MWSUG 2018 - Paper AA-121

How to Score Big with SAS® Solutions: Various Ways to Score New Data with Trained Models

Scott Koval, Pinnacle Solutions, Inc., Indianapolis, IN

ABSTRACT

After training a statistical model, the next step is to put it into production in order to score new data. While it might be tempting to manually write code to score data, this can lead to problems with precision, complexity, and updating. SAS solutions offer a wide variety of methods to do this. This paper covers several common techniques, including PROC SCORE, the CODE and STORE statements, and PROC ASTORE. By learning these approaches, SAS can easily put complex models to work.

INTRODUCTION

SAS has always been a powerful tool used by analysts to create statistical models that help detect and explain patterns that emerge within data. After training a series of models and reviewing the results, the user then selects a champion model. This champion model is usually then deployed to score incoming new data, offering predictions for future outcomes.

While analysts and programmers may be tempted to just write code based on the results to score new data, there are several problems to this approach. The first problem comes with precision—rounding or truncating numeric parameters that make up a model may lead to potential discrepancies in the scored value. Another possible problem with coding the model comes from its complexity. Manually coding a model may be easier for some types of models, such as regressions. This becomes much more difficult when trying to incorporate models created through newer techniques, such as random forests, neural networks, or gradient boosting. The final problem with this approach involves maintaining and updating the code when a new model becomes champion.

The purpose of this paper is to review some of the alternative methods to scoring data. As with many things in SAS, there are multiple ways of doing so. By knowing what options are available, an analyst can select the best method suited to meet their organization's need.

PROC SCORE

A traditional way of using a model to score new data involves saving the parameter table that is produced during the training phase. An example of this technique involves using the CARS table from the SASHELP library. A simple regression model is produced and the parameter estimates are stored in a table called MODEL by using the outest option in the PROC REG statement. The code used to do this is found below:

```
proc reg data=sashelp.cars outest=model;
    PredMSRP: model msrp = horsepower mpg_city weight;
run;
```

The parameter estimate values that are found in this table (See Table 1) can be stored in a permanent library for further use.

MODEL	_TYPE_	_DEPVAR_	_RMSE_	Intercept	Horsepower	MPG_City	Weight	MSRP
PredMSRP	PARMS	MSRP	10707.98151	-26293.8752	256.0674687	454.7604638	-1.49117	-1

Table 1. Parameter Estimates for Predicting Car MSRP

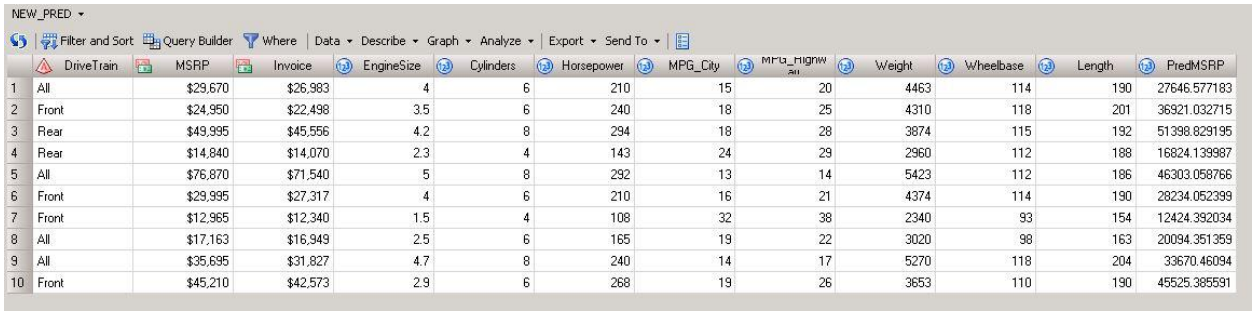
The _MODEL_ and _TYPE_ variable refer to the name of the predicted variable created in the regression procedure, as well as the type of information that is stored within it. This table also contains the name of the dependent variable, as well as the estimated values for the intercept and independent variables. Now that this table is available, it can be used to create predicted values for new data.

In this example, a random sample of rows were taken from the CARS dataset to score. The SCORE procedure is then used to apply the information stored in the MODEL table and create a predicted MSRP for the sample data.

```
proc surveyselect data=sashelp.cars out=to_score(drop=numberhits) n=10
      m=urs seed=12345;
run;

proc score data=to_score score=model type=parms predict out=new_pred;
    var horsepower mpg_city weight;
run;
```

The table NEW_PRED now contains an additional column in it with the predicted MSRP (See Figure 1).



