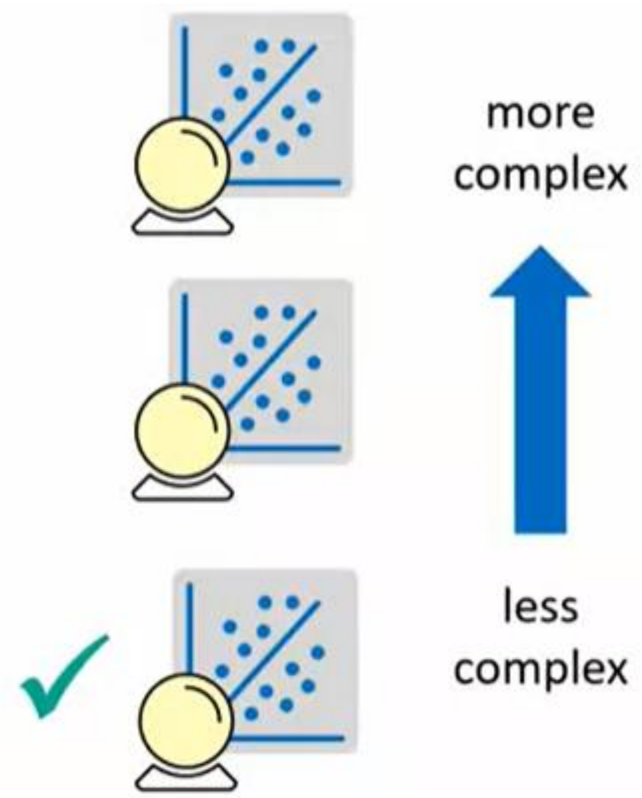
SBA Statistical Business Analyst using SAS

SBA3 Predictive Modeling with Logistic Regression

W5E Model Selection Plots

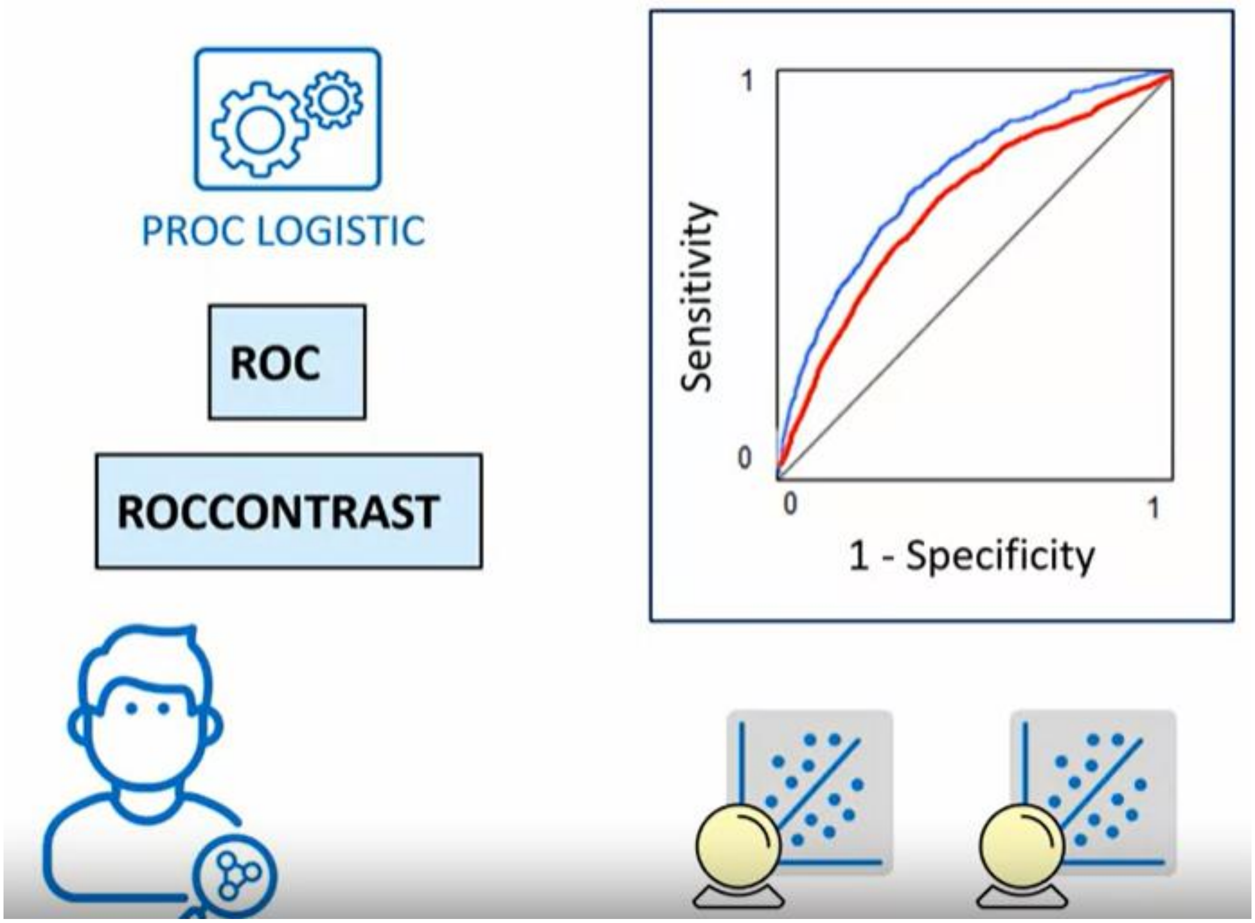
Introduction



In this topic, you learn to do the following:

- use the ROC and ROCCONTRAST statements in PROC LOGISTIC
- use the ASSESS and FITANDSCORE macros to generate and evaluate many models

Comparing ROC Curves of several Models"



ROC *<'label'> <specification> </ option(s)>*;
ROCCONTRAST *<'label'> <contrast> </ option(s)>*;

```
proc logistic data=work.sco_validate;
   model ins(event='1')=p_ch2 p_sel / nofit;
   roc "Chapter 2 Model" p_ch2;
   roc "Chapter 4 Model" p_sel;
   roccontrast "Comparing the Two Models";
run;
```

```
PRED=variable
```

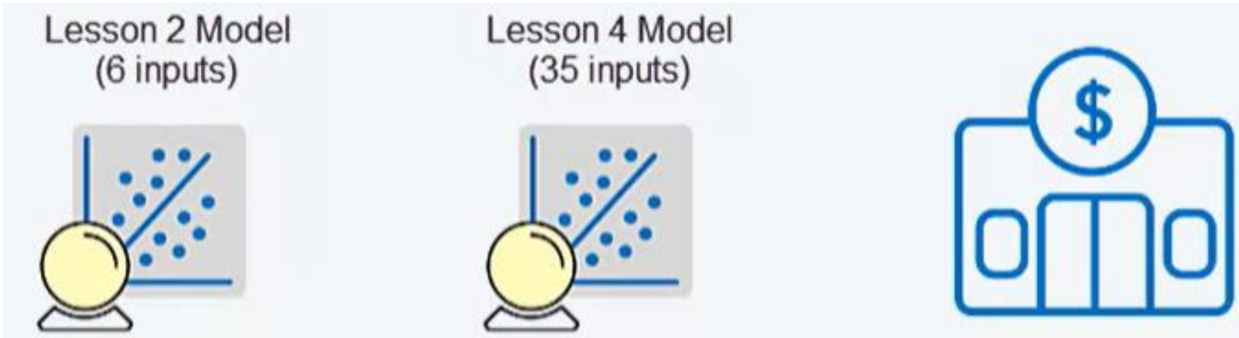
```
ROC <'label'> <specification> </ option(s)>;
ROCCONTRAST <'label'> <contrast> </ option(s)>;
```

```
proc logistic data=work.sco_validate;
   model ins(event='1')=p_ch2 p_sel / nofit;
   roc "Chapter 2 Model" p_ch2;
   roc "Chapter 4 Model" p_sel;
   roccontrast "Comparing the Two Models";
run;
```

```
ROC <'label'> <specification> </ option(s)>;
ROCCONTRAST <'label'> <contrast> </ option(s)>;
```

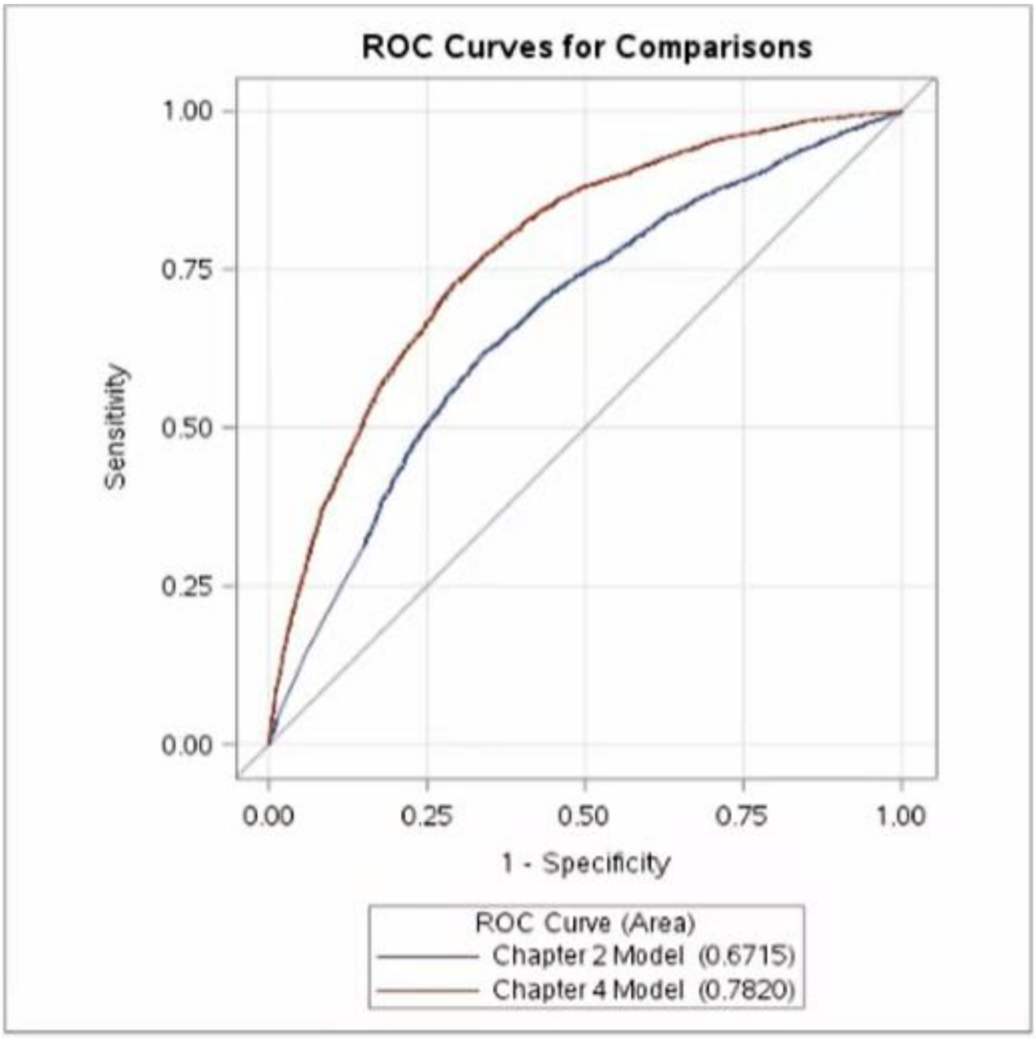
```
proc logistic data=work.sco_validate;
   model ins(event='1')=p_ch2 p_sel / nofit;
   roc "Chapter 2 Model" p_ch2;
   roc "Chapter 4 Model" p_sel;
   roccontrast "Comparing the Two Models";
run;
```

Demo Comparing ROC Curves to Measure Model Performance



pmlr04d05.sas

```
proc logistic data=work.train_imputed_swoe_bins noprint;
   class res;
   model ins(event='1')=dda ddabal dep depamt checks res;
   score data=work.valid_imputed_swoe_bins
         out=work.sco_validate(rename=(p_1=p_ch2));
run;

proc logistic data=work.train_imputed_swoe_bins noprint;
   model ins(event='1')=&selected;
   score data=work.sco_validate out=work.sco_validate(rename=(p_1=p_sel));
run;

title1 "Validation Data Set Performance";
ods select ROCOverlay ROCAssociation ROCContrastTest;
proc logistic data=work.sco_validate;
   model ins(event='1')=p_ch2 p_sel / nofit;
   roc "Chapter 2 Model" p_ch2;
   roc "Chapter 4 Model" p_sel;
   roccontrast "Comparing the Two Models";
run;
```



ROC Association Statistics							
	Mann-Whitney						
ROC Model	Area	Standard Error	95% Wald Confidence Limits		Somers' D	Gamma	Tau-a
Chapter 2 Model	0.6715	0.00543	0.6609	0.6821	0.3430	0.3475	0.1553
Chapter 4 Model	0.7820	0.00456	0.7730	0.7909	0.5639	0.5640	0.2554

ROC Contrast Test Results			
Contrast	DF	Chi-Square	Pr > ChiSq
Comparing the Two Models	1	528.4259	<.0001

