Paper 3568-2019

Measuring Model Stability

Daymond Ling, Professor, Seneca College

ABSTRACT

If the stability or variation of a model's performance is important, how would you measure it before deploying the model into production? This paper discusses the use of randomization test or bootstrap sampling as a post model build technique to understand the variations in a model's performance. This situation may arise if you are vetting a model built by someone else for instance. Awareness of the variance in model performance can influence deployment decision and/or manage performance expectation when deployed. During model build, this method may also be of use in choosing amongst candidate models.

1. INTRODUCTION

During model development the performance metrics of a model is calculated on a development sample, it is then calculated for validation samples which could be another sample at the same timeframe or other time shifted samples. If the performance metrics are similar, the model is deemed stable or robust. If a model has the highest validation performance amongst candidate models, it is deemed to be the Champion and may be accepted for use in production.

These decisions of model stability and performance are based on a single point estimate derived from one sample, performance variation is usually ignored. Understanding performance variability can assist in choosing alternative models, e.g., less performance variability may be preferred over insignificant performance difference. It can also temper expectation, for instance, if in-field model performance varies within expectation there is no need to raise false alarm. While models built from large datasets generally have stable metrics, models from small datasets are more susceptible to performance variation, the techniques outlined below are more valuable.

This paper will use a binary classifier to illustrate the idea, the principle is generally applicable.

2. PERFORMANCE OF A BINARY CLASSIFIER

The questions of interest are usually of two types:

1. Did performance hold between development and validation?
2. Which model performs the best?

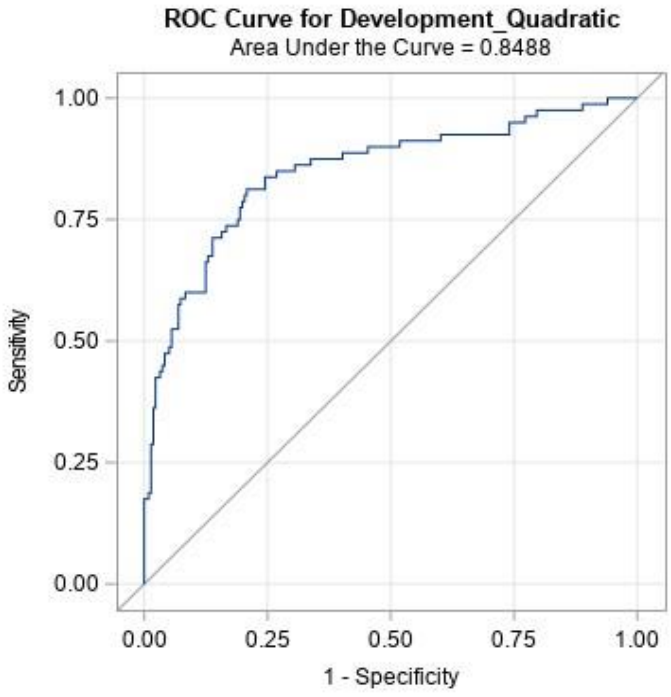
There are two ways to assess the performance of a binary classifier:

1. Discrimination measures the ability of the model to properly rank order the target from low to high, the degree of agreement between predicted rank order and actual class is an indication of a model's discrimination power, ROC and KS are example metrics for binary classifiers. Discrimination is important when your usage is based on prediction rank ordering rather than the value of the prediction, e.g., selecting the most likely to respond 10% of customer depend only on rank ordering.
2. Accuracy measures the ability of the model to provide accurate point estimates of the probability of event, models with systemic bias in accuracy may indicate lack of fit. Accuracy is also important when you need to use the prediction value directly, for example, the expected return can be calculated as probability of purchase x sales

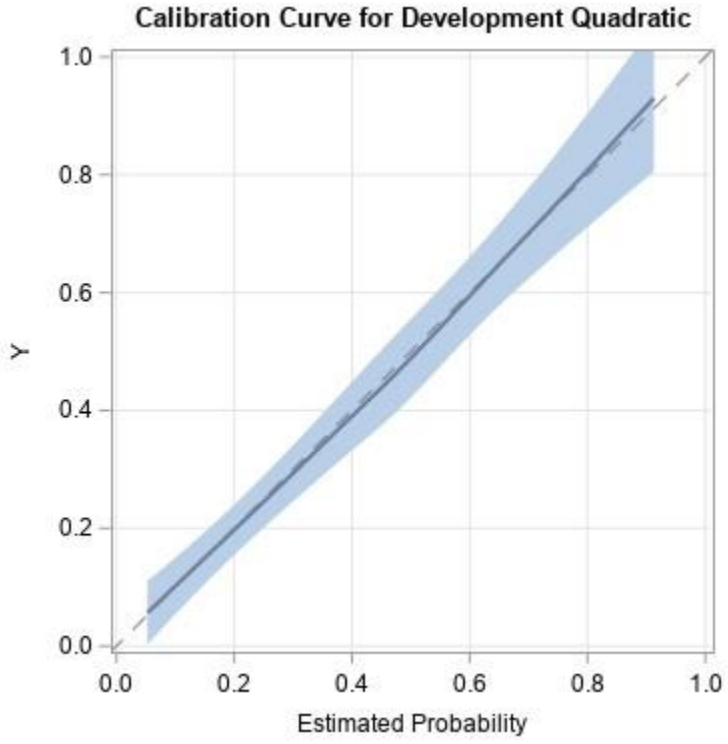
price. Accuracy is viewed on a Calibration Curve (described later). It is possible for a model to provide high discrimination power without being accurate, e.g., translation and/or scaling will affect accuracy but not discrimination.

