

Integrating PROC REG and PROC LOGISTIC for Collinearity Examination, Sample Scoring and Model Evaluation

Alec Zhixiao Lin, Loan Depot, Foothill Ranch, CA

ABSTRACT

At the final stage of regression, a modeler needs to examine the multicollinearity between model attributes, to score all sample files and to evaluate model performance. Existing options in PROC LOGISTIC and PROC REG are somewhat different for obtaining variance inflation factor (VIF), conditional index as well as for scoring sample files. This paper provides an efficient and foolproof process in SAS® that integrates those functionalities with a minimal manual handling needed. Multiple standardized summaries from the SAS output also provide valuable insights that can be shared with business peers.

INTRODUCTION

Linear regression and logistic regression have different model assumptions and outcomes, so it is understandable that SAS provides different options for PROC REG and PROC LOGISTIC. However, there are some options that one would wish to be readily available in both procedures. For example, variance inflation factor (VIF) and conditional index are commonly used to examine the collinearity between model attributes, but they are treated differently by the two procedures.

VIF AND CONDITIONAL INDEX

PROC REG has a built-in option for VIF as follows:

```
proc reg data=modeling_sample;
model y = X1 X2 ... Xn / vif tol collinooint; run;
```

PROC REG will automatically retain those attributes selected by the regression equation to compute VIF and conditional index. As VIF and conditional index are generated together with regression summaries from the above procedure, modelers can manually drop those variables with high VIF or Conditional Index and rerun the regression with ease till an acceptable level of collinearity is achieved.

Unfortunately, the same option of VIF is not available to PROC LOGISTIC. Modelers often run the following two steps to obtain VIF and conditional index for a logistic regression:

```
proc logistic data=modeling_sample descending;
model y =X1 X2 X3 ... Xn
/selection=forward stop=10 include=2 sle=0.01; run;
```

```
proc reg data=modeling_sample;
model y=X1 X2 ... Xn / vif; run;
```

As simple as the above steps look, there are two hassles: 1) Modelers need to manually copy the attributes selected by the logistic regression and paste them to PROC REG in order to run for VIF and conditional index. 2) Some model attributes could have been truncated in the SAS output (see Table 1 for an example), so they need to be manually restored to their full names before entering PROC REG for computing VIF and conditional index.

SCORING SAMPLES

In the final stage of regression, both the modeling sample and validation sample need to be scored in order to evaluate model performance. One can run the following for a logistic regression:

```
proc logistic data=modeling_sample out=chk_modeling;
model ybinary= X1 X2 ... Xn / selection=forward sle=.01;
score data=modeling_sample; run;
```

```
proc logistic data=modeling_sample out=chk_validation;
model ybinary= X1 X2 ... Xn / selection=forward sle=.01;
score data=validation_sample; run;
```

In the above example, PROC LOGISTIC uses the modeling sample to generate a regression equation and then apply it to both samples to produce two scored files.

Many modelers have wondered why the same SCORE option is not available in PROC REG. Instead, one usually runs the following for a linear regression:

```
proc reg data=modeling_sample outest=estout;
  yHat: model ycontinue=X1 X2 X3 ... Xn; run;

proc score data=modeling_sample score=estout out=chk_modeling type=parms;
  var X1 X2 X3 ... Xn; run;

proc score data=validation_sample score=estout out=chk_validation type=parms;
  var X1 X2 X3 ... Xn; run;
```

This paper introduces a process that will integrate PROC REG and PROC LOGISTIC for collinearity examination, sample scoring and model evaluation at the final stage of regression.

