

Preparing for Basel II

Common Problems, Practical Solutions

Part 3: Model Validation

by Jeffrey S. Morrison

Previous articles in this series have focused on the problems of missing data and how to get the most out of the information you do have. This article offers some practical statistical advice for overcoming the challenges of validating a PD model when you have relatively few defaults. Hypothetical datasets are used to model the probability of default with predictive attributes such as LTV, payment history, bureau scores, and income.

Put simply, *model validation* is checking the accuracy of your model over some period of time. For example, the number of loans that actually defaulted during the year is compared with the *predicted* default probabilities. Although accuracy can be measured a variety of ways, one of the most common is through the *KS value*, as calculated in Figure 1.

Once scored with the default model, the validation sample is sorted by score, and counts are developed in 5% increments. The KS value, then, is simply the maximum difference in the cumulative percentage counts between the two populations—defaults and nondefaults (columns F & G). The KS

value in Figure 1 is found in the ninth bucket with a value of 74.1%. This process is discussed in detail in the June 2003 issue of *The RMA Journal*.¹ In general and with all other things remaining equal, the higher the KS value, the more accurate the model. A perfect model would have a KS value of 100.

If sample size and the number of defaults are plentiful, then setting aside a “holdout” sample to be used in the testing process is the most common approach to model validation. You simply estimate the model once using a single sample, apply your predictive model to your other (holdout) sample, and then calculate your accuracy measures. This technique often is used in the world of credit scoring,

© 2004 by RMA. Jeff Morrison is vice president, Credit Metrics—PRISM Team, at SunTrust Banks, Inc., Atlanta, Georgia.

Figure 1

Typical Validation Report							
A	B	C	D	E	F	G	H
5% Bucket	Min Probability	Max Probability	# of Defaults	# of Non-defaults	Cumulative % Defaults	Cumulative % Nondefaults	Difference in % Cumulatives
1	0.987637	0.998625	33	4	10.1852	0.9390	9.2
2	0.961737	0.987637	35	3	20.9877	1.6432	19.3
3	0.932949	0.961737	36	1	32.0988	1.8779	30.2
4	0.897801	0.932949	36	2	43.2099	2.3474	40.9
5	0.821660	0.897801	30	7	52.4691	3.9906	48.5
6	0.813351	0.821660	38	0	64.1975	3.9906	60.2
7	0.545989	0.813351	28	9	72.8395	6.1033	66.7
8	0.530848	0.545989	24	14	80.2469	9.3897	70.9
9 →	0.398955	0.519828	22	15	87.0370	12.9108	74.1
10	0.314351	0.398955	8	30	89.5062	19.9531	69.6
11	0.192926	0.295930	5	33	91.0494	27.6995	63.3
12	0.132092	0.192926	3	34	91.9753	35.6808	56.3
13	0.117058	0.132092	5	33	93.5185	43.4272	50.1
14	0.099934	0.117058	0	37	93.5185	52.1127	41.4
15	0.099934	0.099934	0	38	93.5185	61.0329	32.5
16	0.090868	0.099934	7	30	95.6790	68.0751	27.6
17	0.090868	0.090868	0	38	95.6790	76.9953	18.7
18	0.069882	0.090868	12	25	99.3827	82.8638	16.5
19	0.045697	0.057391	2	36	100.0000	91.3146	8.7
20	0.036294	0.045697	0	37	100.0000	100.0000	0

where consumer credit-card data is rich with default information.

Data splitting. The *single holdout* procedure is the simplest case of a more general procedure called *data splitting*. Instead of just splitting data a single time into an estimation sample and a holdout sample, data splitting allows you to repeat the procedure without additional information. The model is estimated multiple times, and its performance is tested using different holdout samples through a random sampling process. In data splitting, there is no replacement of observations on each random draw. When repeated tests are made for a single holdout observation, the procedure is referred to as *jackknifing*. Jackknifing is especially useful when you wish to examine the impact of each observation on your modeling coefficients. If you use a larger holdout sample, then this procedure can easily be applied to broader validation analyses.

Let's take an example of 10,000 accounts available to build and test our model. Validating model performance using data splitting requires two inputs: 1) the number of times the model is to be estimated;

and 2) the number of accounts in the holdout sample. If we wanted our model to be estimated 100 times using a holdout sample of 1,000, the procedure would randomly select 9,000 accounts for model development and reserve 1,000 accounts for validation. The process would repeat itself 100 times in which the sample is resplit, a new model is estimated, a holdout sample of 1,000 accounts is scored using the new coefficients, and a new KS value is computed. It is important to keep in mind that the same variable specification must be used each time a new model is estimated. For example, you can't include the bureau score in one model and not in the next model. The predictor variables have to be the same each time. Once the procedure is completed, the KS values are averaged for the 100 validation samples, yielding a potentially more robust picture of validation than might be obtained from a single holdout sample.