Binary classifier is usually built using PROC LOGISTIC, ROC is calculated by the procedure directly. To draw calibration curves, the prediction and target need to be stored and plotted separately. The following code simulates a development and a validation dataset, ROC and Calibration curve for the development dataset are produced:

```
title;  ods noproctitle;  ods graphics / width=6in height=4in noborder;

%let nobs = 500;
%let splt = 0.6;
data Dev Val;
    call streaminit(5678);
    do i = 1 to &nobs;
        x = rand("Uniform", -3, 3);
        logit = -2.5 + 0.8*x + 0.3*x**2 + rand("Normal", 0, 0.1);
        prob = logistic(logit);
        Y = rand("Bernoulli", prob);
        if rand('uniform') <= &splt then output Dev;
                                    else output Val;
    end;
    keep y x;
run;

ods select ROCCurve;
proc logistic data=Dev plots(only)=roc;
    Development_Quadratic: model Y(event='1') = x x*x;
    output out=DQScore predicted=Prob;
    score data=Val out=VQScore(keep=y x p_1 rename=(P_1=Prob));
run;
```



```
title 'Calibration Curve for Development Quadratic';
proc sgplot data=DQScore noautolegend aspect=1;
    loess x=Prob y=y / interpolation=cubic clm nomarkers;
    lineparm x=0 y=0 slope=1 / lineattrs=(color=grey pattern=dash);
    yaxis grid values=(0 to 1 by 0.2);  xaxis grid values=(0 to 1 by 0.2);
run;
```

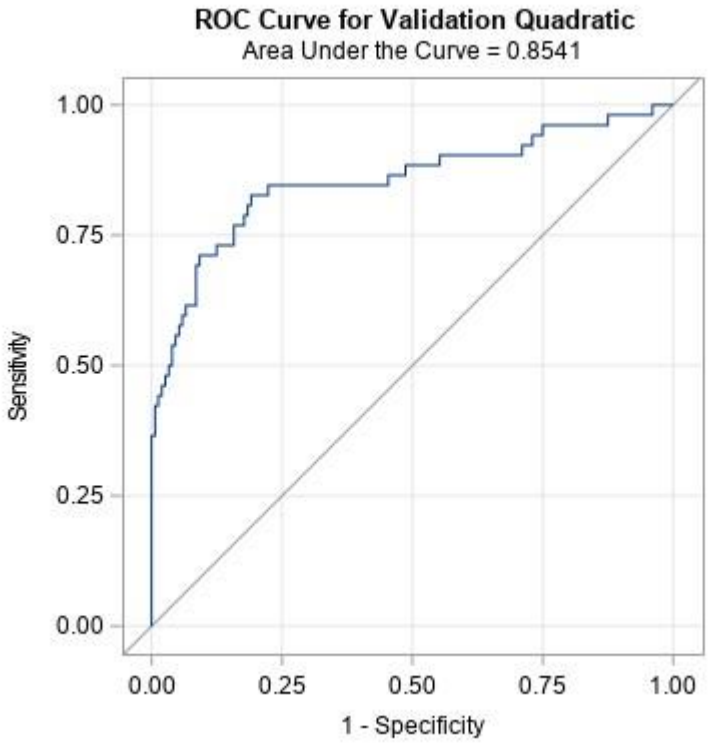


3. VALIDATION PERFORMANCE

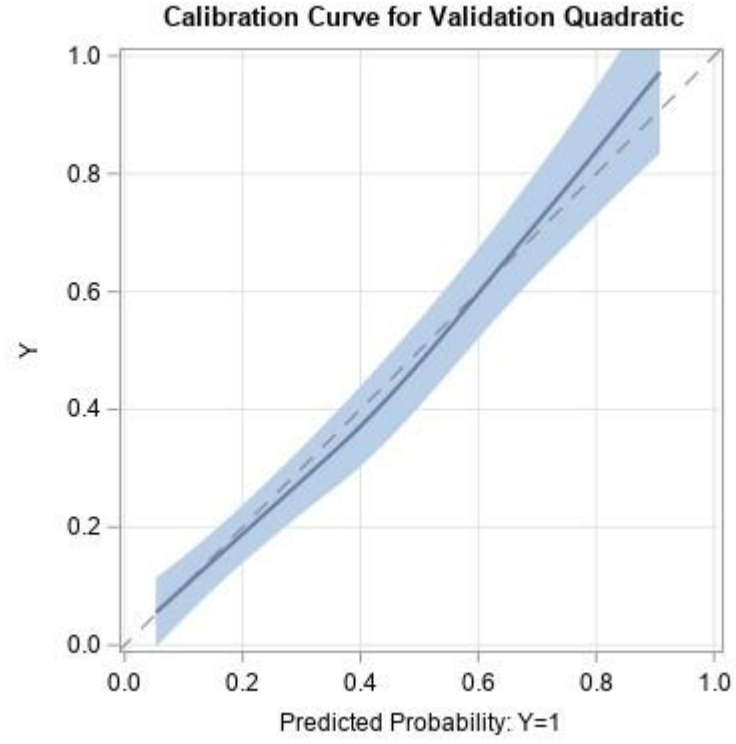
When assessing the performance of a binary classifier on another dataset, we only need the actual target and the prediction of the model. The code above scored the validation dataset at the same time as model development for convenience, but scoring can be accomplished in many ways, e.g., the code statement can be used to store the equation and score other datasets later.

