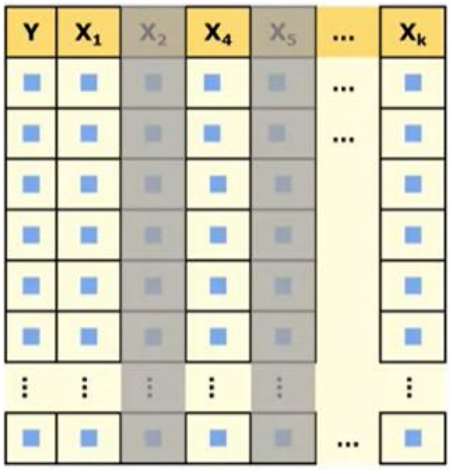
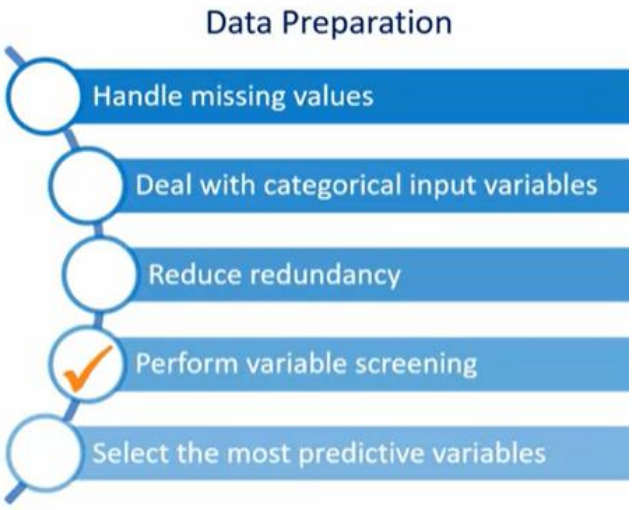
SBA Statistical Business Analyst using SAS

SBA3 Predictive Modeling with Logistic Regression

W4A Performing Variable Screening

Introduction

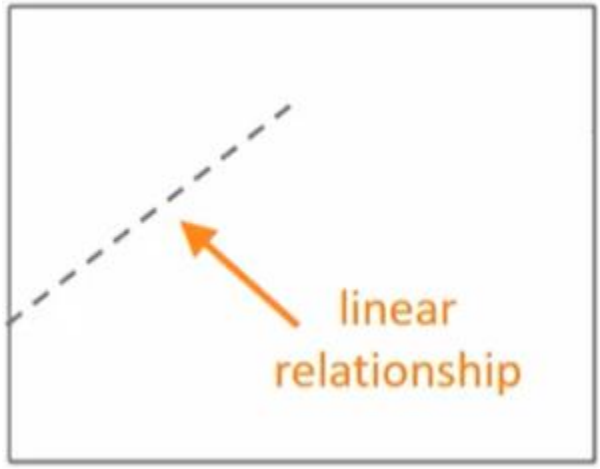


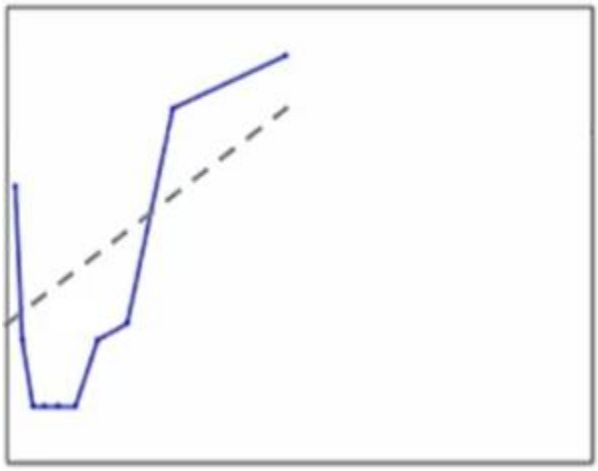


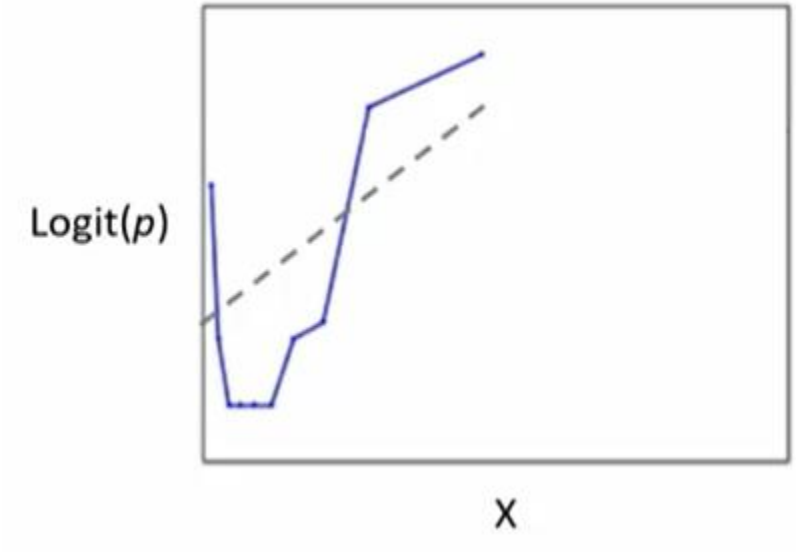
In this topic, you learn to do the following:

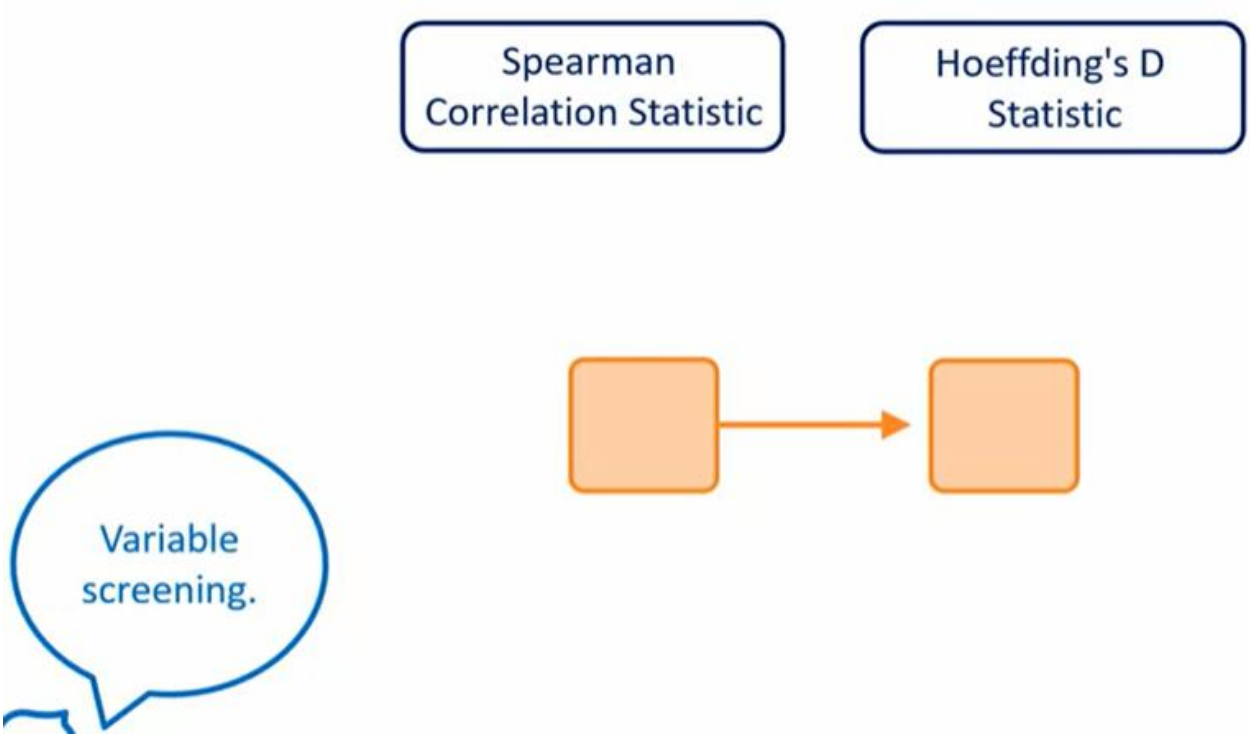
- perform variable screening by using the Spearman and Hoeffding correlation statistics
- create empirical logit plots
- reimpute missing values to accommodate nonlinear relationships between the input variables and the target variable

Detecting Non Linear Relationship

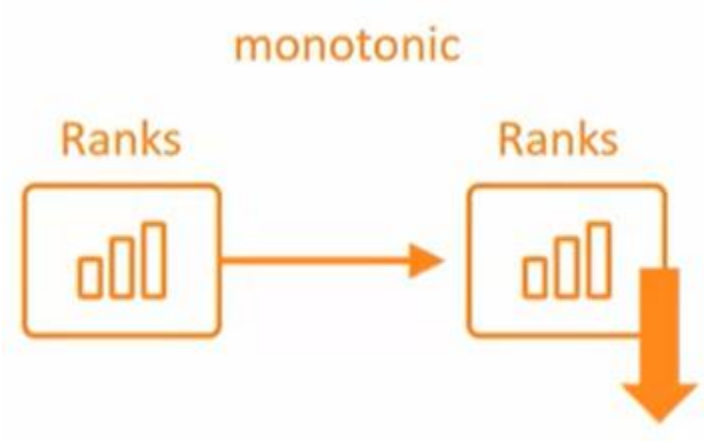


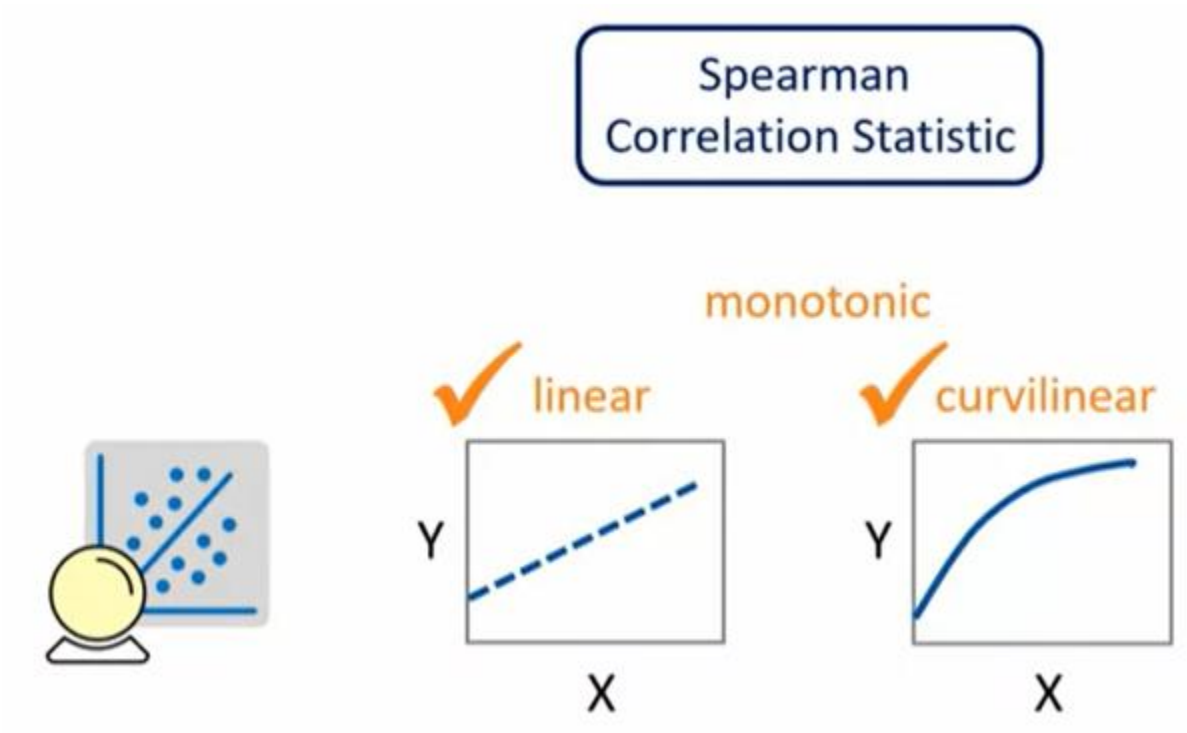


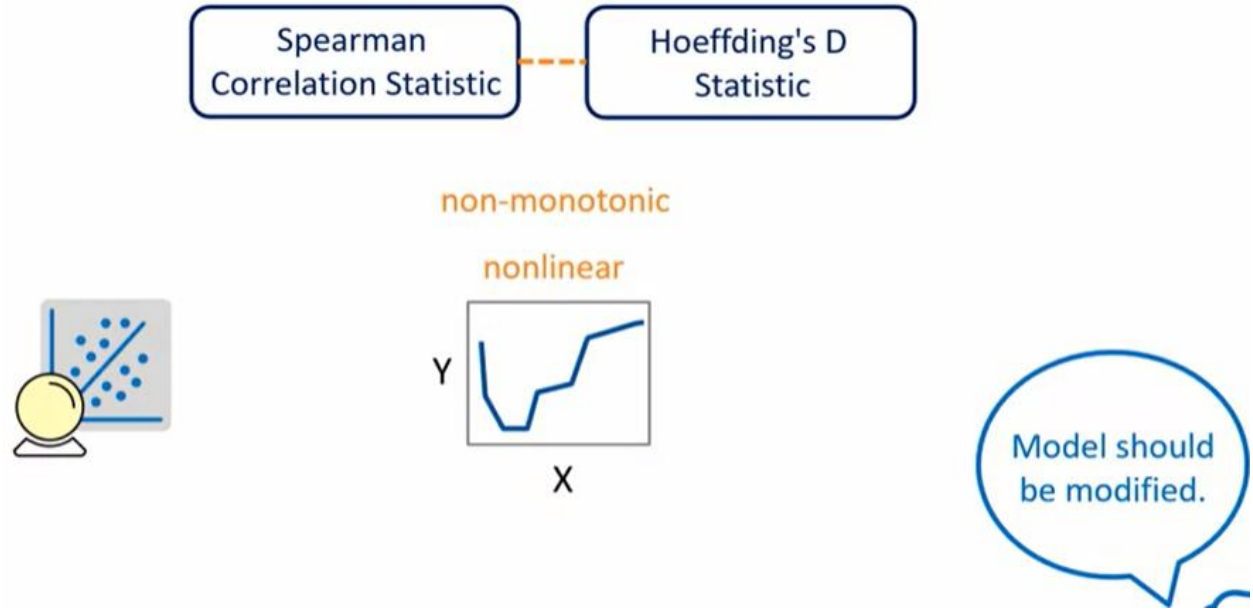


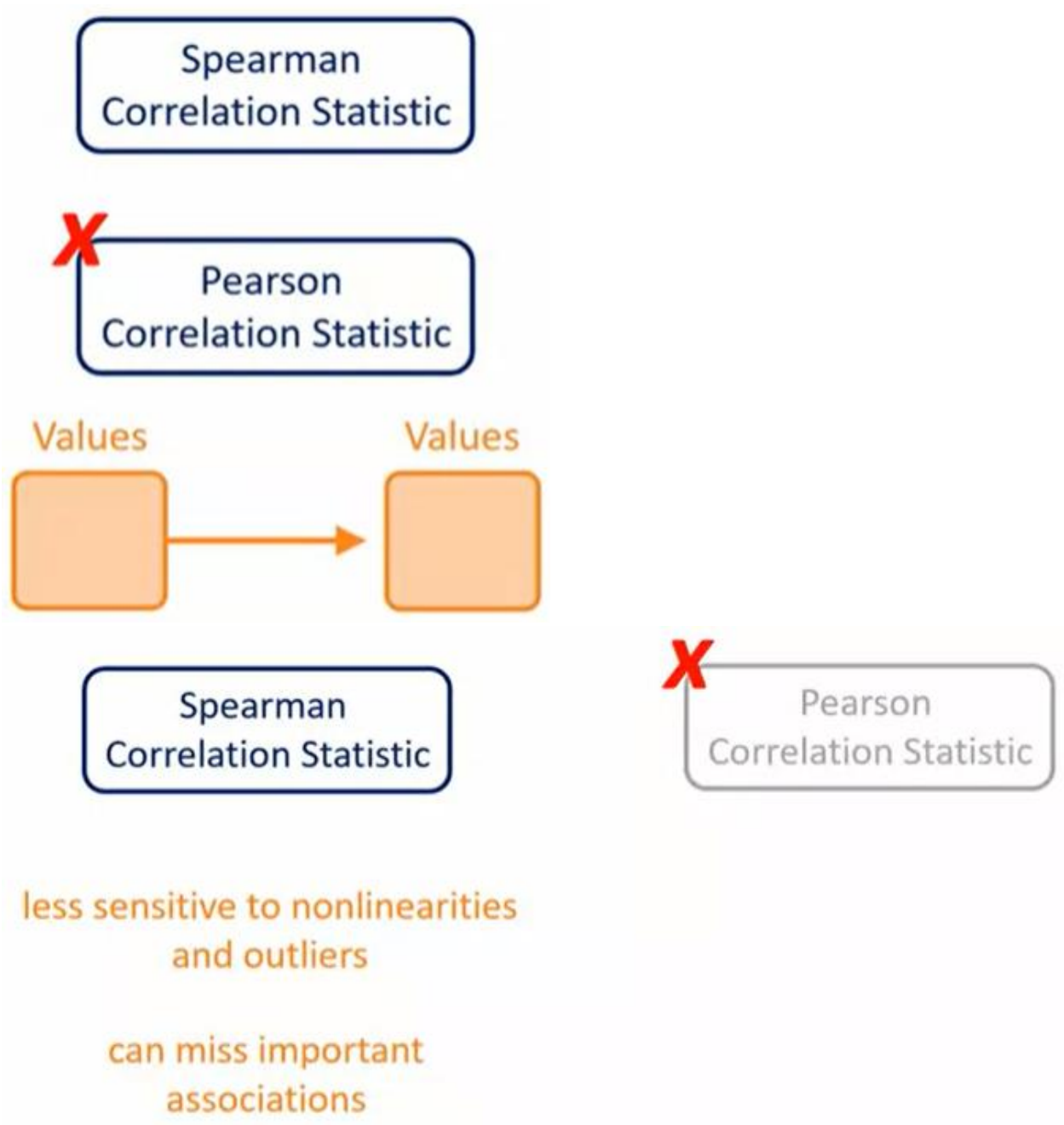

Spearman Correlation Statistic
Hoeffding's D Statistic
Variable screening.