THE SAS PROGRAM

The main part of the suggested process consists of two SAS macros. It will automatically identify the regression type (logistic vs. linear) and invoke suitable codes accordingly. At the end of the program the following summaries will be generated for review:

- Model specifications, including parameter coefficients, results of significance tests, etc.
- VIF and conditional index that examine the collinearity between attributes selected by the model
- Model performance from multiple perspectives such as:
 - Comparison between estimated outcome and actual outcome
 - Lift over baseline
 - Incremental lift over a benchmark score
 - KS statistics for a binary outcome (for logistic regression only)
 - Tables that contain all summaries above
 - Behavior of each model attribute in relation to the target outcome
 - Score stability between the modeling sample and the validation sample

The SAS program in the appendix might look long and intimidating to some, but users do not need to digest or make any changes to the macros in order to use them. Instead, one only needs to make very limited changes to several macro statements for data specifications. The following is what the process looks like for a logistic regression (one can change PROC LOGISTIC to PROC REG to run the same process for a linear regression without changing anything inside the macros):

```
** Part I: macro for regression. No need to make any changes;
%macro runmodel;
  (Copy the program code from the appendix.)
%mend;

** Part II: macro for model evaluation. No need to make any changes;
%macro chkmodel;
  (Copy the program code from the appendix.)
%mend;

** Part III: run regression your dataset;
%let modsamp=modeling_sample;          /* modeling sample */
%let valsamp=validation_sample;        /* validation sample */
%let ytarget=usage;                    /* target variable y */
%let ylabel=Usage Rate;                /* name of target variable y */
%let yscore=usage_score;                /* score name */
%let yformat=percent8.2;                /* format of target variable y */
%let bestx=total_purchases_in_90days; /* benchmark score or predictor */
%let binnum=10;                        /* # bins for graphing. 10 for decile, 20 for twentile */
```

```

%let outfold=C:/SAS/output;                /* folder for SAS output */
proc logistic data=&modsamp descending outest=estout; /* outest a must */
model &ytarget=X1 X2 X3 ... Xn              /* all candidates for model attributes */
/selection=forward stop=8 include=2 sle=0.01;
run;
%runmodel;
%chkmodel;

```

Here are the suggested steps for running the process:

- Copy the entire program from the appendix.
- Make changes to all macro values listed above in the series of %let statements.
- Run the program from the beginning to %runmodel. One can add more options for the regression.
- Examine the regression summaries from %runmodel. The following are some rules of thumb for detecting an “undesirable” attribute in a model:
 - A low Chi-Square statistics for a logistic regression or a low F statistics for a linear regression.
 - A high VIF or a high share of contribution to conditional index.
 - A wrong sign for the coefficient estimate. This is critical to a model subject to a review by the compliance department.

Users can exclude “undesirable” variables from PROC REG or PROC LOGISTIC and re-run the regression till all model attributes look proper, such as the following:

Analysis of Maximum Likelihood Estimates					
Parameter	DF	Estimate	Standard	Wald	Pr > ChiSq
			Error	Chi-Square	
Intercept	1	0.4809	0.0631	58.1659	<.0001
total_purchases_in_9	1	0.2016	0.00447	2030.7716	<.0001
number_merchant_IDs	1	0.1179	0.0116	103.0845	<.0001
number_IP_90days	1	0.1102	0.0109	102.3681	<.0001
merchant_group_rate	1	0.1694	0.00658	663.0983	<.0001
avg_active_status_in	1	0.5098	0.0219	541.4526	<.0001
credit_limit_log	1	-0.2319	0.00745	968.7437	<.0001
credit_utilization_f	1	0.00829	0.000196	1796.5495	<.0001
pct_purchase_over_tr	1	-0.1022	0.0109	87.2667	<.0001

Table 1 – Model Specifications

Parameter Estimates							
Variable	DF	Parameter	Standard	t Value	Pr > t	Tolerance	Variance
		Estimate	Error				Inflation
Intercept	1	0.5743	0.01214	47.31	<.0001	.	0
total_purchases_in_90days	1	0.03582	0.000701	51.07	<.0001	0.55711	1.79498
number_merchant_IDs	1	0.0164	0.00205	8.01	<.0001	0.34852	2.86926
number_IP_90days	1	0.02052	0.00199	10.31	<.0001	0.37705	2.65219
merchant_group_rate	1	0.03841	0.00132	29.02	<.0001	0.61425	1.628
avg_active_status_in_6months	1	0.11847	0.00439	26.98	<.0001	0.37284	2.68211
credit_limit_log	1	-0.04516	0.00141	-31.93	<.0001	0.62254	1.60632
credit_utilization_from_bureau	1	0.00179	4.01E-05	44.51	<.0001	0.89964	1.11155
pct_purchase_over_trans_1year	1	-0.02177	0.00217	-10.03	<.0001	0.43827	2.28172

Table 2 – VIF analysis

(The table for conditional index is too long and not pasted above.)