When given a dataset with target and prediction, ROC and accuracy can be assessed using PROC LOGISTIC. The nofit option will suppresses model fitting, the roc statement will produce the roc curve using the predictor as is:

```
ods select ROCCurve;
proc logistic data=VQScore plots(only)=roc;
    model Y(event='1') = Prob / nofit;
    roc 'Validation Quadratic' pred=Prob;
run;
```



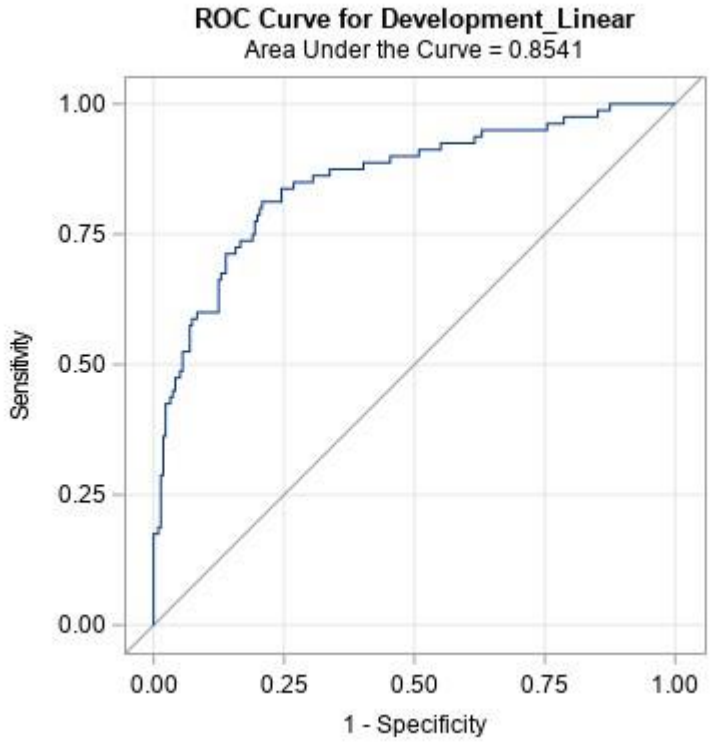

```
title 'Calibration Curve for Validation Quadratic';
proc sgplot data=VQScore noautolegend aspect=1;
    loess x=Prob y=y / interpolation=cubic clm nomarkers;
    lineparm x=0 y=0 slope=1 / lineattrs=(color=grey pattern=dash);
    yaxis grid values=(0 to 1 by 0.2);  xaxis grid values=(0 to 1 by 0.2);
run;
```



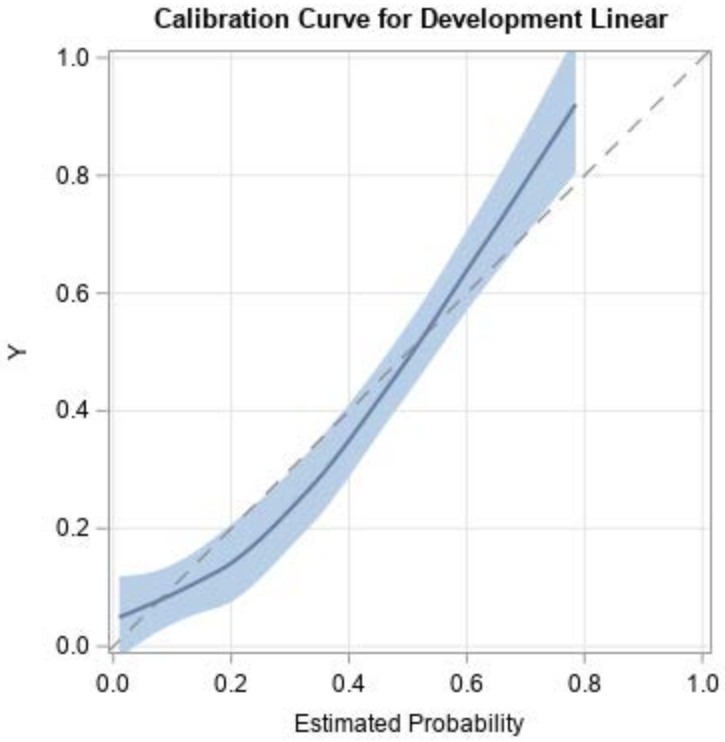
4. LINEAR MODEL

Using a linear model leads to loss of accuracy with similar performance:

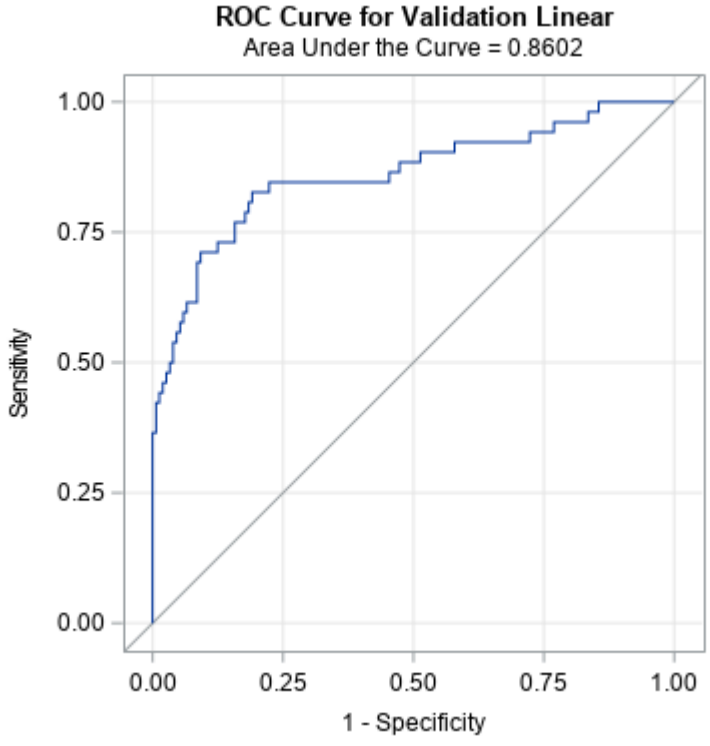
```
ods select ROCCurve;
proc logistic data=Dev plots(only)=roc;
    Development_Linear: model Y(event='1') = x;
    output out=DLScore predicted=Prob;
    score data=Val out=VLScore(keep=y x p_1 rename=(P_1=Prob));
run;
```