/* Code for the Lesson 1, 2 and 3 Demonstrations in the SAS e-Course

```
  "Predictive Modeling Using Logistic Regression" */

/* The demonstrations in this SAS e-course build on each
other. This file contains the code for all demonstrations in
Lesson 1, 2 and 3.

If you started a new SAS session since you ran the previous
demonstration(s), you need to set up access to the course
files (see the Course Overview and Data Setup) and then and
re-run the code for all previous demonstrations. The title of
each demonstration and the corresponding program file name
appear in a comment above the code for that demo.

Before you submit the code, make any necessary modifications to
the code, if indicated in comments.
Note: Most of the code requires no modifications.

Submit the code and check the log to verify that it ran without errors.

After performing the steps above, you are ready to proceed with the
current demonstration!
*/


/* ====================================================== */
/* Lesson 1, Section 1: l1d1.sas
  Demonstration: Examining the Code for Generating
  Descriptive Statistics and Frequency Tables
  [m641_1_i; derived from pmlr01d01.sas]            */
/* ====================================================== */
```

```
data work.develop;
  set pmlr.develop;
run;

%global inputs;
%let inputs=ACCTAGE DDA DDABAL DEP DEPAMT CASHBK
        CHECKS DIRDEP NSF NSFAMT PHONE TELLER
        SAV SAVBAL ATM ATMAMT POS POSAMT CD
        CDBAL IRA IRABAL LOC LOCBAL INV
        INVBAL ILS ILSBAL MM MMBAL MMCRED MTG
        MTGBAL CC CCBAL CCPURC SDB INCOME
        HMOWN LORES HMVAL AGE CRSCORE MOVED
        INAREA;

proc means data=work.develop n nmiss mean min max;
  var &inputs;
run;

proc freq data=work.develop;
  tables ins branch res;
run;

/* ===================================================== */
/* Lesson 1, Section 2: l1d2.sas
  Demonstration: Splitting the Data
  [m641_2_h; derived from pmlr01d02.sas]          */
/* ===================================================== */
```

```
/* Sort the data by the target in preparation for stratified sampling. */

proc sort data=work.develop out=work.develop_sort;
  by ins;
run;

/* The SURVEYSELECT procedure will perform stratified sampling
  on any variable in the STRATA statement. The OUTALL option
  specifies that you want a flag appended to the file to
  indicate selected records, not simply a file comprised
  of the selected records. */

proc surveyselect noprint data=work.develop_sort
            samprate=.6667 stratumseed=restore
            out=work.develop_sample
            seed=44444 outall;
  strata ins;
run;

/* Verify stratification. */

proc freq data=work.develop_sample;
  tables ins*selected;
run;

/* Create training and validation data sets. */

data work.train(drop=selected SelectionProb SamplingWeight)
```

```
    work.valid(drop=selected SelectionProb SamplingWeight);
   set work.develop_sample;
   if selected then output work.train;
   else output work.valid;
run;



/* ===================================================== */
/* Lesson 2, Section 1: l2d1.sas
   Demonstration: Fitting a Basic Logistic Regression Model,
   Parts 1 and 2
   [m642_1_k1, m642_1_k2; derived from pmlr02d01.sas]    */
/* ===================================================== */

title1 "Logistic Regression Model for the Variable Annuity Data Set";
proc logistic data=work.train
          plots(only maxpoints=none)=(effect(clband x=(ddabal depamt checks res))
          oddsratio (type=horizontalstat));
   class res (param=ref ref='S') dda (param=ref ref='0');
   model ins(event='1')=dda ddabal dep depamt
           cashbk checks res / stb clodds=pl;
   units ddabal=1000 depamt=1000 / default=1;
   oddsratio 'Comparisons of Residential Classification' res / diff=all cl=pl;
   effectplot slicefit(sliceby=dda x=ddabal) / noobs;
   effectplot slicefit(sliceby=dda x=depamt) / noobs;
run;
title1;
```

```
/* ===================================================== */
/* Lesson 2, Section 1: l2d2.sas
   Demonstration: Scoring New Cases
   [m642_1_n; derived from pmlr02d02.sas]              */
/* ===================================================== */

/* Score a new data set with one run of the LOGISTIC procedure with the
   SCORE statement.  */

proc logistic data=work.train noprint;
   class res (param=ref ref='S');
   model ins(event='1')= res dda ddabal dep depamt cashbk checks;
   score data = pmlr.new out=work.scored1;
run;

title1 "Predicted Probabilities from Scored Data Set";
proc print data=work.scored1(obs=10);
   var p_1 dda ddabal dep depamt cashbk checks res;
run;

title1 "Mean of Predicted Probabilities from Scored Data Set";
proc means data=work.scored1 mean nolabels;
   var p_1;
run;

/* Score a new data set with the OUTMODEL= amd INMODEL= options */

proc logistic data=work.train outmodel=work.scoredata noprint;
```

```
  class res (param=ref ref='S');
  model ins(event='1')= res dda ddabal dep depamt cashbk checks;
run;

proc logistic inmodel=work.scoredata noprint;
  score data = pmlr.new out=work.scored2;
run;

title1 "Predicted Probabilities from Scored Data Set";
proc print data=work.scored2(obs=10);
  var p_1 dda ddabal dep depamt cashbk checks res;
run;

/* Score a new data set with the CODE Statement  */

proc logistic data=work.train noprint;
  class res (param=ref ref='S');
  model ins(event='1')= res dda ddabal dep depamt cashbk checks;
  code file="&PMLRfolder/pmlr_score.txt";
run;

data work.scored3;
  set pmlr.new;
  %include "&PMLRfolder/pmlr_score.txt";
run;

title1 "Predicted Probabilities from Scored Data Set";
proc print data=work.scored3(obs=10);
  var p_ins1 dda ddabal dep depamt cashbk checks res;
```

```
run;
title1 ;


/* ======================================================== */
/* Lesson 2, Section 2: l2d3.sas
   Demonstration: Correcting for Oversampling
   [m642_2_f; derived from pmlr02d03.sas]              */
/* ======================================================== */

/* Specify the prior probability to correct for oversampling.  */
%global pi1;
%let pi1=.02;

/* Correct predicted probabilities */

proc logistic data=work.train noprint;
   class res (param=ref ref='S');
   model ins(event='1')=dda ddabal dep depamt cashbk checks res;
   score data=pmlr.new out=work.scored4 priorevent=&pi1;
run;

title1 "Adjusted Predicted Probabilities from Scored Data Set";
proc print data=work.scored4(obs=10);
   var p_1 dda ddabal dep depamt cashbk checks res;
run;

title1 "Mean of Adjusted Predicted Probabilities from Scored Data Set";
proc means data=work.scored4 mean nolabels;
```

```
  var p_1;
run;
title1 ;

/* Correct probabilities in the Score Code */

proc logistic data=work.train noprint;
  class res (param=ref ref='S');
  model ins(event='1')=dda ddabal dep depamt cashbk checks res;
  /* File suffix "txt" is used so you can view the file */
  /* with a native text editor. SAS prefers "sas", but  */
  /* when specified as a filename, SAS does not care.   */
  code file="&PMLRfolder/pmlr_score_adj.txt";
run;

%global rho1;
proc SQL noprint;
  select mean(INS) into :rho1
  from work.train;
quit;

data new;
  set pmlr.new;
  off=log(((1-&pi1)*&rho1)/(&pi1*(1-&rho1)));
run;

data work.scored5;
  set work.new;
  %include "&PMLRfolder/pmlr_score_adj.txt";
```

```
  eta=log(p_ins1/p_ins0) - off;
  prob=1/(1+exp(-eta));
run;

title1 "Adjusted Predicted Probabilities from Scored Data Set";
proc print data=scored5(obs=10);
  var prob dda ddabal dep depamt cashbk checks res;
run;
title1 ;




/* ===================================================== */
/* Lesson 3, Section 1: l3d1.sas
  Demonstration: Imputing Missing Values
  [m643_1_h; derived from pmlr03d01.sas]              */
/* ===================================================== */

title1 "Variables with Missing Values";
proc print data=work.train(obs=15);
  var ccbal ccpurc income hmown;
run;
title1 ;

/* Create missing indicators */
data work.train_mi(drop=i);
  set work.train;
  /* name the missing indicator variables */
  array mi{*} MIAcctAg MIPhone MIPOS MIPOSAmt
```

```
            MIInv MIInvBal MICC MICCBal
            MICCPurc MIIncome MIHMOwn MILORes
            MIHMVal MIAge MICRScor;
  /* select variables with missing values */
  array x{*} acctage phone pos posamt
           inv invbal cc ccbal
           ccpurc income hmown lores
           hmval age crscore;
  do i=1 to dim(mi);
    mi{i}=(x{i}=.);
    nummiss+mi{i};
  end;
run;

/* Impute missing values with the median */
proc stdize data=work.train_mi reponly method=median out=work.train_imputed;
  var &inputs;
run;

title1 "Imputed Values with Missing Indicators";
proc print data=work.train_imputed(obs=12);
  var ccbal miccbal ccpurc miccpurc income miincome hmown mihmown nummiss;
run;
title1 ;



/* =================================================== */
/* Lesson 3, Section 2: l3d2a.sas
  Demonstration: Collapsing the Levels of a Nominal Input,
```

```
  Part 1
  [m643_2_g1; derived from pmlr03d02.sas]              */
/* ====================================================== */

proc means data=work.train_imputed noprint nway;
  class branch;
  var ins;
  output out=work.level mean=prop;
run;

title1 "Proportion of Events by Level";
proc print data=work.level;
run;

/* Use ODS to output the ClusterHistory output object into a data set
  named "cluster." */

ods output clusterhistory=work.cluster;

proc cluster data=work.level method=ward outtree=work.fortree
     plots=(dendrogram(vertical height=rsq));
  freq _freq_;
  var prop;
  id branch;
run;


/* ====================================================== */
/* Lesson 3, Section 2: l3d2b.sas
```

```
  Demonstration: Collapsing the Levels of a Nominal Input,
  Part 2
  [m643_2_g2; derived from pmlr03d02.sas]            */
/* ===================================================== */

/* Use the FREQ procedure to get the Pearson Chi^2 statistic of the
  full BRANCH*INS table. */

proc freq data=work.train_imputed noprint;
  tables branch*ins / chisq;
  output out=work.chi(keep=_pchi_) chisq;
run;

/* Use a one-to-many merge to put the Chi^2 statistic onto the clustering
  results. Calculate a (log) p-value for each level of clustering. */

data work.cutoff;
  if _n_=1 then set work.chi;
  set work.cluster;
  chisquare=_pchi_*rsquared;
  degfree=numberofclusters-1;
  logpvalue=logsdf('CHISQ',chisquare,degfree);
run;

/* Plot the log p-values against number of clusters. */

title1 "Plot of the Log of the P-Value by Number of Clusters";
proc sgplot data=work.cutoff;
  scatter y=logpvalue x=numberofclusters
```

```
        / markerattrs=(color=blue symbol=circlefilled);
  xaxis label="Number of Clusters";
  yaxis label="Log of P-Value" min=-120 max=-85;
run;
title1 ;

/* Create a macro variable (&ncl) that contains the number of clusters
  associated with the minimum log p-value. */

proc sql;
  select NumberOfClusters into :ncl
  from work.cutoff
  having logpvalue=min(logpvalue);
quit;

proc tree data=work.fortree nclusters=&ncl out=work.clus noprint;
  id branch;
run;

proc sort data=work.clus;
  by clusname;
run;

title1 "Levels of Branch by Cluster";
proc print data=work.clus;
  by clusname;
  id clusname;
run;
title1 ;
```

```
/* The DATA Step creates the scoring code to assign the branches to a cluster. */

filename brclus "&PMLRfolder/branch_clus.sas";

data _null_;
  file brclus;
  set work.clus end=last;
  if _n_=1 then put "select (branch);";
  put "  when ('" branch +(-1) "') branch_clus = '" cluster +(-1) "';";
  if last then do;
    put "  otherwise branch_clus = 'U';" / "end;";
  end;
run;

data work.train_imputed_greenacre;
  set work.train_imputed;
  %include brclus / source2;
run;



/* ===================================================== */
/* Lesson 3, Section 2: l3d3.sas
  Demonstration: Computing the Smoothed Weight of Evidence
  [m643_2_j; derived from pmlr03d03.sas]              */
/* ===================================================== */

/* Rho1 is the proportion of events in the training data set. */
%global rho1;
```

```
proc sql noprint;
  select mean(ins) into :rho1
  from work.train_imputed;
run;

/* The output data set from PROC MEANS will have the number of
  observations and events for each level of branch. */

proc means data=work.train_imputed sum nway noprint;
  class branch;
  var ins;
  output out=work.counts sum=events;
run;

/* The DATA Step creates the scoring code that assigns each branch to
  a value of the smoothed weight of evidence. */


filename brswoe "&PMLRfolder/swoe_branch.sas";

data _null_;
  file brswoe;
  set work.counts end=last;
  logit=log((events + &rho1*24)/(_FREQ_ - events + (1-&rho1)*24));
  if _n_=1 then put "select (branch);" ;
  put "  when ('" branch +(-1) "') branch_swoe = " logit ";" ;
  if last then do;
  logit=log(&rho1/(1-&rho1));
  put "  otherwise branch_swoe = " logit ";" / "end;";
```

```
  end;
run;

data work.train_imputed_swoe;
  set work.train_imputed;
  %include brswoe / source2;
run;



/* ====================================================== */
/* Lesson 3, Section 3: l3d4.sas
  Demonstration: Reducing Redundancy by Clustering Variables
  [m643_3_i; derived from pmlr03d04.sas]              */
/* ====================================================== */

/* Use the ODS OUTPUT statement to generate data sets based on the variable
  clustering results and the clustering summary. */

ods select none;
ods output clusterquality=work.summary
        rsquare=work.clusters;

proc varclus data=work.train_imputed_swoe maxeigen=.7 hi;
  var &inputs branch_swoe miacctag
     miphone mipos miposamt miinv
     miinvbal micc miccbal miccpurc
     miincome mihmown milores mihmval
     miage micrscor;
run;
```

```
ods select all;

/* Use the CALL SYMPUT function to create a macro variable:&NVAR =
   the number of of clusters. This is also the number of variables
   in the analysis, going forward. */

%global nvar;
data _null_;
   set work.summary;
   call symput('nvar',compress(NumberOfClusters));
run;

title1 "Variables by Cluster";
proc print data=work.clusters noobs label split='*';
   where NumberOfClusters=&nvar;
   var Cluster Variable RSquareRatio VariableLabel;
   label RSquareRatio="1 - RSquare*Ratio";
run;
title1 ;

title1 "Variation Explained by Clusters";
proc print data=work.summary label;
run;

/* Choose a representative from each cluster.  */
%global reduced;
%let reduced=branch_swoe MIINCOME Dep CCBal MM Income ILS POS NSF CD
         DDA LOC Age Inv InArea AcctAge Moved CRScore MICRScor
         IRABal MIAcctAg SavBal CashBk DDABal SDB InvBal CCPurc
```

```
          ATMAmt Sav CC Phone HMOwn DepAmt IRA MTG ATM LORes;


/* ===================================================== */
/* Lesson 3, Section 4:  l3d5a.sas
   Demonstration: Performing Variable Screening, Part 1
   [m643_4_e1; derived from pmlr03d05.sas]             */
/* ===================================================== */

ods select none;
ods output spearmancorr=work.spearman
         hoeffdingcorr=work.hoeffding;

proc corr data=work.train_imputed_swoe spearman hoeffding;
   var ins;
   with &reduced;
run;

ods select all;

proc sort data=work.spearman;
    by variable;
run;

proc sort data=work.hoeffding;
    by variable;
run;

data work.correlations;
```

```
  merge work.spearman(rename=(ins=scorr pins=spvalue))
        work.hoeffding(rename=(ins=hcorr pins=hpvalue));
  by variable;
  scorr_abs=abs(scorr);
  hcorr_abs=abs(hcorr);
run;

proc rank data=work.correlations out=work.correlations1 descending;
   var scorr_abs hcorr_abs;
   ranks ranksp rankho;
run;

proc sort data=work.correlations1;
  by ranksp;
run;

title1 "Rank of Spearman Correlations and Hoeffding Correlations";
proc print data=work.correlations1 label split='*';
  var variable ranksp rankho scorr spvalue hcorr hpvalue;
  label ranksp ='Spearman rank*of variables'
        scorr  ='Spearman Correlation'
        spvalue='Spearman p-value'
        rankho ='Hoeffding rank*of variables'
        hcorr  ='Hoeffding Correlation'
        hpvalue='Hoeffding p-value';
run;


/* ======================================================= */
```

```
/* Lesson 3, Section 4: l3d5b.sas
   Demonstration: Performing Variable Screening, Part 2
   [m643_4_e2; derived from pmlr03d05.sas]             */
/* ===================================================== */

/* Find values for reference lines */
%global vref href;
proc sql noprint;
   select min(ranksp) into :vref
   from (select ranksp
   from work.correlations1
   having spvalue > .5);

   select min(rankho) into :href
   from (select rankho
   from work.correlations1
   having hpvalue > .5);
quit;

/* Plot variable names, Hoeffding ranks, and Spearman ranks. */

title1 "Scatter Plot of the Ranks of Spearman vs. Hoeffding";
proc sgplot data=work.correlations1;
   refline &vref / axis=y;
   refline &href / axis=x;
   scatter y=ranksp x=rankho / datalabel=variable;
   yaxis label="Rank of Spearman";
   xaxis label="Rank of Hoeffding";
run;
```

```
title1 ;

%global screened;
%let screened=SavBal Dep DDA CD Sav CC ATM MM branch_swoe Phone IRA IRABal
        DDABal ATMAmt ILS POS NSF CCPurc SDB DepAmt CCBal Inv InArea
        Age CashBk MICRScor Income;


/* ===================================================== */
/* Lesson 3, Section 4: l3d6.sas
   Demonstration: Creating Empirical Logit Plots
   [m643_4_i; derived from pmlr03d06.sas]          */
/* ===================================================== */

%global var;
%let var=DDABal;

/* Group the data by the variable of interest in order to create
   empirical logit plots.   */

proc rank data=work.train_imputed_swoe groups=100 out=work.ranks;
   var &var;
   ranks bin;
run;

title1 "Checking Account Balance by Bin";
proc print data=work.ranks(obs=10);
   var &var bin;
run;
```

```
/* The data set BINS will contain:INS=the count of successes in each bin,
   _FREQ_=the count of trials in each bin, DDABAL=the avg DDABAL in each bin. */

proc means data=work.ranks noprint nway;
   class bin;
   var ins &var;
   output out=work.bins sum(ins)=ins mean(&var)=&var;
run;

title1 "Number of Observations, Events, and Average Checking Account Balance by Bin";
proc print data=work.bins(obs=10);
run;

/* Calculate the empirical logit */

data work.bins;
   set work.bins;
   elogit=log((ins+(sqrt(_FREQ_ )/2))/
         ( _FREQ_ -ins+(sqrt(_FREQ_ )/2)));
run;

title1 "Empirical Logit against &var";
proc sgplot data=work.bins;
   reg y=elogit x=&var /
      curvelabel="Linear Relationship?"
      curvelabelloc=outside
      lineattrs=(color=ligr);
   series y=elogit x=&var;
```

```
run;

title1 "Empirical Logit against Binned &var";
proc sgplot data=work.bins;
  reg y=elogit x=bin /
     curvelabel="Linear Relationship?"
     curvelabelloc=outside
     lineattrs=(color=ligr);
  series y=elogit x=bin;
run;


/* ===================================================== */
/* Lesson 3, Section 4: l3d7a.sas
  Demonstration: Accommodating a Nonlinear Relationship,
  Part 1
  [m643_4_m1; derived from pmlr03d07.sas]             */
/* ===================================================== */

title1 "Checking Account Balance and INS by Checking Account";
proc means data=work.train_imputed_swoe mean median min max;
  class dda;
  var ddabal ins;
run;

/* A possible remedy for that non-linearity is to replace the logical
  imputation of 0 for non-DDA customers with the mean. */

%global mean;
```

```
proc sql noprint;
  select mean(ddabal) into :mean
  from work.train_imputed_swoe where dda;
quit;

data work.train_imputed_swoe_dda;
  set work.train_imputed_swoe;
  if not dda then ddabal=&mean;
run;

/* Create new logit plots */
%global var;
%let var=DDABal;

proc rank data=work.train_imputed_swoe_dda groups=100 out=work.ranks;
  var &var;
  ranks bin;
run;

proc means data=work.ranks noprint nway;
  class bin;
  var ins &var;
  output out=work.bins sum(ins)=ins mean(&var)=&var;
run;

/* Calculate the empirical logit */
data work.bins;
  set work.bins;
  elogit=log((ins+(sqrt(_FREQ_ )/2))/
```

```
        ( _FREQ_ -ins+(sqrt(_FREQ_ )/2)));
run;

title1 "Empirical Logit against &var";
proc sgplot data=work.bins;
  reg y=elogit x=&var /
      curvelabel="Linear Relationship?"
     curvelabelloc=outside
     lineattrs=(color=ligr);
  series y=elogit x=&var;
run;

title1 "Empirical Logit against Binned &var";
proc sgplot data=work.bins;
  reg y=elogit x=bin /
      curvelabel="Linear Relationship?"
      curvelabelloc=outside
      lineattrs=(color=ligr);
  series y=elogit x=bin;
run;


/* ===================================================== */
/* Lesson 3, Section 4: l3d7b.sas
  Demonstration: Accommodating a Nonlinear Relationship,
  Part 2
  [m643_4_m2; derived from pmlr03d07.sas]             */
/* ===================================================== */
```

```
/* Using the binned values of DDABal may make for a more linear
   relationship between the input and the target. The following code
   creates DATA step code to bin DDABal, yielding a new predictor, B_DDABal.  */

/* Rank the observations. */

proc rank data=work.train_imputed_swoe_dda groups=100 out=work.ranks;
   var ddabal;
   ranks bin;
run;

/* Save the endpoints of each bin */

proc means data=work.ranks noprint nway;
   class bin;
   var ddabal;
   output out=endpts max=max;
run;

title1 "Checking Account Balance Endpoints";
proc print data=work.endpts(obs=10);
run;

/* Write the code to assign individuals to bins according to the DDABal. */

filename rank "&PMLRfolder/rank.sas";

data _null_;
   file rank;
```

```
  set work.endpts end=last;
  if _n_=1 then put "select;";
  if not last then do;
    put "  when (ddabal <= " max ") B_DDABal =" bin ";";
  end;
  else if last then do;
    put "  otherwise B_DDABal =" bin ";" / "end;";
  end;
run;

/* Use the code. */

data work.train_imputed_swoe_bins;
  set work.train_imputed_swoe_dda;
  %include rank / source;
run;

title1 "Minimum and Maximum Checking Account Balance by Bin";
proc means data=work.train_imputed_swoe_bins min max;
  class B_DDABal;
  var DDABal;
run;
title1 ;

/* Switch the binned DDABal (B_DDABal) for the originally scaled
  DDABal input in the list of potential inputs. */
%global screened;
%let screened=SavBal Dep DDA CD Sav CC ATM MM branch_swoe Phone IRA
         IRABal B_DDABal ATMAmt ILS POS NSF CCPurc SDB DepAmt
```

```
          CCBal Inv InArea Age CashBk MICRScor Income;


/* ===================================================== */
/* Lesson 3, Section 5: l3d8a.sas
   Demonstration: Detecting Interactions
   [m643_5_m; derived from pmlr03d08.sas]             */
/* ===================================================== */

title1 "P-Value for Entry and Retention";

%global sl;
proc sql;
   select 1-probchi(log(sum(ins ge 0)),1) into :sl
   from work.train_imputed_swoe_bins;
quit;

title1 "Interaction Detection using Forward Selection";
proc logistic data=work.train_imputed_swoe_bins;
   class res (param=ref ref='S');
   model ins(event='1')= &screened res
            SavBal|Dep|DDA|CD|Sav|CC|ATM|MM|branch_swoe|Phone|IRA|
            IRABal|B_DDABal|ATMAmt|ILS|POS|NSF|CCPurc|SDB|DepAmt|
            CCBal|Inv|InArea|Age|CashBk|MICRScor|Income|res @2 / include=28 clodds=pl
      selection=forward slentry=&sl;
run;


/* ===================================================== */
```

```
/* Lesson 3, Section 5: l3d8b.sas
  Demonstration: Using Backward Elimination to Subset the
  Variables
  [m643_5_n; derived from pmlr03d08.sas]           */
/* ===================================================== */

title1 "Backward Selection for Variable Annuity Data Set";
proc logistic data=work.train_imputed_swoe_bins;
  class res (param=ref ref='S');
  model ins(event='1')= &screened res SavBal*B_DDABal MM*B_DDABal
           branch_swoe*ATMAmt B_DDABal*Sav SavBal*SDB
           SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt
           SavBal*IRA SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM
           IRA*B_DDABal CD*MM MM*IRABal CD*Sav B_DDABal*CashBk Sav*CC
          / clodds=pl
      selection=backward slstay=&sl hier=single fast;
run;


/* ===================================================== */
/* Lesson 3, Section 5: l3d8c.sas
  Demonstration: Displaying Odds Ratios for Variables
  Involved in Interactions
  [m643_5_o; derived from pmlr03d08.sas]           */
/* ===================================================== */

title1 "Candidate Model for Variable Annuity Data Set";
ods select OddsRatiosPL;
proc logistic data=work.train_imputed_swoe_bins;
```

```
  model ins(event='1')= SavBal Dep DDA CD Sav CC ATM MM branch_swoe IRA B_DDABal
                ATMAmt ILS NSF SDB
                DepAmt Inv SavBal*B_DDABal MM*B_DDABal
                branch_swoe*ATMAmt Sav*B_DDABal
                SavBal*SDB SavBal*DDA AtmAmt*DepAmt B_DDABAL*ATMAmt SavBal*IRA
                SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM IRA*B_DDABal
                CD*MM CD*Sav Sav*CC / clodds=pl;
  oddsratio B_DDABAL / at(savbal=0, 1211, 52299) cl=pl;
run;



/* ===================================================== */
/* Lesson 3, Section 5: l3d8d.sas
  Demonstration: Creating an Interaction Plot
  [m643_5_r; derived from pmlr03d08.sas]          */
/* ===================================================== */

/*----  MACRO INTERACT  ----*\
Reserved data set names: work.percentiles
                work.plot
\*--------------------------*/
%macro interact(data=,target=,event=,inputs=,var1=,var2=,mean_inputs=);

proc logistic data=&data noprint;
  model &target(event="&event")= &inputs;
  code file="&PMLRfolder/interaction.txt";
run;

proc univariate data=&data noprint;
```

```
  var &var1 &var2;
  output out=work.percentiles pctlpts=5 25 50 75 95 pctlpre=&var1._p &var2._p;
run;

data _null_;
  set work.percentiles;
  call symput("&var1._p5",&var1._p5);
  call symput("&var1._p25",&var1._p25);
  call symput("&var1._p50",&var1._p50);
  call symput("&var1._p75",&var1._p75);
  call symput("&var1._p95",&var1._p95);
  call symput("&var2._p5",&var2._p5);
  call symput("&var2._p25",&var2._p25);
  call symput("&var2._p50",&var2._p50);
  call symput("&var2._p75",&var2._p75);
  call symput("&var2._p95",&var2._p95);
run;

proc means data=&data noprint;
  var &mean_inputs;
  output out=work.plot mean=;
run;

data work.plot(drop=_type_ _freq_);
  set work.plot;
  do &var2=&&&var2._p5,&&&var2._p25,&&&var2._p50,&&&var2._p75,&&&var2._p95;
    do &var1=&&&var1._p5,&&&var1._p25,&&&var1._p50,&&&var1._p75,&&&var1._p95;
      %include "&PMLRfolder/interaction.txt";
      output;
```

```
    end;
  end;
run;

title1 "Interaction Plot of &var2 by &var1";
proc sgplot data=work.plot;
  series y=p_&target&event x=&var2 / group=&var1;
  yaxis label="Probability of &target";
run;

%mend interact;

%interact(data=train_imputed_swoe_bins,target=ins,event=1,
          inputs=SavBal Dep DDA CD Sav CC ATM MM branch_swoe
          IRA B_DDABal ATMAmt ILS NSF SDB DepAmt Inv
          SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt Sav*B_DDABal
          SavBal*SDB SavBal*DDA AtmAmt*DepAmt B_DDABAL*ATMAmt SavBal*IRA
          SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM IRA*B_DDABal
          CD*MM CD*Sav Sav*CC,var1=SavBal,var2=B_DDABal,mean_inputs=SavBal Dep
          DDA CD Sav CC ATM MM branch_swoe IRA B_DDABal ATMAmt ILS NSF SDB
          DepAmt Inv);


/* ===================================================== */
/* Lesson 3, Section 5: l3d8e.sas
  Demonstration: Using the Best-Subsets Selection Method
  [m643_5_s; derived from pmlr03d08.sas]          */
/* ===================================================== */
```

```
data work.train_imputed_swoe_bins;
  set work.train_imputed_swoe_bins;
  resr=(res='R');
  resu=(res='U');
run;

/* Run best subsets */
title1 "Models Selected by Best Subsets Selection";
proc logistic data=work.train_imputed_swoe_bins;
   model ins(event='1')=&screened resr resu SavBal*B_DDABal MM*B_DDABal
               branch_swoe*ATMAmt B_DDABal*Sav SavBal*SDB
               SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt
               SavBal*IRA SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM
               IRA*B_DDABal CD*MM MM*IRABal CD*Sav B_DDABal*CashBk Sav*CC
               / selection=score best=1;

run;



/* ===================================================== */
/* Lesson 3, Section 5: l3d8f.sas
   Demonstration: Using Fit Statistics to Select a Model
   [m643_5_L; derived from pmlr03d08.sas]              */
/* ===================================================== */

/* The fitstat macro generates model fit statistics for the
   models selected in the all subsets selection. The macro
   variable IM is set equal to the variable names in the
   model_indx model while the macro variable IC is set
```

```
  equal to the number of variables in the model_indx model. */

%macro fitstat(data=,target=,event=,inputs=,best=,priorevent=);

ods select none;
ods output bestsubsets=work.score;

proc logistic data=&data namelen=50;
  model &target(event="&event")=&inputs / selection=score best=&best;
run;

/* The names and number of variables are transferred to macro
  variables using PROC SQL. */

proc sql noprint;
 select variablesinmodel into :inputs1 -
 from work.score;

 select NumberOfVariables into :ic1 -
 from work.score;
quit;

%let lastindx=&SQLOBS;

%do model_indx=1 %to &lastindx;

%let im=&&inputs&model_indx;
%let ic=&&ic&model_indx;
```

```
ods output scorefitstat=work.stat&ic ;
proc logistic data=&data namelen=50;
  model &target(event="&event")=&im;
  score data=&data out=work.scored fitstat
        priorevent=&priorevent;
run;

proc datasets
   library=work
   nodetails
   nolist;
   delete scored;
run;
quit;

%end;

/* The data sets with the model fit statistics are
   concatenated and sorted by BIC. */

data work.modelfit;
   set work.stat1 - work.stat&lastindx;
   model=_n_;
run;

%mend fitstat;

%fitstat(data=train_imputed_swoe_bins,target=ins,event=1,inputs=&screened resr resu
             SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt B_DDABal*Sav SavBal*SDB
```

```
        SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt SavBal*IRA
        SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM IRA*B_DDABal CD*MM
        MM*IRABal CD*Sav B_DDABal*CashBk Sav*CC,best=1,priorevent=0.02);

proc sort data=work.modelfit;
  by bic;
run;

title1 "Fit Statistics from Models selected from Best-Subsets";
ods select all;
proc print data=work.modelfit;
  var model auc aic bic misclass adjrsquare brierscore;
run;

%global selected;
proc sql;
  select VariablesInModel into :selected
  from work.score
  where numberofvariables=35;
quit;




/* ===================================================== */
/* Lesson 4, Section 1: l4d1.sas
  Demonstration: Preparing the Validation Data
  [m644_1_g; derived from pmlr04d01.sas]          */
/* ===================================================== */
```

```
title1 "Variables with Missing Values on the Validation Data Set";
proc means data=work.valid nmiss;
  var SavBal DDA CD Sav MM IRA IRABal ATMAmt ILS NSF SDB CCBal Inv
     DepAmt Dep ATM CC;
run;

proc univariate data=work.train_imputed_swoe_bins noprint;
  var cc ccbal inv;
  output out=work.medians
       pctlpts=50
       pctlpre=cc ccbal inv;
run;

data work.valid_imputed_swoe_bins(drop=cc50 ccbal50 inv50 i);
  if _N_=1 then set work.medians;
  set work.valid;
  array x(*) cc ccbal inv;
  array med(*) cc50 ccbal50 inv50;
  do i=1 to dim(x);
    if x(i)=. then x(i)=med(i);
  end;
  %include brswoe;
  if not dda then ddabal=&mean;
  %include rank;
run;



/* ===================================================== */
```

```
/* Lesson 4, Section 2: l4d2.sas
   Demonstration: Measuring Model Performance Based on
   Commonly-Used Metrics
   [m644_2_i; derived from pmlr04d02.sas]             */
/* ===================================================== */

ods select roccurve scorefitstat;
proc logistic data=work.train_imputed_swoe_bins;
   model ins(event='1')=&selected;
   score data=work.valid_imputed_swoe_bins out=work.scoval
         priorevent=&pi1 outroc=work.roc fitstat;
run;

title1 "Statistics in the ROC Data Set";
proc print data=work.roc(obs=10);
   var _prob_ _sensit_ _1mspec_;
run;

data work.roc;
   set work.roc;
   cutoff=_PROB_;
   specif=1-_1MSPEC_;
   tp=&pi1*_SENSIT_;
   fn=&pi1*(1-_SENSIT_);
   tn=(1-&pi1)*specif;
   fp=(1-&pi1)*_1MSPEC_;
   depth=tp+fp;
   pospv=tp/depth;
   negpv=tn/(1-depth);
```

