Paper 074-30

Combining Decision Trees with Regression in Predictive Modeling with SAS® Enterprise Miner™

Kattamuri S. Sarma

Abstract

The purpose of this paper is to illustrate how the Decision Tree node can be used to optimally bin the inputs for use in a logistic regression. Binning can be viewed as a complex non-linear transformation of the inputs. By utilizing this technique, we may be able to capture complex non-linear relationships between an input variable and the target variable. Univariate trees are created by using the Tree node with one input at a time, resulting in an optimal binning of the input. The SAS code generated is used to create transformations of the original inputs. The binned variables are then used in a logistic regression. The paper shows how this can be done using the GUI of the Enterprise Miner and also using the underlying procedures in a batch mode.

Introduction

This paper shows how to use decision trees for variable selection and transformation in developing logistic regression models. The target variable is binary and indicates response to a mail campaign for auto insurance products. It takes on only two values: 1 for response and 0 for non-response. The data used is hypothetical, but it has many features of the real world. For example, it has a large number of inputs with different measurement scales. There are continuous (interval scaled) inputs, nominal scaled categorical inputs with many levels, ordinal and binary inputs.

In order to eliminate irrelevant variables, an initial variable selection is made. The initial variable selection results in a reduction of the size of the input set to a manageable number. The tree algorithm is then applied to the selected inputs one at a time, and for each input the resulting SAS code is saved. The SAS code shows the ranges into which the input is split and also gives the mean of the target for each interval of the input. In the terminology of a tree, these input ranges are also called leaf nodes or simply leaves of the tree. Inputs that do not produce meaningful trees are eliminated.

Initial screening with Variable Selection node

The initial screening of the inputs is done by the Variable Selection node of the SAS Enterprise Miner using the R-square measure as the selection criterion. Interval scaled variables are tested in their original form and also in the binned form[1]. The R-square is calculated between the target and each interval scaled input in its original form, as well as between the target and each input in its transformed form, as a categorical (binned) variable. The binned variables in the Variable Selection node are called AOV16 variables, and named as "A16XXX", where XXX is the name of the input in its original

[1] A detailed discussion of the variable selection node is included in a forthcoming BBU book by the author.

form. For example, if a continuous input such as income is binned, the binned variable is called A16INCOME. The Variable Selection node treats the AOV16 variables as class variables and uses one-way ANOVA to calculate R-square with the target. Correlation between the target and the original variable measures the linear relationship between the input and the target, while the correlation between the binned variable (A16XXX) and the target measures the non-linear relationship between the original input variable and the target. The bins in the AOV16 variables are of equal size. That is, the input is divided into intervals of equal length. They are not optimal in the sense that they are not based on the relationship with the target. In the Variable Selection node, they are used only to uncover non-linear relationships. Therefore, we do not use the binned variables (A16XXXX) in the creation of tree in the next step. Instead we use the original variables corresponding to the selected AOV16 variables and other selected variables in the Tree node, to create more optimal bins.

Unlike the bins created by the Variable Selection node, the intervals created by the tree algorithm are, in general, not of equal length. All variables that have an R-square measure with the target above a specified minimum value are selected in the initial variable selection process. The minimum required R-square is set at a low level so as to minimize the chance of rejecting inputs that may have individually weak relationship with the target, but may have a stronger relationship with the target when other variables are introduced.

Univariate Trees: Applying the Tree node to individual inputs using Graphics User interface.

All variables that are selected in the Variable Selection node are passed to the decision node in their original forms. That is, whether a variable is selected in its original form or in its binned form, we send only the original form to the Tree node for optimal binning. One can eliminate the Variable Selection node altogether and go directly to the Decision Tree node for optimal binning. In our example, since the original data set contains a large number of variables, the Variable Selection node is applied first, and only selected variables are passed to the Tree node. It should be noted that the Transformation Node in Enterprise Miner could also be used to create optimal bins. However, we found that the decision tree is more convenient for this purpose. Therefore, we decided to use the decision tree for optimal binning.

One tree is created for each input variable. The Decision Tree node creates intervals of the inputs so as to optimize the relation with the target. As a result, the bins created by the decision tree differ from those created by the Variable Selection node.

