Paper Number PT-02- 2013

Nancy Hu, Chicago, IL

Abstract

This study is intended to assist Analysts to generate the best of variables using simple arithmetic operators (square root, log, loglog, exp and rcp) and such as monthly amount paid, daily number of received customer service calls, weekly worked hours on a project, or annual number total sales for a specific product. During a statistical data modeling process, Analysts are often confronted with the task of computing derived variables using the existing available variables. The advantage of this methodology is that the new variables may be more significant than the original ones. This paper gives a new way to compute all the possible variables using a set of math transformation. The codes include many SAS features that are very useful tools for SAS programmers to incorporate in their future codes such as %SYSFUNC, SQL, %INCLUDE, CALL SYMPUT, %MACRO, SORT, CONTENTS, MERGE, MACRO _NULL_, as well as %DO …%TO … and many more.

Introduction

Statistical data modeling process often leads to the computation of new derived variables using the existing available variable. The main reason for doing so is that the derived data may be potentially stronger predictors for the models to be developed. This project is intended to assist statistician modelers to generate the best of variables using simple arithmetic operators (square root, log, loglog, exp and rcp) and a given characteristic that can be pre-selected based on the purpose of the analysis. An arithmetic formula was developed to calculate the information value of new variables that can be generated. The paper is divided into two different sections.

Part One

The first part prepares the variable transformation into a 5 new variables. Once this section is completed then the second part will compute the best of fit in term of binning variables and WOE and related information value. Let's assume that a modeler was asked to develop a statistical model. During the Exploratory Data Analysis, it was observed that the available data set contains a set of variables such as amount paid and number of minutes used in the dataset. Table 1 is an example of variables in the original dataset.

Table1

ID	DATE	TEST_A	TEST_B	bad
83382960	10/1/2012	21.4	20.4	1
83382961	10/1/2012	21.8	20.5	0
83382962	10/1/2012	24	26	1
83382963	10/1/2012	28	29	0
83382964	10/1/2012	32	32	1
83382965	10/1/2012	24	26	1
83382966	10/1/2012	28	29	0
83382967	10/1/2012	32	32	1
83382968	10/1/2012	28	29	0
83382969	10/1/2012	32	32	1
83382970	10/1/2012	34	36	1
83382971	10/1/2012	28	39	0
83382972	10/1/2012	32	32	1

The first part of the code transforms the original bariable into the format shown below in Table 2. Log_a is log(a), exp_a is exp(a), sqrt_a is squre_root(a), loglog_a is log(log(a)) and rcp_a is 1/a., after that, call %auto_mapp to pick up the best transformation for the dependant variable bad.

ID	DATE	TEST_A	TEST_B	bad	log_a	exp_a	sqrt_a	loglog_
83382960	10/1/2012	21.4	20.4	1	1.330414	1967441884	4.626013	0.12
83382961	10/2/2012	21.8	20.5	0	1.338456	2935078394	4.669047	0.12
83382962	10/3/2012	24	26	1	1.380211	2.6489E+10	4.898979	0.13
83382963	10/4/2012	28	29	0	1.447158	1.4463E+12	5.291503	0.16
83382964	10/5/2012	32	32	1	1.50515	7.8963E+13	5.656854	0.1
83382965	10/6/2012	24	26	1	1.380211	2.6489E+10	4.898979	0.13
83382965	10/7/2012	28	29	0	1.447158	1.4463E+12	5.291503	0.16
83382965	10/8/2012	32	32	1	1.50515	7.8963E+13	5.656854	0.1
83382965	10/9/2012	28	29	0	1.447158	1.4463E+12	5.291503	0.16
83382965	10/10/2012	32	32	1	1.50515	7.8963E+13	5.656854	0.1
83382965	10/11/2012	34	36	1	1.531479	5.8346E+14	5.830952	0.18
83382965	10/12/2012	28	39	0	1.447158	1.4463E+12	5.291503	0.16
83382965	10/13/2012	32	32	1	1.50515	7.8963E+13	5.656854	0.1

Part Two

Call macro of `%plott;` ***to choose the best transform variable from pre-selected variables list for the model and plot top variable which has best information value;***

Conclusion

This method of computing the derived variables can be useful for modeling purposes and save time during the variable creation stage. These macros are very straightforward but the user needs a strong understanding of SAS programming especially in SAS MACRO in order to fully take advantage of the codes.

Contact Information:

Your comments and questions are encouraged. Please contact the author at the following:

Nancy Hu – (nancyhu6@gmail.com)
1100 Shagbark Ct,
Hoffman Estates, IL 60192

Reference

SAS Codes

1. Main Codes

```
libname in "C:\in";     /* input data set*/
libname out "C:\out";   /* output dataset */

filename keeplist "C:\automap1.kep1";    /* keeplist for the final model */
filename droplst "C:\automap1.drp1"; /* droplist from the original dataset*/

%macro keeplst0;
perf key p1_y
 %mend keeplst0;
options mlogic mprint;
data bt;
  set in.test_data;
  weight=1;
  if p1='Y' then perf=1;
  if p1='N' then perf=0;
  weight=1;
  run;

%LET OR_DATA = bt;              /* INPUT DATA SET */
%LET FDATA  = out.test_out;  /* OUTPUT DATA SET contains variable
transformation list*/
%LET SUBV =%KEEPLST0;           /* NULL: ALL; O/W: SUBSET OF INPUT DATA SET
                                   FORMAT: %X, X IS THE MACRO NAME IN SUBDATA */
%LET BAD = perf;                /* DEPENDENT VARIABLE */
%LET WEIGHT_ = Weight;             /* WEIGHT VARIABLE */
%LET PORTION = 1;               /* PROPORTION OF DATA SET USED FOR SELECTION */
%LET LEFT_PCT = 0;              /* THE LEFT SIDE PERCENTILE TRUNCATION: 0-5 */
%LET RITE_PCT = 100;            /* THE RITE SIDE PERCENTILE TRUNCATION: 95-100*/
%LET DELTA = 1;                 /* SMALL ADJUSTIFICATION PARAMETER: .01-1 */
%LET DROP_ALL = 0;              /* 1: DROP ALL OLD VARIABLES ON VARLIST */

/*************** use default values ***************/
%LET SX1 = P;                   /* SUFFIX FOR LOG      */
%LET SX2 = Z;                   /* SUFFIX FOR LOGLOG   */
%LET SX3 = J;                   /* SUFFIX FOR EXP      */
%LET SX4 = U;                   /* SUFFIX FOR SQRT     */
%LET SX5 = Y;                   /* SUFFIX FOR INVERSE */
%LET DATA_IN = out.MODELI;       /* USE TEMP SPACE     */
%LET DATA_OUT =out.MODELO;       /* USE TEMP SPACE     */
/*************************************************/

*PTIONS MEMSIZE=200M SORTPGM=BEST ERRORS=1 S=80;
*OPTIONS NOMPRINT NOMLOGIC NOMTRACE NOSYMBOLGEN;
*options compress=yes;
*_____________________________________________________________;

%MACRO READLIST(LIST,PREFIX=READ,N_ITEM=N_WORD,UPCASE1=1);
   %LOCAL FLAG NN ;
  %LET FLAG = 0;
  %LET NN = 1;
  %GLOBAL &N_ITEM;

%DO %UNTIL (&FLAG=1);
   %IF %QSCAN(&LIST,&NN,' ')= %THEN %DO;
     %LET &N_ITEM=%EVAL(&NN-1);
     %LET FLAG=1;
   %END;
   %ELSE %DO;
```

```
    %GLOBAL &PREFIX&NN;
    %IF &UPCASE1=1 %THEN
          %LET &PREFIX&NN=%UPCASE(%QSCAN(&LIST,&NN,' '));
    %ELSE %LET &PREFIX&NN=%QSCAN(&LIST,&NN,' ');
    %LET NN=%EVAL(&NN+1);
  %END;
%END;

%MEND READLIST;


%MACRO DSN_VAR(DSN,PREFIX=DSVAR,N_ITEM=N_DSVAR,LAST_N=,TYPE=1,ORDER=2);

 /*   A TOOL:
       CONVERTS THE (LAST_N) VARIABLES WITH THE SELECTED TYPE
       (TYPE=1: NUMERIC; TYPE=2: CHARACTER; TYPE=  :ALL THE TYPES)
       IN THE DATA SET INTO A SQUENCE OF GLOBAL MACRO VARIABLES WITH
       THE SPECIFIED PREFIX IN A SPECIAL ORDER (ORDER=1: THE CREATION
       ORDER; ORDER=2: THE ALPHABETICAL ORDER)
 */
  %GLOBAL &N_ITEM;

             /* GET THE VARIABLE LIST FROM THE DATA SET  */
  PROC CONTENTS DATA=&DSN OUT=DS_VAR NOPRINT; RUN;

