Paper 143-2008

Evaluating Predictive Models: Computing and Interpreting the c Statistic

Sigurd W. Hermansen, Westat, Rockville, MD, USA

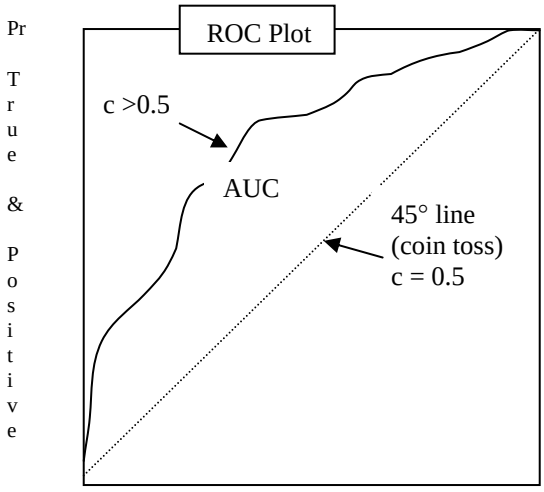
ABSTRACT

Automation of predictive model selection has become the alchemy of today's Business Intelligence (BI), with BI practitioners hoping to transform jargon and acronyms into gold. While statistical modeling experts disparage 'black-box' methods, business analysts tend to ignore theory and traditional model diagnostics, and rely on 'lift' or 'gain' charts; that is, evidence of improved accuracy of predictions of outcomes for groups, often represented graphically by Area Under Curve (AUC) of signal detection Receiver Operating Characteristic (ROC) plots. Even where interest in competitive advantage overrides concern for meaningful model parameter estimates, a concordance c statistic offers a simpler and more intuitive measure of accuracy of model predictions. SAS/Stat PROCs LOGISTIC and SURVEYLOGISTIC generate a c statistic. GENMOD and others do not, but a straightforward SAS/Base PROC SQL program computes c efficiently for at least a few thousand predictors and outcomes. A more robust hashing algorithm computes the c statistic efficiently for very large numbers of predictions and outcomes. A weighted or grouped c statistic, combined with other standard model diagnostics, effectively summarizes predictive model lift and gain. SAS Macros serve as templates for interesting c statistic algorithms, and examples from biomedical and administrative studies help analysts understand how to compute and interpret a important measure of predictive model accuracy.

INTRODUCTION

"The quality of model predictions calls for at least two numbers: one number to indicate accuracy of prediction …., and another number to reflect its generalizability …" (Bruer, 2006)

For developers and users of predictive models, assessing and predicting the quality of a model can be as important as predictions that the model generates. Business intelligence support systems, notably SAS/EM (Brocklebank and Brown, 2003), offer analysts an initially bewildering array of tools for evaluating predictive models. Trade-offs between sensitivity (probability of true prediction given positive outcome) and specificity (probability of false prediction given negative outcome) have a prominent role. Plots of the area under curve (AUC) in a Receiver Operating Characteristic (ROC) plot show the trade-offs in a graphic fashion.



The 45° line represents pairs of sensitivity and specificity values that essentially offset one another. False predictions (signals) cancel out information that a receiver obtains from true predictions. At points above the 45° line, the probability of true predictions exceeds that of false predictions, and the greater the distance of a point from the 45° line, the more information the receiver obtains.

Any point above the 45° line represents a better than random guess, but, unless we know the benefits and costs of true and false predictions, no one point in one ROC plot represents a better choice than another. Nonetheless, if every point in the ROC plot for model A lies above every point in the ROC plot for model B, model A dominates model B. ROC curves for alternative models in the same plot make obvious at which levels of sensitivity and specificity, if any, one model dominates others. These decision rules, unfortunately, pertain only to predictions computed using only one and the same sample.

Model *gain* and *lift* depict model accuracy for segments of a sample and extrapolate to a population for planning purposes. After ranking entities in descending order of predicted outcome, to compute gain and lift we group entities in deciles and take a model's predictions as expected responses per decile.