NEW_PRED

Filter and Sort | Query Builder | Where | Data | Describe | Graph | Analyze | Export | Send To

	DriveTrain	MSRP	Invoice	EngineSize	Cylinders	Horsepower	MPG_City	MPG_Highway	Weight	Wheelbase	Length	PredMSRP
1	All	$29,670	$26,983	4	6	210	15	20	4463	114	190	27646.577183
2	Front	$24,950	$22,498	3.5	6	240	18	25	4310	118	201	36921.032715
3	Rear	$49,995	$45,556	4.2	8	294	18	28	3874	115	192	51398.829195
4	Rear	$14,840	$14,070	2.3	4	143	24	29	2960	112	188	16824.139987
5	All	$76,870	$71,540	5	8	292	13	14	5423	112	186	46303.058766
6	Front	$29,995	$27,317	4	6	210	16	21	4374	114	190	28234.052399
7	Front	$12,965	$12,340	1.5	4	108	32	38	2340	93	154	12424.392034
8	All	$17,163	$16,949	2.5	6	165	19	22	3020	98	163	20094.351359
9	All	$35,695	$31,827	4.7	8	240	14	17	5270	118	204	33670.46094
10	Front	$45,210	$42,573	2.9	6	268	19	26	3653	110	190	45525.385591

Figure 1. Screenshot of Scored Data

While this scoring method works, it is limited to modeling techniques that use coefficients to produce a scored variable. Because of this, it might be necessary to use a different technique based on the type of modeling produced.

THE CODE STATEMENT

A second method for deploying a model to score data involves using the CODE statement in the modeling procedure. When used, SAS will automatically generate a program that can be referenced in a data step to score data. This is very similar to the syntax that is created using a Score node in SAS Enterprise Miner, or requesting score code from a model built using SAS Visual Analytics or SAS Visual Statistics.

Building off the previous example, the syntax of the procedure does not change much:

```
%let _Path = C:\SAS Programs\MSRP_ScoreCode.sas;
proc reg data=sashelp.cars;
    model msrp = horsepower mpg_city weight;
    code file="&_Path";
run;
```

It is useful, but not necessary to use a macro variable to point to the location to save the generated scoring code. In this example, the main difference is not having to produce a table containing parameters, but instead adding the code statement. The file option tells the procedure where to score the SAS scoring code. The code that is produced is as follows:

```
*****************************************;
** SAS Scoring Code for PROC REG;
*****************************************;

label P_MSRP = 'Predicted: MSRP' ;
drop _LMR_BAD;
_LMR_BAD=0;

*** Check interval variables for missing values;
if nmiss(Horsepower,MPG_City,Weight) then do;
   _LMR_BAD=1;
   goto _SKIP_000;
end;

*** Compute Linear Predictors;
drop _LP0;
_LP0 = 0;

*** Effect: Horsepower;
_LP0 = _LP0 + (256.06746867158) * Horsepower;
*** Effect: MPG_City;
_LP0 = _LP0 + (454.760463827586) * MPG_City;
*** Effect: Weight;
_LP0 = _LP0 + (-1.49117700009139) * Weight;

*** Predicted values;
_LP0 = _LP0 +    -26293.8752444884;
_SKIP_000:
if _LMR_BAD=1 then do;
   P_MSRP = .;
end;
else do;
   P_MSRP = _LP0;
end;
```

The code that is created contains the formula for computing the MSRP prediction based off the coefficients produced by the trained model. In order to apply this SAS program to score new data, all one has to do is place it within a data step:

```
data new_pred;
    set to_score;
    %include "&_Path";
run;
```

In this example, the %INCLUDE statement was used alongside the macro variable that contains the value for the score code file's path. There is an advantage to using this technique over simply just cutting and pasting the score syntax into the data step. In case the model is updated and the model has changed, there is no need to change the code—it is already being evoked using the %INCLUDE statement.

