Managing and Monitoring Statistical Models

Nate Derby, Stakana Analytics, Seattle, WA

ABSTRACT

Managing and monitoring statistical models can present formidable challenges when you have multiple models used by a team of analysts over time. How can you efficiently ensure that you're always getting the best results from your models? In this paper, we'll first examine these challenges and how they can affect your results. We'll then look into solutions to those challenges, including lifecycle management and performance monitoring. Finally, we'll look into implementing these solutions both with an in-house approach and with SAS®Model Manager.

KEYWORDS: SAS, manage, monitor, statistical models.

All data sets and code in this paper are available at `http://nderby.org/docs/StatModels.zip`.

INTRODUCTION: WHY MANAGE AND MONITOR MODELS?

Managing and monitoring statistical models is crucial if your organization periodically runs a large number (say, over 10) of statistical models. However, these issues are important even when there are just a few of them in production. Common challenges include the following:

- Keeping all the input correct and fresh.
- Making sure the outputs go to the right places, in the correct formats.
- Keeping the code organized for effective updating and maintenance.
- Creating and maintaining effective documentation.
- Assessing and tracking model performance.
- Effectively (preferably automatically) deciding when to update the model.

Why are these issues important? There are two main reasons, both of which result in lost money:

- *Lost employee time, energy, or productivity*. This can take various forms:
  - *Bad documentation (or lack thereof)*. When there isn't effective and updated documentation, you depend on a few key employees to keep your results accurate and updated. If one of those employees should leave for whatever reason, it's *exceedingly* difficult for the new employees to maintain those results at the same level of accuracy and timeliness. You're one employee away from possibly restarting from scratch!
  - *Using old/erroneous code*. Without proper code organization, your employees could be using old or erroneous code, resulting in lost time spent fixing it or getting erroneous results.
  - *Redoing old work*. Employees could be re-inventing the wheel.
- *Suboptimal results*. If business decisions are made based on suboptimal results, this quickly translates to lost money.

While the first reason is indeed important, this paper is more concerned with the second one. If your statistical models guide important business decisions (and if they don't, why are you maintaining them?), shouldn't those statistical models be accurate? The problem is often that the statistical model is built off of data from one time period and then never updated. While the existing statistical model is often re-run with new data, it's often never re-calibrated to include new variables that might not have been relevant during the initial time period, or to remove old variables that aren't relevant.

These are cases where *a key underlying assumption of the model changes after the model has been built*. When an underlying assumption changes dramatically, such as the sudden introduction of a new competitor,[1] it typically gets enough attention from upper management to attract quick corrective action. Often big problems happen when fundamental changes to the marketplace happen steadily but too slowly to attract attention at any one point in time:

- The migration toward smartphones and away from land lines.
- Reading the news from the web rather than print.
- Effects of climate change.
- Buying music or renting movies electronically rather than from a physical store.
- Using car sharing (in some urban areas) rather than owning a car or using public transportation.

How bad can the effect be? The *Gaussian Cupola*, described in Salmon (2009) and Jones (2009), was a statistical model guiding key decisions in the financial industry that depended on the bond market operating a certain way. When the bond market (and related markets) changed slowly over the past ten years, the assumptions behind the model were violated, but too slowly to really notice. The result was a significant contribution to the 2008 financial crisis.

If an outdated statistical model can contribute to a global financial meltdown, how can we avoid disaster with our own models? There's actually a simple answer: Incorporate some basic measures of statistical accuracy into your results. If you're using a dashboard, just add on some basic monitoring tools (and learn to use them!). These tools have been mentioned in various sources (e..g, Wei et al. (2009)), but this paper will walk through them in depth – both in terms of understanding them and incorporating them with SAS.

MANAGING

if you're running multiple models periodically, there can be many details to coordinate! However, managing statistical models has much in common with code organization, which has been well documented (e.g., Derby (2007, 2010), Dosani et al. (2010), Fecht and Stewart (2008), and Winn Jr. (2004)). While there are differing opinions on specifics, two common themes are to

- *Organize* as much as possible.
- *Automate* as much as possible.

While they are both generally important, *code organization* can often be more important than *code efficiency* because of the human element: Time/resources wasted due to inefficient code is often dwarfed by the time or resources wasted due to code disorganization. For instance, a well-written but inefficient query that takes two hours longer than it should is easier to deal with than a query that's so badly written that a programmer has to re-run it multiple times and troubleshoot it – which would easily waste way more than two hours.

Code organization is one of many ideas for effective statistical model management:

- Code organization (using macros!).
- Documentation organization.
- Frequent result assessment.
- Protocol for model updates.
- Leadership, employee training and buy-in.

The last item in the above list is critical – the best model organization in the world won't help if the employees won't buy into it. As such, it's crucial that ideas for model management are embraced from upper management, a theme mentioned in many analytics books (e.g., Davenport and Harris (2007), Davenport et al. (2010)).

[1] e.g., Car2Go's sudden entry into the Seattle car sharing market in December 2012, challenging Zipcar's dominance.

