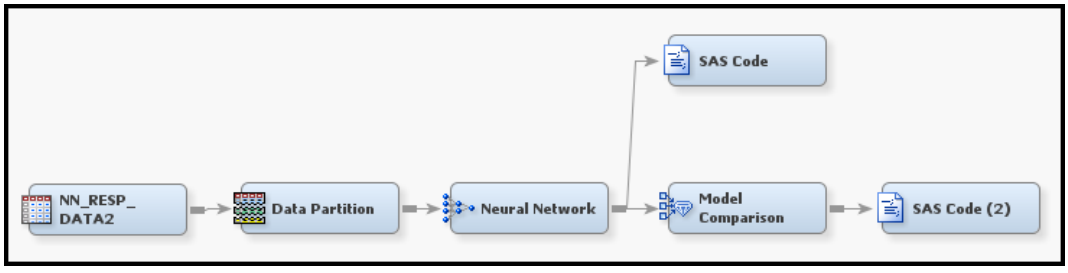
This section discusses the neural network model developed to predict the response to a planned direct mail campaign. The campaign's purpose is to solicit customers for a hypothetical insurance company. A two-layered network with one hidden layer was chosen. Three units are included in the hidden layer. In the hidden layer, the combination function chosen is linear, and the activation function is hyperbolic tangent. In the output layer, a logistic activation function and Bernoulli error function are used. The logistic activation function results in a logistic regression type model with non-linear transformation of the inputs, as shown in Equation 5.14 in Section 5.2.4. Models of this type are in general estimated by minimizing the Bernoulli error functions shown in Equation 5.16. Minimization of the Bernoulli error function is equivalent to maximizing the likelihood function.

Display 5.1 shows the process flow for the response model. The first node in the process flow diagram is the **Input Data** node, which makes the SAS data set available for modeling. The next node is **Data Partition**, which creates the Training, Validation, and Test data sets. The Training data set is used for preliminary model fitting. The Validation data set is used for selecting the optimum weights. The **Model Selection Criterion** property is set to Average Error.

As pointed out earlier, the estimation of the weights is done by minimizing the error function. This minimization is done by an iterative procedure. Each iteration yields a set of weights. Each set of weights defines a model. If I set the **Model Selection Criterion** property to Average Error, the algorithm selects the set of weights that results in the smallest error, where the error is calculated from the Validation data set.

Since both the Training and Validation data sets are used for parameter estimation and parameter selection, respectively, an additional holdout data set is required for an independent assessment of the model. The Test data set is set aside for this purpose.

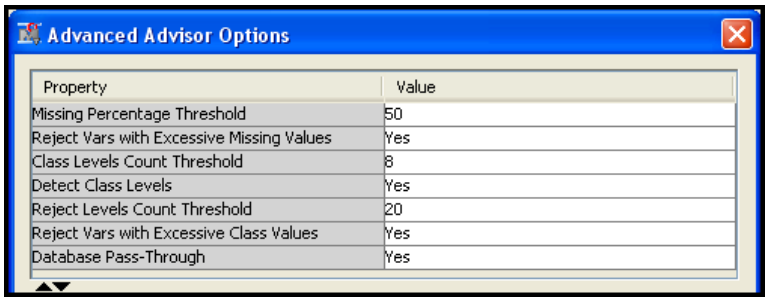
Display 5.1



Input Data Node

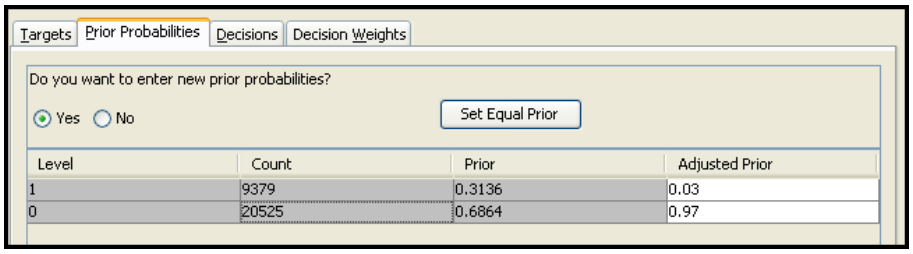
I create the data source for the **Input Data** node from the data set NN_RESP_DATA2. I create the metadata using the Advanced Advisor Options, and I customize it by setting the **Class Levels Count Threshold** property to 8, as shown in Display 5.2

Display 5.2



I set adjusted prior probabilities to 0.03 for response and 0.97 for non-response, as shown in Display 5.3.

Display 5.3



Data Partition Node

The input data is partitioned such that 60% of the observations are allocated for training, 30% for validation, and 10% for Test, as shown in Display 5.4.

Display 5.4

Data Set Allocations	
Training	60.0
Validation	30.0
Test	10.0

5.4.1 Setting the Neural Network Node Properties

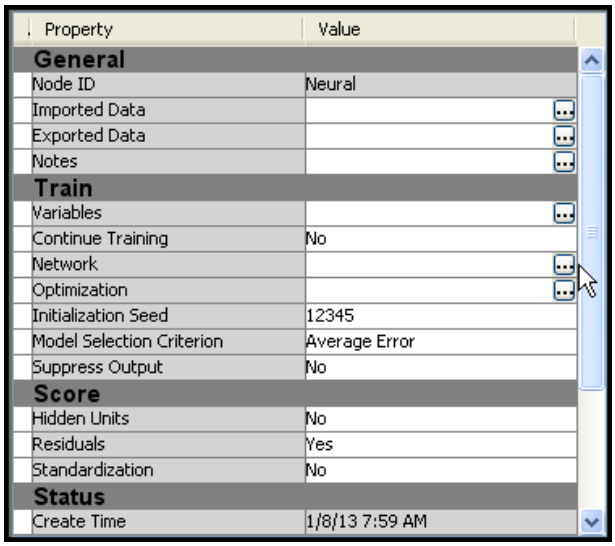
Here is a summary of the neural network specifications for this application:

- One hidden layer with three neurons
- Linear combination functions for both the hidden and output layers
- Hyperbolic tangent activation functions for the hidden units
- Logistic activation functions for the output units
- The Bernoulli error function
- The **Model Selection Criterion** is Average Error

These settings are shown in Displays 5.5–5.7.

