Paper AA-03-2015

Fitting and Evaluating Logistic Regression Models

Bruce Lund, Magnify Analytic Solutions, a Division of Marketing Associates, LLC
Detroit MI, Wilmington DE, and Charlotte NC

ABSTRACT

Logistic regression models are commonly used in direct marketing and consumer finance applications. In this context the paper discusses topics in the fitting and evaluation of logistic regression models. The first topic is the screening and transformation of predictor variables. The second topic is a comparison of two methods for finding multiple candidate models. The first method is the familiar "best subsets" approach. Then best subsets is compared to an outline of a new method based on combining models produced by backward and forward selection plus the models considered by backward and forward. This second method uses HPLOGISTIC with selection of predictors by SBC (Schwarz Bayes). The third topic is a discussion of model evaluation statistics to measure predictive accuracy and goodness-of-fit in support of the choice of a final model. The paper uses Base SAS® and SAS/STAT.

INTRODUCTION

This paper focuses on fitting of binary logistic regression models for direct marketing, customer relationship management, credit scoring, or other applications where (i) samples for modeling are large, (ii) there are many predictors, and (iii) the emphasis is on using the models to score future observations.

The fitting of logistic regression models includes the following three steps:

1. Screening and preparation of predictors to prepare for inclusion in logistic regression models
2. Finding a multiple number of promising candidate models for comparison and evaluation
3. Evaluating these candidate models and final model selection

Only some of the topics that fall into these three steps above are discussed in the paper.

Step 1: Screening and preparation of predictors to prepare for inclusion in logistic regression models

There are three topic areas:

- Screening nominal and discrete predictors to find and remove those with weak predictive power
- Binning nominal and discrete predictors and weight-of-evidence coding
- Screening and transforming of continuous predictors

Important but omitted topics include: (i) Resolving multicollinearity, (ii) Interaction detection, (iii) and more.

Step 2: Finding a multiple number of promising candidate models for comparison and evaluation

There are two topic areas:

- Running Best Subsets (SELECTION=SCORE) of PROC LOGISTIC
- Running PROC HPLOGISTIC with METHOD=BACKWARD and with METHOD=FORWARD and then combining the results of all models, either selected or considered

Step 3: Evaluation of these candidate models and final model selection

Brief comments are given on three topic areas:

- Measuring goodness-of-fit
- Measuring predictive accuracy
- Ranking of models in term of parsimonious fit

ASSUMPTIONS: It is assumed there is an abundant population from which to sample and that large sub-samples have been selected for the **training, validation, and test data sets**. It is also assumed that a large set of potential predictor variables X1 - XK have been given values as of an observation date. This observation date could vary by record. Finally, it is assumed that the **target Y** has been carefully defined and assigned a value of 0 or 1 where this value is determined by whether an event occurred within a specified time period following the observation date.

PREDICTOR CATEGORIES: (1a) NOMINAL / ORDINAL, (1b) DISCRETE, (2) CONTINUOUS

Predictors can be categorized into one of: (1a), (1b) or (2).

(1a) Nominal / Ordinal: The values of nominal and ordinal predictors cannot be placed on the number line so that meaningful arithmetical operations can be performed on these values. Although an ordinal predictor has an ordering of its values, logistic regression modeling cannot utilize this ordering.

(1b) Discrete: The values of a discrete predictor have meaningful numeric values but the number of distinct values are few. "Few" is a subjective term. An upper threshold might be set at 20. Often a discrete variable is a "count" such as "number of children in household".

For the purpose of screening and transforming of predictors, (1a) and (1b) will be combined.

(2) Continuous: Values of a continuous predictor have meaningful numeric values and many distinct values. Predictors that measure distance, time, or money are often best categorized as continuous.

The distinction between (1a & 1b) and (2) is important because the classification of a predictor into one of these categories determines the screening and transformation process that is applied to the predictor.

The acronym for Nominal / Ordinal / Discrete will be "**NOD"**. Distinct values of a NOD predictor will often be called "levels".

NOD (Nominal / Ordinal / Discrete) Predictors and Logistic Regression

A NOD predictor C with L levels is entered into the logistic model with a CLASS statement or, equivalently, through the use of L-1 DUMMY variables C_dum_k for k = 1 to L-1. [1] [2]

PROC LOGISTIC; CLASS C; MODEL Y = C <and other predictors>;
or
PROC LOGISTIC; MODEL Y = C_dum_k where k = 1 to L-1 <and other predictors>;

SCREENING AND TRANSFORMING NOD PREDICTORS

INFORMATION VALUE (IV)

An often-used measure of predictive power of predictor C is Information Value (IV). It measures predictive power without regard to an ordering of a predictor. TABLE 1 gives a cross-tab of the values of a predictor C and target Y. It is notationally convenient to use G_k to refer to the counts of Y = 1 and B_k to refer to the counts of Y = 0 when C = C_k. Let G = $\sum_k G_k$. Then g_k is defined as $g_k = G_k / G$. Similarly for b_k. IV is the sum of the IV Terms. The range of IV is the non-negative numbers.

TABLE 1 – INFORMATION VALUE EXAMPLE

C	Y = 0 "B_k"	Y = 1 "G_k"	Col % Y=0 "b_k"	Col % Y=1 "g_k"	Log(g_k/b_k)	g_k - b_k	**IV Terms** (g_k - b_k) * Log(g_k/b_k)
C1	2	1	0.250	0.125	-0.69315	-0.125	0.08664
C2	1	1	0.125	0.125	0.00000	0	0.00000
C3	5	6	0.625	0.750	0.18232	0.125	0.02279
SUM	8	8				**IV =**	**0.10943**

IV can be computed for any predictor provided none of the g_k or b_k is zero. As a formula, IV is given by:

$$\mathbf{IV = \sum_{k=1}^{L} (g_k - b_k) * \log(g_k / b_k)}$$

where L $\geq$ 2 and where g_k and b_k > 0 for all k = 1,…,L

---

[1] "CLASS C;" creates a coefficient in the model for each of L-1 of the L levels. The modeler's choice of "reference level coding" determines how the L^{th} level enters into the calculation of the model scores. See SAS/STAT(R) 14.1 User's Guide (2015), LOGISTIC procedure, CLASS statement.

[2] In PROC LOGISTIC: CLASS and DUMMY are equivalent when there is not a variable selection using stepwise, forward, or backward. When a selection method is used, then all levels of a CLASS variable are either selected or not selected but DUMMY variable coding allows individual levels (dummies) to be selected or not selected.

Note: If two levels of C are collapsed (binned together), the new value of IV is less than or equal to the old value. The new IV value is equal to the old IV value if and only if the ratios g_r / b_r and g_s / b_s are equal where levels C_r and C_s were collapsed together.[3]

Predictive Power of IV:

Guidelines for interpretation of values of the IV of a predictor in an applied setting are given below. These guidelines come from Siddiqi (2006, p.81). For logistic modeling applications it is unusual to see IV $\geq$ 0.5.

TABLE 2 – PRACTICAL GUIDE TO INTERPRETING IV

IV Range	Interpretation
IV < 0.02	"Not Predictive"
IV in [0.02 to 0.1)	"Weak"
IV in [0.1 to 0.3)	"Medium"
IV $\geq$ 0.3	"Strong"

A generalization is that the greater the IV of a predictor C, the greater is its "predictive power". This relationship of IV to "predictive power" might be operationalized as the relationship between IV of C to the Log Likelihood (LL) of the model PROC LOGISTIC; CLASS C; MODEL Y = C;. Given sample size N and number of levels L of C and also limiting IV to a relevant range of IV $\leq$ 1, there is a strong linear relationship between IV and LL. For example, if N = 10,000 and L = 5, then a simple linear regression of IV to LL is a good model. Based on a simulation (with 500 samples) the R-square is 61%.[4]

SCREENING NOD PREDICTORS AND PRELIMINARY BINNING

The fact that IV is non-increasing when levels of C are collapsed might suggest that more levels of C are better than fewer. But this conclusion is counter to the need to achieve model parsimony. Each level of C requires a degree of freedom.

A compromise is needed between the desire for predictive power and the need for parsimony. This compromise will be realized by a process for "optimal" binning which is discussed in a later section. But first, in the following sections, the screening of NOD predictors and their preliminary binning (fine classing) is discussed.

Screening Nominal / Ordinal / Discrete Predictors and %IV_X_C_STAT

If a modeler has a large number of NOD predictors, perhaps over 20, then an automated tool for measuring predictive power of these predictors is helpful. Such a tool is the macro %IV_X_C_STAT.

For numeric predictors %IV_X_C_STAT computes the c-statistic. For any predictor the macro also computes IV [5] and the "x-statistic". The term "x-statistic" refers to the c-statistic produced by the model: PROC LOGISTIC; CLASS C; MODEL Y = C;.[6] See Raimi and Lund (2012, 2013) for more discussion.[7]

X-Statistic and C-Statistic for Discrete (or Ordered) Predictors

Definition: Assume predictor X is ordered. That is: $X_k < X_{k+1}$ for k = 1 to L-1 where L is the number of levels of predictor X. Let $P_k = g_k / (b_k + g_k)$ where g_k and b_k are defined as in TABLE 1. The P_k are "monotonic increasing" if $P_k \leq P_{k+1}$ for k = 1 to L-1. A corresponding definition is given for "monotonic decreasing". The P_k are "monotonic" if either "monotonic increasing" or "monotonic decreasing".[8]

There is an interesting and useful relationship between x-statistic and c-statistic.

---

[3] See Lund and Brotherton (2013, p. 17) for a proof.
[4] Simulation code is available from the author.
[5] IV has a missing value if any $g_k = 0$ or $b_k = 0$.
[6] The computation of x-statistic can be done in a data step without use of PROC LOGISTIC.
[7] In Raimi and Lund (2012, 2013) a macro %XC_STAT is presented which computes IV, x-statistic, and c-statistic for numeric variables. This macro incorporates an optional PROC RANK to reduce the number of levels of the variables and provides extensive input data checking. Character variables are excluded.
[8] This condition is equivalent to requiring g_k / b_k to be monotonic.