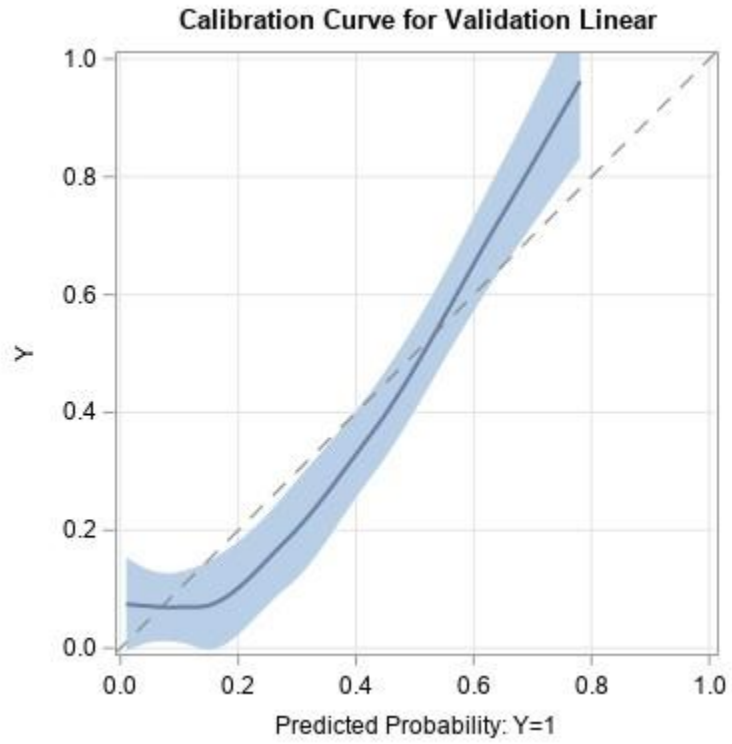

```
title 'Calibration Curve for Development Linear';
proc sgplot data=DLScore noautolegend aspect=1;
    loess x=Prob y=y / interpolation=cubic clm nomarkers;
    lineparm x=0 y=0 slope=1 / lineattrs=(color=grey pattern=dash);
    yaxis grid values=(0 to 1 by 0.2); xaxis grid values=(0 to 1 by 0.2);
run;
```




```
ods select ROCCurve;
proc logistic data=VLScore plots(only)=roc;
    model Y(event='1') = Prob / nofit;
    roc 'Validation Linear' pred=Prob;
run;
```



```
title 'Calibration Curve for Validation Linear';
proc sgplot data=VLScore noautolegend aspect=1;
    loess x=Prob y=y / interpolation=cubic clm nomarkers;
    lineparm x=0 y=0 slope=1 / lineattrs=(color=grey pattern=dash);
    yaxis grid values=(0 to 1 by 0.2);     xaxis grid values=(0 to 1 by
0.2);
run;
```



It's interesting to note that, for both development and validation, the linear model has what appears to be better discrimination power while being less accurate. The calibration curve is giving us insight into model behaviour that is not visible from ROC alone.

5. COMPARISON BETWEEN DEVELOPMENT AND VALIDATION

At this point we have four point estimates of model performance,

ROC	Development	Validation
Quadratic	0.8488	0.8541
Linear	0.8541	0.8602

It looks like the linear model is slightly better than the quadratic model, and both model perform better on validation than development. But, without an understanding of the variability of performance, we can't really be sure.

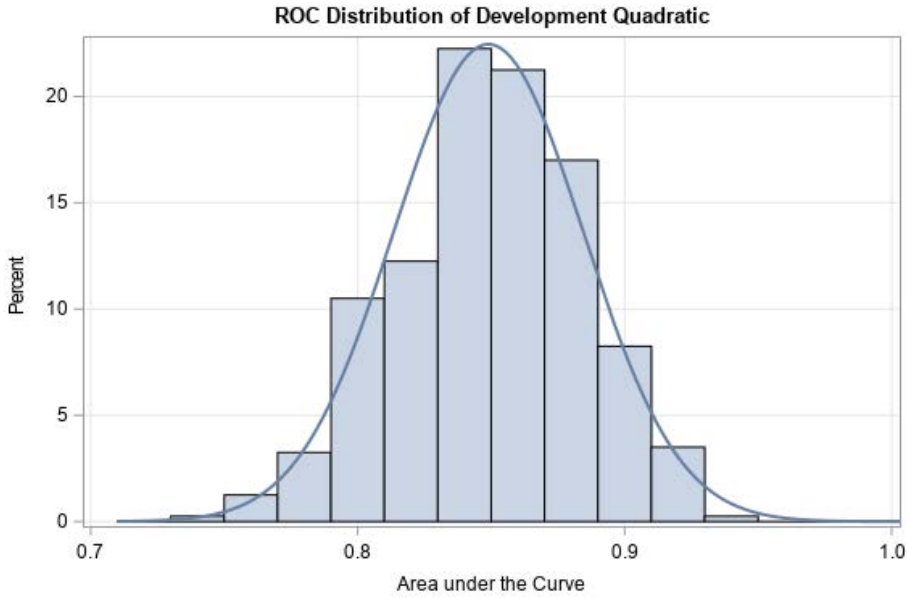
One way to obtain more robust estimate of model performance is to evaluate its performance on a large number of bootstrap samples. Using a macro loop to take a sample, compute ROC, append to a dataset is slow and ill-advised when you can leverage by-group processing built into SAS. The code below use PROC SURVEYSELECT with reps= to create many bootstrap samples and runs PROC LOGISTIC with a by statement to calculate ROC for each replicate:

```
proc surveyselect data=DQScore out=DQReps samprate=0.5 method=urs
seed=1234 reps=400 noprint;  run;
ods select none;                        *--- suppress output ---;
ods table ROCAssociation=DQAssoc;       *--- capture table   ---;
proc logistic data=DQReps plots=none;
    by replicate;
    model Y(event='1') = Prob / nofit;
    roc 'Development Quadratic' pred=Prob;
run;
ods select all;

title 'ROC Distribution of Development Quadratic';
proc means  data=DQAssoc(keep=Area rename=(Area=ROC)) n min p5 p10 mean std
p90 p95 max;  run;
```