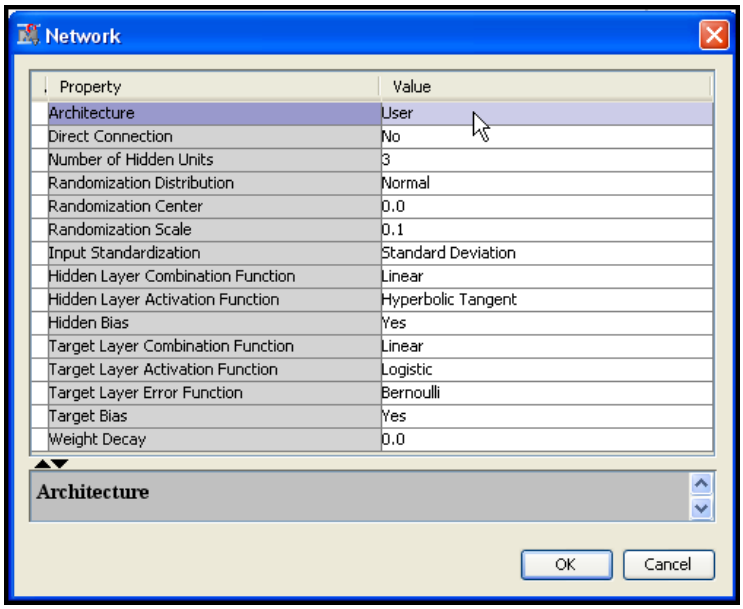
Display 5.5 shows the Properties panel for the **Neural Network** node.

Display 5.5



To define the network architecture, click  located to the right of the **Network** property. The Network Properties panel opens, as shown in Display 5.6.

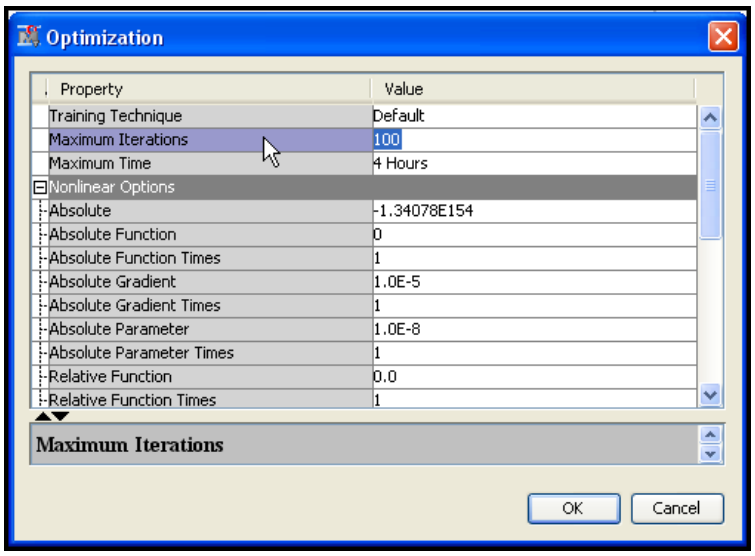
Display 5.6



Set the properties as shown in Display 5.6 and click **OK**.

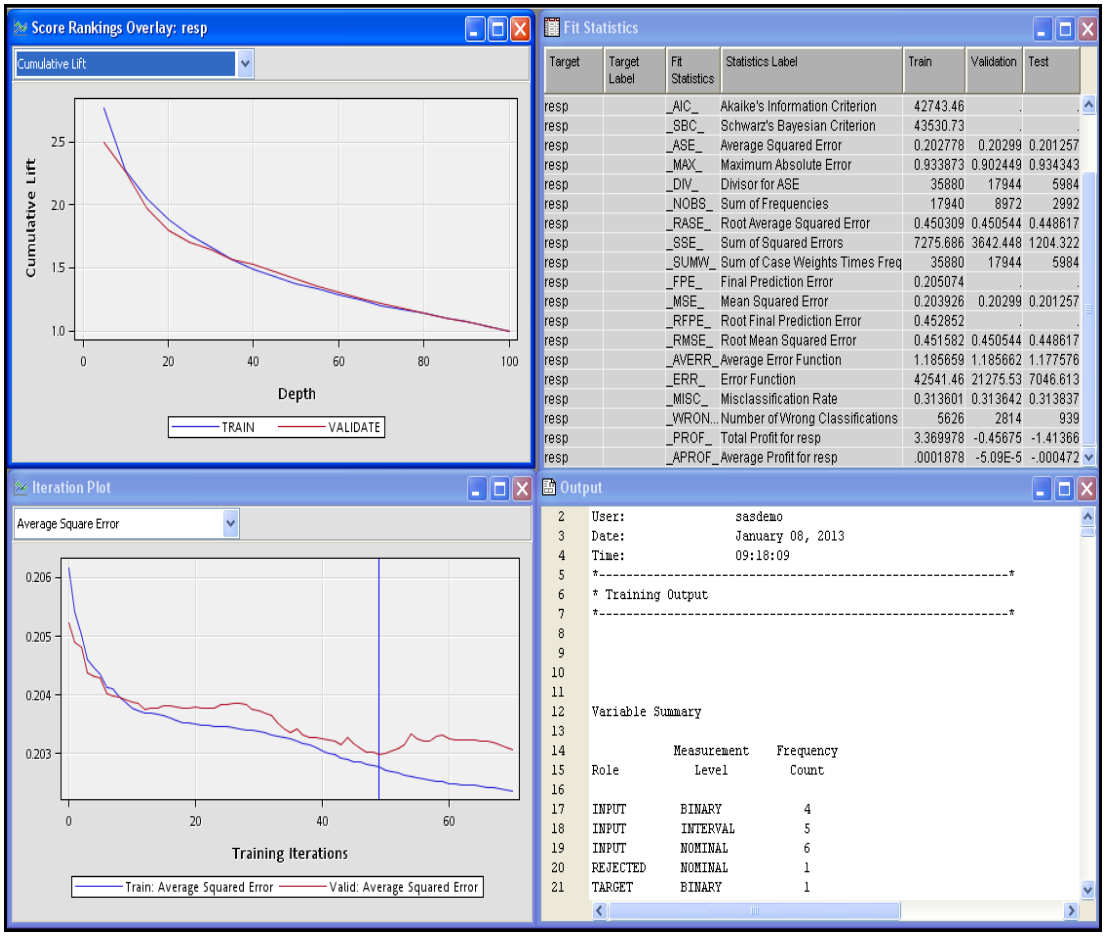
To set the iteration limit, click  located to the right of the **Optimization** property. The Optimization Properties panel opens, as shown in Display 5.7. Set **Maximum Iterations** to 100.

Display 5.7



After running the **Neural Network** node, you can open the Results window, shown in Display 5.8. The window contains four windows: Score Rankings Overlay, Iteration Plot, Fit Statistics, and Output.