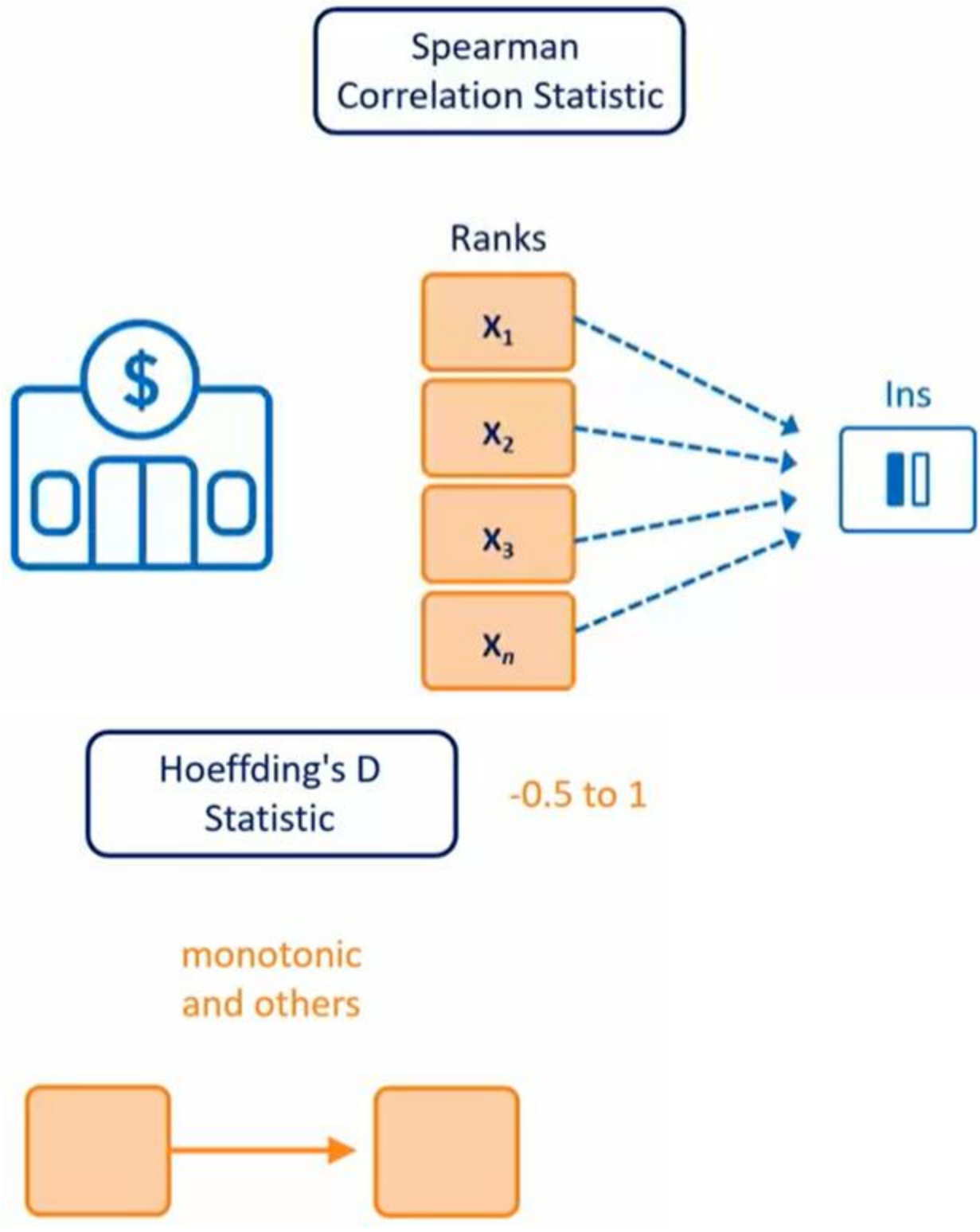

Spearman Correlation Statistic
monotonic
Ranks
Ranks
monotonic
Ranks
Ranks


monotonic
Ranks
Ranks


Spearman
Correlation Statistic
monotonic
linear
curvilinear
Y
X
Y
X


Spearman Correlation Statistic
Hoeffding's D Statistic
non-monotonic
nonlinear
Y
X
Model should be modified.


Spearman Correlation Statistic
Pearson Correlation Statistic
Values
Values
Spearman Correlation Statistic
Pearson Correlation Statistic
less sensitive to nonlinearities and outliers
can miss important associations


Spearman
Correlation Statistic
Ranks
$
x_1
x_2
x_3
x_n
Ins
Hoeffding's D
Statistic
-0.5 to 1
monotonic
and others

Spearman Rank	Hoeffding Rank	Association
High	High	Monotonic
Low	High	Non-monotonic
High	Low	Monotonic
Low	Low	Weak

1, 2, 3, 4, ...

The higher the rank, the lower the number.

Spearman Rank	Hoeffding Rank	Association	
High	High	Monotonic	No problem.
Low	High	Non-monotonic	Investigate.
High	Low	Monotonic	No problem.
Low	Low	Weak	Eliminate.

irrelevant input

Spearman Rank	Hoeffding Rank	Association
Low	Low	Weak

Eliminate.

Demo Performing Variable Screening Part1

* Use the CORR procedure to examine the associations between the inputs chosen from PROC VARCLUS and the target variable.

* Use the SPEARMAN option to request the Spearman correlation statistic and the HOEFFDING option to request Hoeffding's D statistic.

* Create a table that compares the rank order of the Spearman correlation statistic to the rank order of the Hoeffding's D statistic.

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;
```

Variable

Ins

PIns

```
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;
```

Rank of Spearman Correlations and Hoeffding Correlations

Obs	Variable	Spearman rank of variables	Hoeffding rank of variables	Spearman Correlation	Spearman p-value	Hoeffding Correlation	Hoeffding p-value
1	SavBal	1	1	0.25090	<.0001	0.00981	<.0001
2	CD	2	7	0.20283	<.0001	0.00186	<.0001
3	DDA	3	5	-0.19512	<.0001	0.00237	<.0001
4	MM	4	12	0.15949	<.0001	0.00103	<.0001
5	Dep	5	2	-0.15414	<.0001	0.00362	<.0001
6	Sav	6	4	0.15154	<.0001	0.00238	<.0001
7	CC	7	6	0.14636	<.0001	0.00216	<.0001
8	ATM	8	8	-0.12290	<.0001	0.00147	<.0001
9	IRA	9	17	0.11230	<.0001	0.00020	0.0001
10	IRABal	10	18	0.11122	<.0001	0.00018	0.0002
11	Phone	11	14	-0.10133	<.0001	0.00056	<.0001
12	Inv	12	31	0.09852	<.0001	0.00004	0.0583
13	branch_swoe	13	10	0.09398	<.0001	0.00142	<.0001
14	CCPurc	14	16	0.08323	<.0001	0.00023	<.0001
15	POS	15	15	-0.07757	<.0001	0.00041	<.0001
16	SDB	16	19	0.07477	<.0001	0.00017	0.0004
17	CCBal	17	13	0.07023	<.0001	0.00061	<.0001
18	ATMAmt	18	9	-0.06973	<.0001	0.00142	<.0001
19	InvBal	19	32	0.06828	<.0001	-0.00004	1.0000
20	NSF	20	20	-0.06648	<.0001	0.00009	0.0079
21	InArea	21	37	-0.06166	<.0001	-0.00000	0.4178
22	DepAmt	22	11	-0.04875	<.0001	0.00133	<.0001
23	DDABal	23	3	0.04818	<.0001	0.00299	<.0001
24	CashBk	24	28	-0.04601	<.0001	-0.00005	1.0000
25	ILS	25	26	-0.01780	0.0090	-0.00006	1.0000
26	Age	26	36	0.01437	0.0350	-0.00001	0.5353

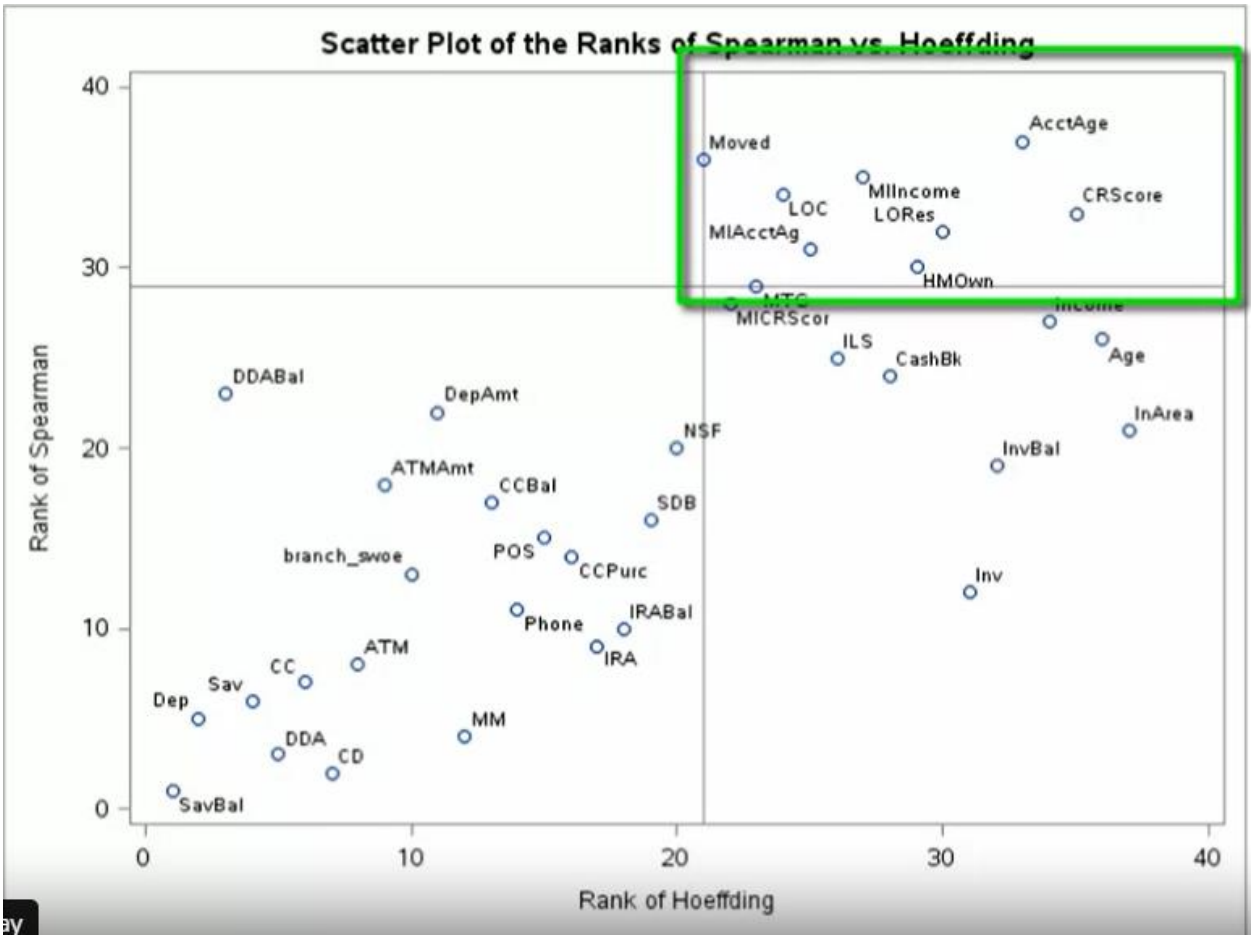
Demo Performing Variable Screening Part2

* Create a scatterplot of the ranks of Spearman versus the ranks of Hoeffding's D using PROC SGPLOT.

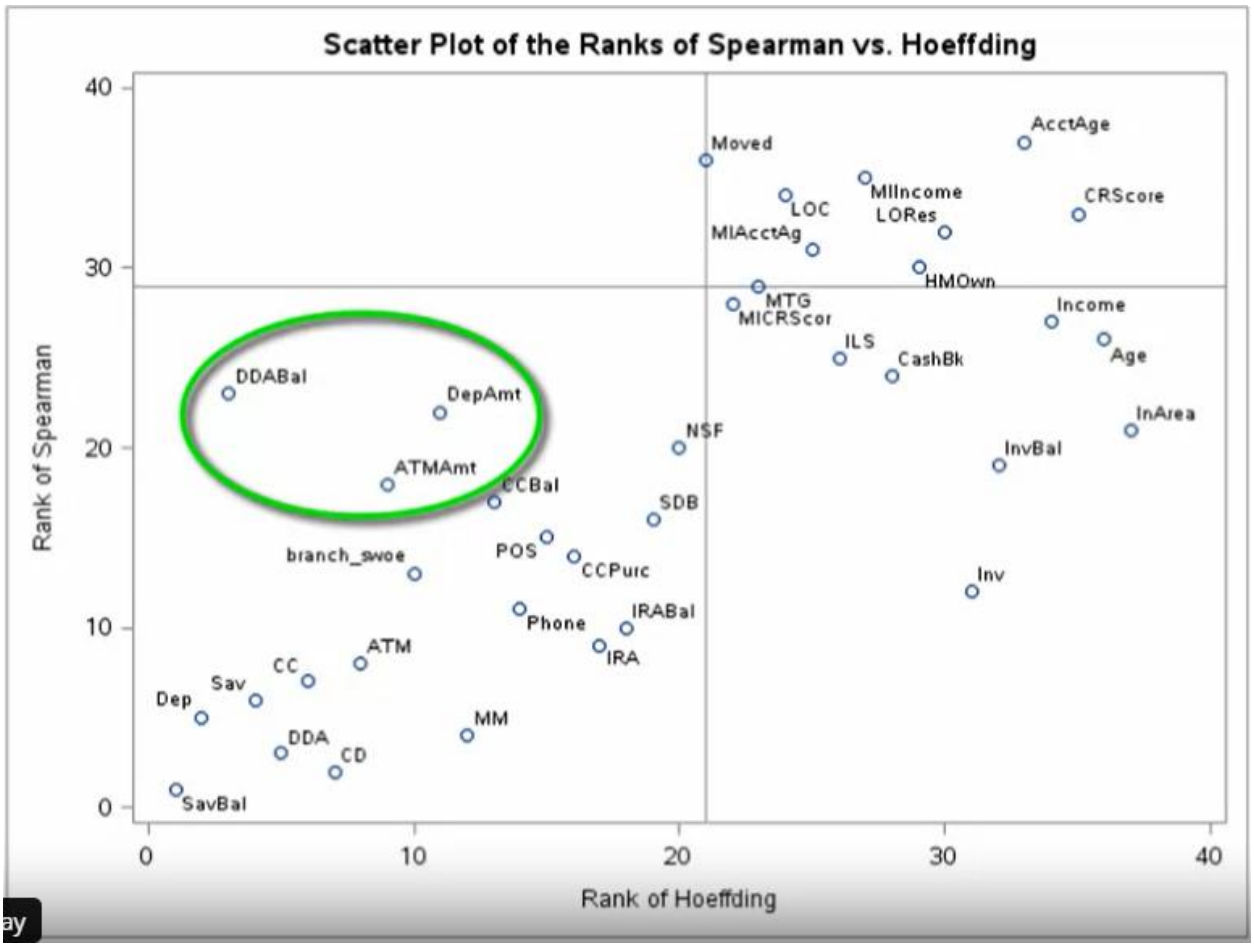
* Add reference lines and data labels to the plot.

```
/* 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 ;
```



The Variables in the upper right corner should be eliminated



DDABal, DeptAmt, and ATMAmt has high rank of Spearman and low rank of Hoeffding, hence they should be investigated.