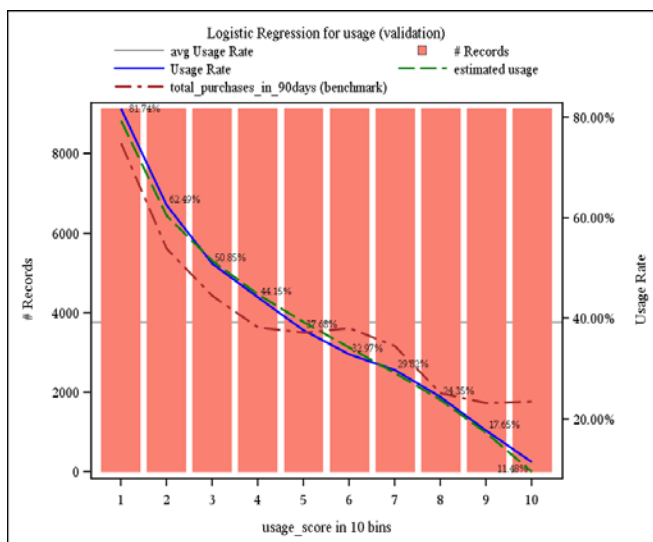
Please note that names of some model attributes are truncated in Table 1 of model specifications, but one does not need to be concerned with this. No manual “repair” is needed for the program to proceed.

- When one is satisfied with all model attributes, kick off %chkmodel. This macro will generate multiple SAS outputs for model evaluation.

SAS OUTPUTS FOR MODEL EVALUATION

Besides model specifications, modelers and business peers are usually interested in reviewing the following:

- 1) The regression equation.
- 2) Correlation between the model score and a benchmark score. A benchmark score can be – but not limited to – one of the following:
 - An old model score for predicting the same outcome.
 - Standard or most commonly used industry score, e.g., FICO score for risk prediction.
 - The most powerful numeric predictor in the model, usually the one with the highest Chi-Square Statistics for a logistic regression or the highest F Statistics for a linear regression. If you cannot decide, use your best judgment to pick the one you consider to be the most powerful.
- 3) Score performance in bins (deciles in our example) and its incremental lift over the benchmark score.

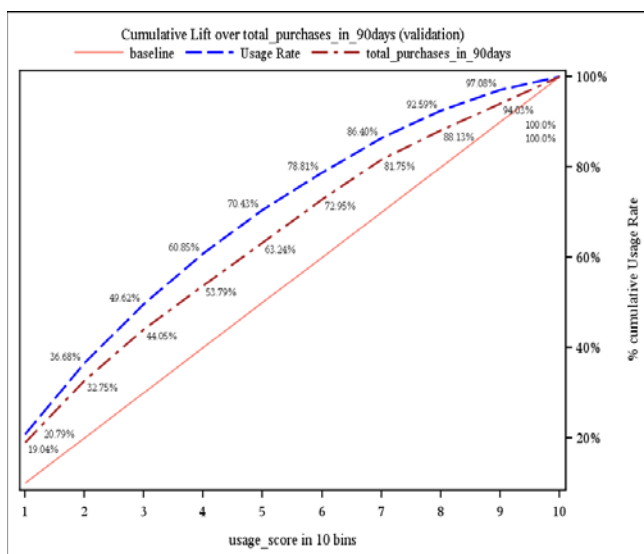


Items for review:

- The separation power of the score across deciles
- Closeness of the estimated probability to actual performance
- Incremental lift of the model score over the benchmark score

The grey line behind the distribution bars is the average usage rate.

Figure 1 – Model Performance



Items for review:

- How much lift does the score have over the baseline?
- What is the incremental lift of the model score on top of the benchmark score? This is the gain by the model.

The cumulative percentage based on the model score is not drawn here.