Relationship between c-statistic and x-statistic:

i. x-statistic $\geq$ max(c-statistic, 1 - c-statistic)
ii. x-statistic = max(c-statistic, 1 - c-statistic) if and only if P_K is monotonic

This relationship shows that the x-statistic can be seen as an extension of the c-statistic for measuring the strength of non-monotonic relationships of X to Y. The "1 - c-statistic" quantity is included in the "max" to produce the version of the c-statistic that is greater than or equal to 0.5.[9]

EXAMPLE 1: %IV_X_C_STAT

This example includes a data set "example1". In data set "example1" the variable Y is the 0/1 target variable, C1-C2 are character variables and X1-X3 are numeric variables.

```
DATA example1;
INPUT C1 $ C2 $ X1 X2 X3 Y @@;
DATALINES;
A AA 1 2 1 0  A CC 1 1 1 1  A CC 3 1 1 0  B XX 3 4 5 1  B XX 3 4 5 0  B XX 1 5 4 0
B AA 1 7 3 1  B CC 0 2 1 0  A AA 1 2 1 1  A AA 1 1 3 0  A CC 3 0 1 1  B XX 3 4 5 0
B XX 3 4 5 0  B XX 1 5 4 1  B AA 1 7 4 0  B CC 0 2 3 1  B XX 3 1 5 0  B XX 3 1 5 0
;
%IV_X_C_STAT(example1, Y, X1 X2 X3, C1 C2);
```

The results of processing of "example1" through the macro %IV_X_C_STAT are given in TABLE 3.

- IV is not computed for X2 because there are "zero cells", that is, levels where either g_k or b_k is zero.
- The predictor X1 is monotonic. This is indicated because the c-statistic equals the x-statistic.
- The c-statistic is not computed for C1-C2 since character variables are assumed to be unordered.

TABLE 3: PROCESSING DATA SET "example2" VIA %IV_X_C_STAT

Obs	VAR_NAME	NOMINAL	Levels	IV (Info Value)	C_STAT	X_STAT	MONOTONIC
1	C1	YES	2	0.1080	n/a	0.5779	
2	C2	YES	3	0.3801	n/a	0.6623	
3	X1		3	0.2853	0.6298	0.6298	YES
4	X2		6	n/a	0.5129	0.7077	
5	X3		4	0.6636	0.6753	0.7142	

There is a rough relationship between IV and x-statistic given by TABLE 4 below. TABLE 4 was developed by a simulation study. A wide range of practical sample sizes N and number of levels L of a predictor were selected. In each case IV was fixed in narrow intervals around 0.02, 0.1, 0.2, or 0.3 and the mean of the x-statistic was determined for each IV value range. The results were fairly consistent across the cases but there was considerable variation around the means. See Lund and Brotherton (2013) for details of the simulation.

TABLE 4 – APPROXIMATE RELATIONSHIP BETWEEN IV AND X-STATISTIC

IV	X-Statistic	Siddiqi (2006 p. 81):
0.02	0.55	IV < 0.02 "Not Predictive"
0.1	0.58	IV in [0.02 to 0.1) "Weak"
0.2	0.61	IV in [0.1 to 0.3) "Medium"
0.3	0.64	IV $\geq$ 0.3 "Strong"

Related Work: Alex Lin addresses the screening of NOD predictors by measuring their IV. He provides extensive supporting SAS code. See Lin (2013, 2015).

Preliminary Binning (Fine Classing)

If a predictor has a large number of levels, then IV and x-statistic are "over-fitted" by the excessive degrees of freedom. Preliminary binning (collapsing of levels) will decrease both IV and x-statistic to reduce over-fitting. Preliminary binning is a loosely-defined process. See Finlay (2010, p. 114) for

[9] Contact the author for a proof of the c-statistic, x-statistic relationship.

discussion. Minimum requirements include reducing the number of levels to, perhaps, 15 or less and removing all "zero cells". Preliminary binning should be performed prior to running %IV_X_C_STAT.

Preliminary binning of a batch of numeric predictors can be obtained through PROC RANK. The ranks replace the original values of the predictors. The modeler might select GROUPS=15 for PROC RANK.

For a nominal variable C the ratios g_k / b_k can be used to order levels of C for use in PROC RANK.[10] [11]

Screening Predictors by IV

Once preliminary binning is completed, then screening by %IV_X_C_STAT is used to rank the predictors by IV. Per Siddiqi's guidelines, predictors with an IV less than 0.02 are screened out. The modeler then might also decide to eliminate some lower ranked predictors.

FINAL BINNING OF NOD PREDICTORS

NOD predictors that pass screening by %IV_X_C_STAT still need to receive final binning (also called coarse classing). The goals of final binning of a predictor X include:

- Collapsing pairs of levels X_r and X_s where g_r / b_r = g_s / b_s since such a collapse will not decrease IV.[12]
- Achieving monotonicity when X is ordered and monotonicity is needed due to business requirements.
- Reducing degrees of freedom while maintaining IV to the fullest extent possible.

Macro %BEST_COLLAPSE addresses the goals of final binning by providing the following features:[13]

- **Choice of Eligible Pairs**: At the modeler's choice %BEST_COLLAPSE will consider for collapsing: (i) adjacent levels only (in the ordering of X) or (ii) any two levels.
- **Achieving Monotonicity for Discrete (or Ordered) Predictors**: When collapsing adjacent levels the c-statistic and x-statistic are computed by the macro. Monotonicity is achieved at the first collapse where c-statistic equals x-statistic.
- **Maximizing IV**: At each step the eligible pair of levels that is chosen for collapse is the pair whose collapse will decrease IV the least. That is, the IV after this step is maximized.
    - Despite selecting levels at each step for collapse which maximize IV, it is possible for this collapsing process to lead to a suboptimal solution.[14]

The stopping point of the collapsing process in final binning of X is a subjective decision by the modeler.[15] At a stopping point the modeler should be satisfied with (i) the trend across the final groupings of the levels (monotonic or otherwise) when X is ordered, (ii) the magnitude of the separation between g_k / b_k across k (the good-bad odds), and (iii) a logical business interpretation of the final groupings of the levels.

EXAMPLE 2: %BEST_COLLAPSE

The data set "example1" is re-used in EXAMPLE 2 to illustrate %BEST_COLLAPSE. The numeric predictor X3 from "example1" will be collapsed. In TABLE 5 the counts of Y versus X3 are shown.

---

[10] If b_k = 0, then g_k / b_k is set to 99999999. A new variable "R" is created. For an observation where C = "C_k" the value of R = g_k / b_k. PROC RANK is applied to R. The motivation for ranking by R is based on the fact that IV does not decrease if levels C_r and C_s are collapsed where g_r / b_r = g_s / b_s. So, if g_r / b_r and g_s / b_s are adjacent, the collapsing of these levels should not decrease IV "by much". Contact the author for SAS code for preparing a data set with character variables and their associated R variables.

[11] A related interesting idea: Collapsing two levels X_r and X_s where g_r / b_r and g_s / b_s are *closest* together would seemingly maximize IV among the other choices. But the algorithm can fail to select the best two levels X_r and X_s to collapse in order to maximize IV. An example is given by Lund and Brotherton (2013, p. 7).

[12] This practice ignores a remote possibility of informative interactions between levels "r" and "s" and other predictors.

[13] An early version of %BEST_COLLAPSE appeared in Lund and Brotherton (2013).

[14] The collapsing process can become suboptimal. A non-maximal collapse at step "n" can lead to an opportunity for a greater IV when collapsing at step "n+1". See Lund and Brotherton (2013) for an example and discussion.

[15] For statistics to support a stopping decision, see Lund and Brotherton (2013)

TABLE 5 – DATA SET TO ILLUSTRATE %BEST_COLLAPSE

Table of X3 by Y				
X3	Y			Memo:
	0	1	TOTAL	Pk
1	3	3	6	0.500
3	1	2	3	0.667
4	2	1	3	0.333
5	5	1	6	0.167
TOTAL	11	7	18	

This macro call specifies that only adjacent levels will be collapsed (parameter value "**J**").

```
%BEST_COLLAPSE(example1, X3, Y, 1, IV, J, NOMISS, YES, , WOE);
```

Among the %BEST_COLLAPSE reports is the Summary Report (TABLE 6). It shows the collapsing at each step (columns L1 – L4). After the first step (where k = 3 and where 1 and 3 are collapsed), the new X3 becomes monotonic as signaled by x-statistic = c-statistic = 0.69481.

In this hypothetical example the IV's are very high. Stopping the collapsing at k = 3 preserves a high IV while achieving monotonicity.

TABLE 6 – SUMMARY REPORT FROM %BEST_COLLAPSE

k	IV	X_STAT	C_STAT	MONO	L1	L2	L3	L4
4	0.66368	0.71429	0.67532		1	3	4	5
3	0.60689	0.69481	0.69481	YES	1+3	4	5	
2	0.51753	0.67532	0.67532	YES	1+3	4+5		

TRANSFORMING NOD PREDICTORS BY WEIGHT-OF-EVIDENCE (WOE)

An alternative to CLASS / DUMMY variable coding of a NOD predictor is the weight-of-evidence (WOE) transformation. For predictor C and target Y the weight-of-evidence transformation of C is given by the right-most column in TABLE 7.

TABLE 7 – WEIGHT OF EVIDENCE TRANSFORMATION OF C

C	Y = 0 "B_k"	Y = 1 "G_k"	Col % Y=0 "b_k"	Col % Y=1 "g_k"	**WOE= Log(g_k/b_k)**
C1	2	1	0.250	0.125	-0.69315
C2	1	1	0.125	0.125	0.00000
C3	5	6	0.625	0.750	0.18232
SUM	8	8			

The formula for the transformation is: If C = "C_k" then C_woe = log(g_k / b_k) for k = 1 to L where g_k, b_k > 0.

Woe Coding Vs. Dummy Coding and Class Variable Coding

Counts in TABLE 7 give rise to data set "example3" where C_dum1 and C_dum2 give DUMMY variable coding of C and C_woe is the WOE transform of C. A numeric predictor X1 is added for later discussion.

As seen in TABLE 8A the three ways of coding C (WOE transform, DUMMY coding, and the CLASS statement) produce models with exactly the same probabilities.

EXAMPLE 3: WOE CODING VERSUS DUMMY CODING AND CLASS VARIABLE CODING

```
DATA example3;
INPUT Y C $ C_woe C_dum_1 C_dum_2 X1 @@;
DATALINES;
0 C1 -0.69315 1 0  2  0 C1 -0.69315 1 0  1  1 C1 -0.69315 1 0  1  0 C2  0.00000 0 1  1
1 C2  0.00000 0 1  1  0 C3  0.18232 0 0  1  0 C3  0.18232 0 0  1  1 C3  0.18232 0 0  1
0 C3  0.18232 0 0  1  0 C3  0.18232 0 0  1  1 C3  0.18232 0 0  1  1 C3  0.18232 0 0  1
0 C3  0.18232 0 0  1  1 C3  0.18232 0 0  1  1 C3  0.18232 0 0  1  1 C3  0.18232 0 0  2
;
```

```
PROC LOGISTIC DATA = example3; MODEL Y = C_woe;
OUTPUT OUT = OUT1A  P = P1;
PROC LOGISTIC DATA = example3; CLASS C; MODEL Y = C;
OUTPUT OUT = OUT2A  P = P2;
PROC LOGISTIC DATA = example3; MODEL Y = C_dum_1 C_dum_2;
OUTPUT OUT = OUT3A  P = P3;
DATA OUT123A; MERGE OUT1A OUT2A OUT3A;
PROC PRINT DATA = OUT123A; VAR P1 P2 P3;
```

TABLE 8A – PROBABILITIES FROM OUT123A

Obs	P1	P2	P3
1	0.66664	0.66664	0.66664
2	0.66664	0.66664	0.66664
3	0.66664	0.66664	0.66664
13 more observations omitted			

The results of TABLE 8A are an example of the general case: The three ways of coding C (WOE transform, DUMMY coding, and the CLASS statement) produce models with exactly the same probabilities provided there are no other predictors in the model.

