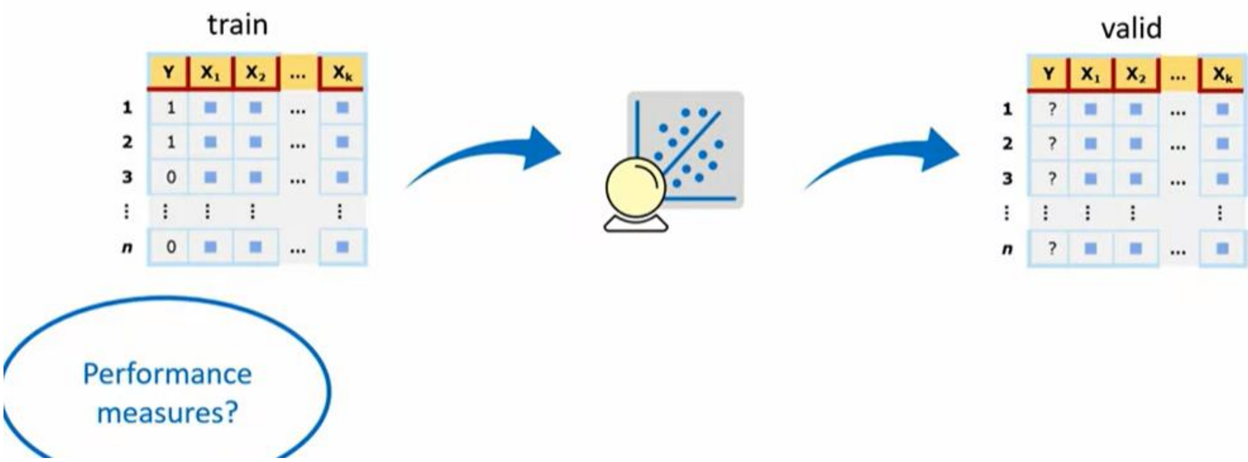
SBA Statistical Business Analyst using SAS

SBA3 Predictive Modeling with Logistic Regression

W5B Common Metrics for Model Performance

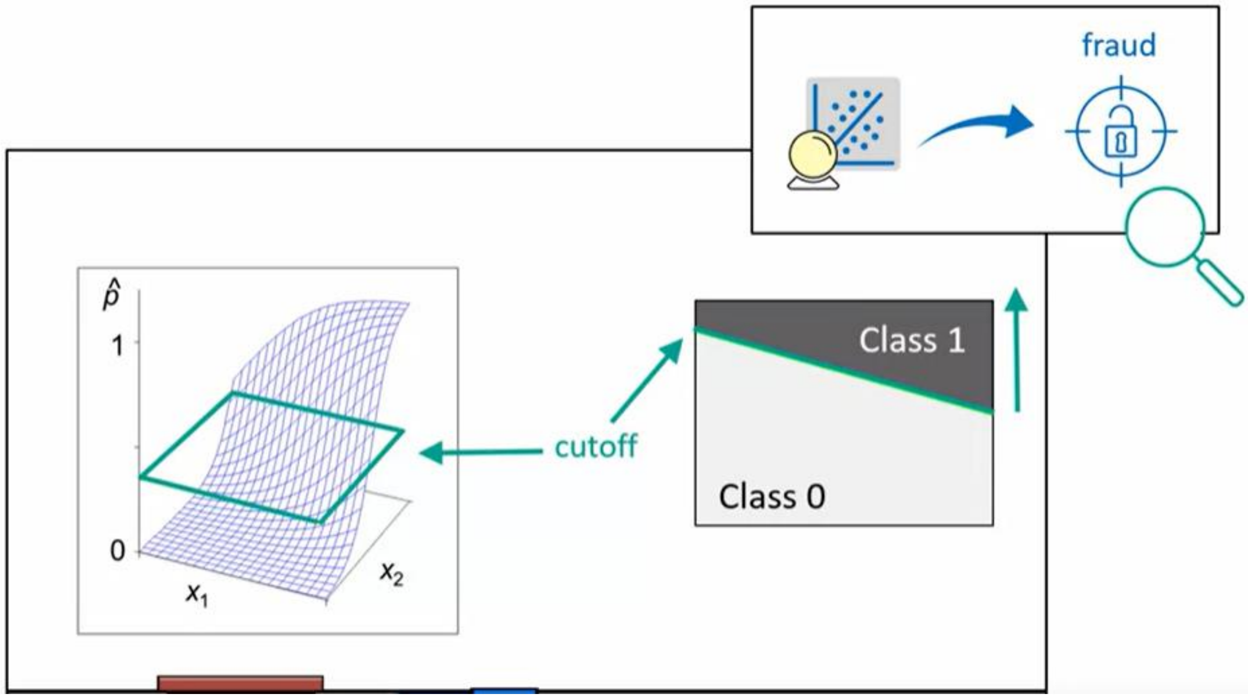
Introduction

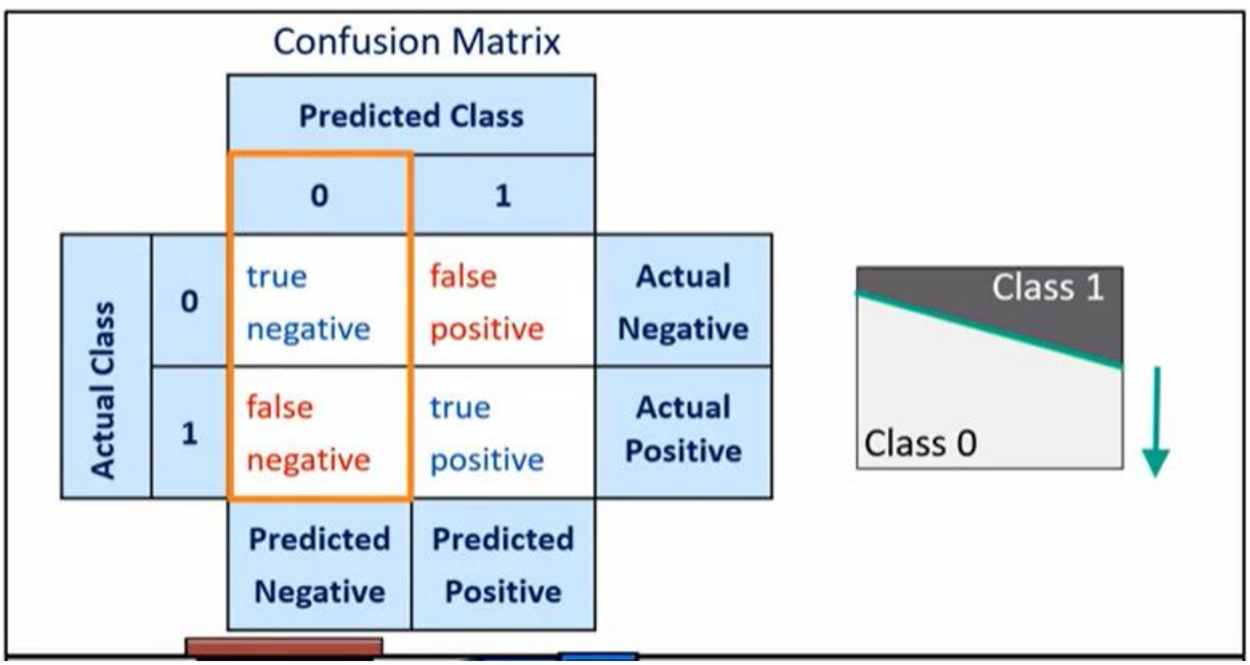


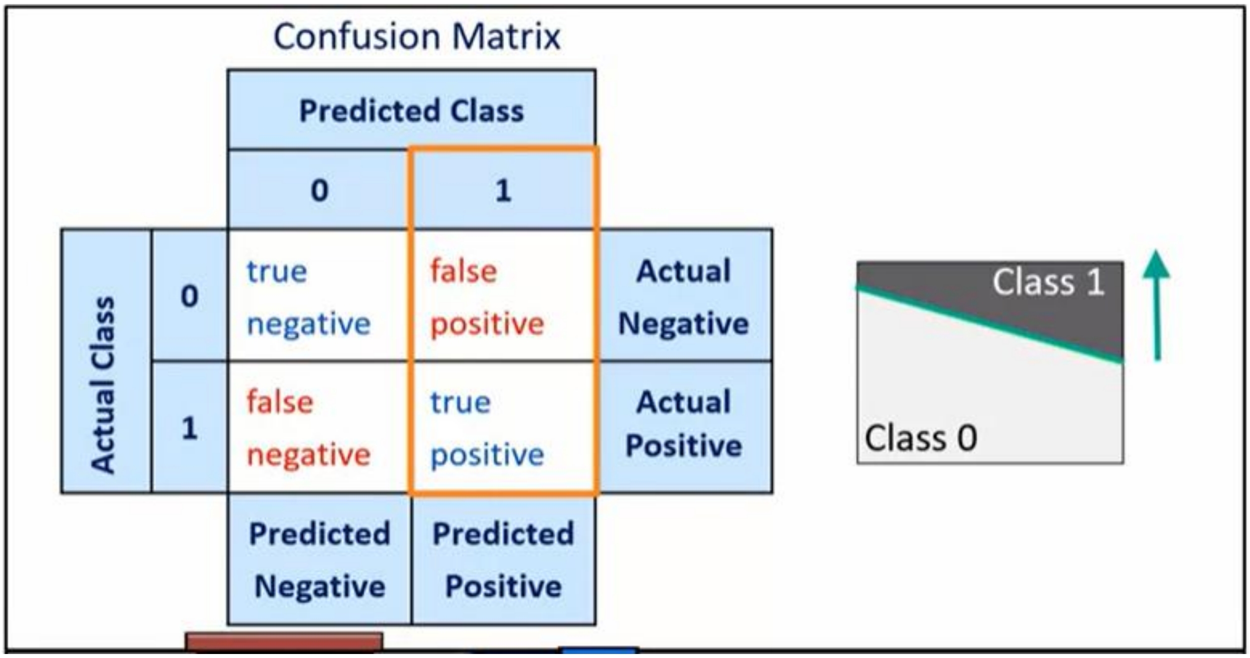
In this topic, you learn to do the following:

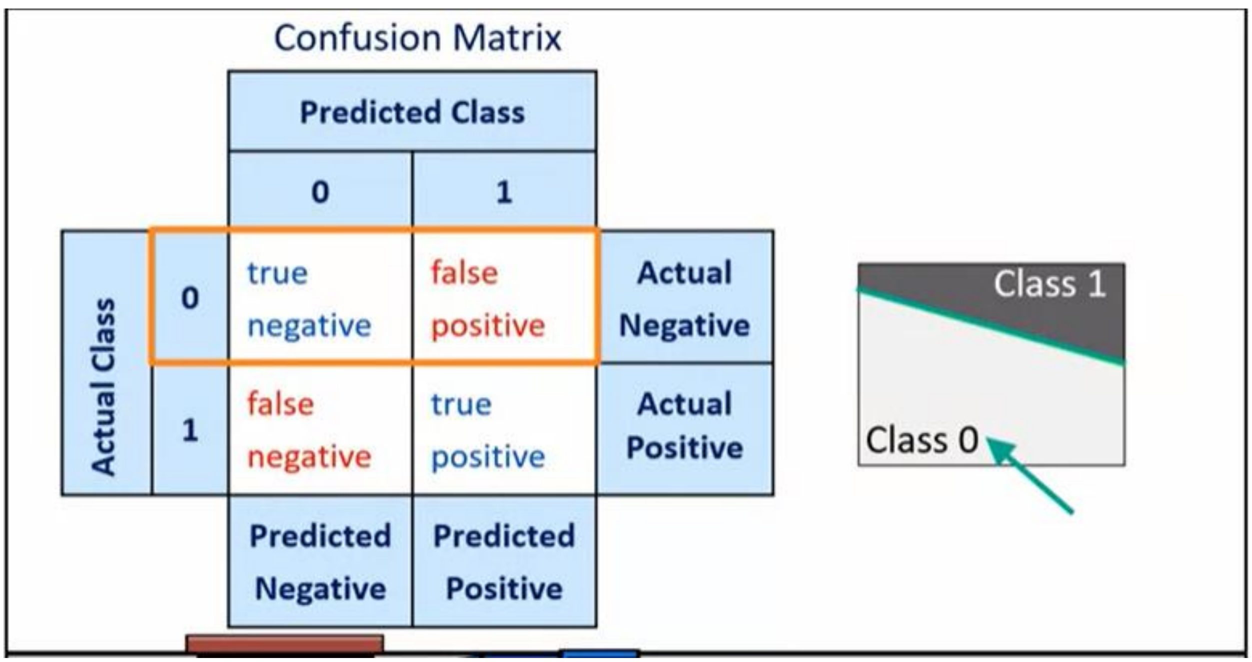
- describe several model performance measures
- adjust the confusion matrix for oversampling
- create ROC curves, gains charts, and lift charts on the validation data set

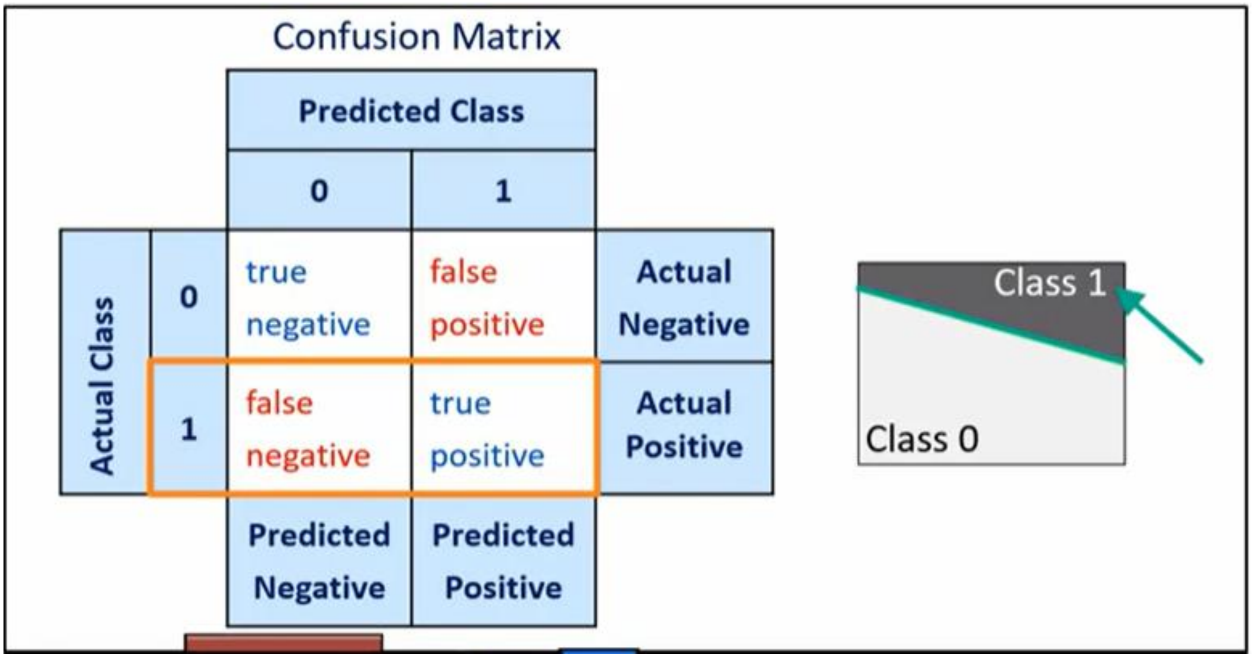
Understanding the Confusion Matrix

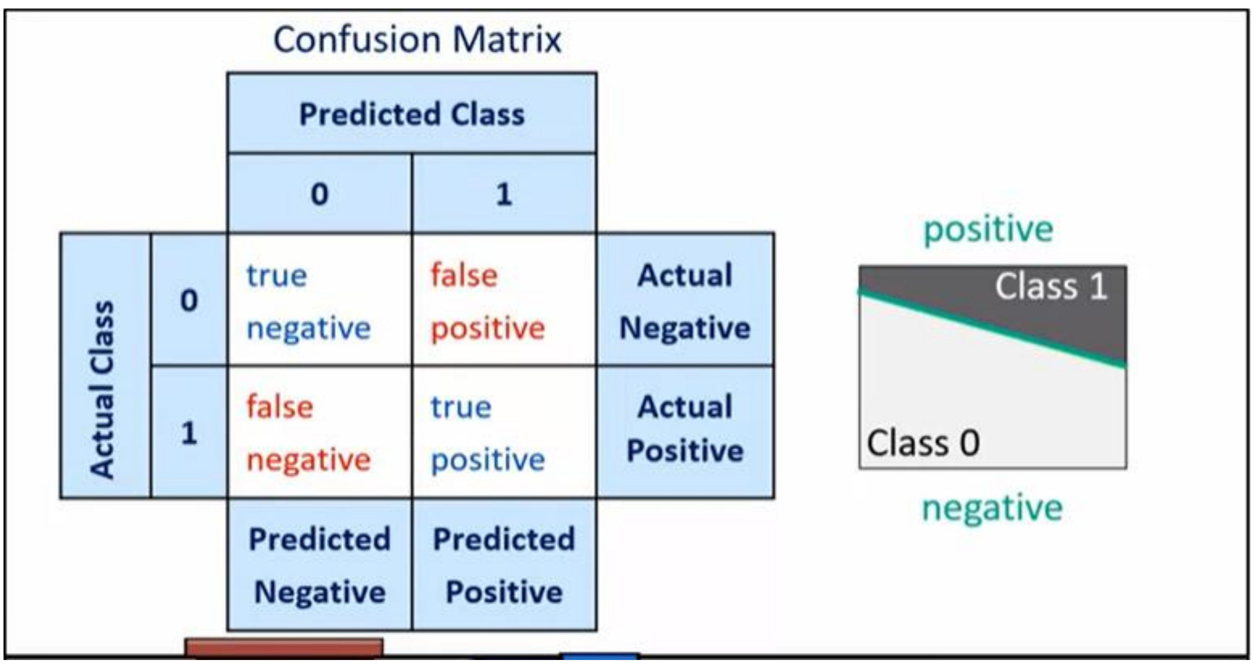


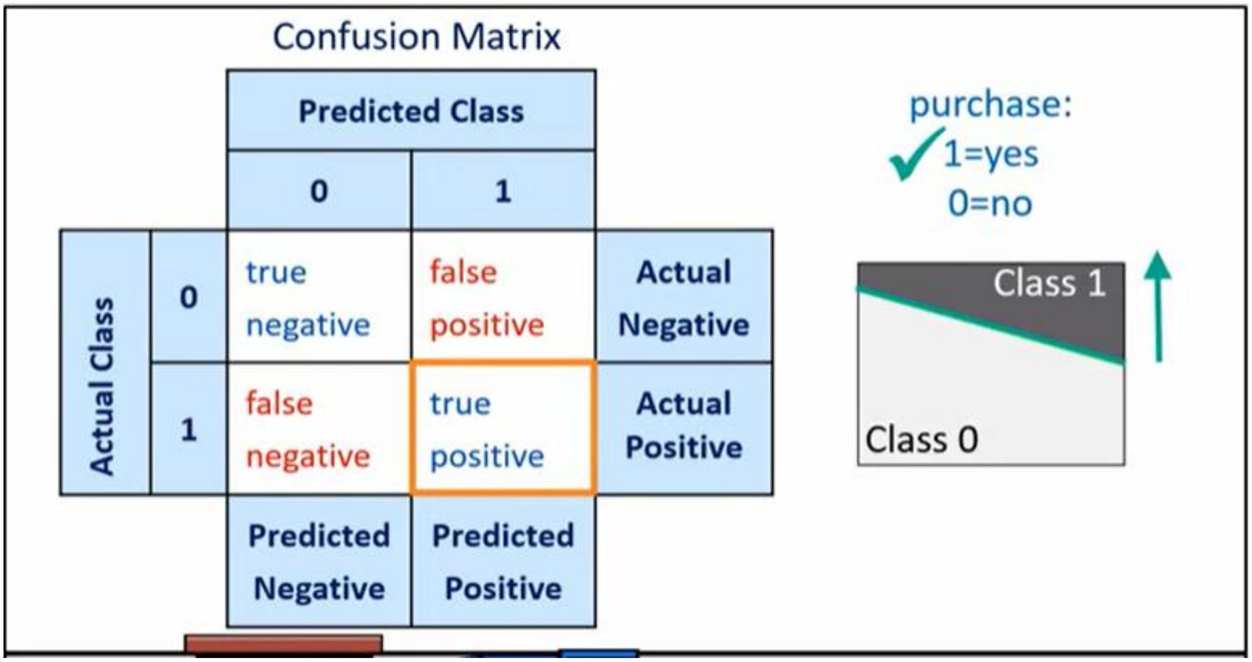


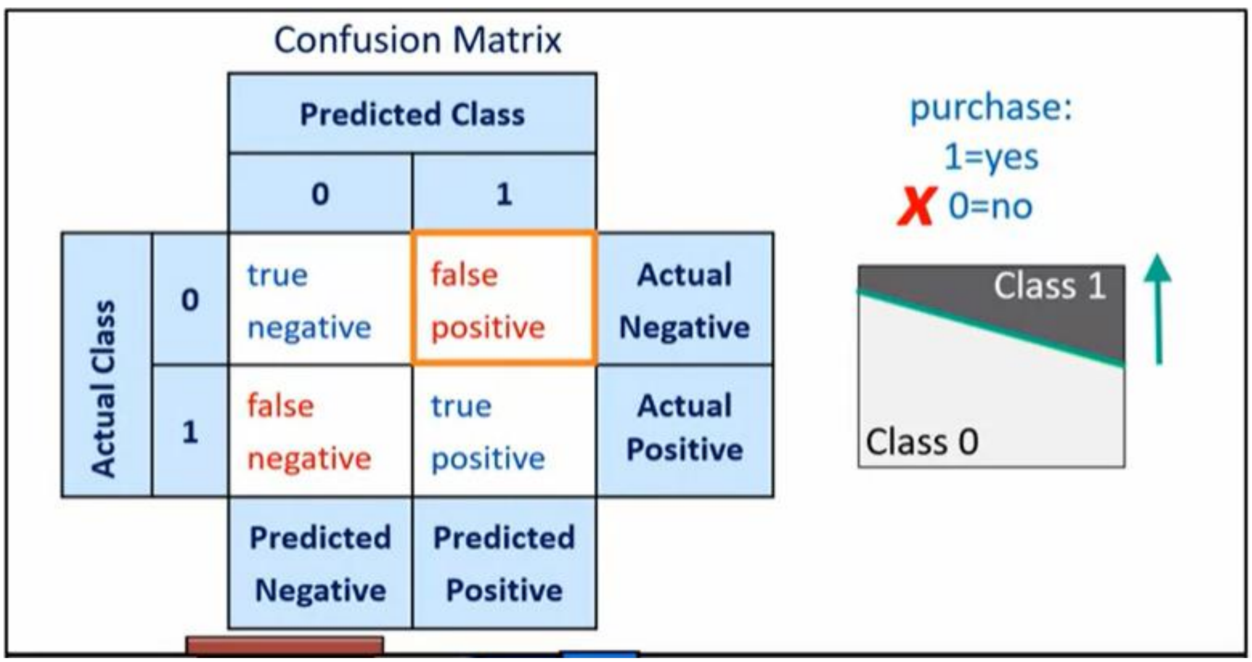

Confusion Matrix
Predicted Class
0
1
Actual Class
0
1
true negative
false positive
false negative
true positive
Actual Negative
Actual Positive
Predicted Negative
Predicted Positive
Class 1
Class 0