Concepts from software development can be applied to statistical models as well. *Lifecycle management* is the process by which you create, use, and retire a model. Cheng (2009) describes the basics and applying it to SAS code. A major portion of this is monitoring the results, which we'll cover in the next section. It's basically a paradigm to effectively deal with the fact that a statistical model is only good for so long.

MONITORING

Monitoring is the process of paying constant attention to the accuracy of the models to decide when a statistical model should be recalibrated or rebuilt. We'll illustrate some basic concepts in model monitoring using passengers from the *Titanic* passengers in 1912.[2] First we remove passengers with missing data in the variables we'll be using:

```
DATA titanic_nomissing;
  SET home.titanic;
  IF CMISS( survived, passenger_class, sex, age, siblings_and_spouses ) = 0;
RUN;
```

In the traditional data mining spirit, we'll separate the data into the *training set* and the *test set*:

- The *training set* includes data used to develop the model.
- The *test set* (or *validation set*) includes data used to analyze the results from the model.

This framework avoids the problem of *overfitting*, which fits the peculiarities of our one data set (i.e., the "noise") that can distort the main results (the "signal") guiding our business decisions.[3] Following convention, we'll use simple random sampling to pick 60% of the data for the training set and the remaining 40% for the test set:

```
PROC SURVEYSELECT DATA=titanic_nomissing OUT=training METHOD=srs SAMPRATE=0.60
  SEED=12345 NOPRINT;
RUN;

DATA test;
  MERGE titanic_nomissing ( IN=a ) training ( IN=b );
    BY name;
  IF a AND NOT b;
RUN;
```

Are these two data sets distributed about the same? Let's compare our two data sets:

```
PROC FREQ DATA=training;
  TABLES survived*passenger_class*sex;
RUN;

PROC FREQ DATA=test;
  TABLES survived*passenger_class*sex;
RUN;
```

The results are shown in Table 1. We could do a rigorous statistical test to see if these distributions are about equal, but it looks fine just by looking at it. For example, in both data sets less than 1% of the data are first-class females who perished, 9-10% are first-class males who perished, etc. Naturally, we could look at the distribution over other variables – but since we took a simple random sample (using `PROC SURVEYSELECT` above), the two data sets should have the same distributions over *any* variables. Here we just look at three variables (survival status, class, and sex) for a quick check. This is always a good idea, to see if we see anything strange.

Now let's fit a model on our training data set and score the test data set (with the model from the training set). We're trying to predict whether a given passenger will survive the Titanic based on the passenger class, sex, age,

[2] This source data set comes with the standard JMP installation. A SAS version of this data set is included in the code repository mentioned below the abstract on the first page of this paper.

[3] This terminology is the inspiration behind the title of Silver (2012).

Counts								
	Training Set (*N*=628)				Test Set (*N*=418)			
	Did Not Survive		Survived		Did Not Survive		Survived	
Class	Female	Male	Female	Male	Female	Male	Female	Male
1	4	57	79	33	1	41	49	20
2	7	73	53	12	4	62	39	11
3	51	182	41	36	29	108	31	23

Percentages								
	Training Set (*N*=628)				Test Set (*N*=418)			
	Did Not Survive		Survived		Did Not Survive		Survived	
Class	Female	Male	Female	Male	Female	Male	Female	Male
1	0.64	9.08	12.58	5.25	0.24	9.81	11.72	4.78
2	1.11	11.62	8.44	1.91	0.96	14.83	9.33	2.63
3	8.12	28.98	6.53	5.73	6.94	25.84	7.42	5.50

Table 1: *Titanic* passengers in the training and test sets, segmented in (above) counts and (below) percentages.

and how many siblings/spouses were also on the *Titanic*. We'll do this with `PROC LOGISTIC` for convenience, but the statistical measures described in this paper could be applied to the results of any model (i.e., not just models created with `PROC LOGISTIC`).

```
PROC LOGISTIC DATA=training OUTMODEL=model;  ❶
  CLASS passenger_class sex / PARAM=ref;  ❷
  MODEL survived ( event='1' ) = passenger_class sex age siblings_and_spouses;
RUN;

PROC LOGISTIC INMODEL=model;  ❸
  SCORE DATA=test OUT=test_scored OUTROC=test_roc;  ❹
RUN;
```

We broke the modeling into two separate calls to `PROC LOGISTIC` to make sure that we're fitting the model from the training set and applying it to the test set (i.e., not fitting it from the test set). A few notes on the syntax above:

- ❶ The `OUTMODEL=` option exports the model coefficients, which we'll use at ❸.
- ❷ The `CLASS` statement recodes categorical variables `passenger_class` and `sex` as numerical variables, which is needed for `PROC LOGISTIC` to work. `PARAM=ref` tells us to use the standard parametrization so that the coefficients (not shown here but often of interest) are easily interpretable.
- ❸ We use the `INMODEL=` option to import the model from ❶.
- ❹ the `OUT=` and `OUTROC=` options generate data sets that we'll use below.

Since the emphasis of this paper is on monitoring the model performance and not on the model itself, we won't show the results of the model.