Diagram I shows the process flow for applying the tree algorithm one variable at a time. The data set is split into Training, Validation and Test groups. Fifty percent of the original data is used for Training, 25% for Validation and 25% for Testing. The tree is run repeatedly by assigning the model role of “input” to one variable at a time.

Diagrams 2 and 3 show the basic tab and advanced tab of the Tree node. In the basic tab, one selects the splitting criterion, and in the advanced tab one selects the model assessment criterion.

Diagram 1: The Process Flow

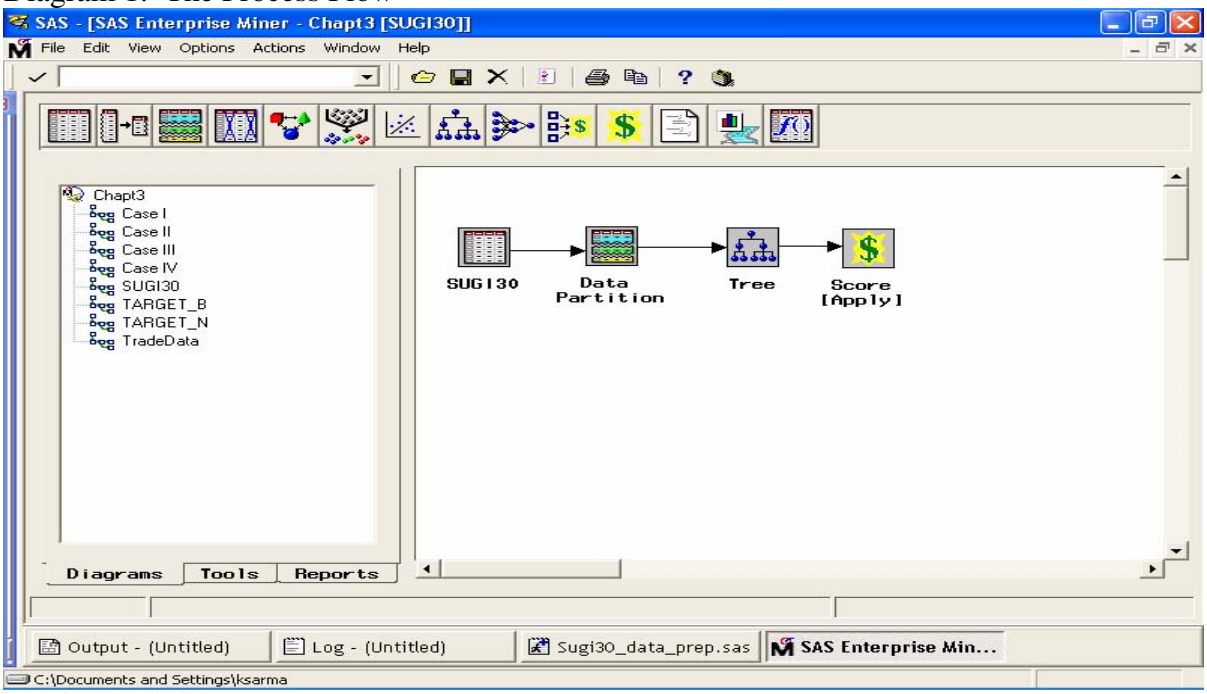
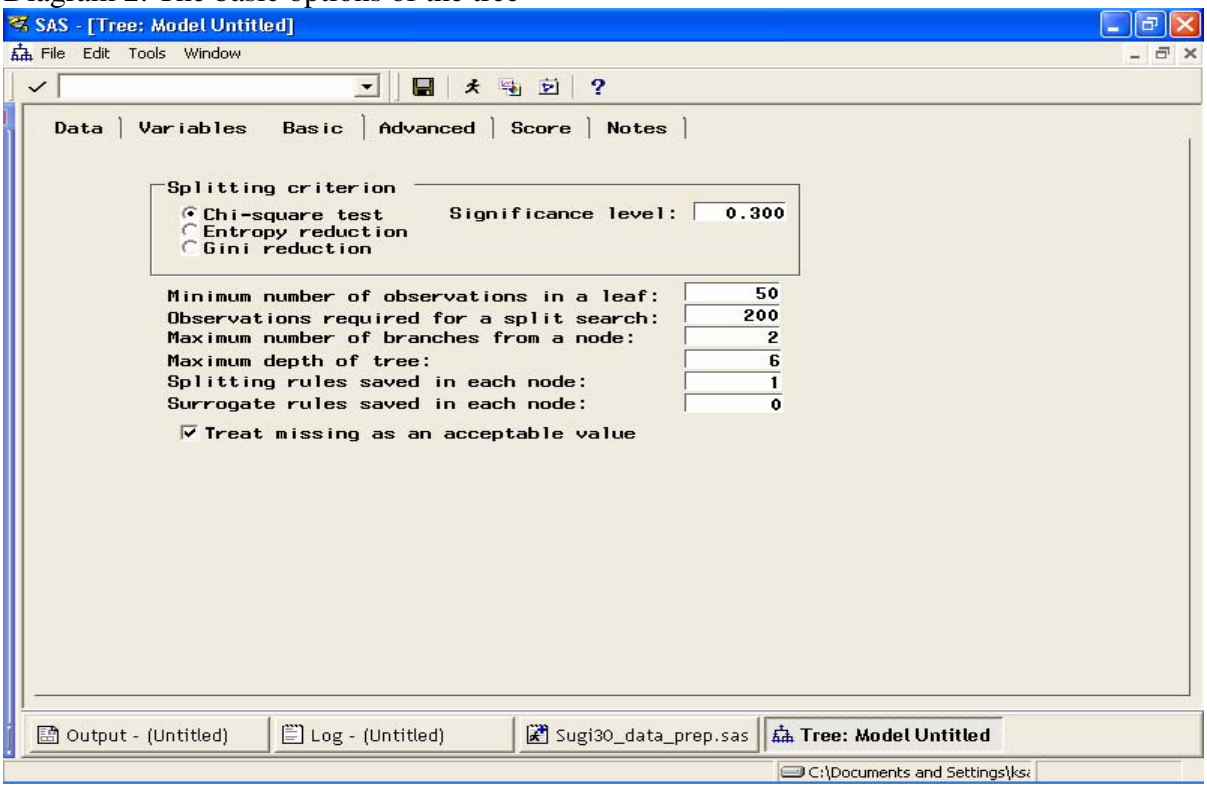