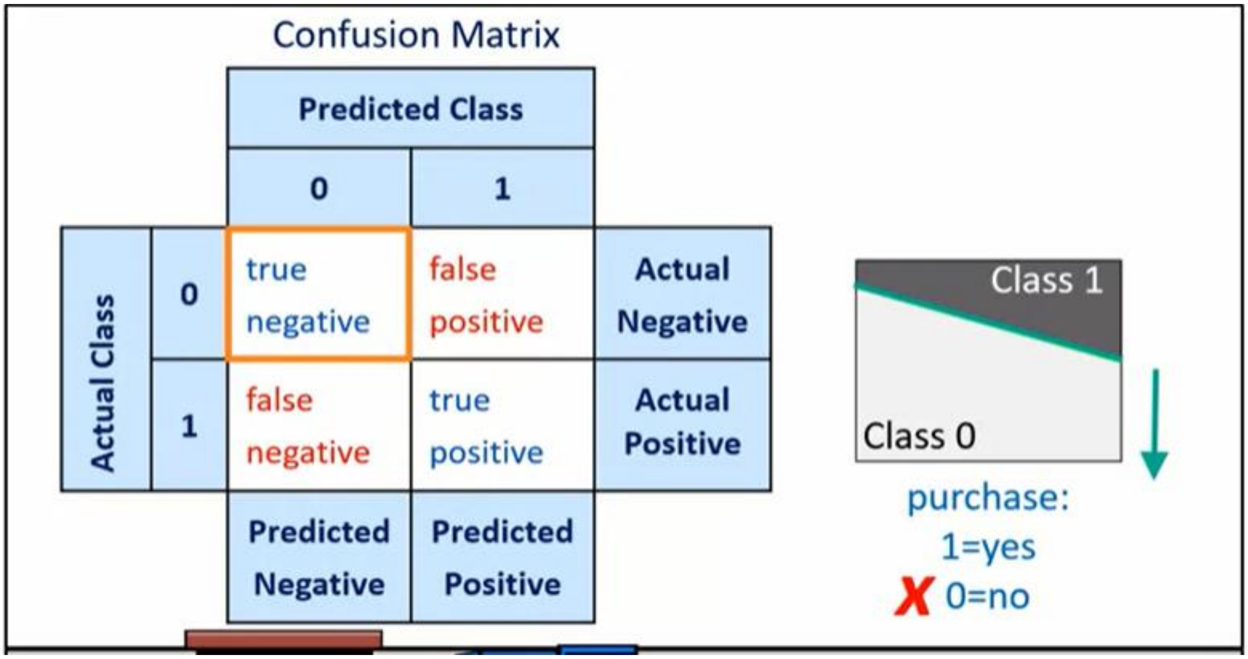

Confusion Matrix
Predicted Class
0
1
Actual Class
0
1
true negative
false positive
false negative
true positive
Actual Negative
Actual Positive
Predicted Negative
Predicted Positive
Class 1
Class 0

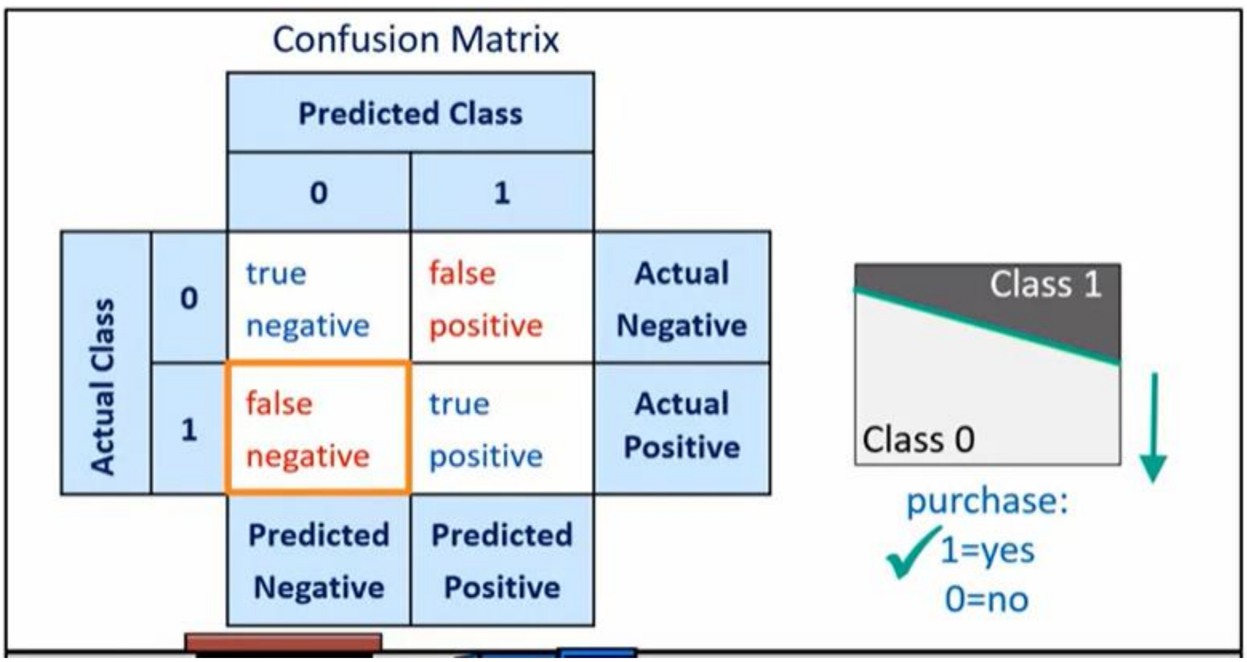

Confusion Matrix
Predicted Class
0
1
Actual Class
0
1
true negative
false positive
Actual Negative
false negative
true positive
Actual Positive
Predicted Negative
Predicted Positive
Class 1
Class 0

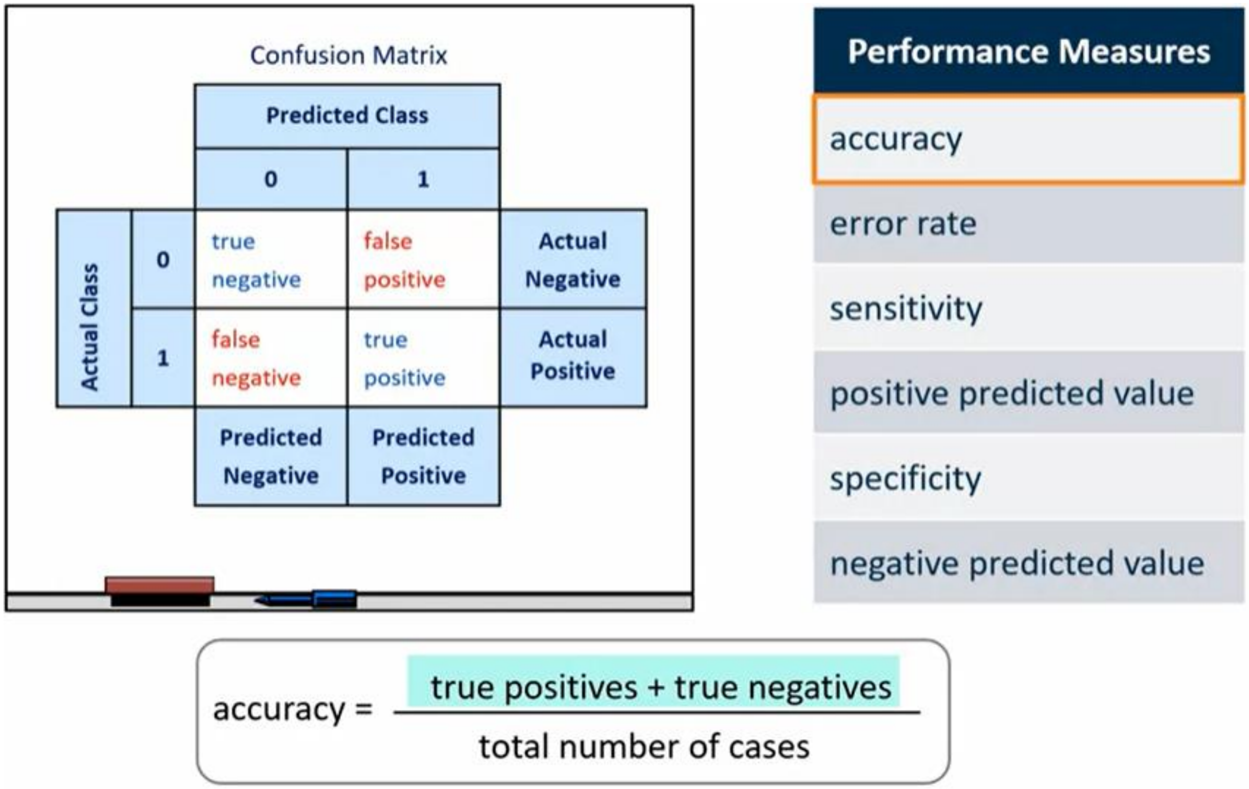

Confusion Matrix
Predicted Class
0
1
Actual Class
0
1
true negative
false positive
Actual Negative
false negative
true positive
Actual Positive
Predicted Negative
Predicted Positive
positive
Class 1
Class 0
negative

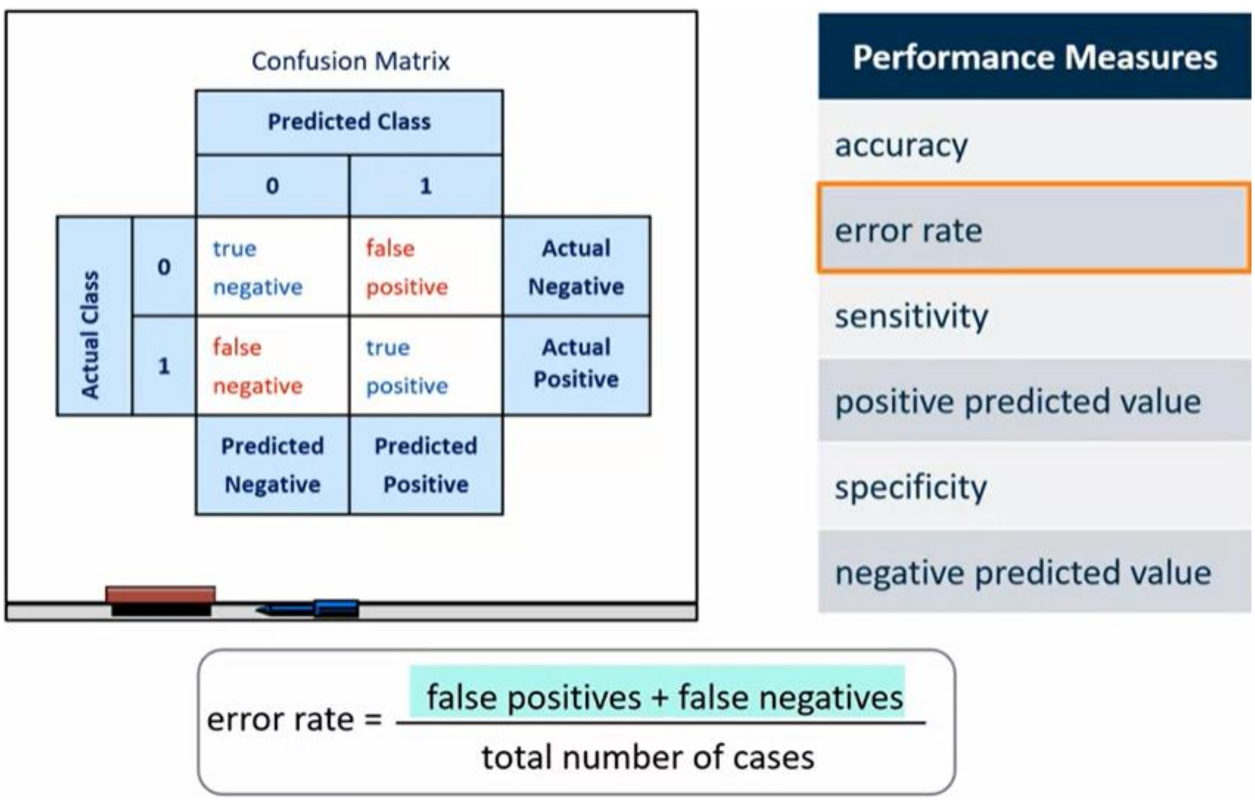

Confusion Matrix
Predicted Class
0
1
Actual Class
0
1
true negative
false positive
Actual Negative
false negative
true positive
Actual Positive
Predicted Negative
Predicted Positive
purchase:
1=yes
0=no
Class 1
Class 0

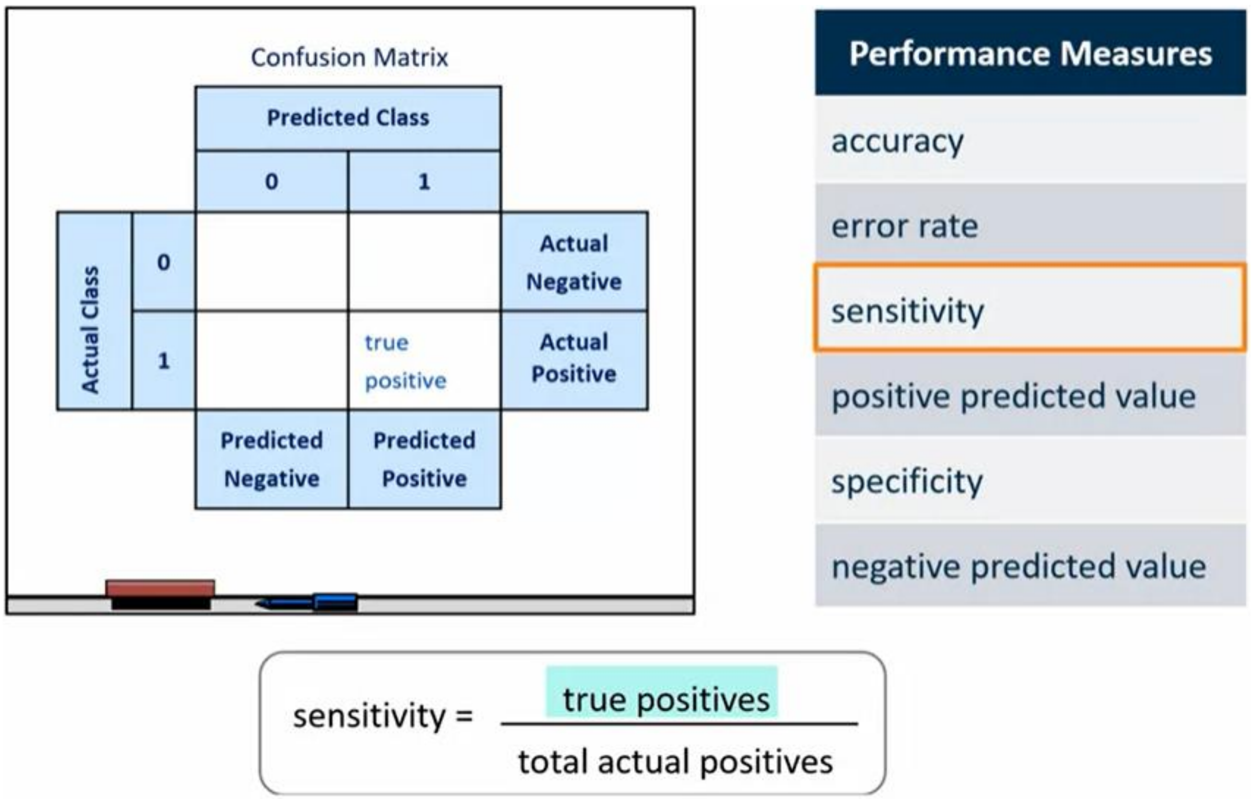

Confusion Matrix
Predicted Class
0
1
Actual Class
0
1
true negative
false positive
Actual Negative
false negative
true positive
Actual Positive
Predicted Negative
Predicted Positive
purchase:
1=yes
0=no
Class 1
Class 0

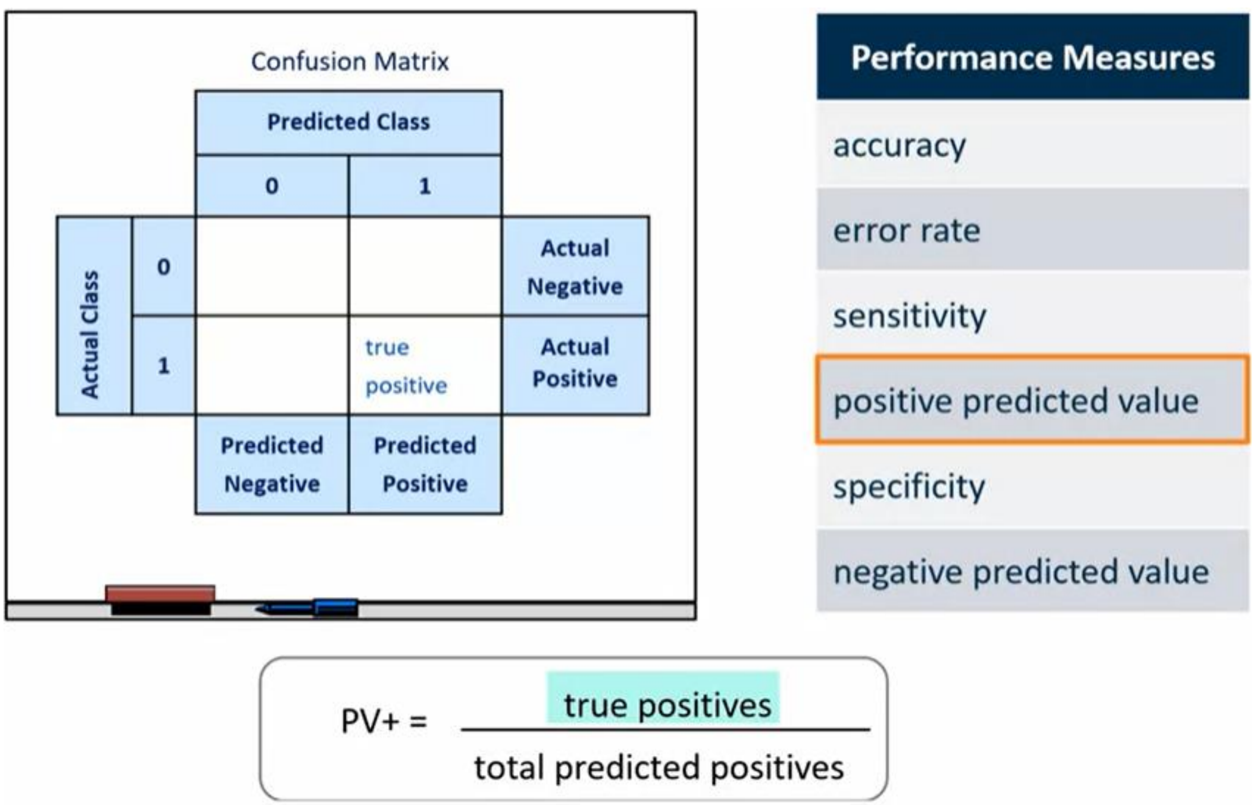

Confusion Matrix
Predicted Class
0
1
Actual Class
0
1
true negative
false positive
Actual Negative
false negative
true positive
Actual Positive
Predicted Negative
Predicted Positive
Class 1
Class 0
purchase:
1=yes
0=no

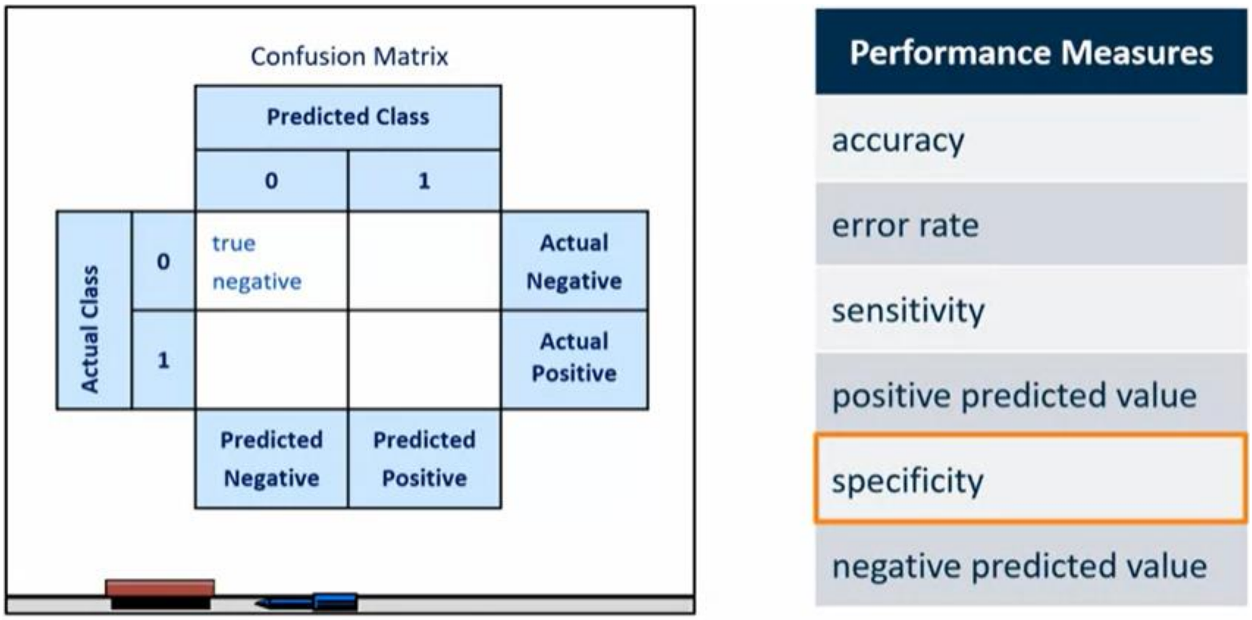

Confusion Matrix
Predicted Class
0
1
Actual Class
0
1
true negative
false positive
Actual Negative
false negative
true positive
Actual Positive
Predicted Negative
Predicted Positive
Class 1
Class 0
purchase:
1=yes
0=no


