Chapter 7: Comparison and Combination of Different Models

We compare the results of three modeling tools

- **Decision Tree**,
- **Neural Network**
- **Regression**

Consider the following models:

- The first model has a binary target and predicts the probability of customer attrition for a fictitious bank.
- The second model has an ordinal target, which is a discrete version of a continuous variable, and predicts risk (as measured by loss frequency) for a fictitious auto insurance company.
- This chapter also provides a method of computing the lift and capture rates of these models using the expected value of the target variable.
- In this chapter the methods of boosting and combining predictive models is illustrated using the **Gradient Boosting** and **Ensemble** nodes.
- The predictive performance of these two methods are compared.

Models for Binary Targets: An Example of Predicting Attrition

- Attrition can be modeled as either a binary or a continuous target. When you model attrition as a binary target, you predict the probability of a customer “attriting” during the next few days or months, given the customer’s general characteristics and change in the pattern of the customer’s transactions.
- With a continuous target, you predict the expected time of attrition, or the residual lifetime, of a customer. For example, if a bank or credit card company wants to identify customers who are likely to terminate their accounts at any point within a predefined interval of time in the future, the company can model attrition as a binary target.
- If, on the other hand, they are interested in predicting the specific time at which the customer is likely to attrit, then the company should model attrition as a continuous target, and use techniques such as survival analysis.

Models for Binary Targets: An Example of Predicting Attrition

- In this chapter, we present an example of a hypothetical bank that wants to develop an early warning system to identify customers who are most likely to close their investment accounts in the next three months.
- To meet this business objective, an attrition model with a binary target is developed.
- The customer record in the modeling data set for developing an attrition model consists of three types of variables (fields):

• Variables indicating the customer's past transactions (such as deposits, withdrawals, purchase of equities, etc.) by month for several months

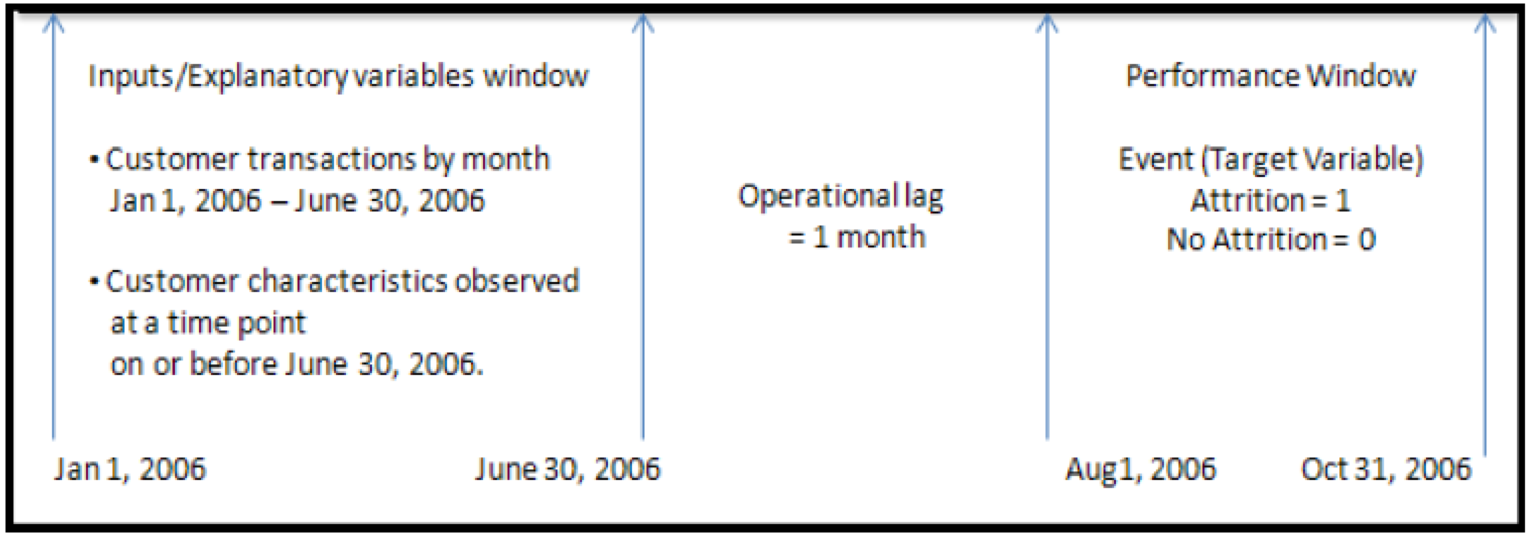
• Customer characteristics (demographic and socio-economic) such as age, income, lifestyle, etc.

• Target variable indicating whether a customer attrited during a pre-specified interval

Assume that the model was developed during December 2006, and that it was used to predict attrition for the period January1, 2007 through March 31, 2007.

Display 7.1 shows a chronological view of a data record in the data set used for developing an attrition model.

Display 7.1


Inputs/Explanatory variables window
• Customer transactions by month Jan 1, 2006 – June 30, 2006
• Customer characteristics observed at a time point on or before June 30, 2006.
Operational lag = 1 month
Performance Window
Event (Target Variable)
Attrition = 1
No Attrition = 0
Jan 1, 2006
June 30, 2006
Aug1, 2006
Oct 31, 2006

Here are some key input design definitions that are labeled in Display 7.1:

Here are some key input design definitions that are labeled in Display 7.1:

• Inputs/explanatory variables window:

This window refers to the period from which the transaction data is collected.