But when other predictors are added, the model with predictor C_woe will not produce the same probabilities as produced by models with CLASS / DUMMY coding. The predictor X1 is added to the three logistic models below. The probabilities from these three models are given in TABLE 8B

```
PROC LOGISTIC DATA = example3; MODEL Y = C_woe X1;
OUTPUT OUT = OUT1B  P = P1;
PROC LOGISTIC DATA = example3; CLASS C; MODEL Y = C X1;
OUTPUT OUT = OUT2B  P = P2;
PROC LOGISTIC DATA = example3; MODEL Y = C_dum_1 C_dum_2 X1;
OUTPUT OUT = OUT3B  P = P3;
DATA OUT123B; MERGE OUT1B OUT2B OUT3B;
PROC PRINT DATA = OUT123B; VAR P1 P2 P3;
```

TABLE 8B – PROBABILITIES FROM OUT123B

Obs	P1	P2	P3
1	**0.61730**	**0.61760**	**0.61760**
2	0.68924	0.69114	0.69114
3	0.68924	0.69114	0.69114
13 more observations omitted			

Which transformation is superior? A WOE transformation or CLASS / DUMMY variable coding?[16]

When other predictors are in the model, CLASS / DUMMY coding produces a model with better fit as measured by log-likelihood. The reason is that the dummy variables, with L-1 coefficients, allow for greater interaction (real or spurious) with other predictors when fitting the logistic model than does the single coefficient of the WOE coded version.

See **Appendix A** for several examples.

If steps were taken to reduce the correlation of the WOE coded predictor with other predictors in the model, then this difference in log likelihood is minimized.

Despite the short-fall in fit by the WOE transformations, the choice of using WOE vs. CLASS / DUMMY is not a pivotal decision in enabling the building of good logistic regression models. The pros and cons are discussed by Finlay (2010, pp. 155-159) and Thomas (2009, pp. 77-78).

Siddiqi recommends the use of WOE coding for credit risk models where he shows how WOE coding naturally supports score-card development. See Siddiqi (2006 pp. 91-92, 116).

[16] Assumes variable selection (stepwise, etc.) is not used so that all dummy variables are in the model.

As discussed later, the WOE transform can be very useful when using NOD predictors in Best Subsets (SELECTION = SCORE) within PROC LOGISTIC.

Degrees of Freedom for WOE Coded Variable

A WOE predictor could not be assigned more degrees of freedom than the corresponding CLASS / DUMMY predictor since a WOE transformation produces a model with less fit. But it should not have less when considering results like that of TABLE 8A. This leads to a conservative working rule:

If C has L levels, then its WOE transformation adds L-1 degrees of freedom to a model.

As discussed later in the paper, the correct degrees of freedom for a WOE coded predictor must be taken into account when computing the Schwarz Bayes criterion (SBC) of a logistic regression model.

WOE and %BEST_COLLAPSE

The "WOE" parameter in %BEST_COLLAPSE specifies that SAS code for WOE transformations be produced for each step in the collapsing. As an illustration the SAS code for the WOE transform for k = 3 in TABLE 6 is given below:

```
if X3 in ( 1,3 ) then X3_woe = 0.6751286751 ;
if X3 in ( 4 ) then X3_woe = -0.241162057 ;
if X3 in ( 5 ) then X3_woe = -1.157452789 ;
```

Final Binning is Final: I think that all the final groups (i.e. groups of collapsed levels), after final binning of a predictor, should be offered to the logistic model either through a CLASS statement or through weight-of-evidence coding. The predictor would either appear or not appear in the logistic regression model. A variable selection process, such as stepwise, could remove some groups while other groups entered the model. This would be "unintended collapsing".

CONCLUSION

Final binning of NOD predictors via %BEST_SUBSETS (after preliminary binning) is recommended as the method to prepare NOD predictors for logistic regression modeling. After final binning the modeler may choose either CLASS coding or WOE coding of the predictor.

DUMMY coding is not recommended if used with a variable selection method (stepwise, etc.). The non-selection of a dummy variable violates the "final binning is final" guideline.

SELECTION AND TRANSFORMATION OF CONTINUOUS PREDICTORS

Now the focus changes to continuous predictors. An approach to the selection and transformation of continuous predictors is the function selection procedure (FSP). FSP is explained in detail in a book by Royston and Sauerbrei (2008). Using SAS macro %FSP_LR8 described in this section it is possible to screen and transform 100 continuous predictors using the FSP in only a couple of minutes. The SAS macros for FSP are discussed in Lund (2015). A short summary of the FSP is presented next.

Fractional Polynomials

A class of transformations of X, called fractional polynomials (FP), provides the tool set for FSP. The use of fractional polynomial transformations first requires that X be translated so that the values of X are positive. Then the fractional polynomial transformations of X are given by:

X^p where p is taken from S = {-2, -1, -0.5, 0, 0.5, 1, 2, 3} and where "0" denotes log(x)

FP1 refers to the collection of functions formed by the selection of one X^p. That is,

$$g(X,p) = \beta_0 + \beta_1 X^p$$

FP2 refers to the collection of functions formed by selection of two X^p. That is,

$$G(X,p_1,p_2) = \beta_0 + \beta_1 X^{p1} + \beta_2 X^{p2} \qquad p_1 \neq p_2$$

$$G(X,p_1,p_1) = \beta_0 + \beta_1 X^{p1} + \beta_2 X^{p1} \log(X) \qquad p_1 = p_2$$

FP2 produces a variety of non-monotonic curves while FP1 produces only monotonic curves. There are 8 FP1 functions and 36 FP2 functions.

Selecting the Best Transformation for X using FSP

The FSP conducts two major steps:

First, the function within FP1 having the maximum likelihood and the function within FP2 having the maximum likelihood are found by an exhaustive search via PROC LOGISTIC runs.

Second, FSP conducts a three-step statistical test to decide what transformation, if any, is best. The possible outcomes are: (1) Drop X from consideration, (2) Use X (linear), (3) Use best FP1 solution, (4) Use best FP2 solution (that is, the two fractional polynomials, not the linear combination of the two).

Royston and Sauerbrei (2008 p. 267) give links to software for performing FSP including Stata, R, and SAS. The SAS version, a macro named %MFP8, was current as of 09/02/2015. It was written in SAS version 8.[17] A code change is needed to %MFP8 in order to run under current SAS versions.[18]

%MFP8 has many powerful options and features but it was not designed for efficient screening and transforming of a large number of continuous predictors. %MFP8 processes only one predictor at a time and PROC LOGISTIC is run 47 times for that processing. The user is required to make a preliminary translation of X to insure that X is positive.

Two macros, %FSP_36LR and %FSP_8LR, are presented in Lund (2015) that implement the running of FSP for a large number of predictors in an efficient manner. Both macros allow any number of predictors to be entered into a macro parameter. Predictors whose minimum value is less than 1 are first translated so that the new minimum value equals 1. Then the output of processing of these predictors is consolidated into a single report. To prepare for FSP processing the training dataset is passed only twice. It is passed once to find the minimum value of the predictors and once more to translate the predictors and to create the FSP transformations.

An important difference between %FSP_36LR and %FSP_8LR is the following:

- %FSP_36LR runs 36 PROC LOGISTIC's for each X and finds the optimal FP1 and FP2 solutions.
- %FSP_8LR runs 8 PROC LOGISTIC's for each X and finds the optimal FP1 solution but may not find the optimal FP2 solution.

A study of occurrence rate and severity of non-optimal FP2 solutions from %FSP_8LR is needed. Of cases examined, the occurrence rate is not high and the severity is not material. See Lund (2015) for more explanation and discussion of the macros %FSP_36LR and %FSP_8LR.

CONCLUSION

Transforming continuous predictors via %FSP_8LR is recommended as the method to select and prepare continuous predictors for logistic regression modeling.

FSP Solutions and Degrees of Freedom when Fitting the Logistic Model

When using the best FP1 solution in a logistic regression model the number of degrees of freedom to be associated with that solution should be 2. In the preliminary step of screening and transforming of the predictor X there was a search for the best exponent from the set S = {-2, -1, -0.5, 0, 0.5, 1, 2, 3}. Although this search of exponents was constrained to S, the conservative number of degrees of freedom for the best FP1 transformation of X should be 2. Likewise, each of the two FP2 predictors contributes 2 degrees of freedom.[19]

---

[17] Meier-Hirmer, Ortseifen, and Sauerbrei (2003). Downloaded from http://portal.uni-freiburg.de/imbi/mfp

[18] See Lund (2015, Appendix B)

[19] The search for the exponents for FP1 and FP2 solutions could be extended beyond the subset S by using PROC NLMIXED where each exponent becomes one of the parameters to fit. However, for large scale screening and transforming of continuous predictors this is not a workable addition to FSP.

SCHWARZ BAYES CRITERION

In direct marketing, customer relationship management, or credit scoring (where there are large training data set samples and many candidate predictors) it is easy and tempting to over fit a logistic regression model. It is natural to want to use all the information in the fitting of models that has been uncovered through data discovery. But the impact of using "all the data" in fitting a logistic model may be the creation of data management overhead with either minimal or no benefit for out-of-sample prediction by the model.

Of course, a purpose of a validation data set is to detect over-fitting. But a "penalized measure of fit" can help to detect over-fitting before proceeding to validation. A well-known penalized measure of fit for logistic regression models is the Schwarz Bayes criterion (SBC). The formula is given below:

$$SBC = - 2 * LL + \log(n) * K$$

where LL is log likelihood of the logistic model, K is degrees of freedom (including the intercept) and n is the sample size.

The theory supporting the Schwarz Bayes criterion is complex, both conceptually and mathematically. For a logistic modeling practitioner the primary practical consequence of the theory is that a model with smaller SBC value is preferred over a model with a larger SBC value.[20]

It is important to compute the K of the SBC by counting both actual and implicit degrees of freedom:

- Each WOE contributes L-1 degrees of freedom where L is the number of levels of the predictor.
- FS1 predictor contributes 2 degrees of freedom and each of the two FS2 predictors contributes 2 degrees of freedom.