Confusion Matrix
Predicted Class
0
1
Actual Class
0
true negative
false positive
Actual Negative
1
false negative
true positive
Actual Positive
Predicted Negative
Predicted Positive
Performance Measures
accuracy
error rate
sensitivity
positive predicted value
specificity
negative predicted value
accuracy = true positives + true negatives / total number of cases

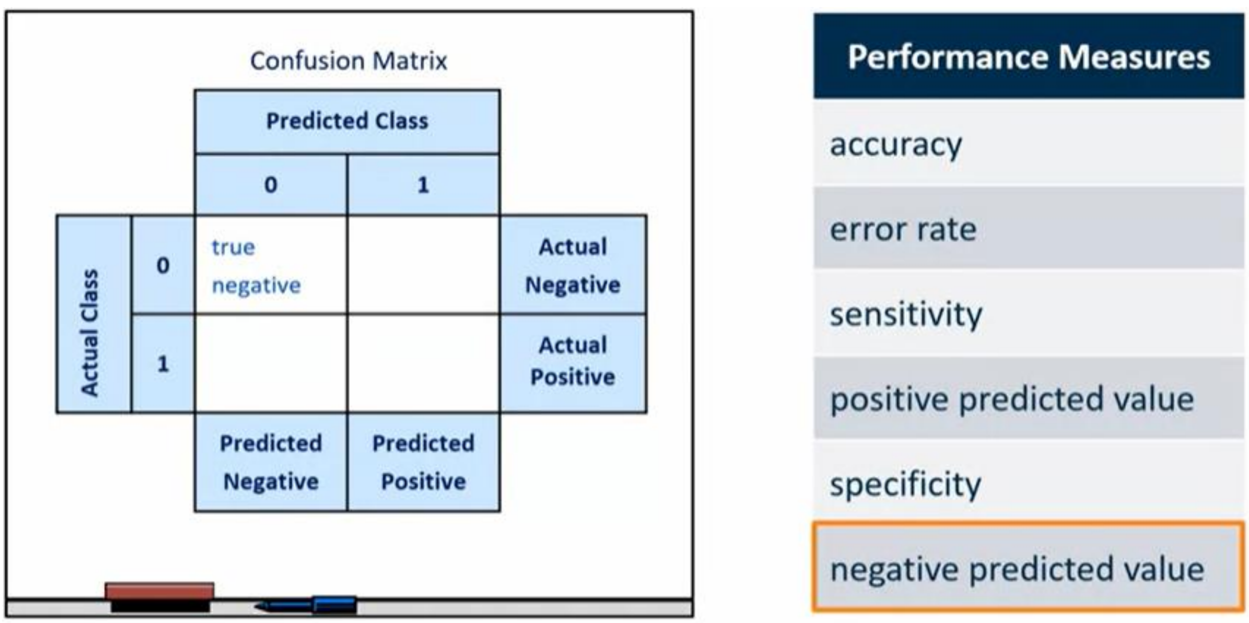

Confusion Matrix
Predicted Class
0
1
Actual Class
0
true negative
false positive
Actual Negative
1
false negative
true positive
Actual Positive
Predicted Negative
Predicted Positive
Performance Measures
accuracy
error rate
sensitivity
positive predicted value
specificity
negative predicted value
error rate = false positives + false negatives / total number of cases


Confusion Matrix
Predicted Class
0
1
Actual Class
0
1
true
positive
Actual
Negative
Actual
Positive
Predicted
Negative
Predicted
Positive
Performance Measures
accuracy
error rate
sensitivity
positive predicted value
specificity
negative predicted value
sensitivity = true positives / total actual positives


Confusion Matrix
Predicted Class
0
1
Actual Class
0
1
true
positive
Actual
Negative
Actual
Positive
Predicted
Negative
Predicted
Positive
Performance Measures
accuracy
error rate
sensitivity
positive predicted value
specificity
negative predicted value
PV+ = true positives / total predicted positives

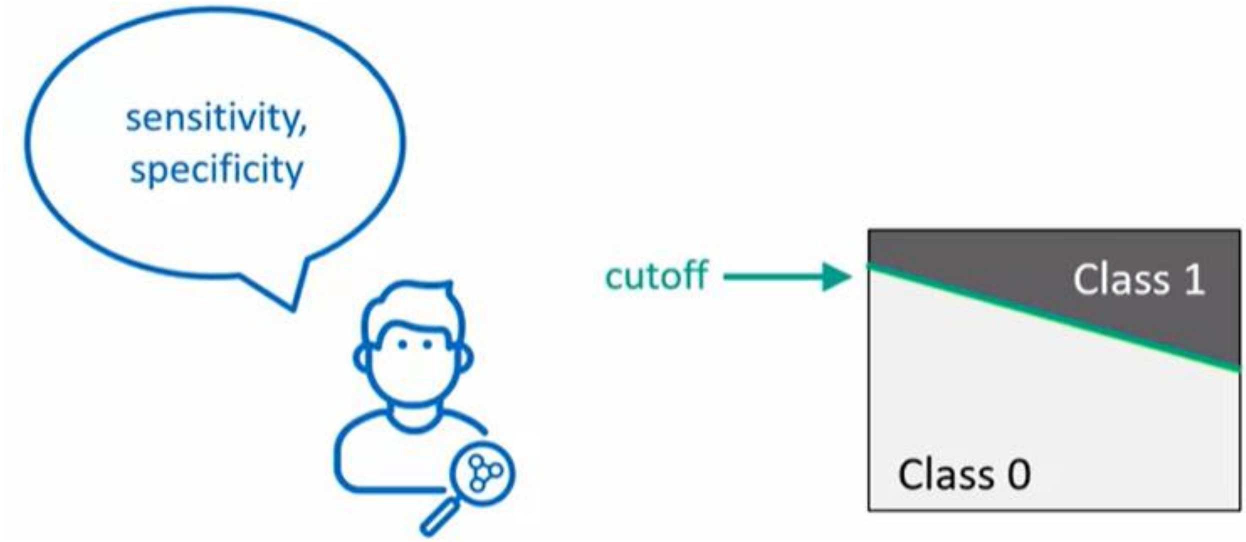


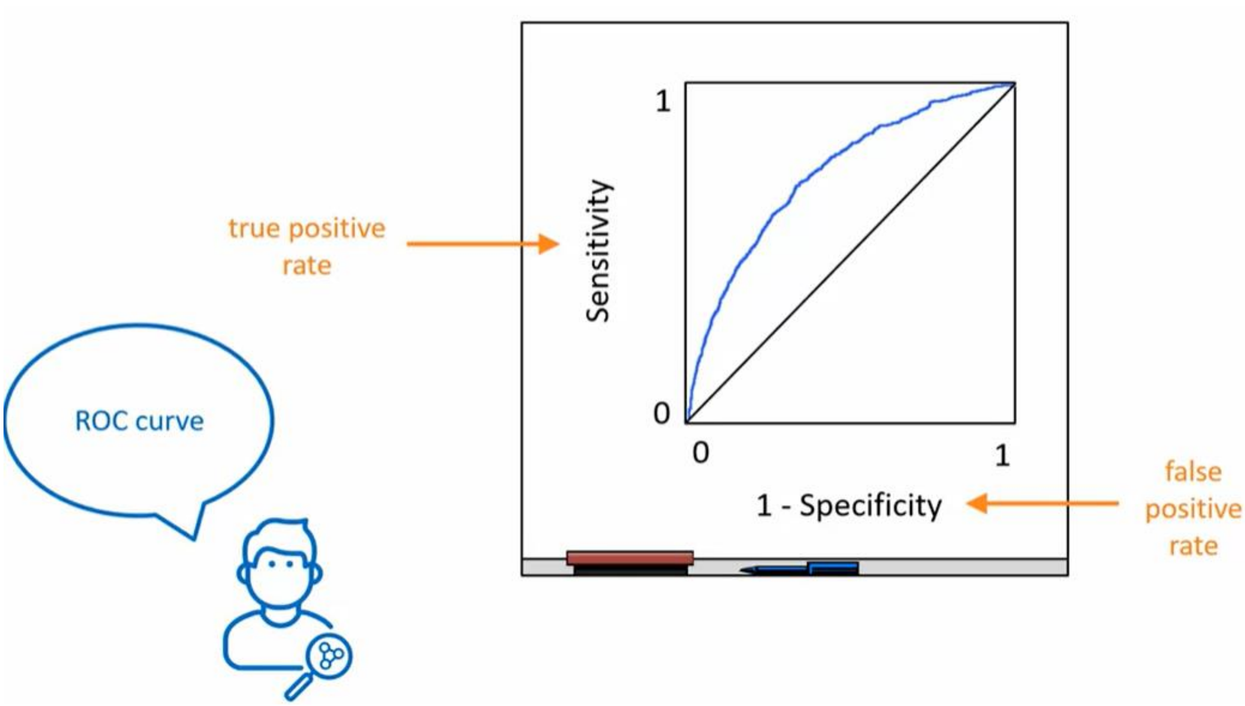
$$\text{specificity} = \frac{\text{true negatives}}{\text{total actual negatives}}$$

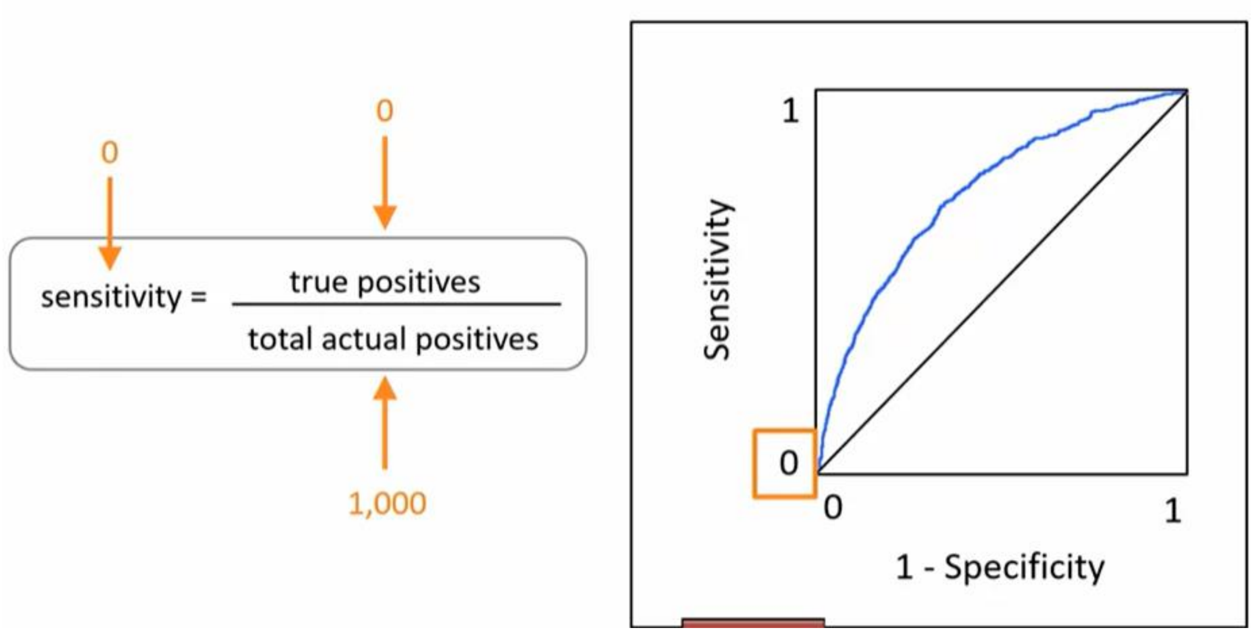


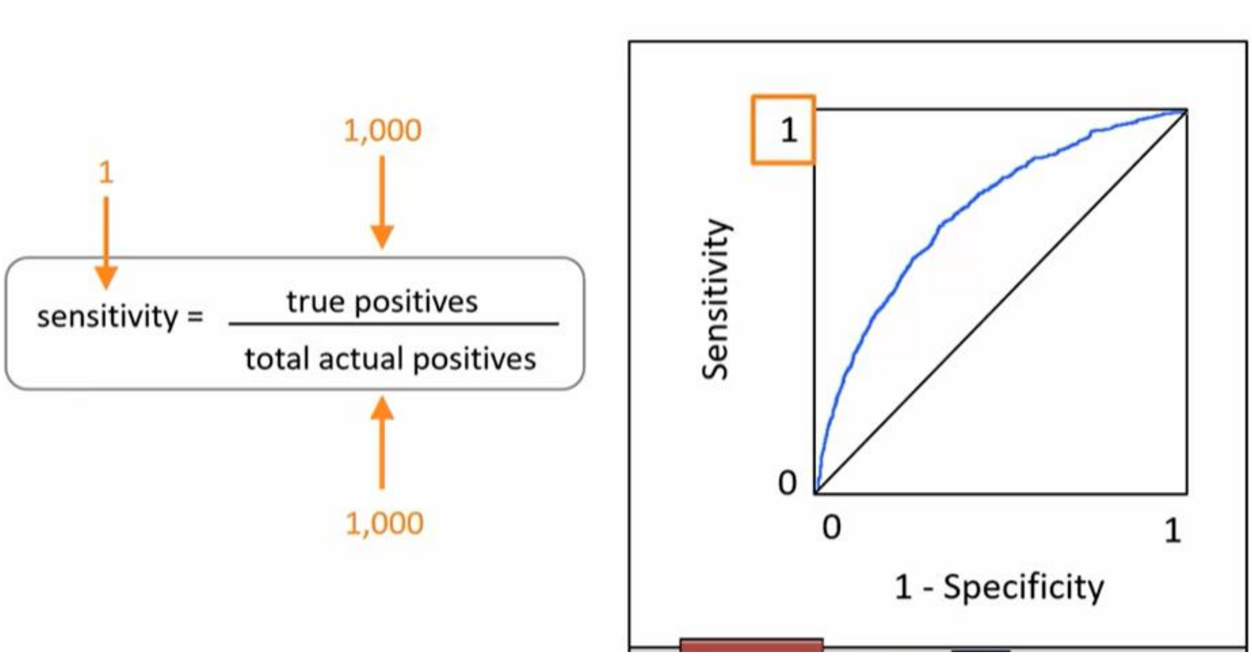
$$\text{PV-} = \frac{\text{true negatives}}{\text{total predicted negatives}}$$

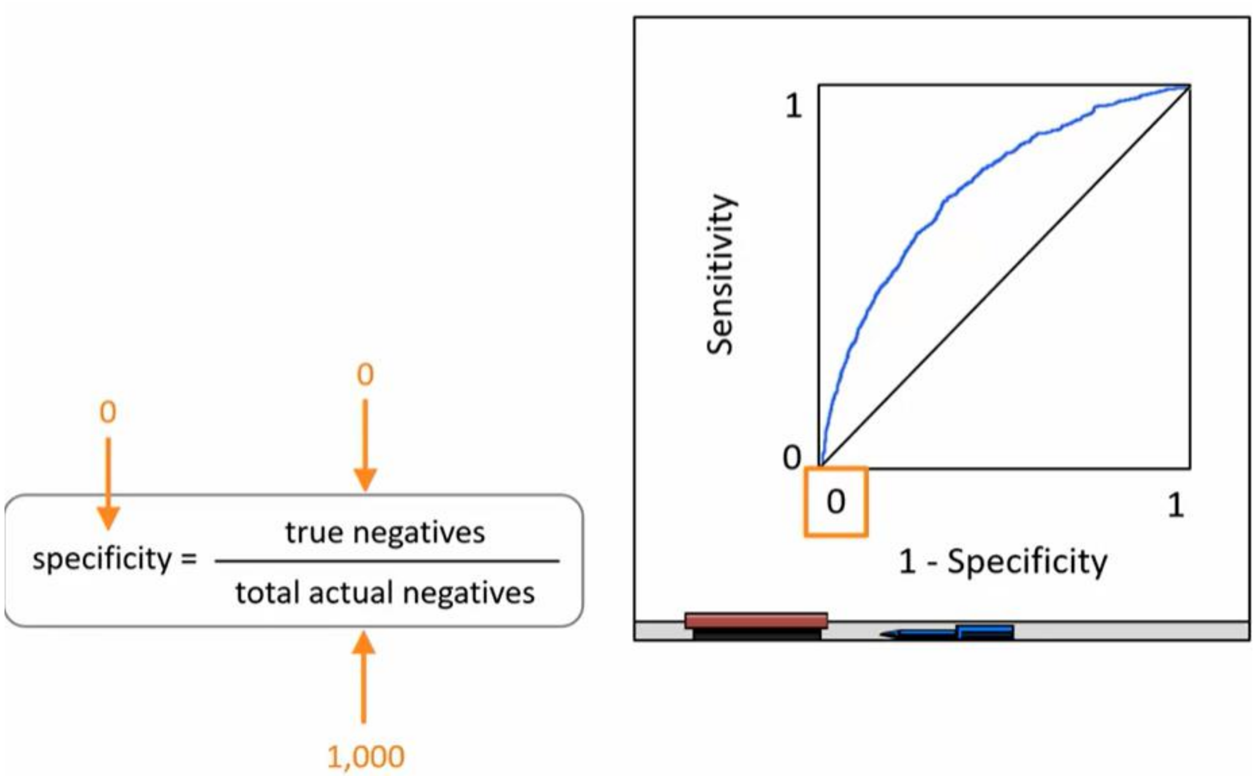
Measuring Performance across Cutoffs by Using the ROC Curve

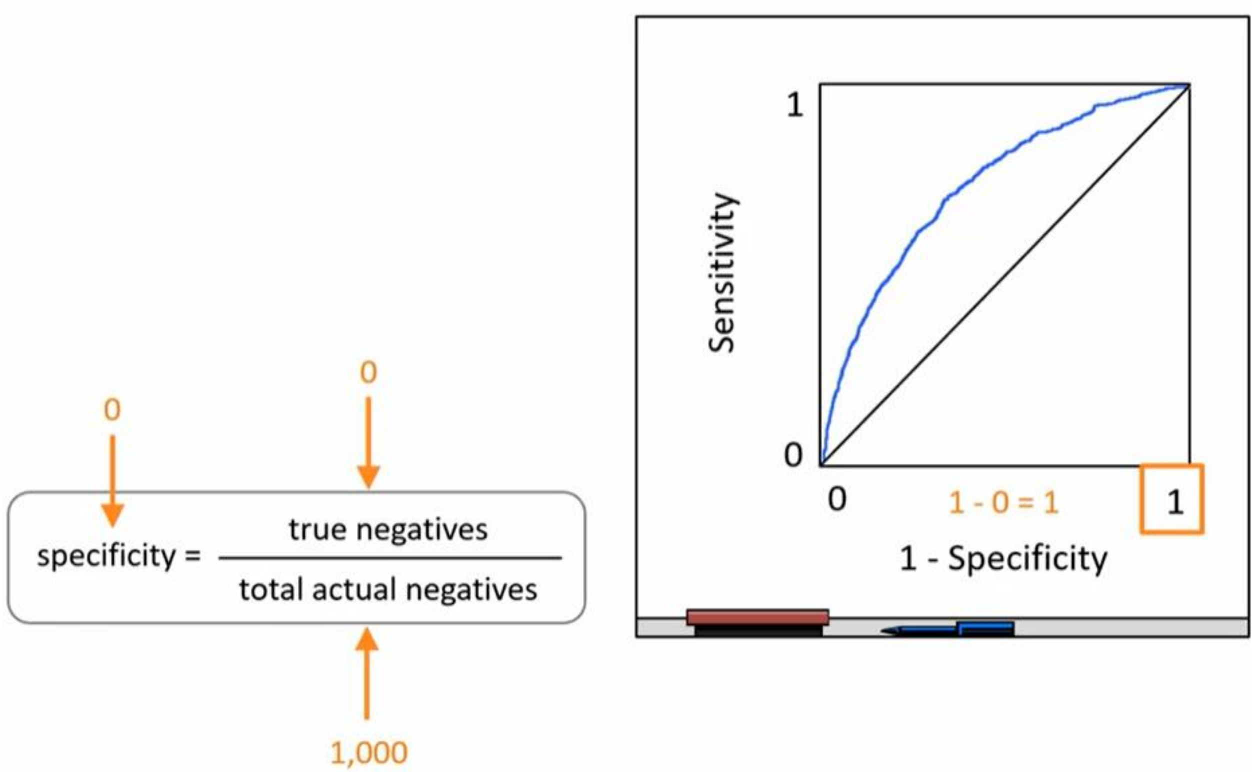


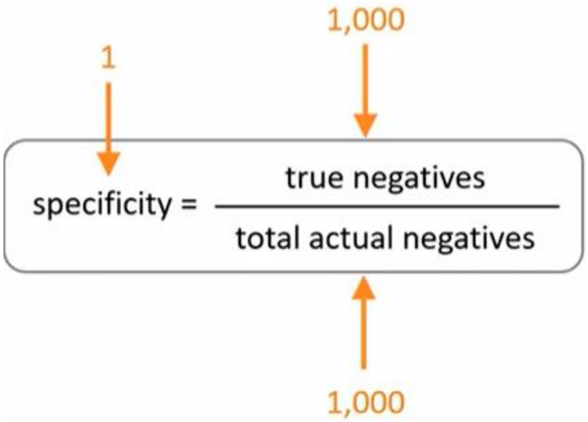


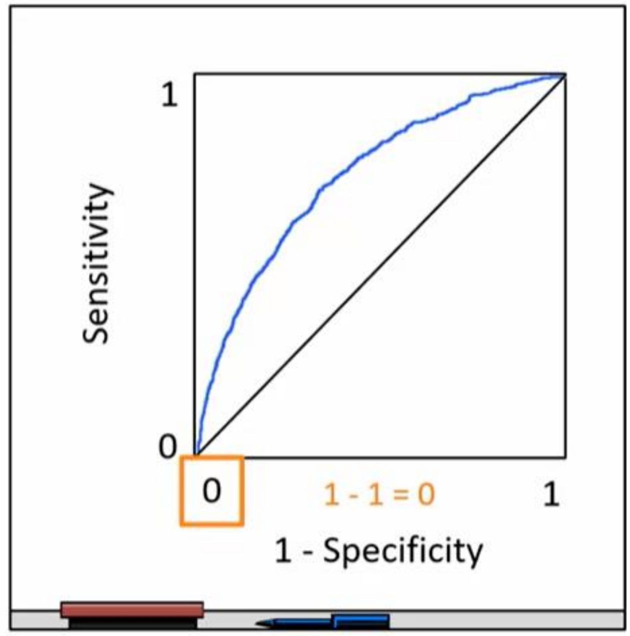

0
0
sensitivity =
true positives
total actual positives
1,000
1
0
Sensitivity
0
1
1 - Specificity