• Operational lag: The model excludes any data for the month of July 2006 to

allow for the operational lag that comes into play when the model is used

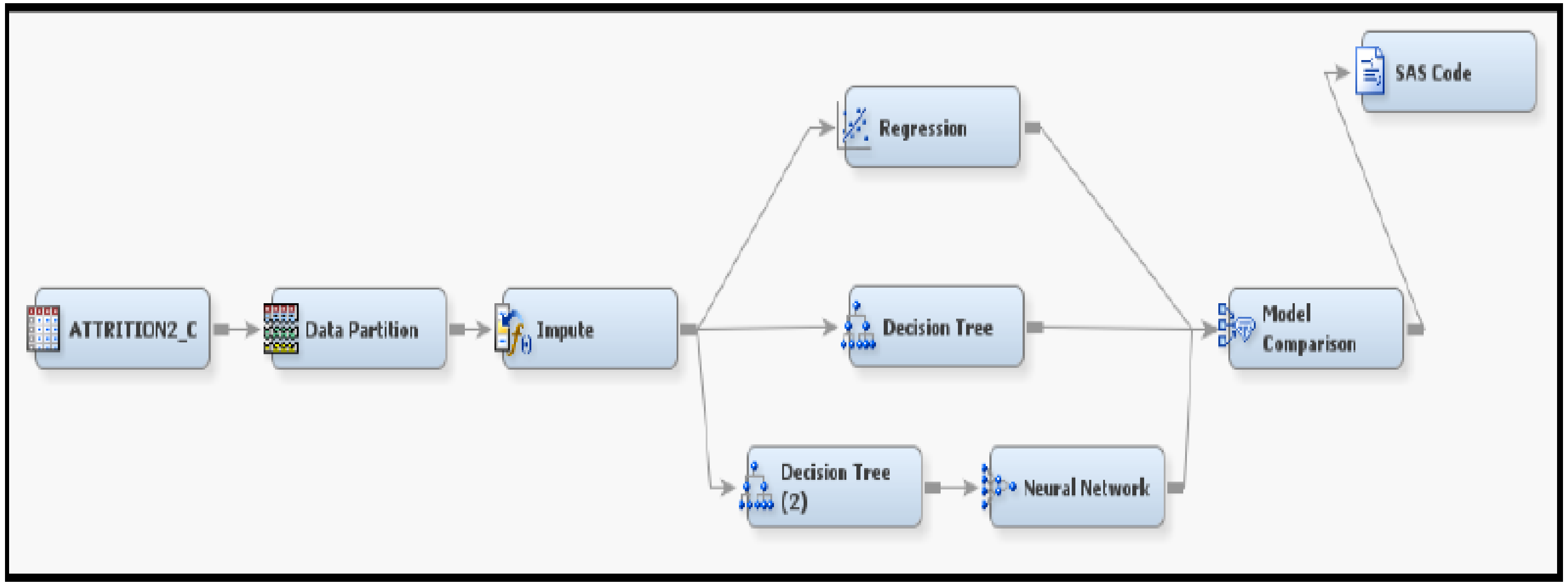
for forecasting in real time.

- This lag may represent the period between the time at which a customer's transaction takes place and the time at which it is recorded in the data base and becomes available to be used in the model.

• performance window: This is the time interval in which the customers in the sample data set are observed for attrition. If a customer attrited during this interval, then the target variable ATTR takes the value 1; if not, it takes the value 0.

In the process flow shown in Display 7.3, the data source is created first from the dataset ATTRITION2_C (see Section 7.2).

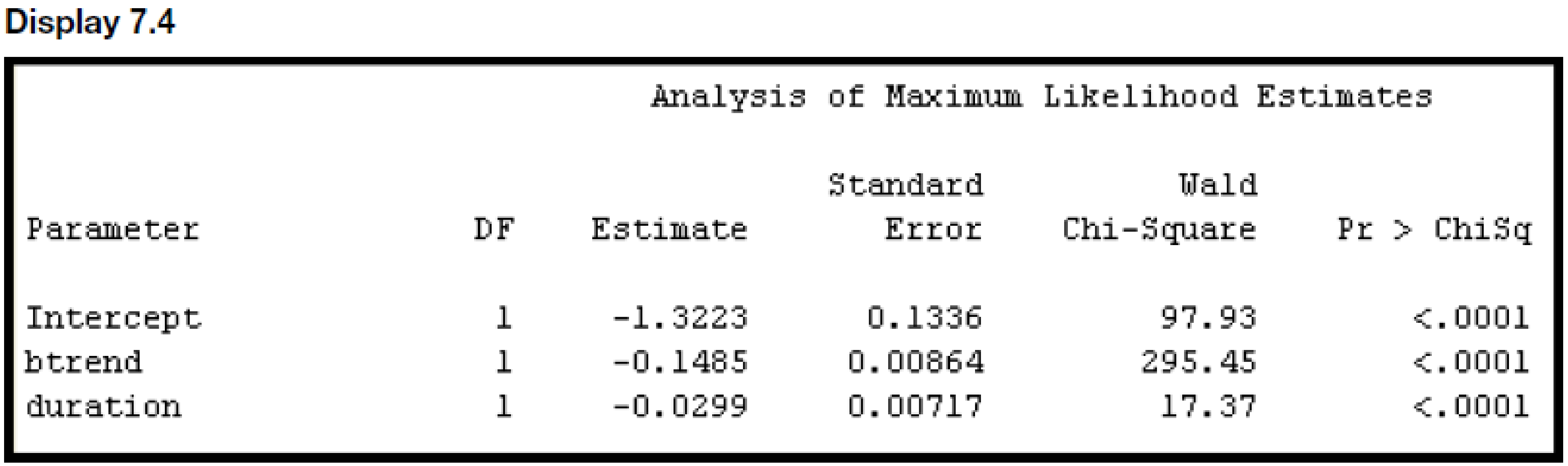
Display 7.3



Logistic Regression for Predicting Attrition

- In the **Regression** node, we set the **Model Selection** property to Stepwise, as this method was found to produce the most parsimonious model.
- For the stepwise selection, we set the **Entry Significance Level** property to 0.05 and the **Stay Significance Level** property to 0.05.
- We set the **Use Selection Defaults** property to No and the Maximum **Number of Steps** property to 100. Before making a final choice of value for the **Selection Criterion** property, we tested the model using different values.
- In the example with results displayed in Table 7.4 , the **Validation Error** criterion produced a model that made the most business sense.
- Display 7.4 shows the model that is estimated by the **Regression** node with these property settings.

