Outline

1. Introduction
   - Introduction

2. Managing Models
   - Managing Models

3. Monitoring Models
   - Preliminaries
   - Accuracy Measurements
   - Monitoring over Time
   - Model Manager

4. Conclusions
   - Conclusions
Introduction: Why Manage and Monitor Models?

Many reasons:

- Keeping all 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.
- Assessing and tracking model performance.
- Effectively/automatically deciding when to update the model.

Why are these important?
Introduction: Why Manage and Monitor Models?

Importance:

- Lost employee time/energy/productivity.
  - Bad/no documentation.
  - Using old/erroneous code.
  - Redoing old work.

- Suboptimal results $\rightarrow$ lost money.
  - If decisions are based on statistical models, those models should be accurate.

Why are these important?
Business Horror Stories

Underlying model assumptions can change:

- Migration toward smartphones, away from LAN lines.
- Reading the news from the web rather than print.
- Effects of climate change.
- Buying music/renting movies online rather than from a store.
- Car sharing rather than owning car/using public transportation.
- The *Gaussian Cupola* $\rightarrow$ 2008 financial meltdown.

Wrong assumptions $\rightarrow$ wrong results
Avoiding Disaster

How can we avoid disaster?

- Incorporate basic measures of statistical accuracy into our results.
  - Rebuild/recalibrate model when accuracy goes down.
- Can be used with dashboards.
Code/model Organization:

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

Code organization often more important than code efficiency

- The human element.
Effective model management has many parts:

- Code organization (with macros!).
- Effective documentation.
- Frequent result assessment.
- Protocol for model updates.
- Leadership, employee training and buy-in.
Monitoring Statistical Models

- Monitoring = paying constant attention to the accuracy of models to decide when to rebuild/recalebrate a model.
  - Lifecycle Management
- We’ll illustrate concepts via the Titanic.

Survivor data from the Titanic

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

- **Training Set** = data for building the model.
- **Test Set** = data for evaluating model results.

Avoids overfitting = fits peculiarities of the data (*noise*) that can distort the main results (*signal*).

**Training: 60%, Test: 40%**

```
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;
```
Do We Have a Good Partition?

Are these two distributed about the same? (We hope so!)