1
1,000
sensitivity =
true positives
total actual positives
1,000
1
0
Sensitivity
0
1
1 - Specificity

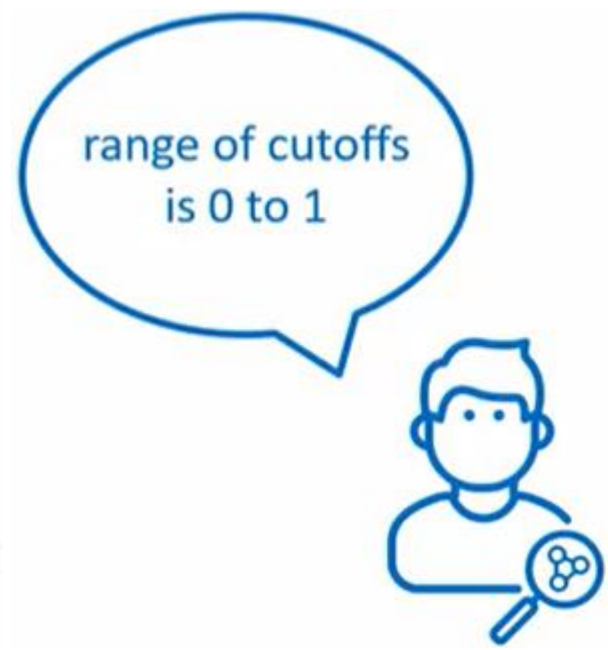

0
0
specificity = true negatives / total actual negatives
1,000
1
Sensitivity
0
0
1
1 - Specificity

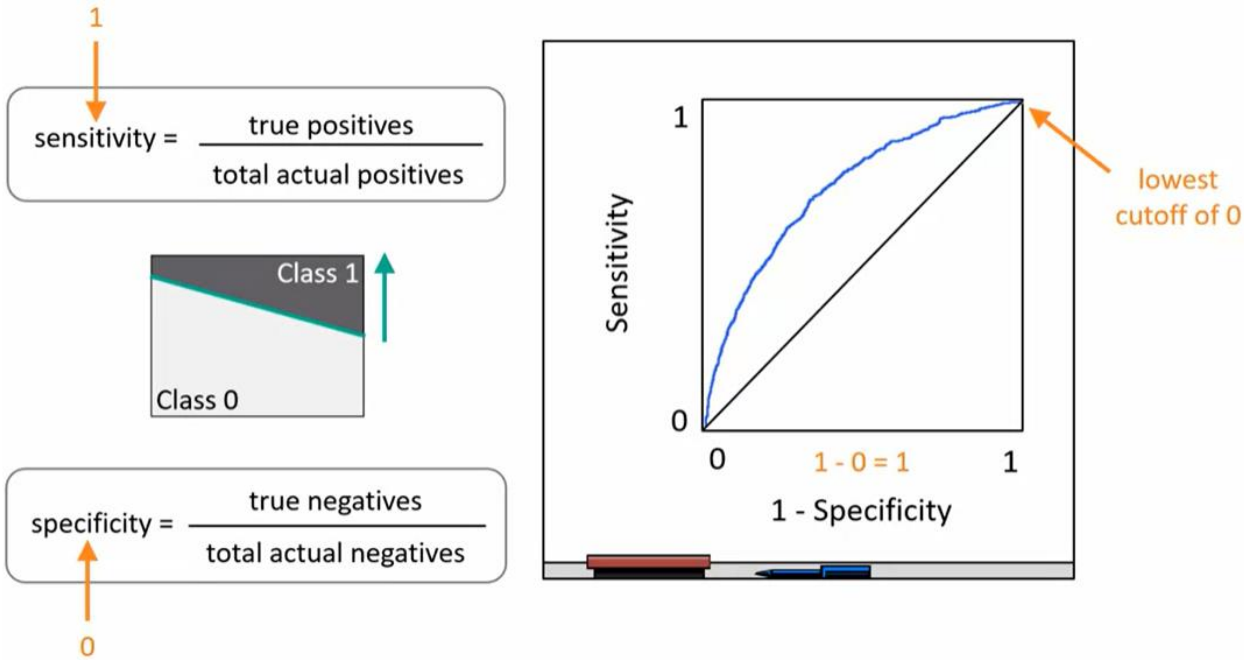

0
0
specificity = true negatives / total actual negatives
1,000
1
Sensitivity
0
0
1 - 0 = 1
1
1 - Specificity

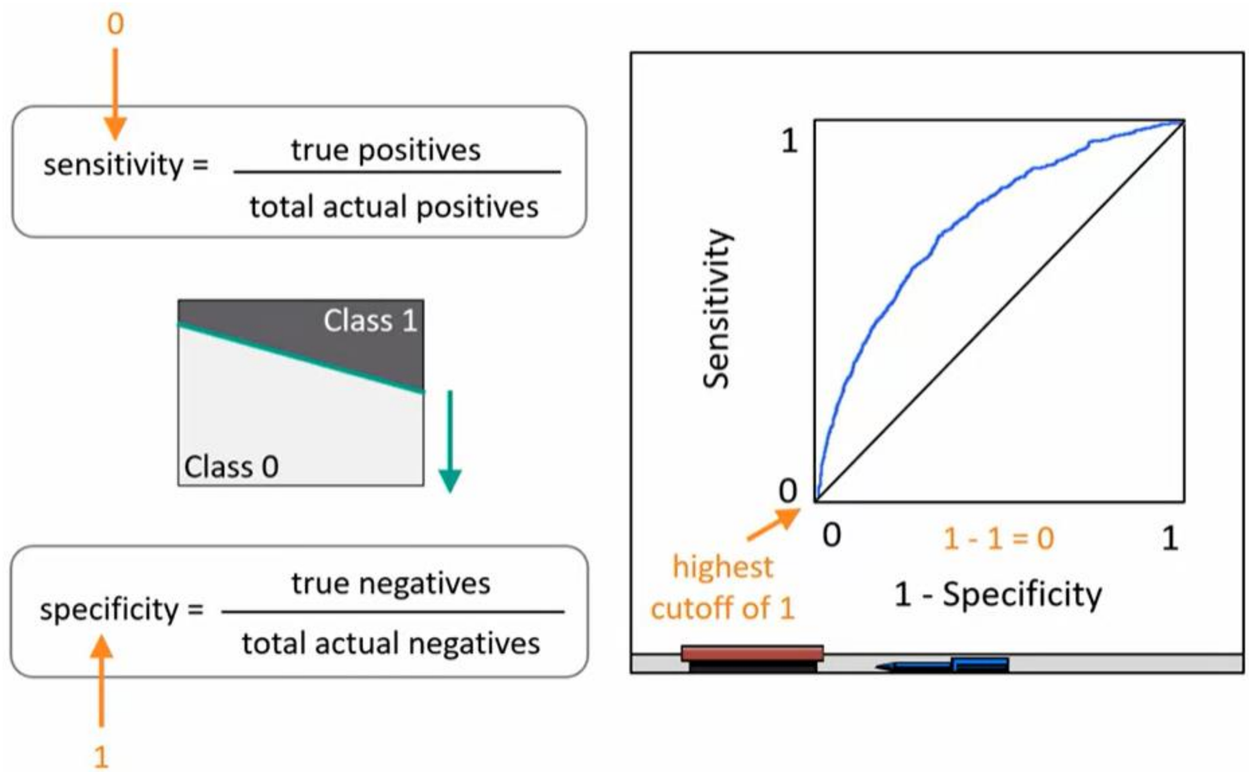

1
1,000
specificity =
true negatives
total actual negatives
1,000

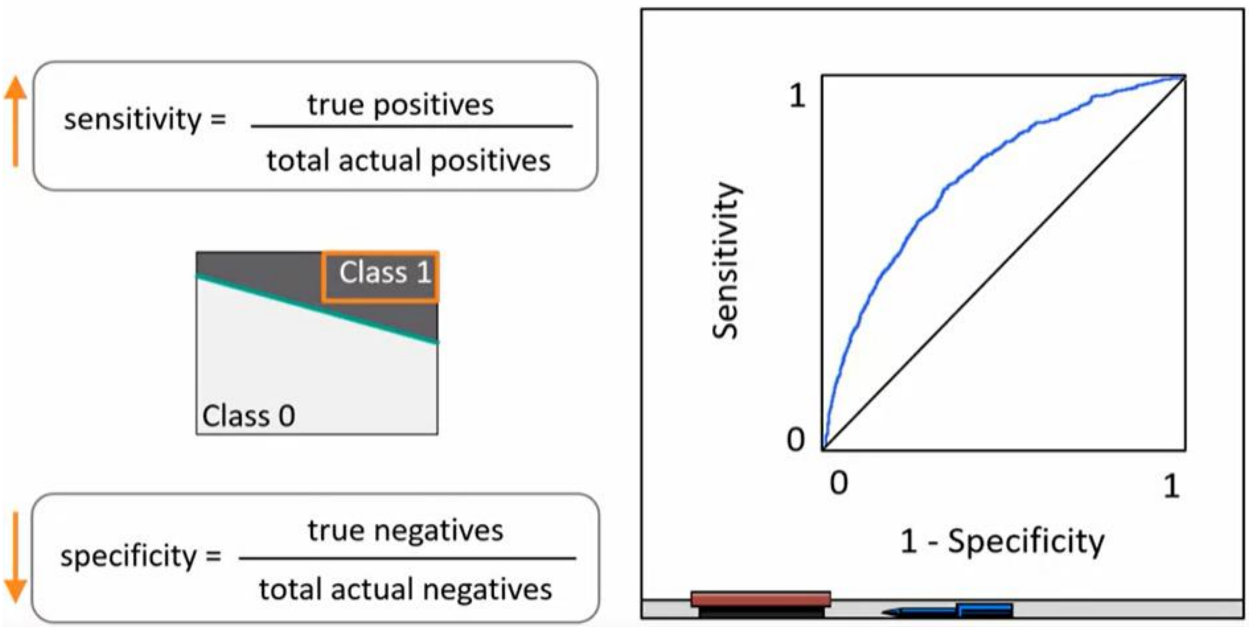

1
Sensitivity
0
0
1 - 1 = 0
1
1 - Specificity

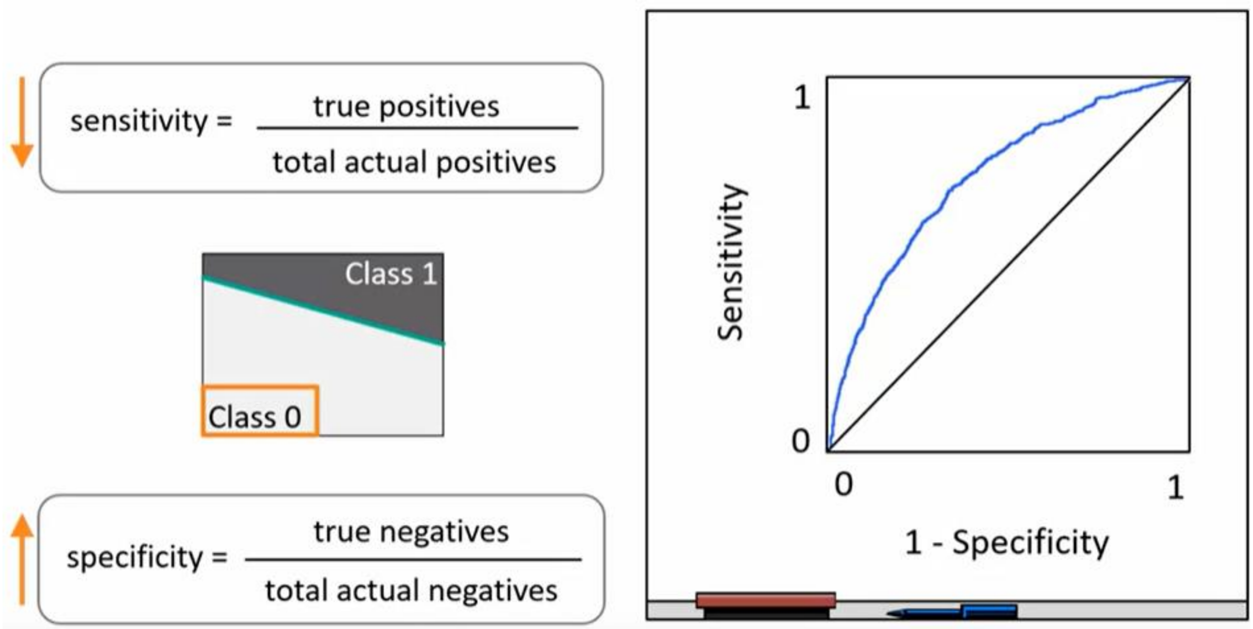

entire range of cutoffs

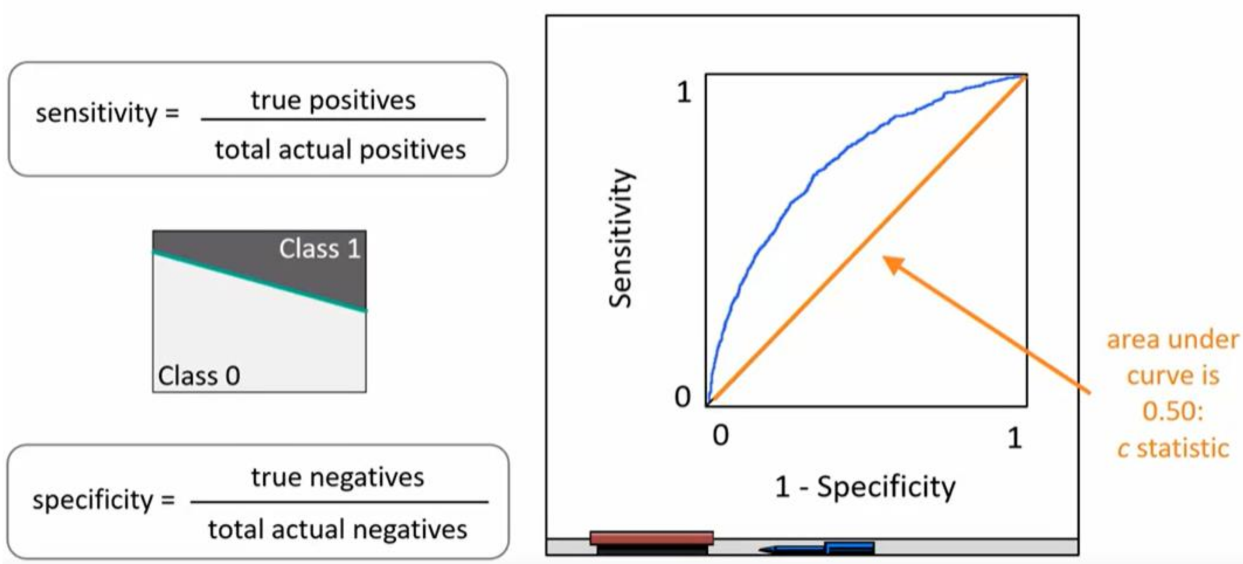

range of cutoffs is 0 to 1

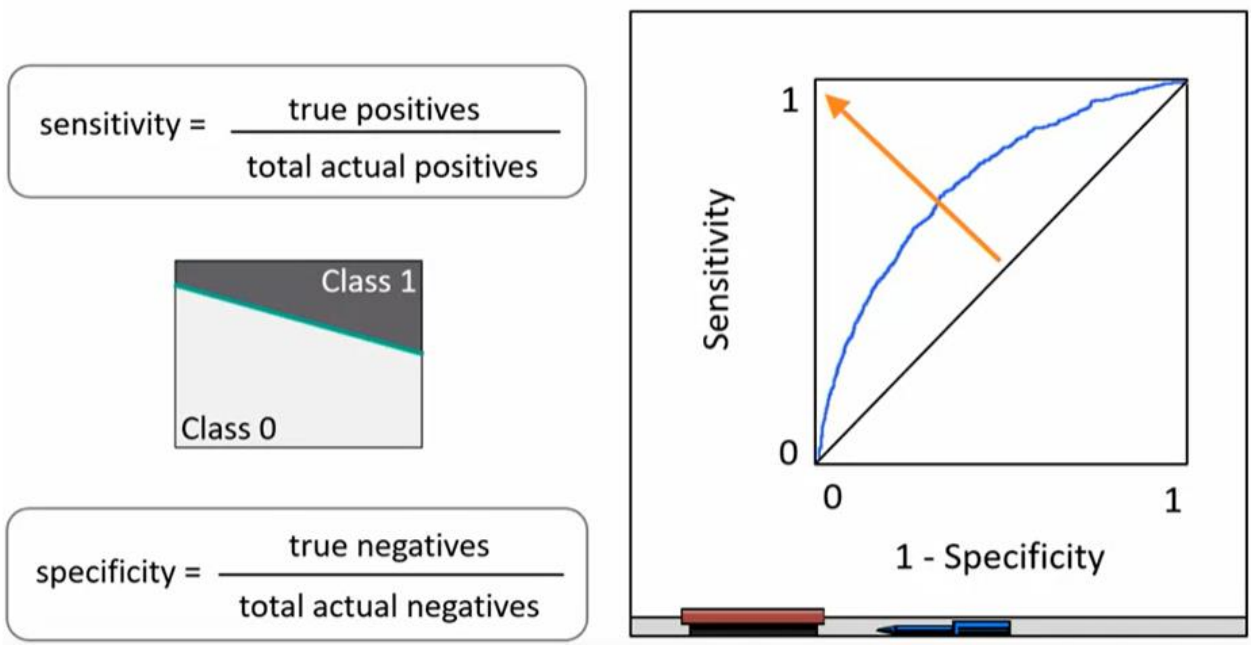

1
sensitivity = true positives / total actual positives
Class 1
Class 0
specificity = true negatives / total actual negatives
0
1
Sensitivity
0
0
1 - 0 = 1
1
1 - Specificity
lowest
cutoff of 0


0
sensitivity = true positives / total actual positives
Class 1
Class 0
specificity = true negatives / total actual negatives
1
1
Sensitivity
0
0
1 - 1 = 0
1
1 - Specificity
highest
cutoff of 1

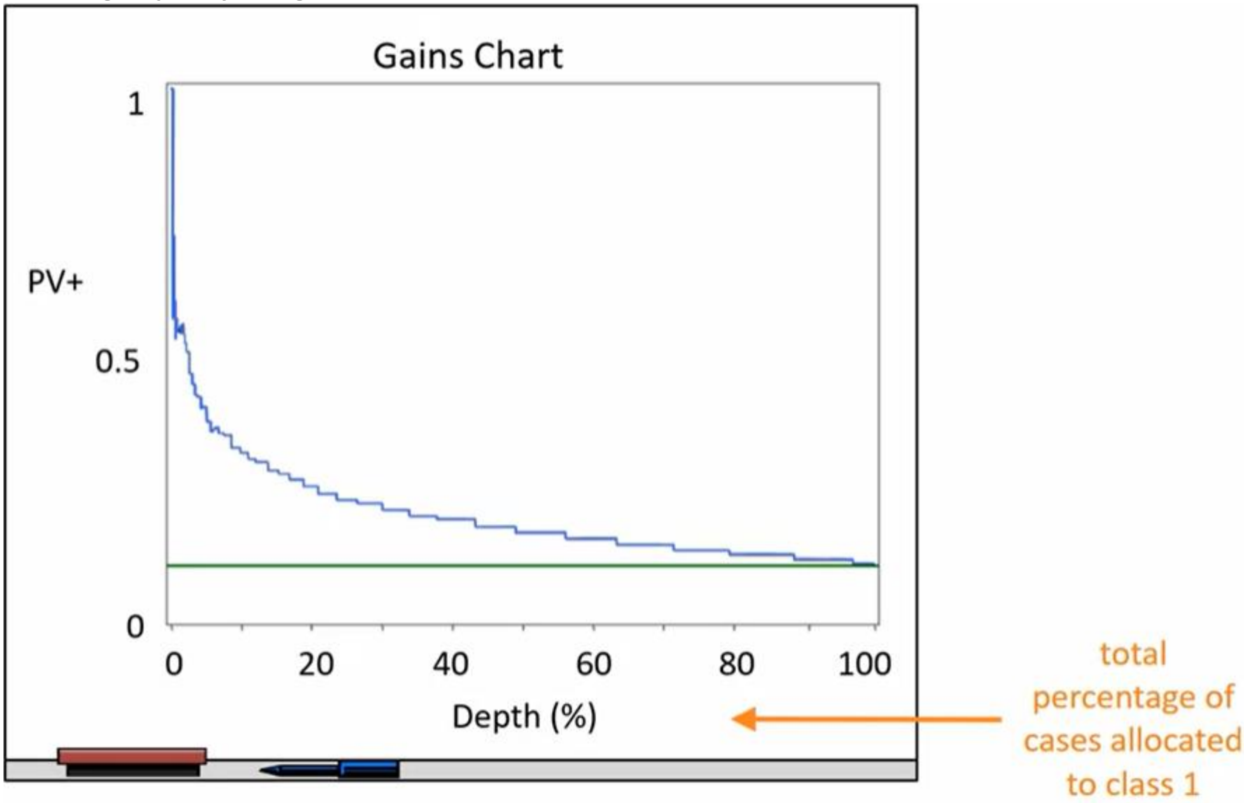

sensitivity = true positives / total actual positives
Class 1
Class 0
specificity = true negatives / total actual negatives
1
Sensitivity
0
0
1
1 - Specificity