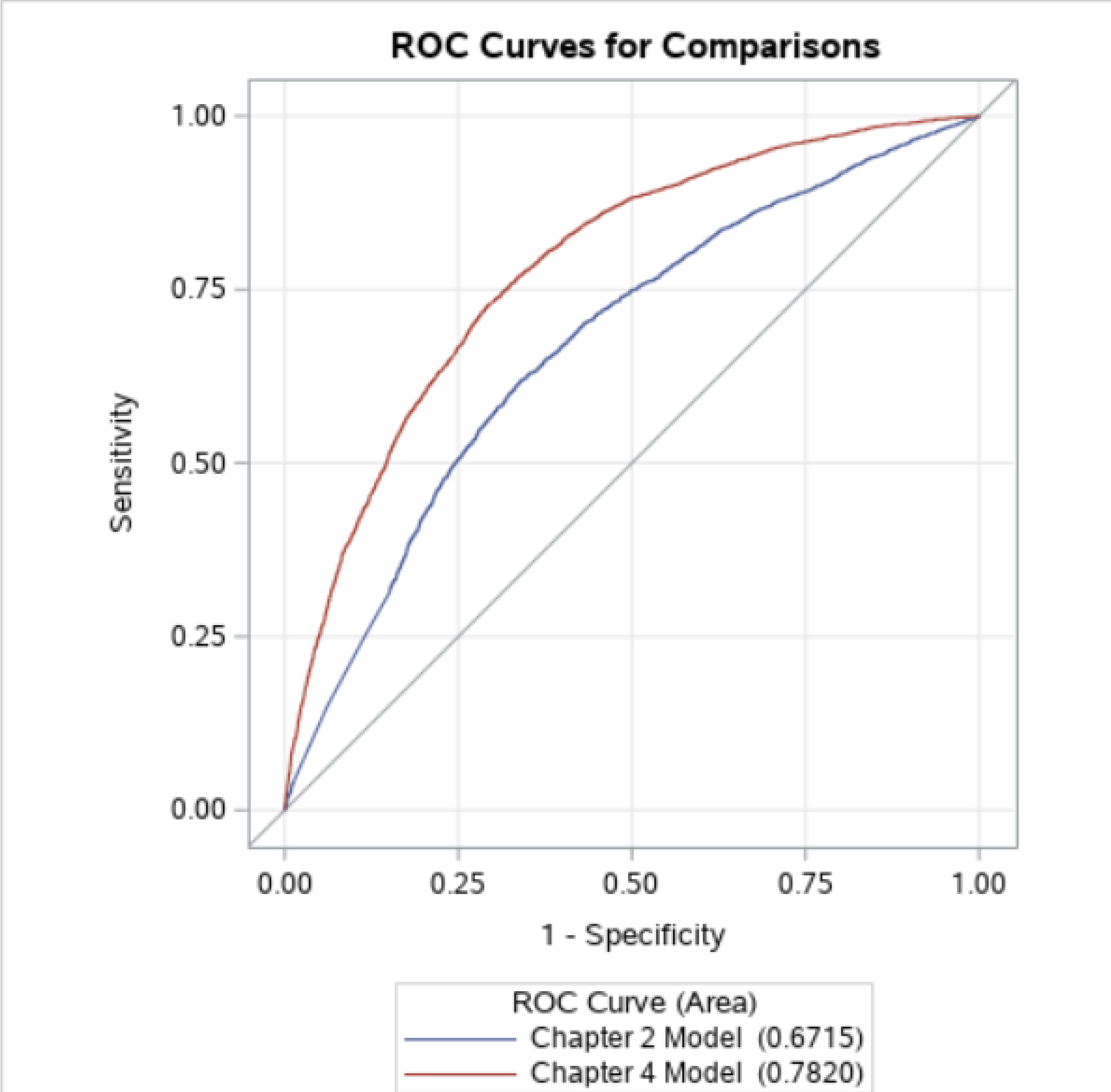
```
  acc=tp+tn;
  lift=pospv/&pi1;
  keep cutoff tn fp fn tp
       _SENSIT_ _1MSPEC_ specif depth
       pospv negpv acc lift;
run;

/* Create a lift chart */
title1 "Lift Chart for Validation Data";
proc sgplot data=work.roc;
  where 0.005 <= depth <= 0.50;
  series y=lift x=depth;
  refline 1.0 / axis=y;
  yaxis values=(0 to 9 by 1);
run; quit;
title1 ;



/* ===================================================== */
/* Lesson 4, Section 3: l4d3.sas
  Demonstration: Using a Profit Matrix to Measure Model
  Performance
  [m644_3_i; derived from pmlr04d03.sas]              */
/* ===================================================== */

/* Add the decision variable    */
/* (based on the profit matrix) */
/* and calculate profit         */
```

```
%global rho1;
proc SQL noprint;
  select mean(INS) into :rho1 from pmlr.develop;
quit;

data work.scoval;
   set work.scoval;
   sampwt=(&pi1/&rho1)*(INS)
          + ((1-&pi1)/(1-&rho1))*(1-INS);
   decision=(p_1 > 0.01);
   profit=decision*INS*99
          - decision*(1-INS)*1;
run;

/* Calculate total and average profit */

title1 "Total and Average Profit";
proc means data=work.scoval sum mean;
   weight sampwt;
   var profit;
run;

/* Investigate the true positive and */
/* false positive rates              */
data work.roc;
   set work.roc;
   AveProf=99*tp - 1*fp;
run;
```

```
title1 "Average Profit Against Depth";
proc sgplot data=work.roc;
  series y=aveProf x=depth;
  yaxis label="Average Profit";
run;

title1 "Average Profit Against Cutoff";
proc sgplot data=work.roc;
  where cutoff le 0.05;
  refline .01 / axis=x;
  series y=aveProf x=cutoff;
  yaxis label="Average Profit";
run;



/* ===================================================== */
/* Lesson 4, Section 4: l4d5.sas
  Demonstration: Using the K-S Statistic to Measure Model
  Performance
  [m644_4_g; derived from pmlr04d04.sas]             */
/* ===================================================== */

title1 "K-S Statistic for the Validation Data Set";
proc npar1way edf data=work.scoval;
  class ins;
  var p_1;
run;
```

```
/* ===================================================== */
/* Lesson 4, Section 5: l4d5.sas
   Demonstration: Comparing ROC Curves to Measure Model
   Performance
   [m644_5_g1; derived from pmlr04d05.sas]            */
/* ===================================================== */

proc logistic data=work.train_imputed_swoe_bins noprint;
   class res;
   model ins(event='1')=dda ddabal dep depamt checks res;
   score data=work.valid_imputed_swoe_bins
         out=work.sco_validate(rename=(p_1=p_ch2));
run;

proc logistic data=work.train_imputed_swoe_bins noprint;
   model ins(event='1')=&selected;
   score data=work.sco_validate out=work.sco_validate(rename=(p_1=p_sel));
run;

title1 "Validation Data Set Performance";
ods select ROCOverlay ROCAssociation ROCContrastTest;
proc logistic data=work.sco_validate;
   model ins(event='1')=p_ch2 p_sel / nofit;
   roc "Chapter 2 Model" p_ch2;
   roc "Chapter 4 Model" p_sel;
   roccontrast "Comparing the Two Models";
run;
```