The following statistical output applies to any scored data, whether or not they were generated from `PROC LOGISTIC`.[4] Much of this information roughly follows Sayad (2012). If we're assessing models that give us something other than scores (e.g., linear/Poisson regression or time-series methods, which give us quantities rather than scores), we'd use the same basic framework (i.e., training and test sets) but different graphics and assessment measures which are not described in this paper.

[4]For generating graphics, there's a `plots=all` option for `PROC LOGISTIC` which can generate ROC charts, but it doesn't work for a `score` statement (even though the scored data set contains all the information necessary for the graphics).

CONFUSION MATRIX

A *confusion matrix* shows the numbers of correct and incorrect outcomes of a model and is used to compute model accuracy. As illustrated in Sayad (2012),

		Target (Actual Value)		
		Positive	Negative	
Model (Predicted Value)	Positive	a	b	Positive Precision = $a/(a+b)$
	Negative	c	d	Negative Precision = $d/(c+d)$
		Sensitivity	Specificity	Accuracy = $\frac{a+d}{a+b+c+d}$
		$a/(a+c)$	$d/(b+d)$	

a and d are the number of correct predictions, while b and c are the incorrect ones. The calculated values are defined as follows:

- *Positive Precision*: The proportion of positive predicted values that were correctly identified.
- *Negative Precision*: The proportion of negative predicted values that were correctly identified.
- *Sensitivity*: The proportion of positive actual values that were correctly identified.
- *Specificity*: The proportion of negative actual values that were correctly identified.
- *Accuracy*: The proportion of the predicted values that were correctly identified.

We can get these numbers from SAS via `PROC FREQ`:

```
PROC FREQ DATA=test_scored;
  TABLE f_survived*i_survived;
RUN;
```

Giving us the following:

		Target (Actual Value)		
		Survival	Non-Survival	
Model (Predicted Value)	Survival	121	52	Positive Precision = $121/(121+52)$ = **69.94%**
	Non-Survival	38	207	Negative Precision = $38/(38+207)$ = **15.51%**
		Sensitivity	Specificity	Accuracy = $\frac{121+207}{121+52+38+207}$ = **78.47%**
		$121/(121+38)$ = **76.10%**	$207/(52+207)$ = **79.92%**	

So our model correctly predicted 121 survivors and 207 non-survivors, giving us an accuracy rate of 78.47% accurate. But the sensitivity and specificity rates are also important, as sometimes a model will be better at predicting positive values (survivors) or negative ones (non-survivors). However, in this case they are about the same:

- *Sensitivity*: 76.10% of positive actual cases were correctly identified.
- *Specificity*: 79.92% of negative actual cases were correctly identified.

However, these are all assuming that cases with a score (survival probability) of 0.5 or greater will survive and those with a score of less than 0.5 will not survive. What if we changed these cutoff values? We will return to this question a little later.

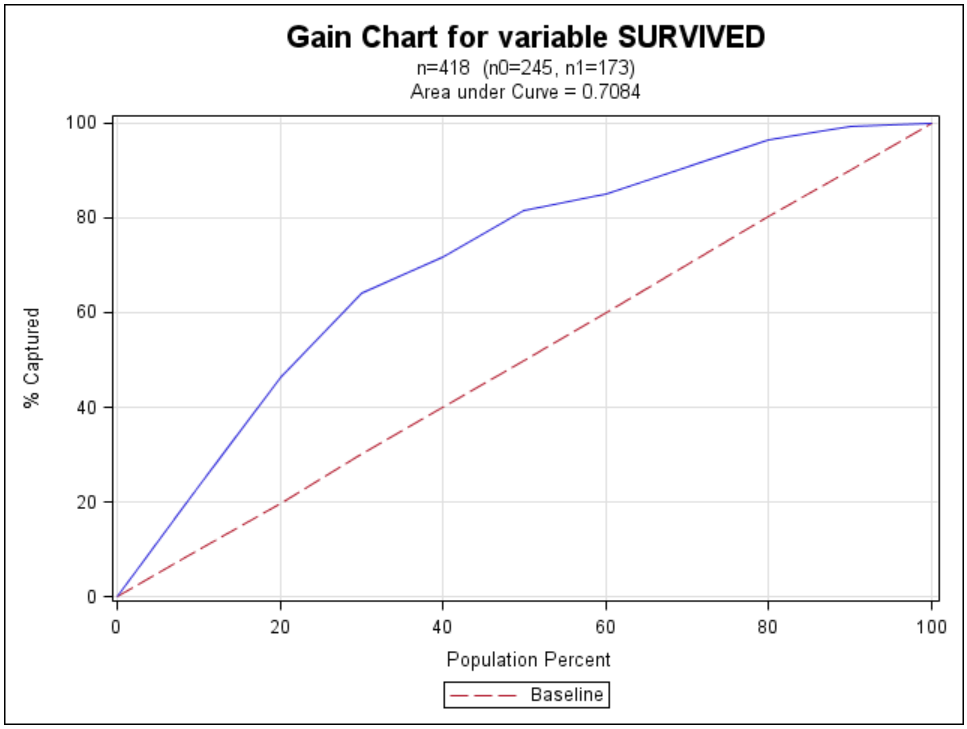


Figure 1: A gain chart.