Diagram 2: The basic options of the tree



Splitting criterion:

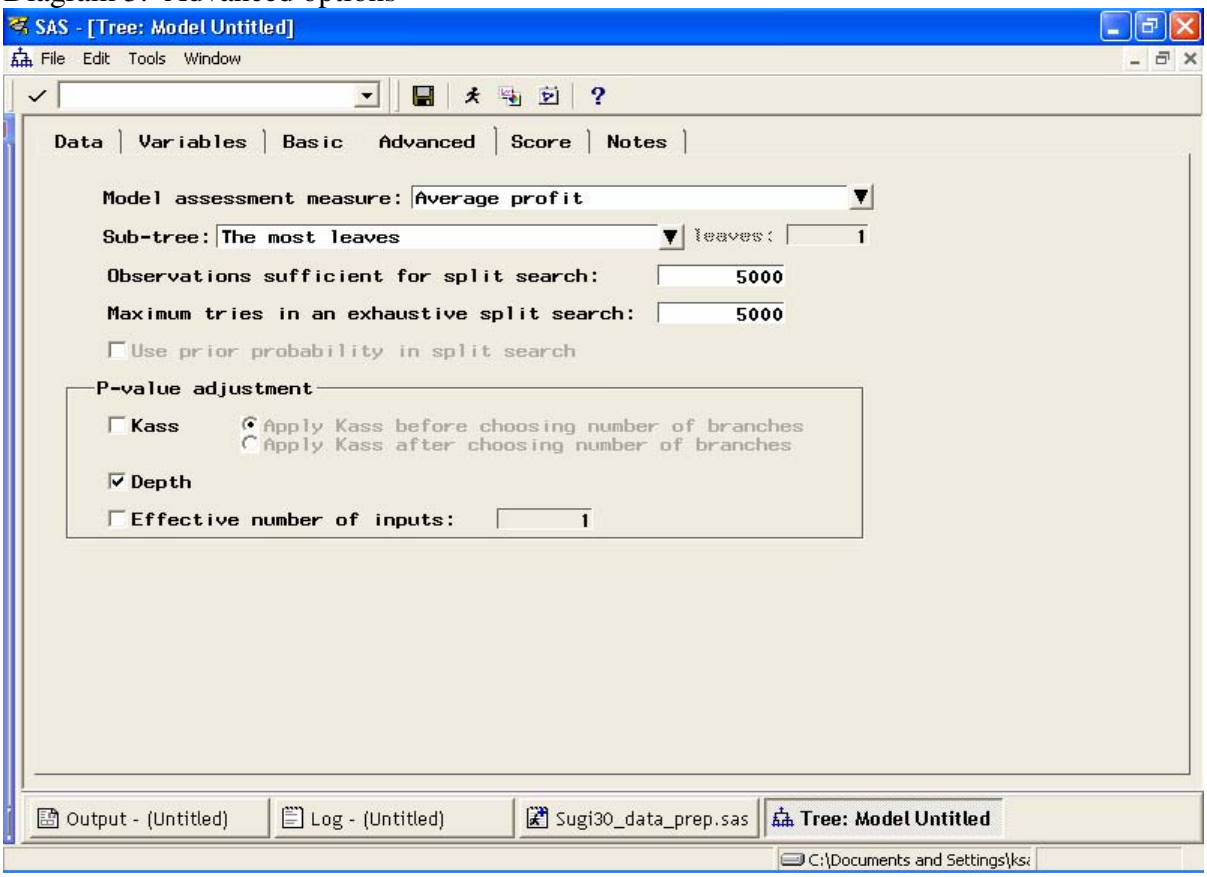
In the basic tab of the Tree node we specified that the criterion is chi-square. What this means is that the algorithm makes a chi-square test at each potential split on the input variable, and selects the best split. This is done by first creating a contingency table at each potential split on the input variable. If we use age as the input, the potential split values are all ages between minimum value (19 years, in this sample) to the maximum (89 years). There are 68 split values starting at 20, and ending in 88. At each split value, a contingency table is created. For example, at split value of 20, all cases which have an age of 20 or less are put into one group and others in another group. Then the number of responders and non-responders in each group are counted. The contingency table looks something like the following:

Table 1: Contingency table at a potential split

	$Age \le 20$	$Age > 20$
Target		
1	n_{11}	n_{12}
0	n_{01}	n_{02}
	n_1	n_2

There are n_1 cases with $Age \le 20$. Of these, n_{11} are responders and n_{01} are non-responders. There are n_2 cases with $Age > 20$. Of these, n_{12} are responders and n_{02} are non-responders. A chi-square test is performed to test whether there is any significant difference between response rates between persons 20 years or younger and those older than 20 years. The corresponding p-value and the *logworth* are calculated. The *logworth*, which is equal to $-\log_{10}(p-value)$, is calculated at all other splits: age 21, 22, up to and including age 88. The split at which the *logworth* is the maximum is selected. The data set is divided into two parts at that split value. Let us say that the best split is 45. At this point the data set is divided into two parts, one with those 45 or younger and those older than 45. Each part is again divided further in the same way using the same input variable.

Diagram 3: Advanced options

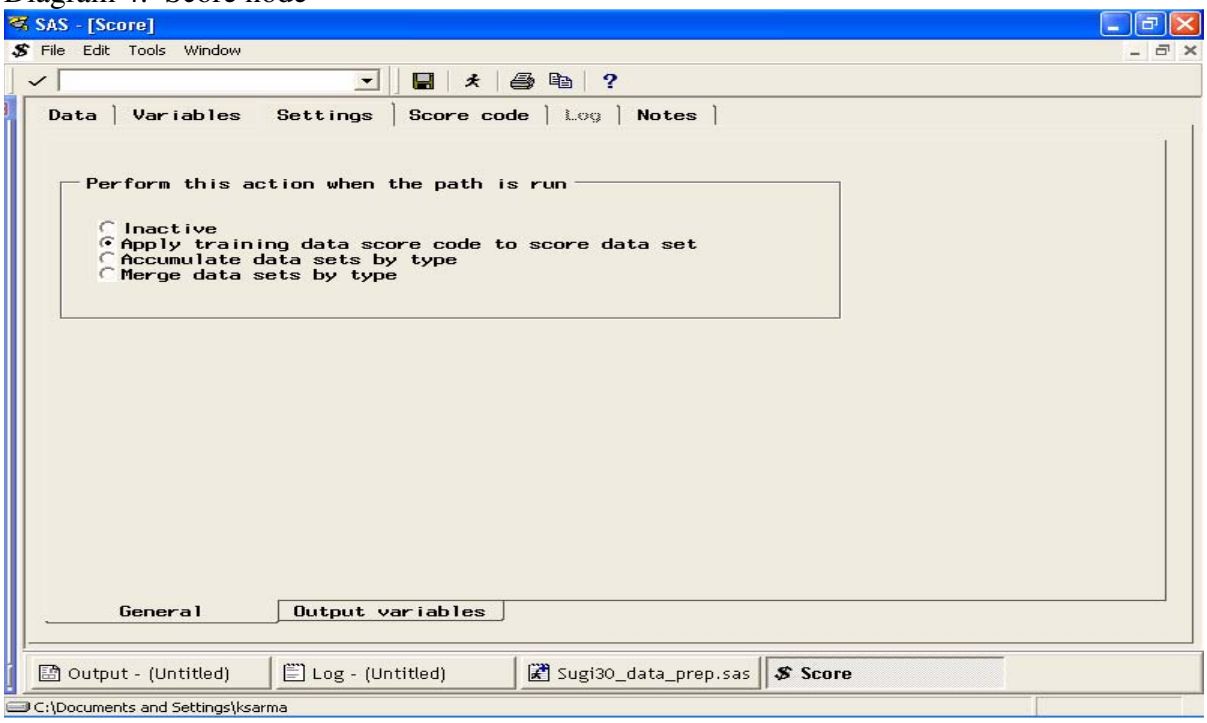