```
%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;
```

/* Run this code before demo l3d5a */

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

/* Lesson 1, Section 1: l1d1.sas

Demonstration: Examining the Code for Generating

Descriptive Statistics and Frequency Tables */

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

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


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

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

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

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

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

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

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

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

/* 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;
```

Rank of Spearman Correlations and Hoeffding Correlations

Obs	Variable	Spearman rank of variables	Hoeffding rank of variables	Spearman Correlation	Spearman p-value	Hoeffding Correlation	Hoeffding p-value
1	SavBal	1	1	0.25090	<.0001	0.00981	<.0001
2	CD	2	7	0.20283	<.0001	0.00186	<.0001
3	DDA	3	5	-0.19512	<.0001	0.00237	<.0001
4	MM	4	12	0.15949	<.0001	0.00103	<.0001
5	Dep	5	2	-0.15414	<.0001	0.00362	<.0001
6	Sav	6	4	0.15154	<.0001	0.00238	<.0001
7	CC	7	6	0.14636	<.0001	0.00216	<.0001
8	ATM	8	8	-0.12290	<.0001	0.00147	<.0001
9	IRA	9	17	0.11230	<.0001	0.00020	0.0001
10	IRABal	10	18	0.11122	<.0001	0.00018	0.0002
11	Phone	11	14	-0.10133	<.0001	0.00056	<.0001
12	Inv	12	31	0.09852	<.0001	0.00004	0.0583
13	branch_swoe	13	10	0.09398	<.0001	0.00142	<.0001
14	CCPurc	14	16	0.08323	<.0001	0.00023	<.0001
15	POS	15	15	-0.07757	<.0001	0.00041	<.0001
16	SDB	16	19	0.07477	<.0001	0.00017	0.0004
17	CCBal	17	13	0.07023	<.0001	0.00061	<.0001
18	ATMAmt	18	9	-0.06973	<.0001	0.00142	<.0001
19	InvBal	19	32	0.06828	<.0001	-0.00004	1.0000
20	NSF	20	20	-0.06648	<.0001	0.00009	0.0079
21	InArea	21	37	-0.06166	<.0001	-0.00000	0.4178
22	DepAmt	22	11	-0.04875	<.0001	0.00133	<.0001
23	DDABal	23	3	0.04818	<.0001	0.00299	<.0001
24	CashBk	24	28	-0.04601	<.0001	-0.00005	1.0000
25	ILS	25	26	-0.01780	0.0090	-0.00006	1.0000
26	Age	26	36	0.01437	0.0350	-0.00001	0.5353
27	Income	27	34	0.00964	0.1576	-0.00003	1.0000
28	MICRScor	28	22	0.00891	0.1915	-0.00007	1.0000
29	MTG	29	23	-0.00428	0.5299	-0.00006	1.0000
30	HMOwn	30	29	-0.00411	0.5470	-0.00005	1.0000
31	MIAcctAg	31	25	0.00323	0.6353	-0.00006	1.0000
32	LORes	32	30	0.00270	0.6925	-0.00004	1.0000
33	CRScore	33	35	0.00152	0.8231	-0.00003	1.0000
34	LOC	34	24	0.00137	0.8405	-0.00006	1.0000
35	MIIncome	35	27	-0.00085	0.9006	-0.00006	1.0000
36	Moved	36	21	0.00069	0.9193	-0.00007	1.0000
37	AcctAge	37	33	-0.00009	0.9895	-0.00003	1.0000

/* Run this code before doing practice l3p5 */

/* == */

/* Lesson 1, Practice 1

Practice: Exploring the Veterans' Organization Data

```
  Used in the Practices                        */
/* ================================================== */

data pmlr.pva(drop=control_number
              MONTHS_SINCE_LAST_PROM_RESP
              FILE_AVG_GIFT
              FILE_CARD_GIFT);
  set pmlr.pva_raw_data;
  STATUS_FL=RECENCY_STATUS_96NK in("F","L");
  STATUS_ES=RECENCY_STATUS_96NK in("E","S");
  home01=(HOME_OWNER="H");
  nses1=(SES="1");
  nses3=(SES="3");
  nses4=(SES="4");
  nses_=(SES="?");
  nurbr=(URBANICITY="R");
  nurbu=(URBANICITY="U");
  nurbs=(URBANICITY="S");
  nurbt=(URBANICITY="T");
  nurb_=(URBANICITY="?");
run;

proc contents data=pmlr.pva;
run;

proc means data=pmlr.pva mean nmiss max min;
  var _numeric_;
run;
```

```
proc freq data=pmlr.pva nlevels;
  tables _character_;
run;


/* ================================================== */
/* Lesson 1, Practice 2
  Practice: Splitting the Data                        */
/* ================================================== */

proc sort data=pmlr.pva out=work.pva_sort;
  by target_b;
run;

proc surveyselect noprint data=work.pva_sort
              samprate=0.5 out=pva_sample seed=27513
              outall stratumseed=restore;
  strata target_b;
run;

data pmlr.pva_train(drop=selected SelectionProb SamplingWeight)
    pmlr.pva_valid(drop=selected SelectionProb SamplingWeight);
  set work.pva_sample;
  if selected then output pmlr.pva_train;
  else output pmlr.pva_valid;
run;


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

```
/* Lesson 2, Practice 1
  Practice: Fitting a Logistic Regression Model        */
/* ================================================== */

/* Modifications for your SAS software:
  ----------------------------------------------------
  (Optional) To avoid a warning in the log about the
  suppression of plots that have more than 5000
  observations, you can add the MAXPOINTS= option
  to the PROC LOGISTIC statement like this:
  plots(maxpoints=none only). Omitting the
  MAXPOINTS= option does not affect the results
  of the practices in this course.
*/

%global ex_pi1;
%let ex_pi1=0.05;

title1 "Logistic Regression Model of the Veterans' Organization Data";
proc logistic data=pmlr.pva_train plots(only)=
          (effect(clband x=(pep_star recent_avg_gift_amt
          frequency_status_97nk)) oddsratio (type=horizontalstat));
  class pep_star (param=ref ref='0');
  model target_b(event='1')=pep_star recent_avg_gift_amt
             frequency_status_97nk / clodds=pl;
  effectplot slicefit(sliceby=pep_star x=recent_avg_gift_amt) / noobs;
  effectplot slicefit(sliceby=pep_star x=frequency_status_97nk) / noobs;
  score data=pmlr.pva_train out=work.scopva_train priorevent=&ex_pi1;
run;
```

```
title1 "Adjusted Predicted Probabilities of the Veteran's Organization Data";
proc print data=work.scopva_train(obs=10);
  var p_1 pep_star recent_avg_gift_amt frequency_status_97nk;

run;
title;



/* ================================================== */
/* Lesson 3, Practice 1
  Practice: Imputing Missing Values                  */
/* ================================================== */

data pmlr.pva_train_mi(drop=i);
  set pmlr.pva_train;
  /* name the missing indicator variables */
  array mi{*} mi_DONOR_AGE mi_INCOME_GROUP
          mi_WEALTH_RATING;
  /* select variables with missing values */
  array x{*} DONOR_AGE INCOME_GROUP WEALTH_RATING;
  do i=1 to dim(mi);
    mi{i}=(x{i}=.);
    nummiss+mi{i};
  end;
run;

proc rank data=pmlr.pva_train_mi out=work.pva_train_rank
       groups=3;
```

```
  var recent_response_prop recent_avg_gift_amt;
  ranks grp_resp grp_amt;
run;

proc sort data=work.pva_train_rank out=work.pva_train_rank_sort;
  by grp_resp grp_amt;
run;

proc stdize data=work.pva_train_rank_sort method=median
        reponly out=pmlr.pva_train_imputed;
  by grp_resp grp_amt;
  var DONOR_AGE INCOME_GROUP WEALTH_RATING;
run;

options nolabel;
proc means data=pmlr.pva_train_imputed median;
  class grp_resp grp_amt;
  var DONOR_AGE INCOME_GROUP WEALTH_RATING;
run;
options label;



/* ================================================== */
/* Lesson 3, Practice 2
  Practice: Collapsing the Levels of a Nominal Input

  Note: After you submit this code, a note in the log
  indicates that argument 3 to the LOGSDF function
  is invalid. You can ignore this note; it is not
```

```
  important for this analysis. The note pertains
  to the situation in which the number of clusters is 1.
  In this case, the degrees of freedom is 0 (degrees of
  freedom is equal to the number of clusters minus 1) and
  the mathematical operation cannot be performed in the
  LOGSDF function. Therefore, the log of the p-value is
  set to missing.                              */
/* ================================================== */

proc means data=pmlr.pva_train_imputed noprint nway;
  class cluster_code;
  var target_b;
  output out=work.level mean=prop;
run;

ods output clusterhistory=work.cluster;

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

proc freq data=pmlr.pva_train_imputed noprint;
  tables cluster_code*target_b / chisq;
  output out=work.chi(keep=_pchi_) chisq;
run;
```

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

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=-40 max=0;
run;

title1;

%global ncl;

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 cluster_code;
run;

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

title1 "Cluster Assignments";
proc print data=work.clus;
  by clusname;
  id clusname;
run;

filename clcode "&PMLRfolder/cluster_code.sas";

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

data pmlr.pva_train_imputed_clus;
  set pmlr.pva_train_imputed;
  %include clcode;
```

```
run;


/* ================================================== */
/* Lesson 3, Practice 3
   Practice: Computing the Smoothed Weight of Evidence   */
/* ================================================== */

%global rho1_ex;
proc sql noprint;
   select mean(target_b) into :rho1_ex
   from pmlr.pva_train_imputed;
run;

proc means data=pmlr.pva_train_imputed
          sum nway noprint;
   class cluster_code;
   var target_b;
   output out=work.counts sum=events;
run;

filename clswoe "&PMLRfolder/swoe_cluster.sas";

data _null_;
   file clswoe;
   set work.counts end=last;
      logit=log((events + &rho1_ex*24)/
              (_FREQ_ - events + (1-&rho1_ex)*24));
   if _n_=1 then put "select (cluster_code);" ;
```

```
  put "  when ('" cluster_code +(-1) "') cluster_swoe=" logit ";" ;
  if last then do;
    logit=log(&rho1_ex/(1-&rho1_ex));
    put "  otherwise cluster_swoe=" logit ";" / "end;";
  end;
run;

data pmlr.pva_train_imputed_swoe;
  set pmlr.pva_train_imputed;
  %include clswoe;
run;

title;

proc print data=pmlr.pva_train_imputed_swoe(obs=1);
  where cluster_code = "01";
  var cluster_code cluster_swoe;
run;



/* ================================================== */
/* Lesson 3, Practice 4
  Practice: Reducing Redundancy by Clustering Variables */
/* ================================================== */

/*Note: If you run this code in 32-bit SAS, the variable
  assignments to clusters might vary from what is shown
  in the results in this course. This discrepancy does
  not affect the results of the remaining practices in
```

```
  this course.
*/

%let ex_inputs= MONTHS_SINCE_ORIGIN
DONOR_AGE IN_HOUSE INCOME_GROUP PUBLISHED_PHONE
MOR_HIT_RATE WEALTH_RATING MEDIAN_HOME_VALUE
MEDIAN_HOUSEHOLD_INCOME PCT_OWNER_OCCUPIED
PER_CAPITA_INCOME PCT_MALE_MILITARY
PCT_MALE_VETERANS PCT_VIETNAM_VETERANS
PCT_WWII_VETERANS PEP_STAR RECENT_STAR_STATUS
FREQUENCY_STATUS_97NK RECENT_RESPONSE_PROP
RECENT_AVG_GIFT_AMT RECENT_CARD_RESPONSE_PROP
RECENT_AVG_CARD_GIFT_AMT RECENT_RESPONSE_COUNT
RECENT_CARD_RESPONSE_COUNT LIFETIME_CARD_PROM
LIFETIME_PROM LIFETIME_GIFT_AMOUNT
LIFETIME_GIFT_COUNT LIFETIME_AVG_GIFT_AMT
LIFETIME_GIFT_RANGE LIFETIME_MAX_GIFT_AMT
LIFETIME_MIN_GIFT_AMT LAST_GIFT_AMT
CARD_PROM_12 NUMBER_PROM_12 MONTHS_SINCE_LAST_GIFT
MONTHS_SINCE_FIRST_GIFT STATUS_FL STATUS_ES
home01 nses1 nses3 nses4 nses_ nurbr nurbu nurbs
nurbt nurb_;

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

proc varclus data=pmlr.pva_train_imputed_swoe
          hi maxeigen=0.70;
```

```
  var &ex_inputs mi_DONOR_AGE mi_INCOME_GROUP
     mi_WEALTH_RATING cluster_swoe;
run;

ods select all;

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;
  label RSquareRatio="1 - RSquare*Ratio";
run;

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


/* Practice: l3p5.sas step 1 */

%let ex_reduced=
LIFETIME_GIFT_COUNT LAST_GIFT_AMT MEDIAN_HOME_VALUE
FREQUENCY_STATUS_97NK MONTHS_SINCE_LAST_GIFT  nses_
```

```
mi_DONOR_AGE PCT_MALE_VETERANS PCT_MALE_MILITARY
PCT_WWII_VETERANS LIFETIME_AVG_GIFT_AMT cluster_swoe
PEP_STAR nurbu nurbt home01 nurbr DONOR_AGE STATUS_FL
MOR_HIT_RATE nses4 INCOME_GROUP RECENT_STAR_STATUS IN_HOUSE
WEALTH_RATING PUBLISHED_PHONE PCT_OWNER_OCCUPIED nurbs;

/* Solution for l3p5p */

/* step 2 */

%let ex_reduced=
LIFETIME_GIFT_COUNT LAST_GIFT_AMT MEDIAN_HOME_VALUE
FREQUENCY_STATUS_97NK MONTHS_SINCE_LAST_GIFT  nses_
mi_DONOR_AGE PCT_MALE_VETERANS PCT_MALE_MILITARY
PCT_WWII_VETERANS LIFETIME_AVG_GIFT_AMT cluster_swoe
PEP_STAR nurbu nurbt home01 nurbr DONOR_AGE STATUS_FL
MOR_HIT_RATE nses4 INCOME_GROUP RECENT_STAR_STATUS IN_HOUSE
WEALTH_RATING PUBLISHED_PHONE PCT_OWNER_OCCUPIED nurbs;



/* step 3 */

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

proc corr data=pmlr.pva_train_imputed_swoe
       spearman hoeffding;
  var target_b;
```

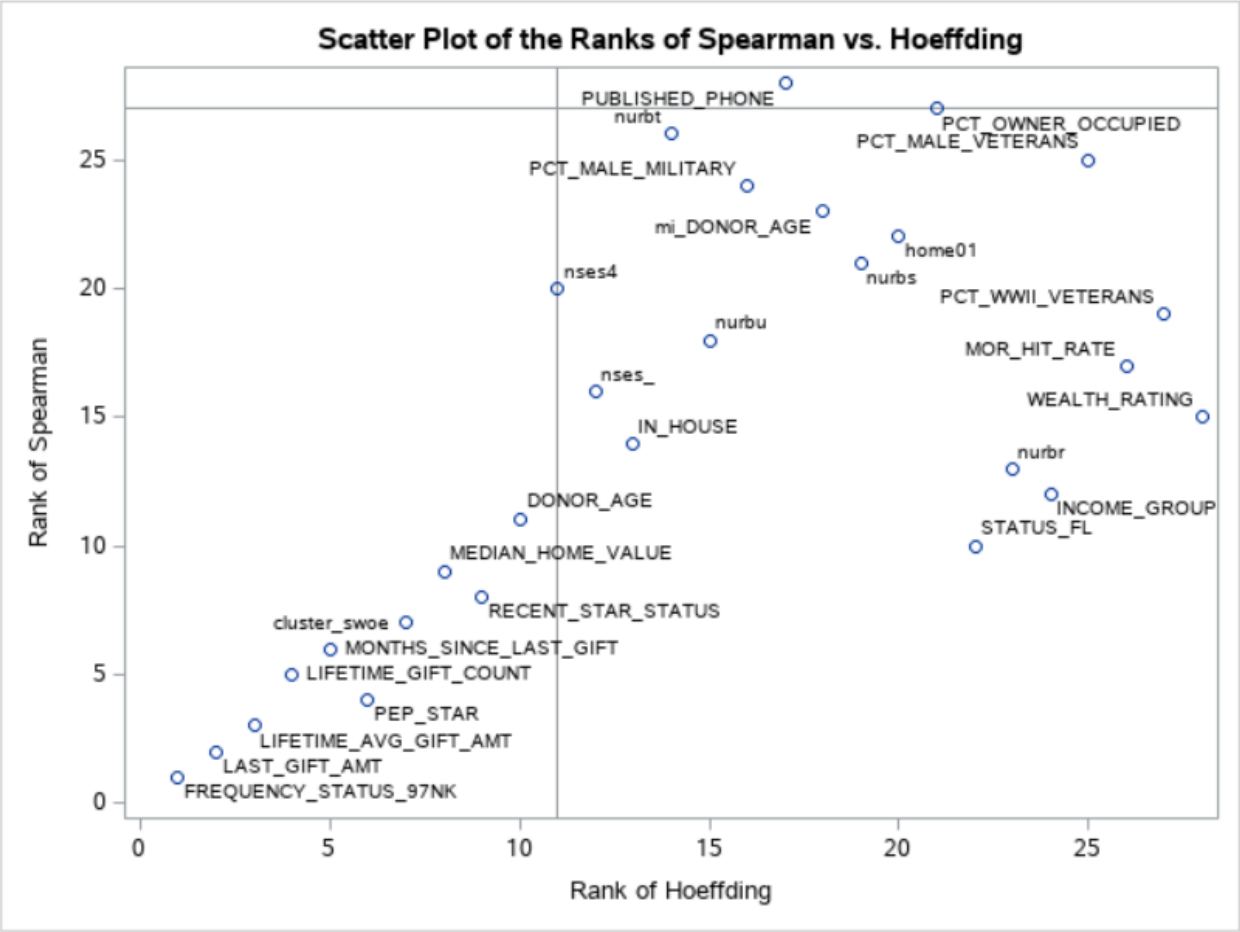
```
  with &ex_reduced;
run;

ods select all;



/* step 4 */

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

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

data work.correlations;
  attrib variable length=$32;
  merge work.spearman(rename=
       (target_b=scorr ptarget_b=spvalue))
       work.hoeffding
       (rename=(target_b=hcorr ptarget_b=hpvalue));
  by variable;
  scorr_abs=abs(scorr);
  hcorr_abs=abs(hcorr);
run;



/* step 5 */
```

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


/* step 6 */

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;


/* step 7 */

%global vref href;
```

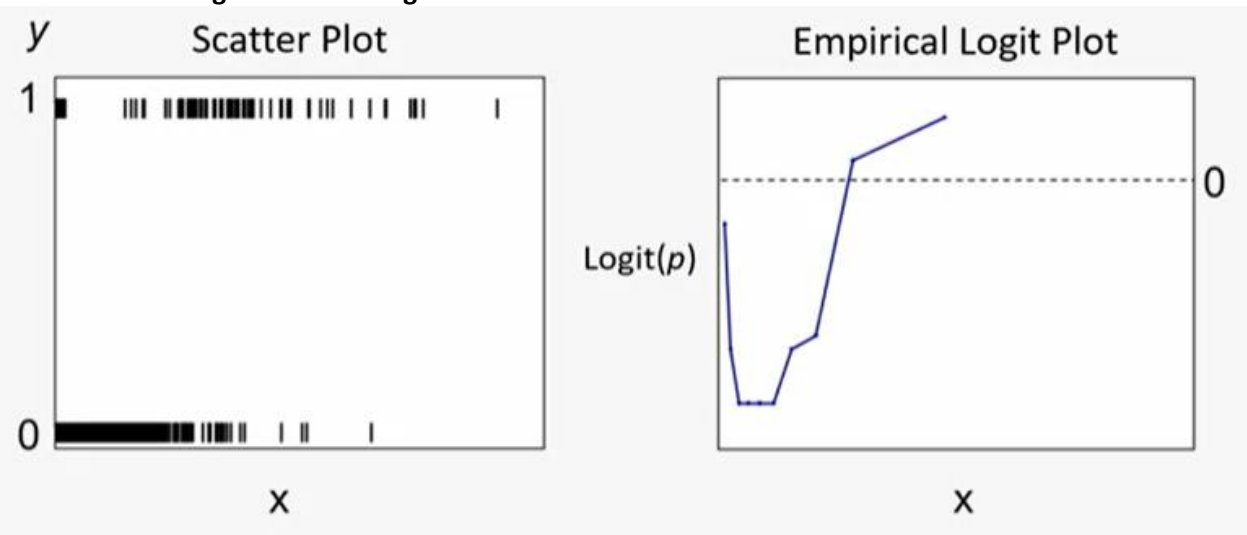
```
proc sql noprint;
  select min(ranksp) into :vref
  from (select ranksp
       from work.correlations1
       having spvalue > .5);
  select min(rankho) into :href
  from (select rankho
       from work.correlations1
       having hpvalue > .5);
quit;