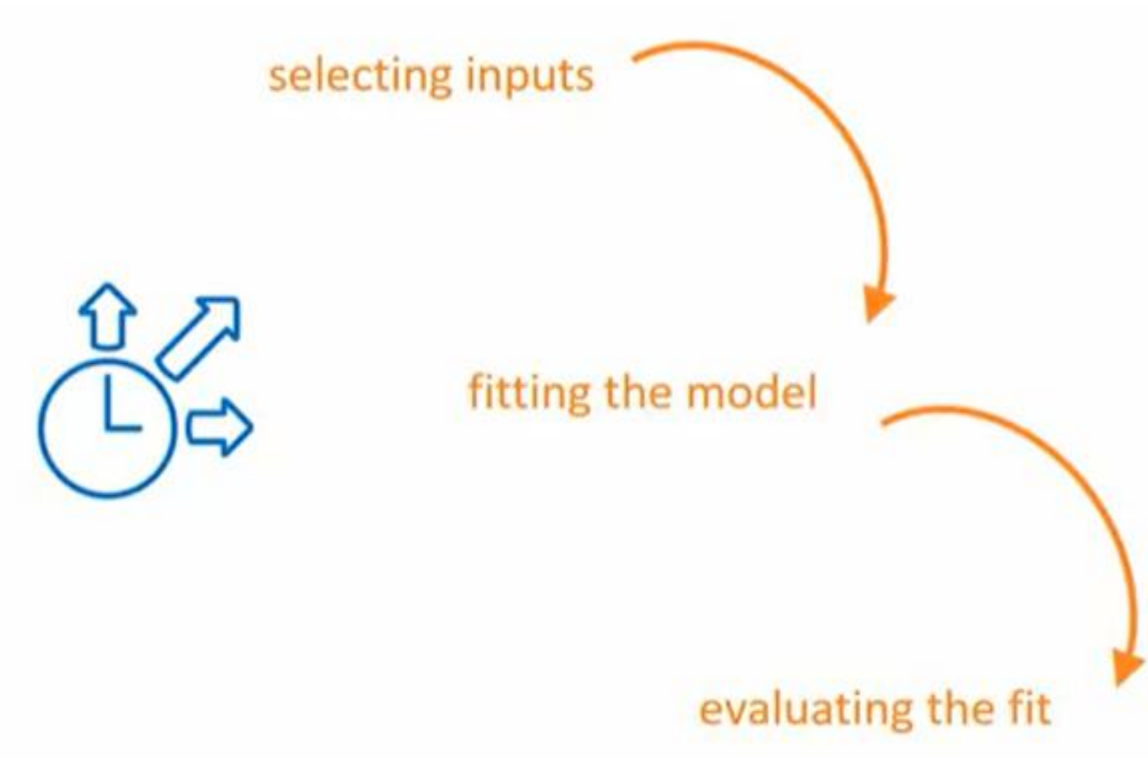
The LOGISTIC Procedure

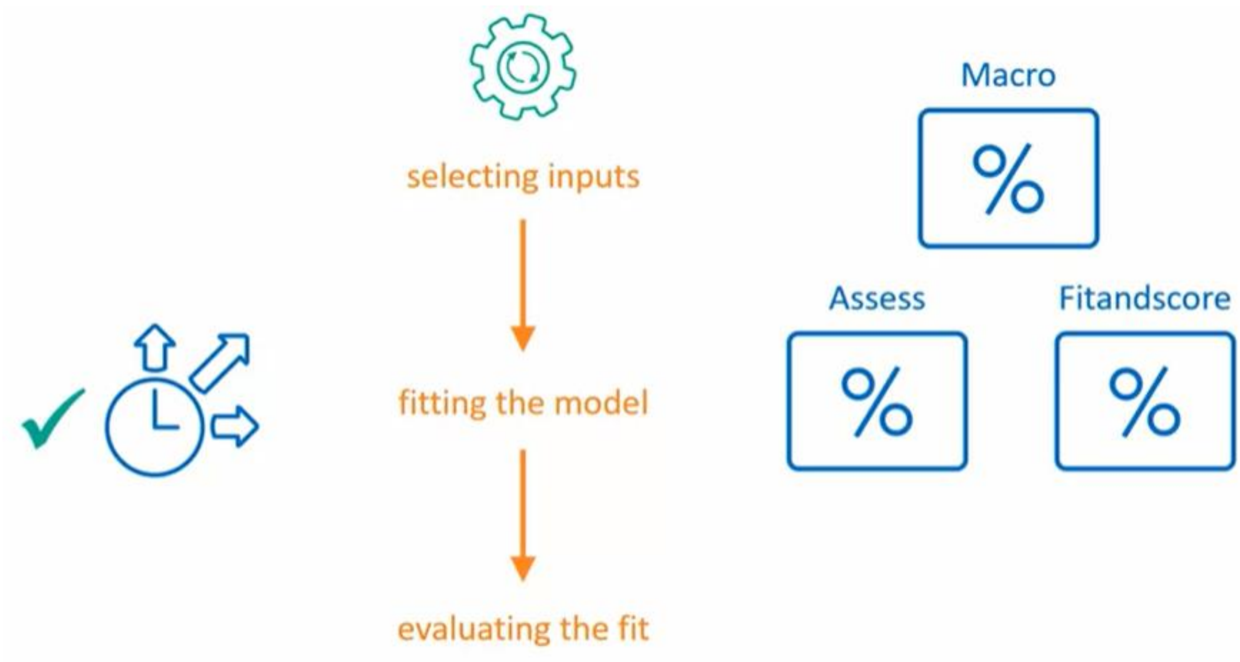


ROC Association Statistics							
	Mann-Whitney						
ROC Model	Area	Standard Error	95% Wald Confidence Limits		Somers' D	Gamma	Tau-a
Chapter 2 Model	0.6715	0.00543	0.6609	0.6821	0.3430	0.3475	0.1553
Chapter 4 Model	0.7820	0.00456	0.7730	0.7909	0.5639	0.5640	0.2554

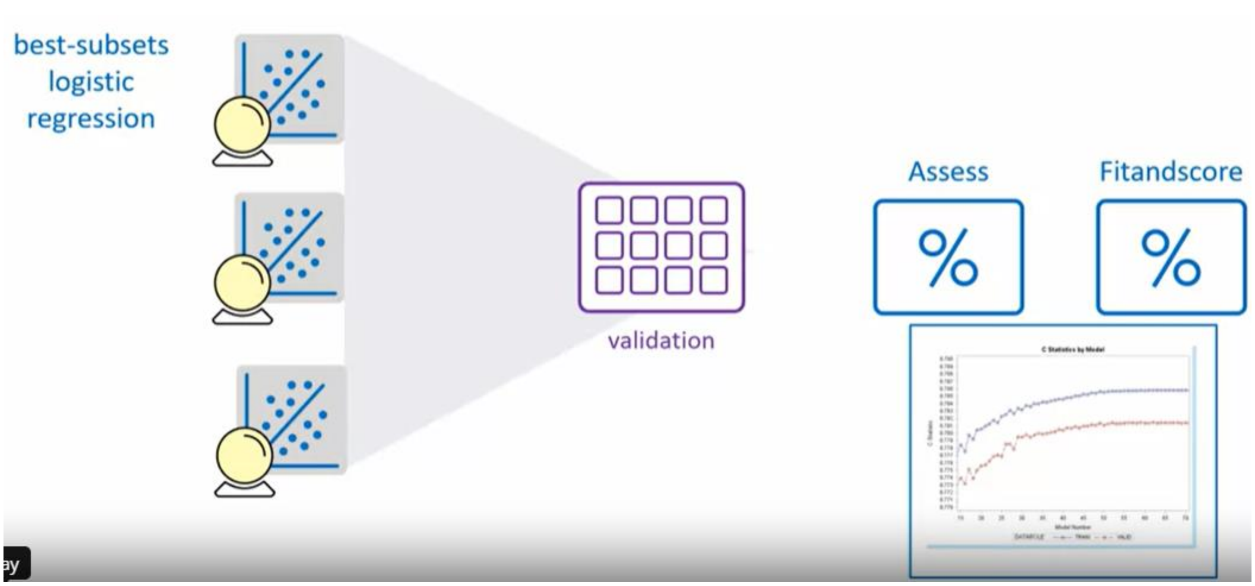
ROC Contrast Test Results			
Contrast	DF	Chi-Square	Pr > ChiSq
Comparing the Two Models	1	528.4259	<.0001

Using Macros to Compare Many Models



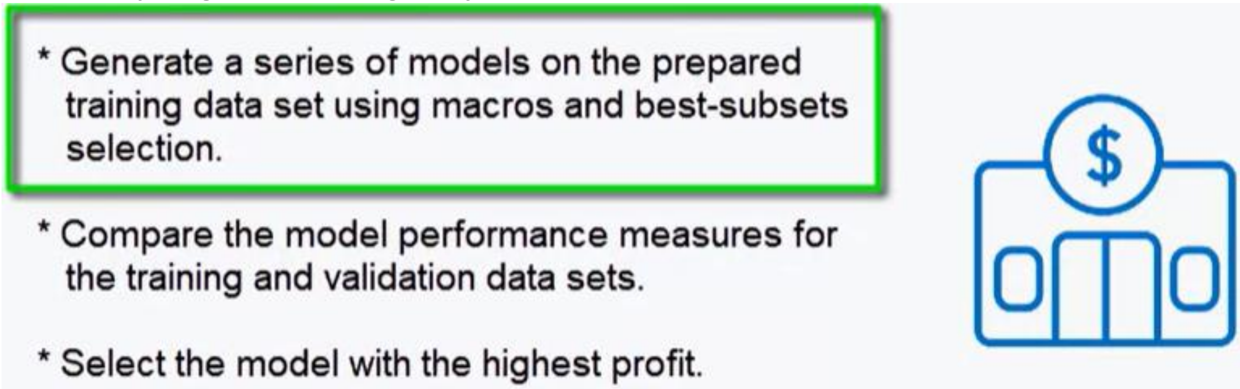


selecting inputs
fitting the model
evaluating the fit
Macro
%
Assess
%
Fitandscore
%



best-subsets
logistic
regression
validation
Assess
%
Fitandscore
%

Demo Comparing and Evaluating Many Models, Part 1



pmlr04d06.sas

```
*/
%put &screened;

data work.valid_imputed_swoe_bins;
   set work.valid_imputed_swoe_bins;
   MICRScor=(crscore=.);
   resr=(res='R');
   resu=(res='U');
run;

title1 "Variables with Missing Values on the Validation Data Set";
proc means data=work.valid_imputed_swoe_bins nmiss;
   var &screened;
run;
```

```
2559  %put &screened;
SavBal Dep DDA CD Sav CC ATM MM branch_swoe Phone IRA                IRABal B_DDABal ATMAmt I
POS NSF CCPurc SDB DepAmt                          CCBal Inv InArea Age CashBk MICRScor Income
```

```
title1 "Variables with Missing Values on the Validation Data Set";
proc means data=work.valid_imputed_swoe_bins nmiss;
   var &screened;
run;
```

Variables with Missing Values on the Validation Data Set

The MEANS Procedure

Variable	Label	N Miss
SavBal	Saving Balance	0
Dep	Checking Deposits	0
DDA	Checking Account	0
CD	Certificate of Deposit	0
Sav	Saving Account	0
CC	Credit Card	0
ATM	ATM	0
MM	Money Market	0
branch_swoe		0
Phone	Number Telephone Banking	1350
IRA	Retirement Account	0
IRABal	IRA Balance	0
B_DDABal		0
ATMAmt	ATM Withdrawal Amount	0
ILS	Installment Loan	0
POS	Number Point of Sale	1350
NSF	Number Insufficient Fund	0
CCPurc	Credit Card Purchases	1350
SDB	Safety Deposit Box	0
DepAmt	Amount Deposited	0
CCBal	Credit Card Balance	0
Inv	Investment	0
InArea	Local Address	0
Age	Age	2061
CashBk	Number Cash Back	0
MICRScor		0
Income	Income	1866

```
proc stdize data=work.train_imputed_swoe_bins method=median reponly
            OUTSTAT=med;
   var &screened;
run;
```

```
proc stdize data=work.valid_imputed_swoe_bins reponly method=in(med)
            out=work.valid_imputed_swoe_bins;
   var &screened;
run;
```

pmlr04d06.sas

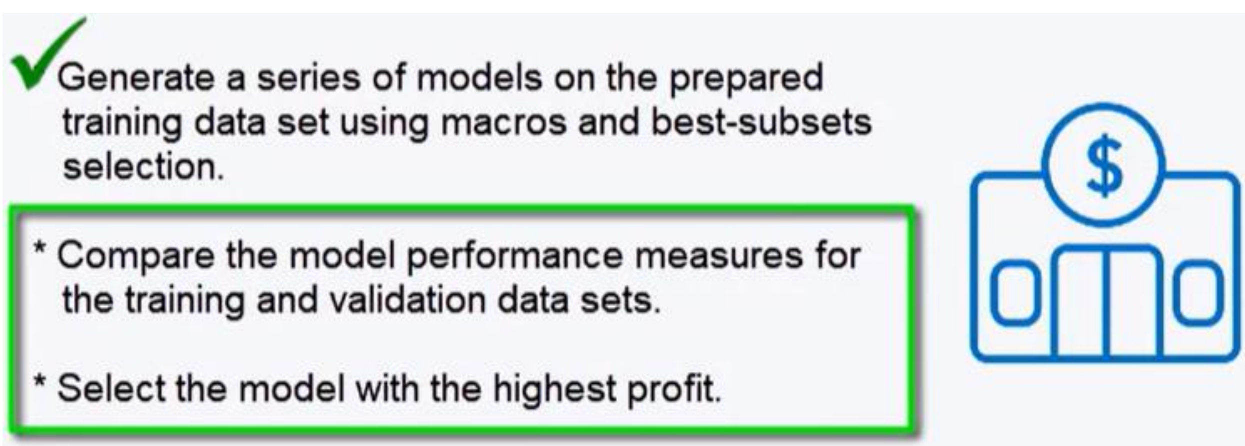
```
%include "&PMLRfolder\pmlr04d06a.sas";
%include "&PMLRfolder\pmlr04d06b.sas";

%fitandscore(data_train=train_imputed_swoe_bins,
             data_validate=valid_imputed_swoe_bins,
             target=ins,predictors=&screened resr resu SavBal*B_DDABal
             MM*B_DDABal branch_swoe*ATMAmt B_DDABal*Sav SavBal*SDB
             SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt
             SavBal*IRA SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM
             IRA*B_DDABal CD*MM MM*IRABal CD*Sav B_DDABal*CashBk Sav*CC,
             best=2,profit00=0,profit01=-1,profit10=0,profit11=99,pi1=0.02
```

Assess

Fitandscore

For more information, see Assess and FitandScore Macros in the Resources section.