Regression node


Display 7.4

```
                       Analysis of Maximum Likelihood Estimates

                                         Standard          Wald
Parameter          DF      Estimate        Error     Chi-Square     Pr > ChiSq

Intercept           1       -1.3223       0.1336        97.93         <.0001
btrend              1       -0.1485      0.00864       295.45         <.0001
duration            1       -0.0299      0.00717        17.37         <.0001
```

Decision Tree Model for Predicting Attrition

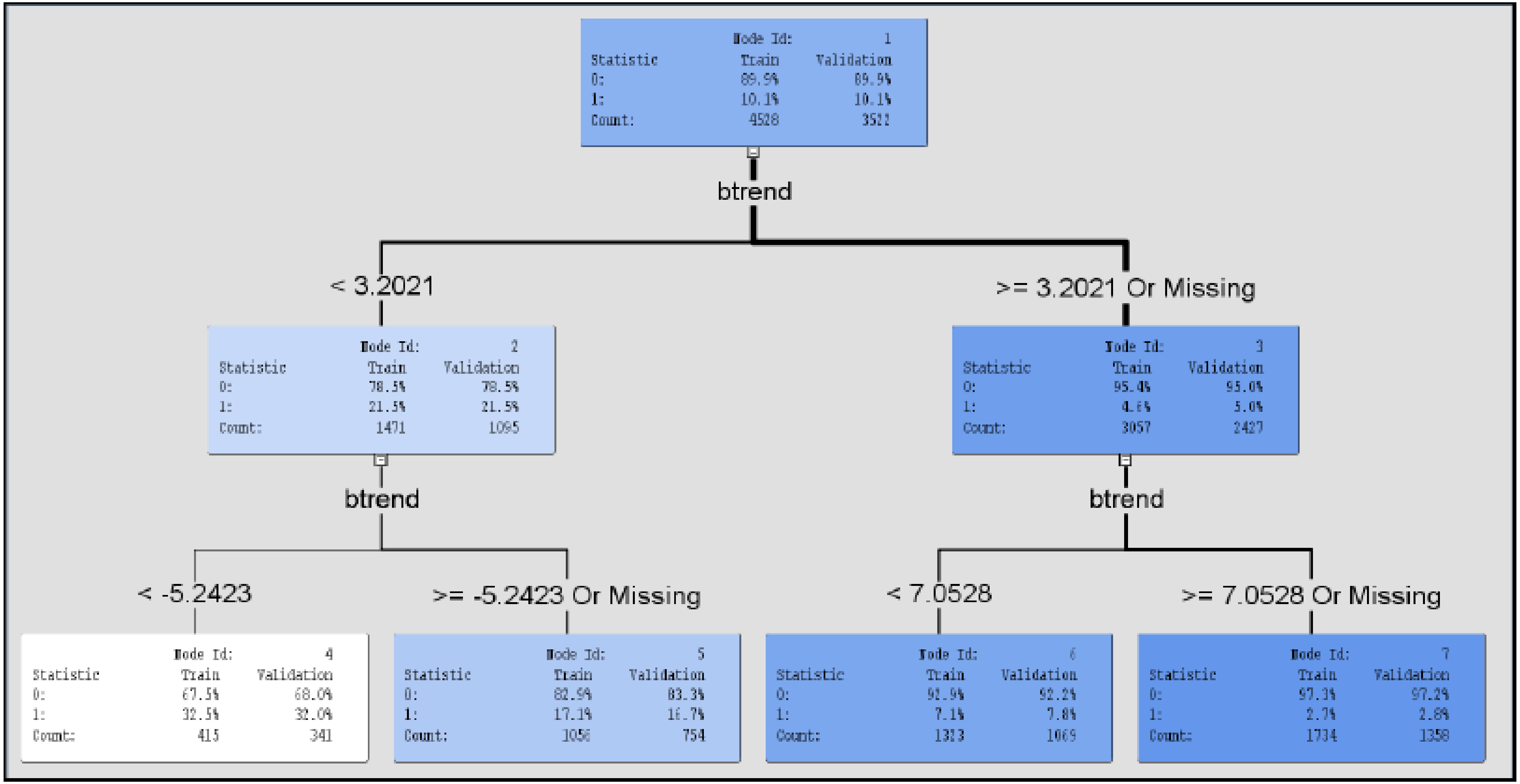
- The middle section of Display 7.3 shows the process flow for the Decision Tree model of attrition.
- We set the **Splitting Rule Criterion** property to ProbChisq, the **Subtree Method** property to Assessment, and the **Assessment Measure** property to Average Square Error.
- Our choice of these property values is somewhat arbitrary, but it resulted in a tree that can serve as an illustration.
- You can set alternative values for these properties and examine the trees produced.
- According to the property values we set, the nodes are split on the basis of *p*-values of the Pearson Chi-Square, and the sub-tree selected is based on the Average Square Error calculated from the Validation data set.

Comparing **Regression** node and **Decision Tree** node

The Display 7.6 in Chapter 7, Section 7.2 shows the tree produced according to the appropriate property values.

- The **Decision Tree** node selected only one variable, btrend, in the tree. The variable btrend is the trend in the customer balances.
- The **Regression** node selected two variables, which were both numeric, while the **Decision Tree** selected one numeric variable only.
- You could either combine the results from both of these models or create a larger set of variables that would include the variables selected by the **Regression** node and the **Decision Tree** node.

Display 7.6



```
                    Node Id:          1
Statistic           Train    Validation
0:                  89.9%         89.9%
1:                  10.1%         10.1%
Count:               4528          3522
```

```
                    Node Id:          2
Statistic           Train    Validation
0:                  78.5%         78.5%
1:                  21.5%         21.5%
Count:               1471          1095
```

```
                    Node Id:          3
Statistic           Train    Validation
0:                  95.4%         95.0%
1:                   4.6%          5.0%
Count:               3057          2427
```

```
                    Node Id:          4
Statistic           Train    Validation
0:                  67.5%         68.0%
1:                  32.5%         32.0%
Count:                415           341
```

```
                    Node Id:          5
Statistic           Train    Validation
0:                  82.9%         83.3%
1:                  17.1%         16.7%
Count:               1056           754
```

```
                    Node Id:          6
Statistic           Train    Validation
0:                  92.9%         92.2%
1:                   7.1%          7.8%
Count:               1323          1069
```

```
                    Node Id:          7
Statistic           Train    Validation
0:                  97.3%         97.2%
1:                   2.7%          2.8%
Count:               1734          1358
```

A Neural Network Model for Predicting Attrition

- The bottom segment of Display 7.3 shows the process flow for the Neural Network model of attrition. In developing a Neural Network model of attrition, we tested the following two approaches:

• Use all the inputs in the data set, set the **Architecture** property to MLP (multi/layer perceptron), the **Model Selection Criterion** property to Average Error, and the **Number of Hidden Units** property to 10.

• Use the same property settings as above, but use only selected inputs. The included variables are: _NODE_, a special variable created by the Decision Tree Node that preceded the Neural Network Node; btrend, a trend variable selected by the Decision Tree Node and duration which was manually included by setting the “use” property to “Yes” in the variables table of the Neural Network Node.