(1)	(2)	(2)/(1)	Δ(2)/Δ(1)
decile	cum% positive response	gain	lift
1	25	2.50	2.50
2	45	2.25	2.00
3	60	2.00	1.50
4	70	1.75	1.00
5	78	1.56	0.80
6	85	1.42	0.70
7	91	1.30	0.60
8	95	1.19	0.40
9	98	1.09	0.30
10	100	1.00	0.20

A lift value of one (decile 4) indicates that the expected incremental increase in positive responses in that decile equals the average expected positive response in the population. The gain value of 1.30 for decile 7 predicts that 70% of the population selected by the model would be expected to provide a 30% greater number of positive responses (70% X 1.3 = 91%) than 70% of the population selected at random. Though valuable aids to decisions in fields as diverse as medicine, insurance, marketing, and finance (especially when augmented with dollar benefits of positive and costs of negative responses and of prediction errors), many practicioners of predictive modeling are looking for a simpler and more intuitive measure of model accuracy.

Forced to choose a single summary measure of model accuracy, statistical modeling experts would likely opt for the concordance c statistic as a good summary measure of model accuracy (e.g., Harrell, 2001). The c statistic has several virtues. Its validity does not depend on properties of a parametric distribution. It has the range of a probability measure and much the same feel as well. It relates closely to traditional non-parametric statistical tests such as the Mann-Whitney. Best of all, in this crowd at least, a relatively straightforward SAS Proc SQL program computes it quickly for relatively large numbers of predictions and outcomes. As a bonus, the program proves to be easy to explain, understand, and adapt to different situations. We expand on that theme next.

THE CONCORDANCE STATISTIC

How should one measure the accuracy of a predictive model? Say we have no concern about computation and are looking for the best measure possible.

We begin with a situation featuring two disjoint sets of entities: one of positive outcomes (e.g., positive test result, survey or marketing campaign respondents, voters for a candidate, insurance claimants, fraud cases, etc.) and another of negative outcomes. In a predictive modeling situation, outcomes will occur in the future so we do not know which of the entities will be observed in which set. We do observe the identities of the entities and some of their attributes. We also have prior outcomes for a sample of entities from the two sets. Using the prior outcomes in the sample and entity attributes as predictors, we can develop predictive models that generate a probability or likelihood of a positive outcome for each entity in the sample of entities with prior outcomes. Then, prior to deciding what action to take, comes model evaluation time.

For each prior outcome we have a probability, likelihood ratio, or binary prediction that in some sense estimates the chance of a positive outcome. Each possible pairing of prior positive and negative outcomes and predictions yields a pair of predictions: (pr_1,pr_{-1}). We would expect a highly accurate predictive model to assign greater probabilities or likelihoods to positive outcomes than to negative outcomes. If we count the number of pairs of predictions in which that is true, the result should be close to the total number of pairs of outcomes. The possibility of equal values of predictions adds a bit more complexity to the algorithm:

Pr_1	Pr_{-1}	add to correct	add to N
0.8	0.2	+1	+1
0.6	0.7	+0	+1
0.5	0.5	+0.5	+1
0.3	0.1	+1	+1
0.9	0.7	+1	+1

c = correct / N = 3.5/5 = 0.7

Through the fifth pair of predictions, the c statistic equals 0.7. Ties add 0.5 to the counter ‘correct’.

A Data step creates an artificial dataset of three sets of pairs of predictions that have values of either 0 or 1 (though predictions with any numeric values will work just as well). Each outcome relates to three predictions: pred95 (95% correct), predNI (non-informative 50% correct), and predCW (completely wrong):

```
data predictions;
  do i= 1 to 2000;
     if i>=1001 then outcome=1;
                else outcome=0;
     if ranuni(23457)<=0.90 then pred95=outcome;
                            else pred95=ceil(ranuni(23571));
     if pred95=1 then predCW=0;
                 else predCW=1;
     predNI=round(ranuni(345673),1.);
     output;
  end;
run;
```

A 'brute force' method in a SAS Macro executes a SAS Proc SQL program from inside out; it

1. computes a Cartesian product of positive and negative outcomes and their corresponding predictions;
2. assigns values 1, 0.5, or 0 to an accumulator variable, cVal, depending on whether the prediction of the positive outcome exceeds, equals, or fall under the prediction of the negative outcome;
3. computes c by adding 1 to c when cval=1, 0.5 to c when cval=0.5, and 0 when cval=0.

```
*/ %Macro cStatistic(__pred);
proc sql;
select (sum(cVal))/count as c
 from (select case when t1.LL>t2.LL then 1
                   when t1.LL=t2.LL then 0.5
                                    else 0
               end as cVal,count(*) as count
       from
            (select &__pred as LL
             from predictions
             where outcome=1) as t1,
            (select &__pred as LL
             from predictions
             where outcome=0) as t2
       )
;
quit;
%Mend cStatistic;
```