The key conclusion is that, once degrees of freedom have been adjusted, SBC provides a means to rank all the possible models that are fitted to the training data set (with a fixed target variable).

MULTIPLE PROMISING CANDIDATE MODELS FOR COMPARISON AND EVALUATION

The building of a logistic regression model is often an iterative process of fitting models and looking at the results. It is better to replace the iterative process with an automated and structured process. A strategy for automated, structured model development involves these two steps:

Step 1: Use an automated process to find "**M**" promising candidate models where M might be between 10 and 30. These models are found by modeling on the training data set.

Step 2: Use a structured evaluation of these M models on the validation data set with regard to: (i) parsimonious fit, (ii) goodness-of-fit, (iii) predictive accuracy, and, (iv) more subjectively, satisfying business requirements.

Following Steps 1 and 2, the "final" model is selected and a final measure of performance of this model is made on the Test Data Set. This section of the paper addresses Step 1.

Two automated processes for finding multiple candidate models are presented next.

METHOD 1: PROC LOGISTIC USING BEST SUBSETS AND PENALIZED SCORE CHI-SQUARE [21]

The Best Subsets method of finding multiple candidate models is realized by using PROC LOGISTIC with SELECTION option "SCORE". SCORE has 3 options: START, STOP, BEST.

PROC LOGISTIC; MODEL Y = <X's> / SELECTION = SCORE START=s1 STOP=s2 BEST=b;

The SELECTION options "START" and "STOP" restrict the models to be considered to those where the number of predictors is between s1 and s2. Then for each k in [s1, s2] the option "BEST" will produce the b "best" models having k predictors. These b "best" models are the ones with highest score chi-square.

Comment: The SELECTION = SCORE can efficiently find the "best" models as directed by START, STOP, BEST because the "score chi-square" is computed for the models without actually solving the

[20] An introductory discussion of "Information Criteria" (SBC, AIC, and more) is given by Dziak, et al. (2012).
[21] This approach generally follows the approach of SAS Institute (2012, section 3.5) but with modifications.

likelihood equations to find the maximum likelihood. When running SELECTION = SCORE the model coefficients are not found and log likelihood (LL) statistics for the model are not computed. This includes statistics that are derived from LL including SBC. The score chi-square closely approximates the likelihood-ratio chi-square. [22]

Example of START, STOP, BEST:

If there are 4 predictors, X1 – X4 and START=1, STOP=4, and BEST=3, then a total of 10 models is selected as shown:

- From the four 1 variable models: {X1}, {X2}, {X3}, {X4}, take the 3 with highest score chi-square
- From the six 2 variable models: {X1 X2}, {X1 X3}, {X1 X4}, {X2 X3}, {X2 X4}, {X3 X4}, take the 3 with highest score chi-square
- From the four 3 variable models: {X1 X2 X3}, {X1 X2 X4}, {X1 X3 X4}, {X2 X3 X4}, take the 3 with highest score chi-square
- From the one 4 variable model: {X1 X2 X3 X4}, take the 1 model

Penalized Score Chi-Square

A "best" model is best only within a set of models with the same number of predictors. For example, {X1} might have the largest score chi-square among 1 variable models but {X1, X2} is guaranteed to have a larger score chi-square simply because a predictor was added to the model. To make the models comparable, a penalty term is needed to reflect the number of degrees of freedom in the model and the sample size. In the absence of SBC a substitute is "ScoreP", a penalized score chi-square defined by:

$$\text{ScoreP} = \text{-Score Chi-Sq} + \log(n) * K \quad \ldots \ [23]$$

where K is the degrees of freedom in the model (counting the intercept) and n is the sample size.

If all possible models are ranked by ScoreP and also by SBC there is no guarantee that the rankings will be the same but the rankings will be very similar.

Problem and Solution but Another Problem

The SELECTION = SCORE option does not support CLASS statements. A solution is to convert class variables into a collection of dummy variables. But with even a modest number of class variables the conversion to dummy variables could cause the total number of predictors to be 100 or more. Run time for PROC LOGISTIC begins to increase exponentially as the number of predictors for SELECTION = SCORE becomes 50 and greater. This makes large scale dummy variable conversion not practical when using SELECTON = SCORE.[24]

A solution is to transform class variables to weight-of-evidence coded predictors before running SELECTION = SCORE. The run-time issue is solved but the ranking of models by score chi-square does not take into account the implicit degrees of freedom of WOE predictors. (This same degrees of freedom issue applies to FP1 and FP2 solutions).

Best Subsets and Handling WOE Coded Predictors

Here is a two-step approach which resolves the degree of freedom issue for WOE coded predictors:

---

[22] See Hosmer et al. (2013 p.15 and p. 42) for a brief discussion of score chi-square.

[23] The connection between SBC and ScoreP is given here:
Likelihood ratio chi-square is:
LR $\chi 2 = -2*LL_r - (-2*LL_f)$ where "r" is intercept-only model and "f" is model with covariates.
$SBC = -2*LL_f + \log(n)*K = -LR\ \chi 2 + \log(n)*K - 2*LL_r$
The SELECTION option gives Score $\chi 2$. But: LR $\chi 2 \sim$ Score $\chi 2$
So now, $SBC \sim -\text{Score}\ \chi 2 + \log(n)*K - 2*LL_r$
Definition: $\text{ScoreP} = -\text{Score}\ \chi 2 + \log(n)*K$ … by dropping the constant $-2*LL_r$

[24] SAS Institute (2012, chapter 3, page 78). SAS Training suggests running a preliminary PROC LOGISTIC with SELECTION = BACKWARD FAST to reduce the number of predictors to the point where SELECTION=SCORE can be run. But BACKWARD is likely to not select all the dummies associated with a class variable. This would distort the results of final binning of the class variables.

A. First the modeler selects values of START and STOP that are reasonable for the model. Then a value of BEST is selected so that all possible models between START and STOP are produced.[25] All possible models are needed to insure that the best ScoreP models can be identified in Step B.

B. In a following DATA STEP the ScoreP is computed while taking into account the correct degrees of freedom for the predictors in the model. The correct degrees of freedom are provided by a call to a FORMAT within the DATA STEP.

Finally, ScoreP can rank the models and the best **M** (lowest ScoreP) can be selected for evaluation on the validation sample.

EXAMPLE 4: BEST SUBSETS

EXAMPLE 4 gives an example of the two-step approach. The data set in this example is "example4".[26] It has 100 observations. The first 4 observations are shown below and the complete "example4" data set is given in **Appendix B**. There is one character predictor variable C with 7 levels, a 0/1 target Y, and 3 numeric predictors X1, X2, X8. Of course, data set "example4" is much smaller than encountered in practical applications.

```
DATA example4;
input C$ Y X1 X2 X8 @@;
datalines;
D  0  10.2  6  0.8    A  1  12.2  6  0.6    D  1   7.7  1  0.6    G  1  10.9  7  0.2
<96 more observations>
;
```

1. C_woe is coded (this woe code was produced by %BEST_COLLAPSE).

```
* WOE coding for any CLASS variables is completed first;
DATA example4; SET example4;
if C in ( "A" ) then C_woe = -0.809318612 ;
if C in ( "B" ) then C_woe = 0.1069721196 ;
if C in ( "C" ) then C_woe = 0.6177977433 ;
if C in ( "D" ) then C_woe = -0.403853504 ;
if C in ( "E" ) then C_woe = -1.145790849 ;
if C in ( "F" ) then C_woe = -0.703958097 ;
if C in ( "G" ) then C_woe = 1.4932664807 ;
run;
```

2. A format associates C_woe with its d.f. of 6. The format is enclosed in the macro %UPCASE_FORMAT to insure that the values to be formatted are in upper case. Upper case is useful for down-stream processing. In the case of "example4" there is only one predictor "C_woe" where degrees of freedom will need to be corrected before ScoreP calculation. But if, for example, X1 was the FP1 transformation from FSP, then another row, "X1" = "2", would be entered in the format.

```
* Formats are defined which associate number L-1 to WOE variables having L levels;
%MACRO UPCASE_FORMAT;
PROC FORMAT LIBRARY = Work;
VALUE $df
/* enter VARIABLES and DF ... converts to UPCASE */
%UPCASE
(
"C_woe" = "6"
)
Other = "1"
;
run;
%MEND;
%UPCASE_FORMAT;
```

---

[25] A large enough number is $\binom{K}{\text{FLOOR}[\frac{K}{2}]}$, where K is the count of predictors and ( ) is the combination symbol.

[26] Data set "example4" is an edited adaptation of data set "getStarted" from SAS HPLOGISTIC documentation.

3. %LET statements specify START, STOP, the data set name, the target, and the predictors. Further processing determines the number of predictors NPRED and computes a default value for BEST.

```
/* ++++++++++++++   PARAMETERS   ++++++++++++++++++++++++++ */
/* Set Parameters for SELECTION = SCORE and ScoreP calculations */
%LET DATASET = example4;
%LET INPUT = X1 X2 X8 C_woe;
%LET Y = Y;
* Number of model candidates printed by PROC PRINT statements;
%LET Pobs = 4;
* Determination of the number of predictor variables;
DATA _NULL_; SET &DATASET(obs=1);
   ARRAY Vars {*} &INPUT;
   NPRED = DIM(Vars);
   CALL SYMPUT('NPRED',NPRED);
run;
/* Set START, STOP, BEST parameters for SELECTION = SCORE
   START is defaulted to 1. Change the LET to override.
   STOP is set at the maximum number of variables. Change the LET to override.
   BEST equals COMB(NPRED, FLOOR(NPRED/2)). This value causes all subsets to be
   created so that ScoreP is computed using format-corrected DFs for all Models */
%LET START = 1;
%LET STOP = &NPRED;
/* END: ++++++++++   PARAMETERS   ++++++++++++++++++++++++++ */
DATA _NULL_;
   BEST = COMB(&NPRED, FLOOR(&NPRED/2)); PUT BEST= ;
   CALL SYMPUT('BEST',BEST);
run;
```

4. After PROC LOGISTIC is run, the ODS “Bestsubsets” output is written to “Score_Out” and printed.

```
ODS OUTPUT Nobs = Nobs;
ODS OUTPUT Bestsubsets = Score_Out;
ODS EXCLUDE Bestsubsets;  /* Suppresses print out from PROC LOGISTIC */
PROC LOGISTIC DATA = &DATASET DESCENDING; MODEL &Y = &INPUT
   /SELECTION= SCORE START= &START STOP= &STOP BEST= &BEST;
PROC PRINT DATA = Score_Out(obs = &Pobs);
   VAR NumberOfVariables ScoreChiSq VariablesInModel;
Title1 "First &Pobs Observations";
Title2 "Before computing DF-corrected ScoreP";
```