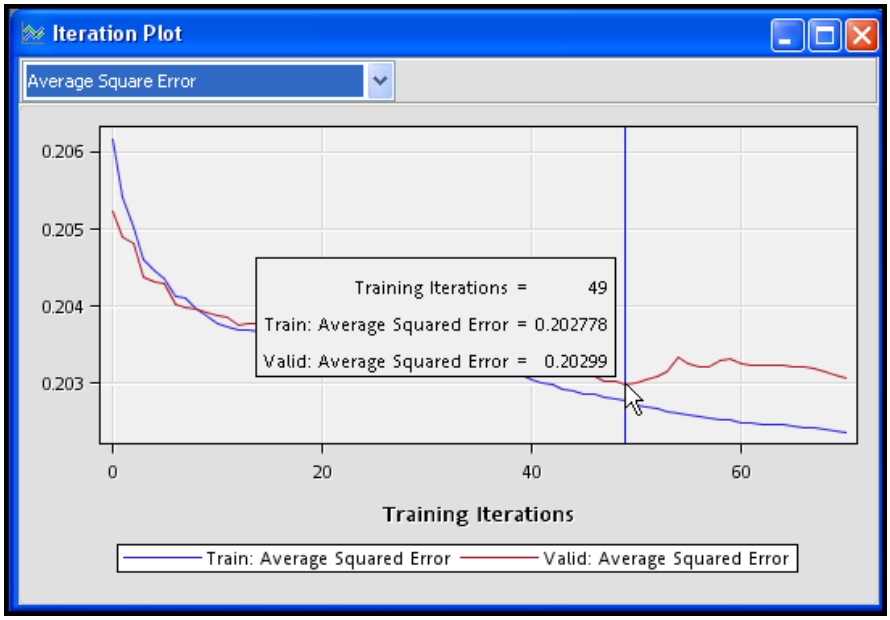
Display 5.8



The Score Rankings Overlay window in Display 5.8 shows the cumulative lift for the Training and Validation data sets. Click the down arrow next to the text box to see a list of available charts that can be displayed in this window.

Display 5.9 shows the iteration plot with Average Squared Error at each iteration for the Training and Validation data sets. The estimation process required 70 iterations. The weights from the 49th iteration were selected. After the 49th iteration, the Average Squared Error started to increase in the Validation data set, although it continued to decline in the Training data set.

Display 5.9



You can save the table corresponding to the plot shown in Display 5.9 by clicking the **Tables** icon and then selecting **File→ Save As**. Table 5.1 shows the three variables _ITER_ (iteration number), _ASE_ (Average Squared Error for the Training data), and _VASE_ (Average Squared Error from the Validation data) at iterations 41-60.

Table 5.1

Training Iterations	Train: Average Squared Error	Valid: Average Squared Error
41	0.20300	0.20324
42	0.20298	0.20321
43	0.20293	0.20314
44	0.20291	0.20328
45	0.20287	0.20316
46	0.20285	0.20308
47	0.20282	0.20303
48	0.20280	0.20303
49	0.20278	0.20299
50	0.20272	0.20301
51	0.20268	0.20304
52	0.20267	0.20308
53	0.20263	0.20314
54	0.20260	0.20334
55	0.20258	0.20325
56	0.20258	0.20320
57	0.20255	0.20322
58	0.20253	0.20330
59	0.20252	0.20332
60	0.20248	0.20326

You can print the variables _ITER_, _ASE_, and _VASE_ by using the SAS code shown in Display 5.10.

Display 5.10

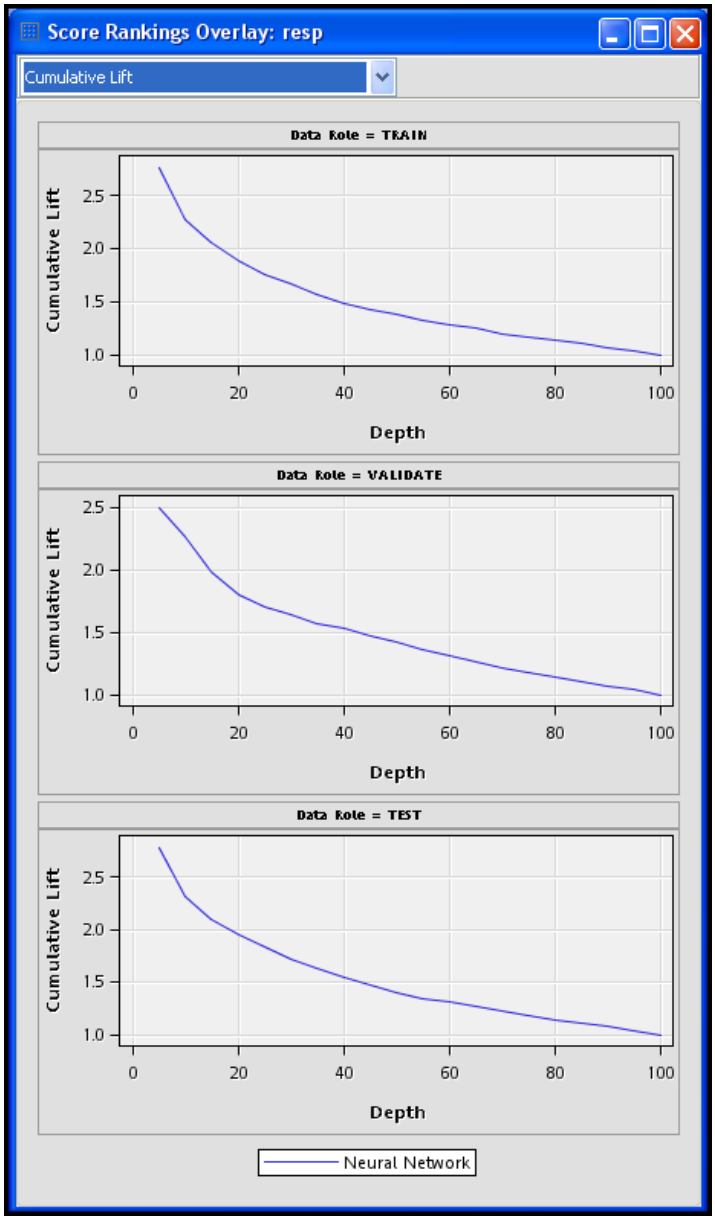
Training Code

```
proc print data=&em_lib..neural_plotds noobs label;
 var _ITER_ _ASE_ _VASE_ ;
 where 41 le _ITER_ le 60;
run;
```

5.4.2 Assessing the Predictive Performance of the Estimated Model

In order to assess the predictive performance of the neural network model, run the **Model Comparison** node and open the Results window. In the Results window, the Score Rankings Overlay shows the Lift charts for the Training, Validation, and Test data sets. These are shown in Display 5.11.

Display 5.11



Click the arrow in the box at the top left corner of the Score Ranking Overlay window to see a list of available charts.

SAS Enterprise Miner saves the Score Rankings table as EMWS.MdlComp_EMRANK. Tables 5.2, 5.3, and 5.4 are created from the saved data set using the simple SAS code shown in Display 5.12.