In general, when a tree is constructed, the training data set is used to construct a maximal tree. The maximal tree is the tree with the maximum number of leaves possible under the constraints of stopping rules specified. The stopping rules are (1) minimum number of observations in a leaf, and (2) observations required for a split search and (3) depth adjustment. Within the maximal tree there are a number of sub-trees. In the usual tree development, the validation data set is used to evaluate the profitability of each sub-tree within the maximum sub-tree. The sub-tree where the profit is maximized is selected. The optimal sub-tree is the one at which the marginal profit (extra profit per additional leaf) is either zero or negative.

In the advanced tab (Diagram 3) we specify that the sub-tree be selected on the basis of maximum number of leaves, so as to prevent the pruning of the tree. These options can be changed easily.

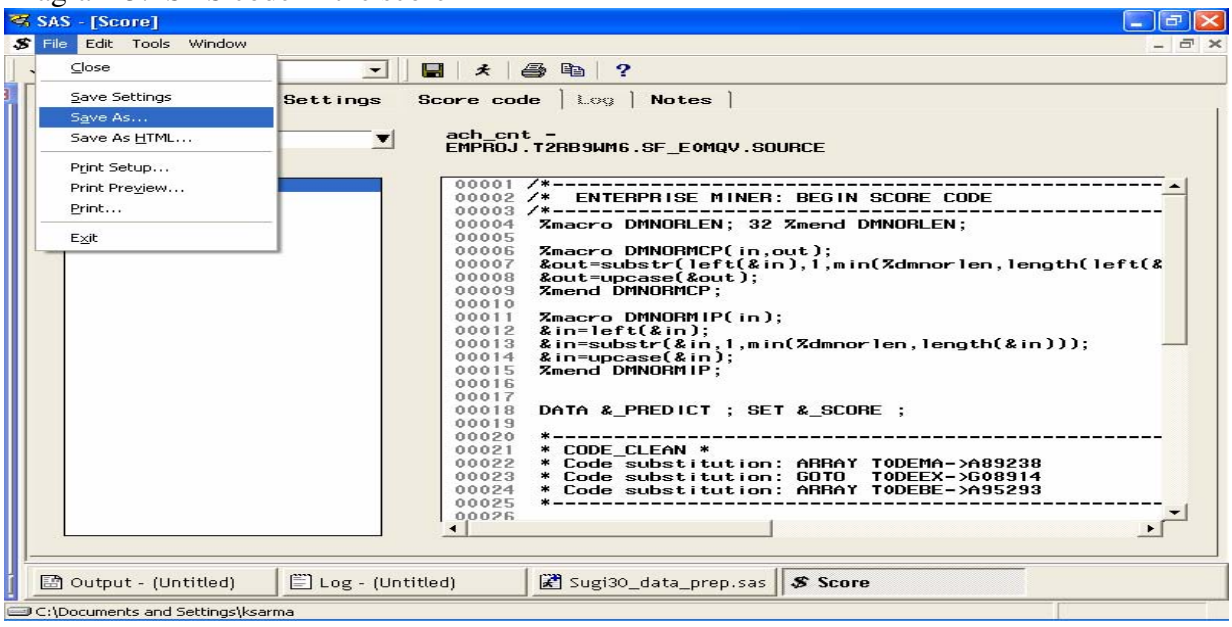
After we run the Tree node, we end up with a tree that is, essentially, a representation of the partitioning of the data set using input ranges. In addition, the Tree node generates SAS code showing the rules of partitioning. The SAS code gives the ranges of the input that define the leaf nodes or partitions, and the mean of the target in each partition.