TABLE 9 – FIRST 4 OBS. FROM PROC LOGISTIC SELECTION=SCORE ODS OUTPUT

Obs	NumberOfVariables	ScoreChiSq	VariablesInModel
1	1	12.3744	C_woe
2	1	4.2986	X8
3	1	3.9882	X2
4	1	1.2090	X1

5. A DATA STEP reads “Score_Out” and writes outs “SCORE_Out2”. During the DATA STEP processing the correct d.f. for C_woe is computed by usage of the format call. Associating the format to C_woe required tricky string manipulation. Then ScoreP is computed.

```
DATA _NULL_; SET Nobs;
IF label = "Number of Observations Used";
CALL SYMPUT('obs', N);
DATA _null_;
   IF (CEXIST("WORK.formats.df.formatc")) THEN CALL SYMPUT('df_exists', "Y");
    ELSE CALL SYMPUT('df_exists', "N");
DATA _null_;
   IF "&df_exists" = "Y" THEN CALL SYMPUT( 'df_call',"+ (put(varname,$df.) - 1)");
    ELSE CALL SYMPUT('df_call', "");
DATA _null_;
   PUT "&df_call";
DATA Score_Out2; SET Score_Out;
```

```
DROP I VARNAME control_var;
LABEL DF = "DF with Intercept";
DF = numberofvariables + 1;
DF_add = 0;
   DO I = 1 TO &NPRED;
      varname = upcase(left(trim(scan(VariablesInModel,I,' '))));
      IF varname > '' THEN DF_add = DF_add &df_call;
      END;
DF = DF + DF_add;
ScoreP = -scorechisq + log(&obs) * DF;
PROC PRINT DATA = Score_Out2(obs = &Pobs) LABEL;
Title1 "First &Pobs Observations";
Title2 "Correct DF and ScoreP";
Title3 "Before sorting by ScoreP";
run;
```

TABLE 10 – FIRST 4 OBS. WITH ScoreP

Obs	NumberOfVariables	ScoreChiSq	VariablesInModel	DF with Intercept	ScoreP
1	1	12.3744	C_woe	7	19.8618
2	1	4.2986	X8	2	4.9117
3	1	3.9882	X2	2	5.2221
4	1	1.2090	X1	2	8.0014

6. The models are sorted by ScoreP. The best "ScoreP" model is {X2, X8} with ScoreP = 4.8656.

```
PROC SORT DATA = Score_Out2 OUT = Score_Out3; BY ScoreP;
PROC PRINT DATA = Score_Out3(obs = &Pobs) LABEL;
Title1 "First &Pobs Observations";
Title2 "Sorted by ScoreP";
run;
```

TABLE 11 – FIRST 4 OBS. WITH SORTED ScoreP

Obs	NumberOfVariables	ScoreChiSq	VariablesInModel	DF with Intercept	ScoreP
1	2	8.9499	X2 X8	3	4.8656
2	1	4.2986	X8	2	4.9117
3	1	3.9882	X2	2	5.2221
4	3	10.4563	X1 X2 X8	4	7.9644

METHOD2: BACKWARD AND FORWARD SELECTIONS (B-F)

Unlike PROC LOGISTIC, PROC HPLOGISTIC does not offer the option SELECTION = SCORE (as of SAS/STAT 14.1). But PROC HPLOGISTIC might provide a substitute to "Best Subsets" by a process I'll call "**B-F**" for Backward and Forward Selections.

B-F is a proposal that needs exhaustive simulation testing before it can be used in practical applications.

In B-F the first step is running PROC HPLOGISTIC with SELECTION METHOD = BACKWARD with the selection of predictors-to-remove being determined by the predictor which gives best (lowest) SBC after removal. Then the second step is running PROC HPLOGISTIC with SELECTION METHOD = FORWARD with the selection of predictors-to-enter being determined by the predictor which gives best (lowest) SBC when entered. All models and their SBC's from "removal" and "entry" are captured. Additionally, all candidate predictors for removal and entry lead to models, and all these additional models and their SBC are captured. These models are then ranked by their SBC's.

If it is assumed that the removal of predictors by BACKWARD is the reverse of entry of predictors by FORWARD, then the combined number of models is K*(K-1) + 1 where K is the number of predictors. For each k between 1 and K-1 there are K models giving a total of (K-1)*K. Then the full model adds one more.

Approximate SBC: The SBC from removal and entry of predictors by HPLOGISTIC is an approximate SBC.[27] The approximate SBC and true SBC can be modestly different. Also models in common to the Backward and Forward steps often have different approximate SBC's. The best combination of the two approximate SBC's seems to be the "average". The minimum SBC is also reported. The average-approximate-SBC is used to rank the models in the report at the end of the B-F process.

Hereafter, "SBC" will be used in lieu of "approximate SBC" unless it is necessary to make the distinction.

SBC and Transformed Predictors:

If no WOE-coded predictors or FSP transformations are used in modeling, then the SBC for the models are correctly calculated by HPLOGISTIC (including the use of CLASS variables). The B-F process will be fully successful at ranking the models by SBC.

In the case of WOE-coded predictors and FSP transformations there are two issues:

1. A WOE-coded predictor can be entered as a CLASS variable so that degrees of freedom are properly accounted for when calculating SBC of the WOE-coded predictor. But then the log-likelihood of the model is increased (vs. entering the WOE-coded predictor as a simple numeric predictor). This decreases the SBC of the model. The degrees of freedom correction is likely to have more impact on SBC than does the increase in log-likelihood.

   When re-fitting the selected M candidate models from B-F on the validation data set the usage of numeric WOE predictors (replacing their CLASS designation) will modestly change the models. This is satisfactory for modelers who want to use WOE predictors in their models.

   The SBC arising from fitting the M candidate models on the validation data set will need to be recomputed with a degrees of freedom adjustment for WOE predictors or any transformed FSP predictors.

2. The degrees of freedom adjustment for FSP transformations cannot be completed before BACKWARD and FORWARD predictor selections are made by PROC HPLOGISTIC. This is a deficiency in B-F with no work-around. As a result the selection of FSP transforms will be favored. Corrected SBC's will be computed in a later DATA STEP for those models with an FSP transform.

EXAMPLE 5: BACKWARD AND FORWARD SELECTIONS (B-F)

In EXAMPLE 5 the "example4" data set and the same predictors X1, X2, X8, C and target Y will be re-used to explain and illustrate B-F. In a preliminary step the character predictor C is converted to C_woe. Then C_woe will be entered as a CLASS variable in the two PROC HPLOGISTIC's.

SAS code for this example appears in **Appendix C**. (Requires SAS/STAT 13.2 or greater.)

FORMAT: If FSP transformed predictors appear in models from the B-F process, then the SBC for these models requires correction. This is done in a DATA STEP through the "$df" format. In EXAMPLE 5 there are no SBC corrections. In this case the $df FORMAT can be omitted with no downstream impact.

```
%MACRO UPCASE_FORMAT;
PROC FORMAT LIBRARY = Work;
VALUE $df
/* enter VARIABLES and DF ... converts to UPCASE */
%UPCASE
(

)
Other = "1"
;
run;
%MEND;
```

[27] SAS/STAT 14.1 User's Guide, High-Performance Procedures p. 434: … if you specify SELECT=AIC, AICC, or BIC, the selection criteria are estimated … and hence do not match the values that are computed when that model is fit outside of the selection routine.

```
%UPCASE_FORMAT;
```

BACKWARD: PROC HPLOGISTIC is run with METHOD = BACKWARD. The option "SELECT = SBC" specifies that the predictor to be removed by BACKWARD is the one that gives the lowest approximate SBC. The "CHOOSE = SBC" is not relevant to this discussion. The "STOP=NONE" directs BACKWARD to continue to the end. "DETAILS = ALL" is required to generate the ODS data sets, discussed next.

```
ods output SelectionDetails = seldtl_b;
ods output CandidateDetails = candtl_b;
PROC HPLOGISTIC DATA= example4;
   CLASS C_woe;
   MODEL Y (descending) = X1 X2 X8 C_woe;
   SELECTION METHOD = BACKWARD
      (SELECT = SBC CHOOSE = SBC STOP = NONE) DETAILS=ALL;
```

The ODS statements collect information about the predictors that are removed by the BACKWARD option (see SelectionDetails) as well as the predictors that were candidates for removal at each step (see CandidateDetails). In addition to the step number and the predictor name, the SBC value is obtained that is the SBC for the model after removal of a predictor or candidate predictor.

```
DATA canseldtl_b; MERGE seldtl_b candtl_b; BY step;
PROC PRINT DATA=canseldtl_b;
```

TABLE 12 shows the results of printing data set "canseldtl_b". The Criterion is "approximate SBC". The predictors selected "for removal" are given in the "Effect-Removed" column. Predictors considered as "candidates for removal" are listed in the "Effect" column. For example, Obs #3 shows the SBC of **141.73** that would result if X1 (instead of C_woe) were removed in Step 1. Obs #2 shows the lower SBC of **129.31** that results from removing C_woe in Step 1. C_woe was entered as a CLASS variable and the correct degrees of freedom (d.f. = 6) were counted in the computation of SBC.

The best model is {X2, X8} with an SBC of **128.19**. The model {X2, X8} is the result of removing C_woe and X1 in step 1 and step 2.

TABLE 12 – MODELS FROM METHOD=BACKWARD (BEFORE CORRECTION FOR DF)

Obs	Step	Number InModel	Effect Removed	Effect	Criterion (approx SBC)
1	0	5			.
2	1	4	C_woe	C_woe	**129.31**
3	1	4	C_woe	X1	**141.73**
4	1	4	C_woe	X2	146.51
5	1	4	C_woe	X8	146.83
6	2	3	X1	X1	**128.19**
7	2	3	X1	X8	131.61
8	2	3	X1	X2	131.74
9	3	2	X2	X2	128.38
10	3	2	X2	X8	128.70
11	4	1	X8	X8	128.22

BACKWARD models (from predictors selected for removal or considered as a candidate for removal) will create K*(K+1) / 2 models where K is the number of predictors in the MODEL statement. But for even modest size K the model count of K*(K+1) / 2 is far less than the 2^K - 1 of all possible models.