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;
```

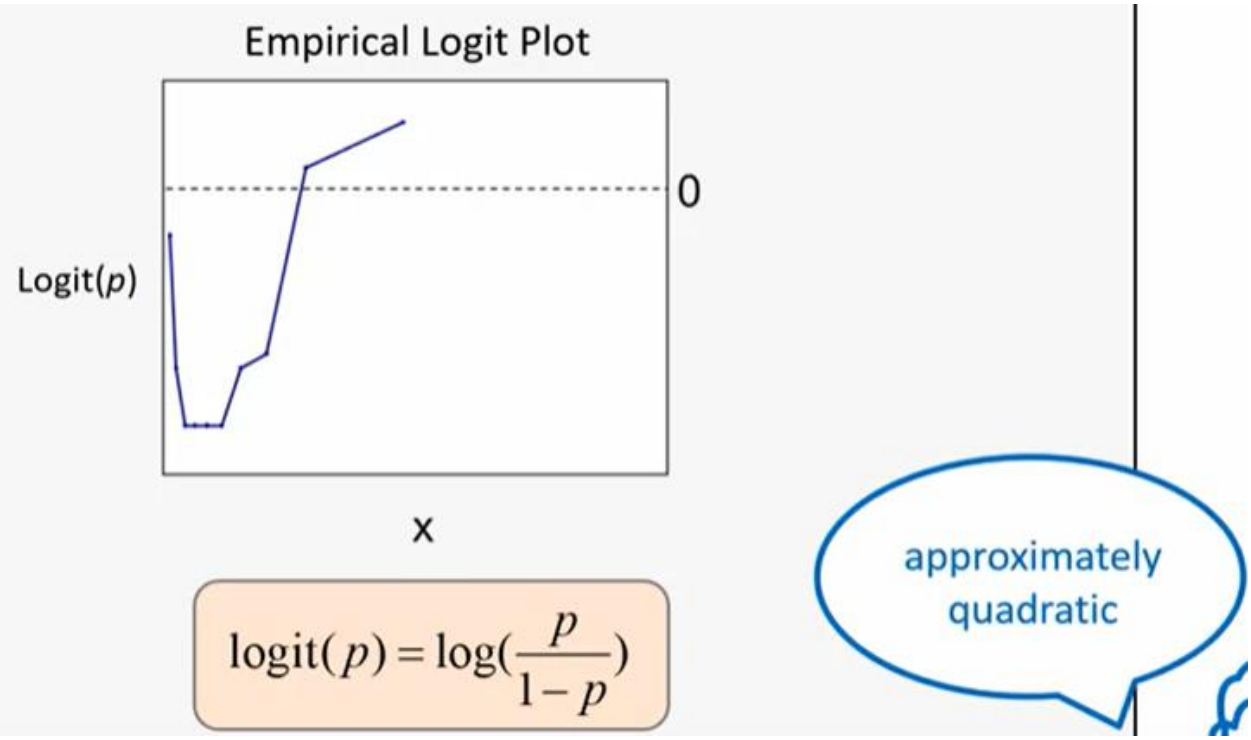
Rank of Spearman Correlations and Hoeffding Correlations

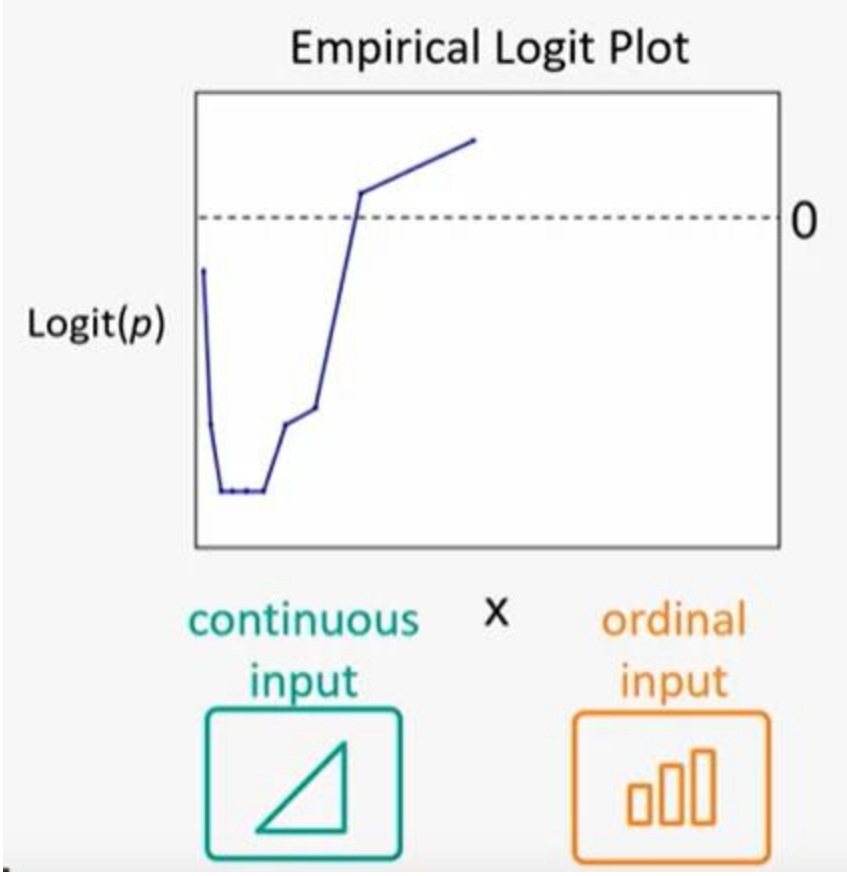
Obs	variable	Spearman rank of variables	Hoeffding rank of variables	Spearman Correlation	Spearman p-value	Hoeffding Correlation	Hoeffding p-value
1	FREQUENCY_STATUS_97NK	1	1	0.13777	<.0001	0.00213	<.0001
2	LAST_GIFT_AMT	2	2	-0.12345	<.0001	0.00197	<.0001
3	LIFETIME_AVG_GIFT_AMT	3	3	-0.11888	<.0001	0.00188	<.0001
4	PEP_STAR	4	6	0.11235	<.0001	0.00099	<.0001
5	LIFETIME_GIFT_COUNT	5	4	0.10943	<.0001	0.00156	<.0001
6	MONTHS_SINCE_LAST_GIFT	6	5	-0.09190	<.0001	0.00103	<.0001
7	cluster_swoe	7	7	0.08150	<.0001	0.00080	<.0001
8	RECENT_STAR_STATUS	8	9	0.07289	<.0001	0.00035	0.0006
9	MEDIAN_HOME_VALUE	9	8	0.06225	<.0001	0.00044	0.0001
10	STATUS_FL	10	22	-0.04935	<.0001	-0.00008	1.0000
11	DONOR_AGE	11	10	0.04266	<.0001	0.00022	0.0055
12	INCOME_GROUP	12	24	0.03859	0.0001	0.00008	0.0674
13	nurbr	13	23	-0.02987	0.0033	-0.00008	1.0000
14	IN_HOUSE	14	13	0.02963	0.0035	-0.00013	1.0000
15	WEALTH_RATING	15	28	0.02757	0.0066	-0.00001	0.4862
16	nses_	16	12	0.02538	0.0125	-0.00015	1.0000
17	MOR_HIT_RATE	17	26	0.02354	0.0205	-0.00006	0.9920
18	nurbu	18	15	-0.02238	0.0277	-0.00012	1.0000
19	PCT_WWII_VETERANS	19	27	0.01851	0.0685	-0.00004	0.8780
20	nses4	20	11	-0.01773	0.0810	-0.00015	1.0000
21	nurbs	21	19	0.01701	0.0942	-0.00011	1.0000
22	home01	22	20	0.01311	0.1970	-0.00010	1.0000
23	mi_DONOR_AGE	23	18	-0.01289	0.2048	-0.00012	1.0000
24	PCT_MALE_MILITARY	24	16	0.01256	0.2165	-0.00012	1.0000
25	PCT_MALE_VETERANS	25	25	0.01171	0.2493	-0.00008	1.0000
26	nurbt	26	14	0.01044	0.3043	-0.00013	1.0000
27	PCT_OWNER_OCCUPIED	27	21	0.00652	0.5211	-0.00010	1.0000
28	PUBLISHED_PHONE	28	17	-0.00268	0.7919	-0.00012	1.0000

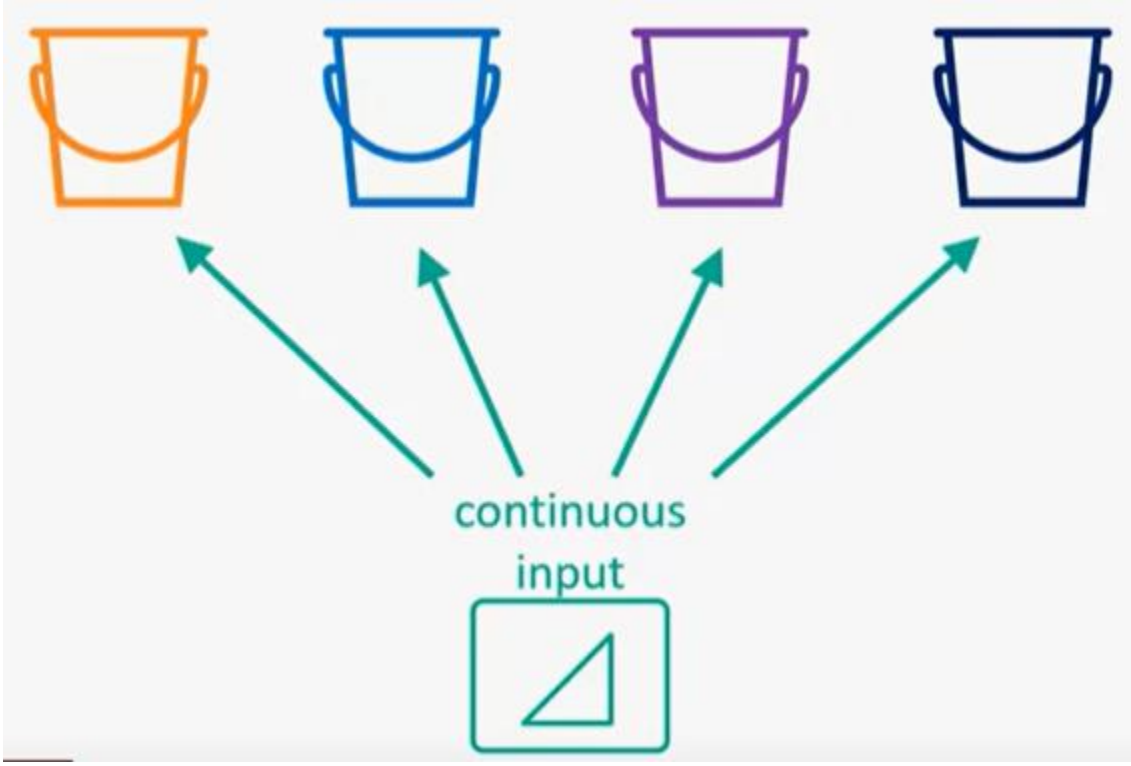

Scatter Plot of the Ranks of Spearman vs. Hoeffding
Rank of Spearman
Rank of Hoeffding
0
5
10
15
20
25
PUBLISHED_PHONE
nurbt
PCT_OWNER_OCCUPIED
PCT_MALE_VETERANS
PCT_MALE_MILITARY
mi_DONOR_AGE
home01
nurbs
nses4
PCT_WWII_VETERANS
nurbu
MOR_HIT_RATE
nses_
WEALTH_RATING
IN_HOUSE
nurbr
INCOME_GROUP
DONOR_AGE
STATUS_FL
MEDIAN_HOME_VALUE
RECENT_STAR_STATUS
cluster_swoe
MONTHS_SINCE_LAST_GIFT
LIFETIME_GIFT_COUNT
PEP_STAR
LIFETIME_AVG_GIFT_AMT
LAST_GIFT_AMT
FREQUENCY_STATUS_97NK

Univariate Binning and Smoothing

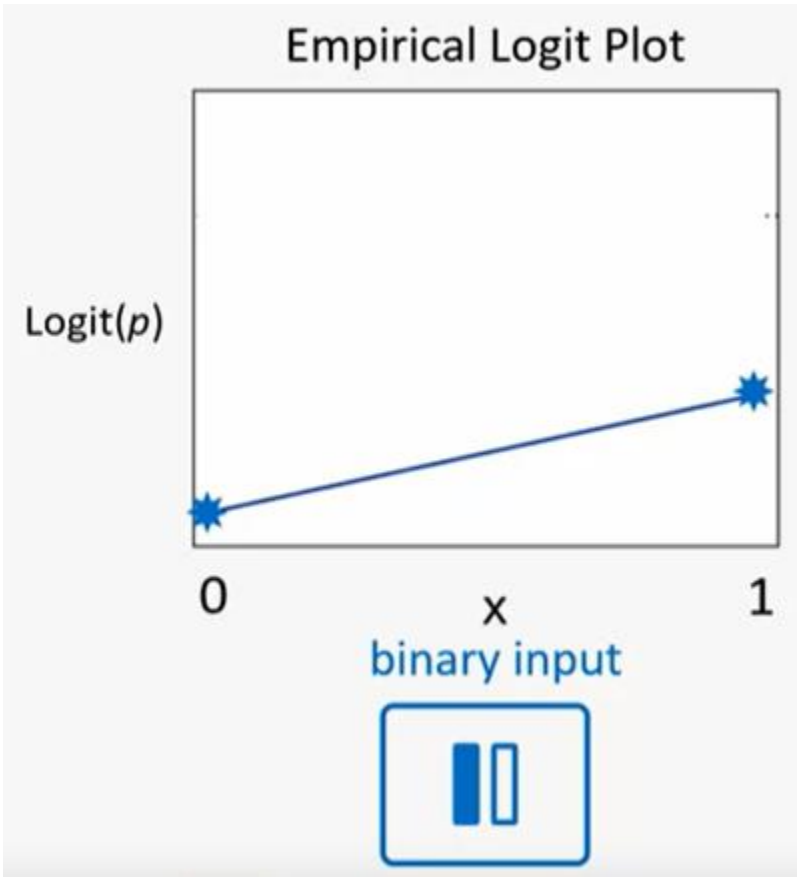





Empirical Logit Plot
0
Logit(p)
continuous input
x
ordinal input


continuous input

Bin	Min X	Max X	Number of Cases	Number of Events	Empirical *p*	Empirical Logit	Smoothed Logit
1	0	50	10000	500	0.05	-1.28	-1.24
2	51	91	10000	10000	0.10	-0.95	-0.94
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
9	321	360	10000	8200	0.82	0.66	0.65
10	361	400	10000	9000	0.90	0.95	0.94



$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right)$$

$$\log\left(\frac{1}{1-1}\right) = \infty$$

$$\log\left(\frac{0}{1-0}\right) = -\infty$$

When p = 1 or 0, the logit is infinite.

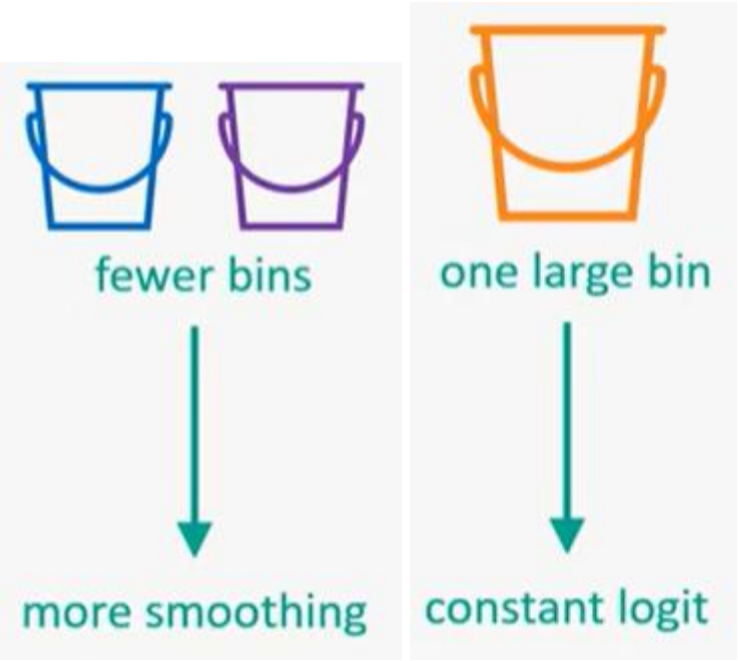
Empirical Logits for Quantiles

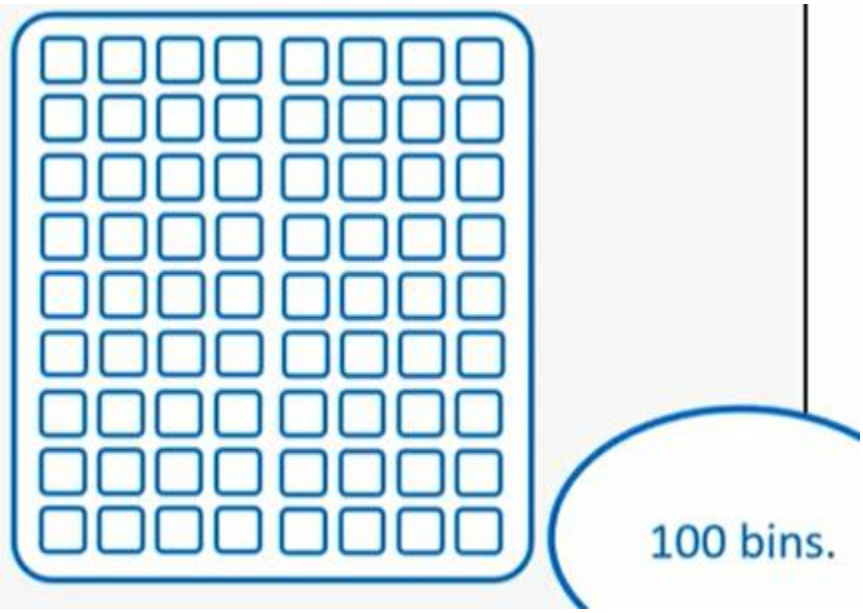
$$\ln\left(\frac{m_i + \frac{\sqrt{M_i}}{2}}{M_i - m_i + \frac{\sqrt{M_i}}{2}}\right)$$

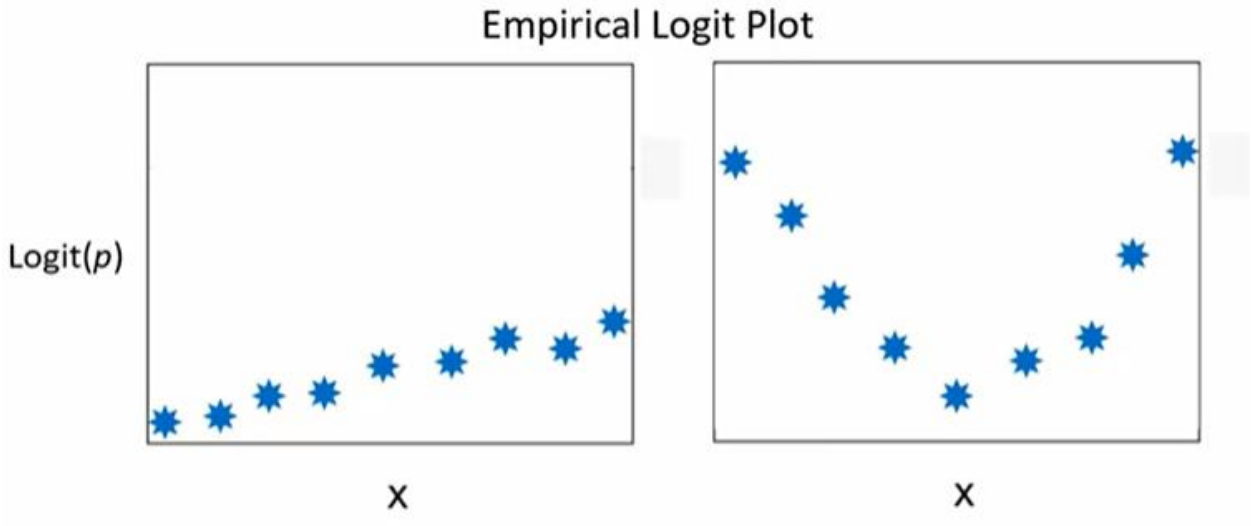
where
m_i = number of events
M_i = number of cases

number of bins

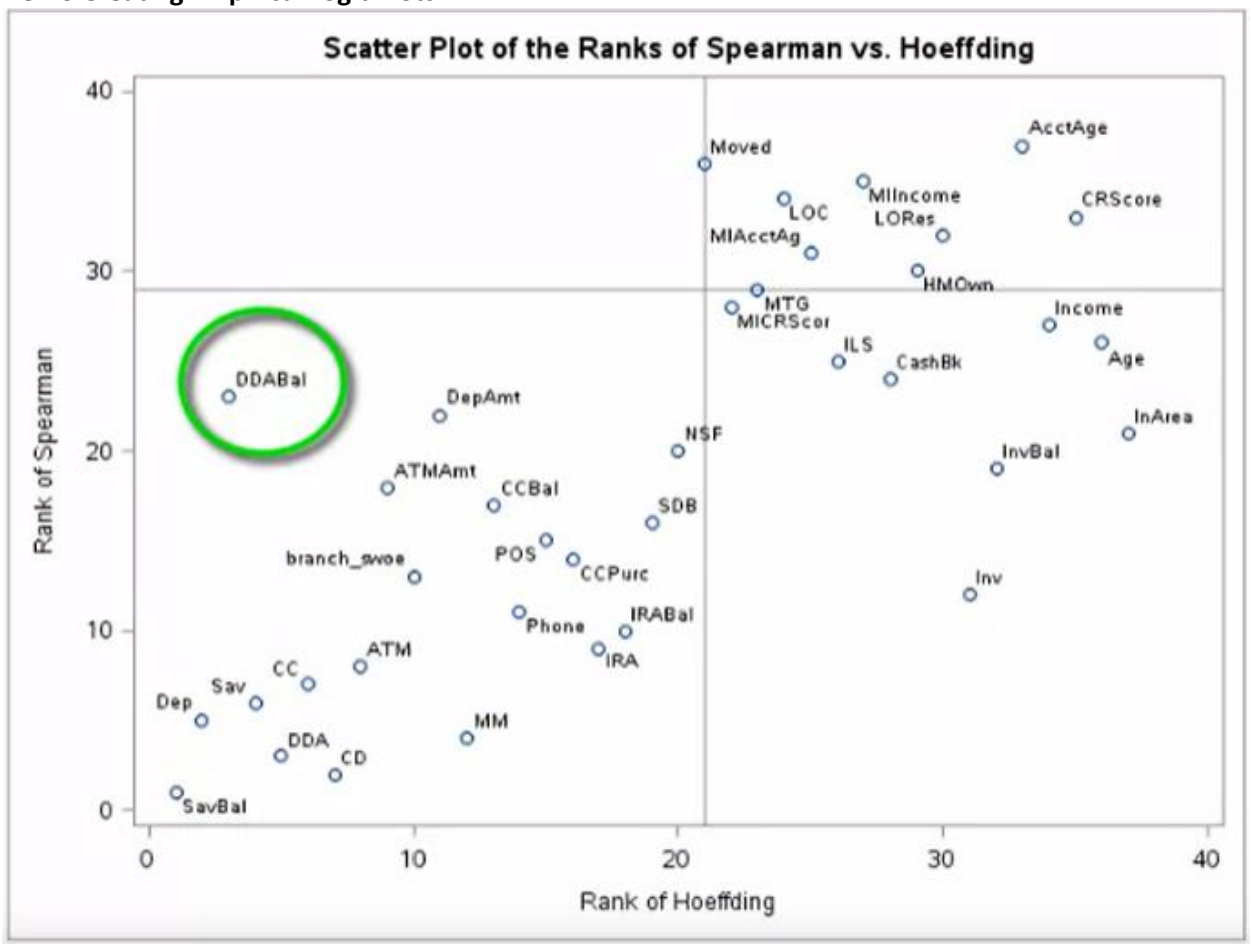
amount of smoothing


fewer bins
one large bin
more smoothing
constant logit


100 bins.



Demo Creating Empirical Logit Plots



* Bin the values of DDABal (checking account balance) using PROC RANK.

* Calculate the empirical logit values using PROC MEANS and a DATA step.

* Create empirical logit plots using PROC SGPLOT.

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;
```