              /*  SELECT THE RIGHT TYPE OF VARIABLES   */
  %IF &TYPE NE %THEN %DO;
    DATA DS_VAR;
      SET DS_VAR;
        WHERE TYPE=&TYPE;
    RUN;
  %END;

       /* SORT BY THE CREATION ORDER INSTEAD OF THE ALPHBETICAL ONE  */
  %IF &ORDER=1 %THEN %DO;
    PROC SORT DATA=DS_VAR; BY NPOS; RUN;
  %END;

       /* GET THE TOTAL NUMBER (OF THE VARIABLES) INTO A MACRO VAR  */
  PROC CONTENTS DATA=DS_VAR OUT=TOT NOPRINT; RUN;

  DATA _NULL_;
    CALL SYMPUT('NOBS',LEFT(NOBS));
    STOP;
    SET TOT NOBS=NOBS;
  RUN;

             /* DECIDE HOW BIG A TAIL YOU WANT TO CUT OFF   */
  %IF &LAST_N NE %THEN %DO;
    %IF &NOBS > &LAST_N %THEN %DO;
      DATA DS_VAR;
        SET DS_VAR;
        IF _N_ GE (&NOBS-&LAST_N+1);
      RUN;
      %LET NOBS=&LAST_N;
    %END;
  %END;
                  /* CONVERT TO GLOBAL MAROC VARIABLES   */
  %DO SS=1 %TO &NOBS;
    %GLOBAL &PREFIX&SS;
  %END;

  DATA _NULL_;
    SET DS_VAR END=LAST;
       CALL SYMPUT("&PREFIX"||LEFT(_N_), NAME);
```

```
      IF LAST THEN CALL SYMPUT("&N_ITEM",LEFT(_N_));
  RUN;

%MEND DSN_VAR;

%MACRO GETLIST(DSN,V_LIST=&VARLIST,PREFIX=DSVAR,N_ITEM=N_DSVAR);

  %IF &V_LIST= %THEN %DO;
    %DSN_VAR(&DSN,PREFIX=&PREFIX,N_ITEM=&N_ITEM)
    %DO TT=1 %TO &&&N_ITEM;
      %LET VARLIST=&VARLIST &&&PREFIX&TT;
    %END;
  %END;
  %ELSE %DO;
    %READLIST(&V_LIST,PREFIX=&PREFIX,N_ITEM=&N_ITEM)
  %END;

%MEND GETLIST;

%MACRO VNAME_N(VN,VN_N,N=16,SEPARATE=);
  /*
   TOOL: THIS PROGRAM IS HELPFUL FOR THE NAME CONVENTION WHEN
         CREATING RELATED NEW VARIABLES.
         N LETTERS (DEFAULT=16) ARE EXTRACT FROM &VN TO CREATE &VN_N
         WITH AN OPTION TO SUFFIX A SEPARATION SYMBOL (DEFAULT=_).
   */
%GLOBAL &VN_N;
     /*
  %IF %LENGTH(&VN) <= &N %THEN %LET &VN_N = &VN;
  %ELSE %DO;
     */
  %LET V1 = %SCAN(&VN,1,'_');
  %LET V2 = %SCAN(&VN,2,'_');
  %LET V3 = %SCAN(&VN,3,'_');
  %LET V4 = %SCAN(&VN,4,'_');
  %LET VNN = &V1&V2&V3&V4;
  %IF %LENGTH(&VNN) <= &N %THEN %LET &VN_N = &VNN;
  %ELSE %LET &VN_N = %SUBSTR(&VNN,1,&N);
  /* %END;  */

%IF &SEPARATE NE %THEN %LET &VN_N = &&&VN_N..&SEPARATE;

%MEND VNAME_N;

%MACRO LOG_FUNC(VN,LO_LIM=,HI_LIM=,D=&DELTA);

  /* FUNCTION VNL=LOG(VN-LO_LIM+D)  */

  %VNAME_N(&VN,VN_6);
  %LET VNN1 = &VN_6&SX1;
    IF &VN < 0 THEN &VNN1 = 0;
    ELSE &VNN1 = LOG(&VN+&D);
     %IF &HI_LIM NE %THEN %DO;
       IF &VN > &HI_LIM THEN &VNN1 = LOG(&HI_LIM+&D);
     %END;
    IF LENG <= 34 THEN VARLAB1 = SUBSTR(VARLAB,1,LENG)||' (LOG)';
    ELSE VARLAB1 = SUBSTR(VARLAB,1,34)||' (LOG)';

%MEND LOG_FUNC;

%MACRO LLG_FUNC(VN,LO_LIM,HI_LIM=,D=&DELTA);
  /* FUNCTION VNL=LOG(LOG(VN+&D+1))  */

  %VNAME_N(&VN,VN_6);
  %LET VNN1 = &VN_6&SX2;
```

```
    IF &VN < 0 THEN &VNN1 = LOG(LOG(3));
    ELSE &VNN1 = LOG(LOG(&VN+&D+2));
     %IF &HI_LIM NE %THEN %DO;
      IF &VN > &HI_LIM THEN &VNN1 = LOG(LOG(&HI_LIM+&D+2));
     %END;
    IF LENG <= 33 THEN VARLAB2 = SUBSTR(VARLAB,1,LENG)||' (LLOG)';
    ELSE VARLAB2 = SUBSTR(VARLAB,1,33)||" (LLOG)";

%MEND LLG_FUNC;

%MACRO EXP_FUNC(VN,LO_LIM=,HI_LIM=,A=0,B=);
  /* FUNCTION VNE=EXP(A*VN)      */

  %VNAME_N(&VN,VN_6)
  %LET VNN1 = &VN_6&SX3;