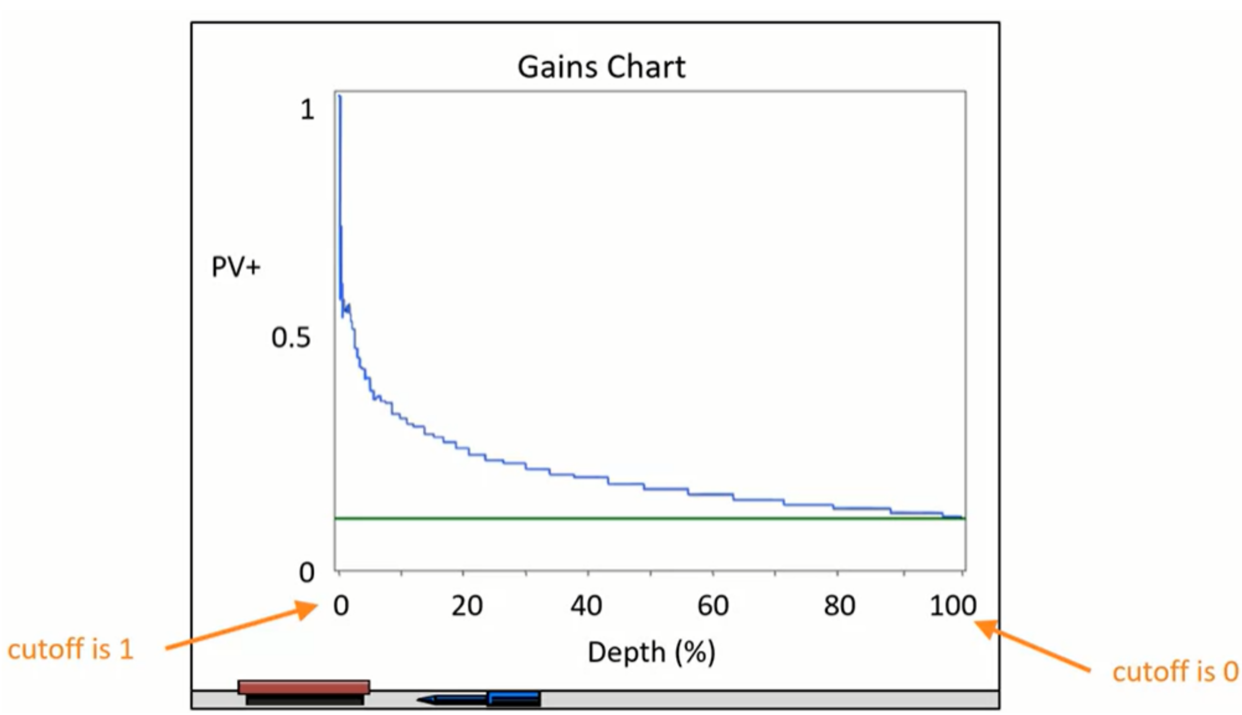

sensitivity = true positives / total actual positives
Class 1
Class 0
specificity = true negatives / total actual negatives
1
Sensitivity
0
0
1
1 - Specificity

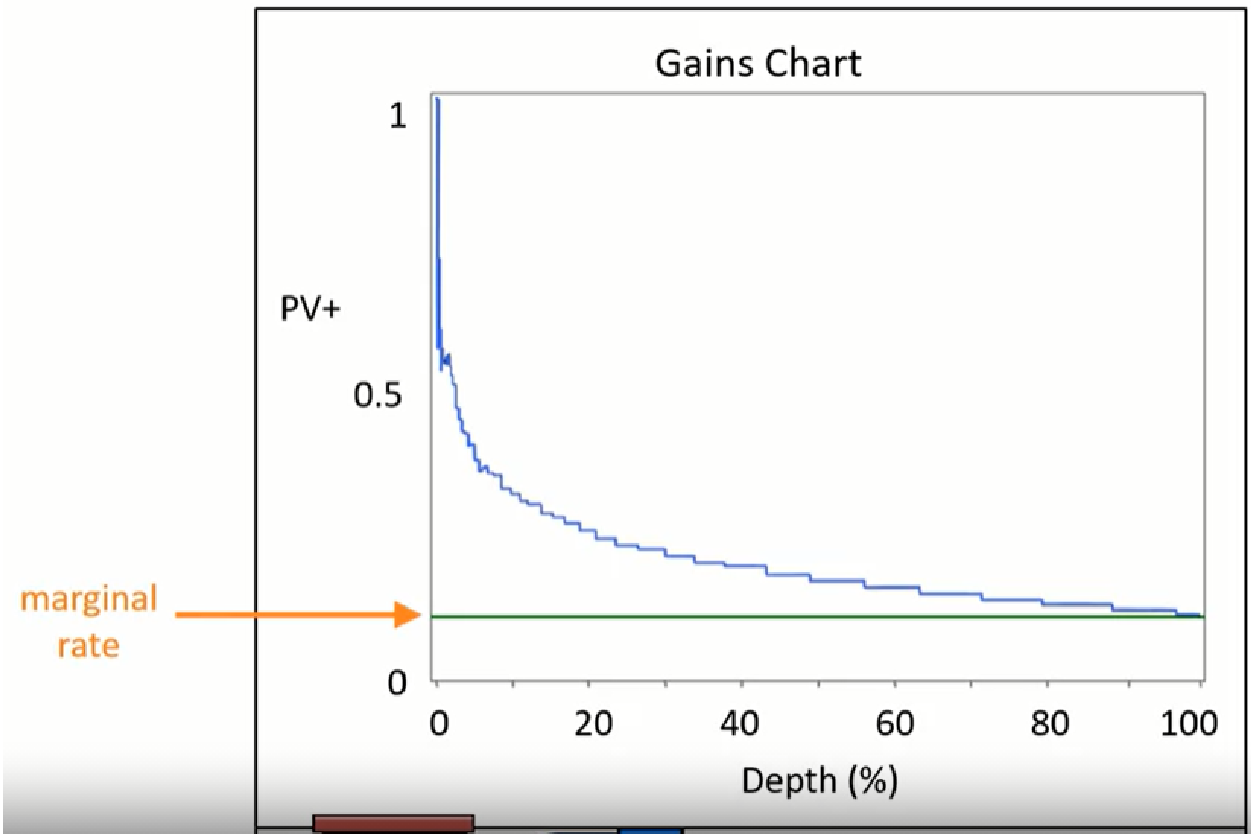

sensitivity = true positives / total actual positives
Class 1
Class 0
specificity = true negatives / total actual negatives
1
Sensitivity
0
0
1
1 - Specificity
area under
curve is
0.50:
c statistic

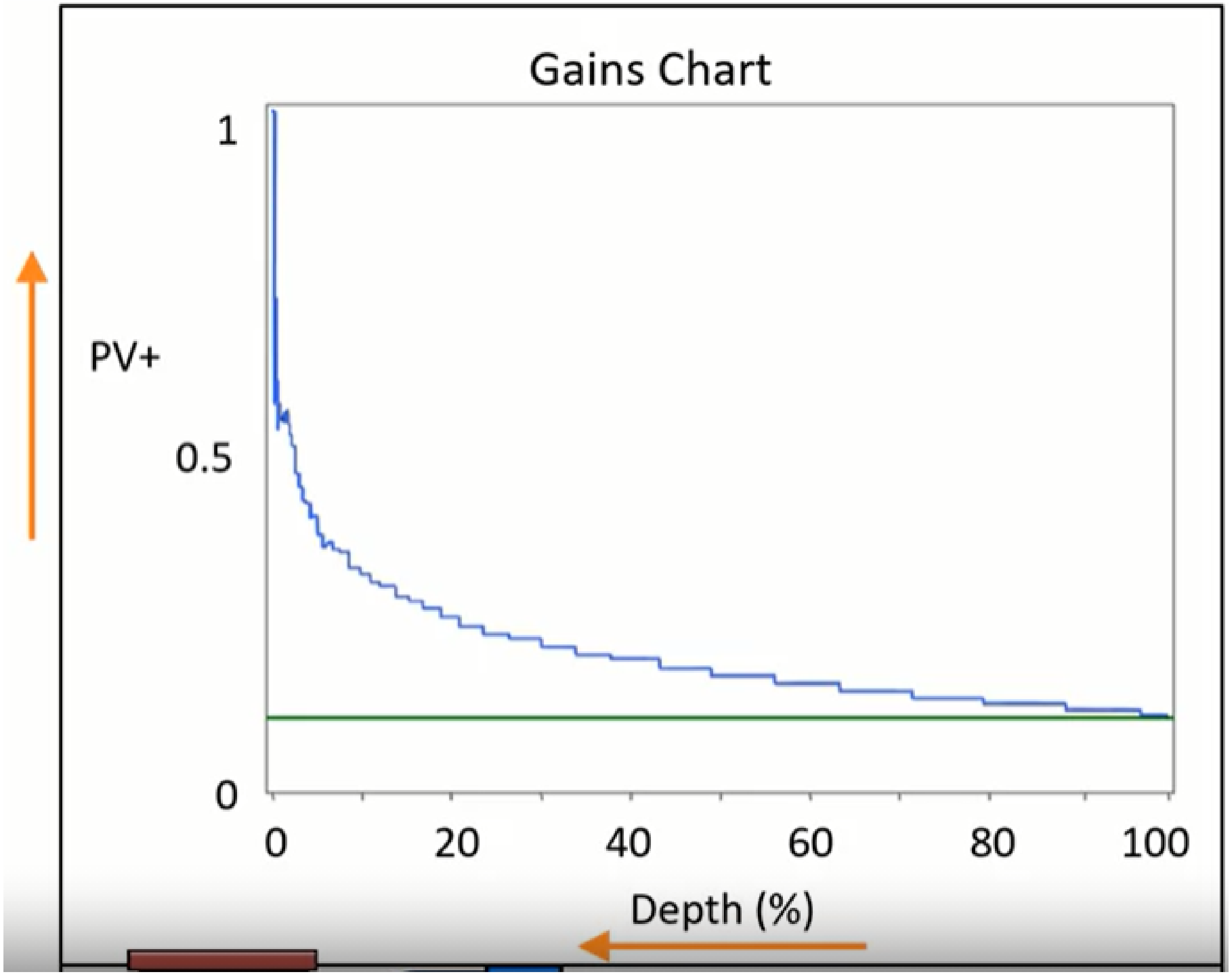

sensitivity = true positives / total actual positives
Class 1
Class 0
specificity = true negatives / total actual negatives
1
Sensitivity
0
0
1
1 - Specificity

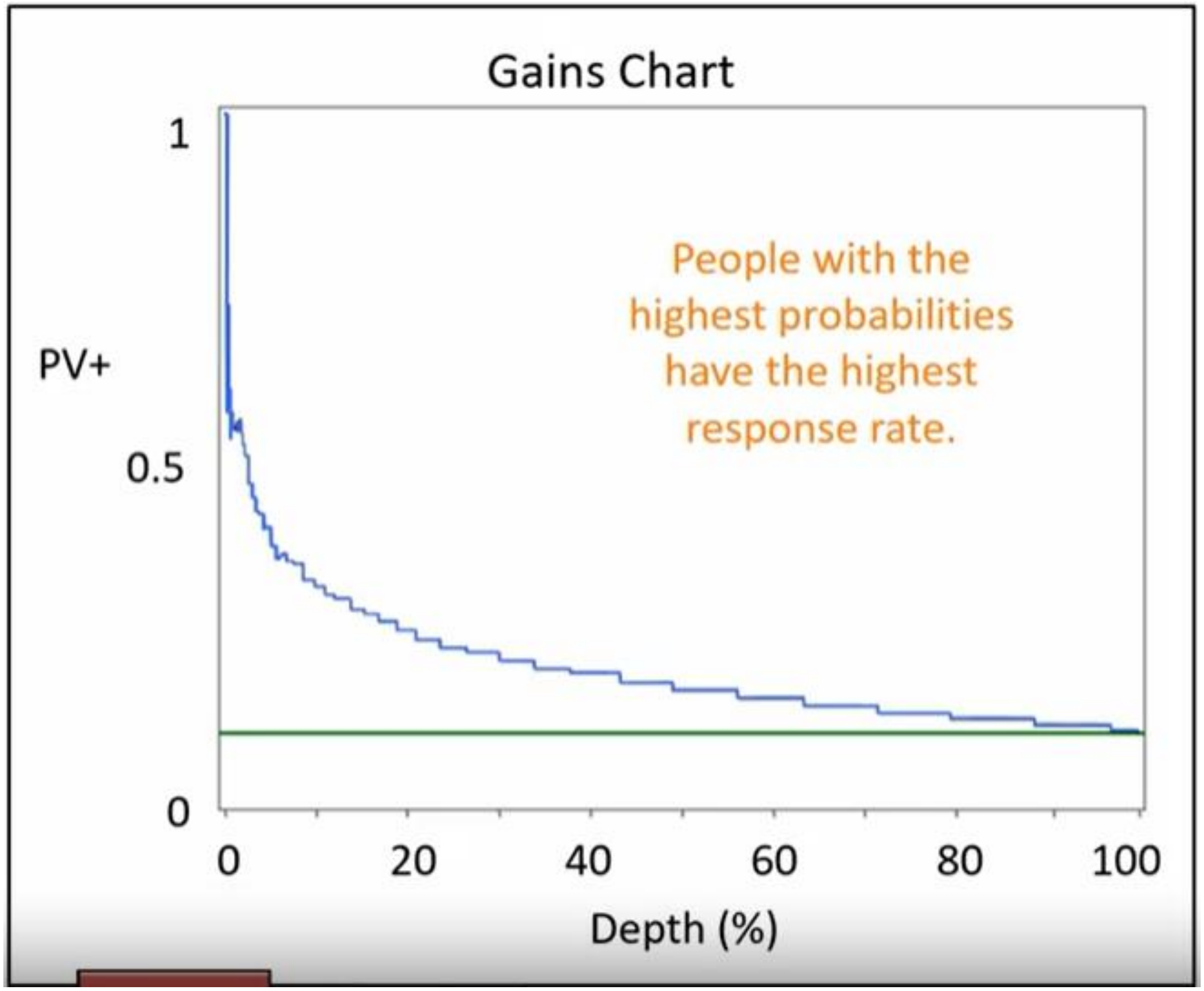
Choosing Depth by Using the Gains Chart

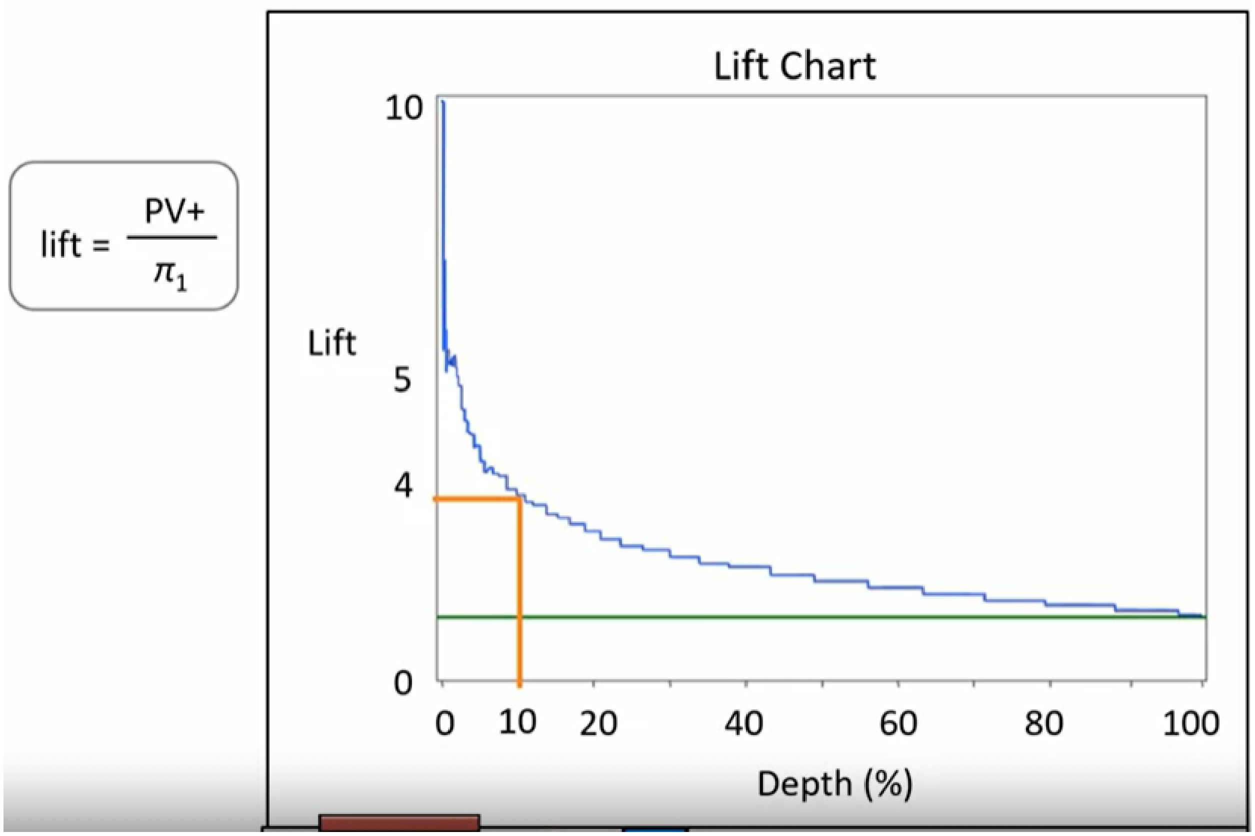




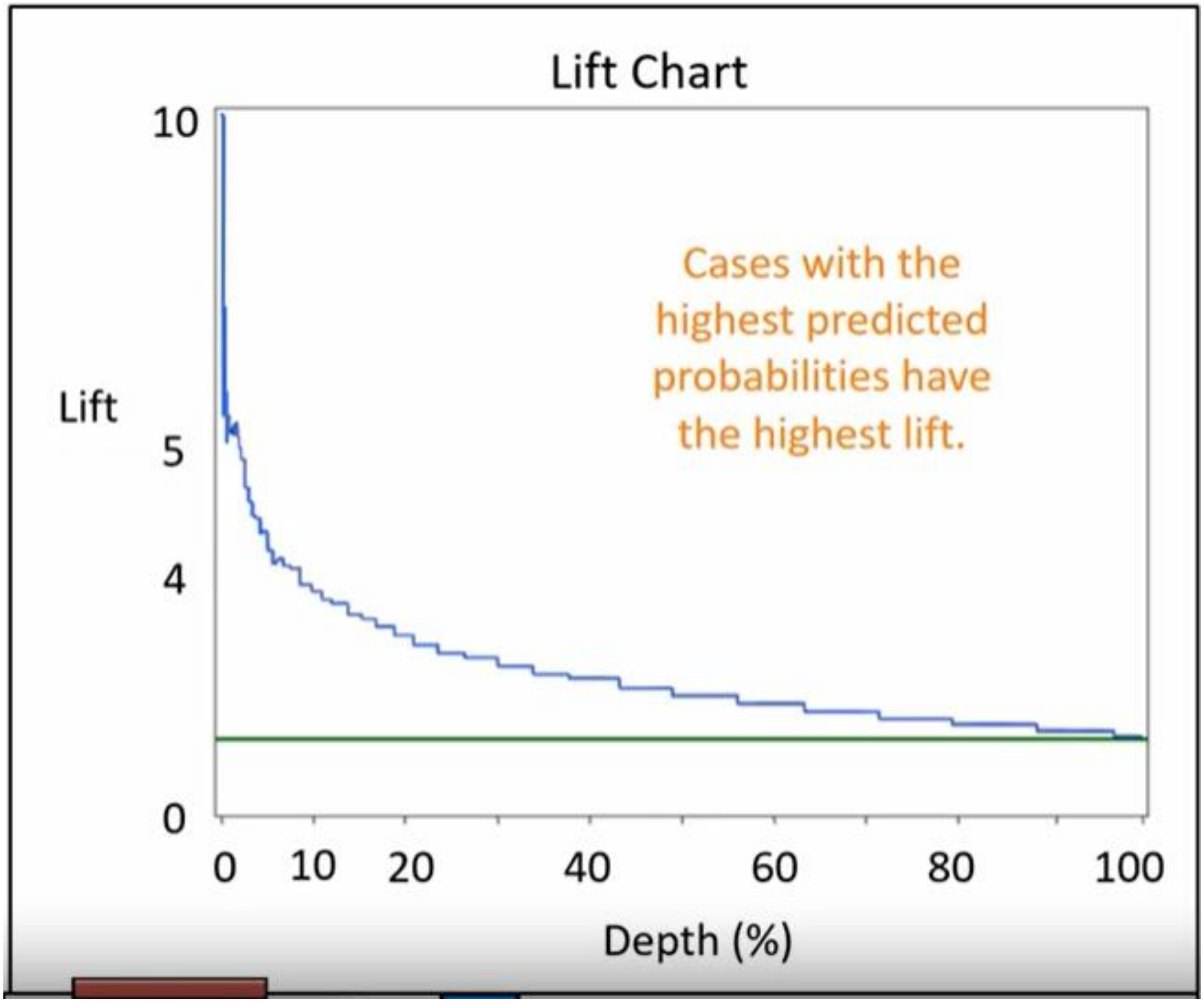

Gains Chart
1
PV+
0.5
0
marginal
rate
0
20
40
60
80
100
Depth (%)


Gains Chart
1
PV+
0.5
0
0
20
40
60
80
100
Depth (%)

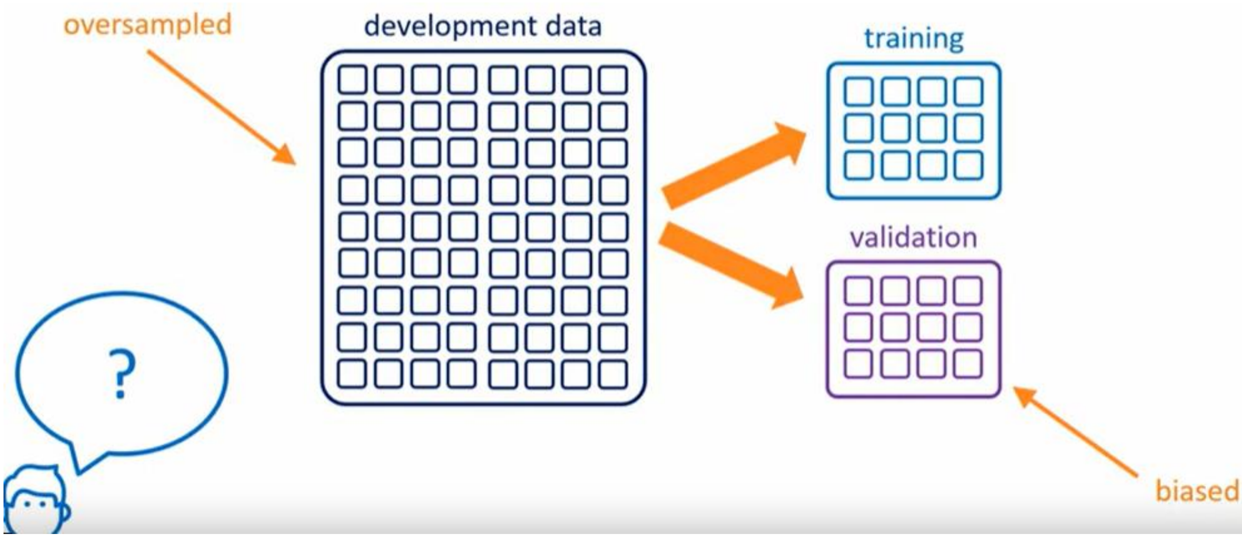

Gains Chart
1
PV+
0.5
0
People with the highest probabilities have the highest response rate.
0
20
40
60
80
100
Depth (%)