GAIN CHARTS

A *gain chart* illustrates the effectiveness of a classification model calculated as the ratio between the results obtained with and without the model. Suppose we ordered our cases by the scores (in our case, the probability of survival on the *Titanic*).

- If we take the top 10% of our model results, what percentage of actual positive values would we get?

In our example,

- If we take the top 10% of our model results, what percentage of actual survived passengers would we get?

If we then do this for 20%, 30%, etc., and then graph them, we get a gain chart as in Figure 1.[5]

For a baseline comparison, let's now order our cases (i.e., names of *Titanic* passengers) at random. On average, if we take the top 10% of our results, we should expect to capture about 10% of our actual positive values. If we do this for all deciles, we get the straight baseline in Figure 1. If our model is any good, it should certainly be expected to do better than that! As such, the chart for our model (the solid line) should be above the dotted line.

How do we use this chart to assess our model? In general, the better our model, the steeper the solid line in our gain chart. this is commonly reflected in two statistical measurements (as well as the 40% lift, explained later):

- *The area under the curve*: The better the model, the closer the area under the curve is to 1. In our case (Figure 1), we have 70.84%.
- *The 40% measure*: What percentage of our actual targets are captured by our top 40% of predicted values? In our case (Figure 1), we have around 72%.

In practice, these measures don't mean very much unless we compare them to measures from other models. Indeed, some phenomena are easier to predict than others.

[5]We could do this for *any* percentile, but it's typically just done for deciles.

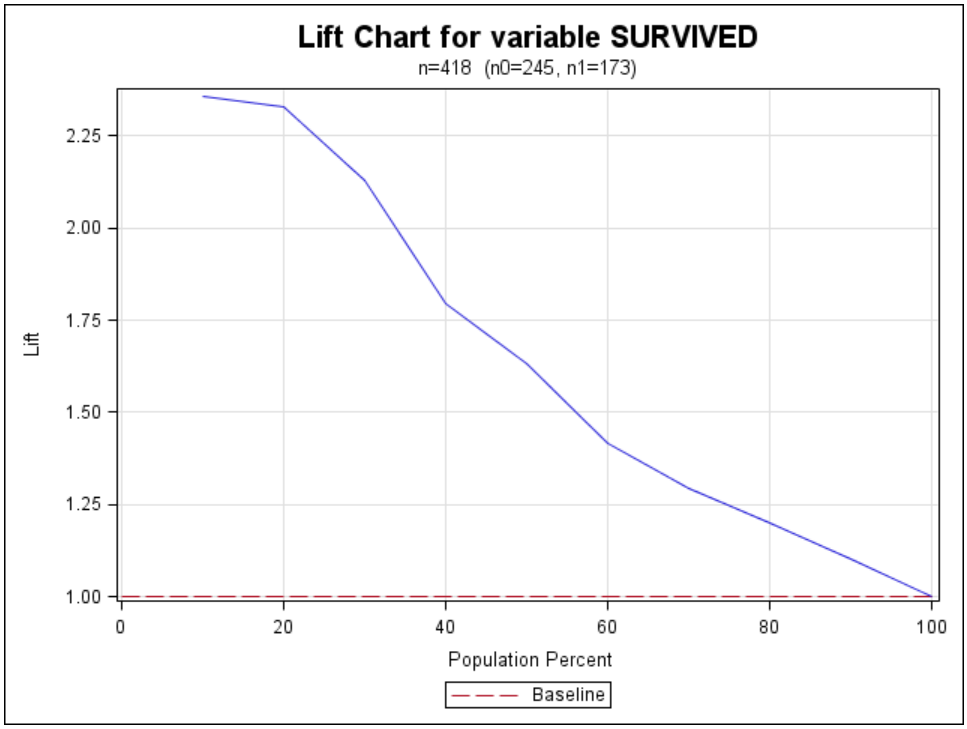


Figure 2: A lift chart.

How did we make the chart in Figure 1? SAS Institute (2011) gives us a macro that makes it relatively easy to create a gain chart. We created our own macro based on that one but creating some more details, as shown in Figure 1:

```
%makeCharts( DATA=test_scored, RESPONSE=survived, P=p_1, EVENT=1, GROUPS=10,
  PLOT=gain, OUT=gainChart, PATH=&outroot, FILENAME=Gain Chart );
```

The parameters are the following:

- `DATA`: The input data set. (Optional: default is `_last_`, the last created data set)
- `RESPONSE`: The response variable.
- `P`: The probability/score variable.
- `EVENT`: Is an event defined when the `RESPONSE` variable is 0 or 1? (Optional: default is `1`)
- `PLOT`: What graph do we want to plot? Three options (all described in this paper): `gain`, `lift` or `ks`.
- `GROUPS`: Do we want to break the data down in groups of ten (at 10% intervals) or twenty (at 5% intervals)? (Optional: default is 20)
- `OUT`: The output data set. (Optional: default is `_chart`)
- `PATH`: The path of the resulting graph (as a PNG file).
- `FILENAME`: The name of the resulting graph (as a PNG file). (Optional: default is `output`)