  %LET RANGE = %SYSEVALF(&B-&A);
  %IF &RANGE = 0 %THEN %LET RANGE = 1;
  IF &VN < 0 THEN &VNN1 = -1;
  ELSE &VNN1 = -EXP((-1) * &VN / &RANGE );
    %IF &HI_LIM NE %THEN %DO;
      IF &VN > &HI_LIM THEN &VNN1 = -EXP((-1) * &HI_LIM / &RANGE );
    %END;
    &VNN1 = ROUND(&VNN1,0.00001);
    IF LENG <= 34 THEN VARLAB3 = SUBSTR(VARLAB,1,LENG)||' (EXP)';
    ELSE VARLAB3 = SUBSTR(VARLAB,1,34)||' (EXP)';

%MEND EXP_FUNC;

%MACRO SQR_FUNC(VN,LO_LIM=,HI_LIM=,A=0);
  /* FUNCTION VN2=SQRT(VN+&A)   */

  %VNAME_N(&VN,VN_6)
  %LET VNN1 = &VN_6&SX4;
   IF &VN < 0 THEN &VNN1 = 0;
   ELSE &VNN1 = SQRT(&VN+&A);
    %IF &HI_LIM NE %THEN %DO;
      IF &VN > &HI_LIM THEN &VNN1 = SQRT(&HI_LIM+&A);
    %END;
    IF LENG <= 33 THEN VARLAB4=SUBSTR(VARLAB,1,LENG)||' (SQRT)';
    ELSE VARLAB4 = SUBSTR(VARLAB,1,33)||' (SQRT)';

%MEND SQR_FUNC;

%MACRO RCP_FUNC(VN,LO_LIM=,HI_LIM=,D=&DELTA);
  /* FUNCTION VN2=1/(X+&D) */

  %VNAME_N(&VN,VN_6)
  %LET VNN1 = &VN_6&SX5;
  IF &VN < 0 THEN &VNN1 = -1;
  ELSE &VNN1 = -1 / (&VN+&D);
   %IF &HI_LIM NE %THEN %DO;
    IF &VN > &HI_LIM THEN &VNN1 = -1 / (&HI_LIM+&D);
   %END;
    IF LENG <= 34 THEN VARLAB5 = SUBSTR(VARLAB,1,LENG)||' (RCP)';
    ELSE VARLAB5 = SUBSTR(VARLAB,1,34)||' (RCP)';

%MEND RCP_FUNC;

%MACRO PCTL(DSN,PCNTILE,VARLIST=,LEFT=0,RITE=100);
  %GETLIST(&DSN,V_LIST=&VARLIST,PREFIX=VAR,N_ITEM=N_VAR)
  %DO II=1 %TO &N_VAR;
    %GLOBAL P&II._&LEFT P&II._&RITE
            LAB&II._1 LAB&II._2 LAB&II._3 LAB&II._4 LAB&II._5
            VAR&II._1 VAR&II._2 VAR&II._3 VAR&II._4 VAR&II._5;
```

```
  %END;

  PROC MEANS DATA=&DSN NOPRINT;
    VAR &VARLIST;
    %IF &WEIGHT_ NE %THEN %DO;
      WEIGHT &WEIGHT_;
    %END;
    OUTPUT OUT=PCNTILE
           MIN=%DO II=1 %TO &N_VAR; P&II._&LEFT %END;
           MAX=%DO II=1 %TO &N_VAR; P&II._&RITE %END; %STR(;);
  RUN;

  DATA _NULL_;
    SET PCNTILE;
    %DO II=1 %TO &N_VAR;
      %IF &LEFT NE %THEN %DO;
        CALL SYMPUT("P&II._&LEFT",LEFT(P&II._&LEFT));
      %END;
      %IF &RITE NE %THEN %DO;
        CALL SYMPUT("P&II._&RITE",LEFT(P&II._&RITE));
      %END;
    %END;
  RUN;

%MEND PCTL;


%MACRO SLCTVAR(SDATA,INDSN,DROPFILE,NUMKP=1,SLEVEL=0.05);

 DATA SUBSET;
  SET &INDSN;
  %IF &PORTION NE %THEN %DO;
  IF RANUNI(374521) <= &PORTION THEN OUTPUT;
  %END;
 RUN;

 %DO II=1 %TO &N_VAR;
 %IF &&P&II._&RITE_PCT NE . %THEN %DO;
  PROC LOGISTIC DATA=SUBSET(KEEP=&BAD &&VAR&II &&VAR&II._1 &&VAR&II._2
                            &&VAR&II._3 &&VAR&II._4 &&VAR&II._5 &WEIGHT_)
                OUTEST=EST1 NOPRINT;
       MODEL &BAD =&&VAR&II &&VAR&II._1 &&VAR&II._2 &&VAR&II._3 &&VAR&II._4
                   &&VAR&II._5 / SELECTION=FORWARD STOP=&NUMKP SLE=&SLEVEL;
    %IF &WEIGHT_ NE %THEN %DO;
       WEIGHT &WEIGHT_;
    %END;
  RUN;

  DATA ONEVAR(KEEP=DROPVAR);
   LENGTH DROPVAR $8;
   SET EST1(WHERE=(_TYPE_='PARMS'));
   ARRAY VNNS &&VAR&II._1 &&VAR&II._2 &&VAR&II._3 &&VAR&II._4
              &&VAR&II._5;
   DO K=1 TO DIM(VNNS);
    IF VNNS{K} <= .Z THEN DO;
     CALL VNAME(VNNS{K},DROPVAR);
     OUTPUT;
    END;
   END;
  RUN;
  %END;

  %ELSE %DO;
  DATA ONEVAR(KEEP=DROPVAR);