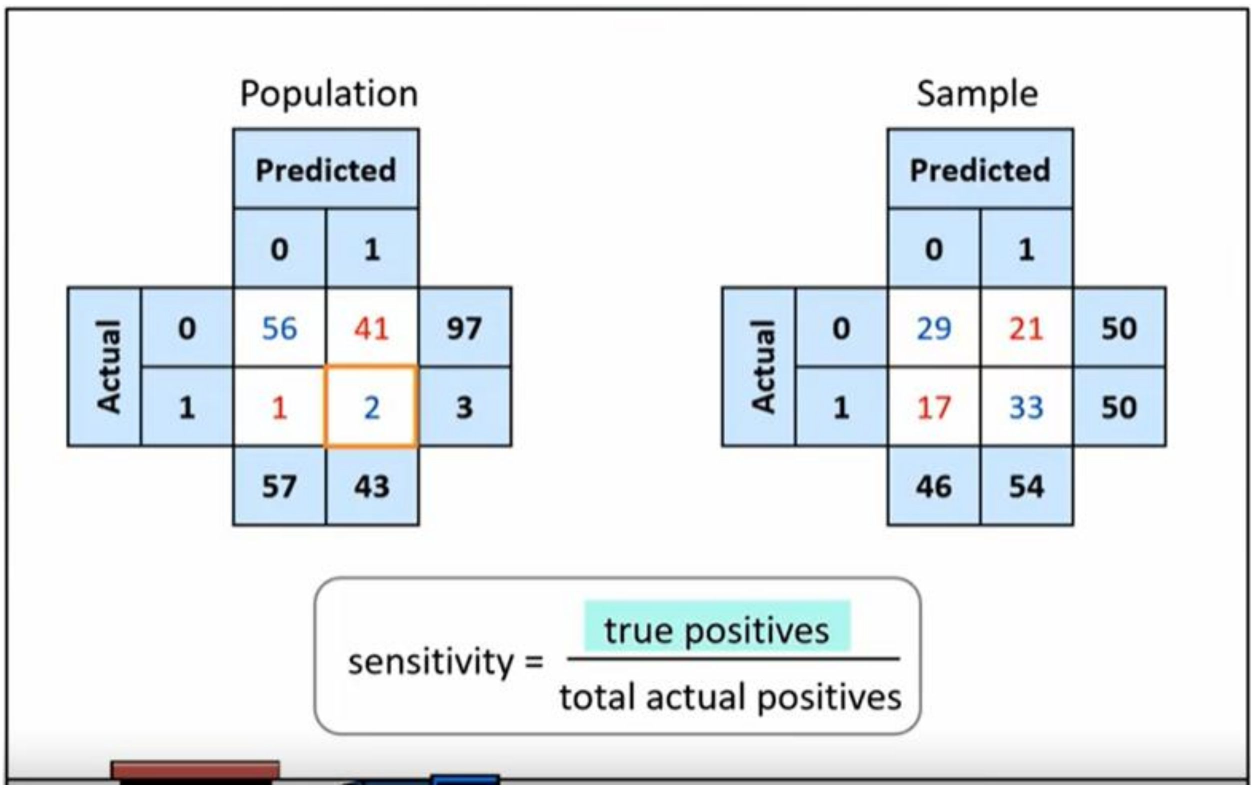


lift = PV+ / π1
Lift Chart
Lift
10
5
4
0
0
10
20
40
60
80
100
Depth (%)

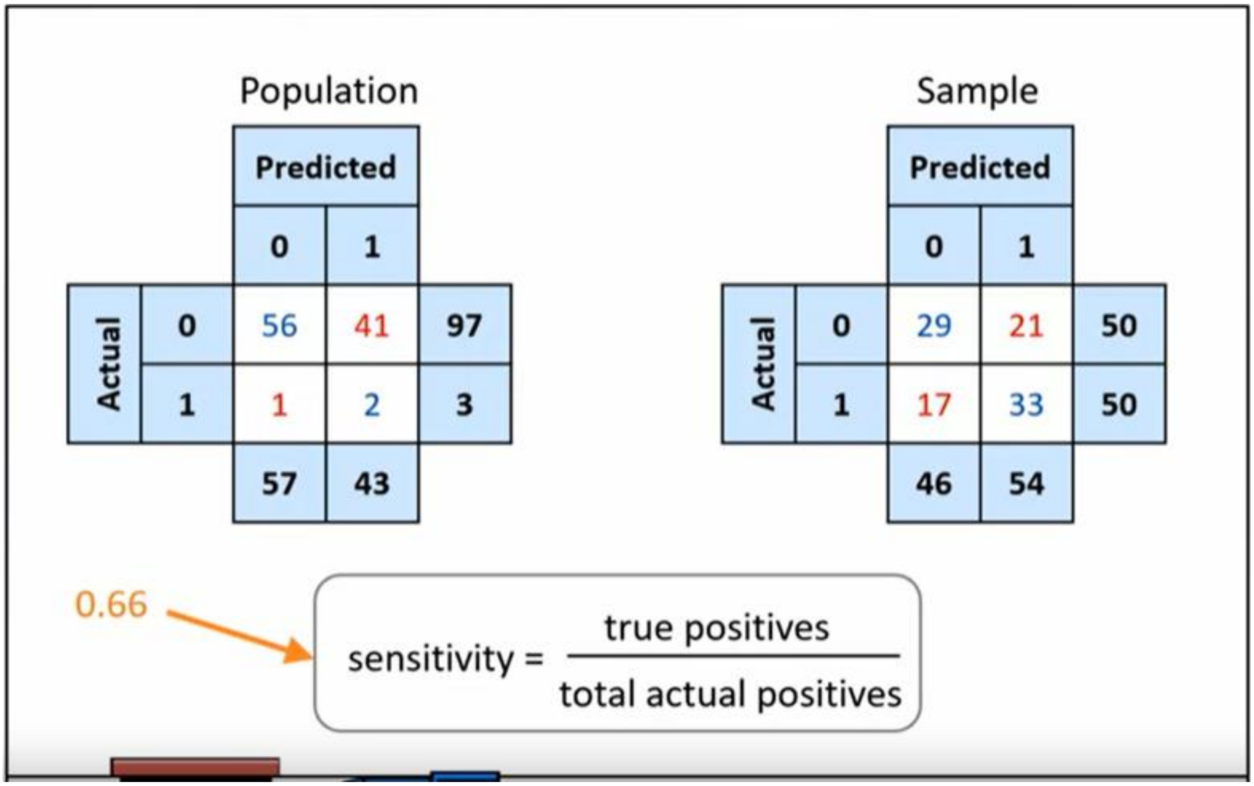

Lift Chart
10
5
4
0
Lift
Cases with the highest predicted probabilities have the highest lift.
0
10
20
40
60
80
100
Depth (%)

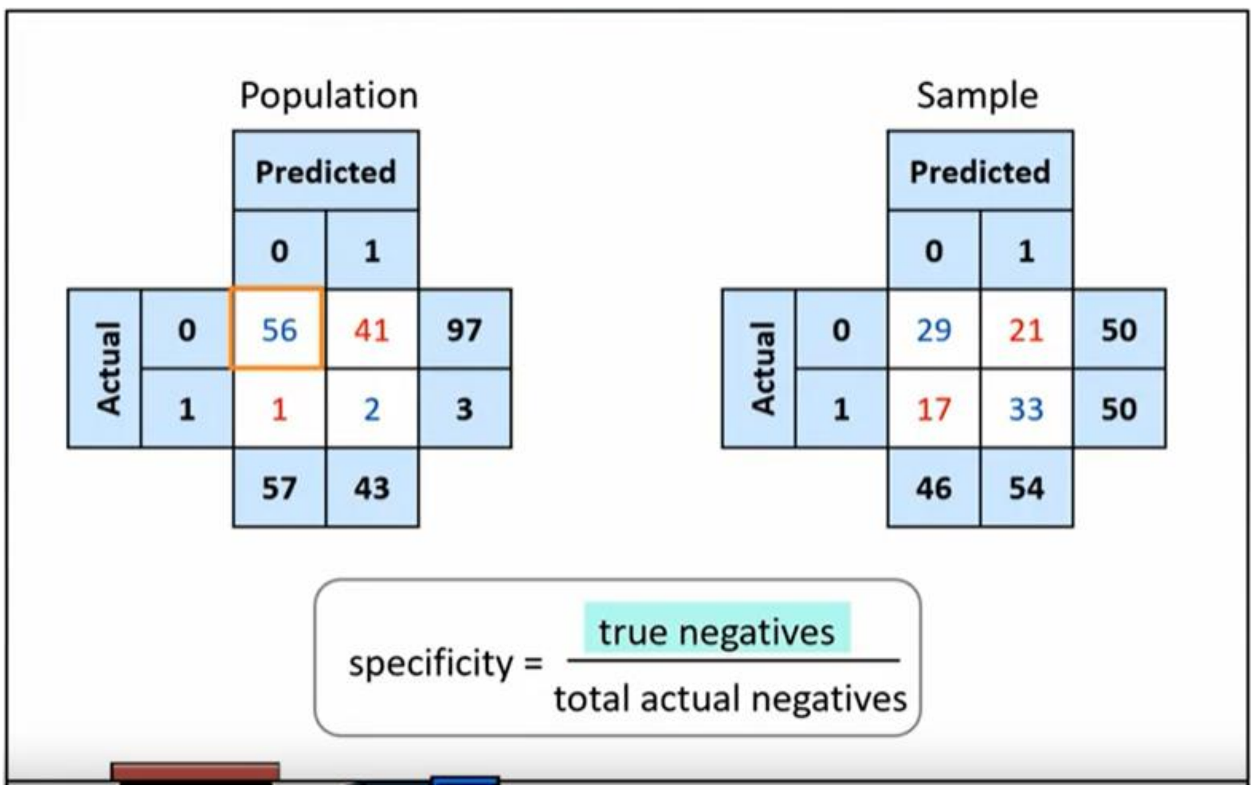
Effects of Oversampled Data on Performance Measures



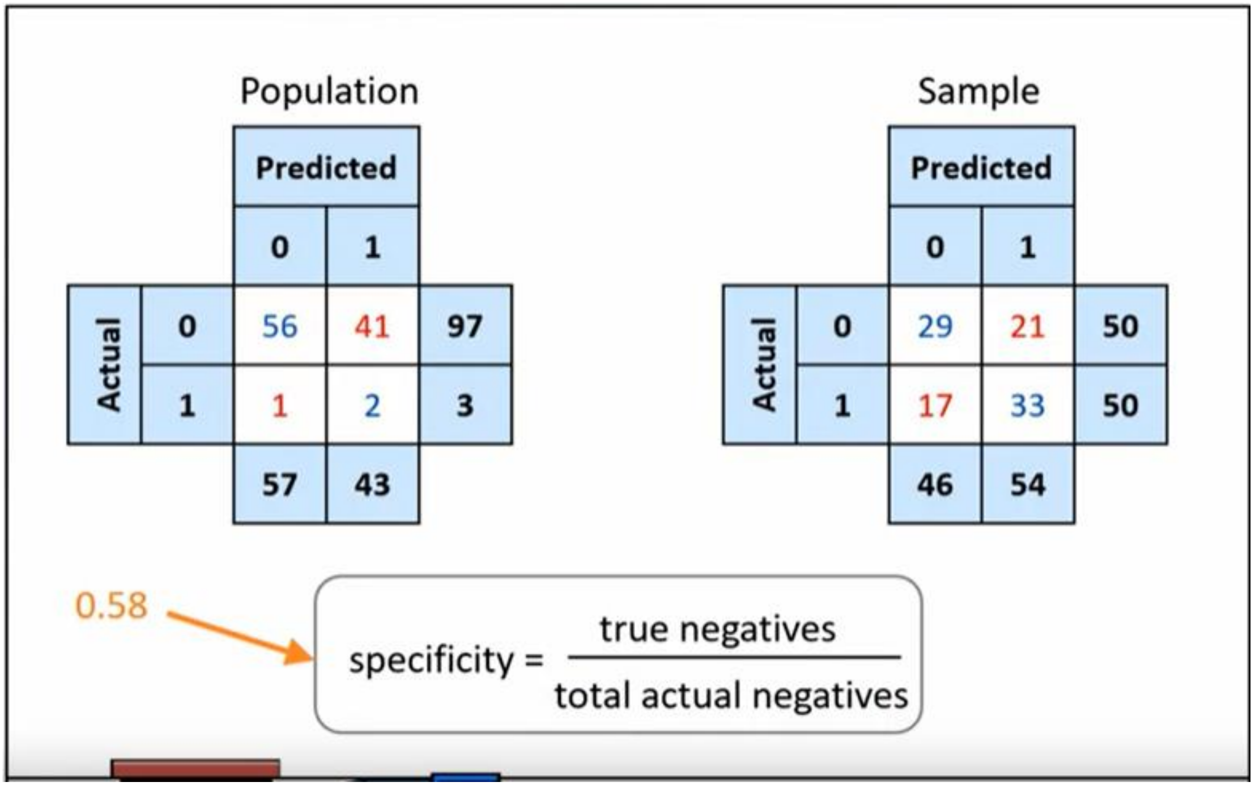


2 / 3 = 0.66 = 33 / 50 (both population and sample have same sensitivity)

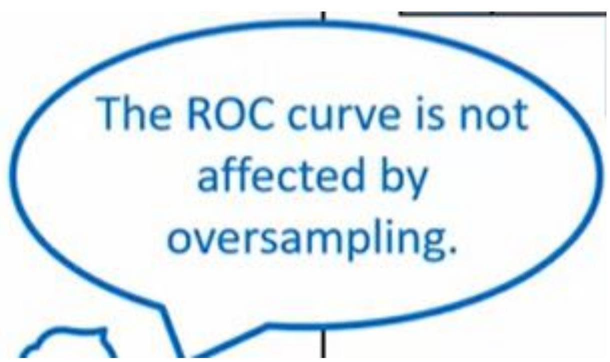


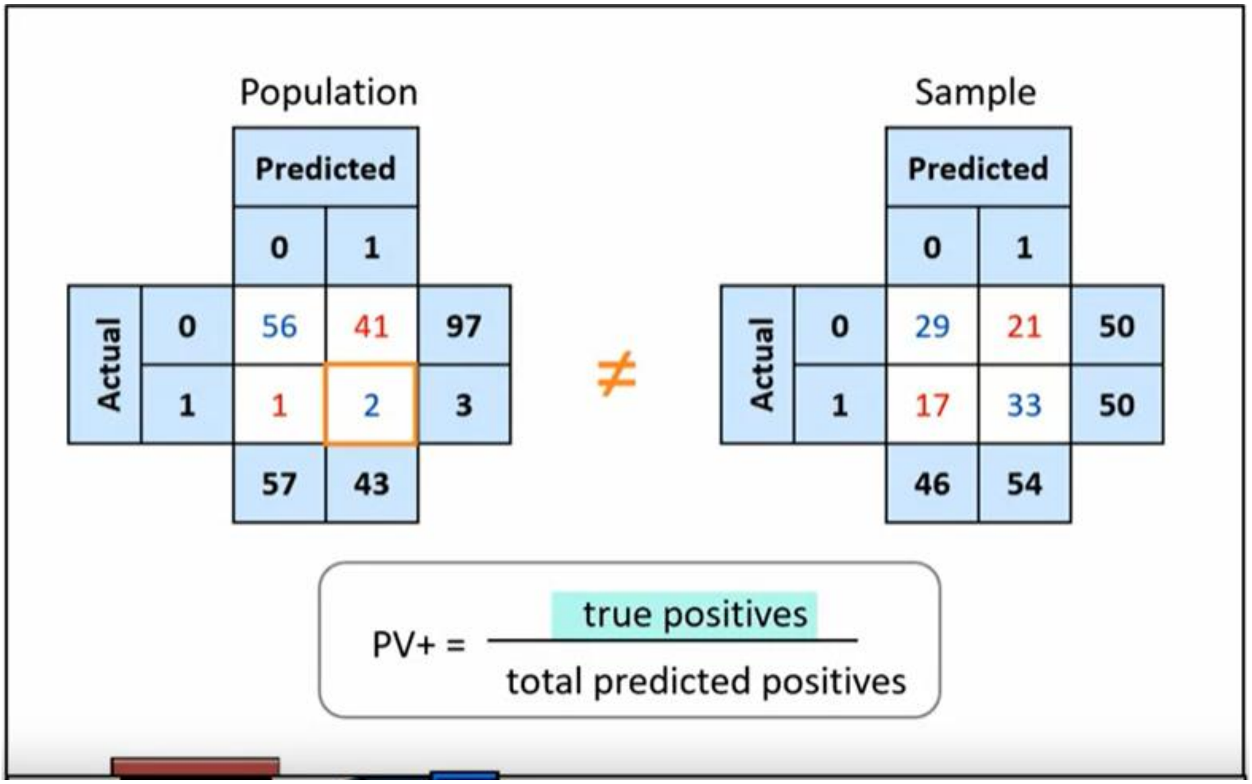


56 / 97 = 0.58 = 29 / 50 (both population and sample have the same specificity value)

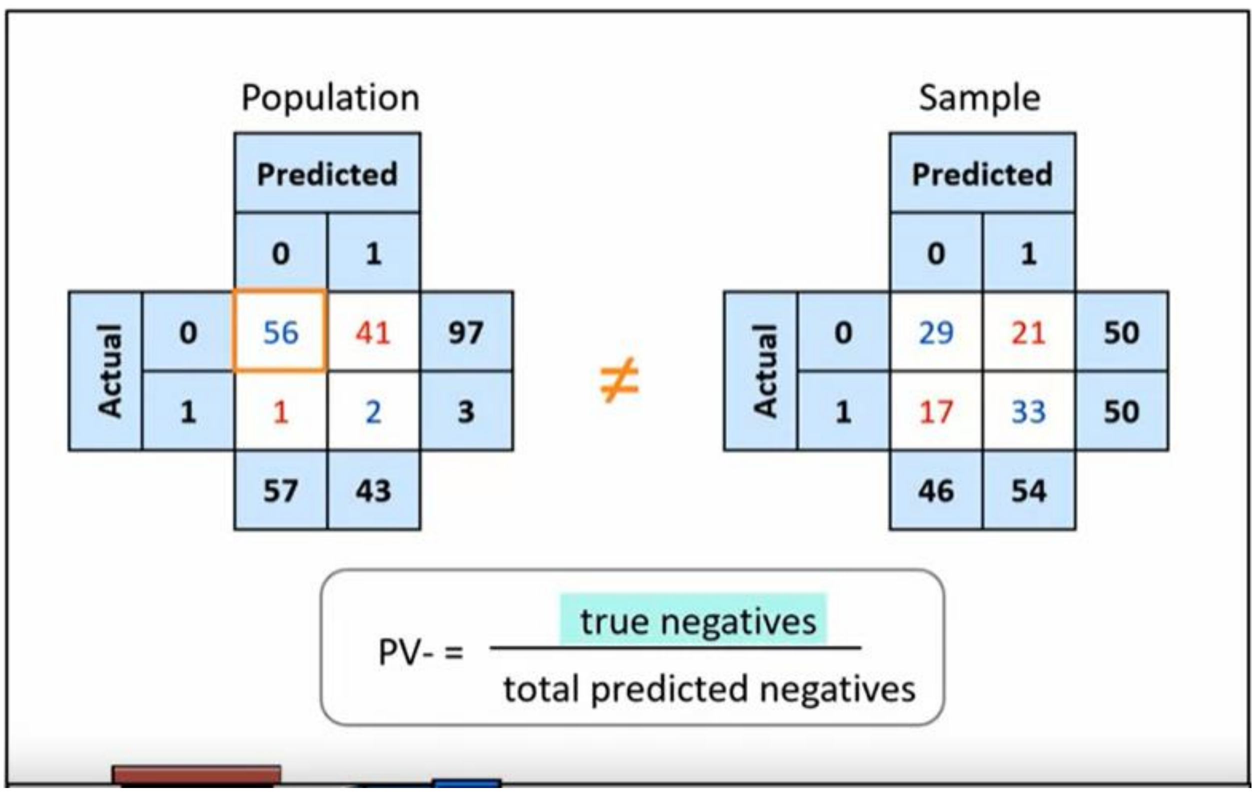

Population
Predicted
0
1
Actual
0
56
41
97
1
1
2
3
57
43
Sample
Predicted
0
1
Actual
0
29
21
50
1
17
33
50
46
54
0.58
specificity = true negatives / total actual negatives


Oversampling does not affect sensitivity or specificity.


The ROC curve is not affected by oversampling.