FORWARD: To supplement the K*(K+1)/2 models from BACKWARD, PROC HPLOGISTIC can be run again with FORWARD and SELECT = SBC. The ODS statements collect information about predictors that are entered by the FORWARD option (see SelectionDetails) and predictors that were candidates for entry at each step (see CandidateDetails).

```
ods output SelectionDetails = seldtl_f;
ods output CandidateDetails = candtl_f;
PROC HPLOGISTIC DATA = example4;
   CLASS C_woe;
   MODEL Y (descending) = X1 X2 X8 C_woe;
```

```
   SELECTION METHOD = FORWARD
      (SELECT = SBC CHOOSE = SBC STOP = NONE) DETAILS=ALL;
DATA canseldtl_f; MERGE seldtl_f candtl_f;
BY step;
run;
```

Processing "canseldtl_b" and "canseldtl_f": The information in data sets "canseldtl_b" and "canseldtl_f" is processed to produce a variable called VariablesInModel whose values are the list of predictors in the model. Each observation corresponds to a model and has an associated SBC. The same model (same VariablesInModel) can appear in both "canseldtl_b" and "canseldtl_f".

Degrees of Freedom and Correction to SBC: In a DATA STEP corrections to SBC (for example, for FSP transformed predictors) are made by applying the FORMAT $df to the uncorrected SBC.

Deduped Models and Average SBC: The models from BACKWARD and FORWARD are deduped (with respect to having the same variables-in-model). For duplicate models the average SBC is computed (as well as the minimum SBC).

In TABLE 13 the best model according to average SBC is {X2, X8}. The true SBC appears in the "memo" column. The approximate and true SBC's give the same ranking except for Obs 7 and 8.

TABLE 13 – MODELS FROM B-F

Obs	VariablesInModel from consolidated BACKWARD and FORWARD	Average SBC (ranking)	Minimum SBC	# of Models in Average	MEMO: True SBC
1	X2 X8	128.237	128.185	2	128.212
2	Intercept Only	128.321	128.218	2	128.425
3	X8	128.554	128.377	2	128.672
4	X2	128.869	128.695	2	129.014
5	X1 X2 X8	130.239	129.307	2	131.167
6	X1 X2	131.608	131.608	1	131.932
7	X1	131.822	131.822	1	132.096
8	X1 X8	131.922	131.744	2	131.819
9	X2 X8 C_WOE	141.778	141.734	2	141.740
10	X8 C_WOE	142.610	142.610	1	142.961
11	C_WOE	143.541	143.541	1	143.767
12	X1 X2 X8 C_WOE	145.570	145.558	2	145.582
13	X1 X8 C_WOE	146.515	146.515	1	147.016
14	X1 X2 C_WOE	146.834	146.834	1	147.454

Comments: Candidate Models from Backward and Forward (B-F)

- If BACKWARD and FORWARD follow the same sequence of variable removals and, in reverse order, entries, then the total number of candidate models is:

  $$K*(K-1) + 1$$

  For large K this is roughly double the number from BACKWARD alone. For each k between 1 and K-1 there are K models giving a total of (K-1)*K. Then the full model adds one more.

  Without assuming the BACKWARD and FORWARD selections are equal, a non-tight upper bound for the number of models is simply the sum of the BACKWARD and FORWARD models:

  $$K*(K+1)/2 + K*(K+1)/2 = K*(K+1)$$

- Using Average SBC (which is based on approximate SBC) the modeler can decide on a cut-off of "**M**" models for a final evaluation on a validation data set by selecting the M models with lowest SBC.
- Because of the high-performance feature of HPLOGISTIC the number of predictors to be offered to the fitting of the model is less of an issue than for SELECTION = SCORE of PROC LOGISTIC.
- Does B-F find the best model with respect to SBC? It is very doubtful there is an analytic answer to this question and the recourse is to run extensive simulations.

- A process that is similar to B-F could be centered around SELECTION METHOD = STEPWISE. A speculation is that STEPWISE could go "off-track" as more and more predictors are added. The candidate models with many predictors found by STEPWISE might have higher SBC than the models with many predictors found by BACKWARD. This speculation has not been tested.

Comments: Fitting the M candidate models on the validation data set

- The usage of WOE coded variables as CLASS variables is a convenience in that it places the WOE predictor variable name within the value of "VariablesInModel". See, for example, {X1 X2 C_WOE} in the last row of TABLE 13. Later, using a DATA STEP the values of "VariableInModel" can be put into macro variables by way of SYMPUT to automate the re-fitting of the M candidate models on the validation data set. This is done by macro %CANDIDATE_STATS to be discussed in the final section.

EVALUATION OF THE CANDIDATE MODELS AND FINAL MODEL SELECTION

Now that the predictors for M candidate models have been specified by Best Subsets or B-F, these models will be fitted on the validation data set. There are at least four aspects of evaluation and ranking of the models after fitting them to the validation data set:

- Business Requirements: For example, it may be of interest to have a model which finds the best 10% of a population for direct marketing. Performance of the model outside the top decile might be of less interest. Additionally, a model might be required to have predictors that are understood by the business and which have the expected relationships to the target.
- Specification: Goodness-of-fit (GOF) statistics measure if the model was correctly specified. These statistics might reveal the existence of outliers or the need to add interactions or higher order terms. But GOF statistics are of less interest to the modeler than the measures of predictive accuracy and SBC ranking.
- Predictive Accuracy: Measures of predictive accuracy include the c-statistic, several R-squares, and coefficient of discrimination
- Parsimonious Fit: The models with the best parsimonious fit are those with lowest SBC.

A macro called %CANDIDATE_STATS provides measures of Specification, Accuracy, and Parsimonious Fit for the collection of candidate models that are found by Best Subsets or B-F or by some other method. This macro will be applied to the data sets of EXAMPLE 6.

If a modeler has, perhaps, 6 predictors (i.e. 63 models), then an automated process of finding candidate models might be skipped and the modeler could run %CANDIDATE_STATS on all models (of interest).

EXAMPLE 6: %CANDIDATE_STATS

A training and validation data set are created by the SAS program below:

```
DATA example6T example6V;
DO ID = 1 to 2000;
   cumLogit = ranuni(3);
   e = 1*log(cumLogit/(1-cumLogit));
   X1 = rannor(3);
   X2 = (ranuni(3) < .2);
   H = X1*X2; /* "hidden" */
   W  = floor(20*ranuni(3) + 1);
   X3 = log(W); /* Assume found by FSP - count as 2 d.f. */
   Y_star = 0.2*X1 + 0.5*X2 + 2.0*H + W**(0.10) + 0.5*e;
   Y = (Y_star > 1);
   IF ID <= 1000 THEN OUTPUT example6T; /* Training */
    ELSE OUTPUT example6V; /* Validation */
   END;
```

It is assumed: Variables X1 X2 X3 are the predictors selected by the modeler and X3 was the result of applying FSP and selecting the FP1 solution of "Log" so that 2 d.f. are associated with X3.

In the true model for Y an interaction H = X1*X2 in included. Also, instead of X3 = log(W) in the true model, the fractional polynomial W**(0.10) is used in the true model for Y.

The “Model_Dataset” (collection of models with lists of variables-in-the-models) that is required by %CANDIDATE_STATS can be obtained from Best Subsets or B-F. But instead, the seven possible models are simply read into the data set “Models” as shown:

```
DATA Models;
Length VIM $50;
INFILE datalines delimiter=',';
INPUT VIM $;
DATALINES;
X1,
X2,
X3,
X1 X2,
X1 X3,
X2 X3,
X1 X2 X3,
;
```

A format may be required to associate corrected degrees of freedom to predictors in these models. It is assumed in this example that X3 requires 2 degrees of freedom.

```
%MACRO UPCASE_FORMAT;
PROC FORMAT LIBRARY = Work;
VALUE $df
/* enter VARIABLES and DF ... MACRO converts to UPCASE */
%UPCASE
(
"X3" = "2"
)
Other = "1"
;
run;
%MEND;
%UPCASE_FORMAT;
```

The macro call for %CANDIDATE_STATS is shown next:

```
%CANDIDATE_STATS(7, Models, VIM, Y, , example6T, example6V);
```

Parameters for %CANDIDATE_STATS:

Parameter	From EXAMPLE 6	Description
1. M_ModelsX	7	The first **M_ModelsX** observations from **Model_Dataset** will be processed. If **M_ModelsX** exceeds the count of observations in **Model_Dataset**, it is reset to equal this observation count.
2. Model_Dataset	Models	A data set where each observation describes a candidate model by specifying its predictors. These predictors are listed in the variable in **Model_Dataset** that is identified by parameter **VariablesInModel**.
3. VariablesInModel	VIM	Variable name in **Model_Dataset** that contains the list of predictors in the candidate model. (Predictors are separated by spaces).
4. Target	Y	Variable in training data set **FitData** that gives the target variable for the models specified by **VariablesInModel. Target** has values 0 and 1.
5. Class	“space”	“Space” or All predictors that appear in **VariablesInModel** that will be treated as a CLASS variable when fitting the Logistic Model. Predictors are separated by spaces. Predictors are treated as CLASS variables for every occurrence.
6. FitData	Example6T	The training data set on which the models from **Model_Dataset** are fit.
7. ScoreData	Example6V	The validation data set on which the fitted models are evaluated.

Report from %CANDIDATE_STATS

The report produced by %CANDIDATE_STATS is shown below in TABLE 14. The SBC, after d.f. correction, is called SBC_add_DF. After adding one d.f. to X3, the model {X1 X3} loses its position as best SBC model to model {X1}. The standardized Pearson statistic is not significant for any model despite

the omission in these models of the X1 and X2 interaction which is present in the true model. The Coefficient of Discrimination is closely aligned with R-Square. The AUC (c-statistic) falls below 0.70 for each model which shows no model has strong predictive accuracy (Hosmer, et al. 2013, p. 177).

TABLE 14 - EVALUATION OF THE CANDIDATE MODELS FROM EXAMPLE 6

Model #	Variables in Model	DF Added	SBC	SBC_ add_DF	AUC	R-Square	Max-Rescaled R-Square	Coeff of Discrim	z-value of Std Pearson	2-Tail Prob of z-value
1	**X1**	**0**	**1263.66**	**1263.66**	0.6751	0.083	0.113	0.084	0.5699	0.5687
2	X2	0	1350.50	1350.50	0.5095	0.000	0.000	0.000	-0.7074	0.4793
3	X3	1	1343.82	1350.72	0.5513	0.007	0.009	0.005	-0.9619	0.3361
4	X1 X2	0	1270.52	1270.52	0.6750	0.083	0.113	0.084	0.5081	0.6114
5	**X1 X3**	**1**	**1263.45**	**1270.35**	0.6827	0.090	0.122	0.090	0.4575	0.6473
6	X2 X3	1	1350.15	1357.06	0.5501	0.007	0.010	0.006	-1.2729	0.2031
7	X1 X2 X3	1	1270.21	1277.12	0.6831	0.090	0.122	0.090	0.3899	0.6966

Source of Evaluation Statistics from %CANDIDATE_STATS

- SBC_add_DF is obtained from PROC LOGISTIC option FITSTAT but with correction for degrees of freedom for WOE and FSP predictors and any other correction that modeler decides to make. The correction depends on a user-provided format.
- AUC (c-statistic) is obtained from PROC LOGISTIC option FITSTAT.
- R-Square and Max Rescaled R-Square are obtained from PROC LOGISTIC option FITSTAT.
- Coefficient of Discrimination is found by a calculation on data set **SCOREDATA**. For discussion of the coefficient of discrimination see Allison (2014).
- Two-tail probability for standardized Pearson statistic is found by a calculation on **SCOREDATA**. A two-tail probability which is less than a pre-set alpha value is reason to question whether the model is correctly specified.[28] But GOF statistics seem of limited value for evaluating predictive models.