Diagram 4: Score node



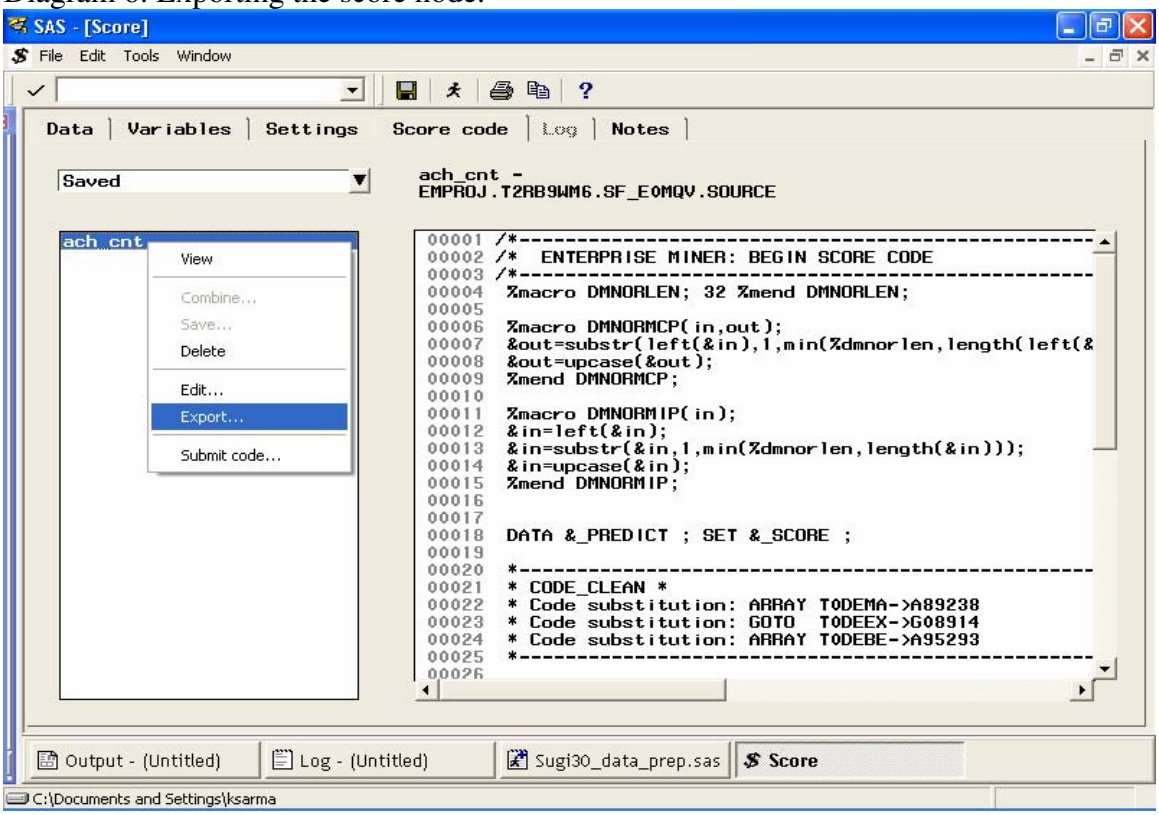
In order to capture the SAS code, we attach a Score node to the Tree node as shown in the process flow (Diagram 1). The set up of the Score node is shown in Diagram 4. The code is saved as shown in Diagram 5.