Figure 2 shows an example of the data-splitting process in which 100 models are requested with a holdout sample of 1,000 accounts each. The predictive attributes are shown as generic names, such as X1, X2, and X3, as our interest is on the validation procedures rather than the actual predictive attrib-

Figure 2

Split Sampling Technique: 100 model estimations, holdout sample = 1,000

Model	KS	Intercept	X1 Coeff	X2 Coeff	X3 Coeff	X4 Coeff	X5 Coeff	X6 Coeff	X7 Coeff
1	50.3	-0.15743	-0.06920	-0.0002	0.04475	0.24436	0.3561	0.11466	-0.1152
2	55.9	-0.26316	-0.06855	-0.0002	0.04424	0.25439	0.3824	0.11319	-0.1167
3	63.9	-0.16548	-0.07002	-0.0001	0.04428	0.21611	0.3705	0.10761	-0.1158
4	59.4	0.03743	-0.05818	-0.0002	0.04426	0.22283	0.3538	0.10939	-0.1161
5	72.7	-0.22145	-0.08037	-0.0002	0.04497	0.18191	0.3699	0.09938	-0.1127
6	67	-0.39643	-0.06863	-0.0002	0.04345	0.24208	0.3761	0.10381	-0.1119
7	68.6	0.17012	-0.06821	-0.0002	0.04356	0.23447	0.3421	0.09618	-0.1139
8	57.7	0.30702	-0.06292	-0.0003	0.04499	0.21070	0.3498	0.11077	-0.1195
9	61.9	-0.61718	-0.06892	-0.0002	0.04491	0.22334	0.4028	0.11524	-0.1141
10	71.3	-0.18459	-0.06159	-0.0002	0.04532	0.21734	0.3608	0.10964	-0.1139
-	-	-	-	-	-	-	-	-	-
91	66.2	-0.18810	-0.05191	-0.0003	0.04512	0.23346	0.3639	0.10374	-0.1131
92	60.7	-0.15959	-0.08957	-0.0002	0.04627	0.20821	0.3817	0.10491	-0.1157
93	64.6	-0.16384	-0.06396	-0.0002	0.04675	0.21762	0.3565	0.10839	-0.1143
94	62	-0.20946	-0.06968	-0.0002	0.04396	0.22130	0.3581	0.11276	-0.1142
95	56.5	-0.07595	-0.07654	-0.0002	0.04454	0.21509	0.3640	0.11212	-0.1170
96	65.9	0.04393	-0.08882	-0.0002	0.04530	0.24979	0.3667	0.09993	-0.1165
97	55.5	0.20491	-0.06890	-0.0002	0.04523	0.20139	0.3393	0.11227	-0.1173
98	58.2	0.22441	-0.07109	-0.0002	0.04474	0.21894	0.3382	0.10890	-0.1177
99	69.3	0.33554	-0.07299	-0.0002	0.04586	0.23804	0.3332	0.09963	-0.1164
100	60.4	-0.40298	-0.07276	-0.0002	0.04441	0.23192	0.3644	0.11966	-0.1130

utes. For illustrative purposes, only the first and last 10 models are listed. The KS value for each model is reported in column KS, while the model’s parameter estimates are reported in columns X1 through X7. Notice how the parameter estimates vary across the different models. This is because our sampling scheme splits the data each time using a random sampling procedure. Also notice how the KS values differ—some are as high as 72 (model 5) and others are as low as 50 (model 1). Even if the size of the samples is increased, the spread of the KS values is significant. When the KS is averaged across the 100 holdout samples, the KS value is 61.4.

Bootstrapping. Data-splitting techniques can add additional insight into model accuracy in cases where the sample data is readily attainable. However, if the sample size and the number of defaults are not plentiful, then a different validation approach is needed. For example, if your portfolio had only 500 accounts default within a one-year time frame, you might be hard pressed to sacrifice any data for a holdout sample. One approach that could help is called *bootstrapping*. The dictionary defines the

process of bootstrapping as “to promote and develop by use of one’s own initiative and work without reliance on outside help.” The same thing applies to statistical bootstrapping. Basically, it means you’re on your own, having to use the information on hand without relying on other sources of data. Although you could perform a simple validation on the model’s estimation data rather than a holdout sample, such a procedure typically overstates the level of accuracy when used to score new data. Without some bootstrapping procedure in place, this less-than-satisfactory approach might be your only alternative.

In data splitting, remember there is no replacement of observations upon each random draw. However, in bootstrapping, samples are created from the original dataset with replacement. So when a bootstrap sample is created, it can (and usually will) contain the same observation from the original data more than one time. However, this weakness is mitigated to a large extent by running the procedure time after time in an iterative framework.

There are other differences between bootstrapping and data splitting. In data splitting, you always

have a separate holdout sample to compute your accuracy measure that is not used in the estimation of the model parameters: The analyst repeatedly creates a random sample from the original sample and estimates the model from it. However, in bootstrapping, the model is then used to score the original dataset in order to compute measures of validation accuracy. Once this is done the desired number of times (usually 100-200 repetitions), the accuracy measures are averaged. This procedure is perfect for applications where you cannot afford to waste any valuable default information simply to test your model's accuracy. There is a sizable body of literature showing that this type of validation procedure will lead to a much more realistic performance measure than one obtained by simply using the model development sample for validation.