Testing a variety of architectural specifications

- For purposes of illustration, we present below the results from a model with the following property settings:

• **Architecture** property: Multilayer Perceptron

• **Model Selection Criterion** property: Average Error

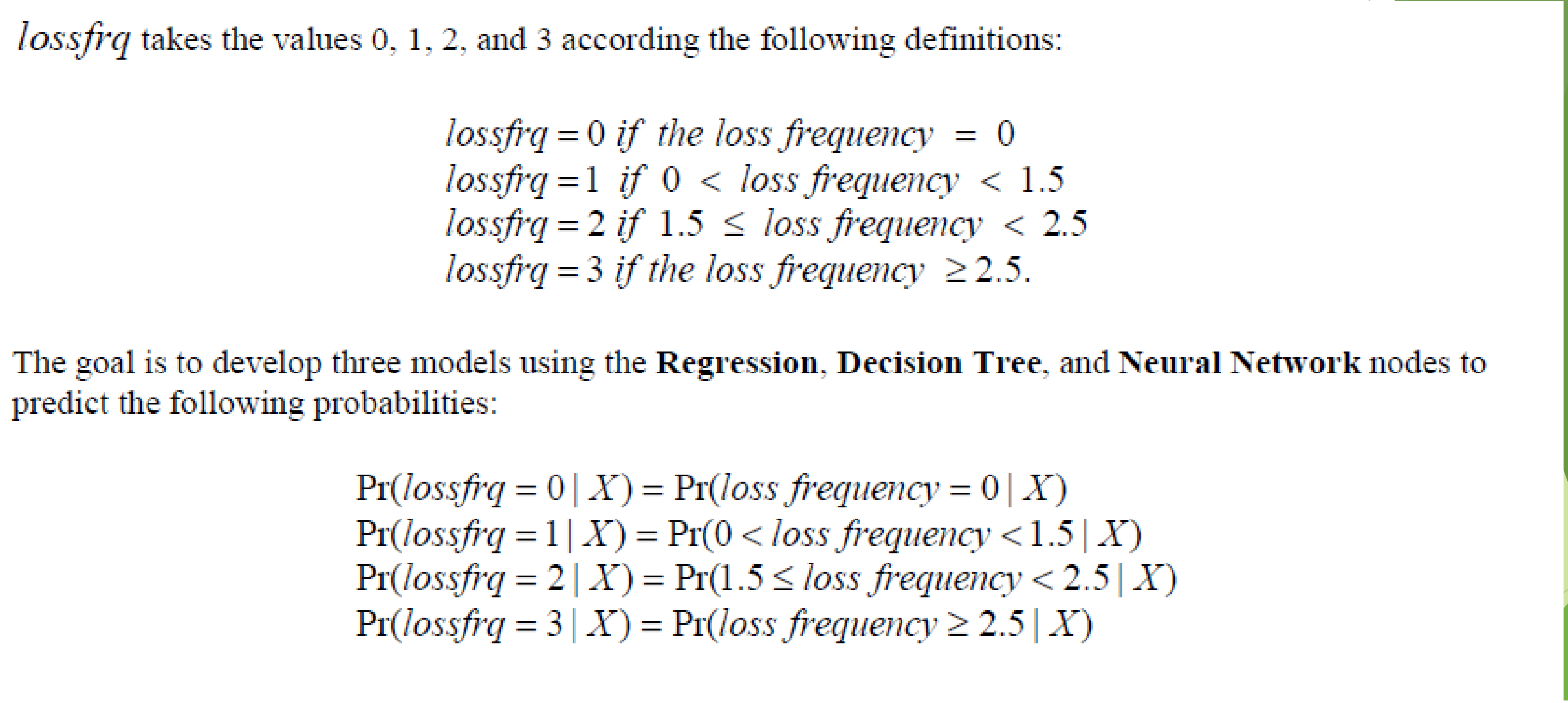
• **Number of Hidden Units** property: 10

- The bottom segment of Display 7.3 shows the process flow for the neural network model of attrition.
- In this process flow, we used the **Decision Tree** node with the **Splitting Rule Criterion** property set to ProbChisq, **Assessment Measure** property set to Average Square Error, **Leaf Variable property** to Yes**, Variable Selection property** to Yes, and **Leaf Role property** to Input to select the inputs for use in the **Neural Network** node.

Models for Ordinal Targets: An Example of Predicting the Risk of Accident Risk

- In Chapter 5, we demonstrated a neural network model to predict the loss frequency for a hypothetical insurance company.
- The loss frequency was a continuous variable, but we used a discrete version of it as the target variable.
- Here we follow the same procedure, and create a discretized variable *lossfrq* from the continuous variable loss frequency.

The discretization is done as follows:

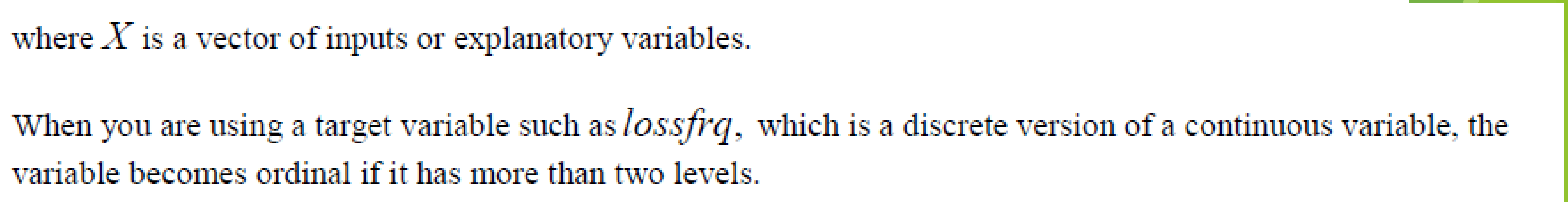


lossfrq takes the values 0, 1, 2, and 3 according the following definitions:

$$
\begin{aligned}
lossfrq &= 0 \text{ if the loss frequency } = 0 \\
lossfrq &= 1 \text{ if } 0 < \text{loss frequency} < 1.5 \\
lossfrq &= 2 \text{ if } 1.5 \le \text{loss frequency} < 2.5 \\
lossfrq &= 3 \text{ if the loss frequency } \ge 2.5.
\end{aligned}
$$