Figure 2 – Model Performance (lift chart)

3) Variable Behavior

A visual representation of variable behavior provides a direct view of model contents and helps to convey the story of the model effectively to business peers. The SAS process introduced in this paper automatically retains those model

attributes selected by the regression equation for graphing. We have also included the score prediction (in broken green line in Figure 3 and Figure 4), along with a reference line (in solid grey) for average usage rate in our example.

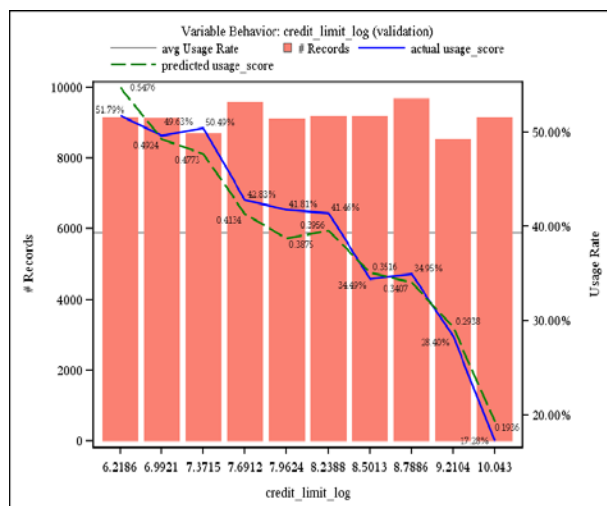


Figure 3 – Variable Behavior

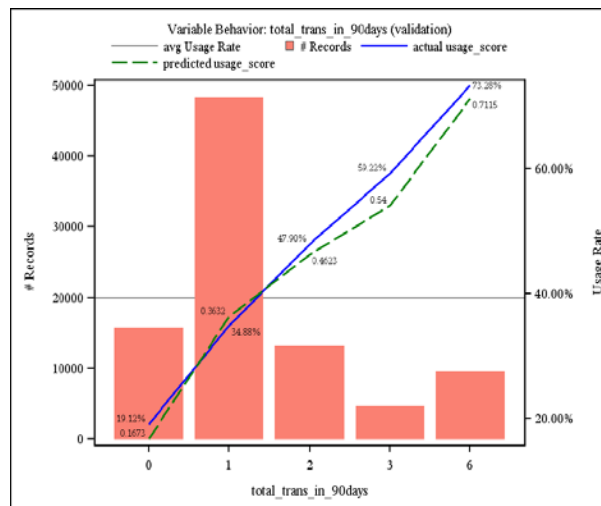


Figure 4 – Variable Behavior

The indicator beneath each distribution bar is the median value of the predictor for the associated bin.

4) Multiple summary tables on which the above graphs are based. Modelers can directly copy graphs from SAS output to a presentation deck or reproduce them in other applications. Even though most graphs shown are in deciles, one can change them into any number of bins by modifying the macro statement `%let binnum=`.

5) Comparison between the modeling sample and the validation sample for the predicted outcome. A tight closeness between the two curves usually suggests a strong stability in the score from the modeling sample to the validation sample.