Display 5.12

Training Code

```
options center ;
proc print data=&EM_LIB..MdlComp_EMRANK label noobs ;
 where upcase(datarole) = "TRAIN" and bin ne . ;
 var bin decile resp respc lift liftc cap capc ;
 title "Lift and Capture Rates: Training Data set";
run;
proc print data=&EM_LIB..MdlComp_EMRANK label noobs;
 where upcase(datarole) = "VALIDATE" and bin ne . ;
 var bin decile resp respc lift liftc cap capc ;
 title "Lift and Capture Rates: Validation Data set";
run;
proc print data=&EM_LIB..MdlComp_EMRANK label noobs;
 where upcase(datarole) = "TEST" and bin ne . ;
 var bin decile resp respc lift liftc cap capc ;
 title "Lift and Capture Rates: Test Data set";
run;
```

Table 5.2

Lift and Capture Rates: Training Data set

Bin	Depth	% Response	Cumulative % Response	Lift	Cumulative Lift	% Captured Response	Cumulative % Captured Response
1	5	8.31775	8.31775	2.77258	2.77258	13.8642	13.8642
2	10	5.31131	6.81464	1.77044	2.27155	8.8518	22.7160
3	15	4.84188	6.15707	1.61396	2.05236	8.0697	30.7856
4	20	4.12431	5.64861	1.37477	1.88287	6.8788	37.6644
5	25	3.83131	5.28540	1.27710	1.76180	6.3811	44.0455
6	30	3.58335	5.00173	1.19445	1.66724	5.9723	50.0178
7	35	2.91134	4.70309	0.97045	1.56770	4.8525	54.8702
8	40	2.78073	4.46259	0.92691	1.48753	4.6392	59.5094
9	45	2.91268	4.29046	0.97089	1.43015	4.8525	64.3619
10	50	2.64467	4.12588	0.88156	1.37529	4.4081	68.7700
11	55	2.61352	3.98844	0.87117	1.32948	4.3548	73.1248
12	60	2.46397	3.86142	0.82132	1.28714	4.1059	77.2307
13	65	2.34569	3.74481	0.78190	1.24827	3.9104	81.1411
14	70	1.71661	3.59991	0.57220	1.19997	2.8617	84.0028
15	75	2.15451	3.50356	0.71817	1.16785	3.5905	87.5933
16	80	1.97366	3.40798	0.65789	1.13599	3.2883	90.8816
17	85	1.67467	3.30604	0.55822	1.10201	2.7906	93.6722
18	90	1.58819	3.21056	0.52940	1.07019	2.6484	96.3207
19	95	1.34421	3.11236	0.44807	1.03745	2.2396	98.5603
20	100	0.86420	3.00000	0.28807	1.00000	1.4397	100.000

Table 5.3

Lift and Capture Rates: Validation Data set

Bin	Depth	% Response	Cumulative % Response	Lift	Cumulative Lift	% Captured Response	Cumulative % Captured Response
1	5	7.49586	7.49586	2.49862	2.49862	12.5089	12.5089
2	10	6.05928	6.77825	2.01976	2.25942	10.0924	22.6013
3	15	4.24246	5.93308	1.41415	1.97769	7.0718	29.6731
4	20	3.84086	5.41043	1.28029	1.80348	6.3966	36.0697
5	25	3.85782	5.09981	1.28594	1.69994	6.4321	42.5018
6	30	4.00842	4.91792	1.33614	1.63931	6.6809	49.1827
7	35	3.42772	4.70477	1.14257	1.56826	5.7214	54.9041
8	40	3.65246	4.57347	1.21749	1.52449	6.0768	60.9808
9	45	3.11138	4.41094	1.03713	1.47031	5.1883	66.1692
10	50	2.78928	4.24858	0.92976	1.41619	4.6553	70.8244
11	55	2.26190	4.06814	0.75397	1.35605	3.7669	74.5913
12	60	2.28275	3.91945	0.76092	1.30648	3.8024	78.3937
13	65	2.08817	3.77850	0.69606	1.25950	3.4826	81.8763
14	70	1.90004	3.64450	0.63335	1.21483	3.1628	85.0391
15	75	2.04643	3.53794	0.68214	1.17931	3.4115	88.4506
16	80	1.59942	3.41680	0.53314	1.13893	2.6652	91.1158
17	85	1.61828	3.31087	0.53943	1.10362	2.7008	93.8166
18	90	1.57989	3.21483	0.52663	1.07161	2.6297	96.4463
19	95	1.32169	3.11518	0.44056	1.03839	2.2033	98.6496
20	100	0.81060	3.00000	0.27020	1.00000	1.3504	100.000

Table 5.4

Lift and Capture Rates: Test Data set

Bin	Depth	% Response	Cumulative % Response	Lift	Cumulative Lift	% Captured Response	Cumulative % Captured Response
1	5	8.36796	8.36796	2.78932	2.78932	13.9510	13.9510
2	10	5.49570	6.93212	1.83190	2.31071	9.1587	23.1097
3	15	5.02337	6.29385	1.67446	2.09795	8.4132	31.5229
4	20	4.47671	5.84044	1.49224	1.94681	7.4547	38.9776
5	25	4.17028	5.50778	1.39009	1.83593	6.9223	45.8999
6	30	3.50818	5.17403	1.16939	1.72468	5.8573	51.7572
7	35	3.12520	4.88096	1.04173	1.62699	5.2183	56.9755
8	40	3.01327	4.64829	1.00442	1.54943	5.0053	61.9808
9	45	2.54180	4.41310	0.84727	1.47103	4.2599	66.2407
10	50	2.30884	4.20349	0.76961	1.40116	3.8339	70.0745
11	55	2.17732	4.01970	0.72577	1.33990	3.6209	73.6954
12	60	3.00147	3.93481	1.00049	1.31160	5.0053	78.7007
13	65	2.03820	3.78849	0.67940	1.26283	3.4079	82.1086
14	70	2.17732	3.67367	0.72577	1.22456	3.6209	85.7295
15	75	1.52626	3.52989	0.50875	1.17663	2.5559	88.2854
16	80	1.66239	3.41329	0.55413	1.13776	2.7689	91.0543
17	85	1.86309	3.32260	0.62103	1.10753	3.0884	94.1427
18	90	1.85065	3.24072	0.61688	1.08024	3.0884	97.2311
19	95	1.14908	3.13055	0.38303	1.04352	1.9169	99.1480
20	100	0.51256	3.00000	0.17085	1.00000	0.8520	100.000

The lift and capture rates calculated from the Test data set (shown in Table 5.4) should be used for evaluating the models or comparing the models because the Test data set is not used in training or fine-tuning the model.

To calculate the lift and capture rates, SAS Enterprise Miner first calculates the predicted probability of response for each record in the Test data. Then it sorts the records in descending order of the predicted probabilities (also called the scores) and divides the data set into 20 groups of equal size. In Table 5.4, the column Bin shows the ranking of these groups. If the model is accurate, the table should show the highest actual

response rate in the first bin, the second highest in the next bin, and so on. From the column %Response, it is clear that the average response rate for observations in the first bin is 8.36796%. The average response rate for the entire test data set is 3%. Hence the lift for Bin 1, which is the ratio of the response rate in Bin 1 to the overall response rate, is 2.7893. The lift for each bin is calculated in the same way. The first row of the column Cumulative %Response shows the response rate for the first bin. The second row shows the response rate for bins 1 and 2 combined, and so on.