Demo Comparing and Evaluating Many Models, Part 2



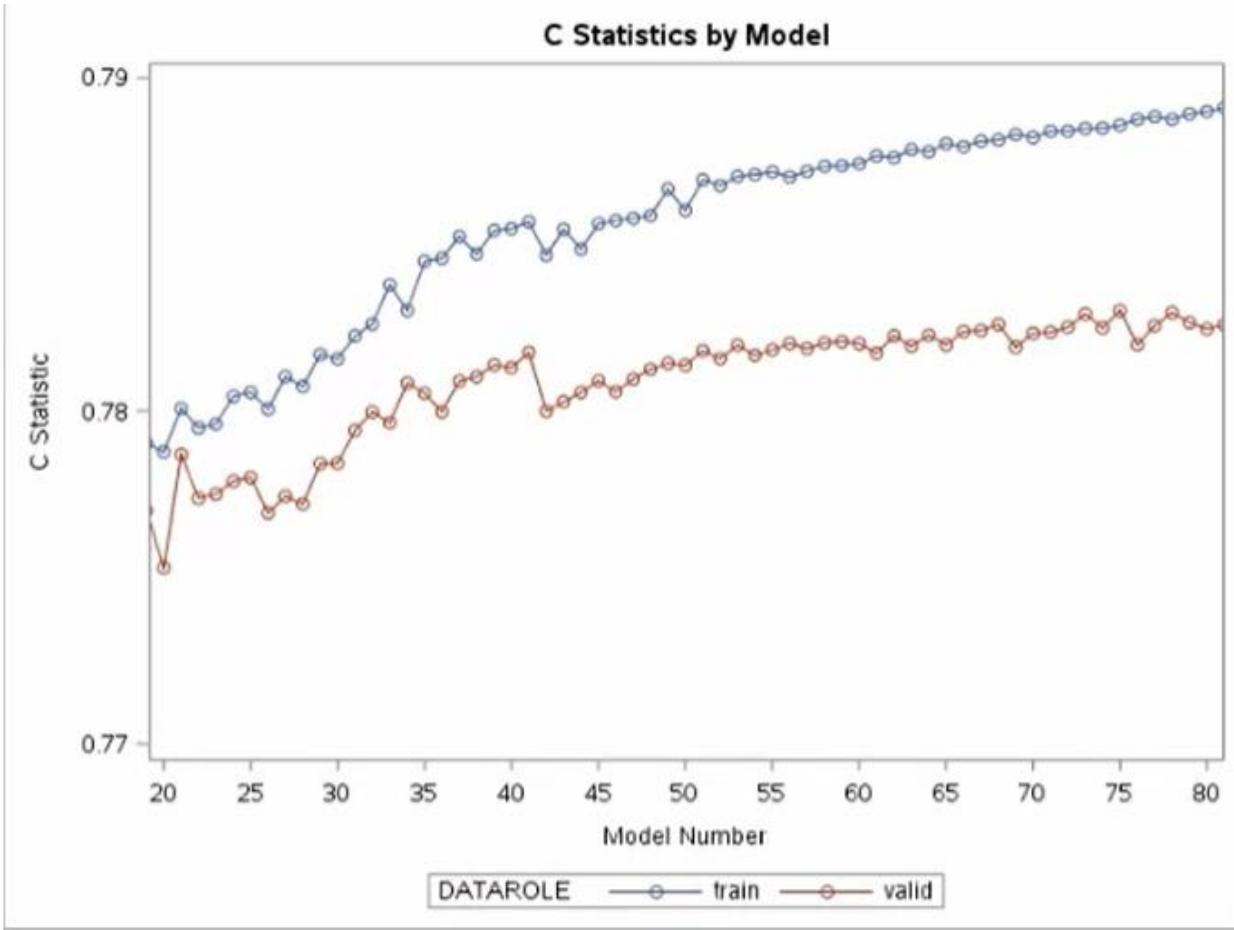
pmlr04d06.sas

```
title1 "Model Performance Measures for Training and Validation Data Sets";
proc print data=work.results(obs=24);
run;
```

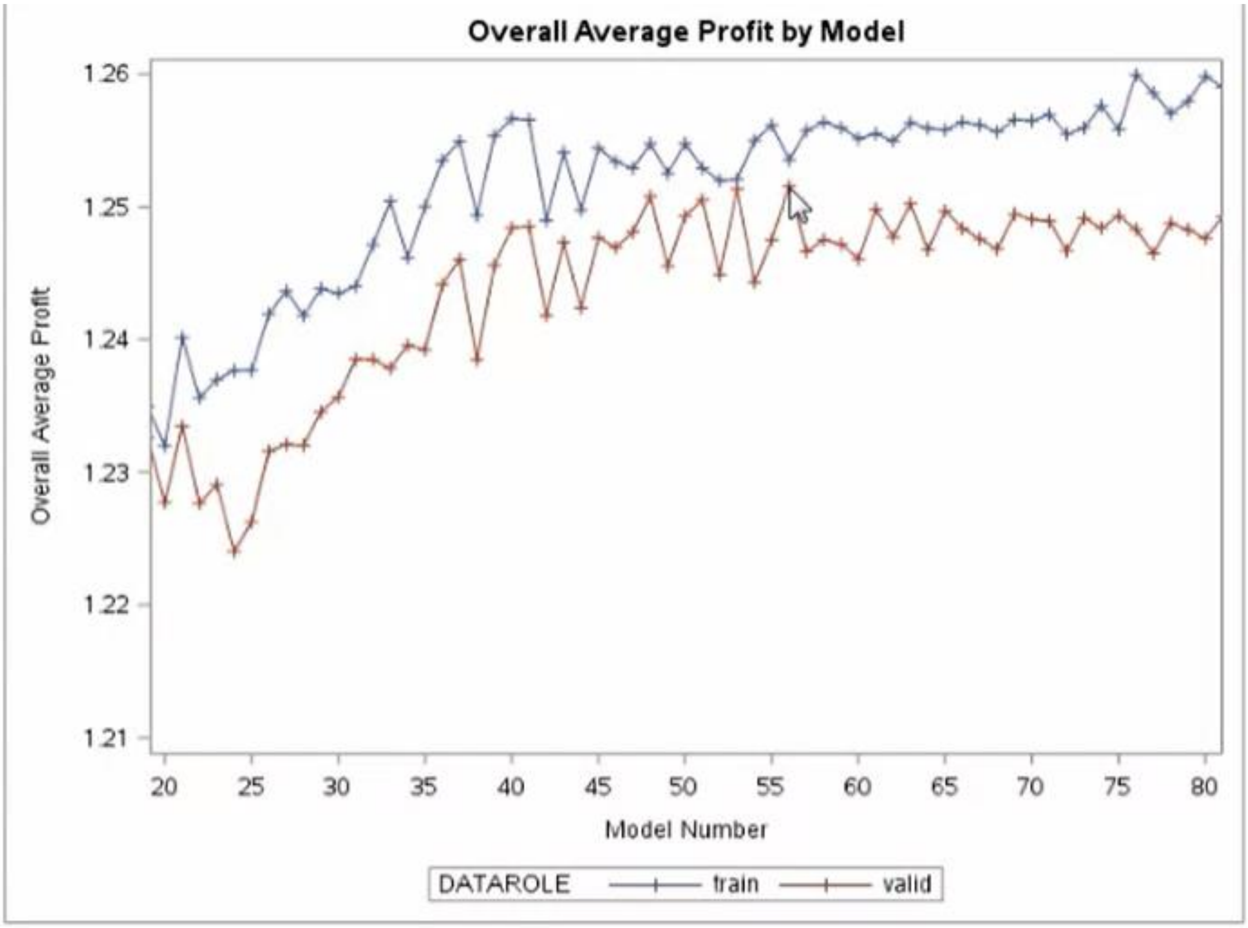
Model Performance Measures for Training and Validation Data Sets

Obs	DATAROLE	INPUT_COUNT	TOTAL_PROFIT	OVERALL_AVG_PROFIT	ASE	C	index
1	train	1	23129.89	1.07521	0.32753	0.65653	1
2	valid	1	11480.13	1.06770	0.32773	0.65483	1
3	train	1	21511.97	1.00000	0.32920	0.48375	2
4	valid	1	10751.16	0.99991	0.32932	0.47975	2
5	train	2	23760.31	1.10451	0.32416	0.68932	3
6	valid	2	11883.25	1.10520	0.32438	0.68850	3
7	train	2	24629.62	1.14492	0.32564	0.69135	4
8	valid	2	12105.74	1.12589	0.32588	0.68752	4
9	train	3	24966.72	1.16059	0.31447	0.73595	5
10	valid	3	12572.67	1.16931	0.31586	0.73638	5
11	train	3	24852.80	1.15530	0.32238	0.71582	6
12	valid	3	12211.33	1.13571	0.32262	0.71388	6
13	train	4	25865.37	1.20237	0.31313	0.75060	7
14	valid	4	12814.52	1.19181	0.31338	0.74923	7
15	train	4	25413.61	1.18137	0.31422	0.74006	8
16	valid	4	12528.90	1.16524	0.31574	0.73772	8
17	train	5	25843.04	1.20133	0.31157	0.76373	9
18	valid	5	12829.41	1.19319	0.31199	0.75949	9
19	train	5	25824.53	1.20047	0.31372	0.75299	10
20	valid	5	12911.14	1.20079	0.31514	0.75457	10
21	train	6	26009.50	1.20907	0.31205	0.76347	11
22	valid	6	12992.10	1.20832	0.31362	0.76183	11
23	train	6	26543.43	1.23389	0.31013	0.77402	12
24	valid	6	13110.44	1.21933	0.31051	0.77099	12

```
title1 "C Statistics by Model";
proc sgplot data=work.results;
   where index > 18;
   series y=c x=index / group=datarole markerattrs=(symbol=circle) markers
   yaxis label="C Statistic" Values=(0.770 to 0.790 by 0.01);
   xaxis label="Model Number" Values=(20 to 80 by 5);
run;
```



```
title1 "Overall Average Profit by Model";
proc sgplot data=work.results;
   where index > 18;
   series y=overall_avg_profit x=index /
          group=datarole markerattrs=(symbol=plus) markers;
   yaxis label="Overall Average Profit" Values=(1.21 to 1.26 by 0.010);
   xaxis label="Model Number" Values=(20 to 80 by 5);
run;
```



```
title1 "Model Number with Highest Profit";
%global index;
proc sql;
   select index into :index
   from work.results
   where datarole='valid'
   having overall_avg_profit=max(overall_avg_profit);
quit;
```

Model Number with Highest Profit

index
56

```
/* Remove all blanks from index */
%let index=%cmpres(&index);
title1 "Logistic Model with Highest Profit";
proc logistic data=work.train_imputed_swoe_bins;
   model ins(event='1')=&&inputs&index;
   score data=work.valid_imputed_swoe_bins out=work.scoval2 fitstat;
run;
```

Analysis of Maximum Likelihood Estimates					
Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	1	-1.9217	0.0926	430.2719	<.0001
SavBal	1	0.000164	9.545E-6	293.4418	<.0001
DDA	1	-0.2544	0.0491	26.8705	<.0001
CD	1	0.9672	0.0522	343.7165	<.0001
Sav	1	0.8459	0.0844	100.5593	<.0001
MM	1	1.7664	0.1462	146.0518	<.0001
branch_swoe	1	0.9408	0.0795	139.8803	<.0001
IRA	1	1.1164	0.1968	32.1891	<.0001
B_DDABal	1	0.0284	0.000996	811.3105	<.0001
ATMAmt	1	0.000201	0.000029	47.0271	<.0001
ILS	1	-0.2437	0.0766	10.1115	0.0015
NSF	1	0.3651	0.0651	31.5015	<.0001
Inv	1	0.4418	0.0980	20.3042	<.0001
SavBal*B_DDABal	1	-1.7E-6	1.219E-7	193.9633	<.0001
MM*B_DDABal	1	-0.0154	0.00213	52.4092	<.0001
branch_swoe*ATMAmt	1	0.000163	0.000022	52.3780	<.0001
Sav*B_DDABal	1	-0.00984	0.00131	56.7718	<.0001
SavBal*SDB	1	-0.00001	3.76E-6	10.0137	0.0016
SavBal*DDA	1	0.000029	4.514E-6	41.8837	<.0001
ATMAmt*DepAmt	1	-1.68E-9	3.34E-10	25.4236	<.0001

Odds Ratio Estimates			
Effect	Point Estimate	95% Wald Confidence Limits	
ILS	0.784	0.674	0.911
NSF	1.441	1.268	1.637
Inv	1.555	1.284	1.885

Association of Predicted Probabilities and Observed Responses			
Percent Concordant	78.7	Somers' D	0.574
Percent Discordant	21.3	Gamma	0.575
Percent Tied	0.0	Tau-a	0.260
Pairs	104768511	c	0.787

Fit Statistics for SCORE Data

Data Set	Total Frequency	Log Likelihood	Error Rate	AIC	AICC	BIC	SC	R-Square	Max-Rescaled R-Square	AUC
WORK.VALID_IMPUTED_SWOE_BINS	10752	-5750.7	0.2672	11559.46	11559.62	11770.66	11770.66	0.197976	0.273139	0.782223

Fit Statistics for SCORE Data

Total Frequency	Log Likelihood	Error Rate	AIC	AICC	BIC	SC	R-Square	Max-Rescaled R-Square	AUC	Brier Score
10752	-5750.7	0.2672	11559.46	11559.62	11770.66	11770.66	0.197976	0.273139	0.782223	0.1768

Question 4.07

Which of the following graphs would be affected if you changed the profit matrix values?