LIFT CHART

A *lift chart* simply looks at the ratio of the gain chart results with our model and with the baseline – i.e., the ratio of the solid line to the dotted line. This is shown in Figure 2. Using our same macro from before, we have the following syntax in SAS:

```
%makeCharts( DATA=test_scored, RESPONSE=survived, P=p_1, EVENT=1, GROUPS=10,
  PLOT=lift, PATH=&outroot, FILENAME=Lift Chart );
```

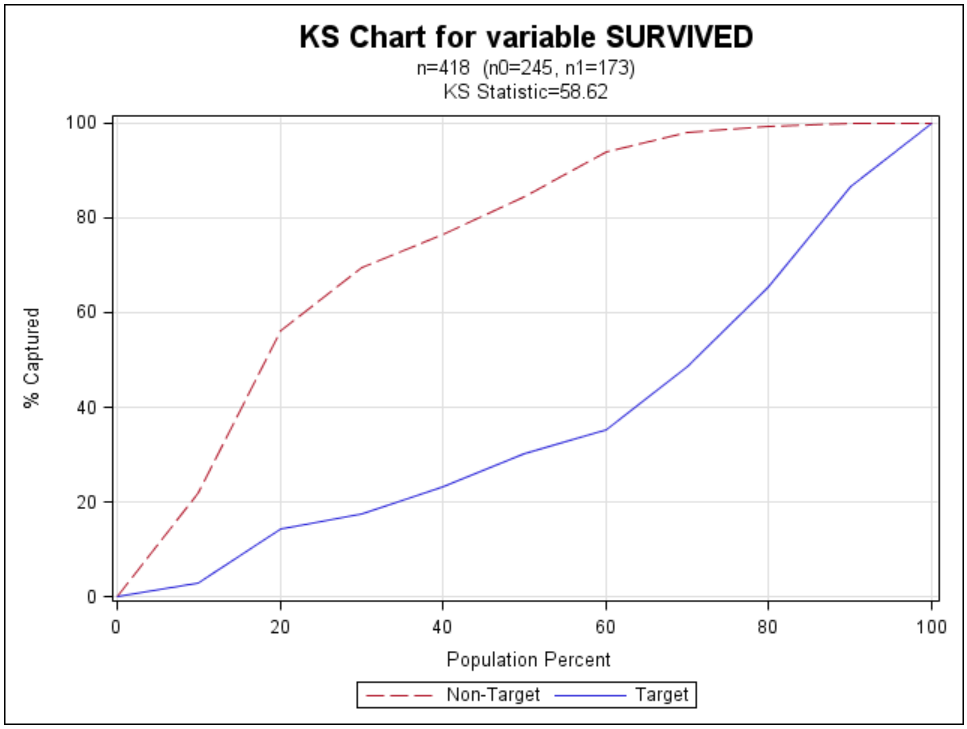


Figure 3: A K-S chart.

K-S CHART

A *K-S Chart* or *Kolmogorov-Smirnov Chart* measures the model performance in a slightly different way:

- If we look at cases with a target probability below 10%, what percentage of actual targets and non-targets would we get?

For our specific situation,

- If we look at cases with a survival probability below 10%, what percentage of actual survivors and non-survivors would we get?

We then look at this for every decile, giving us the graph in Figure 3. We have two lines: For the target (survivor) and for the non-target (non-survivor). Depending on how we defined our target (e.g., whether we model survival=1 or survival=0), the target line will be either above or below the non-target line. What's important is their maximal distance away: If we can find a probability cutoff point that maximizes the difference between the targets and non-targets, we should use that one for optimal results. This maximal distance is called the *K-S statistic* or *Kolmogorov-Smirnov statistic*, which is 58.62 for our data. The higher the K-S statistic, the better the model.

Recall that for the confusion matrix, we had a probability cutoff value set at 50%. Now we're looking at different cutoff values.

To code this in SAS (and generate Figure 3), we use our same macro from before:

```
%makeCharts( DATA=test_scored, RESPONSE=survived, P=p_1, EVENT=1, GROUPS=10,
  PLOT=ks, PATH=&outroot, FILENAME=KS Chart );
```