The goal is to develop three models using the **Regression**, **Decision Tree**, and **Neural Network** nodes to predict the following probabilities:

$$
\begin{aligned}
\Pr(lossfrq = 0 \mid X) &= \Pr(\text{loss frequency} = 0 \mid X) \\
\Pr(lossfrq = 1 \mid X) &= \Pr(0 < \text{loss frequency} < 1.5 \mid X) \\
\Pr(lossfrq = 2 \mid X) &= \Pr(1.5 \le \text{loss frequency} < 2.5 \mid X) \\
\Pr(lossfrq = 3 \mid X) &= \Pr(\text{loss frequency} \ge 2.5 \mid X)
\end{aligned}
$$

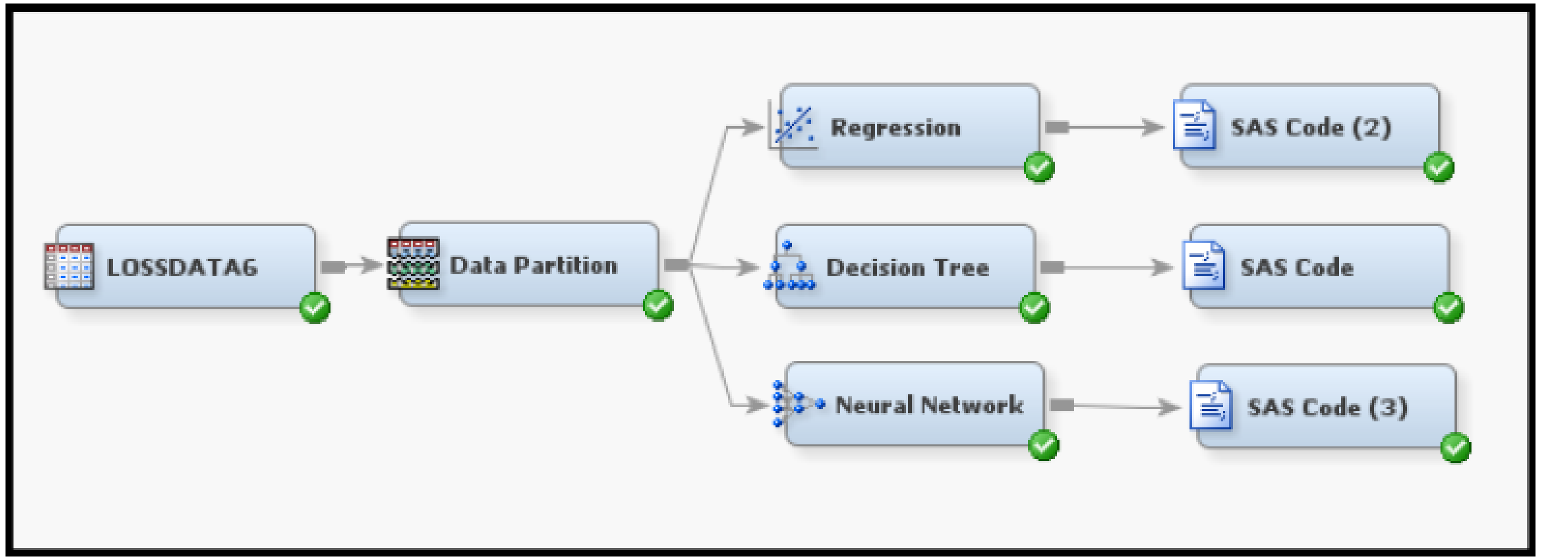


where X is a vector of inputs or explanatory variables.

When you are using a target variable such as *lossfrq*, which is a discrete version of a continuous variable, the variable becomes ordinal if it has more than two levels.

Comparison of Models

Display 7.10



Lift Charts and Capture Rates for Models with Ordinal Targets

Method Used by SAS Enterprise Miner

- When the target is ordinal, SAS Enterprise Miner creates lift charts based on the probability of the highest level of the target variable.

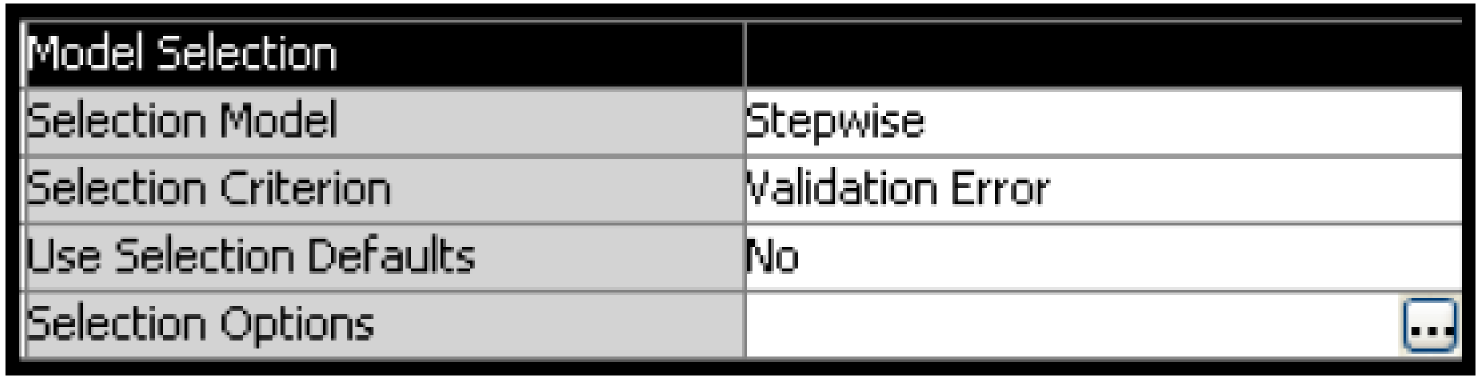
An Alternative Approach Using Expected Lossfrq

- You sometimes need to calculate lift and capture rate based on the expected value of the target variable.