Executing the cStatistic Macro with each of the series of predictions as the argument,

```
%cStatistic(pred95)
%cStatistic(predCW)
%cStatistic(predNI)
```

yields a c statistic for the models that supposedly generated the predictions from samples of observations:

```
pred95        c = 0.954
predNI        c = 0.4945
predCW        c = 0.046
```

This exercise illustrates the range of the c statistic and demonstrates that a really bad model (predCW) has a c statistic value less than 0.5, a very good model (pred95) has a c value that approaches 1, and that if a model has a c value not well above 0.5, one may substitute flipping a coin to predict positive and negative outcomes with little loss of information. The difference between c and 0.5 measures the AUC of an ROC, and a model A that has a higher c value than a model B for a given sample predicts better.

WEIGHTING AND GROUPING

The close correspondence of the c statistic to a probability makes it practical to multiply it by a different weight for each outcome class, and to compute c statistics by groups of entities and compare their values. This can be done immediately prior to executing the %cStatistic() Macro. A c statistic can be "weighted" to reflect, for example, a substantially higher cost of a failure to predict correctly a positive outcome (diagnostic test for HIV or bankruptcy) than benefit of successfully predicting a negative outcome. Weighting alters results of comparisons of predictions of outcomes, penalizing a model that generates small differences in probabilities or likelihood ratios for positive and negative outcomes in an observed sample; weighting does not change the range of the c statistic or distort its interpretation in unexpected ways. Weighting to adjust for stratified sampling works similarly to make the c statistic more representative of a prediction of a population.

Computing the c statistic for groups requires only a small change in the %cStatistic() Macro. Say we have predictions of votes for a candidate (based on data derived from an earlier post-election poll) and

demographic attributes of a sample of voters. To compute a separate c statistic for male and female groups, for example, we add a gender variable to outcome and prediction, change the SELECT clause to

```
Select gender, (sum(cVal))/count as c
```

and add a GROUP BY Clause,

```
Group by gender
```

Provided that sample sizes remain sufficiently large for each group and outcome, grouping by different sets of attributes supports meaningful comparisons of prediction accuracy across groups.

SCALING UP C STATISTIC COMPUTATION

Even when running on a PC of modest capabilities, %cStatistic() Macro computation of the c statistic for sample sizes of up to 2,000 takes only a few seconds per c statistic computed; unfortunately, scaling up the number of predictions by a factor of five increases execution time geometrically. For large numbers of predictions, a workaround takes advantage of grouping of predictions, weighting of outcome-prediction observations, and SAS System Version 9.13 hash objects.

Harding (1984) sets a high standard for efficient computation of statistics that embed the c statistic. For very large numbers of predictions, a less exact method of computation may suffice. As numbers of predictions increase to levels where rounded values of sufficient precision begin to repeat frequently. Then replacing groups of rows with the same information, but in the form of rounded values with summary counts of classes, holds the Cartesian product calculated in %cStatistic() to manageable dimensions. A hash object method in a SAS Data step (copied and quickly adapted from Dorfman (2007)) summarizes a few thousand class values distributed across millions of rows almost as efficiently (O(n)) as a SAS Data step scan of the rows. The hash object summary and the modified version of %cStatistic() execute on a regular PC in less than 10 seconds:

```
data preds;
  do i= 1 to 2000000;
      pred=round(ranuni(310137),.01);
         r=ranuni(301123);
         if pred>=0.7 OR r>0.975  then outcome=1;
         else if pred< 0.7 OR r<0.025  then outcome=0;
         else outcome=0;
      output;
   end;
run;

/* Copied directly (except for one change) from Dorfman (2007). */
data _null_ ;
   if 0 then set preds ;
     dcl hash hh (hashexp:16) ;
     hh.definekey ('pred','outcome') ;
     hh.definedata ('pred', 'outcome', 'n') ;
     hh.definedone () ;
     do until (eof) ;
        set preds end = eof ;
            if hh.find () ne 0 then n = 0 ;
            n ++ 1 ;
            hh.replace () ;
     end ;
     rc = hh.output (dataset: 'predsmry') ;
run ;
```

```
%Macro cStatistic(ds,__pred);
proc sql;
 select (sum(cVal))/sum(count) as c
 from (select (sum(t1.n) + sum(t2.n)) as count,
              case when t1.LL>t2.LL then 1* calculated count
                   when t1.LL=t2.LL then 0.5* calculated count
                                    else 0
              end as cVal
       from
            (select &__pred as LL,n
             from &ds
             where outcome=1) as t1,
            (select &__pred as LL,n
             from &ds
             where outcome=0) as t2
       )
;
quit;
%Mend cStatistic;
%cStatistic(predsmry,pred)
```