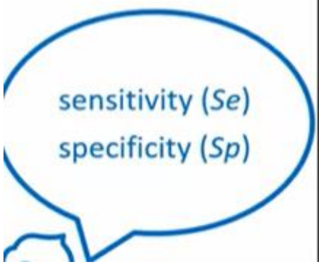


2 / 43 is not equal to 33 / 54

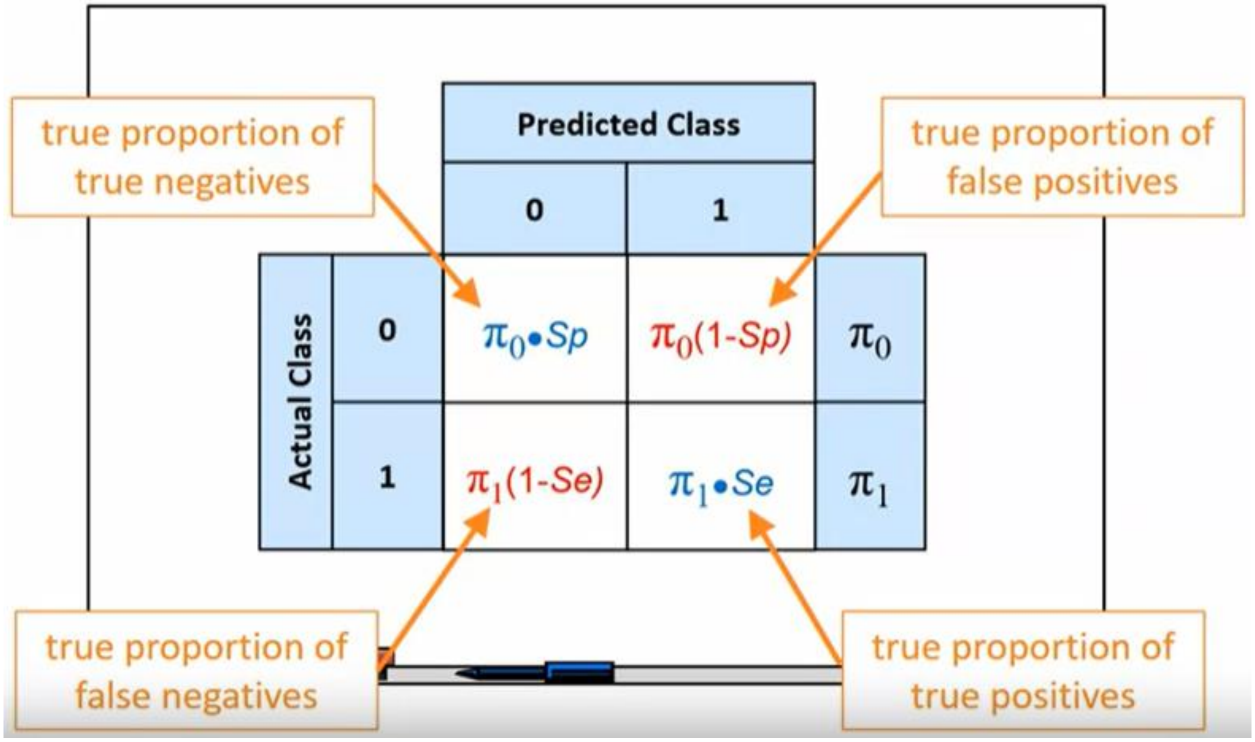


56 / 57 is not equal to 29 / 46

Adjusting a Confusion Matrix for Oversampling



		Predicted Class		
		0	1	
Actual Class	0	$\pi_0 \bullet Sp$	$\pi_0(1-Sp)$	π_0
	1	$\pi_1(1-Se)$	$\pi_1 \bullet Se$	π_1



Demo Measuring Model Performance based on Commonly-Used Metrics

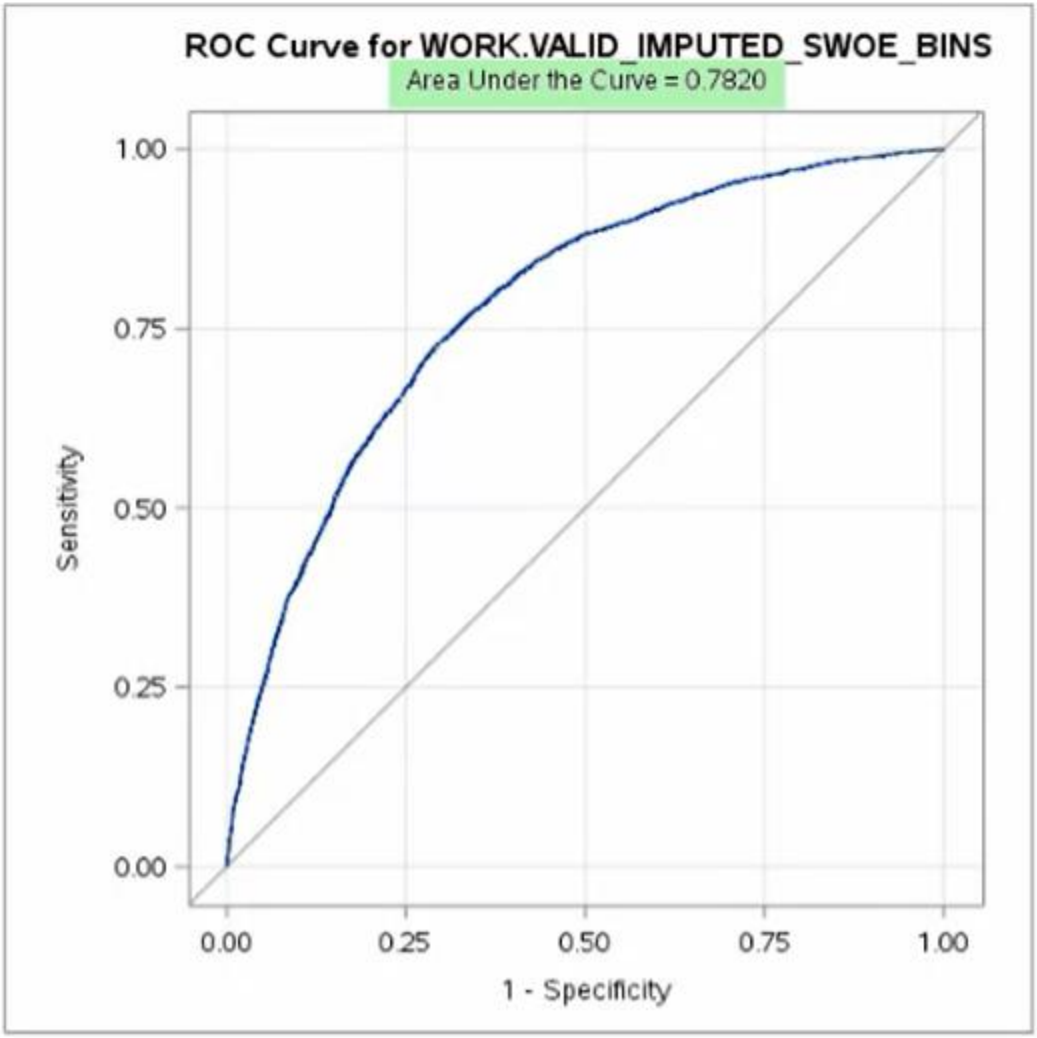
* Score the validation data set using PROC LOGISTIC.

* Adjust the confusion matrix for oversampling using a DATA step.

* Generate a lift chart using PROC SGPLOT.

pmlr04d02.sas

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



Fit Statistics for SCORE Data

t	Total Frequency	Log Likelihood	Error Rate	AIC	AICC	BIC	SC	R-Square	Max-Rescaled R-Square	AUC	Brier Score
/ALID_IMPUTED_SWOE_BINS	10752	-12661.1	0.3406	25394.13	25394.38	25656.31	25656.31	0.316954	0.338919	0.78197	0.308581

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

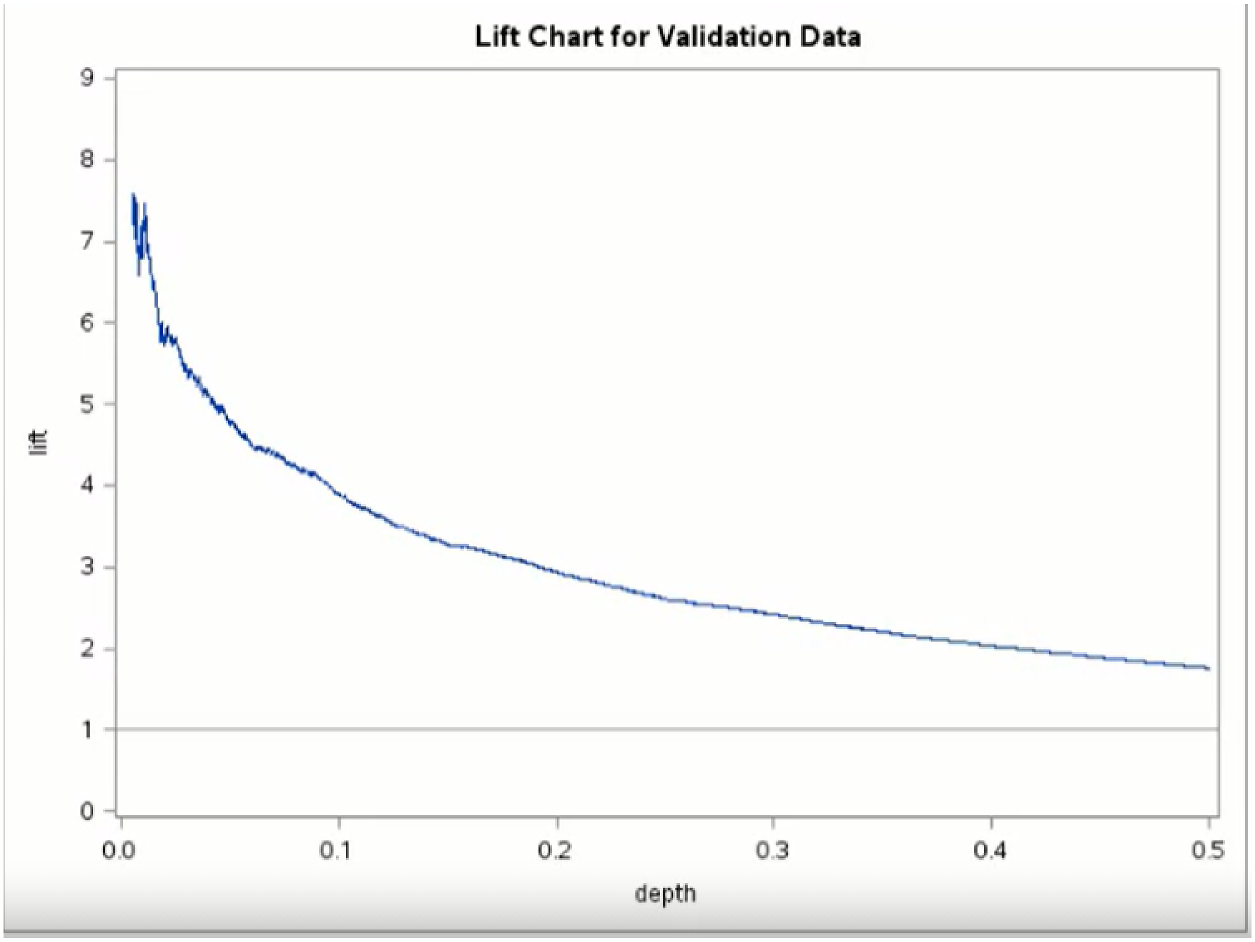
Statistics in the ROC Data Set

Obs	_PROB_	_SENSIT_	_1MSPEC_
1	1.00000	.000537057	.000000000
2	1.00000	.000805585	.000000000
3	1.00000	.001074114	.000000000
4	0.99999	.001342642	.000000000
5	0.99997	.001611171	.000000000
6	0.99948	.001879699	.000000000
7	0.99896	.002148228	.000000000
8	0.99890	.002416756	.000000000
9	0.99875	.002416756	.000142288
10	0.99823	.002416756	.000284576

To see the formula used for the adjustment, see Adjusting the Posterior Probabilites in the Resources section.

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

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



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

"Predictive Modeling Using Logistic Regression" */

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

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

Before you submit the code, make any necessary modifications to

the code, if indicated in comments.

Note: Most of the code requires no modifications.

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

After performing the steps above, you are ready to proceed with the

current demonstration!

*/

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

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

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

```
        HMOWN LORES HMVAL AGE CRSCORE MOVED
        INAREA;

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

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

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


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

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

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

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

/* Verify stratification. */

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

/* Create training and validation data sets. */

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




/* ===================================================== */
/* Lesson 2, Section 1: l2d1.sas
  Demonstration: Fitting a Basic Logistic Regression Model,
  Parts 1 and 2
```

```
  [m642_1_k1, m642_1_k2; derived from pmlr02d01.sas]    */
/* ===================================================== */

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


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

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

proc logistic data=work.train noprint;
  class res (param=ref ref='S');
```

```
  model ins(event='1')= res dda ddabal dep depamt cashbk checks;
  score data = pmlr.new out=work.scored1;
run;

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

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

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

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

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

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

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

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

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

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


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

/* Specify the prior probability to correct for oversampling.  */
%global pi1;
```

```
%let pi1=.02;

/* Correct predicted probabilities */

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

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

title1 "Mean of Adjusted Predicted Probabilities from Scored Data Set";
proc means data=work.scored4 mean nolabels;
  var p_1;
run;
title1 ;

/* Correct probabilities in the Score Code */

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

```
  code file="&PMLRfolder/pmlr_score_adj.txt";
run;

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

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

data work.scored5;
  set work.new;
  %include "&PMLRfolder/pmlr_score_adj.txt";
  eta=log(p_ins1/p_ins0) - off;
  prob=1/(1+exp(-eta));
run;

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

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

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

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

```
run;

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

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


/* ===================================================== */
/* Lesson 3, Section 2: l3d2a.sas
  Demonstration: Collapsing the Levels of a Nominal Input,
  Part 1
  [m643_2_g1; derived from pmlr03d02.sas]          */
/* ===================================================== */

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

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

```
run;

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

ods output clusterhistory=work.cluster;

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



/* ===================================================== */
/* Lesson 3, Section 2: l3d2b.sas
   Demonstration: Collapsing the Levels of a Nominal Input,
   Part 2
   [m643_2_g2; derived from pmlr03d02.sas]            */
/* ===================================================== */

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

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

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

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

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

title1 "Plot of the Log of the P-Value by Number of Clusters";
proc sgplot data=work.cutoff;
   scatter y=logpvalue x=numberofclusters
         / markerattrs=(color=blue symbol=circlefilled);
   xaxis label="Number of Clusters";
   yaxis label="Log of P-Value" min=-120 max=-85;
run;
title1 ;

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

proc sql;
   select NumberOfClusters into :ncl
   from work.cutoff
```

```
  having logpvalue=min(logpvalue);
quit;

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

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

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

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

filename brclus "&PMLRfolder/branch_clus.sas";

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

```
  end;
run;

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


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

/* Rho1 is the proportion of events in the training data set. */
%global rho1;
proc sql noprint;
  select mean(ins) into :rho1
  from work.train_imputed;
run;

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

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

```
run;

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


filename brswoe "&PMLRfolder/swoe_branch.sas";

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

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


/* ===================================================== */
/* Lesson 3, Section 3: l3d4.sas
   Demonstration: Reducing Redundancy by Clustering Variables
```

```
  [m643_3_i; derived from pmlr03d04.sas]            */
/* ===================================================== */

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

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

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

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

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

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

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

/* Choose a representative from each cluster.  */
%global reduced;
%let reduced=branch_swoe MIINCOME Dep CCBal MM Income ILS POS NSF CD
        DDA LOC Age Inv InArea AcctAge Moved CRScore MICRScor
        IRABal MIAcctAg SavBal CashBk DDABal SDB InvBal CCPurc
        ATMAmt Sav CC Phone HMOwn DepAmt IRA MTG ATM LORes;


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

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

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

ods select all;

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

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

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

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

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

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


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

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

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

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

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

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



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

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

%global var;
%let var=DDABal;

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

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

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

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

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

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

```
run;

/* Calculate the empirical logit */

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

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

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

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

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

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

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

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

/* Create new logit plots */
%global var;
```

```
%let var=DDABal;

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

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

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

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

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


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

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

/* Rank the observations. */

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

/* Save the endpoints of each bin */
```

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

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

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

filename rank "&PMLRfolder/rank.sas";

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

/* Use the code. */
```

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

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

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



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

title1 "P-Value for Entry and Retention";

%global sl;
```

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

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


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

title1 "Backward Selection for Variable Annuity Data Set";
proc logistic data=work.train_imputed_swoe_bins;
  class res (param=ref ref='S');
  model ins(event='1')= &screened res SavBal*B_DDABal MM*B_DDABal
          branch_swoe*ATMAmt B_DDABal*Sav SavBal*SDB
          SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt
```

```
            SavBal*IRA SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM
            IRA*B_DDABal CD*MM MM*IRABal CD*Sav B_DDABal*CashBk Sav*CC
            / clodds=pl
        selection=backward slstay=&sl hier=single fast;
run;


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

title1 "Candidate Model for Variable Annuity Data Set";
ods select OddsRatiosPL;
proc logistic data=work.train_imputed_swoe_bins;
   model ins(event='1')= SavBal Dep DDA CD Sav CC ATM MM branch_swoe IRA B_DDABal
                  ATMAmt ILS NSF SDB
                  DepAmt Inv SavBal*B_DDABal MM*B_DDABal
                  branch_swoe*ATMAmt Sav*B_DDABal
                  SavBal*SDB SavBal*DDA AtmAmt*DepAmt B_DDABAL*ATMAmt SavBal*IRA
                  SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM IRA*B_DDABal
                  CD*MM CD*Sav Sav*CC / clodds=pl;
   oddsratio B_DDABAL / at(savbal=0, 1211, 52299) cl=pl;
run;


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

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

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

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

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

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

```
  call symput("&var2._p25",&var2._p25);
  call symput("&var2._p50",&var2._p50);
  call symput("&var2._p75",&var2._p75);
  call symput("&var2._p95",&var2._p95);
run;

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

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

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

%mend interact;
```

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


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

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

/* Run best subsets */
title1 "Models Selected by Best Subsets Selection";
proc logistic data=work.train_imputed_swoe_bins;
   model ins(event='1')=&screened resr resu SavBal*B_DDABal MM*B_DDABal
          branch_swoe*ATMAmt B_DDABal*Sav SavBal*SDB
          SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt
```

```
            SavBal*IRA SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM
            IRA*B_DDABal CD*MM MM*IRABal CD*Sav B_DDABal*CashBk Sav*CC
            / selection=score best=1;

run;



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

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

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

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

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

/* The names and number of variables are transferred to macro
```

```
  variables using PROC SQL. */

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

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

%let lastindx=&SQLOBS;

%do model_indx=1 %to &lastindx;

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

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

proc datasets
   library=work
   nodetails
   nolist;
   delete scored;
```

```
run;
quit;

%end;

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

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

%mend fitstat;

%fitstat(data=train_imputed_swoe_bins,target=ins,event=1,inputs=&screened resr resu
        SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt B_DDABal*Sav SavBal*SDB
        SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt SavBal*IRA
        SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM IRA*B_DDABal CD*MM
        MM*IRABal CD*Sav B_DDABal*CashBk Sav*CC,best=1,priorevent=0.02);

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

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

```
run;

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




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

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

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

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

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

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

title1 "Statistics in the ROC Data Set";
```

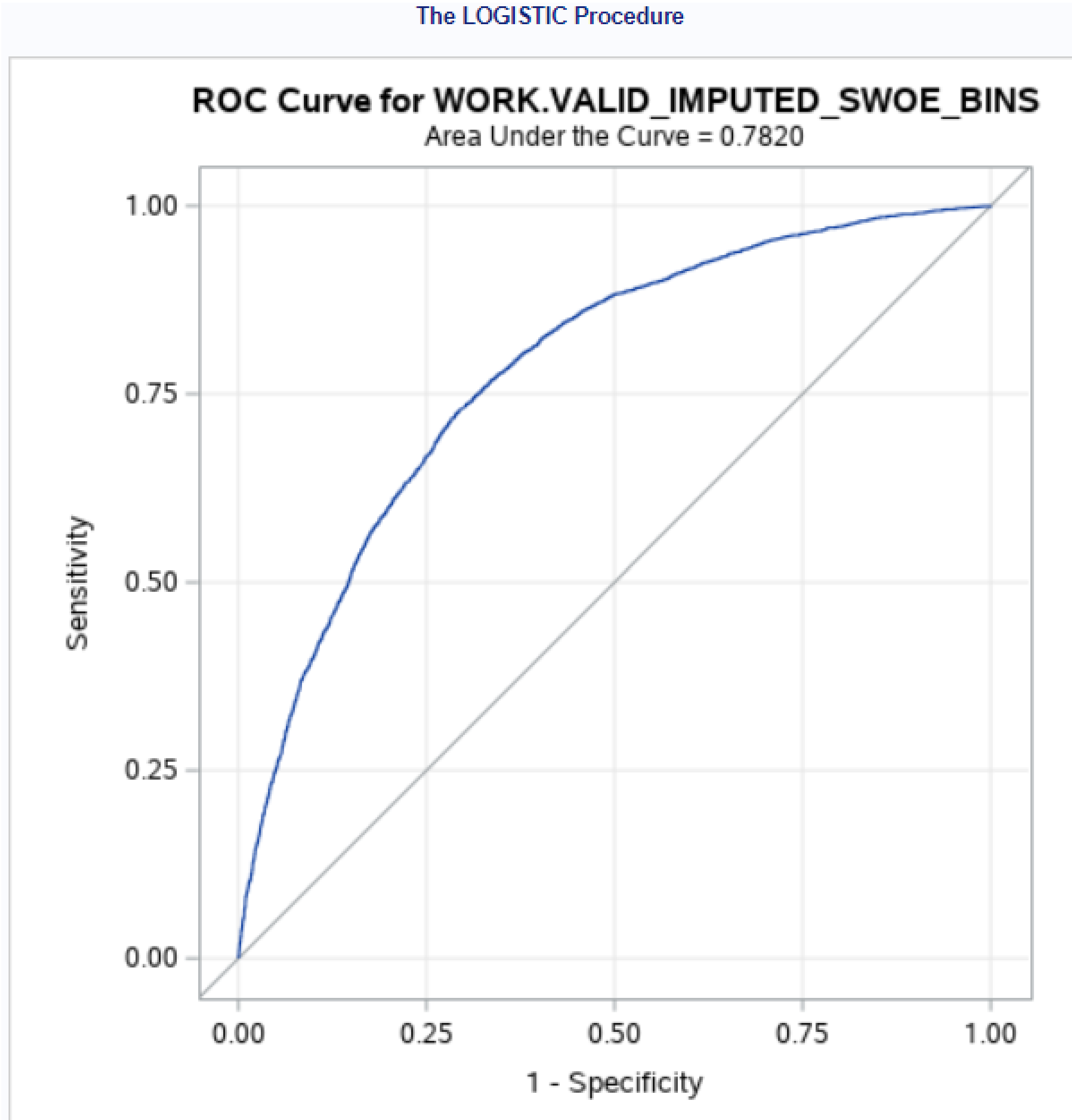
```
proc print data=work.roc(obs=10);
  var _prob_ _sensit_ _1mspec_;
run;

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

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

run; quit;

title1 ;

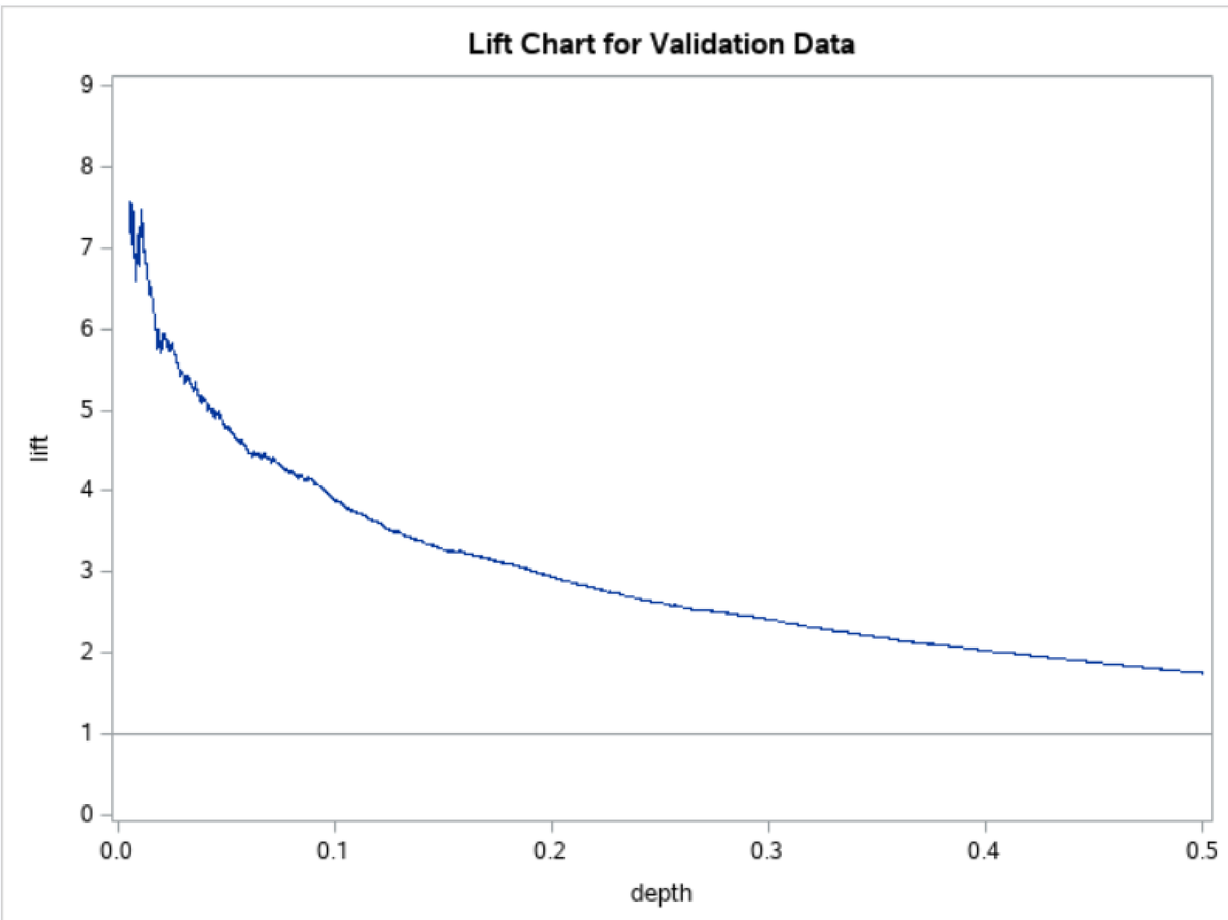


Fit Statistics for SCORE Data

Data Set	Total Frequency	Log Likelihood	Error Rate	AIC	AICC	BIC	SC	R-Square	Max-Rescaled R-Square	AUC	Brier Score
WORK.VALID_IMPUTED_SWOE_BINS	10752	-12661.1	0.3406	25394.13	25394.38	25656.31	25656.31	0.316954	0.338919	0.78197	0.308581

Statistics in the ROC Data Set

Obs	_PROB_	_SENSIT_	_1MSPEC_
1	1.00000	.000537057	.000000000
2	1.00000	.000805585	.000000000
3	1.00000	.001074114	.000000000
4	0.99999	.001342642	.000000000
5	0.99997	.001611171	.000000000
6	0.99948	.001879699	.000000000
7	0.99896	.002148228	.000000000
8	0.99890	.002416756	.000000000
9	0.99875	.002416756	.000142288
10	0.99823	.002416756	.000284576