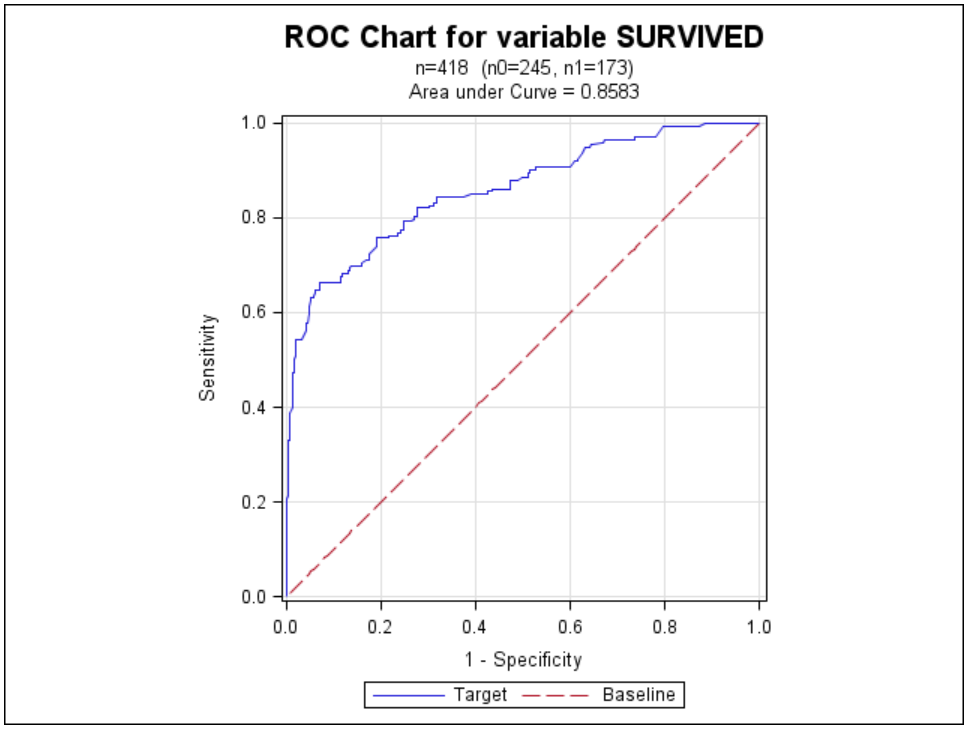


Figure 4: An ROC chart.

ROC CHART

Now we come full circle from our confusion matrix. With the K-S chart, we varied the cutoff values and looked at the percent of the targets and non-targets captured. With an *ROC chart* (*Receiver Operating Characteristic chart*), we simply look at the different values we would get in our confusion matrix:

- *Sensitivity*: The proportion of actual positive cases that were correctly identified.
- *1 - Specificity*: The proportion of actual negative cases that were incorrectly identified.

SAS has made it very easy to produce ROC charts from `PROC LOGISTIC`, but that's only possible for the training set, as it won't work when using the `INMODEL` option! Therefore, we created a macro similar to the ones before, based in part on SAS Institute (2008):

```
%makeROC( OUTROC=test_roc, OUT=test_scored, P=p_1, GRID=yes, RESPONSE=survived,
  PATH=&outroot, FILENAME=ROC Chart );
```

The result is in Figure 4.

MONITORING OVER TIME

While the statistical graphics presented in this paper are shown for assessing *one* model over *one* period in time, there clearly needs to be more of an infrastructure to determine the best of many models, track the performance of various models over time, or determine when the model must be recalibrated. This is a considerable challenge that can be built with a series of SAS macros, but that kind of solution is beyond the scope of this paper. However, SAS has a tool that does this.

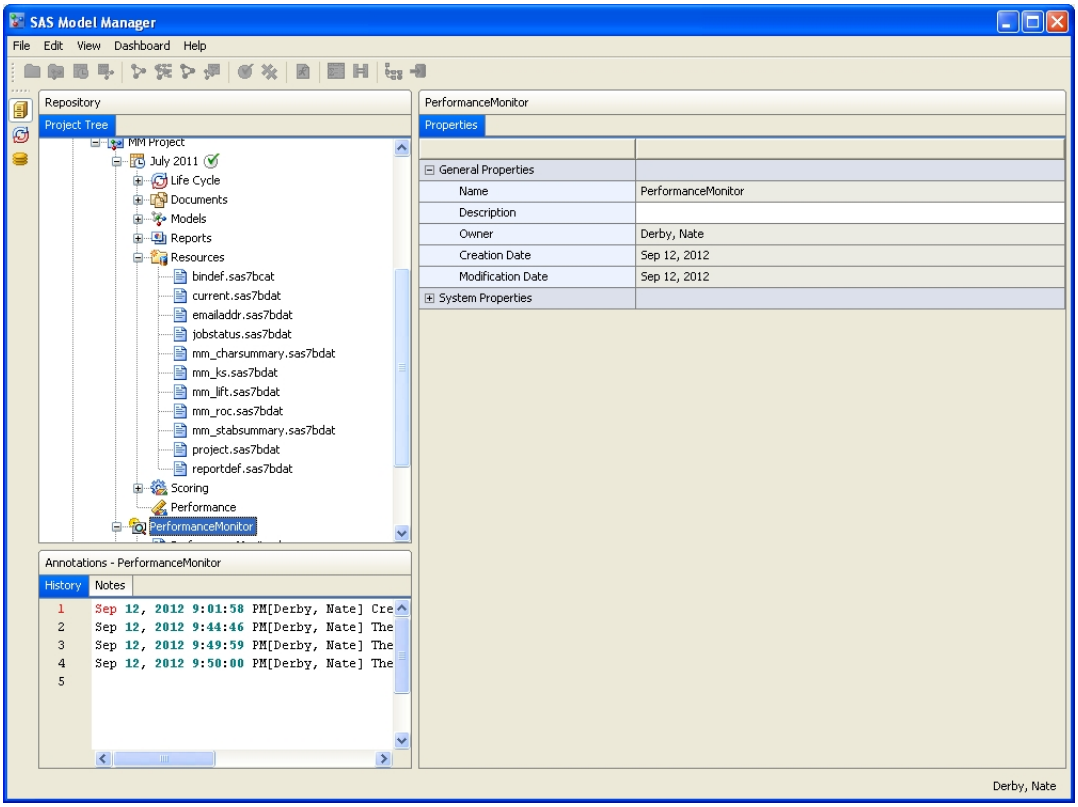


Figure 5: Model Manager interface.