Allison (2014) gives insights into goodness-of-fit and predictive accuracy measures for logistic models.

CONCLUSIONS

Binning NOD predictors via %BEST_COLLAPSE and transforming continuous predictors via %FSP_8LR are recommended for preparing predictors for logistic regression modeling. Once %BEST_COLLAPSE binning is completed, the modeler can choose either CLASS variable coding or WOE coding. But WOE and FSP transformations come at the "price" of degrees of freedom that must be accounted for when ranking models by penalized measures of fit such as SBC.

Best Subsets and B-F can automate the production of candidate models on the training data set where the ranking of these candidate models relies on ScoreP (Best Subsets) or SBC (B-F). It is very possible the best SBC model will be found by B-F even though not all models are compared by this method.

When entering the model evaluation phase, the subjective but expert judgment of the modeler, guided by the summary report of TABLE 14, can lay the foundation for final model selection.

MWSUG 2015, Omaha, NE

[28] Formulas for standardized Pearson statistic appear in Hosmer et al. (2013 p 203). See examples on pp 206-207. %CANDIDATE_STATS reproduces the examples on pp 206-207 [once a few covariate patterns are slightly perturbed to make each covariate pattern have just one observation]. With some additional effort the standardized sum-of-squares statistic could be added to the GOF statistics. Formulas are given in Hosmer, et al. (2013, p. 203).

SAS MACROS DISCUSSED IN THIS PAPER

%CANDIDATE_STATS, %IV_X_C_STAT, %BEST_COLLAPSE, and %FSP_8LR continue to be under development. Contact the author for copies of the works-in-progress.

REFERENCES

Allison, P. (2014). Measures of Fit for Logistic Regression, *Proceedings of the SAS Global Forum 2014 Conference*, Paper 1485-2014.

Dziak, J, Coffman, D., Lanza, S., Li, R. (2012). Sensitivity and Specificity of Information Criteria, The Methodology Center, Pennsylvania State University, Technical Report Series #12-119. https://methodology.psu.edu/media/techreports/12-119.pdf

Finlay, S. (2010). *Credit Scoring, Response Modelling and Insurance Rating*, New York, Palgrave MacMillan.

Hosmer D., Lemeshow S., and Sturdivant R. (2013). *Applied Logistic Regression, 3rd Ed.,* John Wiley & Sons, New York.

Lin, A. (2013). Variable Reduction in SAS by Using Information Value and Weight of Evidence, *Proceedings of the SAS Global Forum 2013 Conference*, Paper 095-2013.

Lin, A. (2015). Entropy-based Measures of Weight of Evidence and Information Value for Variable Reduction and Segmentation for Continuous Dependent Variables, *Proceedings of the SAS Global Forum 2015 Conference*, Paper 3242-2015

Lund, B. (2015). Selection and Transformation of Continuous Predictors for Logistic Regression, *Proceedings of the SAS Global Forum 2015 Conference*, Paper 2687-2015.

Lund B. and Brotherton D. (2013). Information Value Statistic, *MWSUG 2013, Proceedings*, Midwest SAS Users Group, Inc., paper AA-14.

Raimi, S. and Lund, B. (2012). Efficiently Screening Predictor Variables for Logistic Models, *NESUG 2012, Proceedings*, Northeast SAS Users Group, Inc. Paper SA9.

Raimi, S. and Lund, B. (2013). A Flock of C-Stats, or Efficiently Computing Multiple Statistics for Hundreds of Variables, *Proceedings of the SAS Global Forum 2013 Conference*, Paper 034-2015.

Royston P. and Sauerbrei W. (2008). *Multivariate Model-building*, John Wiley & Sons, West Sussex, UK.

SAS Institute (2012). *Predictive Modeling Using Logistic Regression: Course Notes*, Cary, NC, SAS Institute Inc.

Siddiqi, N. (2006). *Credit Risk Scorecards*, Hoboken, NJ, John Wiley & Sons, Inc.

Thomas, L. (2009). *Consumer Credit Models*, Oxford, UK, Oxford University Press.

ACKNOWLEDGMENTS

Dave Brotherton of Magnify Analytic Solutions provided helpful insights and suggestions.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Bruce Lund
Magnify Analytic Solutions, a Division of Marketing Associates, LLC
777 Woodward Ave, Suite 500
Detroit, MI, 48226
blund@marketingassociates.com or blund_data@mi.rr.com

APPENDIX A: CLASS VARIABLES PRODUCE BETTER FIT THAN WOE VARIABLES

Here are examples that show that CLASS coding produces better fit than WOE coding in models which have other predictors.

```
data test;
do i = 1 to 500;
   z = rannor(1);
   C1 = (-2 <= z <= 2) * z;
   C1 = floor(C1);
   C2 = floor(5*ranuni(1));
   X1 = ranpoi(1, 2); /* random Poisson with seed 1 and mean 2 */
   X2 = ranuni(1);
   Y_star = C1 + 0.2*C2 + X1 + 0.1*X2 + 0.3*rannor(1);
   Y = (Y_star > 1);
   output;
   end;
run;

data test2; set test;
if C1 in ( 0 ) then C1_woe = 0.7537248551 ;
if C1 in ( 1 ) then C1_woe = 3.2496813411 ;
if C1 in ( -1 ) then C1_woe = -0.24808805 ;
if C1 in ( -2 ) then C1_woe = -1.740455431 ;

if C2 in ( 0 ) then C2_woe = -0.568030985 ;
if C2 in ( 1 ) then C2_woe = 0.0135637098 ;
if C2 in ( 2 ) then C2_woe = 0.2221084617 ;
if C2 in ( 3 ) then C2 woe = 0.2792668755 ;
if C2 in ( 4 ) then C2_woe = 0.1431347012 ;
run;

proc logistic data = test2 desc; model y = c1_woe x1 x2;                  /* #1 WOE */
proc logistic data = test2 desc; class c1; model y = c1 x1 x2;            /* #1 CLASS */
proc logistic data = test2 desc; model y = c2_woe x1 x2;                  /* #2 WOE */
proc logistic data = test2 desc; class c2; model y = c2 x1 x2;            /* #2 CLASS */
proc logistic data = test2 desc; model y = c1_woe c2_woe x1 x2;           /* #3 WOE */
proc logistic data = test2 desc; class c1 c2; model y = c1 c2 x1 x2;      /* #3 CLASS */
run;
```

TABLE 15 - BETTER FIT (smaller -2*log-likelihood) FOR CLASS VARIABLES

MODEL Number	WOE: -2*Log(L)	CLASS: -2*Log(L)
1	198.663	192.509
2	372.849	372.020
3	178.865	147.931

Comment: Based on several examples, if C_woe and X are uncorrelated, then MODEL Y = C_woe X has the same log-likelihood as CLASS C; MODEL Y = C X.

APPENDIX B: DATA SET FOR EXAMPLE 4 AND EXAMPLE 5

```
data example4;
  input C$ Y X1 X2 X8 @@;
  datalines;
  D  0  10.2  6  0.8     A  1  12.2  6  0.6     D  1   7.7  1  0.6     G  1  10.9  7  0.2
  E  0  17.3  6  0.4     A  0  18.7  4    1     B  0   7.2  1  0.8     D  0   0.1  3  0.8
  B  1   2.4  4    0     G  0  15.6  7    1     G  0  11.1  3  0.6     A  0     4  6  0.8
  A  0   6.2  2  0.2     B  0   3.7  3  0.4     A  1   9.2  3    0     F  0    14  3    0
  E  1  19.5  6  0.6     C  0    11  3  0.8     B  0  15.3  7  0.4     B  1   7.4  4  0.2
  A  0  11.4  2  0.4     C  1  19.4  1    0     F  0   5.9  4  0.4     F  1  15.8  6  0.6
  B  0    10  3  0.4     E  0  15.7  1  0.2     F  0    11  5    1     G  1  16.8  0  0.8
  D  1    11  4  0.6     G  1   4.8  7    1     G  1  10.4  5    0     F  0  12.7  7    1
  F  0   6.8  1  0.8     E  0   8.8  0  0.4     B  1   0.2  0  0.2     G  1   4.6  7  0.4
  G  1   2.3  2  0.6     B  0  10.8  3  0.8     B  0   9.3  2  0.8     A  0   9.2  6  0.6
  D  0   7.4  0  0.8     F  0  18.3  3  0.2     A  0   5.3  4  0.2     C  0   2.6  5    0
  A  0  13.8  4  0.4     B  1  12.4  6  0.2     B  0   1.3  1  0.6     A  0  18.2  7    1
  G  0   5.2  2  0.8     F  1   9.4  2  0.4     G  1  10.4  2  0.2     G  0    13  1  0.6
  A  0  17.9  4  0.6     D  1  19.4  6  0.2     B  0  19.6  3  0.6     B  1     6  2  0.4
  F  0  13.8  1  0.8     B  0  14.3  4  0.6     E  0  15.6  0  0.4     D  0    14  2  0.6
```

```
  C  1   9.4  5    0     B  0  13.2  1  0.2     A  0  13.5  5  0.4     E  0   2.6  4  0.4
  E  0  12.4  3  0.8     D  0   7.6  2  0.8     B  0  12.7  1  0.4     C  1  10.7  4  0.8
  B  0  10.1  2  0.4     C  1  16.6  1  0.6     B  1   0.2  3  0.4     C  0  10.8  4  0.4
  A  0   7.1  4    0     D  0  16.5  1    0     B  0  17.1  7  0.6     D  0   4.3  1    0
  B  0    15  2    0     F  0  19.7  3  0.6     B  1   2.8  6    1     F  0  16.6  3  0.6
  E  0  11.7  5  0.8     A  0  15.6  3  0.8     C  1   5.3  6  0.4     B  1   8.1  7  0.6
  B  0  14.8  2    0     D  0   7.4  4  0.6     D  0   4.8  3  0.2     A  0   4.5  0  0.8
  D  0   6.9  6  0.4     B  0   4.7  4    1     B  1   7.5  4  0.4     C  0   6.1  0  0.8
  C  0  18.3  1    1     A  0  16.4  7  0.4     B  0   9.4  2  0.4     A  1  17.9  4  0.2
  B  0  14.9  3  0.2     C  0  12.7  0  0.8     A  0   5.4  4  0.4     G  1  12.1  4  0.4
;
run;

DATA example4; SET example4;
if C in ( "A" ) then C_woe = -0.809318612 ;
if C in ( "B" ) then C_woe = 0.1069721196 ;
if C in ( "C" ) then C_woe = 0.6177977433 ;
if C in ( "D" ) then C_woe = -0.403853504 ;
if C in ( "E" ) then C_woe = -1.145790849 ;
if C in ( "F" ) then C_woe = -0.703958097 ;
if C in ( "G" ) then C_woe = 1.4932664807 ;
run;
```