ROC Distribution of Development Quadratic

Analysis Variable : ROC Area under the Curve								
N	Minimum	5th Pctl	10th Pctl	Mean	Std Dev	90th Pctl	95th Pctl	Maximum
400	0.7489796	0.7905921	0.7983760	0.8490065	0.0355129	0.8946365	0.9066227	0.9313804

```
proc sgplot data=DQAssoc(keep=Area rename=(Area=ROC)) noautolegend;
    histogram roc / nbins=20;
    density roc;
    yaxis grid;     xaxis grid values=(0.7 to 1 by 0.1);
run;
```



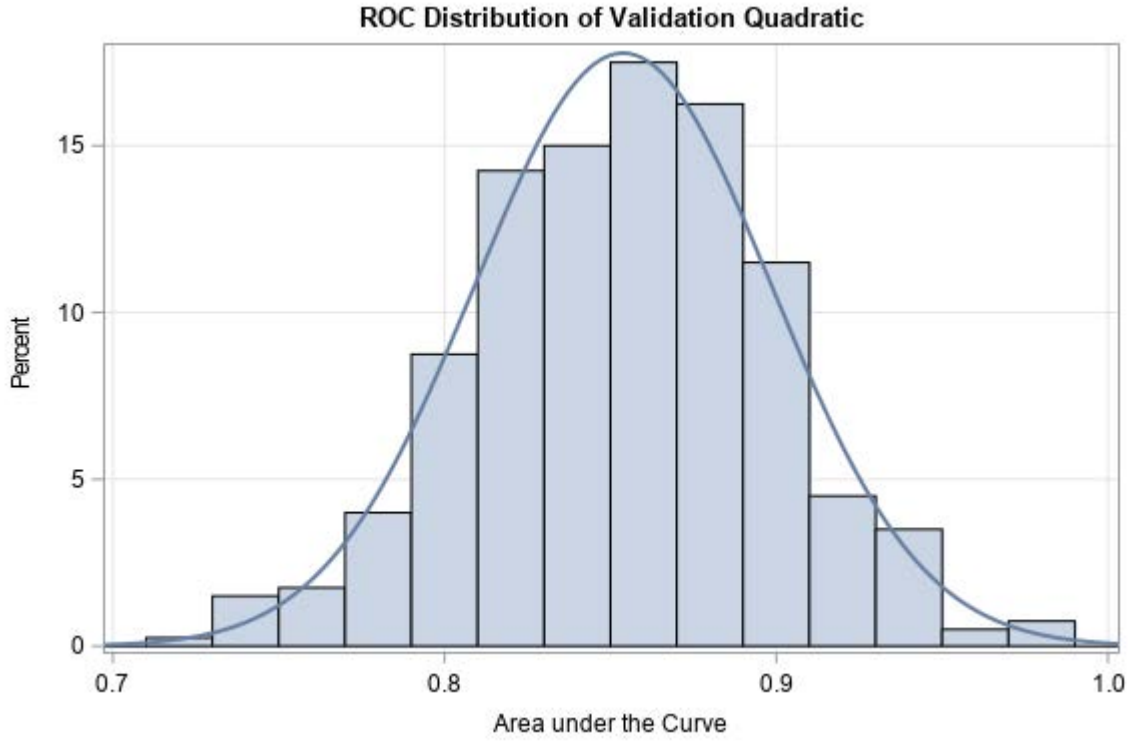
We can repeat the bootstrap process on the validation dataset:

```
proc surveyselect data=VQScore out=Reps samprate=0.5 method=urs
seed=1234 reps=400 noprint;  run;
ods select none;                        *--- suppress output ---;
ods table ROCAssociation=VQAssoc;       *--- capture table   ---;
proc logistic data=Reps plots=none;
    by replicate;
    model Y(event='1') = Prob / nofit;
    roc 'Validation Quadratic' pred=Prob;
run;
ods select all;

title 'ROC Distribution of Validation Quadratic';
proc means  data=VQAssoc(keep=Area rename=(Area=ROC)) n min p5 p10 mean std
p90 p95 max;  run;
```