Checking Account Balance by Bin

Obs	DDABal	bin
1	1986.81	76
2	1594.84	71
3	1437.57	69
4	190.03	33
5	1772.13	73
6	375.62	42
7	324.94	40
8	13.85	21
9	9644.48	95
10	284.88	38

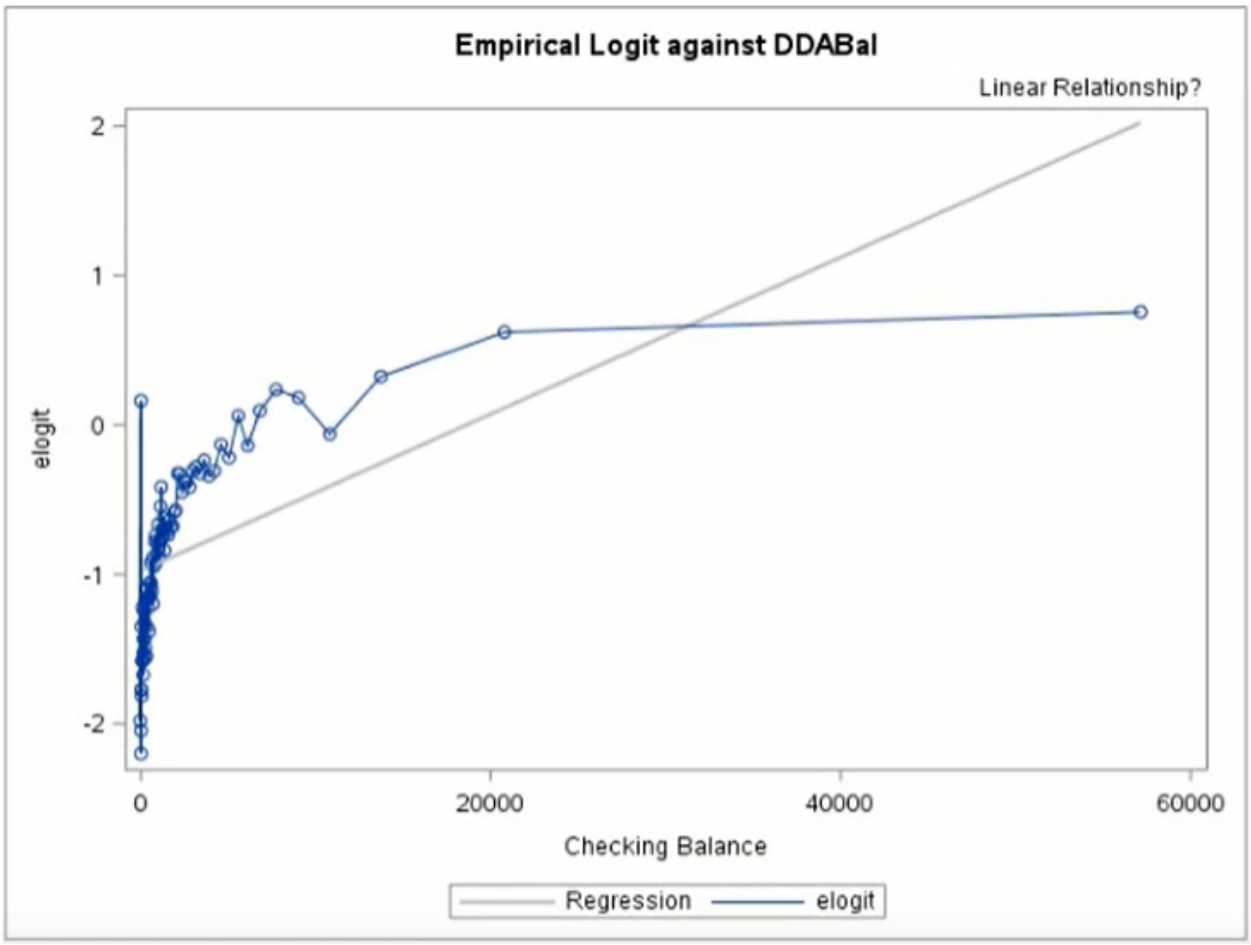
```
proc means data=work.ranks noprint nway;
   class bin;
   var ins &var;
   output out=work.bins sum(ins)=ins mean(&var)=&var;
run;

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

Number of Observations, Events, and Average Checking Account Balance by Bin

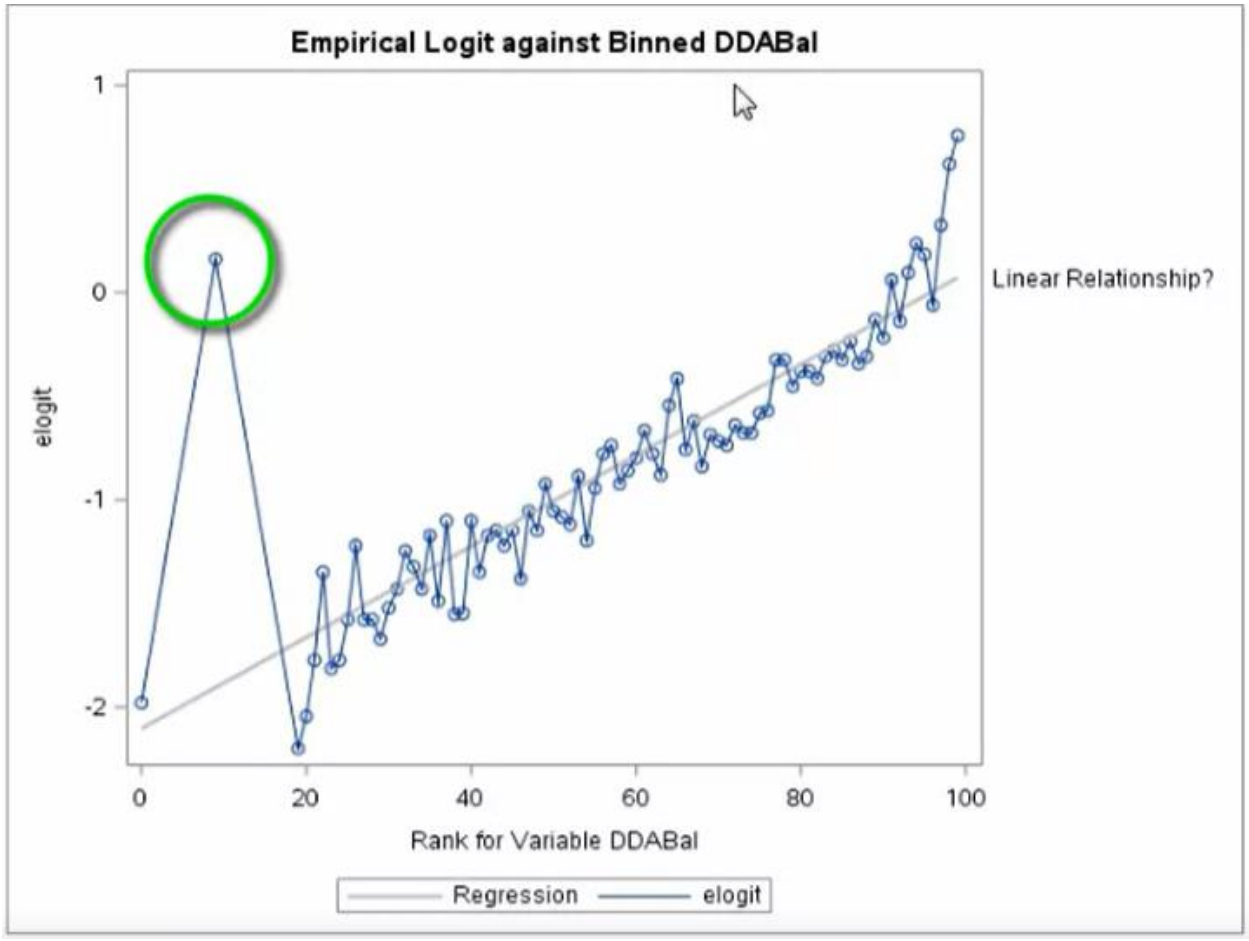
Obs	bin	TYPE	_FREQ_	ins	DDABal
1	0	1	135	12	-32.2597
2	9	1	3994	2161	0.0000
3	19	1	173	12	2.8347
4	20	1	215	19	8.8542
5	21	1	215	26	17.2156
6	22	1	215	40	28.3862
7	23	1	216	25	41.2978
8	24	1	215	26	53.9779
9	25	1	215	32	68.1660
10	26	1	215	45	83.2786

```
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;
```




```
/* Run this code before demo l3d6 */

/* ================================================== */
/* Lesson 1, Section 1: l1d1.sas
   Demonstration: Examining the Code for Generating
   Descriptive Statistics and Frequency Tables          */
/* ================================================== */

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


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

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

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

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

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

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

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

/* 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 */
```

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

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

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

/* 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
```

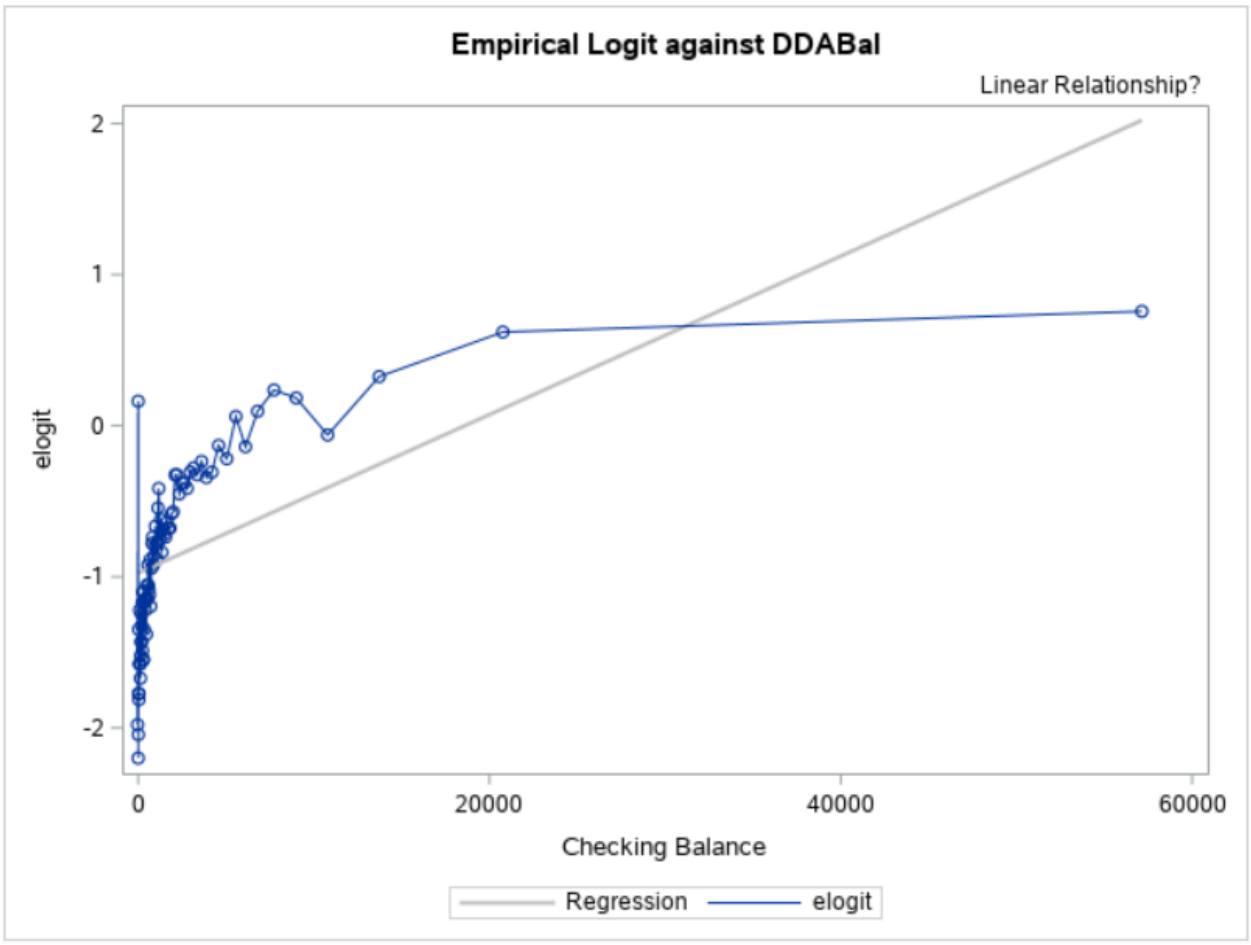
```
    lineattrs=(color=ligr);
  series y=elogit x=&var;
run;


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;
```