Only the graph of the overall average profit is affected by the profit matrix values.

```
/* Code for the Lesson 1, 2 and 3 Demonstrations in the SAS e-Course
   "Predictive Modeling Using Logistic Regression" */

/* The demonstrations in this SAS e-course build on each
other. This file contains the code for all demonstrations in
Lesson 1, 2 and 3.

If you started a new SAS session since you ran the previous
demonstration(s), you need to set up access to the course
files (see the Course Overview and Data Setup) and then and
re-run the code for all previous demonstrations. The title of
each demonstration and the corresponding program file name
appear in a comment above the code for that demo.

Before you submit the code, make any necessary modifications to
the code, if indicated in comments.
Note: Most of the code requires no modifications.

Submit the code and check the log to verify that it ran without errors.

After performing the steps above, you are ready to proceed with the
current demonstration!
*/

/* ====================================================== */
/* Lesson 1, Section 1: l1d1.sas
   Demonstration: Examining the Code for Generating
   Descriptive Statistics and Frequency Tables
   [m641_1_i; derived from pmlr01d01.sas]            */
```

```
/* ===================================================== */

data work.develop;
  set pmlr.develop;
run;

%global inputs;
%let inputs=ACCTAGE DDA DDABAL DEP DEPAMT CASHBK
        CHECKS DIRDEP NSF NSFAMT PHONE TELLER
        SAV SAVBAL ATM ATMAMT POS POSAMT CD
        CDBAL IRA IRABAL LOC LOCBAL INV
        INVBAL ILS ILSBAL MM MMBAL MMCRED MTG
        MTGBAL CC CCBAL CCPURC SDB INCOME
        HMOWN LORES HMVAL AGE CRSCORE MOVED
        INAREA;

proc means data=work.develop n nmiss mean min max;
  var &inputs;
run;

proc freq data=work.develop;
  tables ins branch res;
run;

/* ===================================================== */
/* Lesson 1, Section 2: l1d2.sas
  Demonstration: Splitting the Data
  [m641_2_h; derived from pmlr01d02.sas]          */
/* ===================================================== */
```

```
/* Sort the data by the target in preparation for stratified sampling. */

proc sort data=work.develop out=work.develop_sort;
  by ins;
run;

/* The SURVEYSELECT procedure will perform stratified sampling
  on any variable in the STRATA statement. The OUTALL option
  specifies that you want a flag appended to the file to
  indicate selected records, not simply a file comprised
  of the selected records. */

proc surveyselect noprint data=work.develop_sort
            samprate=.6667 stratumseed=restore
            out=work.develop_sample
            seed=44444 outall;
  strata ins;
run;

/* Verify stratification. */

proc freq data=work.develop_sample;
  tables ins*selected;
run;

/* Create training and validation data sets. */
```

```
data work.train(drop=selected SelectionProb SamplingWeight)
    work.valid(drop=selected SelectionProb SamplingWeight);
  set work.develop_sample;
  if selected then output work.train;
  else output work.valid;
run;



/* ===================================================== */
/* Lesson 2, Section 1: l2d1.sas
  Demonstration: Fitting a Basic Logistic Regression Model,
  Parts 1 and 2
  [m642_1_k1, m642_1_k2; derived from pmlr02d01.sas]    */
/* ===================================================== */

title1 "Logistic Regression Model for the Variable Annuity Data Set";
proc logistic data=work.train
         plots(only maxpoints=none)=(effect(clband x=(ddabal depamt checks res))
         oddsratio (type=horizontalstat));
  class res (param=ref ref='S') dda (param=ref ref='0');
  model ins(event='1')=dda ddabal dep depamt
          cashbk checks res / stb clodds=pl;
  units ddabal=1000 depamt=1000 / default=1;
  oddsratio 'Comparisons of Residential Classification' res / diff=all cl=pl;
  effectplot slicefit(sliceby=dda x=ddabal) / noobs;
  effectplot slicefit(sliceby=dda x=depamt) / noobs;
run;
title1;
```

```
/* ====================================================== */
/* Lesson 2, Section 1: l2d2.sas
   Demonstration: Scoring New Cases
   [m642_1_n; derived from pmlr02d02.sas]              */
/* ====================================================== */

/* Score a new data set with one run of the LOGISTIC procedure with the
   SCORE statement.  */

proc logistic data=work.train noprint;
   class res (param=ref ref='S');
   model ins(event='1')= res dda ddabal dep depamt cashbk checks;
   score data = pmlr.new out=work.scored1;
run;

title1 "Predicted Probabilities from Scored Data Set";
proc print data=work.scored1(obs=10);
   var p_1 dda ddabal dep depamt cashbk checks res;
run;

title1 "Mean of Predicted Probabilities from Scored Data Set";
proc means data=work.scored1 mean nolabels;
   var p_1;
run;

/* Score a new data set with the OUTMODEL= amd INMODEL= options */
```

```
proc logistic data=work.train outmodel=work.scoredata noprint;
  class res (param=ref ref='S');
  model ins(event='1')= res dda ddabal dep depamt cashbk checks;
run;

proc logistic inmodel=work.scoredata noprint;
  score data = pmlr.new out=work.scored2;
run;

title1 "Predicted Probabilities from Scored Data Set";
proc print data=work.scored2(obs=10);
  var p_1 dda ddabal dep depamt cashbk checks res;
run;

/* Score a new data set with the CODE Statement  */

proc logistic data=work.train noprint;
  class res (param=ref ref='S');
  model ins(event='1')= res dda ddabal dep depamt cashbk checks;
  code file="&PMLRfolder/pmlr_score.txt";
run;

data work.scored3;
  set pmlr.new;
  %include "&PMLRfolder/pmlr_score.txt";
run;

title1 "Predicted Probabilities from Scored Data Set";
proc print data=work.scored3(obs=10);
```

```
  var p_ins1 dda ddabal dep depamt cashbk checks res;
run;
title1 ;



/* ===================================================== */
/* Lesson 2, Section 2: l2d3.sas
  Demonstration: Correcting for Oversampling
  [m642_2_f; derived from pmlr02d03.sas]              */
/* ===================================================== */

/* Specify the prior probability to correct for oversampling.  */
%global pi1;
%let pi1=.02;

/* Correct predicted probabilities */

proc logistic data=work.train noprint;
  class res (param=ref ref='S');
  model ins(event='1')=dda ddabal dep depamt cashbk checks res;
  score data=pmlr.new out=work.scored4 priorevent=&pi1;
run;

title1 "Adjusted Predicted Probabilities from Scored Data Set";
proc print data=work.scored4(obs=10);
  var p_1 dda ddabal dep depamt cashbk checks res;
run;

title1 "Mean of Adjusted Predicted Probabilities from Scored Data Set";
```

```
proc means data=work.scored4 mean nolabels;
  var p_1;
run;
title1 ;

/* Correct probabilities in the Score Code */

proc logistic data=work.train noprint;
  class res (param=ref ref='S');
  model ins(event='1')=dda ddabal dep depamt cashbk checks res;
  /* File suffix "txt" is used so you can view the file */
  /* with a native text editor. SAS prefers "sas", but  */
  /* when specified as a filename, SAS does not care.   */
  code file="&PMLRfolder/pmlr_score_adj.txt";
run;

%global rho1;
proc SQL noprint;
  select mean(INS) into :rho1
  from work.train;
quit;

data new;
  set pmlr.new;
  off=log(((1-&pi1)*&rho1)/(&pi1*(1-&rho1)));
run;

data work.scored5;
  set work.new;
```

```
  %include "&PMLRfolder/pmlr_score_adj.txt";
  eta=log(p_ins1/p_ins0) - off;
  prob=1/(1+exp(-eta));
run;

title1 "Adjusted Predicted Probabilities from Scored Data Set";
proc print data=scored5(obs=10);
  var prob dda ddabal dep depamt cashbk checks res;
run;
title1 ;




/* ===================================================== */
/* Lesson 3, Section 1: l3d1.sas
  Demonstration: Imputing Missing Values
  [m643_1_h; derived from pmlr03d01.sas]              */
/* ===================================================== */

title1 "Variables with Missing Values";
proc print data=work.train(obs=15);
  var ccbal ccpurc income hmown;
run;
title1 ;

/* Create missing indicators */
data work.train_mi(drop=i);
  set work.train;
  /* name the missing indicator variables */
```

```
  array mi{*} MIAcctAg MIPhone MIPOS MIPOSAmt
            MIInv MIInvBal MICC MICCBal
            MICCPurc MIIncome MIHMOwn MILORes
            MIHMVal MIAge MICRScor;
  /* select variables with missing values */
  array x{*} acctage phone pos posamt
           inv invbal cc ccbal
           ccpurc income hmown lores
           hmval age crscore;
  do i=1 to dim(mi);
    mi{i}=(x{i}=.);
    nummiss+mi{i};
  end;
run;

/* Impute missing values with the median */
proc stdize data=work.train_mi reponly method=median out=work.train_imputed;
  var &inputs;
run;

title1 "Imputed Values with Missing Indicators";
proc print data=work.train_imputed(obs=12);
  var ccbal miccbal ccpurc miccpurc income miincome hmown mihmown nummiss;
run;
title1 ;



/* ======================================================= */
/* Lesson 3, Section 2: l3d2a.sas
```

```
  Demonstration: Collapsing the Levels of a Nominal Input,
  Part 1
  [m643_2_g1; derived from pmlr03d02.sas]              */
/* ===================================================== */

proc means data=work.train_imputed noprint nway;
  class branch;
  var ins;
  output out=work.level mean=prop;
run;

title1 "Proportion of Events by Level";
proc print data=work.level;
run;

/* Use ODS to output the ClusterHistory output object into a data set
  named "cluster." */

ods output clusterhistory=work.cluster;

proc cluster data=work.level method=ward outtree=work.fortree
     plots=(dendrogram(vertical height=rsq));
  freq _freq_;
  var prop;
  id branch;
run;


/* ===================================================== */
```

```
/* Lesson 3, Section 2: l3d2b.sas

   Demonstration: Collapsing the Levels of a Nominal Input,

   Part 2

   [m643_2_g2; derived from pmlr03d02.sas]              */

/* ===================================================== */


/* Use the FREQ procedure to get the Pearson Chi^2 statistic of the

   full BRANCH*INS table. */


proc freq data=work.train_imputed noprint;

   tables branch*ins / chisq;

   output out=work.chi(keep=_pchi_) chisq;

run;


/* Use a one-to-many merge to put the Chi^2 statistic onto the clustering

   results. Calculate a (log) p-value for each level of clustering. */


data work.cutoff;

   if _n_=1 then set work.chi;

   set work.cluster;

   chisquare=_pchi_*rsquared;

   degfree=numberofclusters-1;

   logpvalue=logsdf('CHISQ',chisquare,degfree);

run;


/* Plot the log p-values against number of clusters. */


title1 "Plot of the Log of the P-Value by Number of Clusters";

proc sgplot data=work.cutoff;
```

```
  scatter y=logpvalue x=numberofclusters
        / markerattrs=(color=blue symbol=circlefilled);
  xaxis label="Number of Clusters";
  yaxis label="Log of P-Value" min=-120 max=-85;
run;
title1 ;

/* Create a macro variable (&ncl) that contains the number of clusters
  associated with the minimum log p-value. */

proc sql;
  select NumberOfClusters into :ncl
  from work.cutoff
  having logpvalue=min(logpvalue);
quit;

proc tree data=work.fortree nclusters=&ncl out=work.clus noprint;
  id branch;
run;

proc sort data=work.clus;
  by clusname;
run;

title1 "Levels of Branch by Cluster";
proc print data=work.clus;
  by clusname;
  id clusname;
run;
```

```
title1 ;

/* The DATA Step creates the scoring code to assign the branches to a cluster. */

filename brclus "&PMLRfolder/branch_clus.sas";

data _null_;
  file brclus;
  set work.clus end=last;
  if _n_=1 then put "select (branch);";
  put "  when ('" branch +(-1) "') branch_clus = '" cluster +(-1) "';";
  if last then do;
    put "  otherwise branch_clus = 'U';" / "end;";
  end;
run;

data work.train_imputed_greenacre;
  set work.train_imputed;
  %include brclus / source2;
run;



/* ===================================================== */
/* Lesson 3, Section 2: l3d3.sas
  Demonstration: Computing the Smoothed Weight of Evidence
  [m643_2_j; derived from pmlr03d03.sas]              */
/* ===================================================== */

/* Rho1 is the proportion of events in the training data set. */
```

```
%global rho1;
proc sql noprint;
  select mean(ins) into :rho1
  from work.train_imputed;
run;

/* The output data set from PROC MEANS will have the number of
  observations and events for each level of branch. */

proc means data=work.train_imputed sum nway noprint;
  class branch;
  var ins;
  output out=work.counts sum=events;
run;

/* The DATA Step creates the scoring code that assigns each branch to
  a value of the smoothed weight of evidence. */


filename brswoe "&PMLRfolder/swoe_branch.sas";

data _null_;
  file brswoe;
  set work.counts end=last;
  logit=log((events + &rho1*24)/(_FREQ_ - events + (1-&rho1)*24));
  if _n_=1 then put "select (branch);" ;
  put "  when ('" branch +(-1) "') branch_swoe = " logit ";" ;
  if last then do;
  logit=log(&rho1/(1-&rho1));
```

```
  put "  otherwise branch_swoe = " logit ";" / "end;";
  end;
run;

data work.train_imputed_swoe;
  set work.train_imputed;
  %include brswoe / source2;
run;



/* ===================================================== */
/* Lesson 3, Section 3: l3d4.sas
  Demonstration: Reducing Redundancy by Clustering Variables
  [m643_3_i; derived from pmlr03d04.sas]              */
/* ===================================================== */

/* Use the ODS OUTPUT statement to generate data sets based on the variable
  clustering results and the clustering summary. */

ods select none;
ods output clusterquality=work.summary
        rsquare=work.clusters;

proc varclus data=work.train_imputed_swoe maxeigen=.7 hi;
  var &inputs branch_swoe miacctag
     miphone mipos miposamt miinv
     miinvbal micc miccbal miccpurc
     miincome mihmown milores mihmval
     miage micrscor;
```

```
run;
ods select all;

/* Use the CALL SYMPUT function to create a macro variable:&NVAR =
  the number of of clusters. This is also the number of variables
  in the analysis, going forward. */

%global nvar;
data _null_;
  set work.summary;
  call symput('nvar',compress(NumberOfClusters));
run;

title1 "Variables by Cluster";
proc print data=work.clusters noobs label split='*';
  where NumberOfClusters=&nvar;
  var Cluster Variable RSquareRatio VariableLabel;
  label RSquareRatio="1 - RSquare*Ratio";
run;
title1 ;

title1 "Variation Explained by Clusters";
proc print data=work.summary label;
run;

/* Choose a representative from each cluster.  */
%global reduced;
%let reduced=branch_swoe MIINCOME Dep CCBal MM Income ILS POS NSF CD
        DDA LOC Age Inv InArea AcctAge Moved CRScore MICRScor
```

```
        IRABal MIAcctAg SavBal CashBk DDABal SDB InvBal CCPurc
        ATMAmt Sav CC Phone HMOwn DepAmt IRA MTG ATM LORes;


/* ======================================================== */
/* Lesson 3, Section 4:  l3d5a.sas
  Demonstration: Performing Variable Screening, Part 1
  [m643_4_e1; derived from pmlr03d05.sas]             */
/* ======================================================== */

ods select none;
ods output spearmancorr=work.spearman
       hoeffdingcorr=work.hoeffding;

proc corr data=work.train_imputed_swoe spearman hoeffding;
  var ins;
  with &reduced;
run;

ods select all;

proc sort data=work.spearman;
   by variable;
run;

proc sort data=work.hoeffding;
   by variable;
run;
```

```
data work.correlations;
  merge work.spearman(rename=(ins=scorr pins=spvalue))
       work.hoeffding(rename=(ins=hcorr pins=hpvalue));
  by variable;
  scorr_abs=abs(scorr);
  hcorr_abs=abs(hcorr);
run;

proc rank data=work.correlations out=work.correlations1 descending;
   var scorr_abs hcorr_abs;
   ranks ranksp rankho;
run;

proc sort data=work.correlations1;
  by ranksp;
run;

title1 "Rank of Spearman Correlations and Hoeffding Correlations";
proc print data=work.correlations1 label split='*';
  var variable ranksp rankho scorr spvalue hcorr hpvalue;
  label ranksp ='Spearman rank*of variables'
       scorr  ='Spearman Correlation'
       spvalue='Spearman p-value'
       rankho ='Hoeffding rank*of variables'
       hcorr  ='Hoeffding Correlation'
       hpvalue='Hoeffding p-value';
run;
```

```
/* ====================================================== */
/* Lesson 3, Section 4: l3d5b.sas
   Demonstration: Performing Variable Screening, Part 2
   [m643_4_e2; derived from pmlr03d05.sas]            */
/* ====================================================== */

/* Find values for reference lines */
%global vref href;
proc sql noprint;
   select min(ranksp) into :vref
   from (select ranksp
   from work.correlations1
   having spvalue > .5);

   select min(rankho) into :href
   from (select rankho
   from work.correlations1
   having hpvalue > .5);
quit;

/* Plot variable names, Hoeffding ranks, and Spearman ranks. */

title1 "Scatter Plot of the Ranks of Spearman vs. Hoeffding";
proc sgplot data=work.correlations1;
   refline &vref / axis=y;
   refline &href / axis=x;
   scatter y=ranksp x=rankho / datalabel=variable;
   yaxis label="Rank of Spearman";
   xaxis label="Rank of Hoeffding";
```

```
run;
title1 ;

%global screened;
%let screened=SavBal Dep DDA CD Sav CC ATM MM branch_swoe Phone IRA IRABal
        DDABal ATMAmt ILS POS NSF CCPurc SDB DepAmt CCBal Inv InArea
        Age CashBk MICRScor Income;



/* ===================================================== */
/* Lesson 3, Section 4: l3d6.sas
  Demonstration: Creating Empirical Logit Plots
  [m643_4_i; derived from pmlr03d06.sas]          */
/* ===================================================== */

%global var;
%let var=DDABal;

/* Group the data by the variable of interest in order to create
  empirical logit plots.   */

proc rank data=work.train_imputed_swoe groups=100 out=work.ranks;
  var &var;
  ranks bin;
run;

title1 "Checking Account Balance by Bin";
proc print data=work.ranks(obs=10);
  var &var bin;
```

```
run;

/* The data set BINS will contain:INS=the count of successes in each bin,
  _FREQ_=the count of trials in each bin, DDABAL=the avg DDABAL in each bin. */

proc means data=work.ranks noprint nway;
  class bin;
  var ins &var;
  output out=work.bins sum(ins)=ins mean(&var)=&var;
run;

title1 "Number of Observations, Events, and Average Checking Account Balance by Bin";
proc print data=work.bins(obs=10);
run;

/* Calculate the empirical logit */

data work.bins;
  set work.bins;
  elogit=log((ins+(sqrt(_FREQ_ )/2))/
       ( _FREQ_ -ins+(sqrt(_FREQ_ )/2)));
run;

title1 "Empirical Logit against &var";
proc sgplot data=work.bins;
  reg y=elogit x=&var /
     curvelabel="Linear Relationship?"
     curvelabelloc=outside
     lineattrs=(color=ligr);
```

```
  series y=elogit x=&var;
run;

title1 "Empirical Logit against Binned &var";
proc sgplot data=work.bins;
  reg y=elogit x=bin /
     curvelabel="Linear Relationship?"
     curvelabelloc=outside
     lineattrs=(color=ligr);
  series y=elogit x=bin;
run;


/* ===================================================== */
/* Lesson 3, Section 4: l3d7a.sas
  Demonstration: Accommodating a Nonlinear Relationship,
  Part 1
  [m643_4_m1; derived from pmlr03d07.sas]              */
/* ===================================================== */

title1 "Checking Account Balance and INS by Checking Account";
proc means data=work.train_imputed_swoe mean median min max;
  class dda;
  var ddabal ins;
run;

/* A possible remedy for that non-linearity is to replace the logical
  imputation of 0 for non-DDA customers with the mean. */
```

```
%global mean;
proc sql noprint;
  select mean(ddabal) into :mean
  from work.train_imputed_swoe where dda;
quit;

data work.train_imputed_swoe_dda;
  set work.train_imputed_swoe;
  if not dda then ddabal=&mean;
run;

/* Create new logit plots */
%global var;
%let var=DDABal;

proc rank data=work.train_imputed_swoe_dda groups=100 out=work.ranks;
  var &var;
  ranks bin;
run;

proc means data=work.ranks noprint nway;
  class bin;
  var ins &var;
  output out=work.bins sum(ins)=ins mean(&var)=&var;
run;

/* Calculate the empirical logit */
data work.bins;
  set work.bins;
```

```
  elogit=log((ins+(sqrt(_FREQ_ )/2))/
        ( _FREQ_ -ins+(sqrt(_FREQ_ )/2)));
run;

title1 "Empirical Logit against &var";
proc sgplot data=work.bins;
  reg y=elogit x=&var /
      curvelabel="Linear Relationship?"
     curvelabelloc=outside
     lineattrs=(color=ligr);
  series y=elogit x=&var;
run;

title1 "Empirical Logit against Binned &var";
proc sgplot data=work.bins;
  reg y=elogit x=bin /