ROC Distribution of Validation Quadratic

Analysis Variable : ROC Area under the Curve								
N	Minimum	5th Pctl	10th Pctl	Mean	Std Dev	90th Pctl	95th Pctl	Maximum
400	0.7294922	0.7812378	0.7981839	0.8538354	0.0448716	0.9090168	0.9288847	0.9800937

```
proc sgplot data=VQAssoc(keep=Area rename=(Area=ROC)) noautolegend;
    histogram roc / nbins=20;
    density roc;
    yaxis grid;     xaxis grid values=(0.7 to 1 by 0.1);
run;
title;
```



We can compare the quadratic model's performance between development and validation with two sample T-test:

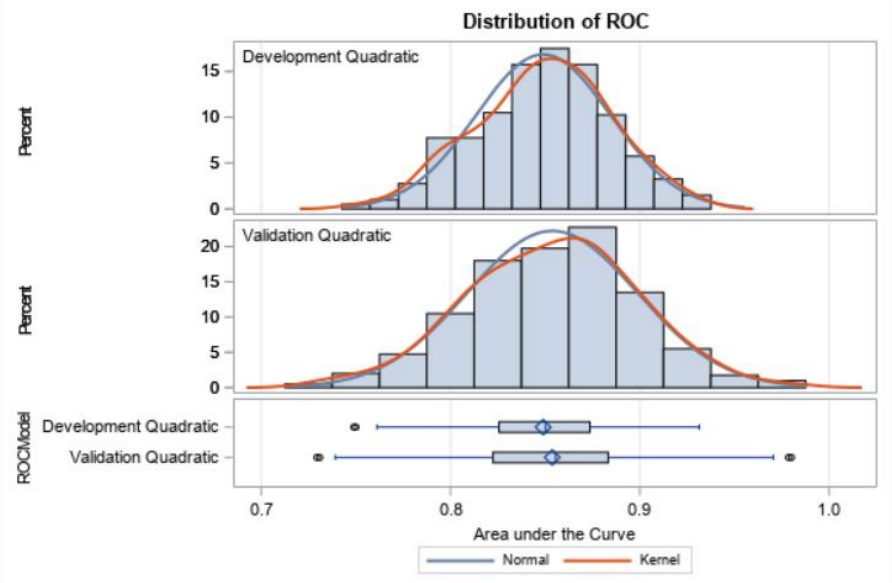
```
data Combine;
    set DQAssoc(rename=(Area=ROC)) VQAssoc(rename=(Area=ROC));
run;

proc ttest;
    ods exclude qqplot;
    class ROCModel;
    var roc;
run;
```

ROCModel	Method	N	Mean	Std Dev	Std Err	Minimum	Maximum
Development Quadratic		400	0.8490	0.0355	0.00178	0.7490	0.9314
Validation Quadratic		400	0.8538	0.0449	0.00224	0.7295	0.9801
Diff (1-2)	Pooled		-0.00483	0.0405	0.00286		
Diff (1-2)	Satterthwaite		-0.00483		0.00286		

ROCModel	Method	Mean	95% CL Mean		Std Dev	95% CL Std Dev	
Development Quadratic		0.8490	0.8455	0.8525	0.0355	0.0332	0.0382
Validation Quadratic		0.8538	0.8494	0.8582	0.0449	0.0420	0.0482
Diff (1-2)	Pooled	-0.00483	-0.0104	0.000788	0.0405	0.0386	0.0426
Diff (1-2)	Satterthwaite	-0.00483	-0.0104	0.000788			

Method	Variances	DF	t Value	Pr > |t|
Pooled	Equal	798	-1.69	0.0919
Satterthwaite	Unequal	758	-1.69	0.0919

Equality of Variances				
Method	Num DF	Den DF	F Value	Pr > F
Folded F	399	399	1.60	<.0001