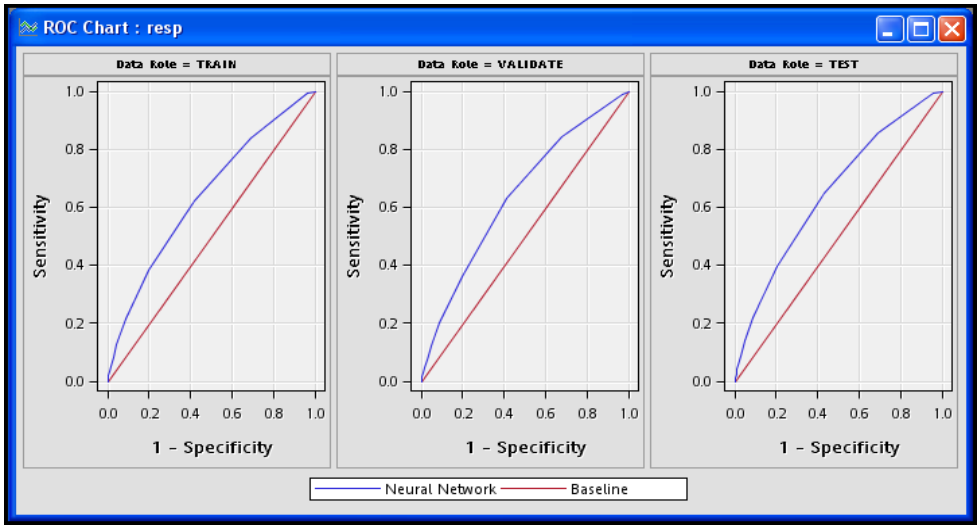
The capture rate of a bin shows the percentage of likely responders that it is reasonable to expect to be captured in the bin. From the column Captured Response, you can see that 13.951% of all responders are in Bin 1.

From the Cumulative % Captured Response column of Table 5.3, you can be seen that, by sending mail to customers in the first four bins, or the top 20% of the target population, it is reasonable to expect to capture 39% of all potential responders from the target population. This assumes that the modeling sample represents the target population.

5.4.3 Receiver Operating Characteristic (ROC) Charts

Display 5.13, taken from the Results window of the **Model Comparison** node, displays ROC curves for the Training, Validation, and Test data sets. An ROC curve shows the values of the *true positive fraction* and the *false positive fraction* at different *cut-off values*, which can be denoted by P_c. In the case of the response model, if the estimated probability of response for a customer record were above a cut-off value P_c, then you would classify the customer as a responder; otherwise, you would classify the customer as a non-responder.

Display 5.13



In the ROC chart, the true positive fraction is shown on the vertical axis, and the false positive fraction is on the horizontal axis for each cut-off value (P_c).

If the calculated probability of response (P_resp1) is greater than equal to the cut-off value, then the customer (observation) is classified as a responder. Otherwise, the customer is classified as non-responder.

True positive fraction is the proportion of responders correctly classified as responders. The false positive fraction is the proportion of non-responders incorrectly classified as responders. The true positive fraction is also called *sensitivity,* and *specificity* is the proportion of non-responders correctly classified as non-responders. Hence, the false positive fraction is 1-specificity. An ROC curve reflects the tradeoff between sensitivity and specificity.

The straight diagonal lines in Display 5.13 that are labeled Baseline are the ROC charts of a model that assigns customers at random to the responder group and the non-responder group, and hence has no predictive power. On these lines, sensitivity = 1- specificity at all cut-off points. The larger the area between the ROC curve of the model being evaluated and the diagonal line, the better the model. The area under the ROC curve is a measure of the predictive accuracy of the model and can be used for comparing different models.

Table 5.5 shows sensitivity and 1-specificity at various cut-off points in the validation data.

Table 5.5

ROC Table: Validation Data

Cutoff	Sensitivity	1-Specificity
1.00000	0.00000	0.00000
0.26583	0.00036	0.00000
0.22075	0.00071	0.00000
0.15334	0.00071	0.00016
0.14645	0.00142	0.00016
0.13954	0.00284	0.00049
0.12783	0.00391	0.00097
0.11956	0.00640	0.00162
0.10987	0.00995	0.00325
0.09991	0.01741	0.00503
0.08999	0.02665	0.01007
0.07991	0.04655	0.01689
0.06999	0.07285	0.02598
0.05995	0.12296	0.04644
0.04999	0.20220	0.08834
0.03998	0.36283	0.19584
0.03000	0.63184	0.41410
0.02000	0.84115	0.68139
0.01000	0.98969	0.96963
0.00000	1.00000	1.00000

From Table 5.5, you can see that at a cut-off probability (P_c) of 0.02, for example, the sensitivity is 0.84115. That is, at this cut-off point, you will correctly classify 84.1% of responders as responders, but you will also *incorrectly* classify 68.1% of non-responders as responders, since 1-specificity at this point is 0.68139. If instead you chose a much higher cut-off point of P_c = 0.13954, you would classify 0.284% of true responders as responders and 0.049% of non-responders as responders. In this case, by increasing the cut-off probability beyond which you would classify an individual as a responder, you would be reducing the fraction of false positive decisions made, while, at the same time, also reducing the fraction of true positive decisions made. These pairings of a true positive fraction with a false positive fraction are plotted as the ROC curve for the VALIDATE case in Display 5.13.

The SAS macro in Display 5.14 demonstrates the calculation of the true positive rate (TPR) and the false positive rate (FPR) at the cut-off probability of 0.02.

Display 5.14

```
%macro roccalc(PC=);
ods html file ="C:/TheBook/EM12.1/Reports/Chapter5/RespRateV.html";
title "Validation Data Set";
proc freq data=&EM_LIB..MdlComp_Validate;
 table resp / noperc nocumperc out=tabl(keep= resp count rename=(count=N));
run;
ods html close ;
ods html file ="C:/TheBook/EM12.1/Reports/Chapter5/RespRateV_cutoff.html";
Title "Cases with Predicted Probability (P_resp1) GE &PC";
title2 "Validation Data Set";
proc freq data=&EM_LIB..MdlComp_Validate;
  table resp / noperc nocumperc out=tab2(keep= resp count rename=(count=NC));
  where P_resp1 ge &PC;
run;
ods html close;
data temp;
 merge tabl tab2 ;
 by resp ;
 if resp=0 then TYPE='FPR' ; else if resp=1 then TYPE='TPR'; Rate= NC/N;
 cutoff=&pc;
 run;
ods html file ="C:/TheBook/EM12.1/Reports/Chapter5/TPR_FPR_Cutoff.html";
 Title "True Positive Rate (TPR) and False Positive Rate (FPR)";
 title2 "at cut-off = &PC";
proc print data=temp label noobs;
 var RESP N NC TYPE RATE ;
 label N = "Number of Observations in the sample";
 label NC = "Number of Observations classified as responders ";
 label RESP = "Actual RESPONSE ";
 label TYPE = "ROC Coordinate  ";
 label Rate = "ROC Coordinate Value";
 run;
ods html close;
%mend roccalc;
%roccalc(PC=0.02);
```

Tables 5.6, 5.7, and 5.8, generated by the macro shown in Display 5.14, show the sequence of steps for calculating TPR and FPR for a given cut-off probability.