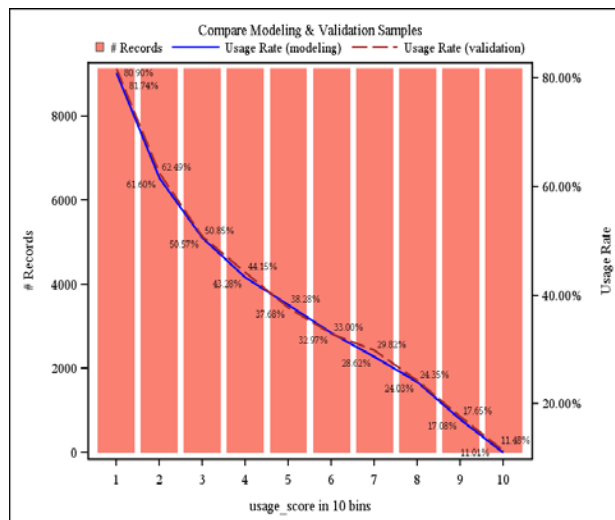


Figure 5 – Modeling vs. Validation

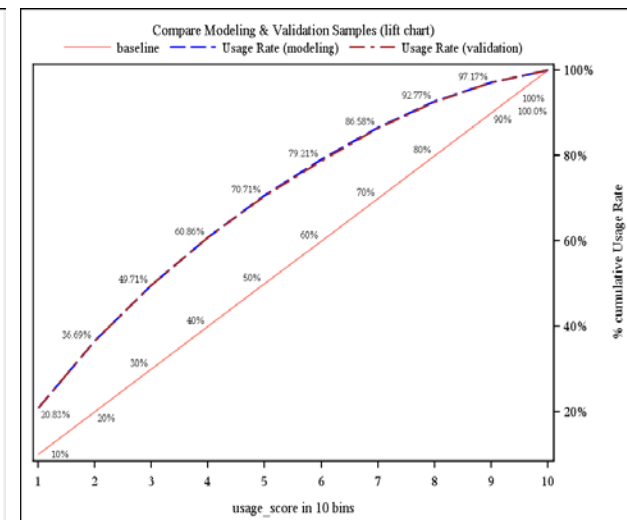


Figure 6 – Modeling vs. Validation (lift chart)

Except for graphs in 5), most of the items for model evaluation will have two sets of outputs: one based on the modeling sample and the other on the validation sample. Interested users are encouraged to add additional items for model evaluation to the process after digesting its contents.

For the graphs and summary tables, one can view them in the SAS output window or find them in the designated folder defined previously by the macro value `%let outfold=...`.

CONCLUSION

This paper introduces a process in SAS that integrates collinearity examination, sample scoring and model validation for both linear regression and logistic regression. We hope the suggested practice will expedite modeling process and generate results that can be easily communicated to business peers.

ACKNOWLEDGEMENTS

I would like to thank the team in Loan Depot led by Shawn Benner for their continuous support.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at

Alec Zhixiao Lin

VP of Modeling

Loan Depot

26642 Towne Center Drive

Foothill Ranch, CA 92610

Email: alecindc@gmail.com

Web: www.linkedin.com/pub/alec-zhixiao-lin/25/708/261/

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® Indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

APPENDIX

**** Part I: macro for regression. No need to make any changes;**

%macro runmodel;

data datatemp;

 dsid=open('estout');

 check=varnum(dsid, '_LINK_');

 if check=1 then do;

 call symputx('ytype', "Logistic");

 call symputx('forlogi', "&yscore=1/(1+exp(-1*&yscore));"); end;

 else do;

 call symputx('ytype', "Linear");

 call symputx('forlogi', " "); end;

run;

%macro logistic;

proc transpose data=estout out=estout2(drop=_label_); run;

data estout3;

length prealgo \$256.;

retain _name_ &ytargtarget prealgo;

set estout2;

if &ytargtarget=. then delete;

if _name_="_LNLIKE_" then delete;

if _name_="Intercept" then prealgo=compress(" "||put(&ytargtarget, 32.8));

else prealgo=compress("+ "||put(&ytargtarget, 32.8)||"*"||_name_); run;

%mend;

%macro linear;

proc transpose data=estout(drop=&ytargtarget _RMSE_) out=estout2(drop=_label_); run;

data estout3;

length prealgo \$256.;

set estout2;

if col1=. then delete;

if _name_="Intercept" then prealgo=compress(" "||put(col1, 32.8));

else prealgo=compress("+ "||put(col1, 32.8)||"*"||_name_); run;

%mend;