```
/* Run this code before doing practice l4p1 */

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

```
/* ====================================================== */
/* Lesson 3, Practice 6
   Practice: Creating Empirical Logit Plots            */
/* ====================================================== */

%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;

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;

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;

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;


/* =========================================================== */
/* Lesson 3, Practice 7
  Practice: Using Forward Selection to Detect Interactions */
/* =========================================================== */

%global ex_screened;

%let ex_screened=
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 nurbs;

%global sl;

title1 "P-Value for Entry and Retention";
proc sql;
  select 1-probchi(log(sum(target_b ge 0)),1) into :sl
  from pmlr.pva_train_imputed_swoe;
quit;
title1;

title1 "Interaction Detection using Forward Selection";
proc logistic data=pmlr.pva_train_imputed_swoe namelen=50;
  model target_b(event='1')= &ex_screened
      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|nurbs @2 / include=26 clodds=pl
     selection=forward slentry=&sl;
run;
title1;
```

```
/* ===================================================== */
/* Lesson 3, Practice 8
  Practice: Using Backward Elimination to Subset the
  Variables                               */
/* ===================================================== */

title1 "Backward Selection for Variable Annuity Data Set";
proc logistic data=pmlr.pva_train_imputed_swoe namelen=50;
  model target_b(event='1')= &ex_screened
      LAST_GIFT_AMT*LIFETIME_AVG_GIFT_AMT
      LIFETIME_AVG_GIFT_AMT*RECENT_STAR_STATUS
      LIFETIME_GIFT_COUNT*MONTHS_SINCE_LAST_GIFT
      / clodds=pl selection=backward slstay=&sl hier=single
      fast;
run;
title1;



/* ===================================================== */
/* Lesson 3, Practice 9
  Practice: Using Fit Statistics to Select a Model      */
/* ===================================================== */

%global ex_selected;

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

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

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

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

%let lastindx=&SQLOBS;

%do model_indx=1 %to &lastindx;

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

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

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

%end;

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

%mend fitstat;

%fitstat(data=pmlr.pva_train_imputed_swoe,target=target_b,event=1,
      inputs=&ex_screened LAST_GIFT_AMT*LIFETIME_AVG_GIFT_AMT
      LIFETIME_AVG_GIFT_AMT*RECENT_STAR_STATUS
      LIFETIME_GIFT_COUNT*MONTHS_SINCE_LAST_GIFT,best=1,
      priorevent=0.05);

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

title1 "Fit Statistics from Models selected from Best-Subsets";
ods select all;
```

```
proc print data=work.modelfit;
  var model auc aic bic misclass adjrsquare brierscore;
run;
title1;

proc sql;
  select VariablesInModel into :ex_selected
  from work.score
  where numberofvariables=9;
quit;

/* Solution for l4p1 */

/* step 2 */

title1 "Variables with Missing Values on the Validation Data Set";
proc means data=pmlr.pva_valid nmiss;
  var LIFETIME_GIFT_COUNT LAST_GIFT_AMT MEDIAN_HOME_VALUE
     FREQUENCY_STATUS_97NK PEP_STAR INCOME_GROUP
     LIFETIME_AVG_GIFT_AMT MONTHS_SINCE_LAST_GIFT;
run;


/* step 3 */

proc univariate data=pmlr.pva_train_imputed_swoe noprint;
  var INCOME_GROUP;
  output out=work.medians
       pctlpts=50
```

```
        pctlpre=income_group;
run;

title1 "Medians for Variables with Missing Values";
proc print data=work.medians;
run;
title1;


/* step 4 */

data pmlr.pva_valid_imputed_swoe(drop=income_group50 i);
  if _N_=1 then set work.medians;
  set pmlr.pva_valid;
  array x(*) income_group;
  array med(*) income_group50;
    do i=1 to dim(x);
      if x(i)=. then x(i)=med(i);
    end;
  %include clswoe;
run;


/* step 5 */

title1 "Training Data Set Model";
proc logistic data= pmlr.pva_train_imputed_swoe;
  model target_b(event='1')=&ex_selected;
  score data= pmlr.pva_valid_imputed_swoe priorevent=&ex_pi1
```

```
        outroc=work.roc fitstat;
run;
title1;


/* step 6 */

data work.roc;
   set work.roc;
   cutoff=_PROB_;
   specif=1-_1MSPEC_;
   tp=&ex_pi1*_SENSIT_;
   fn=&ex_pi1*(1-_SENSIT_);
   tn=(1-&ex_pi1)*specif;
   fp=(1-&ex_pi1)*_1MSPEC_;
   depth=tp+fp;
   pospv=tp/depth;
   negpv=tn/(1-depth);
   acc=tp+tn;
   lift=pospv/&ex_pi1;
   keep cutoff tn fp fn tp
       _SENSIT_ _1MSPEC_ specif depth
       pospv negpv acc lift;
run;
title1 "Lift Chart for Validation Data";
proc sgplot data=work.roc;
   where 0.005 <= depth <= 0.50;
   series y=lift x=depth;
   refline 1.0 / axis=y;
```

```
  yaxis values=(0 to 4 by 1);
run;
quit;
title1;
```

Variables with Missing Values on the Validation Data Set

The MEANS Procedure

Variable	N Miss
LIFETIME_GIFT_COUNT	0
LAST_GIFT_AMT	0
MEDIAN_HOME_VALUE	0
FREQUENCY_STATUS_97NK	0
PEP_STAR	0
INCOME_GROUP	2229
LIFETIME_AVG_GIFT_AMT	0
MONTHS_SINCE_LAST_GIFT	0

Medians for Variables with Missing Values

Obs	income_group50
1	4

Training Data Set Model

The LOGISTIC Procedure

Model Information	
Data Set	PMLR.PVA_TRAIN_IMPUTED_SWOE
Response Variable	TARGET_B
Number of Response Levels	2
Model	binary logit
Optimization Technique	Fisher's scoring

Number of Observations Read	9687
Number of Observations Used	9687

Response Profile		
Ordered Value	TARGET_B	Total Frequency
1	0	7265
2	1	2422

Probability modeled is TARGET_B=1.

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

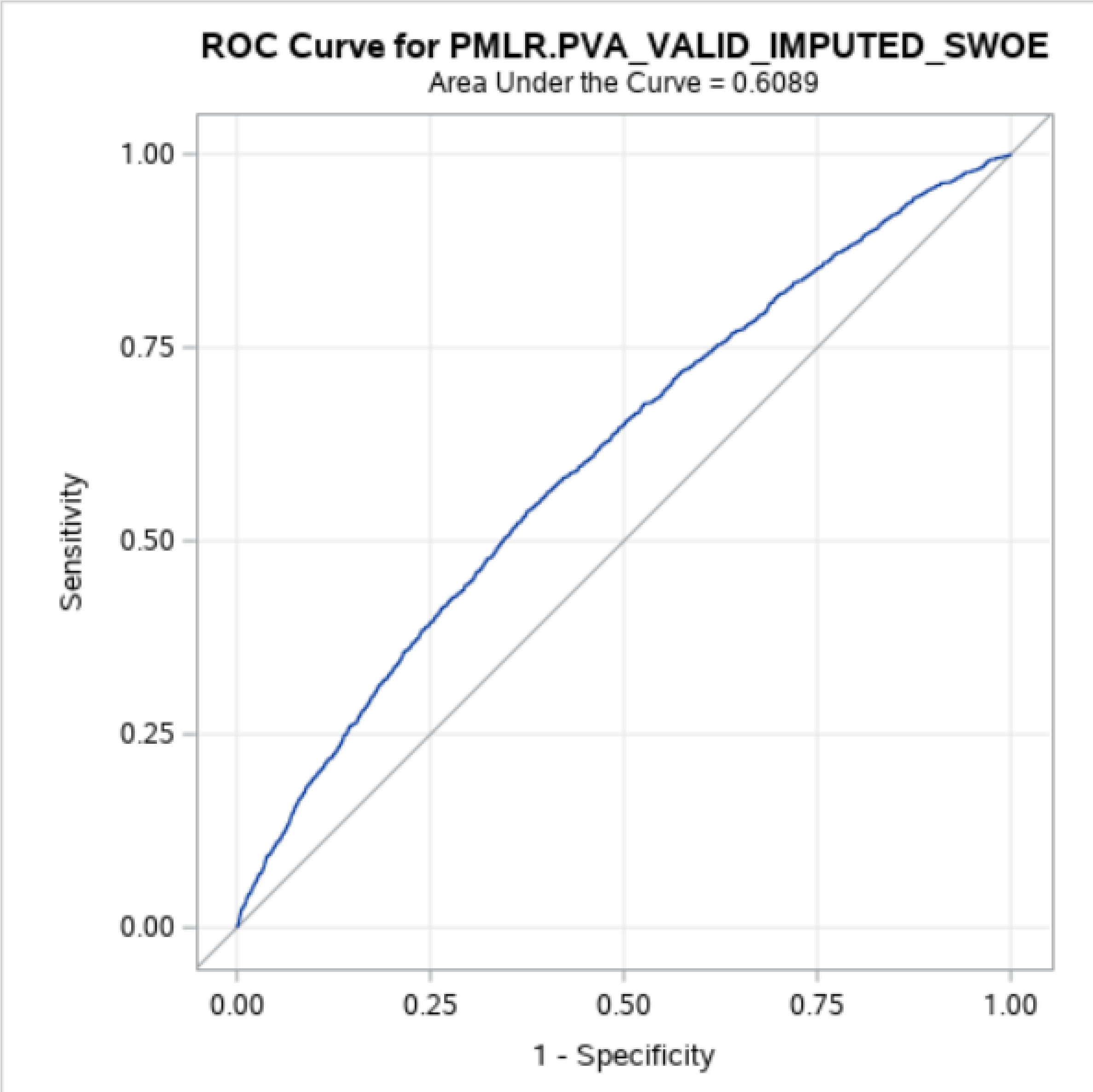
Model Fit Statistics		
Criterion	Intercept Only	Intercept and Covariates
AIC	10897.230	10514.106
SC	10904.409	10585.892
-2 Log L	10895.230	10494.106

Testing Global Null Hypothesis: BETA=0			
Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	401.1240	9	<.0001
Score	405.4144	9	<.0001
Wald	382.4438	9	<.0001

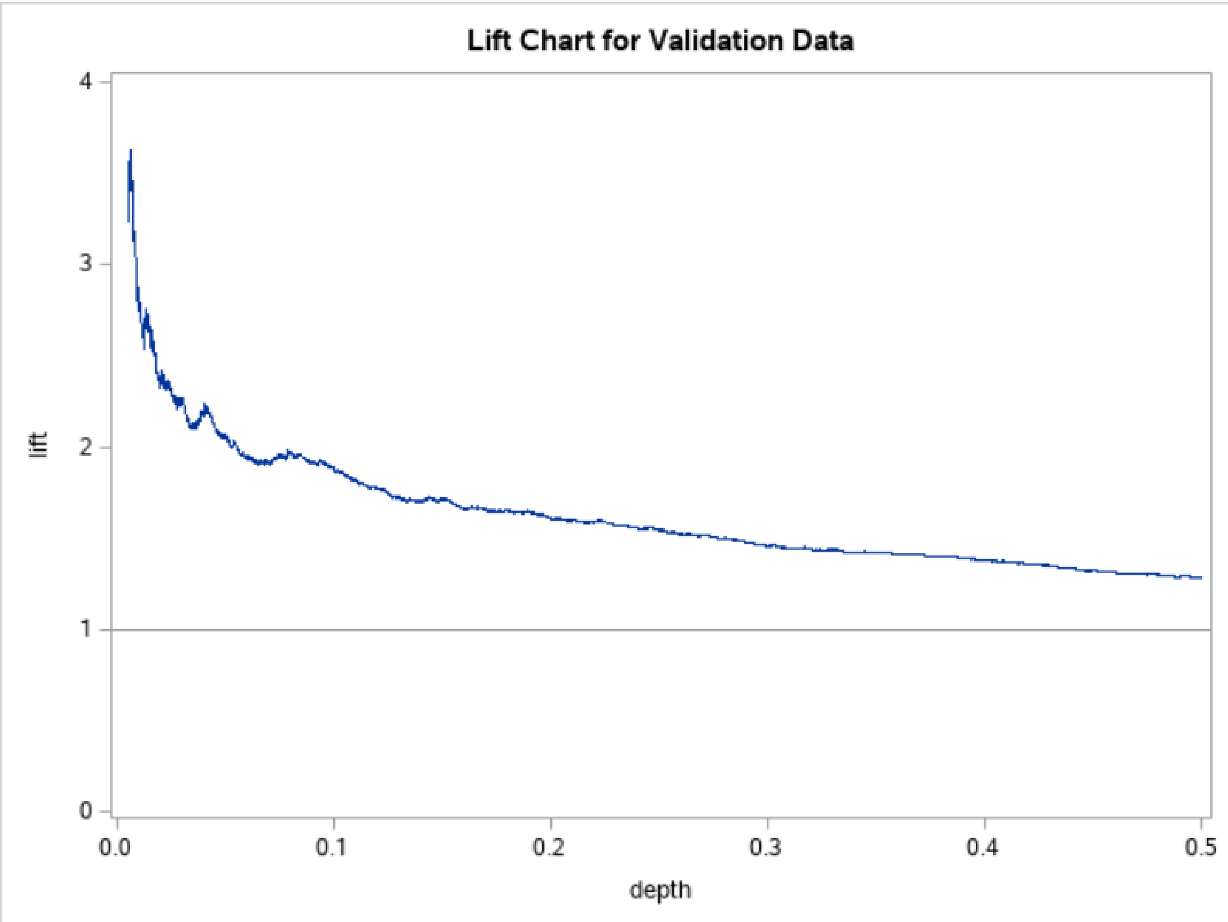
Analysis of Maximum Likelihood Estimates					
Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	1	-0.6217	0.2112	8.6698	0.0032
LIFETIME_GIFT_COUNT	1	0.0401	0.00670	35.9259	<.0001
LAST_GIFT_AMT	1	-0.0183	0.00398	21.1735	<.0001
MEDIAN_HOME_VALUE	1	0.000095	0.000026	13.4529	0.0002
FREQUENCY_STATUS_97N	1	0.1720	0.0253	46.3852	<.0001
cluster_swoe	1	0.9869	0.1493	43.6931	<.0001
PEP_STAR	1	0.3248	0.0614	27.9318	<.0001
INCOME_GROUP	1	0.0471	0.0154	9.3146	0.0023
LAST_GIFT*LIFETIME_A	1	0.000167	0.000050	11.1250	0.0009
LIFETIME_*MONTHS_SIN	1	-0.00211	0.000366	33.3864	<.0001

Odds Ratio Estimates			
Effect	Point Estimate	95% Wald Confidence Limits	
MEDIAN_HOME_VALUE	1.000	1.000	1.000
FREQUENCY_STATUS_97N	1.188	1.130	1.248
cluster_swoe	2.683	2.002	3.595
PEP_STAR	1.384	1.227	1.561
INCOME_GROUP	1.048	1.017	1.080

Association of Predicted Probabilities and Observed Responses			
Percent Concordant	63.2	Somers' D	0.263
Percent Discordant	36.8	Gamma	0.263
Percent Tied	0.0	Tau-a	0.099
Pairs	17595830	c	0.632



Fit Statistics for SCORE Data											
Data Set	Total Frequency	Log Likelihood	Error Rate	AIC	AICC	BIC	SC	R-Square	Max-Rescaled R-Square	AUC	Brier Score
PMLR.PVA_VALID_IMPUTED_SWOE	9685	-7444.5	0.2499	14908.97	14909	14980.76	14980.76	0.036643	0.046213	0.608916	0.223326



Practice: Assessing Model Performance

Question 1

For the veterans' organization project, do the following:

- prepare the validation data set to be scored by the model fitted on the training data set
- fit a logistic model on the training data set
- score the validation data set
- compute model performance statistics and generate graphs on the validation data set

Reminder: If you started a new SAS session, you must run **setup.sas** to define the **pmlr** library before you do this practice.

Step 1: Open **l4p01_runFirst.sas** from the **practices** folder and run the code. You can add to this program or open a new editor to continue the practice.

Step 2: Write a PROC MEANS step to examine which variables in **pmlr.pva_valid** (the validation data set) have missing values. Use the inputs from the model fitted on the training data set. Note:

Exclude **Cluster_Swoe**, which needs to be created, but specify the inputs involved in the interactions.

Submit the code and look at the results.

Which input variable has missing values?

INCOME_GROUP

The results show that the input variable **Income_Group** has missing values.

For the solution code open **l4p1_s.sas** from the **practices/solutions** folder and see Step 2.

Question 2

<u>Step 3</u>: Write a PROC UNIVARIATE step to create a data set with the medians from **pmlr.pva_train_imputed_swoe** (the training data set). Name the new data set **work.medians**. Use the NOPRINT option in the PROC UNIVARIATE statement. Store the medians in a variable whose name is the original variable name followed by **50**.

Add a PROC PRINT step to print the output data set.

Submit the code and look at the results.

What is the median for the variable with missing values?

4

As shown in the results, the median for **Income_Group** is 4.

For the solution code open **l4p1_s.sas** from the **practices/solutions** folder and see Step 3.

Question 3

<u>Step 4</u>: Write a DATA step that does the following:

- imputes the variables with missing values using two ARRAY statements and a DO loop with index **i**
- includes the scoring code to create the smoothed weight of evidence for **Cluster_Code**
- performs a one-to-many merge to create the final version of the **pmlr.pva_valid_imputed_swoe** data set
- drops the variables **Income_Group50** and **i**

Submit the code and look at the log.

How many observations are in **pmlr.pva_valid_imputed_swoe**?

9685

The log indicates that the **pmlr.pva_valid_imputed_swoe** data set has 9685 observations.

For the solution code open **l4p1_s.sas** from the **practices/solutions** folder and see Step 4.

Question 4

Step 5: Write a PROC LOGISTIC step that does the following:

- fits a logistic regression model on pmlr.pva_train_imputed_swoe with **Target_B** as the target variable and the **ex_selected** macro variable (created in the previous practice) specifying the input variables
- uses the EVENT= option to model the probability that **Target_B**=1
- uses the SCORE statement to score **pmlr.pva_valid_imputed_swoe** with an adjustment for oversampling using the PRIOREVENT= option
- uses the OUTROC= option to create a data set named **work.roc** with many of the statistics that are necessary for model assessment and for creating a lift chart for the validation data set
- uses the FITSTAT option to generate model fit statistics

Submit the code and look at the results.

What is the *c* statistic for the validation data set?

0.6089

In the results, the plot of the ROC curve for the validation data set shows that the *c* statistic for the validation data set is 0.6089.

For the solution code open **l4p1_s.sas** from the **practices/solutions** folder and see Step 5.

Question 5

Step 6: Using the data set created by the OUTROC= option, write a DATA step to compute the proportion of true positives, the proportion of false negatives, the proportion of true negatives, the proportion of false positives, the positive predicted value, the negative predicted value, the accuracy, the proportion allocated to class 1 (depth), and the lift.

Add a PROC SGPLOT step that creates a lift chart. Add a reference line at a lift of 1, and restrict the focus to the region where depth is greater than 0.5% and less than 50%. Restrict the Y axis from 0 to 4 by 1.

Submit the code and look at the results.

What is the lift at a depth of 10%?

approximately 1.9

As shown in the results, the lift at a depth of 10% is approximately 1.9.

For the solution code open **l4p1_s.sas** from the **practices/solutions** folder and see Step 6.