Logistic Regression with Proportional Odds for Predicting Risk in Auto Insurance

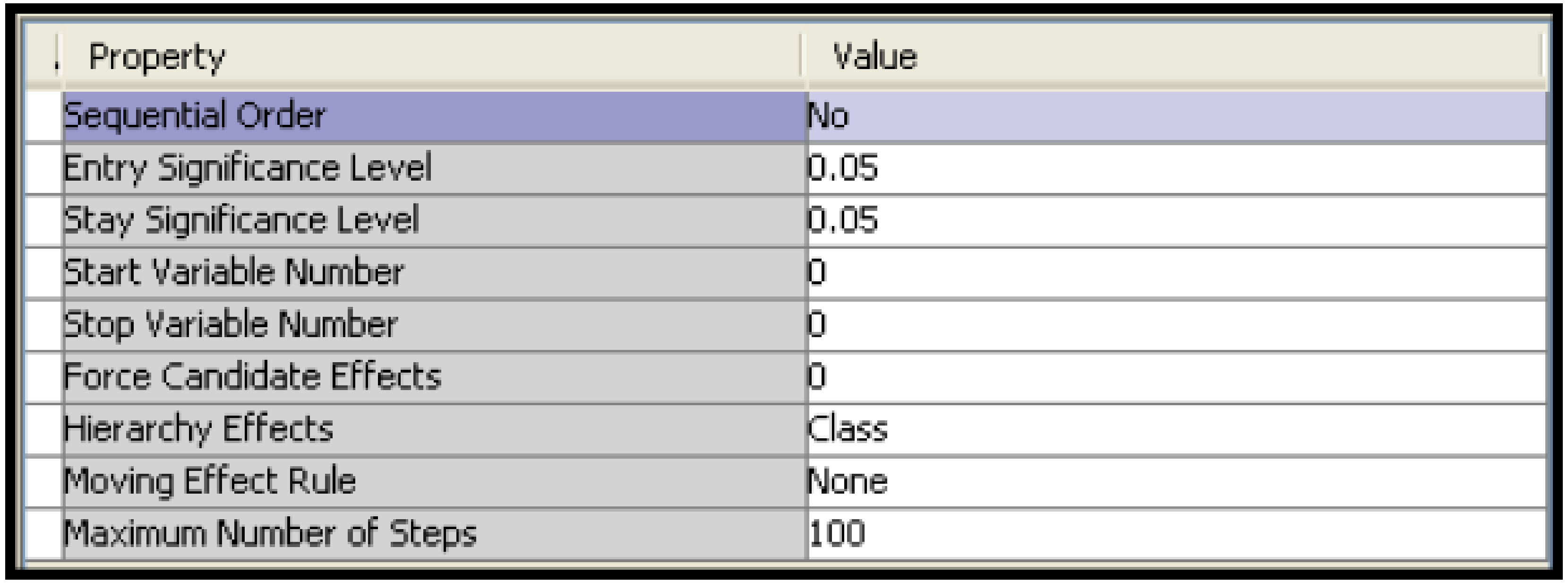
- Given that the measurement level of the target variable is set to Ordinal, a Proportional Odds model is estimated by the **Regression** node using the property settings shown in Section 7.3, Displays 7.11 and 7.12.

Display 7.11



Model Selection	
Selection Model	Stepwise
Selection Criterion	Validation Error
Use Selection Defaults	No
Selection Options	...

Display 7.12

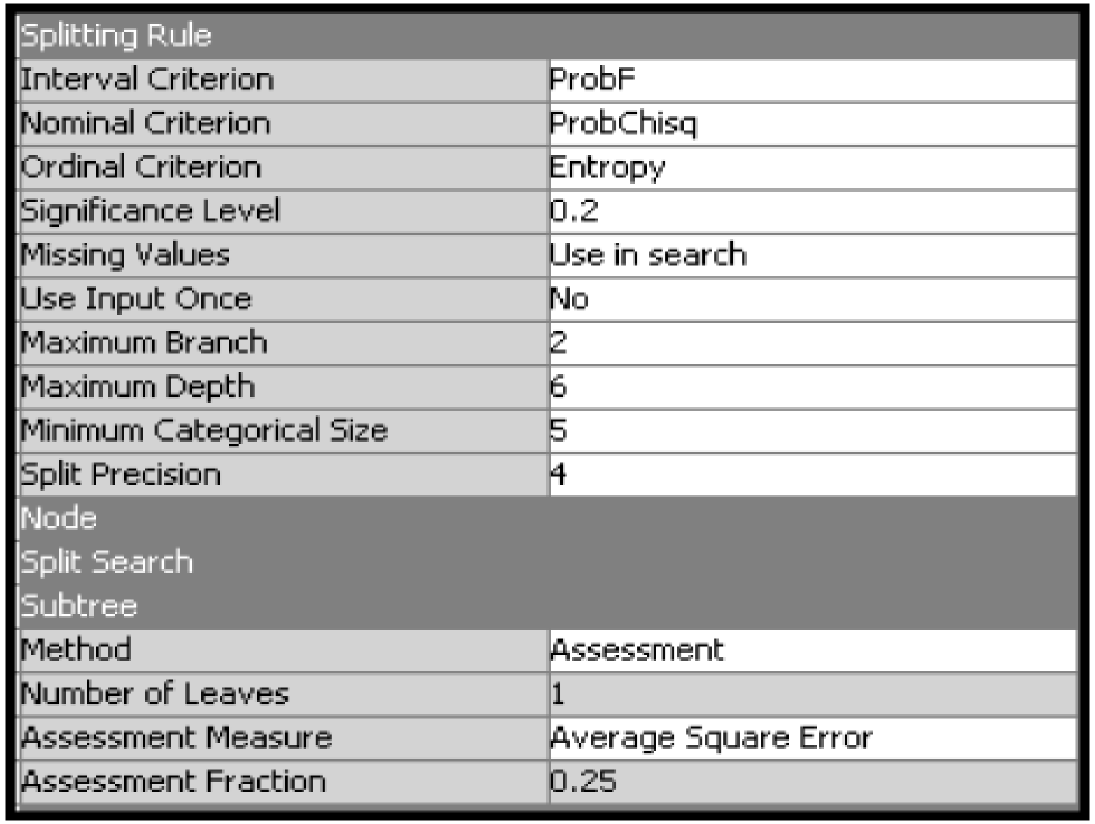


Property	Value
Sequential Order	No
Entry Significance Level	0.05
Stay Significance Level	0.05
Start Variable Number	0
Stop Variable Number	0
Force Candidate Effects	0
Hierarchy Effects	Class
Moving Effect Rule	None
Maximum Number of Steps	100

Decision Tree Model for Predicting Risk in Auto Insurance

Display 7.15 shows the settings of the properties of the **Decision Tree** node.