   LENGTH DROPVAR $8;
```

```
   ARRAY VNNS &&VAR&II &&VAR&II._1 &&VAR&II._2 &&VAR&II._3 &&VAR&II._4
             &&VAR&II._5;
   DO K=1 TO DIM(VNNS);
     VNNS{K} = 0;
     CALL VNAME(VNNS{K},DROPVAR);
     OUTPUT;
   END;
   STOP;
  RUN;
  %END;

  PROC APPEND BASE=DROPLIST DATA=ONEVAR FORCE; RUN;
 %END;

  DATA _NULL_;
    SET DROPLIST END=LAST;
    FILE "DROPFILE.dat";
    IF _N_=1 THEN PUT '%MACRO DROPLIST;';
    PUT DROPVAR;
    IF LAST THEN PUT '%MEND DROPLIST;';
  RUN;

%INCLUDE "dropfile.dat";

    DATA &SDATA;
   SET &INDSN;
   LABEL %DO II=1 %TO &N_VAR;
      &&VAR&II._1="&&LAB&II._1"
      &&VAR&II._2="&&LAB&II._2"
      &&VAR&II._3="&&LAB&II._3"
      &&VAR&II._4="&&LAB&II._4"
      &&VAR&II._5="&&LAB&II._5"
      %END; %STR(;);
   DROP %DROPLIST;;
  RUN;

%MEND SLCTVAR;

%MACRO AUTO_MAPP(DATA_IN=,DATA_OUT=,VARLIST=,PCNTILE=);

  %GLOBAL VNN1;
  %PCTL(&DATA_IN,&PCNTILE,VARLIST=&VARLIST,LEFT=&LEFT_PCT,RITE=&RITE_PCT)

  DATA &DATA_OUT;
    LENGTH VARLAB VARLAB1-VARLAB5 $40;
    SET &DATA_IN;
    %DO II=1 %TO &N_VAR;
     %IF (&&VAR&II NE &BAD AND &&VAR&II NE &WEIGHT_) %THEN %DO;
      CALL LABEL(&&VAR&II,VARLAB);
      LENG=LENGTH(VARLAB);
      %LOG_FUNC(&&VAR&II,LO_LIM=&&P&II._&LEFT_PCT,
                HI_LIM=&&P&II._&RITE_PCT);
      %LET VAR&II._1=&VNN1;
      %LLG_FUNC(&&VAR&II,LO_LIM=&&P&II._&LEFT_PCT,
                HI_LIM=&&P&II._&RITE_PCT);
      %LET VAR&II._2=&VNN1;
      %EXP_FUNC(&&VAR&II,LO_LIM=&&P&II._&LEFT_PCT,
                HI_LIM=&&P&II._&RITE_PCT,B=&&P&II._&RITE_PCT);
      %LET VAR&II._3=&VNN1;
      %SQR_FUNC(&&VAR&II,LO_LIM=&&P&II._&LEFT_PCT,
                HI_LIM=&&P&II._&RITE_PCT);
      %LET VAR&II._4=&VNN1;
      %RCP_FUNC(&&VAR&II,LO_LIM=&&P&II._&LEFT_PCT,
                HI_LIM=&&P&II._&RITE_PCT);
      %LET VAR&II._5=&VNN1;
```

```
      %DO JJ=1 %TO 6;
       CALL SYMPUT("LAB&&II._&JJ",VARLAB&JJ);
      %END;
     %END;
    %END;

    %IF &DROP_ALL=1 %THEN %DO;
      DROP &VARLIST LENG VARLAB VARLAB1-VARLAB5;
    %END;
    %ELSE %DO;
     DROP LENG VARLAB VARLAB1-VARLAB5;
    %END;
  RUN;
%MEND AUTO_MAPP;

%MACRO START_mod;
 %IF &SUBV NE %THEN %DO;

   PROC MEANS DATA=&OR_DATA(WHERE=(&BAD NE .)) NOPRINT;
 %END;
              OUTPUT OUT=DAT1;
   RUN;
 DATA _NULL_;
   SET DAT1(WHERE=(_STAT_='N'));
   CALL SYMPUT('HN',_FREQ_);
 RUN;

 PROC TRANSPOSE DATA=DAT1(WHERE=(_STAT_='N' OR _STAT_='STD'))
                OUT=DAT2; RUN;

 DATA DAT2;
   SET DAT2(KEEP=_NAME_ COL1 COL2 FIRSTOBS=3);
 RUN;

 PROC SORT DATA=DAT2;
           BY _NAME_;
 RUN;
 DATA _NULL_;
   SET DAT2 END=LAST;
   FILE KEEPLIST;
     IF _N_ = 1 THEN PUT '%MACRO KEEPLIST;';
     IF COL1/&HN >= 0.10 AND COL2 > 0 THEN
     IF (_NAME_ NE "&BAD" AND _NAME_ NE "&WEIGHT_") THEN PUT _NAME_;
     IF LAST THEN PUT '%MEND KEEPLIST;';
 RUN;

 %INCLUDE KEEPLIST;
 DATA &DATA_IN;
   SET &OR_DATA(KEEP=&BAD &WEIGHT_ %KEEPLIST WHERE=(&BAD NE .));
 RUN;
 %LET PCNTILE = PCNTILE;
 %AUTO_MAPP(DATA_IN=&DATA_IN,DATA_OUT=&DATA_OUT,VARLIST=%KEEPLIST,
           PCNTILE=&PCNTILE);
 %SLCTVAR(&FDATA,&DATA_OUT,DROPLST,NUMKP=1);

%MEND START_mod;

%START_mod;
```