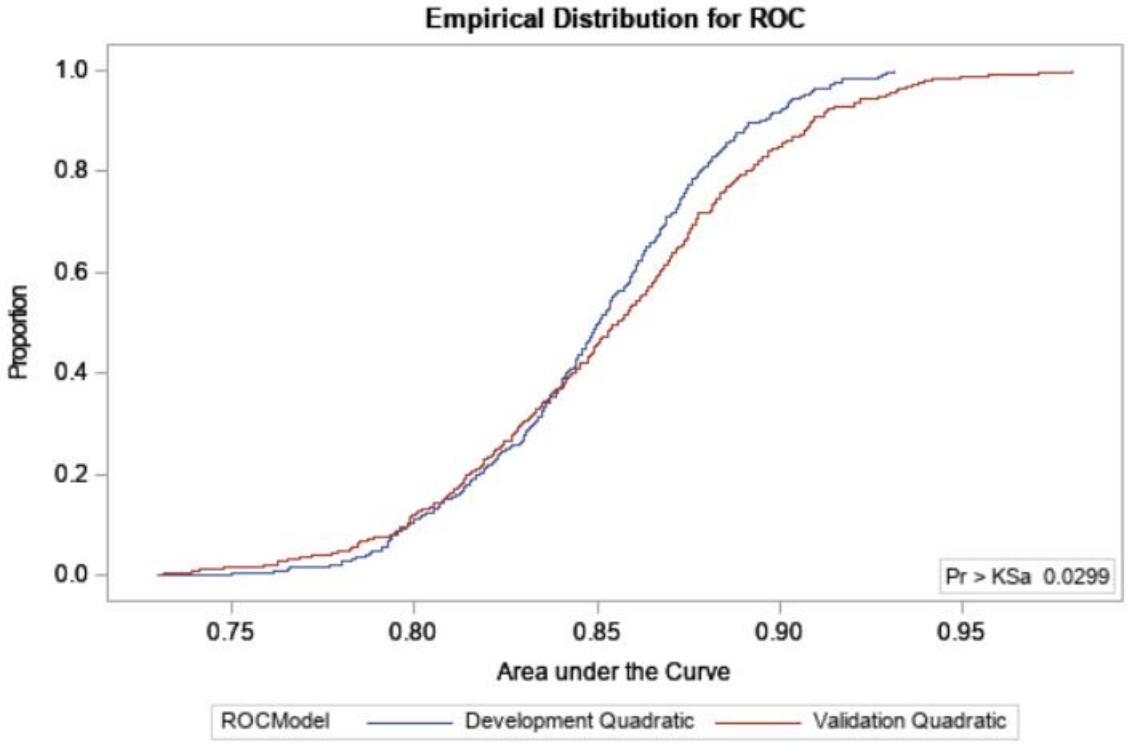


T-test says the mean ROC is the same. We can use PROC NPAR1WAY to test for equivalence of distribution:

```
ods select KSTest KS2Stats EDFPlot;
proc npar1way edf plots=edf;
    class ROCModel;
    var roc;
run;
```

Kolmogorov-Smirnov Test for Variable ROC Classified by Variable ROCModel			
ROCModel	N	EDF at Maximum	Deviation from Mean at Maximum
Development Quadratic	400	0.822500	1.0250
Validation Quadratic	400	0.720000	-1.0250
Total	800	0.771250	
Maximum Deviation Occurred at Observation 170			
Value of ROC at Maximum = 0.880682			

Kolmogorov-Smirnov Two-Sample Test (Asymptotic)			
KS	0.051250	D	0.102500
KSa	1.449569	Pr > KSa	0.0299



NPAR1WAY says the two ROC distributions are not identical, T-test shows validation ROC has larger variance. If we want to understand how often validation performance is better than development performance, we can use PROC LOGISTIC to compute Condordant / Discordant pairs:

```
ods select Association;
proc logistic data=Combine plots=none;
    model ROCModel(event='Validation Quadratic') = roc;
run;
```

Association of Predicted Probabilities and Observed Responses			
Percent Concordant	53.4	Somers' D	0.069
Percent Discordant	46.6	Gamma	0.069
Percent Tied	0.0	Tau-a	0.034
Pairs	160000	c	0.534

Between these two samples, the quadratic model performed better on validation 53.4% of the time while performance on development was better 46.6% of the time.

6. COMPARISON BETWEEN DEVELOPMENT AND VALIDATION

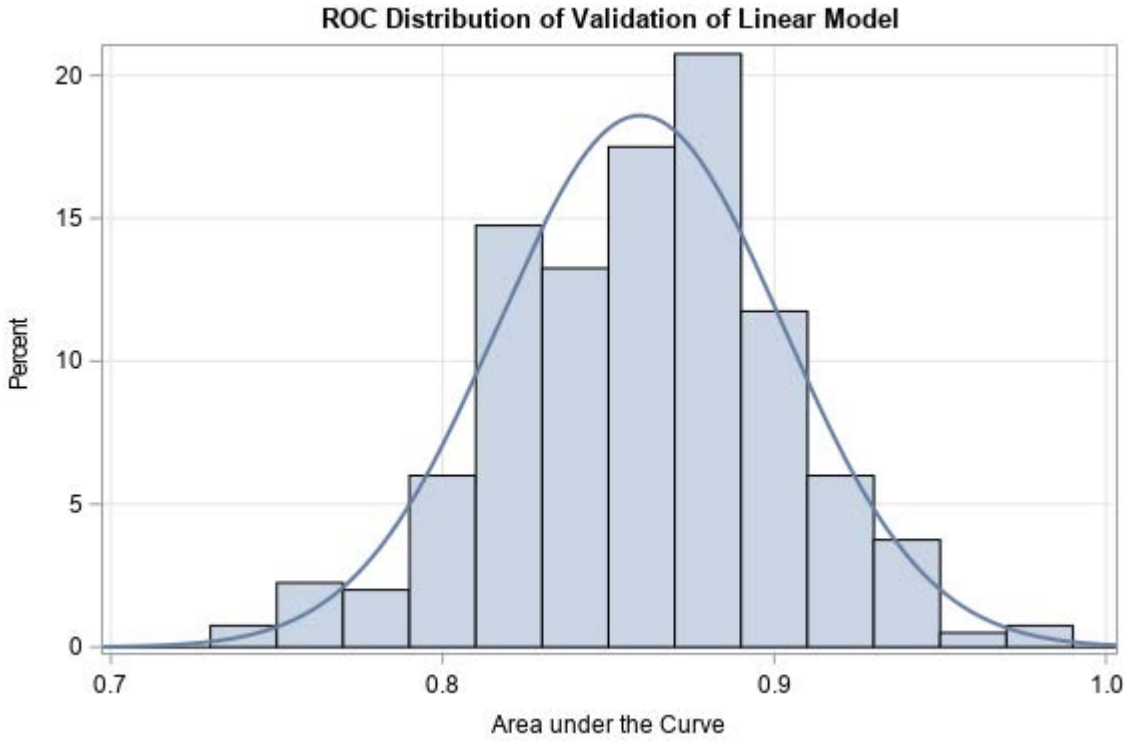
We can compare the two models on validation:

```
proc surveyselect data=VLScore out=VLReps samprate=0.5 method=urs
seed=1234 reps=400 noprint;  run;

ods select none;                          *--- suppress output ---;
ods table ROCAssociation=VLAssoc;         *--- capture table   ---;
proc logistic data=VLReps plots=none;
    by replicate;
    model Y(event='1') = Prob / nofit;
    roc 'Validation Linear' pred=Prob;
run;
ods select all;

title 'ROC Distribution of Validation of Linear Model';
proc means  data=VLAssoc(keep=Area rename=(Area=ROC)) n min p5 p10 mean std
p90 p95 max;  run;
```