      curvelabel="Linear Relationship?"
      curvelabelloc=outside
      lineattrs=(color=ligr);
  series y=elogit x=bin;
run;



/* ===================================================== */
/* Lesson 3, Section 4: l3d7b.sas
  Demonstration: Accommodating a Nonlinear Relationship,
  Part 2
  [m643_4_m2; derived from pmlr03d07.sas]             */
/* ===================================================== */
```

```
/* Using the binned values of DDABal may make for a more linear
   relationship between the input and the target. The following code
   creates DATA step code to bin DDABal, yielding a new predictor, B_DDABal.  */

/* Rank the observations. */

proc rank data=work.train_imputed_swoe_dda groups=100 out=work.ranks;
   var ddabal;
   ranks bin;
run;

/* Save the endpoints of each bin */

proc means data=work.ranks noprint nway;
   class bin;
   var ddabal;
   output out=endpts max=max;
run;

title1 "Checking Account Balance Endpoints";
proc print data=work.endpts(obs=10);
run;

/* Write the code to assign individuals to bins according to the DDABal. */

filename rank "&PMLRfolder/rank.sas";

data _null_;
```

```
  file rank;
  set work.endpts end=last;
  if _n_=1 then put "select;";
  if not last then do;
    put "  when (ddabal <= " max ") B_DDABal =" bin ";";
  end;
  else if last then do;
    put "  otherwise B_DDABal =" bin ";" / "end;";
  end;
run;

/* Use the code. */

data work.train_imputed_swoe_bins;
  set work.train_imputed_swoe_dda;
  %include rank / source;
run;

title1 "Minimum and Maximum Checking Account Balance by Bin";
proc means data=work.train_imputed_swoe_bins min max;
  class B_DDABal;
  var DDABal;
run;
title1 ;

/* Switch the binned DDABal (B_DDABal) for the originally scaled
  DDABal input in the list of potential inputs. */
%global screened;
%let screened=SavBal Dep DDA CD Sav CC ATM MM branch_swoe Phone IRA
```

```
      IRABal B_DDABal ATMAmt ILS POS NSF CCPurc SDB DepAmt
      CCBal Inv InArea Age CashBk MICRScor Income;


/* ===================================================== */
/* Lesson 3, Section 5: l3d8a.sas
  Demonstration: Detecting Interactions
  [m643_5_m; derived from pmlr03d08.sas]            */
/* ===================================================== */

title1 "P-Value for Entry and Retention";

%global sl;
proc sql;
  select 1-probchi(log(sum(ins ge 0)),1) into :sl
  from work.train_imputed_swoe_bins;
quit;

title1 "Interaction Detection using Forward Selection";
proc logistic data=work.train_imputed_swoe_bins;
  class res (param=ref ref='S');
  model ins(event='1')= &screened res
        SavBal|Dep|DDA|CD|Sav|CC|ATM|MM|branch_swoe|Phone|IRA|
        IRABal|B_DDABal|ATMAmt|ILS|POS|NSF|CCPurc|SDB|DepAmt|
        CCBal|Inv|InArea|Age|CashBk|MICRScor|Income|res @2 / include=28 clodds=pl
    selection=forward slentry=&sl;
run;
```

```
/* ===================================================== */
/* Lesson 3, Section 5: l3d8b.sas
   Demonstration: Using Backward Elimination to Subset the
   Variables
   [m643_5_n; derived from pmlr03d08.sas]            */
/* ===================================================== */

title1 "Backward Selection for Variable Annuity Data Set";
proc logistic data=work.train_imputed_swoe_bins;
   class res (param=ref ref='S');
   model ins(event='1')= &screened res SavBal*B_DDABal MM*B_DDABal
            branch_swoe*ATMAmt B_DDABal*Sav SavBal*SDB
            SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt
            SavBal*IRA SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM
            IRA*B_DDABal CD*MM MM*IRABal CD*Sav B_DDABal*CashBk Sav*CC
           / clodds=pl
      selection=backward slstay=&sl hier=single fast;
run;


/* ===================================================== */
/* Lesson 3, Section 5: l3d8c.sas
   Demonstration: Displaying Odds Ratios for Variables
   Involved in Interactions
   [m643_5_o; derived from pmlr03d08.sas]            */
/* ===================================================== */

title1 "Candidate Model for Variable Annuity Data Set";
ods select OddsRatiosPL;
```

```
proc logistic data=work.train_imputed_swoe_bins;
  model ins(event='1')= SavBal Dep DDA CD Sav CC ATM MM branch_swoe IRA B_DDABal
                ATMAmt ILS NSF SDB
                DepAmt Inv SavBal*B_DDABal MM*B_DDABal
                branch_swoe*ATMAmt Sav*B_DDABal
                SavBal*SDB SavBal*DDA AtmAmt*DepAmt B_DDABAL*ATMAmt SavBal*IRA
                SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM IRA*B_DDABal
                CD*MM CD*Sav Sav*CC / clodds=pl;
  oddsratio B_DDABAL / at(savbal=0, 1211, 52299) cl=pl;
run;


/* ===================================================== */
/* Lesson 3, Section 5: l3d8d.sas
  Demonstration: Creating an Interaction Plot
  [m643_5_r; derived from pmlr03d08.sas]          */
/* ===================================================== */

/*----  MACRO INTERACT  ----*\
Reserved data set names: work.percentiles
                work.plot
\*-------------------------*/
%macro interact(data=,target=,event=,inputs=,var1=,var2=,mean_inputs=);

proc logistic data=&data noprint;
  model &target(event="&event")= &inputs;
  code file="&PMLRfolder/interaction.txt";
run;
```

```
proc univariate data=&data noprint;
  var &var1 &var2;
  output out=work.percentiles pctlpts=5 25 50 75 95 pctlpre=&var1._p &var2._p;
run;

data _null_;
  set work.percentiles;
  call symput("&var1._p5",&var1._p5);
  call symput("&var1._p25",&var1._p25);
  call symput("&var1._p50",&var1._p50);
  call symput("&var1._p75",&var1._p75);
  call symput("&var1._p95",&var1._p95);
  call symput("&var2._p5",&var2._p5);
  call symput("&var2._p25",&var2._p25);
  call symput("&var2._p50",&var2._p50);
  call symput("&var2._p75",&var2._p75);
  call symput("&var2._p95",&var2._p95);
run;

proc means data=&data noprint;
  var &mean_inputs;
  output out=work.plot mean=;
run;

data work.plot(drop=_type_ _freq_);
  set work.plot;
  do &var2=&&&var2._p5,&&&var2._p25,&&&var2._p50,&&&var2._p75,&&&var2._p95;
    do &var1=&&&var1._p5,&&&var1._p25,&&&var1._p50,&&&var1._p75,&&&var1._p95;
     %include "&PMLRfolder/interaction.txt";
```

```
      output;
    end;
  end;
run;

title1 "Interaction Plot of &var2 by &var1";
proc sgplot data=work.plot;
  series y=p_&target&event x=&var2 / group=&var1;
  yaxis label="Probability of &target";
run;

%mend interact;

%interact(data=train_imputed_swoe_bins,target=ins,event=1,
          inputs=SavBal Dep DDA CD Sav CC ATM MM branch_swoe
          IRA B_DDABal ATMAmt ILS NSF SDB DepAmt Inv
          SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt Sav*B_DDABal
          SavBal*SDB SavBal*DDA AtmAmt*DepAmt B_DDABAL*ATMAmt SavBal*IRA
          SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM IRA*B_DDABal
          CD*MM CD*Sav Sav*CC,var1=SavBal,var2=B_DDABal,mean_inputs=SavBal Dep
          DDA CD Sav CC ATM MM branch_swoe IRA B_DDABal ATMAmt ILS NSF SDB
          DepAmt Inv);


/* ===================================================== */
/* Lesson 3, Section 5: l3d8e.sas
  Demonstration: Using the Best-Subsets Selection Method
  [m643_5_s; derived from pmlr03d08.sas]             */
/* ===================================================== */
```

```
data work.train_imputed_swoe_bins;
  set work.train_imputed_swoe_bins;
  resr=(res='R');
  resu=(res='U');
run;

/* Run best subsets */
title1 "Models Selected by Best Subsets Selection";
proc logistic data=work.train_imputed_swoe_bins;
   model ins(event='1')=&screened resr resu SavBal*B_DDABal MM*B_DDABal
              branch_swoe*ATMAmt B_DDABal*Sav SavBal*SDB
              SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt
              SavBal*IRA SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM
              IRA*B_DDABal CD*MM MM*IRABal CD*Sav B_DDABal*CashBk Sav*CC
              / selection=score best=1;

run;



/* ===================================================== */
/* Lesson 3, Section 5: l3d8f.sas
   Demonstration: Using Fit Statistics to Select a Model
   [m643_5_L; derived from pmlr03d08.sas]            */
/* ===================================================== */

/* The fitstat macro generates model fit statistics for the
   models selected in the all subsets selection. The macro
   variable IM is set equal to the variable names in the
```

```
  model_indx model while the macro variable IC is set
  equal to the number of variables in the model_indx model. */

%macro fitstat(data=,target=,event=,inputs=,best=,priorevent=);


ods select none;
ods output bestsubsets=work.score;


proc logistic data=&data namelen=50;
  model &target(event="&event")=&inputs / selection=score best=&best;
run;


/* The names and number of variables are transferred to macro
  variables using PROC SQL. */


proc sql noprint;
 select variablesinmodel into :inputs1 -
 from work.score;


 select NumberOfVariables into :ic1 -
 from work.score;
quit;


%let lastindx=&SQLOBS;


%do model_indx=1 %to &lastindx;


%let im=&&inputs&model_indx;
%let ic=&&ic&model_indx;
```

```
ods output scorefitstat=work.stat&ic ;
proc logistic data=&data namelen=50;
  model &target(event="&event")=&im;
  score data=&data out=work.scored fitstat
      priorevent=&priorevent;
run;

proc datasets
   library=work
   nodetails
   nolist;
   delete scored;
run;
quit;

%end;

/* The data sets with the model fit statistics are
   concatenated and sorted by BIC. */

data work.modelfit;
   set work.stat1 - work.stat&lastindx;
   model=_n_;
run;

%mend fitstat;

%fitstat(data=train_imputed_swoe_bins,target=ins,event=1,inputs=&screened resr resu
```

```
        SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt B_DDABal*Sav SavBal*SDB
        SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt SavBal*IRA
        SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM IRA*B_DDABal CD*MM
        MM*IRABal CD*Sav B_DDABal*CashBk Sav*CC,best=1,priorevent=0.02);

proc sort data=work.modelfit;
  by bic;
run;

title1 "Fit Statistics from Models selected from Best-Subsets";
ods select all;
proc print data=work.modelfit;
  var model auc aic bic misclass adjrsquare brierscore;
run;

%global selected;
proc sql;
  select VariablesInModel into :selected
  from work.score
  where numberofvariables=35;
quit;



/* ===================================================== */
/* Lesson 4, Section 1: l4d1.sas
  Demonstration: Preparing the Validation Data
  [m644_1_g; derived from pmlr04d01.sas]           */
/* ===================================================== */
```

```
title1 "Variables with Missing Values on the Validation Data Set";
proc means data=work.valid nmiss;
  var SavBal DDA CD Sav MM IRA IRABal ATMAmt ILS NSF SDB CCBal Inv
     DepAmt Dep ATM CC;
run;

proc univariate data=work.train_imputed_swoe_bins noprint;
  var cc ccbal inv;
  output out=work.medians
        pctlpts=50
        pctlpre=cc ccbal inv;
run;

data work.valid_imputed_swoe_bins(drop=cc50 ccbal50 inv50 i);
  if _N_=1 then set work.medians;
  set work.valid;
  array x(*) cc ccbal inv;
  array med(*) cc50 ccbal50 inv50;
  do i=1 to dim(x);
    if x(i)=. then x(i)=med(i);
  end;
  %include brswoe;
  if not dda then ddabal=&mean;
  %include rank;
run;
```

```
/* ===================================================== */
/* Lesson 4, Section 2: l4d2.sas
  Demonstration: Measuring Model Performance Based on
  Commonly-Used Metrics
  [m644_2_i; derived from pmlr04d02.sas]            */
/* ===================================================== */

ods select roccurve scorefitstat;
proc logistic data=work.train_imputed_swoe_bins;
  model ins(event='1')=&selected;
  score data=work.valid_imputed_swoe_bins out=work.scoval
      priorevent=&pi1 outroc=work.roc fitstat;
run;

title1 "Statistics in the ROC Data Set";
proc print data=work.roc(obs=10);
  var _prob_ _sensit_ _1mspec_;
run;

data work.roc;
  set work.roc;
  cutoff=_PROB_;
  specif=1-_1MSPEC_;
  tp=&pi1*_SENSIT_;
  fn=&pi1*(1-_SENSIT_);
  tn=(1-&pi1)*specif;
  fp=(1-&pi1)*_1MSPEC_;
  depth=tp+fp;
  pospv=tp/depth;
```

```
  negpv=tn/(1-depth);
  acc=tp+tn;
  lift=pospv/&pi1;
  keep cutoff tn fp fn tp
       _SENSIT_ _1MSPEC_ specif depth
       pospv negpv acc lift;
run;

/* Create a lift chart */
title1 "Lift Chart for Validation Data";
proc sgplot data=work.roc;
  where 0.005 <= depth <= 0.50;
  series y=lift x=depth;
  refline 1.0 / axis=y;
  yaxis values=(0 to 9 by 1);
run; quit;
title1 ;



/* ===================================================== */
/* Lesson 4, Section 3: l4d3.sas
  Demonstration: Using a Profit Matrix to Measure Model
  Performance
  [m644_3_i; derived from pmlr04d03.sas]             */
/* ===================================================== */

/* Add the decision variable    */
/* (based on the profit matrix) */
```

```
/* and calculate profit        */
%global rho1;
proc SQL noprint;
  select mean(INS) into :rho1 from pmlr.develop;
quit;

data work.scoval;
   set work.scoval;
   sampwt=(&pi1/&rho1)*(INS)
         + ((1-&pi1)/(1-&rho1))*(1-INS);
   decision=(p_1 > 0.01);
   profit=decision*INS*99
         - decision*(1-INS)*1;
run;

/* Calculate total and average profit */

title1 "Total and Average Profit";
proc means data=work.scoval sum mean;
   weight sampwt;
   var profit;
run;

/* Investigate the true positive and */
/* false positive rates             */
data work.roc;
   set work.roc;
   AveProf=99*tp - 1*fp;
run;
```

```
title1 "Average Profit Against Depth";
proc sgplot data=work.roc;
  series y=aveProf x=depth;
  yaxis label="Average Profit";
run;

title1 "Average Profit Against Cutoff";
proc sgplot data=work.roc;
  where cutoff le 0.05;
  refline .01 / axis=x;
  series y=aveProf x=cutoff;
  yaxis label="Average Profit";
run;


/* ===================================================== */
/* Lesson 4, Section 4: l4d5.sas
  Demonstration: Using the K-S Statistic to Measure Model
  Performance
  [m644_4_g; derived from pmlr04d04.sas]            */
/* ===================================================== */

title1 "K-S Statistic for the Validation Data Set";
proc npar1way edf data=work.scoval;
  class ins;
  var p_1;
run;
```

```
/* ====================================================== */
/* Lesson 4, Section 5: l4d5.sas
   Demonstration: Comparing ROC Curves to Measure Model
   Performance
   [m644_5_g1; derived from pmlr04d05.sas]            */
/* ====================================================== */

proc logistic data=work.train_imputed_swoe_bins noprint;
   class res;
   model ins(event='1')=dda ddabal dep depamt checks res;
   score data=work.valid_imputed_swoe_bins
        out=work.sco_validate(rename=(p_1=p_ch2));
run;

proc logistic data=work.train_imputed_swoe_bins noprint;
   model ins(event='1')=&selected;
   score data=work.sco_validate out=work.sco_validate(rename=(p_1=p_sel));
run;

title1 "Validation Data Set Performance";
ods select ROCOverlay ROCAssociation ROCContrastTest;
proc logistic data=work.sco_validate;
   model ins(event='1')=p_ch2 p_sel / nofit;
   roc "Chapter 2 Model" p_ch2;
   roc "Chapter 4 Model" p_sel;
   roccontrast "Comparing the Two Models";
run;
```

```
/* ===================================================== */
/* Lesson 4, Section 5: l4d6a.sas
   Demonstration: Comparing and Evaluating Many Models, Part 1
   [m644_5_g2; derived from pmlr04d06.sas, pmlr04d06a.sas.,
   pmlr04d06b.sas]
   Note: The code shown in the demonstration video uses
   %INCLUDE statements to include the SAS programs for the
   Assess and Fitandscore macros, respectively. Instead,
   the code below includes all of the code for both macros.
                                     */
/* ===================================================== */

%put &screened;

data work.valid_imputed_swoe_bins;
   set work.valid_imputed_swoe_bins;
   MICRScor=(crscore=.);
   resr=(res='R');
   resu=(res='U');
run;

title1 "Variables with Missing Values on the Validation Data Set";
proc means data=work.valid_imputed_swoe_bins nmiss;
   var &screened;
run;

proc stdize data=work.train_imputed_swoe_bins method=median reponly
         OUTSTAT=med;
   var &screened;
```

```
run;

proc stdize data=work.valid_imputed_swoe_bins reponly method=in(med)
         out=work.valid_imputed_swoe_bins;
  var &screened;
run;

/* Assess macro code */
%macro assess(data=,inputcount=,inputsinmodel=,index=,pi1=,rho1=,target=,
          profit11=,profit01=,profit10=,profit00=);
   %let rho0 = %sysevalf(1-&rho1);
   %let pi0 = %sysevalf(1-&pi1);

   proc sort data=scored&data;
      by descending p_1;
   run;

   /* create assessment data set */
   data assess;
      attrib DATAROLE length=$5;
      retain sse 0 csum 0 DATAROLE "&data";

      /* 2 x 2 count array, or count matrix */
      array n[0:1,0:1] _temporary_ (0 0 0 0);

      /* sample weights array */
      array w[0:1] _temporary_
         (%sysevalf(&pi0/&rho0) %sysevalf(&pi1/&rho1));
      keep DATAROLE INPUT_COUNT INDEX
```

```
  TOTAL_PROFIT OVERALL_AVG_PROFIT ASE C;
set scored&data end=last;

/* profit associated with each decision */
d1=&Profit11*p_1+&Profit01*p_0;
d0=&Profit10*p_1+&Profit00*p_0;

/* T is a flag for response */
t=(strip(&target)="1");

/* D is the decision, based on profit. */
d=(d1>d0);

/* update the count matrix, sse, and c */
n[t,d] + w[t];
sse + (&target-p_1)**2;
csum + ((n[1,1]+n[1,0])*(1-t)*w[0]);

if last then
  do;
    INPUT_COUNT=&inputcount;
    TOTAL_PROFIT = sum(&Profit11*n[1,1],&Profit10*n[1,0],
      &Profit01*n[0,1],&Profit00*n[0,0]);
    OVERALL_AVG_PROFIT = TOTAL_PROFIT/sum(n[0,0],n[1,0],n[0,1],n[1,1]);
    ASE = sse/sum(n[0,0],n[1,0],n[0,1],n[1,1]);
    C = csum/(sum(n[0,0],n[0,1])*sum(n[1,0],n[1,1]));
    index=&index;
    output;
  end;
```

```
  run;

  proc append base=results data=assess force;
  run;

%mend assess;



/* Fitandscore macro code */
/*Usage:*/
/*%fitandscore(data_train=,              training data set*/
/*       data_validate=,           validation data set*/
/*       target=,                  target variable*/
/*       predictors=,              predictor variable list*/
/*       best=,                    # of best subset models to try*/
/*       profit00=,profit01=,profit10=,profit11=, values of the profit matrix*/
/*       pi1=);                    actual population proportion*/

%macro fitandscore(data_train=,
         data_validate=,
         target=,
         predictors=,
         best=,
         profit00=,profit01=,profit10=,profit11=,
         pi1=);

  ods select none;

  ods output bestsubsets=score;
```

```
proc logistic data=&data_train;
   model ins(event='1')=&predictors
      / selection=SCORE best=&best;
run;

%global nmodels;

proc sql;
   select count(*) into :nmodels from score;
run;

%global rho1;

proc sql;
   select mean(INS) into :rho1 from &data_train;
run;

%do i=1 %to &nmodels;
   %global inputs&i;
   %global ic&i;
%end;

proc sql noprint;
   select variablesinmodel into :inputs1 -
      from score;
   select NumberOfVariables into :ic1 -
      from score;
quit;
```

```
proc datasets
   library=work
   nodetails
   nolist;
   delete results;
run;

%do model_indx=1 %to &nmodels;

   %let im=&&inputs&model_indx;
   %let ic=&&ic&model_indx;

   proc logistic data=&data_train;
      model &target(event='1')=&im;
      score data=&data_train
         out=scored&data_train(keep=ins p_1 p_0)
         priorevent=&pi1;
      score data=&data_validate
         out=scored&data_validate(keep=ins p_1 p_0)
         priorevent=&pi1;
   run;

   %assess(data=&data_train,inputcount=&ic,inputsinmodel=&im,index=&model_indx,
      pi1=&pi1,rho1=&rho1,target=&target,profit11=&profit11,profit01=&profit01,
      profit10=&profit10,profit00=&profit00);
   %assess(data=&data_validate,inputcount=&ic,inputsinmodel=&im,index=&model_indx,
      pi1=&pi1,rho1=&rho1,target=&target,profit11=&profit11,profit01=&profit01,
      profit10=&profit10,profit00=&profit00);
```

```
   %end;

   ods select all;

%mend fitandscore;

/* End of macro code */

%fitandscore(data_train=train_imputed_swoe_bins,
         data_validate=valid_imputed_swoe_bins,
         target=ins,predictors=&screened resr resu SavBal*B_DDABal
         MM*B_DDABal branch_swoe*ATMAmt B_DDABal*Sav SavBal*SDB
         SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt
         SavBal*IRA SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM
         IRA*B_DDABal CD*MM MM*IRABal CD*Sav B_DDABal*CashBk Sav*CC,
         best=2,profit00=0,profit01=-1,profit10=0,profit11=99,pi1=0.02);
```