VALIDATION OF THE C STATISTIC THROUGH BOOTSTRAPPING

The c statistic gives us a quick and simple indicator of accuracy of a predictive model. Does another reflect generalizability of a model?

Perhaps not one number by itself, but we can assess the impact of sampling variations on predictions. The well-known bootstrap method of resampling predictions affords us a taste of variations in predictions. Ideally we would have a predictive model that we would feed on different samples and capture predictions BY sample for each observation each a sample. Here for purposes of illustration we settle for a contrived set of data:

```
%macro Creation(__dataSource=Random, __N=100);
/* In the beginning, create data ... */
data test;
   do i=1 to &__N;
      /* Random predictions. */
      %if (&__dataSource=Random) %then %do;
         outcome=floor(ranbin(12357,1000,0.1)/100);
         x1=round(ranuni(67891),1.);
         x2=round(ranuni(78911),.1);
      %end;
      %if (&__dataSource=WeakForce) %then %do;
         /* Some predictive accuracy.*/
         outcome=floor(ranbin(12357,1000,0.1)/100);
         if ranuni(23463)>0.95 then x1=1;
         else if ranuni(34571)<0.95 then x1=0;
         else x1=outcome;
         if ranuni(45677)>0.965 then x2=0;
         else if ranuni(56781)<0.965 then x2=1;
         else x2=(outcome+ 0.5)*100 - ranuni(67891)*200;
      %end;
      %if (&__dataSource=StrongForce) %then %do;
         /* Some predictive accuracy.*/
         outcome=floor(ranbin(12357,1000,0.1)/100);
         if ranuni(23463)>0.95 then x1=1;
         else if ranuni(34571)<0.05 then x1=0;
         else x1=outcome;
         if ranuni(45677)>0.965 then x2=0;
```

```
            else if ranuni(56781)<0.10 then x2=1;
            else x2=floor(((outcome+ 0.5)*100 - ranuni(67891)*200)/10);
         %end;
         output;
      end;
run;
%mend Creation;
%Creation(__dataSource=StrongForce,__N=1000)
```

The SAS Macro Creation has optional paramer values: “Random”, which should and does feature worthless predictors x1 and x2 (c=~0.5, as you might suspect); “WeakForce” with predictors that do somewhat better than chance; and, “StrongForce”, with better predictors.

David Cassell’s (Cassell, 2007) SAS PROC SURVEYSELECT method bootstraps 1,000 randomly reduced sets of predictions:

```
/* Create replicate bootstrap samples. */
sasfile test load;
proc surveyselect data=test out=outboot
seed=30459584
method=urs samprate=1 outhits
rep=1000;
run;
ods listing close;
sasfile test close;
```

Bootstrapping samples in SAS might in other hands involve a long and difficult process, but, thanks to Cassell’s deft touch, the method looks simple and runs extraordinarily quickly. It does, nontheless, generate 1 million replicates that would tax many computing systems, so a summary method reduces the 1 million observations to a fraction of that number:

```
/* Summarize replicates. */
/* Copied directly (except for one change) from Dorfman (2007). */
data _null_ ;
   if 0 then set outboot ;
     dcl hash hh (hashexp:16) ;
     hh.definekey  ('replicate','outcome','x1','x2') ;
     hh.definedata ('replicate','outcome','x1','x2', 'n') ;
     hh.definedone () ;
     do until (eof) ;
        set outboot end = eof ;
            if hh.find () ne 0 then n = 0 ;
            n ++ 1 ;
            hh.replace () ;
     end ;
     rc = hh.output (dataset: 'smryx') ;
run ;

sasfile smryx load;
ods listing close;
proc sort data=smryx out=smry;
   by replicate;
run;
sasfile smryx close;
```