ROC Distribution of Validation of Linear Model

Analysis Variable : ROC Area under the Curve								
N	Minimum	5th Pctl	10th Pctl	Mean	Std Dev	90th Pctl	95th Pctl	Maximum
400	0.7363281	0.7896760	0.8070997	0.8596609	0.0428904	0.9121239	0.9299155	0.9800937

```
proc sgplot data=VLAssoc(keep=Area rename=(Area=ROC)) noautolegend;
    histogram roc / nbins=20;
    density roc;
    yaxis grid;     xaxis grid values=(0.7 to 1 by 0.1);
run;
title;
```



Because this is two models on the same dataset, we can perform paired T-test:

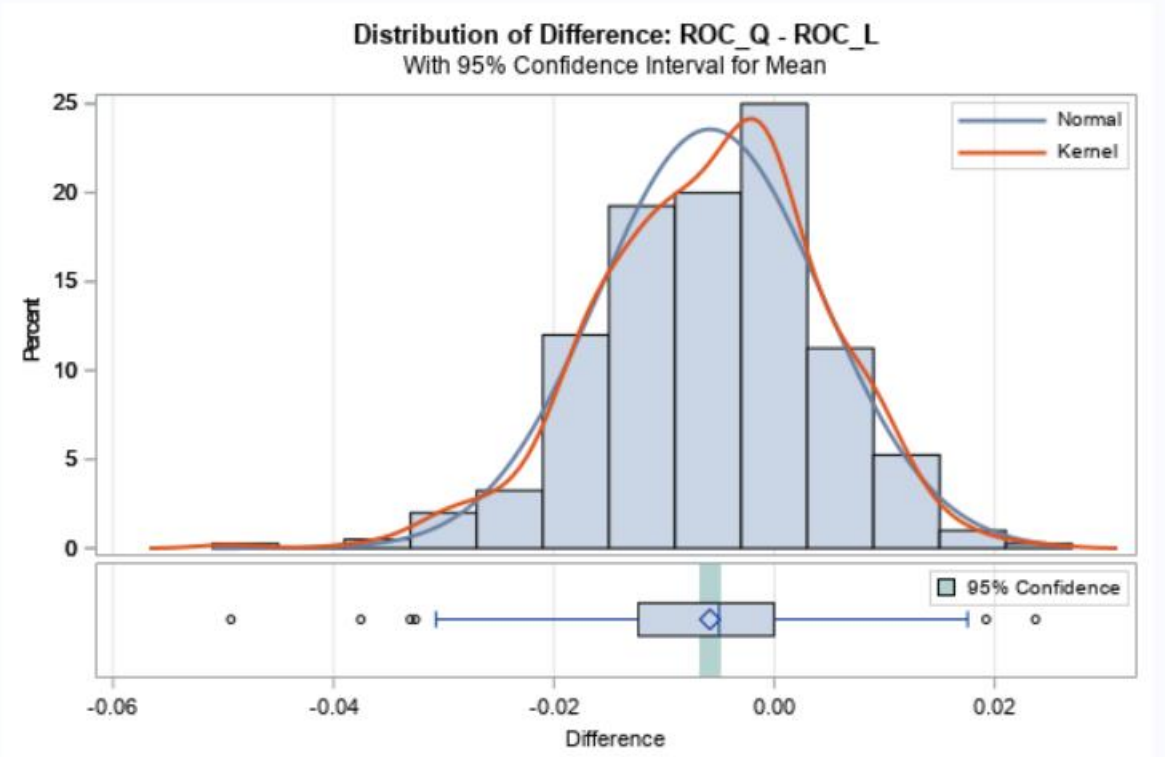
```
data Combine;
    set VQAssoc(rename=(Area=ROC_Q));
    set VLAssoc(rename=(Area=ROC_L));
run;

proc ttest;
    ods select Statistics ConfLimits TTests SummaryPanel;
    paired ROC_Q * ROC_L;
run;
```

Difference: ROC_Q - ROC_L

N	Mean	Std Dev	Std Err	Minimum	Maximum
400	-0.00583	0.0102	0.000508	-0.0493	0.0237

Mean	95% CL Mean		Std Dev	95% CL Std Dev	
-0.00583	-0.00682	-0.00483	0.0102	0.00950	0.0109

DF	t Value	Pr > |t|
399	-11.47	<.0001



The quadratic model has slightly worse discriminatory power than the linear model, however it provides more accurate probability estimates.

The detailed understanding of model performance variability and accuracy now allow us to make a more informed decision than the four ROC point estimates.

7. DURING MODEL BUILD

The approach illustrated above assists in evaluating model performance post build. The same principle can be applied during model build to assess functional form stability:

1. Perform bootstrap sampling
2. Build model with the same functional form for each replicate
3. Calculate performance metric for each replicate

If there is large variation in parameter estimates or performance, it suggests the functional form is not robust and has trouble with certain parts of the predictor space.

8. ACKNOWLEDGEMENT

The code for calibration curve is from The DO Loop Blog by Rick Wicklin: Calibration plots in SAS® (https://blogs.sas.com/content/iml/2018/05/14/calibration-plots-in-sas.html).

This paper is written with SAS® 9.4, JupyterLab (https://github.com/jupyterlab/jupyterlab) with sas_kernel (https://github.com/sassoftware/sas_kernel) developed by SAS (Jared Dean, Tom Weber).

9. CONTACT INFORMATION

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

Daymond Ling
Seneca College of Applied Arts and Technology
daymond.ling@senecacollege.ca