MODEL MANAGER

Model Manager (Chu et al. (2007), Wei et al. (2009)) is a SAS tool that does most of the items mentioned in the introduction:

- Keeping the code organized for effective updating and maintenance.
- Maintaining effective documentation.
- Assessing and tracking model performance (including the graphs mentioned in this paper).
- Providing guidance on deciding when to update the model.

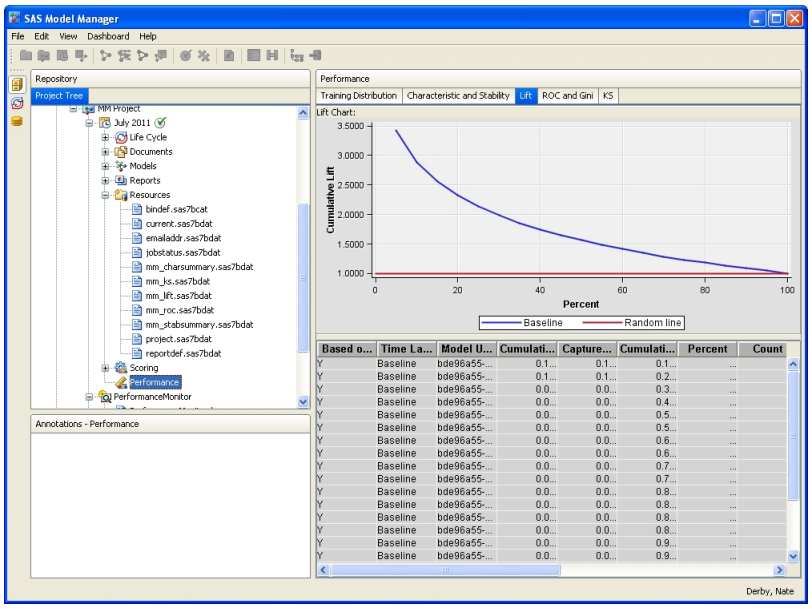
Furthermore, it does this for multiple models over multiple periods of time. However, there are some caveats:

- It's expensive.
- You still need to build and update the models.
- You still need team training/buy-in.

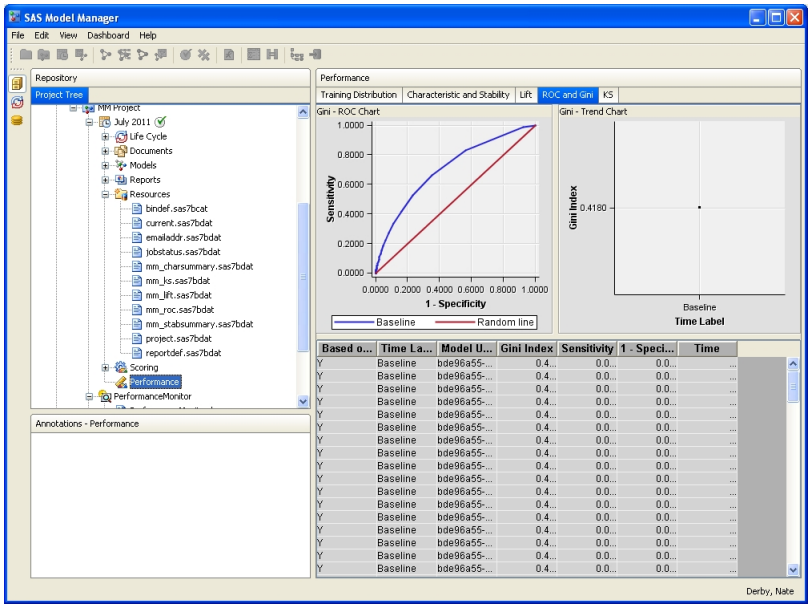
The interface in Figure 5 may look a little strange, as there isn't a code editor or a place to build models. That's intentional, as Model Manager isn't meant to build statistical models or process code. It's a *model repository*, which operates like a code repository: You check in a model (as shown by the check mark by July 2011 in Figure 6), which involves Model Manager making sure that the inputs and outputs are all correct. SAS then generates feedback about the model performance, as shown in screenshots in Figure 6.

While Model Manager is very convenient, arguably the best justification is that it forces users to maintain and update it, which is often challenging to do (even with the macros mentioned in this paper).