Variables with Missing Values on the Validation Data Set

The MEANS Procedure

Variable	Label	N Miss
SavBal	Saving Balance	0
Dep	Checking Deposits	0
DDA	Checking Account	0
CD	Certificate of Deposit	0
Sav	Saving Account	0
CC	Credit Card	0
ATM	ATM	0
MM	Money Market	0
branch_swoe		0
Phone	Number Telephone Banking	1350
IRA	Retirement Account	0
IRABal	IRA Balance	0
B_DDABal		0
ATMAmt	ATM Withdrawal Amount	0
ILS	Installment Loan	0
POS	Number Point of Sale	1350
NSF	Number Insufficient Fund	0
CCPurc	Credit Card Purchases	1350
SDB	Safety Deposit Box	0
DepAmt	Amount Deposited	0
CCBal	Credit Card Balance	0
Inv	Investment	0
InArea	Local Address	0
Age	Age	2061
CashBk	Number Cash Back	0
MICRScor		0
Income	Income	1866

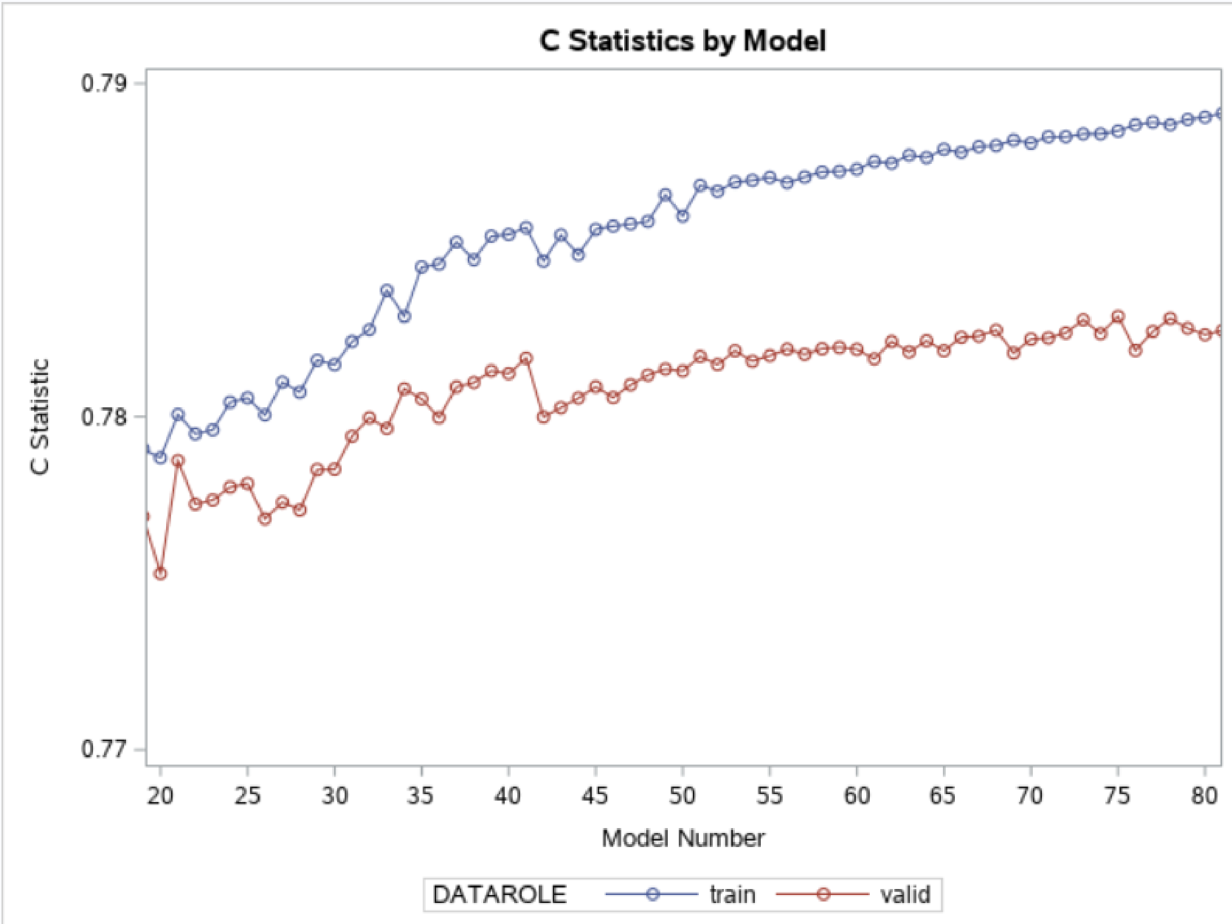
```
/* ===================================================== */
/* Lesson 4, Section 5: l4d6b.sas
   Demonstration: Comparing and Evaluating Many Models, Part 2
   [m644_5_g3; derived from pmlr04d06.sas]            */
/* ===================================================== */

title1 "Model Performance Measures for Training and Validation Data Sets";
proc print data=work.results(obs=24);
run;

title1 "C Statistics by Model";
proc sgplot data=work.results;
```

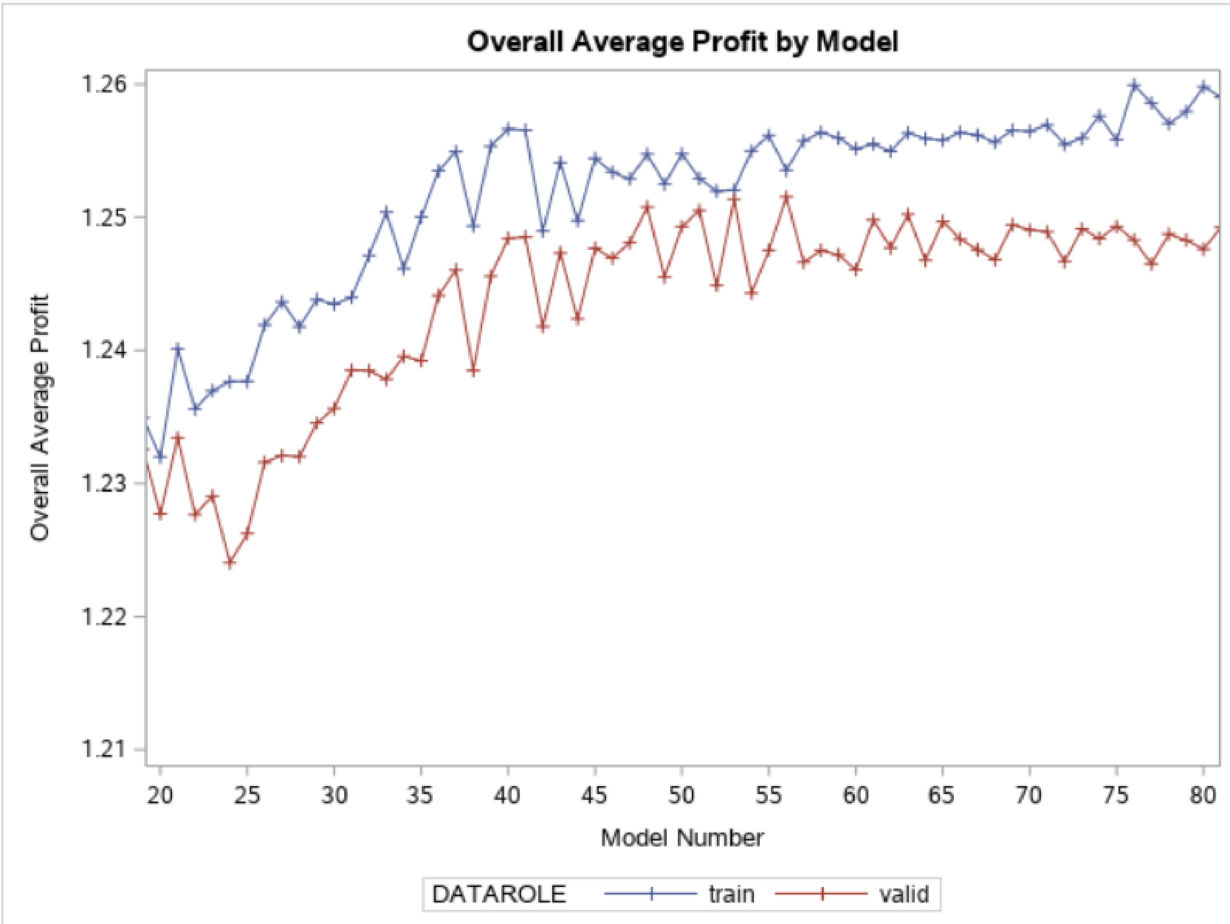
```
  where index > 18;
  series y=c x=index / group=datarole markerattrs=(symbol=circle) markers;
  yaxis label="C Statistic" Values=(0.770 to 0.790 by 0.01);
  xaxis label="Model Number" Values=(20 to 80 by 5);
run;

title1 "Overall Average Profit by Model";
proc sgplot data=work.results;
  where index > 18;
  series y=overall_avg_profit x=index /
        group=datarole markerattrs=(symbol=plus) markers;
  yaxis label="Overall Average Profit" Values=(1.21 to 1.26 by 0.010);
  xaxis label="Model Number" Values=(20 to 80 by 5);
run;

title1 "Model Number with Highest Profit";
%global index;
proc sql;
  select index into :index
  from work.results
  where datarole='valid'
  having overall_avg_profit=max(overall_avg_profit);
quit;

/* Remove all blanks from index */
%let index=%cmpres(&index);
title1 "Logistic Model with Highest Profit";
proc logistic data=work.train_imputed_swoe_bins;
  model ins(event='1')=&&inputs&index;
```

```
score data=work.valid_imputed_swoe_bins out=work.scoval2 fitstat;
run;
```

Model Performance Measures for Training and Validation Data Sets

Obs	DATAROLE	INPUT_COUNT	TOTAL_PROFIT	OVERALL_AVG_PROFIT	ASE	C	index
1	train	1	23129.89	1.07521	0.32753	0.65653	1
2	valid	1	11480.13	1.06770	0.32773	0.65483	1
3	train	1	21511.97	1.00000	0.32920	0.48375	2
4	valid	1	10751.16	0.99991	0.32932	0.47975	2
5	train	2	23760.31	1.10451	0.32416	0.68932	3
6	valid	2	11883.25	1.10520	0.32438	0.68850	3
7	train	2	24629.62	1.14492	0.32564	0.69135	4
8	valid	2	12105.74	1.12589	0.32588	0.68752	4
9	train	3	24966.72	1.16059	0.31447	0.73595	5
10	valid	3	12572.67	1.16931	0.31586	0.73638	5
11	train	3	24852.80	1.15530	0.32238	0.71582	6
12	valid	3	12211.33	1.13571	0.32262	0.71388	6
13	train	4	25865.37	1.20237	0.31313	0.75060	7
14	valid	4	12814.52	1.19181	0.31338	0.74923	7
15	train	4	25413.61	1.18137	0.31422	0.74006	8
16	valid	4	12528.90	1.16524	0.31574	0.73772	8
17	train	5	25843.04	1.20133	0.31157	0.76373	9
18	valid	5	12829.41	1.19319	0.31199	0.75949	9
19	train	5	25824.53	1.20047	0.31372	0.75299	10
20	valid	5	12911.14	1.20079	0.31514	0.75457	10
21	train	6	26009.50	1.20907	0.31205	0.76347	11
22	valid	6	12992.10	1.20832	0.31362	0.76183	11
23	train	6	26543.43	1.23389	0.31013	0.77402	12
24	valid	6	13110.44	1.21933	0.31051	0.77099	12


C Statistics by Model
0.79
0.78
0.77
C Statistic
20
25
30
35
40
45
50
55
60
65
70
75
80
Model Number
DATAROLE
train
valid


Overall Average Profit by Model
1.26
1.25
1.24
1.23
1.22
1.21
Overall Average Profit
20
25
30
35
40
45
50
55
60
65
70
75
80
Model Number
DATAROLE
train
valid

Model Number with Highest Profit

index
56

Logistic Model with Highest Profit

The LOGISTIC Procedure

Model Information	
Data Set	WORK.TRAIN_IMPUTED_SWOE_BINS
Response Variable	Ins
Number of Response Levels	2
Model	binary logit
Optimization Technique	Fisher's scoring

Number of Observations Read	21512
Number of Observations Used	21512

Response Profile		
Ordered Value	Ins	Total Frequency
1	0	14061
2	1	7451

Probability modeled is Ins=1.

Model Convergence Status
Convergence criterion (GCONV=1E-8) satisfied.

Model Fit Statistics		
Criterion	Intercept Only	Intercept and Covariates
AIC	27759.675	22665.951
SC	27767.651	22897.265
-2 Log L	27757.675	22607.951

Testing Global Null Hypothesis: BETA=0			
Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	5149.7241	28	<.0001
Score	4479.4862	28	<.0001
Wald	3438.5476	28	<.0001

Analysis of Maximum Likelihood Estimates					
Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	1	-1.9217	0.0926	430.2719	<.0001
SavBal	1	0.000164	9.545E-6	293.4418	<.0001
DDA	1	-0.2544	0.0491	26.8705	<.0001
CD	1	0.9672	0.0522	343.7165	<.0001
Sav	1	0.8459	0.0844	100.5593	<.0001
MM	1	1.7664	0.1462	146.0518	<.0001
branch_swoe	1	0.9408	0.0795	139.8803	<.0001
IRA	1	1.1164	0.1968	32.1891	<.0001
B_DDABal	1	0.0284	0.000996	811.3105	<.0001
ATMAmt	1	0.000201	0.000029	47.0271	<.0001
ILS	1	-0.2437	0.0766	10.1115	0.0015
NSF	1	0.3651	0.0651	31.5015	<.0001
Inv	1	0.4418	0.0980	20.3042	<.0001
SavBal*B_DDABal	1	-1.7E-6	1.219E-7	193.9633	<.0001
MM*B_DDABal	1	-0.0154	0.00213	52.4092	<.0001
branch_swoe*ATMAmt	1	0.000163	0.000022	52.3780	<.0001
Sav*B_DDABal	1	-0.00984	0.00131	56.7718	<.0001
SavBal*SDB	1	-0.00001	3.76E-6	10.0137	0.0016
SavBal*DDA	1	0.000029	4.514E-6	41.8837	<.0001
ATMAmt*DepAmt	1	-1.68E-9	3.34E-10	25.4236	<.0001
B_DDABal*ATMAmt	1	-1.19E-6	2.386E-7	25.0463	<.0001
SavBal*ATMAmt	1	1.396E-9	6.06E-10	5.3006	0.0213
SavBal*MM	1	-0.00003	6.677E-6	16.9925	<.0001
SavBal*CC	1	-0.00002	4.557E-6	26.1877	<.0001
DDA*ATMAmt	1	0.000076	0.000018	17.3257	<.0001
Dep*ATM	1	-0.1332	0.0120	122.6319	<.0001

Dep*ATM	1	-0.1332	0.0120	122.6319	<.0001
IRA*B_DDABal	1	-0.0117	0.00290	16.2117	<.0001
CD*MM	1	-0.4142	0.1177	12.3753	0.0004
Sav*CC	1	0.3521	0.0528	44.5078	<.0001

Odds Ratio Estimates			
Effect	Point Estimate	95% Wald Confidence Limits	
ILS	0.784	0.674	0.911
NSF	1.441	1.268	1.637
Inv	1.555	1.284	1.885

Association of Predicted Probabilities and Observed Responses			
Percent Concordant	78.7	Somers' D	0.574
Percent Discordant	21.3	Gamma	0.575
Percent Tied	0.0	Tau-a	0.260
Pairs	104768511	c	0.787

Fit Statistics for SCORE Data												
Data Set	Total Frequency	Log Likelihood	Error Rate	AIC	AICC	BIC	SC	R-Square	Max-Rescaled R-Square	AUC	Brier Score
WORK.VALID_IMPUTED_SWOE_BINS	10752	-5750.7	0.2672	11559.46	11559.62	11770.66	11770.66	0.197976	0.273139	0.782223	0.1768