Next SAS PROC GENMOD estimates a prediction of OUTCOME for each {x1,x2} pair in each replicate:

```
/* Estimate parameters of model using bootstrap replicate samples. */
sasfile smry load;
proc genmod data=smry descending;
```

```
   model outcome = x1 x2 /link=logit dist=bin;
   by replicate;
   freq n;
   output out=preds p=pred STDRESDEV=STDRESDEV u=u l=l;
run;
sasfile smry close;
```

Continuing Cassell's bootstrap resampling method, SAS PROC UNIVARIATE computes a statistic that characterizes the distribution of predictions and residuals as flatter (heavier tail(s)), close to, more peaked than a standard normal distribution. Since we are hoping for a narrow range of predictions across various samples of from a population of interest, the kurtosis of a distribution

```
proc univariate data=preds;
var pred;
by replicate;
freq n;
output out=outall kurtosis=curt n=n pctlpts=2.5, 97.5 pctlpre=ci;
run;

ods listing;
title "Kurtosis of predictions";
proc univariate data=outall;
var curt;
freq n;
output out=final pctlpts=2.5, 97.5 pctlpre=ci;
run;
title;
proc print data=final;
run;

ods listing close;
proc univariate data=preds;
var STDRESDEV;
by replicate;
output out=outall kurtosis=curt n=n pctlpts=2.5, 97.5 pctlpre=ci;
run;

Ods listing;
title "Kurtosis of standardized residual contribution to deviance";
proc univariate data=outall;
var curt;
freq n;
output out=final pctlpts=2.5, 97.5 pctlpre=ci;
run;
proc print data=final;
run;
```

Finally the prior version of the cStatistic Macro computes the c statistic:

```
title "c statistic";
sasfile preds load;
%cStatistic(preds,pred)
sasfile preds close;
```

The observed value of the c statistic in the combined 1,000 bootstrapped samples, 0.823527, indicates that "StrongForce" predictions do in fact predict outcomes reasonably well. The mean (1.024072) and median (0.972696) of prediction kurtosis suggest that the model applied to any one sample will select relatively few from the extremes of the distribution of predictions.

CONCLUSION

The c statistic, in pure form or weighted and summarized, offers a simple, intuitive, and robust indicator of the accuracy of predictions, but it does not tell the whole story. It does reduce much of ROC gain and lift

methodology to a number that one can interpret easily. Values above 0.5 suggest some predictive value, but somewhat weak predictive models may generate c statistics in the 0.75 range. A non-parametric statistic, it tends to be reliable under a wider range of conditions than other statistics. Even so, the usual caveats about knowing one's data, reviewing distributions, and considering potential sources of bias apply none the less. As we have seen, bootstrap tests of sampling variability tell us more about the reliability or generalizability of a predictive model.

Here we have emphasized computation methods and have left important statistical issues to real statisticians. For others working as applied statisticians and data analysts, the c statistic tends to have less potential for misleading anyone struggling with the problem of selecting a predictive model. We have demonstrated a relatively simple method for computing it that dispels any mystique around its meaning or power. The SAS System not only empowers us to compute a useful statistic, it makes the process transparent, easy, and highly efficient.

REFERENCES

Breur, T., "Tom's Ten Data Tips – August 2006, Data Mining Models,
<http://www.xlntconsulting.com/resources/newsletter-archive/toms-ten-data-tips-August-2006.htm>
Cassell, D., "Don't Be Loopy: Re-Sampling and Simulation the SAS® Way", Paper 183-2007, SAS Global Forum 2007.
Dorfman, P. and Snell, G., "Hashing Rehashed"
<http://www2.sas.com/proceedings/sugi27/p012-27.pdf >
Harding E F (1983) An efficient minimal-storage procedure for calculating the Mann–Whitney U, generalised U and similar distributions Appl. Statist. 33 1–6.
Harrell, Jr., F., **Regression Modeling Strategies**, Springer (2001) NY.

ACKNOWLEDGMENTS

George Fernandez and Duke Owen provided valuable suggestions and comments. The author has sole responsibility for any errors or oversights.

DISCLAIMERS

The contents of this paper are the work of the author and do not necessarily represent the opinions, recommendations, or practices of Westat.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Sigurd W. Hermansen
Westat
1650 Research Blvd.
Rockville, MD 20850
Work Phone: 301.251.4268
E-mail: hermans1@westat.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.
Other brand and product names are trademarks of their respective companies.