Diagram 5: SAS code in the score



The saved code is then exported as shown in Diagram 6. The exported SAS code is edited and used to define new variables outside of the Enterprise Miner.

Diagram 6: Exporting the score node.



The SAS code for the variable "income" is saved as "income.sas", the code for age is saved as "age.sas", the code for credit is stored as credit.sas, etc.

Using the SAS code to make transformations:

The following code segment shows how the transformations are performed using the SAS code. The SAS code had to be modified slightly before using it for transformations.

```
data   moddat ;
    set t3.sugi30 ;
 %include 'c:\sugi30\income.sas' ;
  d_income= p_resp1;
%include 'c:\sugi30\age.sas' ;
  d_age = p_resp1  ;
%include 'c:\sugi30\credit.sas' ;
  d_credit = p_resp1  ;
 ……

run ;
```

The logistic regression can then be used with the newly created variables in the following way:

```
proc logistic data=moddat  descending ;
 class d_income d_age d_credit / selection= forward  ;
run  ;
```

Univariate Trees: Applying the Tree node to individual inputs in batch mode

Applying the Tree node individually to each input, one by one, using the GUI of the SAS enterprise Miner as outlined above can be very tedious. An alternative way of creating the transformations is to use PROC SPLIT as shown in the pseudo code below. However, since PROC SPLIT and other procedures of the Enterprise Miner are not supported by SAS, one uses these procedures at one's own risk.

To create a tree in a batch mode, we begin by creating a profit matrix. The pseudo code below shows the population priors, and the profit matrix[2] that may be used in assessing the tree.

```
DATA T3.DECDATA (type=profit);
INPUT RESPC $ PRIOR RESP NORESP;
datelines;
1 0.10 10 -0.2
0 0.90 -1  0
;
run;
```

[2] A thorough discussion of population priors, the profit matrix, and how it is used in assessing the sub-trees is included in the forthcoming BBU book by the author.

Finally, a macro is created for building the tree for each input separately and saving the code. Below is pseudo code for creating the macro and an example of the macro call for a nominal variable called “segment”.

```
%MACRO SPLITV(DATA=,TARGET=,TTYPE=,INP=,INPTYPE=);
PROC DMDB BATCH DATA=&DATA OUT=DMTRAIN DMDBCAT=CATTRAIN ;
  CLASS  &TARGET.(DESC)
         &INP ;
  TARGET &TARGET ;
RUN ;

PROC SPLIT DATA=DMTRAIN DMDBCAT=CATTRAIN
     CRITERION=PROBCHISQ
     SPLITSIZE=2
       PADJUST=NONE
     MAXBRANCH=2
       SUBTREE=LARGEST
       ASSESS=PROFIT
       VALIDATA=VALID
     OUTTREE=TREE ;
DECISION DECDATA=T3.DECDATA DECVAR=RESP NORESP  ;
CODE FILE="C:\SUGI30\&INP..SAS" DUMMY  ;
 INPUT &INP /LEVEL=&INPTYPE ;
 TARGET &TARGET /LEVEL=&TTYPE ;
RUN ;
%MEND SPLITV;

%SPLITV(DATA=TRAIN,TARGET=RESPC,TTYPE=NOMINAL,INP=SEGMENT,INP
TYPE=NOMINAL);
```

Conclusions

Although the example shown is for selecting variables after they are transformed individually, the methodology can be extended to capture interactions by using several inputs together. More work needs to be done, and it is hoped that this paper will be instrumental in generating more thought in this direction. The Variable Selection node can be used to bin continuous inputs into equal sized classes. The Transform Variables node can also automatically bin interval variables into buckets using an algorithm based on a decision tree. However, during the course of our work, we found that using the Decision Tree Node for optimal binning to be the more flexible and versatile approach.

Special note

All SAS code shown in this paper is for illustrative purposes only. The code shown is not usable "as is." It is intended to give a broad outline of how to combine the tree with logistic regressions. It does not cover all the possibilities. For example, in order to take account interactions between inputs, one should build the tree using not one input at a time, but several inputs together.

Contact Information:

Kattamuri S. Sarma
White Plains, NY

Phone: 914-428-8733
Fax : 914-428-4551

Email: kssarma@worldnet.att.net
Web: http://www.ecostat-research.com