Table 5.6

Validation Data Set

The FREQ Procedure

resp	Frequency
0	6158
1	2814

Table 5.7

Cases with Predicted Probability (P_resp1) GE 0.02
Validation Data Set

The FREQ Procedure

resp	Frequency
0	4196
1	2367

Table 5.8

True Positive Rate (TPR) and False Positive Rate (FPR) at cut-off = 0.02

Actual RESPONSE	Number of Observations in the sample	Number of Observations classified as responders	ROC Coordinate	ROC Coordinate Value
0	6158	4196	FPR	0.68139
1	2814	2367	TPR	0.84115

For more information about ROC curves, see the textbook by A. A. Afifi and Virginia Clark (2004).[4]

Display 5.15 shows the SAS code that generated Table 5.5.

Display 5.15

```
proc print data=&EM_LIB..mdlcomp_emroc noobs label;
 var cutoff sensitivity oneminusspecificity ;
 where upcase(datarole) = 'VALIDATE' and upcase(model)="NEURAL" and cutoff ne .;
 Title "ROC Table: Validation Data" ;
 label cutoff = "Cutoff" sensitivity = "Sensitivity"
       oneminusspecificity="1-Specificity";
run;
```

5.4.4 How Did the Neural Network Node Pick the Optimum Weights for This Model?

In Section 5.3, I described how the optimum weights are found in a neural network model. I described the two-step procedure of estimating and selecting the weights. In this section, I show the results of these two steps with reference to the neural network model discussed in Sections 5.4.1 and 5.4.2.

The weights such as those shown in Equations 5.1, 5.3, 5.5, 5.7, 5.9, 5.11, and 5.13 are shown in the Results window of the **Neural Network** node. You can see the estimated weights created at each iteration by opening the results window and selecting **View→Model→Weights-History**. Display 5.16 shows a partial view of the Weights-History window.

Display 5.16

Weights - History

ITER	AGE -> HL1	CRED -> HL1	DELINQ -> HL1	MILEAGE -> HL1	NUMTR -> HL1	AGE -> HL2	CRED -> HL2	DELINQ -> HL2	MILEAGE -> HL2	NUMTR -> HL2	AGE -> HL3
30	0.051634	0.013581	-0.04942	0.037994	-0.09108	0.021394	0.015272	-0.01862	0.032955	0.072774	0.03261
31	0.05182	0.014726	-0.05917	0.039808	-0.09145	0.021723	0.013705	-0.02283	0.035434	0.080229	0.033779
32	0.04926	0.014514	-0.06559	0.035954	-0.09289	0.020158	0.013317	-0.01744	0.035331	0.083941	0.035385
33	0.049163	0.015336	-0.07253	0.036421	-0.09141	0.019859	0.012407	-0.01959	0.036852	0.086744	0.036088
34	0.049216	0.015535	-0.07133	0.035055	-0.08937	0.019039	0.012593	-0.01825	0.036613	0.084751	0.036105
35	0.047965	0.016614	-0.08615	0.035396	-0.09089	0.019267	0.010752	-0.02111	0.039862	0.09669	0.037988
36	0.046969	0.015786	-0.08104	0.033397	-0.09163	0.019507	0.011716	-0.01786	0.038681	0.093822	0.036828
37	0.045379	0.017111	-0.09285	0.031359	-0.09064	0.019137	0.010233	-0.01878	0.041725	0.101743	0.03723
38	0.043214	0.017856	-0.10395	0.029164	-0.09141	0.01911	0.009036	-0.01862	0.044226	0.110682	0.037984
39	0.041534	0.018032	-0.10769	0.026898	-0.09245	0.019452	0.008694	-0.01763	0.045423	0.115703	0.037584
40	0.039495	0.019087	-0.11634	0.023946	-0.09319	0.019732	0.007452	-0.01835	0.048368	0.124959	0.036921
41	0.041232	0.019089	-0.10932	0.025544	-0.09307	0.020193	0.007861	-0.02079	0.047792	0.12189	0.035542
42	0.042101	0.019884	-0.11178	0.026301	-0.09267	0.019756	0.007088	-0.02322	0.048779	0.12454	0.0359
43	0.044041	0.020461	-0.10627	0.025526	-0.09514	0.016812	0.006401	-0.02138	0.047231	0.122084	0.036323
44	0.043902	0.022069	-0.1106	0.022244	-0.09944	0.014736	0.004598	-0.02197	0.049496	0.131921	0.035225
45	0.044047	0.021174	-0.1079	0.023537	-0.09872	0.015295	0.005568	-0.02131	0.048206	0.127659	0.035611
46	0.044491	0.021269	-0.10816	0.023826	-0.09965	0.01462	0.005453	-0.02128	0.047965	0.127914	0.036005
47	0.045297	0.021834	-0.10742	0.022637	-0.10376	0.01277	0.004953	-0.0206	0.048183	0.130482	0.03537
48	0.048549	0.022946	-0.10315	0.023189	-0.11333	0.010181	0.004433	-0.02328	0.049119	0.135763	0.032797
49	0.046004	0.022182	-0.10612	0.022121	-0.10898	0.011822	0.004946	-0.02065	0.048699	0.13283	0.033759
50	0.047727	0.022403	-0.10271	0.023855	-0.11699	0.01153	0.005256	-0.02343	0.049256	0.135583	0.031076
51	0.048555	0.022612	-0.09836	0.023541	-0.12846	0.010614	0.005728	-0.02371	0.049642	0.1387	0.027236

The second column in Display 5.16 shows the weight of the variable AGE in hidden unit 1 at each iteration. The seventh column shows the weight of AGE in hidden unit 2 at each iteration. The twelfth column shows the weight of AGE in the third hidden unit. Similarly, you can trace through the weights of other variables. You can save the Weights-History table as a SAS data set.

To see the final weights, open the Results window. Select **View→Model→Weights_Final**. Then, click the **Table** icon. Selected rows of the final weights_table are shown in Display 5.17.