Checking the Distribution

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

PROC FREQ DATA=test;
   TABLES survived*passenger_class*sex;
RUN;
```
### Do We Have a Good Partition?

<table>
<thead>
<tr>
<th>Counts</th>
<th>Training Set (N=628)</th>
<th>Test Set (N=418)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Did Not Survive</td>
<td>Survived</td>
</tr>
<tr>
<td></td>
<td>Female</td>
<td>Male</td>
</tr>
<tr>
<td>Class 1</td>
<td>4</td>
<td>57</td>
</tr>
<tr>
<td>Class 2</td>
<td>7</td>
<td>73</td>
</tr>
<tr>
<td>Class 3</td>
<td>51</td>
<td>182</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Percentages</th>
<th>Training Set (N=628)</th>
<th>Test Set (N=418)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Did Not Survive</td>
<td>Survived</td>
</tr>
<tr>
<td></td>
<td>Female</td>
<td>Male</td>
</tr>
<tr>
<td>Class 1</td>
<td>0.64</td>
<td>9.08</td>
</tr>
<tr>
<td>Class 2</td>
<td>1.11</td>
<td>11.62</td>
</tr>
<tr>
<td>Class 3</td>
<td>8.12</td>
<td>28.98</td>
</tr>
</tbody>
</table>
Building a Model

Building a Statistical Model

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

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

... but the model itself isn’t important for our purposes.

- We focus on the accuracy of the results.
Confusion Matrix

Shows the number of correct and incorrect outcomes.

<table>
<thead>
<tr>
<th>Model (Predicted Value)</th>
<th>Target (Actual Value)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Positive</td>
<td>Negative</td>
<td></td>
</tr>
<tr>
<td>Positive</td>
<td>$a$</td>
<td>$b$</td>
<td>Positive Precision = $a/(a + b)$</td>
</tr>
<tr>
<td>Negative</td>
<td>$c$</td>
<td>$d$</td>
<td>Negative Precision = $d/(c + d)$</td>
</tr>
<tr>
<td>Sensitivity</td>
<td>$a/(a + c)$</td>
<td>$d/(b + d)$</td>
<td></td>
</tr>
</tbody>
</table>

Accuracy = $\frac{a + d}{a + b + c + d}$

We can get these results from `PROC FREQ`.

- **Sensitivity**: Proportion of positive actual values that were correctly identified.
- **Specificity**: Proportion of negative actual values that were correctly identified.
Shows the number of correct and incorrect outcomes.

<table>
<thead>
<tr>
<th>Model (Predicted Value)</th>
<th>Target (Actual Value)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Survival</td>
<td>Non-Survival</td>
<td></td>
</tr>
<tr>
<td>Survival</td>
<td>121</td>
<td>52</td>
<td>Positive Precision = 69.94%</td>
</tr>
<tr>
<td>Non-Survival</td>
<td>38</td>
<td>207</td>
<td>Negative Precision = 15.51%</td>
</tr>
<tr>
<td>Sensitivity</td>
<td>76.10%</td>
<td>Specificity</td>
<td>79.92%</td>
</tr>
<tr>
<td></td>
<td>Accuracy = 78.47%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

... but this assumes a cutoff of 50%. Would we get better results with different cutoff values?
If we take the top 10% of our model results, what percentage of actual positive results would we get?

For our specific situation,

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

Now do this for 10%, 20%, 30%, . . . , and graph it!

For comparison, order our cases at random.

- On average, if we take our top 10%, we should get 10% of our actual survivors.
Gain Chart for variable SURVIVED

n=418  (n0=245, n1=173)
Area under Curve = 0.7084
How Can a Gain Chart Assess our Model?

Two measures:

- **Area under the Curve**: The higher the area, the better the model.
  - Here we have 70.84%.

- **40% Measure**: What percent of our actual targets are captured by our top 40% of modeled values?
  - Here we have about 72%.

These don’t mean much unless we compare multiple models.
How Did We Get That?

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

- **DATA**: The input data set. (optional)
- **RESPONSE**: The response variable.
- **P**: The probability/score variable.
- **EVENT**: Is an event defined when the RESPONSE variable is 0 or 1? (optional)
- **PLOT**: What graph do we want to plot? (gain, lift or ks)
- **GROUPS**: Do we want to break the data down in groups of 10 or 20? (optional)
- **OUT**: The output data set. (optional)
- **PATH**: The path of the resulting graph.
- **FILENAME**: The name of the resulting graph (as a PNG file). (optional)
Lift chart = ratio of gain chart model results over baseline.

- Ratio of the solid line over dotted line.

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

Lift Chart for variable SURVIVED

n=418  (n0=245, n1=173)
Gain Chart for variable SURVIVED

n=418 (n0=245, n1=173)
Area under Curve = 0.7084
K-S Chart

K-S chart = Kolmogorov-Smirnov chart:

- 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?

Now do this for 10%, 20%, 30%, ..., and graph it!

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

KS Chart for variable SURVIVED

n=418  (n0=245, n1=173)
KS Statistic=58.62
How Can a K-S Chart Assess our Model?

*K-S Statistic* = maximal distance between two curves.

- That’s where the probability cutoff maximizes the difference between non-targets and targets.
- The higher, the better.
- Here we have 58.62%.

This doesn’t mean much unless we compare multiple models.
**ROC Chart**

**ROC Chart = Receiver Operating Characteristic Chart = Confusion Matrix measures for different cutoff values:**

- **Sensitivity**: Proportion of positive actual values that were correctly identified.
- **1-Specificity**: Proportion of negative actual values that were incorrectly identified.

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

ROC Chart for variable SURVIVED
n=418 (n0=245, n1=173)
Area under Curve = 0.8583
How Can an ROC Chart Assess our Model?

*Area under the Curve* = the probability that our model will rank a positive actual case higher than a negative one.

- The higher, the better!
- Here we have 85.83%.

This doesn’t mean much unless we compare multiple models.
These measures assess *one* model over *one* period of time.

There needs to be an infrastructure to track performance of multiple models over time.

Can be built by a series of macros outside of this presentation. Or ...
Model Manager

**Model Manager** = SAS tool that does most of this stuff:
- Keep the code organized.
- Maintain effective documentation.
- Assess/track model performance.
- Provide guidance on deciding when to update the model.

But . . .
- It’s expensive.
- You still need to build/update the models.
- You still need team training/buy-in.
Not your typical SAS tool:

- No code editor, no model building.
- It’s a *model repository*: Check models in and out.
- Check-in: Makes sure all inputs, etc. are correct.
- Tracks model performance over time.

Good justification: Forces users to maintain/update it!
Managing/monitoring statistical models is challenging.

The risks of *not* managing/monitoring them effectively can be catastrophic.

- Models can become outdated → erroneous/costly business decisions.
- Behavior of the data changes over time → model needs to adjust!

Effective model management/monitoring detects when data behavior changes and the model must be rebuilt.

This challenge can be alleviated by using SAS macros or using SAS Model Manager.
Too many to list – see the paper!

Nate Derby: ndery@stakana.com