APPENDIX C: CODE FOR B-F FOR EXAMPLE 5

```
* Requires SAS/STAT 13.2 or greater
* Current macro variable values (below) run this program for variables of data set "example4";

/* Enter name of data set */
%LET dataset = example4;
/* Enter Predictor Variables into INPUT for use in HPLOGISTIC MODEL statement */
/* DO NOT use "-" convention */
%LET INPUT = X1 X2 X8 C_woe;
/* Enter CLASS variables from INPUT */
%LET CLASS = C_woe;
/* Enter target variable */
%LET TARGET = Y;
/* Criterion = AIC or SBC */
%LET CRITERION = SBC;
TITLE; /* CLEAR TITLES */
FOOTNOTE; /* CLEAR FOOTNOTES */

/* Length statements for variables:
"__Z_VariablesInModel"
"__Z_Prior_ForwardVariables"
"__Z_Prior_BackwardVariables"
"__Z_vim_new"
containing predictor names is set at 5000 */
/* If length of each name is 32 then the maximum number of predictors for INPUT is = 5000/33 */
/* Or, (sum of lengths of predictors) + (1 per predictor) <= 5000 */

%MACRO UPCASE_FORMAT;
PROC FORMAT LIBRARY = Work;
VALUE $df
/* enter VARIABLES and DF ... MACRO converts to UPCASE */
/* Example of assignment of 2 df to X2.
   Enter within parentheses: "X1" = "2" */
%UPCASE
(

)
Other = "1"
;
run;
%MEND;
%UPCASE_FORMAT;

/* Count number of predictors in INPUT */
DATA _null_;
ARRAY ___xx{*} $ &INPUT;
call symput('NPRED',dim(___xx));
```

```
run;

/* Start Program */
ODS OUTPUT Nobs = Nobs;
ODS OUTPUT SelectionDetails = __X_seldtl_b;
ODS OUTPUT CandidateDetails = __X_candtl_b;

PROC HPLOGISTIC DATA=&dataset;
   CLASS &CLASS;
   MODEL &TARGET (descending) = &INPUT;
   SELECTION METHOD=backward(select = &CRITERION choose = &CRITERION Stop = NONE)
   DETAILS=ALL;
run;
DATA _NULL_; SET Nobs;
If label = "Number of Observations Used";
CALL SYMPUT('obs', N);

DATA __X_canseldtl_b; MERGE __X_seldtl_b __X_candtl_b; BY step;
run;

PROC PRINT DATA = __X_canseldtl_b;
VAR Step NumberInModel EffectRemoved Effect Criterion;
Title "MODELS FROM METHOD=BACKWARD";
run;
Title;

DATA __X_VariablesInModel_B; SET __X_canseldtl_b end=eof; BY step;
LENGTH __Z_Prior_BackwardVariables __Z_VariablesInModel $5000;
RETAIN __Z_Prior_BackwardVariables;

IF "&CRITERION" = "SBC" THEN AICEst = .; /* avoid warning message: AICEst uninitialized */
IF "&CRITERION" = "AIC" THEN BICEst = .; /* avoid warning message: BICEst uninitialized */

/* Criterion for step 0 is missing in candidate data set */
IF step = 0 & "&CRITERION" = "SBC" THEN Criterion = BICEst;
   ELSE IF step = 0 & "&CRITERION" = "AIC" THEN Criterion = AICEst;

IF step = 0 THEN DO;;
   __Z_Prior_BackwardVariables = ","||upcase(left(compbl("&INPUT")))||",";
   __Z_Prior_BackwardVariables = tranwrd(__Z_Prior_BackwardVariables," " , ",");
   END;

Effect_C = trim(compress("," || trim(Upcase(Effect)) || ","));
EffectRemoved_C = trim(compress("," || trim(Upcase(EffectRemoved)) || ","));

IF step >= 0 THEN DO;
   __Z_VariablesInModel = tranwrd(__Z_Prior_BackwardVariables,trim(Effect_C)," ");
   __Z_VariablesInModel = left(compbl(tranwrd(__Z_VariablesInModel,"," , " ")));
   OUTPUT;
   END;

IF last.step & step >= 0 THEN
    _Z_Prior_BackwardVariables = tranwrd(__Z_Prior_BackwardVariables,trim(EffectRemoved_C),",");
run;

ODS OUTPUT SelectionDetails = __X_seldtl_f;
ODS OUTPUT CandidateDetails = __X_candtl_f;

PROC HPLOGISTIC DATA=&dataset;
   CLASS &CLASS;
   MODEL &TARGET (descending) = &INPUT;
   SELECTION METHOD=forward(select = &CRITERION choose = &CRITERION Stop = NONE)
   DETAILS=ALL;
run;
DATA __X_canseldtl_f; MERGE __X_seldtl_f __X_candtl_f; BY step;
   IF "&CRITERION" = "SBC" THEN AICEst = .; /* to avoid warning message: AICEst uninitialized */
   IF "&CRITERION" = "AIC" THEN BICEst = .; /* to avoid warning message: BICEst uninitialized */
   IF EffectEntered = "Initial Model" THEN EffectEntered = "";
/* "Criterion" for step 0 is missing from candidate data set */
   IF step = 0 & "&CRITERION" = "SBC" THEN Criterion = BICEst;
   ELSE IF step = 0 & "&CRITERION" = "AIC" THEN Criterion = AICEst;
```

```
run;

DATA __X_VariablesInModel_F; SET __X_canseldtl_f; BY step;
LENGTH __Z_Prior_ForwardVariables __Z_VariablesInModel $5000;
RETAIN __Z_Prior_ForwardVariables;
IF first.step THEN
   __Z_VariablesInModel = upcase(left(trim(left(__Z_Prior_ForwardVariables)) || " " ||
EffectEntered));
ELSE __Z_VariablesInModel = upcase(left(trim(left(__Z_Prior_ForwardVariables)) || " " ||
Effect));
IF last.step
   THEN __Z_Prior_ForwardVariables = upcase(trim(left(__Z_Prior_ForwardVariables)) || " " ||
EffectEntered);
run;

DATA __X_VariablesInModel; LENGTH CandidateFor $7;
SET   X_VariablesInModel_F   X_VariablesInModel_B;
   LENGTH __Z_n_var $32 __Z_vim_new $5000;
   DROP I __Z_n_len __Z_n_pos __Z_n_var;
   DO I = 1 TO &NPRED;
      __Z_n_var = left(trim(scan(__Z_VariablesInModel,I,' ')));
      IF __Z_n_var > ''
      THEN
      DO;
         __Z_n_len = length(__Z_n_var);
         __Z_n_pos = indexw(upcase("&input"),__Z_n_var);
         SUBSTR(__Z_vim_new,__Z_n_pos,__Z_n_len) = __Z_n_var;
         END;
      END;
   __Z_vim_new = left(trim(compbl(__Z_vim_new)));
run;

DATA _null_;
   IF (CEXIST("WORK.formats.df.formatc")) THEN CALL SYMPUT('df_exists', "Y");
    ELSE CALL SYMPUT('df_exists', "N");
run;
DATA _null_;
put "&df_exists";
IF "&df_exists" = "Y" THEN CALL SYMPUT( 'df_call',"+ (put(varname,$df.) - 1)");
 ELSE CALL SYMPUT('df_call', "");
run;
DATA _null_;
put "&df_call";
run;

DATA __X_VariablesInModel2; SET __X_VariablesInModel;
DROP I varname;
LABEL __Z_DF_add = "Incremental DF from Format";
/* Format based correction to Degrees of Freedom can correct SBC for FSP predictors */
__Z_DF_add = 0;
   DO I = 1 TO &NPRED;
      varname = upcase(left(trim(scan(__Z_VariablesInModel,I,' '))));
      IF varname > '' THEN __Z_DF_add = __Z_DF_add &df_call;
      END;
   IF "&CRITERION" = "SBC" THEN criterion = criterion + log(&obs) * __Z_DF_add;
   IF "&CRITERION" = "AIC" THEN criterion = criterion + 2 * __Z_DF_add;
run;

PROC SORT DATA = __X_VariablesInModel2 OUT = __X_VariablesInModel_sorted;
BY __Z_vim_new criterion;
run;

DATA __X_Report; SET __X_VariablesInModel_sorted;
BY __Z_vim_new criterion;
   LABEL __Z_vim_new = "Variables in Model from consolidated F and B";
   LABEL __Z_average_criterion = "Average &Criterion";
   LABEL __Z_min_criterion = "Minimum &Criterion";
   LABEL __Z_model_count = "# of Models in Average";
   RETAIN __Z_model_count __Z_sum_criterion __Z_min_criterion;
   IF first.__Z_vim_new
   THEN
```

```
   DO;
      __Z_model_count = 0;
      __Z_sum_criterion = 0;
      __Z_min_criterion = .;
      END;
   __Z_model_count+1;
   __Z_sum_criterion + criterion;
   __Z_min_criterion = min(criterion, __Z_min_criterion);
   IF last.__Z_vim_new THEN DO;
   /* Small differences may exist in SBC or AIC in Forward v Backward */
   /* Average is taken */
      __Z_average_criterion = __Z_sum_criterion/__Z_model_count;
      OUTPUT;
      END;
run;
PROC SORT DATA = __X_Report; BY __Z_average_criterion;
run;
/* "Intercept Only" model prints as a space in PROC PRINT */
PROC PRINT DATA = __X_Report LABEL;
VAR __Z_vim_new __Z_average_criterion __Z_min_criterion __Z_model_count;
TITLE "Models from B-F Process - Sorted by Average (approx) &CRITERION";
run;
```