2: Choose the best transform variables from the pre-selected variable list using information value.

```
libname corr 'c:\output';
%let indata = out.test_out;
%let varlist = ;              /* null: plot all; o/w: plot subset of indata
                                 format: %x, x is macro name in varl */
%let dsn = &indata;             /* model data used for printing */
%let bads = perf;
%let weight_ = weight;        /* weight variable */
%let n_groups = 10;           /* number of groups for calculation & plotting
*/
%let type = 1;               /* 0: equal group;  1: non-equal group */
%let ordinal = ;              /* 1: if all variables are ordinal */
%let plot = 1;                /* 1: plotting */
%let cr = 0.6;                /* min. of correlation for plotting */
%let tf = 0.30;                /* min. of info-value for plotting  */
%let droplier = 0;            /* 1: drop the outliers in the plots */
%let suflist = %str('P','Z','J','U','Y');

%global out2;

************************************;
%macro lgt_plot(dsn,bad,varname);

%if &weight_ =  %then %do;
%if &ordinal = 1 %then %do;
data temp;
  set &dsn (keep=&bad &varname) end=last;
  retain n_bads 0;
  format odds 6.2 ;
  length odds_a $5;
  %if &low_cut ne %then %do;
    if &varname >= &low_cut;
  %end;
  %if &high_cut ne %then %do;
    if &varname <= &high_cut;
  %end;
  level=&varname;
  if &bad then n_bads=n_bads+1;
  if last then do;
     call symput('n_bads',trim(left(n_bads)));
     call symput('n_obs',trim(left(_n_)));
     odds=(n_bads)/(_n_-n_bads+0.1);
     odds_a=trim(left(odds));
     call symput('o_odds', odds_a);
  end;
  drop n_bads odds;
run;

proc means data=temp noprint;
 var &varname &bad;
 class level;
 output out=temp2 min=min min2 sum=&varname &bad max=max max2 ;
run;
%end;

%else %do;
data temp;
  set &dsn (keep=&bad &varname) end=last;
  retain n_bads 0;
  format odds 6.2 ;
  length odds_a $5;
  %if &low_cut ne %then %do;
```

```
    if &varname >= &low_cut;
  %end;
  %if &high_cut ne %then %do;
    if &varname <= &high_cut;
  %end;

  if &bad then n_bads=n_bads+1;
  if last then do;
     call symput('n_bads',trim(left(n_bads)));
     call symput('n_obs',trim(left(_n_)));
     odds=(n_bads)/(_n_-n_bads+0.1);
     odds_a=trim(left(odds));
     call symput('o_odds', odds_a);
  end;
  drop n_bads odds;
run;
/*
proc rank data=temp
            out=temp1 ties=low groups=&n_groups;
 var &varname;
 ranks level;
run;
*/

proc sql noprint;
 select count(*) into :macro1 from temp;
quit;

proc sql noprint;
 select count(&varname) into :macro1 from temp;
quit;

proc sql noprint;
 create table temp1 as
 select count(&varname) as cnt
        ,&varname
        ,count(&varname)/&macro1 as cnt_pct
 from temp
 group by &varname;
quit;

proc sort data=temp1;
by  &varname;
run;

data temp1;
 set temp1;
 if missing(&varname) then delete;
 retain cumu_pct 0 level_n 1 ;
 cumu_pct=cumu_pct+cnt_pct;
 if cumu_pct<(1/&n_groups) then do;
 level=level_n;
 end;
 if cumu_pct>=(1/&n_groups) then do;
 cumu_pct=0;
 level=level_n;
 level_n=level_n+1;
 end;

keep level &varname cnt ;
run;

proc sql noprint;
 create table temp1b as
 select sum(cnt) as cnt
```

```
        ,max(&varname) as &varname
        ,sum(cnt)/&macro1 as cnt_pct
 from temp1
 group by level;
quit;

proc means data=temp1b noprint;
 var &varname &bad;
 class level;
 output out=temp2 min=min min2 sum=&varname &bad max=max max2 ;
run;
%end;
%end;
%else %do;    /************* use weight in classification  */

%if &ordinal=1 %then %do;
data temp;
  set &dsn ( keep=&bad &varname &weight_) end=last;
  retain wt_bads wt_accu 0;
  format odds 6.2 ;
  length odds_a $5;
  %if &low_cut ne %then %do;
    if &varname >= &low_cut;
  %end;
  %if &high_cut ne %then %do;
    if &varname <= &high_cut;
  %end;
  level=&varname;

  wt_accu=wt_accu+&weight_;
  if &bad then wt_bads=wt_bads+&weight_;
  if last then do;
     call symput('n_bads',trim(left(wt_bads)));
     call symput('n_obs',trim(left(wt_accu)));
     odds=(wt_bads)/(wt_accu-wt_bads+0.1);
     odds_a=trim(left(odds));
     call symput('o_odds', odds_a);
  end;
  drop wt_bads odds;
run;

proc means data=temp noprint;
 var &varname &bad;
 weight &weight_;
 class level;
 output out=temp2 min=min min2 mean(&varname)=&varname
                  sum(&bad)=&bad max=max max2 ;
run;
%end;