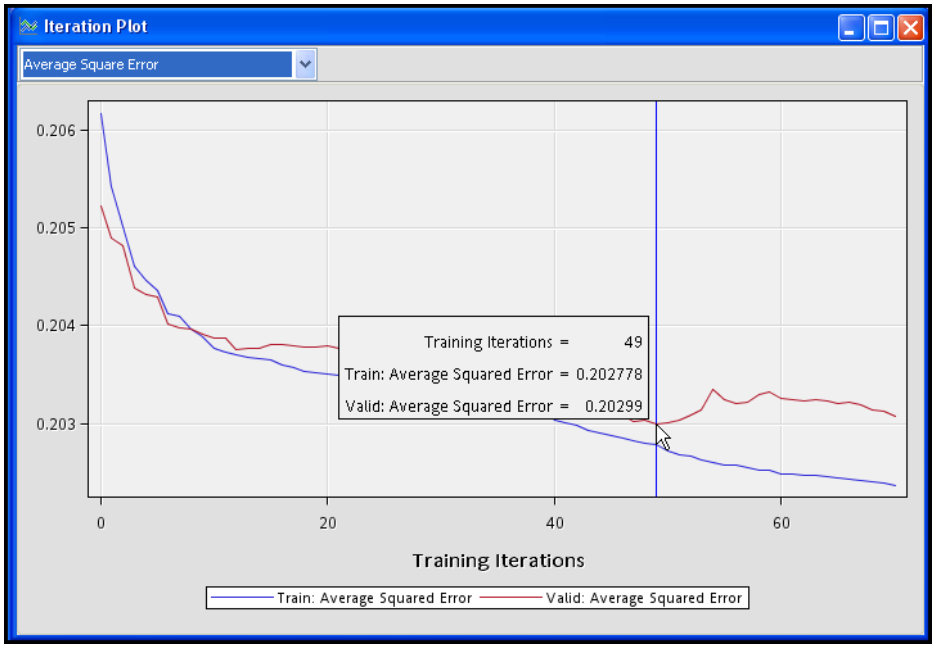
Display 5.17

LABEL	FROM	TO	WEIGHT
AGE -> HL1	AGE	HL1	0.04600
AGE -> HL2	AGE	HL2	0.01182
AGE -> HL3	AGE	HL3	0.03376
BIAS -> HL1	BIAS	HL1	0.21515
BIAS -> HL2	BIAS	HL2	-0.03256
BIAS -> HL3	BIAS	HL3	-0.09328
BIAS -> resp0	BIAS	resp0	1.02894
BIAS -> resp1	BIAS	resp1	-1.02894
CRED -> HL1	CRED	HL1	0.02218
CRED -> HL2	CRED	HL2	0.00495
CRED -> HL3	CRED	HL3	0.03979
HL1 -> resp0	HL1	resp0	1.29172
HL1 -> resp1	HL1	resp1	-1.29172
HL2 -> resp0	HL2	resp0	-1.59773
HL2 -> resp1	HL2	resp1	1.59773
HL3 -> resp0	HL3	resp0	1.56982
HL3 -> resp1	HL3	resp1	-1.56982

Outputs of the hidden units become inputs to the target layer. In the target layer, these inputs are combined using the weights estimated by the **Neural Network** node.

In the model I have developed, the weights generated at the 49th iteration are the optimal weights, because the Average Squared Error computed from the Validation data set reaches its minimum at the 49th iteration. This is shown in Display 5.18.

Display 5.18

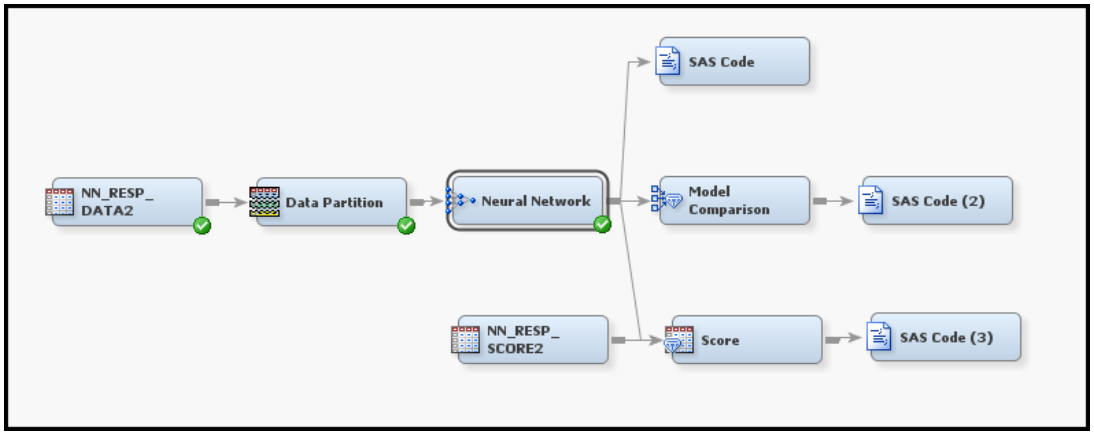


5.4.5 Scoring a Data Set Using the Neural Network Model

You can use the SAS code generated by the **Neural Network** node to score a data set within SAS Enterprise Miner or outside. This example scores a data set inside SAS Enterprise Miner.

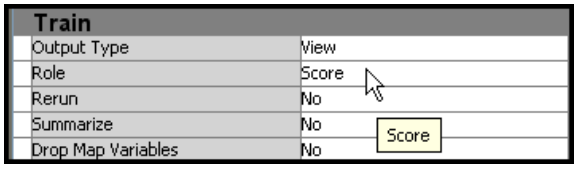
The process flow diagram with a scoring data set is shown in Display 5.19.

Display 5.19



Set the **Role** property of the data set to be scored to Score, as shown in Display 5.20.

Display 5.20



The Score Node applies the SAS code generated by the **Neural Network** node to the Score data set NN_RESP_SCORE2, shown in Display 5.19.

For each record, the probability of response, the probability of non-response, and the expected profit of each record is calculated and appended to the scored data set.

Display 5.21 shows the segment of the score code where the probabilities of response and non-response are calculated. The coefficients of HL1, HL2, and HL3 in Display 5.21 are the weights in the final output layer. These are same as the coefficients shown in Display 5.17.

Display 5.21

```
P_resp1  =    -1.29172482403562 * HL1  +      1.5977312209735 * HL2
       +   -1.56981973510786 * HL3 ;
P_resp0  =     1.29172482403562 * HL1  +    -1.5977312209735 * HL2
       +    1.56981973510786 * HL3 ;
P_resp1  =    -1.0289412689995 + P_resp1 ;
P_resp0  =     1.0289412689995 + P_resp0 ;
DROP _EXP_BAR;
_EXP_BAR=50;
P_resp1  = 1.0 / (1.0 + EXP(MIN( - P_resp1 , _EXP_BAR)));
P_resp0  = 1.0 / (1.0 + EXP(MIN( - P_resp0 , _EXP_BAR)));
```

The code segment given in Display 5.21 calculates the probability of response using the formula $P\_resp1_i = \frac{1}{1+\exp(-\eta_{i21})}$,