Display 7.15

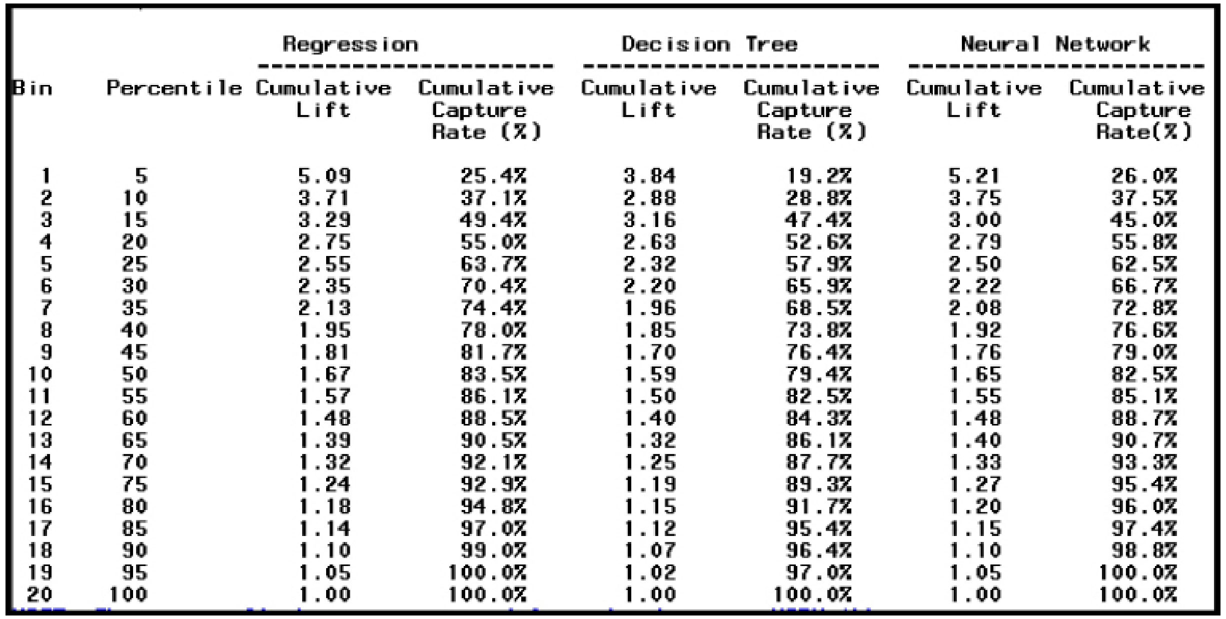


Splitting Rule	
Interval Criterion	ProbF
Nominal Criterion	ProbChisq
Ordinal Criterion	Entropy
Significance Level	0.2
Missing Values	Use in search
Use Input Once	No
Maximum Branch	2
Maximum Depth	6
Minimum Categorical Size	5
Split Precision	4
Node	
Split Search	
Subtree	
Method	Assessment
Number of Leaves	1
Assessment Measure	Average Square Error
Assessment Fraction	0.25

Comparison of All Three Accident Risk Models

Table 7.8 shows the lift and capture rates calculated for the Test data set ranked by $E(lossfrq)$, as outlined in Section 7.3.

Table 7.8



		Regression		Decision Tree		Neural Network	
Bin	Percentile	Cumulative Lift	Cumulative Capture Rate (%)	Cumulative Lift	Cumulative Capture Rate (%)	Cumulative Lift	Cumulative Capture Rate(%)
1	5	5.09	25.4%	3.84	19.2%	5.21	26.0%
2	10	3.71	37.1%	2.88	28.8%	3.75	37.5%
3	15	3.29	49.4%	3.16	47.4%	3.00	45.0%
4	20	2.75	55.0%	2.63	52.6%	2.79	55.8%
5	25	2.55	63.7%	2.32	57.9%	2.50	62.5%
6	30	2.35	70.4%	2.20	65.9%	2.22	66.7%
7	35	2.13	74.4%	1.96	68.5%	2.08	72.8%
8	40	1.95	78.0%	1.85	73.8%	1.92	76.6%
9	45	1.81	81.7%	1.70	76.4%	1.76	79.0%
10	50	1.67	83.5%	1.59	79.4%	1.65	82.5%
11	55	1.57	86.1%	1.50	82.5%	1.55	85.1%
12	60	1.48	88.5%	1.40	84.3%	1.48	88.7%
13	65	1.39	90.5%	1.32	86.1%	1.40	90.7%
14	70	1.32	92.1%	1.25	87.7%	1.33	93.3%
15	75	1.24	92.9%	1.19	89.3%	1.27	95.4%
16	80	1.18	94.8%	1.15	91.7%	1.20	96.0%
17	85	1.14	97.0%	1.12	95.4%	1.15	97.4%
18	90	1.10	99.0%	1.07	96.4%	1.10	98.8%
19	95	1.05	100.0%	1.02	97.0%	1.05	100.0%
20	100	1.00	100.0%	1.00	100.0%	1.00	100.0%

Gradient Boosting

The Gradient Boosting algorithm starts with an initial model represented by a mapping function, which can be written as:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{N} \Psi(y_i, \gamma) \qquad (7.1)$$

where $F_0(x)$ is the initial model, N is the number of records in the Training data, Ψ is a Loss Function, γ is a parameter that can be called the "node score" and y_i is the actual value of the target variable for the i^{th} record. Equation 7.1 implies that the value of the parameter γ is chosen such that the Loss given by $\sum_{i=1}^{N} \Psi(y_i, \gamma)$ is minimized. Suppose we use the Loss Function $(y_i - \gamma)^2$ for the i^{th} record. Then the aggregate loss is calculated as:

$$\sum_{i=1}^{N} \Psi(y_i, \gamma) = \sum_{i=1}^{N} (y_i - \gamma)^2 \qquad (7.2)$$

Minimization of the right-hand side of the equation 7.2 leads to the estimate:

$$\gamma = \frac{\sum_{i=1}^{N} y_i}{N} = \bar{y}$$, which is the mean of the target variable in the Training data set.

Using the initial model specified by Equation 7.1, the predicted value of the target variable for each record is the overall mean. Denoting the predicted value of the target for the i^{th} record from the initial model by $\hat{y}_{i0}$:

$$\hat{y}_{i0} = \bar{y} \qquad (7.3)$$

Hence we can write the Loss Functions in terms of the predicted value of the target as:

$$\left(y_i - \hat{y}_{i0}\right)^2 \qquad (7.4)$$

If we use the Loss Function $|y_i - \gamma|$, then the value of the parameter γ that minimizes the Loss is the median denoted by $\tilde{y}$. The Loss Function $\left(y_i - \gamma\right)^2$ is called the *Least Squares Loss* and the Loss Function $|y_i - \gamma|$ is called the *Least Absolute Deviation Loss*.