THE STORE STATEMENT

The final method to scoring data involves storing information about a model within a binary file. This file contains the results of the modeling procedure that can then be applied to new data using PROC PLM. Implementing this technique is simple as it only requires the addition of a STORE statement within the modeling procedure:

```
proc reg data=sashelp.cars;
    model msrp = horsepower mpg_city weight;
    store work.MSRP_Model;
run;
```

When this syntax is run, the binary file will be saved to a SAS library. It might be useful to save this to a permanent location, rather than a temporary one used in this example. When the model has been created and stored, the PLM procedure can be used to apply the modeling results to new data:

```
proc plm restore=work.MSRP_Model;
    score data=to_score out=new_pred;
run;
```

The RESTORE option in the PLM procedure points to the binary file that was produced in the previous modeling procedure. The SCORE statement specifies the data set that needs to be scored, as well as the name of the table that will contain the scored data.

It is interesting to note that the CODE statement can also be placed within the PLM procedure to generate the SAS scoring code from the model:

```
proc plm restore=work.MSRP_Model;
    code file="&_Path";
run;
```

PROC ASTORE

SAS Viya is a powerful analytic platform that makes use of SAS Cloud Analytic Services (CAS) in order to provide data mining and machine learning procedures. This includes tasks involved in both supervised and unsupervised learning (SAS Institute, Inc., 2018). Complex models are created and stored on the CAS server by using the SAVESTATE statement. This is very similar to the STORE statement, in that is produces a binary file that contains the modeling information after a model has been trained.

Using the SASHELP.CARS data, this technique can be demonstrated. The first step is to establish the user's CAS session and load the training data into a CAS library:

```
libname mycaslib cas caslib=casuser;

caslib _all_ assign;

data mycaslib.cars replace;
     set sashelp.cars;
run;
```

Once this data is loaded, a data mining or machine learning procedure can be trained to create a model. For this example, the FOREST procedure is used to build a model that once again predicts the MSRP of a car:

```
proc forest data=casuser.cars;
      target MSRP / level=interval;
      input EngineSize Cylinders Horsepower MPG_City MPG_Highway Weight
            Wheelbase Length / level=interval;
      input Make Type Origin DriveTrain / level=nominal;
      autotune tuningparameters=(ntrees maxdepth inbagfraction
            vars_to_try(init=12)) objective=ase fraction=0.3;
      savestate rstore=casuser.MSRP_Forest_Model;
run;
```

The SAVESTATE statement is used to create the binary file that contains the modeling information in the CASUSER library. This file can now be applied to score new data. The randomly selected rows from the CARS data set are loaded on to the CAS server to be scored using the ASTORE procedure.

```
proc surveyselect data=sashelp.cars
out=casuser.to_score(drop=numberhits) n=10 m=urs seed=12345;
run;

proc astore;
      score data=casuser.to_score out=casuser.new_pred
            rstore=casuser.MSRP_Forest_Model;
quit;
```

The SCORE statement in the ASTORE procedure specifies the data that needs scoring, the scored table that is produced, and the stored model that is being applied. The resulting table contains the predicted MSRP values that can then be merged back to the data (See Figure 2).

	P_MSRP	_WARN_
1	31004.85098	
2	27837.672631	
3	61531.378047	
4	18017.020382	
5	63540.157798	
6	30326.793518	
7	12680.812914	
8	18565.398351	
9	38912.027943	
10	42407.052505	

Figure 2. Scored Results using PROC ASTORE

CONCLUSION

SAS offers a wide variety of analytical solutions that allows users to build robust models and make predictions. After selecting a champion model, it is simple to deploy it to a production-level environment and put it to use to score new data. There are a variety of techniques to use in order to score new, incoming data. By implementing the techniques demonstrated in this paper, developers can easily put their models to work.

REFERENCES

SAS Institute Inc. “SAS® 9.4 and SAS® Viya® 3.4 Programming Documentation / SAS Visual Data Mining and Machine Learning 8.3: Procedures.” Accessed August 23, 2018. https://documentation.sas.com.

ACKNOWLEDGMENTS

I would love to thank my colleagues at Pinnacle Solutions, Inc. for their support and encouragement.

RECOMMENDED READING

- *Techniques for scoring a regression model in SAS by Rick Wicklin*

CONTACT INFORMATION

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

Scott Koval
Pinnacle Solutions, Inc.
(317) 423-9143
scott.koval@thepinnaclesolutions.com
https://thepinnaclesolutions.com/