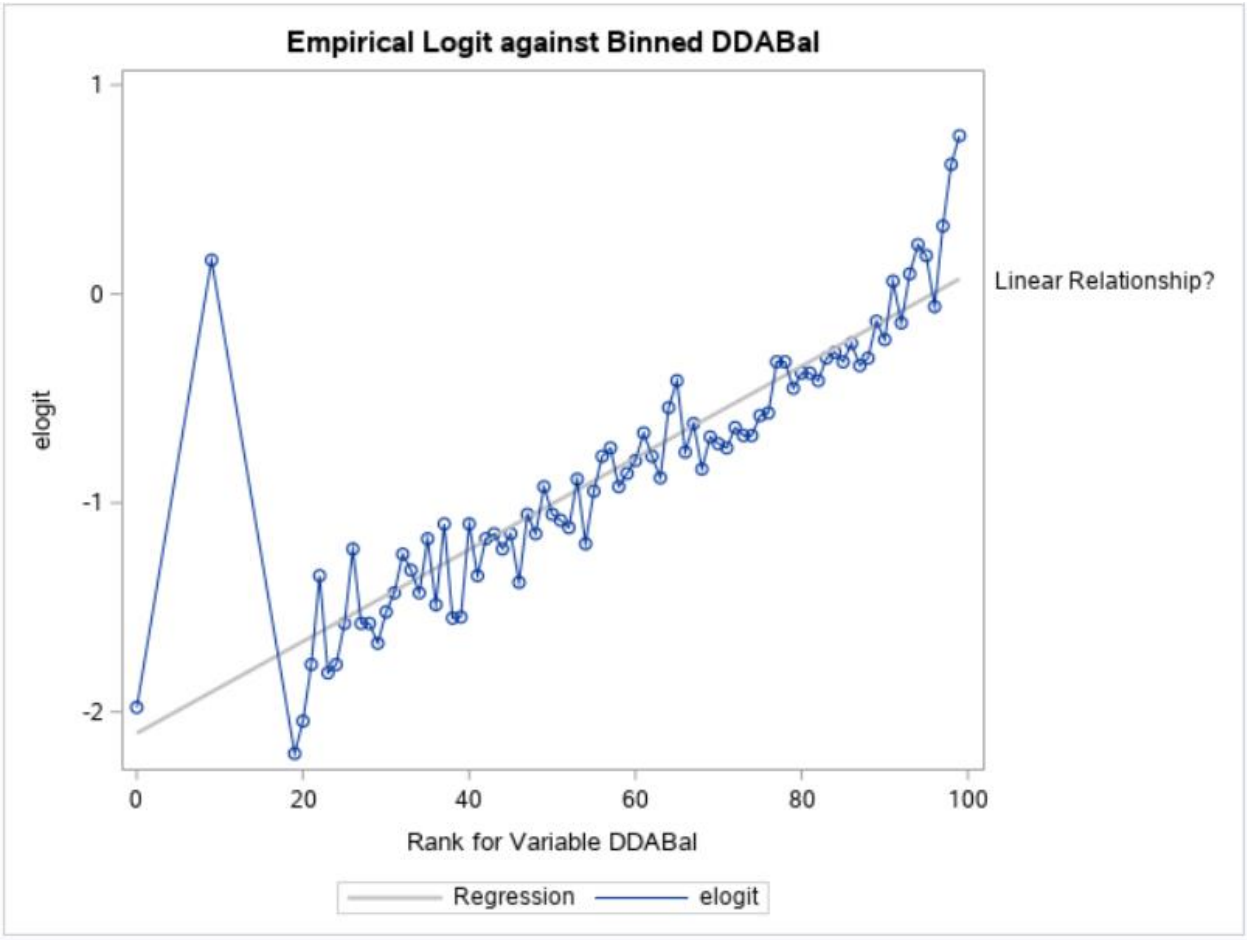
Checking Account Balance by Bin

Obs	DDABal	bin
1	1986.81	76
2	1594.84	71
3	1437.57	69
4	190.03	33
5	1772.13	73
6	375.62	42
7	324.94	40
8	13.85	21
9	9644.48	95
10	284.88	38

Number of Observations, Events, and Average Checking Account Balance by Bin

Obs	bin	_TYPE_	_FREQ_	ins	DDABal
1	0	1	135	12	-32.2597
2	9	1	3994	2161	0.0000
3	19	1	173	12	2.8347
4	20	1	215	19	8.8542
5	21	1	215	26	17.2156
6	22	1	215	40	28.3862
7	23	1	216	25	41.2978
8	24	1	215	26	53.9779
9	25	1	215	32	68.1660
10	26	1	215	45	83.2786


Empirical Logit against DDABal
Linear Relationship?
2
1
0
-1
-2
elogit
0
20000
40000
60000
Checking Balance
Regression
elogit



/* Run this code before doing practice l3p6 */

/* == */

/* Lesson 1, Practice 1

Practice: Exploring the Veterans' Organization Data

Used in the Practices */

/* == */

data pmlr.pva(drop=control_number

MONTHS_SINCE_LAST_PROM_RESP

```
            FILE_AVG_GIFT

            FILE_CARD_GIFT);
  set pmlr.pva_raw_data;
  STATUS_FL=RECENCY_STATUS_96NK in("F","L");
  STATUS_ES=RECENCY_STATUS_96NK in("E","S");
  home01=(HOME_OWNER="H");
  nses1=(SES="1");
  nses3=(SES="3");
  nses4=(SES="4");
  nses_=(SES="?");
  nurbr=(URBANICITY="R");
  nurbu=(URBANICITY="U");
  nurbs=(URBANICITY="S");
  nurbt=(URBANICITY="T");
  nurb_=(URBANICITY="?");
run;


proc contents data=pmlr.pva;
run;


proc means data=pmlr.pva mean nmiss max min;
  var _numeric_;
run;
```

```
proc freq data=pmlr.pva nlevels;
  tables _character_;
run;



/* ================================================== */
/* Lesson 1, Practice 2
  Practice: Splitting the Data                    */
/* ================================================== */

proc sort data=pmlr.pva out=work.pva_sort;
  by target_b;
run;

proc surveyselect noprint data=work.pva_sort
            samprate=0.5 out=pva_sample seed=27513
            outall stratumseed=restore;
  strata target_b;
run;

data pmlr.pva_train(drop=selected SelectionProb SamplingWeight)
    pmlr.pva_valid(drop=selected SelectionProb SamplingWeight);
  set work.pva_sample;
  if selected then output pmlr.pva_train;
```

```
  else output pmlr.pva_valid;
run;



/* ================================================== */
/* Lesson 2, Practice 1
  Practice: Fitting a Logistic Regression Model       */
/* ================================================== */

/* Modifications for your SAS software:
  ------------------------------------------------------
  (Optional) To avoid a warning in the log about the
  suppression of plots that have more than 5000
  observations, you can add the MAXPOINTS= option
  to the PROC LOGISTIC statement like this:
  plots(maxpoints=none only). Omitting the
  MAXPOINTS= option does not affect the results
  of the practices in this course.
*/

%global ex_pi1;
%let ex_pi1=0.05;

title1 "Logistic Regression Model of the Veterans' Organization Data";
```

```
proc logistic data=pmlr.pva_train plots(only)=
          (effect(clband x=(pep_star recent_avg_gift_amt
          frequency_status_97nk)) oddsratio (type=horizontalstat));
  class pep_star (param=ref ref='0');
  model target_b(event='1')=pep_star recent_avg_gift_amt
             frequency_status_97nk / clodds=pl;
  effectplot slicefit(sliceby=pep_star x=recent_avg_gift_amt) / noobs;
  effectplot slicefit(sliceby=pep_star x=frequency_status_97nk) / noobs;
  score data=pmlr.pva_train out=work.scopva_train priorevent=&ex_pi1;
run;

title1 "Adjusted Predicted Probabilities of the Veteran's Organization Data";
proc print data=work.scopva_train(obs=10);
  var p_1 pep_star recent_avg_gift_amt frequency_status_97nk;

run;
title;



/* ================================================== */
/* Lesson 3, Practice 1
  Practice: Imputing Missing Values                */
/* ================================================== */
```

```
data pmlr.pva_train_mi(drop=i);
  set pmlr.pva_train;
  /* name the missing indicator variables */
  array mi{*} mi_DONOR_AGE mi_INCOME_GROUP
          mi_WEALTH_RATING;
  /* select variables with missing values */
  array x{*} DONOR_AGE INCOME_GROUP WEALTH_RATING;
  do i=1 to dim(mi);
    mi{i}=(x{i}=.);
    nummiss+mi{i};
  end;
run;

proc rank data=pmlr.pva_train_mi out=work.pva_train_rank
       groups=3;
  var recent_response_prop recent_avg_gift_amt;
  ranks grp_resp grp_amt;
run;

proc sort data=work.pva_train_rank out=work.pva_train_rank_sort;
  by grp_resp grp_amt;
run;

proc stdize data=work.pva_train_rank_sort method=median
```

```
        reponly out=pmlr.pva_train_imputed;
  by grp_resp grp_amt;
  var DONOR_AGE INCOME_GROUP WEALTH_RATING;
run;


options nolabel;
proc means data=pmlr.pva_train_imputed median;
  class grp_resp grp_amt;
  var DONOR_AGE INCOME_GROUP WEALTH_RATING;
run;
options label;



/* ================================================== */
/* Lesson 3, Practice 2
  Practice: Collapsing the Levels of a Nominal Input

  Note: After you submit this code, a note in the log
  indicates that argument 3 to the LOGSDF function
  is invalid. You can ignore this note; it is not
  important for this analysis. The note pertains
  to the situation in which the number of clusters is 1.
  In this case, the degrees of freedom is 0 (degrees of
  freedom is equal to the number of clusters minus 1) and
```

```
  the mathematical operation cannot be performed in the
  LOGSDF function. Therefore, the log of the p-value is
  set to missing.                          */
/* ================================================== */

proc means data=pmlr.pva_train_imputed noprint nway;
  class cluster_code;
  var target_b;
  output out=work.level mean=prop;
run;

ods output clusterhistory=work.cluster;

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

proc freq data=pmlr.pva_train_imputed noprint;
  tables cluster_code*target_b / chisq;
  output out=work.chi(keep=_pchi_) chisq;
```

```
run;


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


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=-40 max=0;
run;


title1;


%global ncl;


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 cluster_code;
run;

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

title1 "Cluster Assignments";
proc print data=work.clus;
  by clusname;
  id clusname;
run;

filename clcode "&PMLRfolder/cluster_code.sas";

data _null_;
  file clcode;
  set work.clus end=last;
```

```
  if _n_=1 then put "select (cluster_code);";
  put "  when ('" cluster_code +(-1) "')
       cluster_clus='" cluster +(-1) "';";
  if last then do;
    put "  otherwise cluster_clus='U';" / "end;";
  end;
run;

data pmlr.pva_train_imputed_clus;
  set pmlr.pva_train_imputed;
  %include clcode;
run;

/* ========================================================== */
/* Lesson 3, Practice 3
  Practice: Computing the Smoothed Weight of Evidence   */
/* ========================================================== */

%global rho1_ex;
proc sql noprint;
  select mean(target_b) into :rho1_ex
  from pmlr.pva_train_imputed;
run;
```

```
proc means data=pmlr.pva_train_imputed
        sum nway noprint;
  class cluster_code;
  var target_b;
  output out=work.counts sum=events;
run;

filename clswoe "&PMLRfolder/swoe_cluster.sas";

data _null_;
  file clswoe;
  set work.counts end=last;
    logit=log((events + &rho1_ex*24)/
            (_FREQ_ - events + (1-&rho1_ex)*24));
  if _n_=1 then put "select (cluster_code);" ;
  put "  when ('" cluster_code +(-1) "') cluster_swoe=" logit ";" ;
  if last then do;
    logit=log(&rho1_ex/(1-&rho1_ex));
    put "  otherwise cluster_swoe=" logit ";" / "end;";
  end;
run;

data pmlr.pva_train_imputed_swoe;
```

```
  set pmlr.pva_train_imputed;
  %include clswoe;
run;


title;


proc print data=pmlr.pva_train_imputed_swoe(obs=1);
  where cluster_code = "01";
  var cluster_code cluster_swoe;
run;




/* ================================================== */
/* Lesson 3, Practice 4
  Practice: Reducing Redundancy by Clustering Variables */
/* ================================================== */


/*Note: If you run this code in 32-bit SAS, the variable
  assignments to clusters might vary from what is shown
  in the results in this course. This discrepancy does
  not affect the results of the remaining practices in
  this course.
*/
```

```
%let ex_inputs= MONTHS_SINCE_ORIGIN
DONOR_AGE IN_HOUSE INCOME_GROUP PUBLISHED_PHONE
MOR_HIT_RATE WEALTH_RATING MEDIAN_HOME_VALUE
MEDIAN_HOUSEHOLD_INCOME PCT_OWNER_OCCUPIED
PER_CAPITA_INCOME PCT_MALE_MILITARY
PCT_MALE_VETERANS PCT_VIETNAM_VETERANS
PCT_WWII_VETERANS PEP_STAR RECENT_STAR_STATUS
FREQUENCY_STATUS_97NK RECENT_RESPONSE_PROP
RECENT_AVG_GIFT_AMT RECENT_CARD_RESPONSE_PROP
RECENT_AVG_CARD_GIFT_AMT RECENT_RESPONSE_COUNT
RECENT_CARD_RESPONSE_COUNT LIFETIME_CARD_PROM
LIFETIME_PROM LIFETIME_GIFT_AMOUNT
LIFETIME_GIFT_COUNT LIFETIME_AVG_GIFT_AMT
LIFETIME_GIFT_RANGE LIFETIME_MAX_GIFT_AMT
LIFETIME_MIN_GIFT_AMT LAST_GIFT_AMT
CARD_PROM_12 NUMBER_PROM_12 MONTHS_SINCE_LAST_GIFT
MONTHS_SINCE_FIRST_GIFT STATUS_FL STATUS_ES
home01 nses1 nses3 nses4 nses_ nurbr nurbu nurbs
nurbt nurb_;


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

```
proc varclus data=pmlr.pva_train_imputed_swoe
          hi maxeigen=0.70;
  var &ex_inputs mi_DONOR_AGE mi_INCOME_GROUP
     mi_WEALTH_RATING cluster_swoe;
run;

ods select all;

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;
  label RSquareRatio="1 - RSquare*Ratio";
run;

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

```
/* ================================================== */
/* Lesson 3, Practice 5
   Practice: Performing Variable Screening            */
/* ================================================== */

%let ex_reduced=
LIFETIME_GIFT_COUNT LAST_GIFT_AMT MEDIAN_HOME_VALUE
FREQUENCY_STATUS_97NK MONTHS_SINCE_LAST_GIFT  nses_
mi_DONOR_AGE PCT_MALE_VETERANS PCT_MALE_MILITARY
PCT_WWII_VETERANS LIFETIME_AVG_GIFT_AMT cluster_swoe
PEP_STAR nurbu nurbt home01 nurbr DONOR_AGE STATUS_FL
MOR_HIT_RATE nses4 INCOME_GROUP RECENT_STAR_STATUS IN_HOUSE
WEALTH_RATING PUBLISHED_PHONE PCT_OWNER_OCCUPIED nurbs;

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

proc corr data=pmlr.pva_train_imputed_swoe
        spearman hoeffding;
  var target_b;
  with &ex_reduced;
```