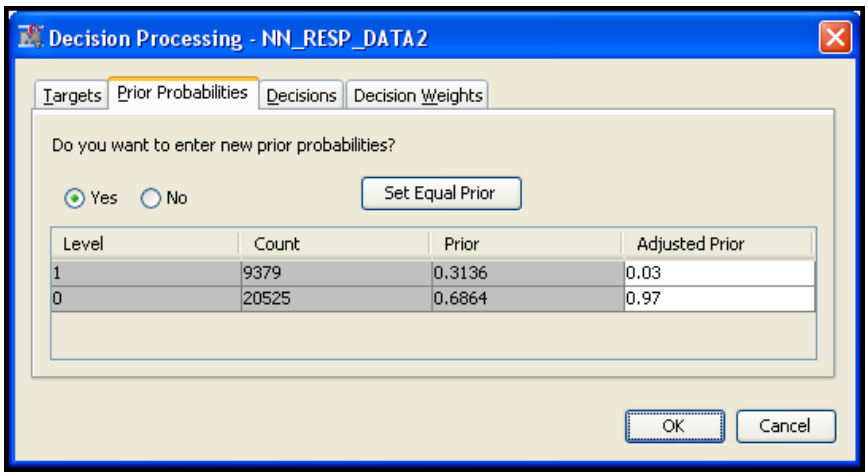
where μ_{i21}= -1.29172481842873 * HL1 + 1.59773122184585 * HL2

\+ -1.56981973539319 * HL3 -1.0289412689995;

This formula is the same as Equation 5.14 in Section 5.2.4. The subscript i is added to emphasize that this is a record-level calculation. In the code shown in Display 5.21, the probability of non-response is calculated as $p\_resp0 = \frac{1}{1+\exp(\eta_{i21})}$.

The probabilities calculated above are modified by the prior probabilities I entered prior to running the **Neural Network** node. These probabilities are shown in Display 5.22.

Display 5.22



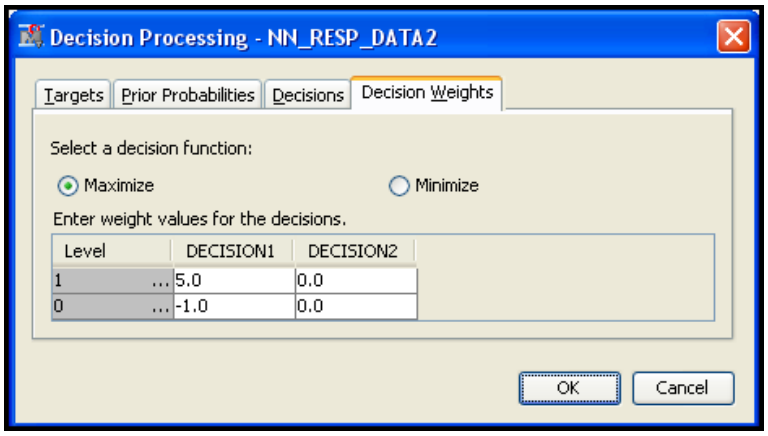
You can enter the prior probabilities when you create the **Data Source**. Prior probabilities are entered because the responders are overrepresented in the modeling sample, which is extracted from a larger sample. In the larger sample, the proportion of responders is only 3%. In the modeling sample, the proportion of responders is 31.36%. Hence, the probabilities should be adjusted before expected profits are computed. The SAS code generated by the **Neural Network** node and passed on to the **Score** node includes statements for making this adjustment. Display 5.23 shows these statements.

Display 5.23

```
*** Update Posterior Probabilities;
P_resp1 = P_resp1 * 0.03 / 0.31368937998772;
P_resp0 = P_resp0 * 0.97 / 0.68631062001227;
drop _sum; _sum = P_resp1 + P_resp0 ;
if _sum > 4.135903E-25 then do;
   P_resp1 = P_resp1 / _sum;
   P_resp0 = P_resp0 / _sum;
end;
```

Display 5.24 shows the profit matrix used in the decision-making process.

Display 5.24



Given the above profit matrix, calculation of expected profit under the alternative decisions of classifying an individual as responder or non-responder proceeds as follows. Using the neural network model, the scoring algorithm first calculates the individual's probability of response and non-response. Suppose the calculated probability of response for an individual is 0.3, and probability of non-response is 0.7. The expected profit if the individual is classified as responder is 0.3x$5 + 0.7x (–$1.0) = $0.8. The expected profit if the individual is

classified as non-responder is 0.3x ($0) + 0.7x ($0) = $0. Hence classifying the individual as responder (Decision1) yields a higher profit than if the individual is classified as non-responder (Decision2). An additional field is added to the record in the scored data set indicating the decision to classify the individual as a responder.

These calculations are shown in the score code segment shown in Display 5.25.

Display 5.25

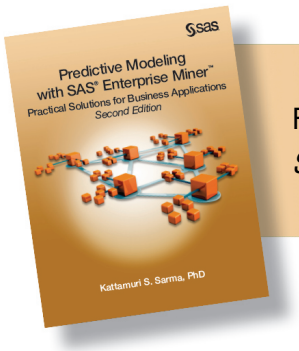
```
*** Decision Processing;
label D_RESP = 'Decision: resp' ;
label EP_RESP = 'Expected Profit: resp' ;

length D_RESP $ 9;

D_RESP = ' ';
EP_RESP = .;

*** Compute Expected Consequences and Choose Decision;
_decnum = 1; drop _decnum;

D_RESP = '1' ;
EP_RESP = P_resp1 * 5 + P_resp0 * -1;
drop _sum;
_sum = P_resp1 * 0 + P_resp0 * 0;
if _sum > EP_RESP + 2.273737E-12 then do;
   EP_RESP = _sum; _decnum = 2;
   D_RESP = '0' ;
end;


*** End Decision Processing ;
```

5.4.6 Score Code

The score code is automatically saved by the **Score** node in the sub-directory \Workspaces\EMWSn\Score within the project directory.

For example, in my computer, the Score code is saved by the **Score** node in the folder C:\TheBook\EM12.1\EMProjects\Chapter5\Workspaces\EMWS3\Score. Alternatively, you can save the score code in some other directory. To do so, run the Score node, and then click Results. Select either the Optimized SAS Code window or the SAS Code window. Click **File→Save As**, and enter the directory and name for saving the score code.

Note: This short excerpt was taken from a chapter in the published book.



From *Predictive Modeling with SAS® Enterprise Miner™, Second Edition.* Full book available for purchase here.

Index

(continued)

(continued)

D

(continued)

E

F

G

H

I

K

L

M

N

(continued)

O

P

(continued)

Q

R

S

T

(continued)

U

V

W

X

About The Author

Kattamuri S. Sarma, PhD, is an economist and statistician with 30 years of experience in American business, including stints with IBM and AT&T. He is the founder and president of Ecostat Research Corp., a consulting firm specializing in predictive modeling and forecasting. Over the years, Dr. Sarma has developed predictive models for the banking, insurance, telecommunication, and technology industries. He has been a SAS user since 1992, and he has extensive experience with multivariate statistical methods, econometrics, decision trees, and data mining with neural networks. The author of numerous professional papers and publications, Dr. Sarma is a SAS Certified Professional and a SAS Alliance Partner. He received his bachelor's degree in mathematics and his master's degree in economic statistics from universities in India. Dr. Sarma received his PhD in economics from the University of Pennsylvania, where he worked under the supervision of Nobel Laureate Lawrence R. Klein.

Learn more about this author by visiting his author page at http://support.sas.com/publishing/authors/sarma.html. There you can download free book excerpts, access example code and data, read the latest reviews, get updates, and more.