%&ytype;

proc sql noprint; select _name_ into :varmodel separated by ' '
from estout3 where _name_ ne 'Intercept'; quit;

proc sql noprint; select prealgo into :algorithm separated by ' '
from estout3; quit;

```

proc reg data=&modsamp;
model &ytarg =
&varmodel / vif tol collinoint; run;

%macro vars;
  %global varlogi yglobal ylogic varselec;
  %let varlogi=&algorithm;
  %let yglobal=&ytype;
  %let ylogic=&forlogi;
  %let varselec=&varmodel;
%mend vars;
%vars;
%mend;
** end of Part I;

** Part II: macro for model evaluation. No need to make any changes;
%macro chkmodel;
proc print data=estout3(rename=(prealgo=regression_equation)) noobs;
var regression_equation; run;
title "&yglobal Regression Equation for &ytarg"; footnote " "; run;

proc template;
define style myfont;
parent=styles.default;
style GraphFonts /
'GraphDataFont'=("Helvetica",8pt)
'GraphUnicodeFont'=("Helvetica",6pt)
'GraphValueFont'=("Helvetica",9pt)
'GraphLabelFont'=("Helvetica",12pt,bold)
'GraphFootnoteFont' = ("Helvetica",6pt,bold)
'GraphTitleFont'=("Helvetica",10pt,bold)
'GraphAnnoFont' = ("Helvetica",6pt);
end; run;
%macro evamod(inset, binnum, samp);
data inset;
set &inset(keep=&ytarg &varselec &bestx);

&yscore=&varlogi;;
&ylogic;

&yscore._r=&yscore+ranuni(123)/10000000000;
bestx_r=&bestx+ranuni(123)/10000000000; run;

proc rank data=inset groups=&binnum out=try;
var &yscore._r bestx_r;
ranks rank_yscore rank_bestx; run;

proc sql noprint; select avg(&ytarg) into :avgtarge from inset; run;

proc means data=try mean nway noprint;
class rank_yscore;
var &yscore &ytarg;
output out=summary_&samp._&binnum(drop=_type_ rename=( _freq=cnt_&samp
rank_yscore=tier))
mean=&yscore._&samp &ytarg._&samp; run;

proc means data=try mean nway noprint;
class rank_bestx;
var &ytarg;
output out=bestx_&samp._&binnum(drop=_type_ _freq_ rename=(rank_bestx=tier))
mean=y_bestx_&samp._&binnum; run;

proc corr data=inset outs=spearman_corr(keep=_type_ &yscore);
var &yscore; with &bestx; footnote " "; run;

```

```

data _null_;
set spearman_corr(where=(_type_='CORR'));
if &yscore ge 0 then call symputx('rankord', 1);
else if &yscore lt 0 then call symputx('rankord', 2); run;

%macro kickks;
%if &yglobal=Logistic %then %do;
proc npar1way data=inset
    edf noprint;
    var &yscore &bestx;
    class &ytarget;
    output out=ks101(keep= _var_ _D_ rename=(_var_=tablevar _D_=ks)); run;
proc print data=ks101; run;
%end;
%mend kickks;
%kickks;

%macro ord1;
data bestx_&samp._&binnum; set bestx_&samp._&binnum; run;
%mend;

%macro ord2;
data bestx_&samp._&binnum; set bestx_&samp._&binnum;
tier=&binnum-1-tier; run;
%mend;
%ord&rankord;

proc sort data=summary_&samp._&binnum; by tier; run;
proc sort data=bestx_&samp._&binnum; by tier; run;

data summary_&samp._&binnum;
merge summary_&samp._&binnum
    bestx_&samp._&binnum;
by tier;
tier=&binnum-tier;
format &ytarget._&samp &yformat;
informat &ytarget._&samp &yformat;
format &yscore._&samp &yformat;
informat &yscore._&samp &yformat; run;

proc sql noprint;
select sum(&ytarget._&samp), sum(cnt_&samp) into :ytotal, :totalcnt
from summary_&samp._&binnum; quit;

proc sql noprint; select avg(&ytarget) into :avgtarge from inset; quit;
proc sort data=summary_&samp._&binnum; by tier; run;

data summary_&samp._&cumu;
set summary_&samp._&binnum;
by tier;

retain cnttier predy bestxy;
predy=sum(predy, &ytarget._&samp);
bestxy=sum(bestxy, y_bestx_&samp._&binnum);
cnttier=sum(cnttier, cnt_&samp);
ypct_&samp=predy/&ytotal;
topxpct_&samp=bestxy/&ytotal;
cntpct_&samp=cnttier/&totalcnt;

score_lift=ypct_&samp-cntpct_&samp;
bestx_lift=topxpct_&samp-cntpct_&samp;

format ypct_&samp percent8.2;
format topxpct_&samp percent8.2;
format cntpct_&samp percent8.0;

informat ypct_&samp percent8.2;