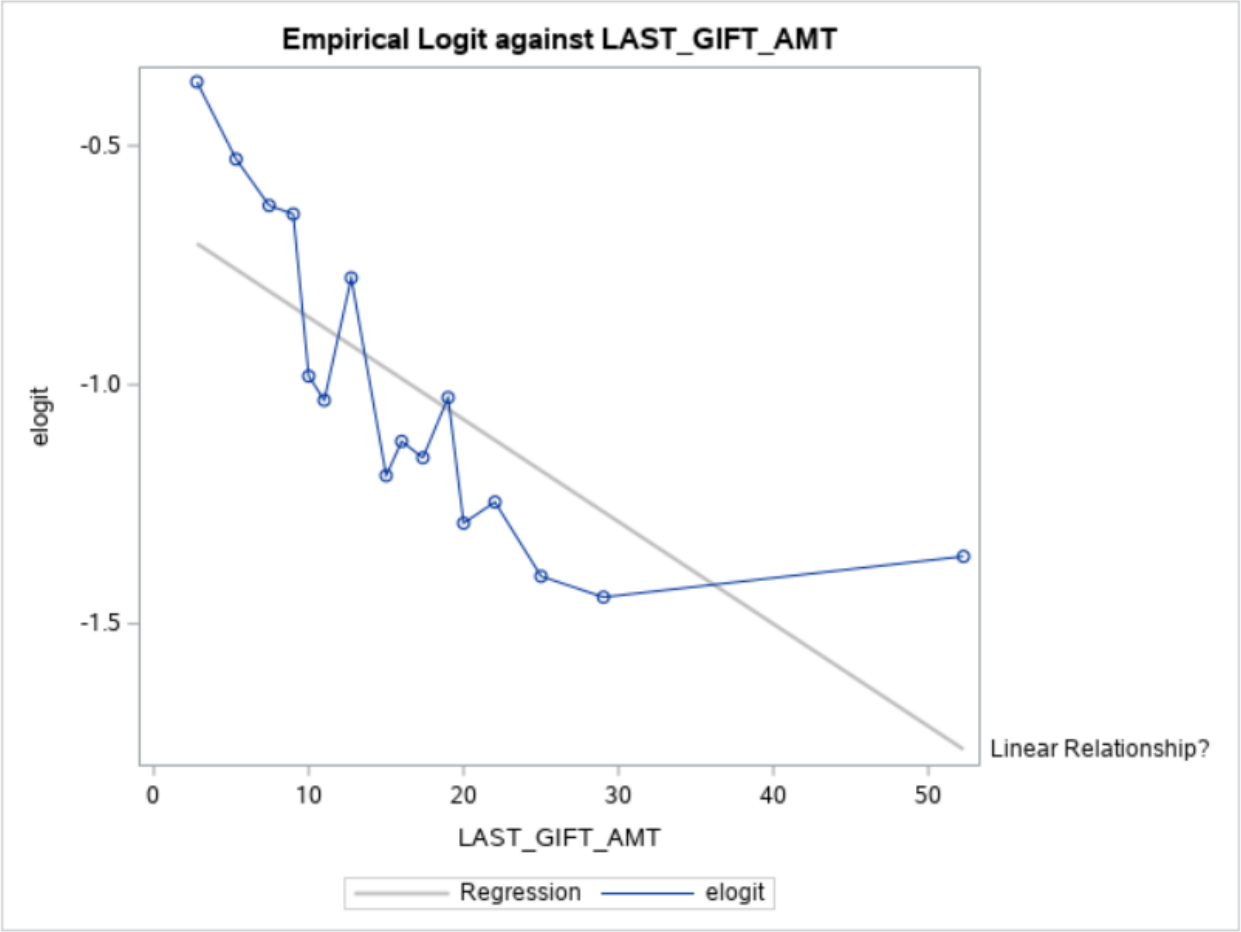
```
run;


ods select all;


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


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


data work.correlations;
  attrib variable length=$32;
  merge work.spearman(rename=
      (target_b=scorr ptarget_b=spvalue))
      work.hoeffding
      (rename=(target_b=hcorr ptarget_b=hpvalue));
  by variable;
  scorr_abs=abs(scorr);
  hcorr_abs=abs(hcorr);
run;


proc rank data=work.correlations
```

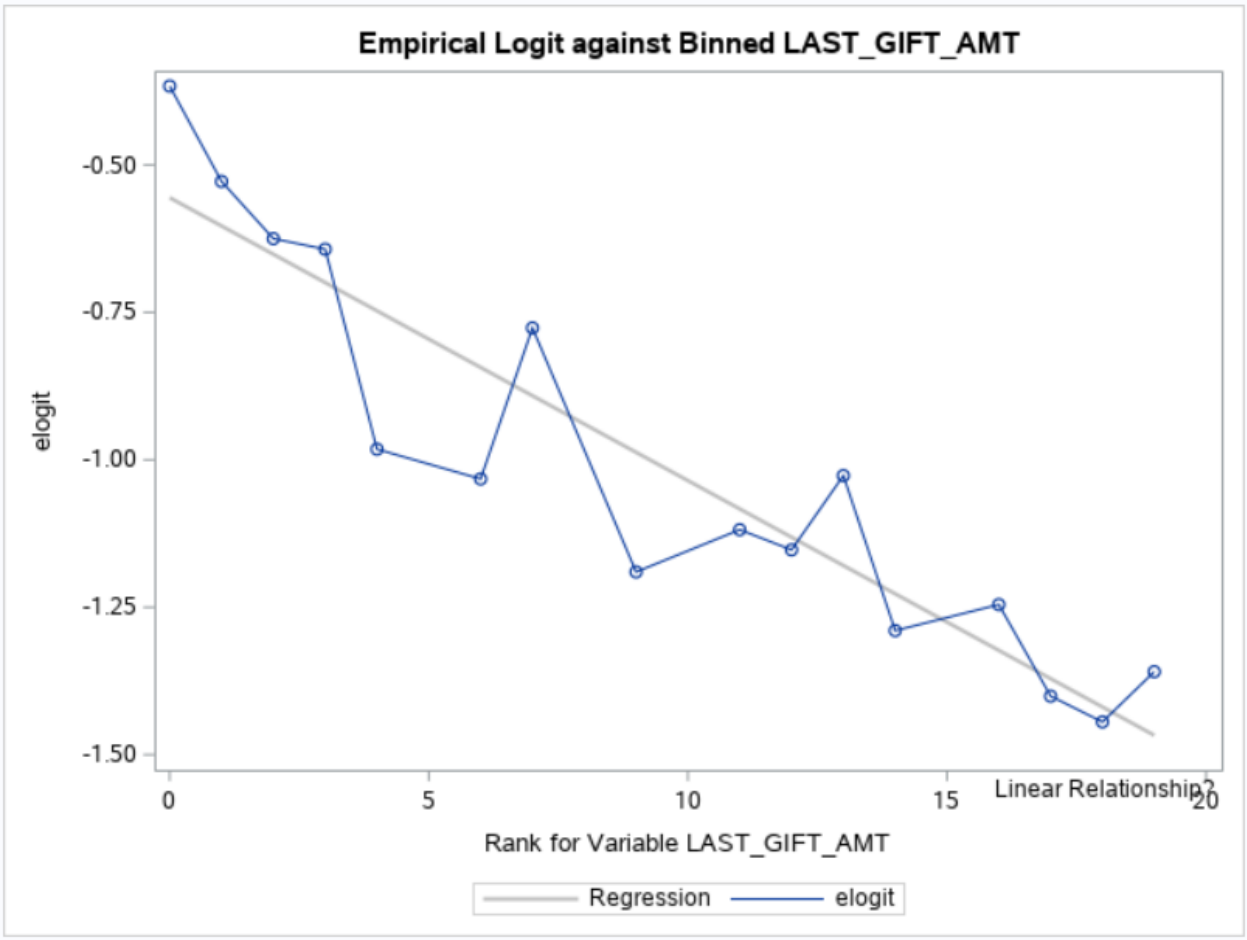
```
        out=work.correlations1 descending;
   var scorr_abs hcorr_abs;
   ranks ranksp rankho;
run;

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;

%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;

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;

/* Solution for l3p6 */

/* step 2 */

%global var;
%let var=LAST_GIFT_AMT;
```

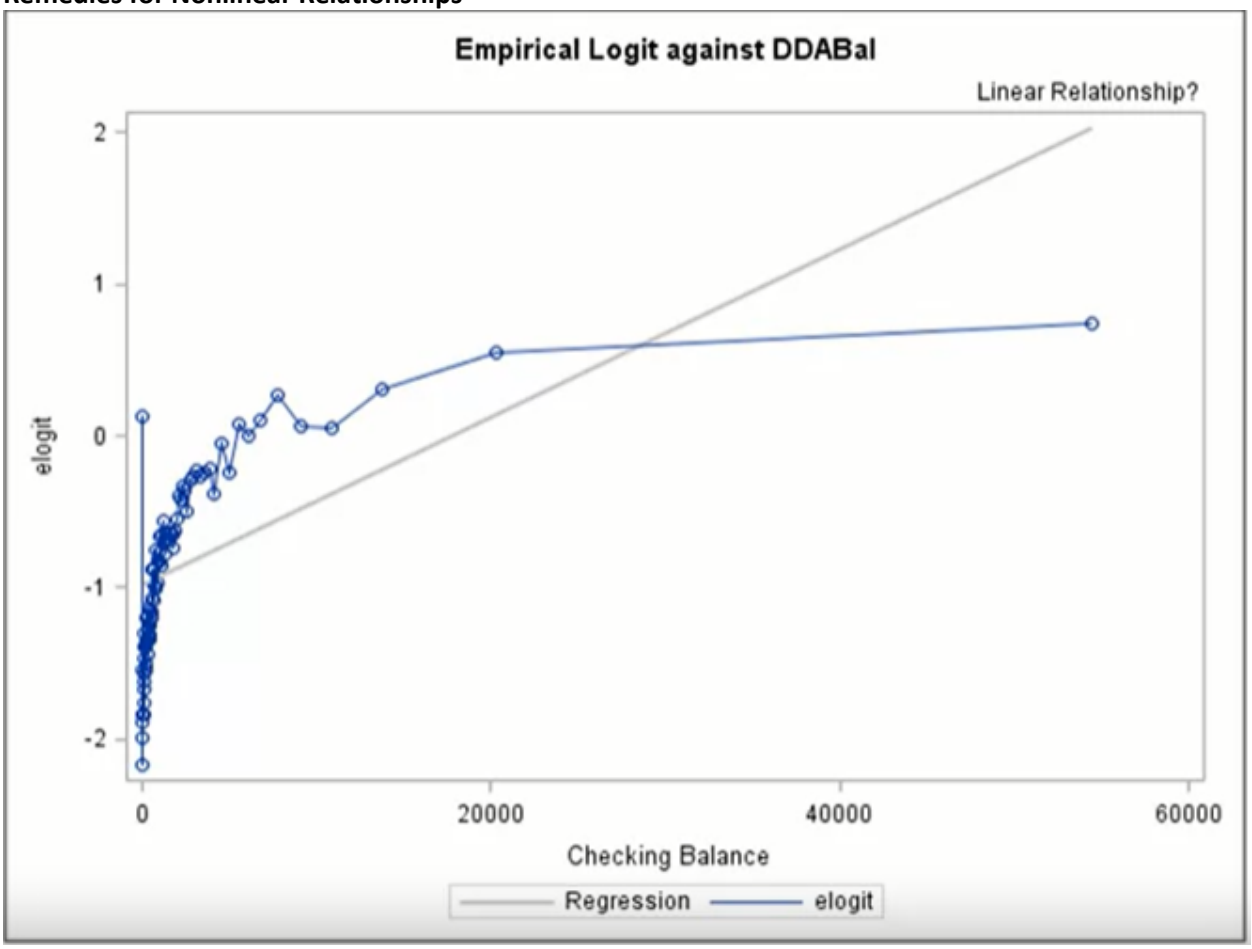
```
proc rank data=pmlr.pva_train_imputed_swoe
        groups=20 out=work.ranks;
  var &var;
  ranks bin;
run;



/* step 3 */


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


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



/* step 4 */
```

```
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;



/* step 5 */


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;
title1;
```


Empirical Logit against LAST_GIFT_AMT
-0.5
-1.0
-1.5
elogit
0
10
20
30
40
50
LAST_GIFT_AMT
Linear Relationship?
Regression
elogit

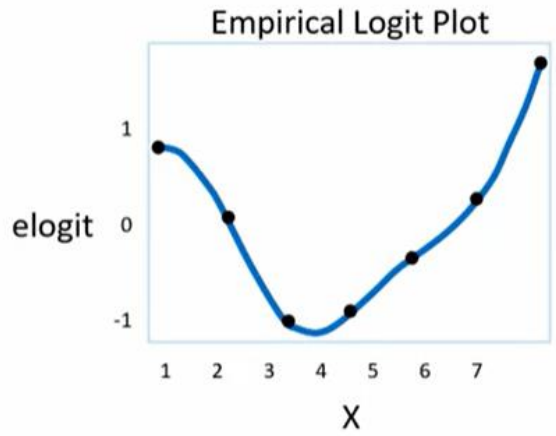

Empirical Logit against Binned LAST_GIFT_AMT
-0.50
-0.75
-1.00
-1.25
-1.50
elogit
0
5
10
15
20
Linear Relationship?
Rank for Variable LAST_GIFT_AMT
Regression
elogit

Remedies for Nonlinear Relationships



Low
Medium
High

Remedies
Create hand-crafted new input variables



add quadratic
and cubic terms

Remedies
Create hand-crafted new input variables
use a polynomial model

- classification trees
- generalized additive models
- neural networks

Remedies
Create hand-crafted new input variables
use a polynomial model
use a flexible multivariate function estimator

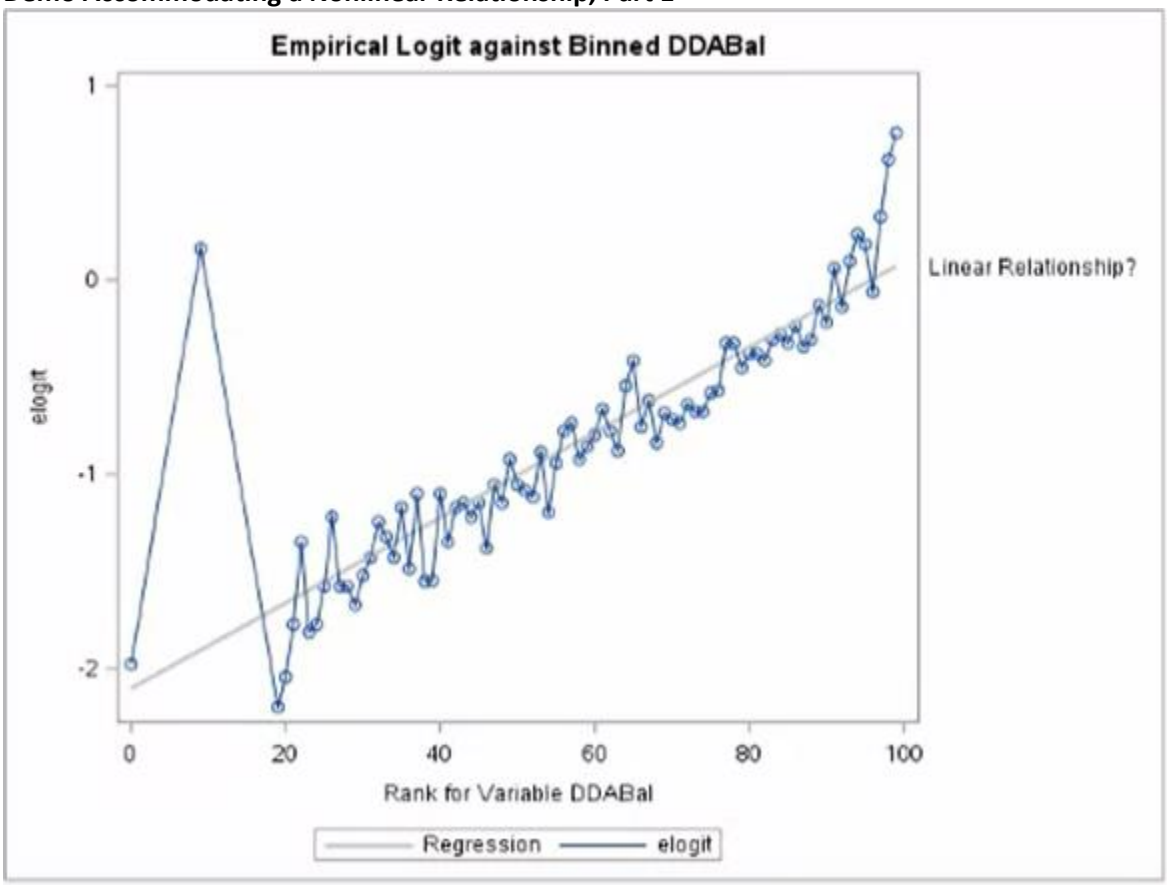
Question 3.07

In which of the following ways can you increase the smoothing of an empirical logit plot?

use a small number of bins with many observations per bin

The number of bins determines the amount of smoothing. Fewer bins mean more smoothing.

Demo Accommodating a Nonlinear Relationship, Part 1



* For customers who have checking accounts, create a macro variable that has the mean value of the checking account balance.

* For customers with no checking accounts, replace the checking account balance with the mean in the macro variable.

* Generate empirical logit plots to verify that the relationship is now linear.