%else %do;                /* THIS IS THE USUAL CASE */
data temp;
 set &dsn;
 keep &bad &varname &weight_;
run;

proc sort data=temp;
  by &varname;
run;

data temp;
  set temp(keep=&bad &varname &weight_) end=last;
  retain wt_bads wt_accu totnonm 0;
  format odds 6.2 ;
  length odds_a $5;
  %if &low_cut ne %then %do;
```

```
    if &varname >= &low_cut;
  %end;
  %if &high_cut ne %then %do;
    if &varname <= &high_cut;
  %end;
  wt_accu=wt_accu+&weight_;
  if &varname > .Z then totnonm = totnonm + 1;
  if &bad then wt_bads=wt_bads+&weight_;
  if last then do;
     call symput('n_bads',trim(left(wt_bads)));
     call symput('n_obs',trim(left(wt_accu)));
     call symput('fsum',trim(left(wt_accu/&n_groups)));
     call symput('totnonm',left(totnonm));
     call symput('ratio',left(100*totnonm/_n_));
     odds=(wt_bads)/(wt_accu-wt_bads+0.1);
     odds_a=trim(left(odds));
     call symput('o_odds', odds_a);
  end;
  drop wt_bads odds;
run;

data temp1;
  set temp;
 if &gtype = 1 then do;
  retain ttw 0;
  retain lav 1;
  ttw=ttw+&weight_;
       if _n_ = 1 then level = 1;
  else if (&varname=lag(&varname) or ttw<=&fsum) then level=lav;
  else do;
   level=lav+1;
   lav=level;
   if level > &n_groups then level = &n_groups;
   ttw=&weight_;
  end;
 end;
 else level = int(&n_groups*wt_accu/(&n_obs+.001));
  one_ = 1;
run;

proc means data=temp1 noprint;
 var &varname &bad one_;
 weight &weight_;
 class level;
 output out=temp2 min=min min2 mean(&varname)=&varname
                  sum(&bad one_)=&bad s_wt max=max max2;
run;
%end;
%end;

%let out2 = N;

data temp3;
  set temp2(firstobs=2) end=last;
    format odds 3. logodds
           log_b_g 6.2 info 7.5;
    retain tot_info 0;
    odds=(&bad+0.1)/(s_wt-&bad+0.1);
    logodds=log(odds);
    WOE=logodds;
    log_b_g=logodds;
    p_good=(s_wt-&bad+0.1)/(&n_obs-&n_bads+0.1);
    p_bad=(&bad+0.1)/(&n_bads+0.1);
    /* calculate the information value for the bin in this group */
             info=(p_good - p_bad) * (log(p_good)-log(p_bad));
```

```
                          if info = . then info = 0;
    tot_info = tot_info + info;
    if last then do;
      if tot_info <= 0.00001 then tot_info = 0.00001;

/* the total info will be usefule to get the rank of the selected variable
into the final model */
      tot_info = round(tot_info,0.00001);
      call symput('tot_info',tot_info);
    end;
    drop _type_ &bad min2 max2 tot_info level;
    if last and _n_ <= 1 then call symput('out2','Y');
run;

 data temp3;
   set temp3;
   format info_rt 4.1;
   info_rt = info/(&tot_info);
 run;

%if &out2 = Y  %then %goto out3;

 proc means data=temp3 noprint;
   var logodds &varname;
   output out=null css=logodds &varname;
 run;

 data _null_;
   set null;
   if logodds <=0.00001 or &varname <=0.00001 then
    call symput('con
dit','ILL');
   else
   call symput('condit','OK');
 run;

%if &condit=ILL %then %goto out3;
%else %do;
 proc corr data=temp3 noprint outp=corr1;
   var logodds &varname;
 run;
%end;

 data  out.corr1;
   set corr1;
      length var $32 corr_a $5;
      format corr &varname 4.2;
      if _type_='CORR' and logodds = 1;
      var="&varname";
      corr=&varname;
      info=&tot_info;
      corr_a=trim(left(corr));
      call symput("&varname",trim(left(corr_a)));
      abscorr=abs(corr);
      keep var corr info abscorr;
 run;

/* out.corr1 is the final data contains variables into the model */

%let vcos = %substr(&&&varname,1,1);
%if %quote(&vcos) = %str(-) %then %let vco = %substr(&&&varname,2,3);
                            %else %let vco = &&&varname;
%if (&plot ne 1 or &vco < &cr or &tot_info < &tf) %then %goto out3;
title  "&varname   vs.   Log(bads/goods)  Bads: &bad";
title2 "Total non-missing observations:  &totnonm / Ratio(%):  &ratio";
```

```
title3 "Correlation: &&&varname  / Information Value: &tot_info ";
title4 "Data: &dsn "
       "/ N: &n_obs / N_bads: &n_bads / Overall odds: &o_odds";

%if &droplier ne 1 %then %goto ot_liers;

%let outlier = 0;
 data temp3;
   set temp3 end=last;
   retain max_d 0;
   diff_lag=&varname-lag(&varname);
   if last=0 and max_d < diff_lag then max_d=diff_lag;
   if last and diff_lag > 3*max_d then do;
    delete;
    call symput('outlier','1');
   end;
 run;

%if &outlier ne 0 %then %do;
  title4 'The last segment deleted as the outlier';