```

```

informat topxpct_&samp percent8.2;
informat cntpct_&samp percent8.0;
run;

proc sql noprint;
select (max(score_lift)-max(bestx_lift))/max(bestx_lift)*100 into :incrvl from
summary_&samp._cumu; quit;

ods pdf file="%outfold/performance_by_&samp._sample.pdf";
proc sgplot data=summary_&samp._&binnum;
refline &avgtarge / axis=y2 name="avg" lineattrs=(color=grey) legendlabel="avg
&ylabel";
vbar tier / response=cnt_&samp nostatlabel nooutline fillattrs=(color="salmon");
vline tier / response=&ytarge._&samp datalabel y2axis lineattrs=(color="blue"
thickness=2) nostatlabel;
vline tier / response=&yscore._&samp y2axis lineattrs=(color="green" thickness=2)
nostatlabel;
vline tier / response=y_bestx_&samp._&binnum y2axis lineattrs=(color="brown"
thickness=2) nostatlabel;
label cnt_&samp="# Records";
label &ytarge._&samp="&ylabel";
label tier="&yscore in &binnum bins";
label y_bestx_&samp._&binnum="&bestx (benchmark)";
label &yscore._&samp="estimated &ytarge";
format &ytarge._&samp &yformat;
keylegend / location = outside
position = top noborder
title="&yglobal Regression for &ytarge (&samp)"; run;

proc sgplot data=summary_&samp._cumu;
y2axis label="% cumulative &ylabel";
vline tier / response=cntpct_&samp y2axis lineattrs=(color="salmon") nostatlabel;
vline tier / response=ypct_&samp datalabel y2axis lineattrs=(color="blue" thickness=2)
nostatlabel;
vline tier / response=topxpct_&samp datalabel y2axis lineattrs=(color="brown"
thickness=2) nostatlabel;
label cntpct_&samp="baseline";
label &ytarge._&samp="cumulative &ylabel";
label tier="&yscore in &binnum bins";
label ypct_&samp="&ylabel";
label topxpct_&samp="&bestx";
label y_bestx_&samp._&binnum="&bestx";
format &ytarge._&samp &yformat;
keylegend / location = outside
position = top noborder
title="Cumulative Lift over &bestx (&samp)";
footnote height=3 justify=center
"incremental value over &bestx: &incrvl.%"; run;
ods pdf close;

proc sort data=summary_&samp._cumu; by tier; run;
proc sql; create table vcnt as select count(*) as vcnt from estout3
where _name_ ne "Intercept"; quit;

data _null_; set vcnt; call symputx('varcnt', vcnt); run;
proc sql noprint;
select _name_ into :x1-:x&varcnt from estout3 where _name_ ne "Intercept"; quit;

** examine variable behavior;
%macro varbehav(ranknum, invar);
proc rank data=inset groups=&binnum out=trysum; var &varselec; ranks rank1-
rank&varcnt; run;

proc means data=trysum mean median nway noprint;
class rank&ranknum;
var &yscore &ytarge;
output out=&invar(drop=_type_ rename=(rank&ranknum=rank))