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;
```

Checking Account Balance and INS by Checking Account

The MEANS Procedure

Checking Account	N Obs	Variable	Label	Mean	Median	Minimum	Maximum
0	3968	DDABal Ins	Checking Balance	0 0.5415827	0 1.0000000	0 0	0 1.0000000
1	17544	DDABal Ins	Checking Balance	2713.45 0.3022116	890.5900000 0	-774.8300000 0	278093.83 1.0000000

pmlr03d07.sas

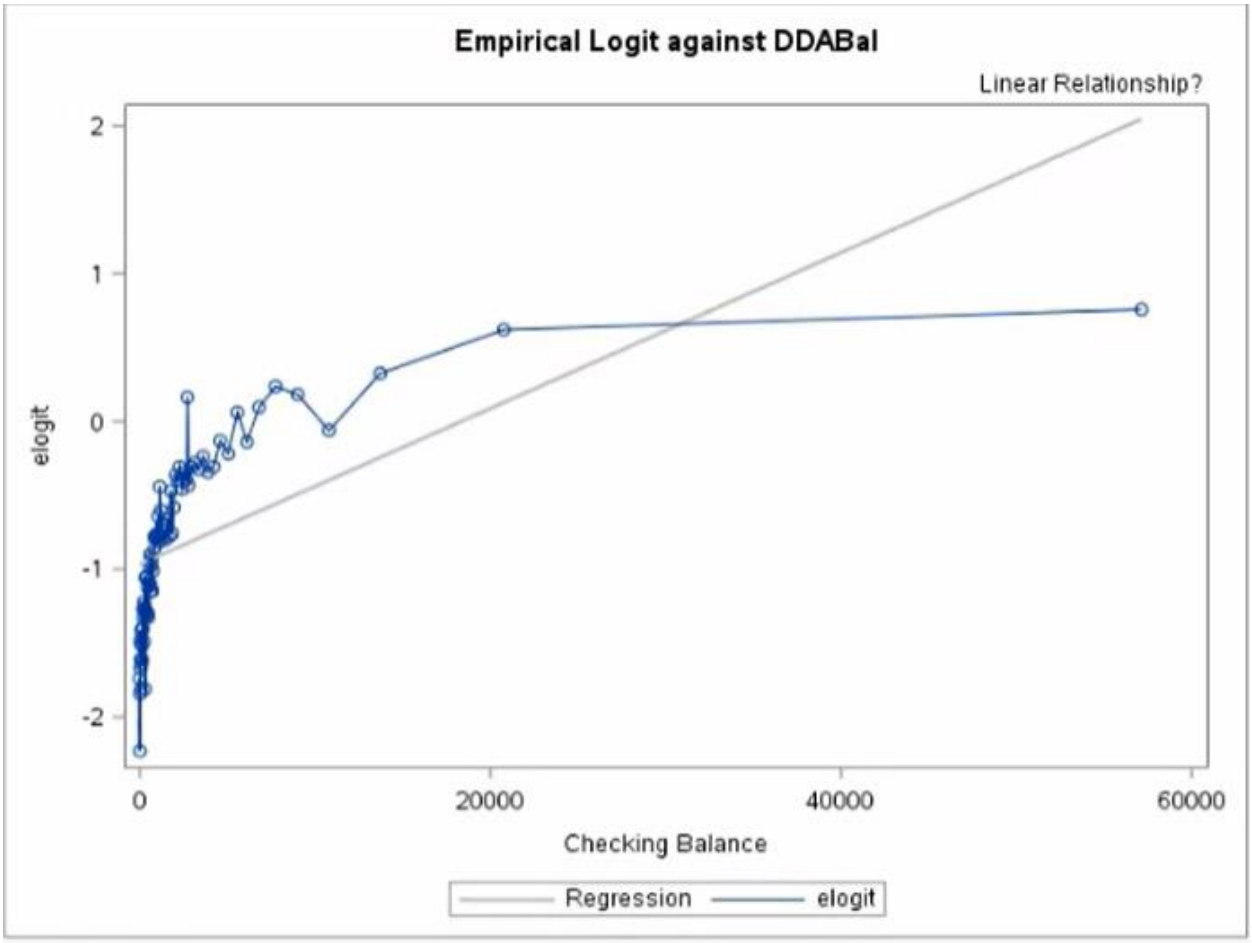
```
%global mean;
proc sql noprint;
   select mean(ddabal) into :mean
   from work.train_imputed_swoe where dda;
quit;

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

```
%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;
```

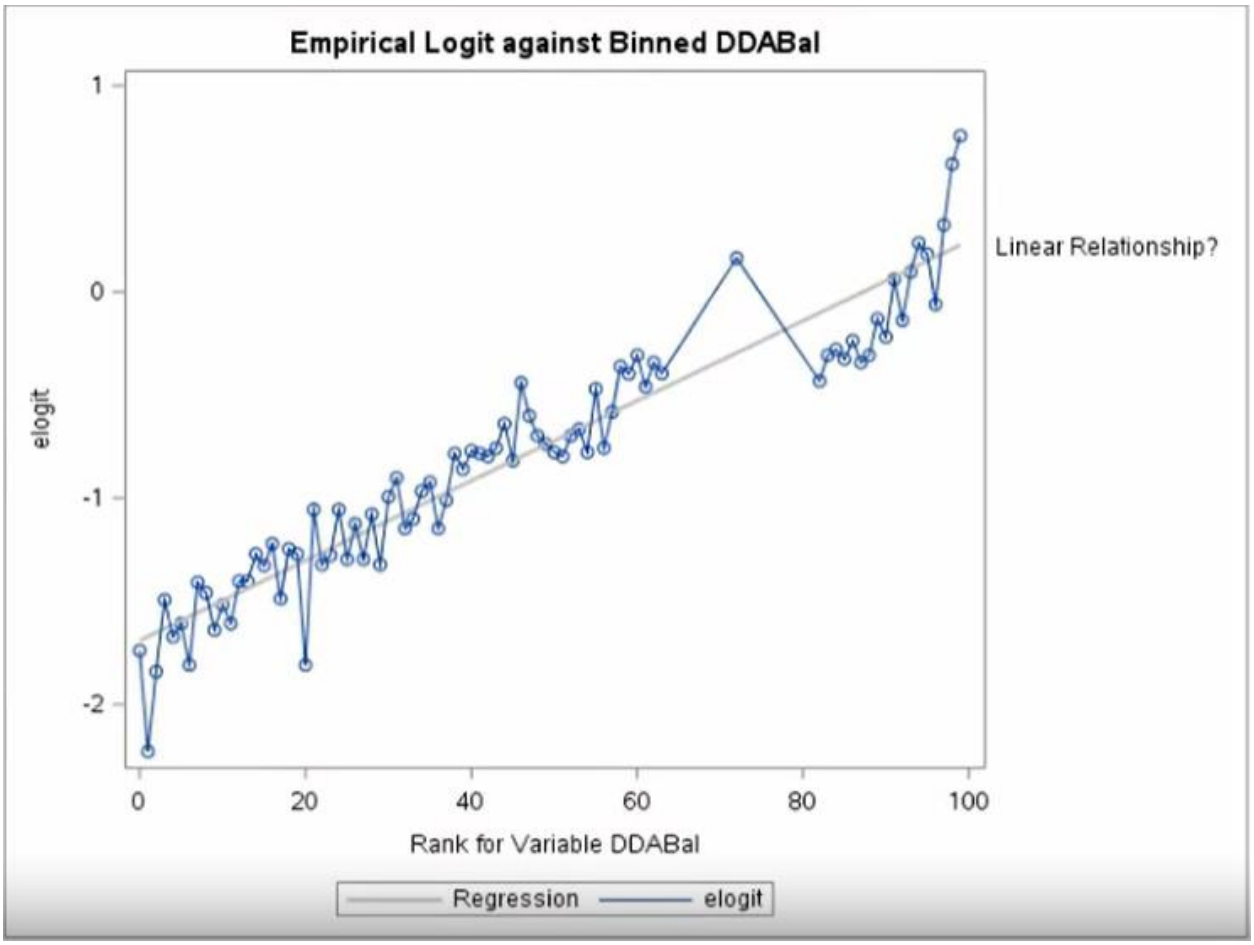
```
/* Calculate the empirical logit */
data work.bins;
   set work.bins;
   elogit=log((ins+(sqrt(_FREQ_ )/2))/
           ( _FREQ_ -ins+(sqrt(_FREQ_ )/2)));
run;
```

```
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;
```




```
/* Run this code before demo l3d7a */

/* ================================================== */
/* Lesson 1, Section 1: l1d1.sas
   Demonstration: Examining the Code for Generating
   Descriptive Statistics and Frequency Tables          */
/* ================================================== */

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


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

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

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

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

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

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

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

/* 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 */
```

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

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

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

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

```
%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;
```

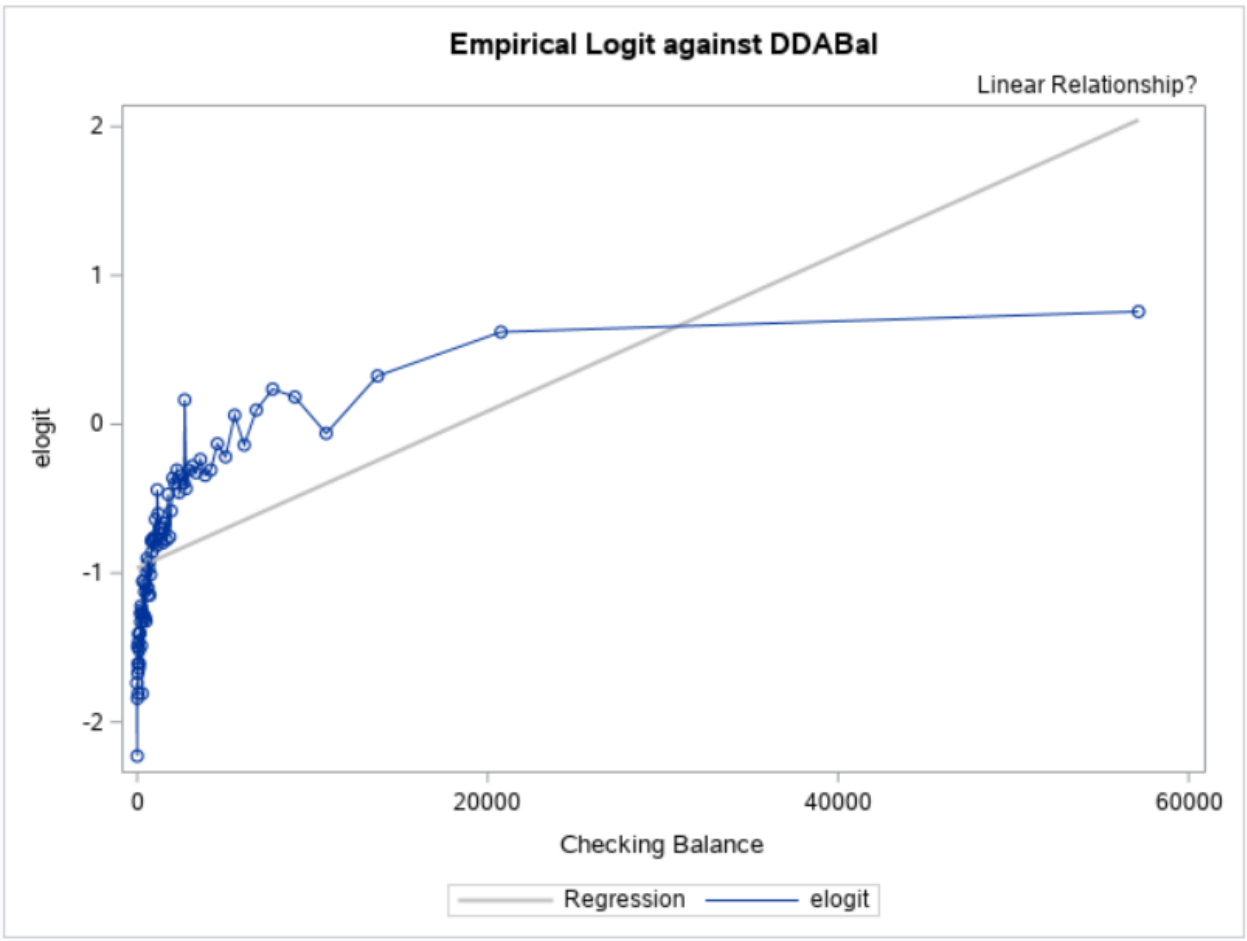
```
%let var=DDABal;

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;
```

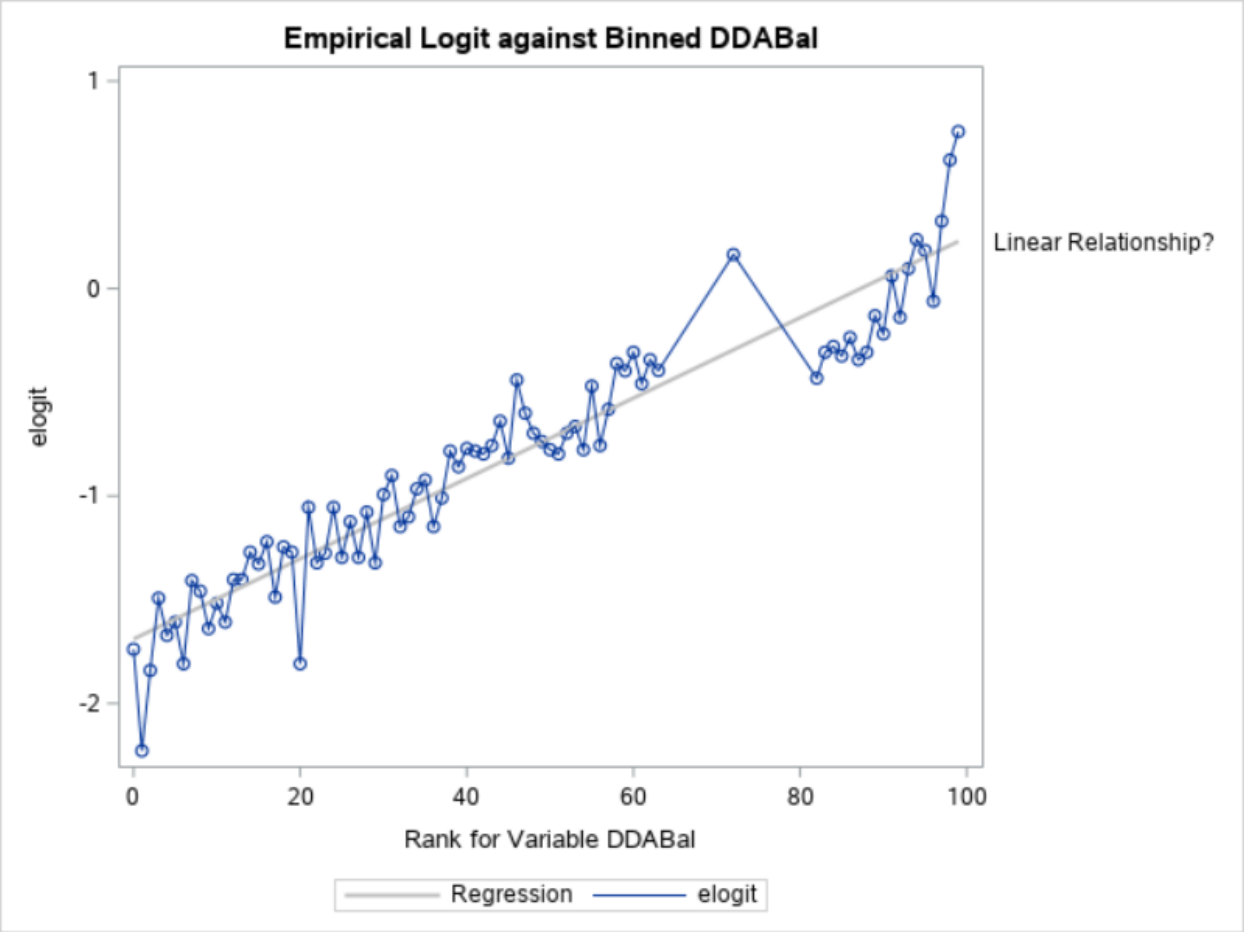
```
title1 "Empirical Logit against Binned &var";
proc sgplot data=work.bins;
  reg y=elogit x=bin /
     curvelabel="Linear Relationship?"
     curvelabelloc=outside
     lineattrs=(color=ligr);
  series y=elogit x=bin;
run;
```

Checking Account Balance and INS by Checking Account

The MEANS Procedure

Checking Account	N Obs	Variable	Label	Mean	Median	Minimum	Maximum
0	3968	DDABal	Checking Balance	0	0	0	0
		Ins		0.5415827	1.0000000	0	1.0000000
1	17544	DDABal	Checking Balance	2713.45	890.5900000	-774.8300000	278093.83
		Ins		0.3022116	0	0	1.0000000


Empirical Logit against DDABal
Linear Relationship?
2
1
0
-1
-2
elogit
0
20000
40000
60000
Checking Balance
Regression
elogit



Demo Accommodating a Nonlinear Relationship, Part 2

* Create a set of rules that bins all the values of checking account balance, and store the score code in a file.

* Create a binned checking account variable in the training data set by using the score code in a DATA step.

* Replace the old input variable (**DDABal**) with the binned input variable (**B_DDABal**) in the **screened** macro variable.

pmlr03d07.sas

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

```
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;
```

Checking Account Balance Endpoints

Obs	bin	_TYPE_	_FREQ_	max
1	0	1	215	1.75
2	1	1	215	8.45
3	2	1	214	16.44
4	3	1	216	27.54
5	4	1	215	41.08
6	5	1	215	53.01
7	6	1	215	66.42
8	7	1	216	82.96
9	8	1	215	96.55
10	9	1	215	111.59

```
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;
```

If **DDABal** <= 1.75 then bin = 0.

rank.sas

```
select;
  when (ddabal <= 1.75 ) B_DDABal =0 ;
  when (ddabal <= 8.45 ) B_DDABal =1 ;
  when (ddabal <= 16.44 ) B_DDABal =2 ;
  when (ddabal <= 27.54 ) B_DDABal =3 ;
  when (ddabal <= 41.08 ) B_DDABal =4 ;
  when (ddabal <= 53.01 ) B_DDABal =5 ;
  when (ddabal <= 66.42 ) B_DDABal =6 ;
  when (ddabal <= 82.96 ) B_DDABal =7 ;
  when (ddabal <= 96.55 ) B_DDABal =8 ;
  when (ddabal <= 111.59 ) B_DDABal =9 ;
  when (ddabal <= 126.69 ) B_DDABal =10 ;
  when (ddabal <= 141.86 ) B_DDABal =11 ;
```

```
  when (ddabal <= 4779.07 ) B_DDABal =89 ;
  when (ddabal <= 5320.9 ) B_DDABal =90 ;
  when (ddabal <= 5817.37 ) B_DDABal =91 ;
  when (ddabal <= 6416.94 ) B_DDABal =92 ;
  when (ddabal <= 7221.55 ) B_DDABal =93 ;
  when (ddabal <= 8296.78 ) B_DDABal =94 ;
  when (ddabal <= 9821.81 ) B_DDABal =95 ;
  when (ddabal <= 11955.74 ) B_DDABal =96 ;
  when (ddabal <= 15939.54 ) B_DDABal =97 ;
  when (ddabal <= 27215.32 ) B_DDABal =98 ;
  otherwise B_DDABal =99 ;
end;
```

```
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 ;
```

Minimum and Maximum Checking Account Balance by Bin

The MEANS Procedure

Analysis Variable : DDABal Checking Balance			
B_DDABal	N Obs	Minimum	Maximum
0	215	-774.8300000	1.7500000
1	215	1.8900000	8.4500000
2	214	8.4600000	16.4400000
3	216	16.4600000	27.5400000
4	215	27.6200000	41.0800000
5	215	41.1900000	53.0100000
6	215	53.0200000	66.4200000
7	216	66.5000000	82.9600000
8	215	83.0600000	96.5500000
9	215	96.6200000	111.5900000

57	215	1903.00	1986.81
58	215	1987.12	2102.44
59	215	2102.58	2202.04
60	215	2202.17	2334.00
61	216	2334.41	2477.67
62	215	2478.17	2618.68
63	132	2618.71	2711.31
72	3968	2713.45	2713.45
82	202	2713.62	2879.45
83	215	2879.53	3055.94
84	216	3057.85	3262.49
85	215	3264.28	3482.74
86	215	3482.81	3739.71
87	215	3741.92	4038.15
88	215	4039.48	4368.90

```
/* 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 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;
```