One thing that data splitting and bootstrapping *do* have in common is the necessity of having the regression model completely specified during the process. In other words, using automatic variable selection techniques is not recommended because slight changes in the sample could result in a different model specification. If that happens, then you would not be validating the same model throughout the process.²

A slight variation to the basic bootstrapping procedure can provide an even better performance picture if the sample size is a limiting factor. Think of this as *enhanced bootstrapping*. The procedure is as follows:

- a. Estimate your model with the entire original sample.
- b. Use this model to score the entire original sample.
- c. Compute the performance measures (KS) on the original sample.
- d. Execute the bootstrap resampling procedure with replacement from the original sample.
- e. Estimate a new model.
- f. Score and validate the bootstrap sample.
- g. Score and validate the original dataset using the new bootstrap model.
- h. Calculate the performance optimism by taking the performance measure from the bootstrap model in (f) and subtract from it the performance measure of the original model as determined in (g).
- i. Repeat steps (d) through (h) the desired number of times (say, 100 times).
- j. Compute the average performance optimism

across the number of repetitions.

- k. Compute the bias-corrected performance measure by subtracting the average optimism from the performance measure of the original sample (step c).

Figure 3 shows the variations in the bootstrap samples. Since the original sample had 10,000 observations, note that each bootstrap sample also has a total of 10,000

observations (N). Remember, the bootstrap samples were derived from the original dataset with replacement, meaning that each bootstrap sample could contain multiple draws of the same record. In the model, defaults were assigned a value of 1. Non-defaults were assigned a value of 0. As shown in Figure 3, the mean value of the "default" variable changes for each sample, again reflecting the bootstrap sampling procedure. In other words, with each iteration, we have a sample that is different from the one before, each having a different number of defaults that are close to but not exactly the same as in the original sample.

Figure 4 shows the variations in the KS values for the first and last 10 samples in the bootstrapping process. The column called KS_BOOT refers to the

Figure 3

Bootstrap Samples
(Only 5 out of 100 shown)

Sample	STAT	Default
1	N	10000
1	MIN	0
1	MAX	1
1	MEAN	0.039106
1	STD	0.193855
2	N	10000
2	MIN	0
2	MAX	1
2	MEAN	0.035598
2	STD	0.185295
3	N	10000
3	MIN	0
3	MAX	1
3	MEAN	0.034459
3	STD	0.182412
4	N	10000
4	MIN	0
4	MAX	1
4	MEAN	0.039719
4	STD	0.195308
5	N	10000
5	MIN	0
5	MAX	1
5	MEAN	0.0363
5	STD	0.187043

Figure 4

Bootstrap Results			
A	B	C	D
Model	BKS_BOOT	KS_DEV_BOOT	OPTIMISM_KS
1	60.3	59.1	1.2
2	63	59.1	3.9
3	58.9	58.4	0.5
4	58.3	58.6	-0.3
5	58.8	59.6	-0.8
6	58.4	59.6	-1.2
7	59.1	58.8	0.3
8	59.2	59.3	-0.1
9	58.9	59.1	-0.2
10	59.1	59.6	-0.5
-	-	-	-
91	56	59.3	-3.3
92	62.5	59.3	3.2
93	58.6	59.1	-0.5
94	57.6	58.7	-1.1
95	62.6	59.6	3
96	59.6	59.6	0
97	61.5	59.1	2.4
98	58.4	59.3	-0.9
99	60	58.6	1.4
100	61.7	58.8	2.9

KS value obtained from the bootstrap model.
KS_DEV_BOOT refers to the KS value obtained from the bootstrap model when applied to the original sample. The difference is the performance optimism, labeled OPTIMISM_KS.

Note how the performance optimism (column D) has values that are both positive and negative. Overall, the average optimism over the 100 iterations measured in terms of KS was 1.02. If we had used the development data to both estimate the model and derive our performance measure (the unsatisfactory solution we mentioned earlier), the KS would have been 59.1. Since our bootstrapping procedure indicated that we have overstated this performance measure by 1.02 units, the new bias-adjusted KS is calculated as $59.1 - 1.02 = 58.08$.

Summary

Model validation is an extremely important part of the Basel Capital Accord. This means that although challenges may exist in the model development and validation process, the analyst must seek alternative solutions to provide the regulators with the most realistic picture of model performance possible. The practical solutions offered in this article will not only help the regulators better examine the accuracy of the bank's models, but will aid the modeler in developing the best model possible. □

Contact Morrison by e-mail at Jeff.Morrison@suntrust.com.

Notes

1 Morrison, Jeffrey S., "Preparing for Modeling Requirements in Basel II—Part 2: Model Validation," *The RMA Journal*, June 2003.

2 Harrell, Frank E. Jr., *Regression Modeling Strategies with Applications to Linear Models, Logistic Regression, and Survival Analysis*, Springer-Verlag New York, Inc., 2001.