```

```

mean(&yscore &ytargt)=predicted actual
median(&invar)=x_median; run;

data &invar; set &invar; varname="&invar"; run;
data trygraph&ranknum;
set &invar;
rename x_median=&invar;
format actual &yformat;
informat actual &yformat; run;

proc sgplot data=trygraph&ranknum;
y2axis label="&ylabel";
refline &avgtarget / axis=y2 name="avgy" lineattrs=(color=grey) legendlabel="avg
&ylabel";
vbar &invar / response=freq_ nostatlabel nooutline fillattrs=(color="salmon");
vline &invar / response=actual datalabel y2axis lineattrs=(color="blue" thickness=2)
nostatlabel;
vline &invar / response=predicted datalabel y2axis lineattrs=(color="green"
thickness=2) nostatlabel;
label _freq_="# Records";
label predicted="predicted &yscore";
label actual="actual &yscore";
keylegend / location = outside
position = top noborder
title="Variable Behavior: &invar (&samp)";
footnote " "; run;
%mend varbehav;

%macro allgraph;
ods pdf file="&outfold/variable_behavior_&samp..pdf";
%do i=1 %to &varcnt;
%varbehav(&i, &&x&i);
%end;
%mend allgraph;
%allgraph;
ods pdf close;
%mend evamod;
%evamod(&modsamp, &binnum, modeling);
%evamod(&valsamp, &binnum, validation);

data summary_&binnum._cumu;
merge summary_modeling_cumu summary_validation_cumu;
by tier; run;

data summary_&binnum;
merge summary_modeling_&binnum summary_validation_&binnum;
by tier; run;

%macro to_excel(data_sum);
PROC EXPORT DATA=&data_sum OUTFILE="&outfold/&data_sum" label DBMS=tab REPLACE; run;
%mend;
%to_excel(summary_&binnum);
%to_excel(summary_&binnum._cumu);

ods pdf file="&outfold/compare_modeling_validation.pdf";
proc sgplot data=summary_&binnum;
y2axis label="&ylabel";
vbar tier / response=cnt_validation nostatlabel nooutline fillattrs=(color="salmon");
vline tier / response=&ytargt._modeling datalabel y2axis lineattrs=(color="blue"
thickness=2) nostatlabel;
vline tier / response=&ytargt._validation datalabel y2axis lineattrs=(color="brown"
thickness=2) nostatlabel;
label cnt_validation="# Records";
label &ytargt._modeling="&ylabel (modeling)";
label &ytargt._validation="&ylabel (validation)";
label tier="&yscore in &binnum bins";
format &ytargt._modeling &yformat; format &ytargt._validation &yformat;

```

```

keylegend / location = outside
position = top noborder
title="Compare Modeling & Validation Samples";
footnote height=3 justify=center "The distribution is based on the validation
sample."; run;

proc sgplot data=summary_&binnum._cumu;
y2axis label="% cumulative &ylabel";
vline tier / response=cntpct_validation datalabel y2axis lineattrs=(color="salmon")
nostatlabel;
vline tier / response=ypct_modeling datalabel y2axis lineattrs=(color="blue"
thickness=2) nostatlabel;
vline tier / response=ypct_validation y2axis lineattrs=(color="brown" thickness=2)
nostatlabel;
label cntpct_validation="baseline";
label tier="&yscore in &binnum bins";
label ypct_modeling="&ylabel (modeling)";
label ypct_validation="&ylabel (validation)";
format ypct_modeling &yformat; format ypct_validation &yformat;
keylegend / location = outside
position = top noborder
title="Compare Modeling & Validation Samples (lift chart)";
footnote "The distribution is based on the validation sample."; run;
ods pdf close;

proc print data=summary_&binnum._cumu; title "Model Summary"; run;
%mend;
** end of Part II;

** Part III: Analyse your data set;
** 1) Make changes to data specifications below;
** 2) After making changes, run from the beginning of the program to
** %runmodel. Delete "undesirable" variables and rerun the regression.
** 3) After finalizing the model, run %chkmodel for model evaluation;
** 4) Most summaries can also be found in the output folder;
%let modsamp=modeling_sample; /* modeling sample */
%let valsamp=validation_sample; /* validation sample */
%let ytarget=usage; /* target variable y */
%let yscore=usage_score; /* score name */
%let ylabel=Usage Rate; /* label for target variable y */
%let yformat=percent8.2; /* format of target variable y */
%let bestx=total_purchases_in_90days; /* benchmark score or predictor */
%let binnum=10; /* # bins for graphing. 10 for decile, 20 for twentile */
%let outfold=C:/SAS/output; /* folder for SAS output */
proc logistic data=&modsamp descending outest=estout; /* outest a must */
model &ytarget=X1 X2 X3 ... Xn /* all candidates for model attributes */
/selection=forward stop=8 include=2 sle=0.01;
run;
%runmodel;
%chkmodel;

```