%end;
%else %do;
  title4 ;
%end;
%ot_liers:
%out3:
%mend lgt_plot;

%macro dsn_var(dsn,prefix=dsvar,n_item=n_var,last_n=,type=1,order=2);

  %global &n_item;

              /* get the variable list from the data set  */
  proc contents data=&dsn(drop=&weight_ )
                out=ds_var noprint; run;

             /*  select the right type of variables   */
  %if &type ne %then %do;
    data ds_var;
      length lastchar $1;
      set ds_var;
        where type=&type;
        leng=length(name);
        lastchar=substr(name,leng,1);
    run;
  %end;

 /* sort by the creation order instead of the alphbetical one  */
  %if &order=1 %then %do;
    proc sort data=ds_var; by npos; run;
  %end;
 /* get the total number (of the variables) into a macro var  */
  proc contents data=ds_var out=tot noprint; run;

  data _null_;
    set tot (obs=1);
      call symput('nobs',left(nobs));
  run;

  %if &last_n ne %then %do;
    %if &nobs > &last_n %then %do;
      data ds_var;
        set ds_var;
        if _n_ ge (&nobs-&last_n+1);
      run;
```

```
      %let nobs=&last_n;
    %end;
  %end;
                  /* convert to global maroc variables    */
  %do ss=1 %to &nobs;
    %global &prefix&ss;
  %end;

  data _null_;
    set ds_var end=last;
       call symput("&prefix"||left(_n_),name);
       if last then call symput("&n_item",left(_n_));
  run;

%mend dsn_var;

*********************************************************;
%macro readlist(list,prefix=read,n_item=n_word,upcase1=1);

  /* Note: a tool:
        reads a list then labels the items in the list as macro
        variables with the specified prefix and an indexed surfix.
        Option: convert to upcase.
        For example,
          %readlist(Life is good!,prefix=miller)
        will give:
        miller1=LIFE, miller2=IS, miller3= ??_!!_$$_~~~~
        and n_word=3.
        NOTE: that the labeled macro variables are in global mode.
 */

  %local flag nn ;
  %let flag=0;
  %let nn=1;
  %global &n_item;

%do %until (&flag=1);
   %if %qscan(&list,&nn,' ')= %then %do;
     %let &n_item=%eval(&nn-1);
     %let flag=1;
   %end;
   %else %do;
     %global &prefix&nn;
     %if &upcase1=1 %then
        %let &prefix&nn=%upcase(%qscan(&list,&nn,' '));
     %else %let &prefix&nn=%qscan(&list,&nn,' ');
     %let nn=%eval(&nn+1);
   %end;
%end;

%mend readlist;
%macro plott;

  data corr.lgt_corr;
    output;
  run;

  proc datasets library=corr memtype=data;
    delete lgt_corr;
  run;

  %if &varlist = %then %do;
    %dsn_var(&dsn,prefix=var,n_item=n_var,last_n=,type=1,order=2)
  %end;
  %else %do;
```

```
    %if %substr(&varlist,1,1) = % %then %do;
    %include varl;
    %end;
    %readlist(&varlist,prefix=var,n_item=n_var)
  %end;

  %do ii=1 %to &n_var;
    %if %upcase(&bads) ne %upcase(&&var&ii) %then %do;
      %lgt_plot(&dsn,&bads,&&var&ii)
    %end;

    %if &out2 = N %then %do;
      proc append base=corr.lgt_corr data=corr1 force; run;
    %end;
  %end;

    title "Correlations with the logit of &bads "
          "and the (weakly controlled) Information Values";

    proc sort data=corr.lgt_corr(where=(info>=&tf and abscorr>=&cr));
      by descending abscorr;
    run;

    options nocenter ps=1000;
    proc printto print=pcor1 new;
    proc print data=corr.lgt_corr noobs;
    run;

%mend plott;


/* call plott macro to get the log odds of the best transform variable and
information value plot */

%plott;
```

Appendix 1 Modeling Process

- **Project Initiation**
  - **Define the problem**
  - **Choose the appropriate model type**
- **Data**
  - **Collect, Preprocess and clean data**
  - **Data reduction**
  - **Look for variable transformations**
- **Model Assessment and Validation**
  - **Does the model perform as intended?**
  - **Does the model represent and correctly reproduce the behaviors of the real world system?**

Appendix 2 Modeling Measurement

Five Measuremens: Key Principles

- **Population Stability-Consistent results (PSI <10)**
- **Accuracy-Forecast future values - False Positive and False Negative (At least < 20%).**
- **Discriminatory power – KS (at Least 45) ,GINI(at least 40) and Divergence measures the discriminatory power of a model**
- **Rank order risks – Lorenz Curve and Bad Rate Plot**
- **Differentiation vs. homogeneity-Applies to pools or ratings -Rank Test**