Stochastic Gradient Boosting

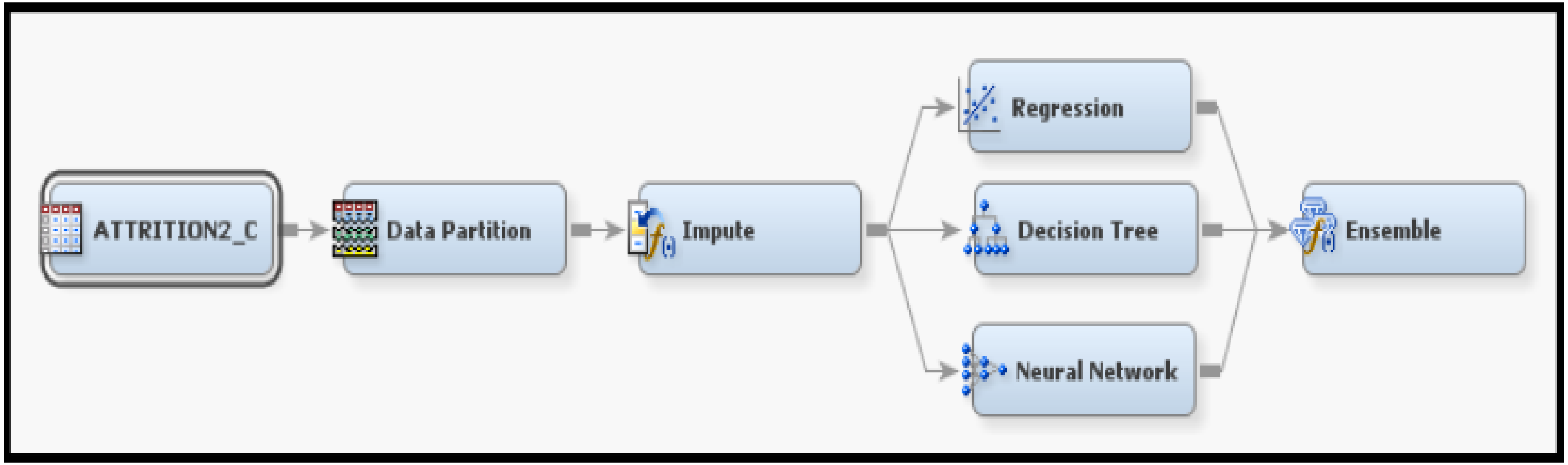
- In Stochastic Gradient Boosting, at each iteration, the tree is developed using a random sample (without replacement) from the Training data set.

- In the sampling at each step, weights are assigned to the observations according the prediction errors in the previous step, assigning higher weights to observations with larger errors.

The Ensemble Node

We can use the **Ensemble** node to combine the three models' **Logistic Regression**, **Decision Tree**, and **Neural Network** nodes, as shown in Display 7.25.

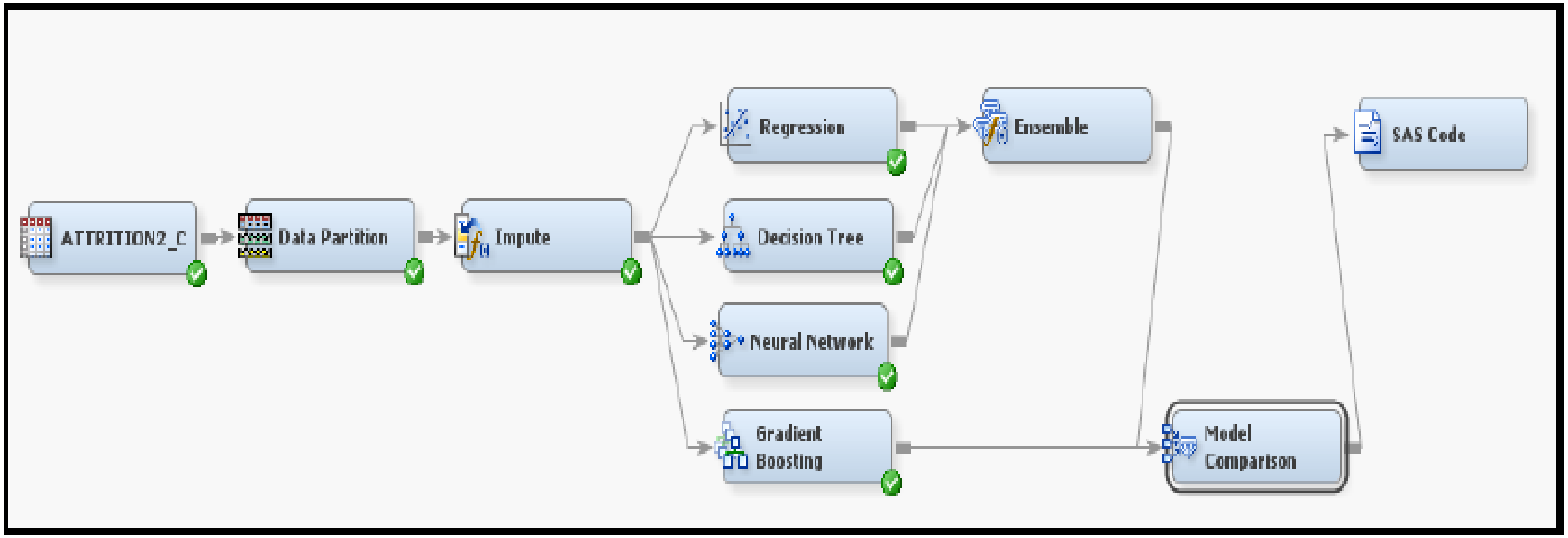
Display 7.25



Comparing the Gradient Boosting and Ensemble Methods of Combining Models

- In this section, we compare the combined model generated by the **Ensemble** node with the one generated by the **Gradient Boosting** node.
- Display 7.26 shows the process flow diagram for demonstrating the **Ensemble** node and also for comparing the models generated by the **Ensemble** and **Gradient Boosting** nodes.

Display 7.26


ATTRITION2_C
Data Partition
Impute
Regression
Decision Tree
Neural Network
Gradient Boosting
Ensemble
Model Comparison
SAS Code