(a)



(b)



(c)

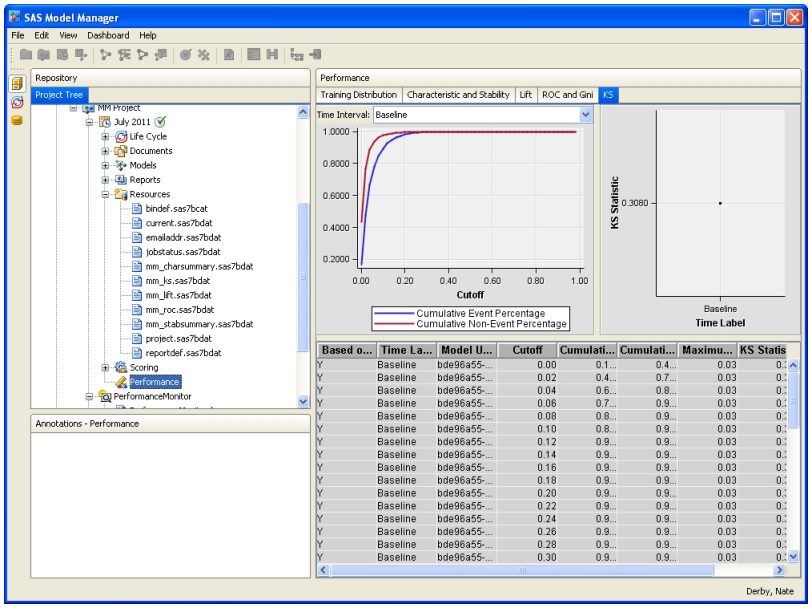


Figure 6: Output from Model Manager: (a) Lift chart, (b) Gain chart, and (c) K-S chart.

CONCLUSIONS

Managing and monitoring statistical models is challenging when there are many models used by multiple people over an extended time. However, the risks of *not* managing and monitoring them effectively can be catastrophic, as it allows statistical models to become outdated, leading to potential erroneous and costly business decisions. This is because the behavior of the data (i.e., the business behavior that we wish to model) change over time, often in ways that the statistical model wasn't built to anticipate.

Effective model management and monitoring detects when the data behavior changes and the model must be recalibrated. This challenge can be alleviated by using SAS macros or by using SAS Model Manager.

REFERENCES

Cheng, E. (2009), Better, faster, and cheaper SAS software lifecycle, *Proceedings of the 2009 SAS Global Forum*, paper 186-2009.
http://support.sas.com/resources/papers/proceedings09/186-2009.pdf

Chu, R., Duling, D. and Thompson, W. (2007), Best practices for managing predictive models in a production environment, *Proceedings of the 2007 Midwest SAS Users Group Conference*.
http://www.mwsug.org/proceedings/2007/saspres/MWSUG-2007-SAS02.pdf

Davenport, T. H. and Harris, J. G. (2007), *Competing on Analytics: The New Science of Winning*, Harvard Business School Press, Boston, MA.

Davenport, T. H., Harris, J. G. and Morison, R. (2010), *Analytics at Work: Smarter Decisions, Better Results*, Harvard Business School Press, Boston, MA.

Derby, N. (2007), Guidelines for organizing SAS project files, *Proceedings of the 2007 Western Users of SAS Software Conference*.

Derby, N. (2010), Suggestions for organizing SAS code and project files.
http://www.nderby.org/docs/DerbyN-SASOrg-cur.pdf

Dosani, F., Fecht, M. and Eckler, L. (2010), Creating easily-reusable and extensible processes: Code that thinks for itself, *Proceedings of the 2010 SAS Global Forum*, paper 004-2010.
http://support.sas.com/resources/papers/proceedings10/004-2010.pdf

Fecht, M. and Stewart, L. (2008), Are your SAS programs running you?, *Proceedings of the 2008 SAS Global Forum*, paper 164-2008.
http://www2.sas.com/proceedings/forum2008/164-2008.pdf

Jones, S. (2009), The formula that felled Wall St, *Financial Times*, Apr. 24.

Salmon, F. (2009), Recipe for disaster: The formula that killed Wall Street, *Wired*, Feb. 23.
http://www.wired.com/techbiz/it/magazine/17-03/wp_quant

SAS Institute (2008), Plot ROC curve with labelled points for a binary-response model.
http://support.sas.com/kb/25/018.html

SAS Institute (2011), Gains and lift plots for binary-response models.
http://support.sas.com/kb/41/683.html

Sayad, S. (2012), Model evaluation - Classification.
http://www.saedsayad.com/model_evaluation_c.htm

Silver, N. (2012), *The Signal and the Noise: Why So Many Predictions Fail – but Some Don't*, Penguin Press, New York.

Wei, J., Gao, E. Y., Wang, F. J. and Chu, R. (2009), Dashboard reports for predictive model management, *Proceedings of the 2009 SAS Global Forum*, paper 045-2009.
http://support.sas.com/resources/papers/proceedings09/045-2009.pdf

Winn Jr., T. J. (2004), Guidelines for coding of SAS programs, *Proceedings of the Twenty-Ninth SAS Users Group International Conference*, paper 258-29.
http://www2.sas.com/proceedings/sugi29/258-29.pdf

ACKNOWLEDGMENTS

I'd like to thank Perry Watts for her help with the graphics. She was a tremendous help.

CONTACT INFORMATION

Comments and questions are valued and encouraged. Contact the author:

Nate Derby
Stakana Analytics
815 First Ave., Suite 287
Seattle, WA 98104-1404
nderby@stakana.com
http://nderby.org
http://stakana.com