Paper SAS537-2017

Bringing Real-Time Scoring to Your SAS® Visual Analytics Dashboards with SAS® Visual Statistics Score Code

Eli Kovick, SAS Institute Inc., Cary, NC

ABSTRACT

Whether you are calculating a credit risk, a health risk, or something entirely different, there is a need for instant, on-the-fly risk score calculation across multiple industries. This paper demonstrates how you can produce individualized risk scores through interactive dashboards. Your risk scores are backed by powerful SAS® analytics because they leverage score code that you produce in SAS® Visual Statistics. Advanced topics, including the use of calculated items and parameters in your dashboards, as well as how to develop SAS® Stored Processes capable of accepting parameters that are passed through your SAS® Visual Analytics Dashboard are covered in detail.

INTRODUCTION

Using score code from an analytical model is easy if you are a SAS® programmer. However, coders and statisticians are not typically charged with making decisions based on analytical output. To put the power of the analytical model into the hands of the people who need it, you must operationalize the score code. We will discuss two approaches for operationalizing score code in SAS® Visual Analytics. First, we will look at a technique that involves scoring an individual observation based on values supplied by a user of a SAS® Visual Analytics Dashboard. Next, we will outline a process for uploading an entire table of observations into a SAS Visual Analytics Dashboard for bulk scoring. To follow along, you will need to be familiar with the SAS programming language, parameterized SAS® Stored Processes, and SAS Visual Analytics development. In each approach, we will be using score code produced by a SAS® Visual Statistics logistic regression.

INTERACTIVE SCORING – END USER SUPPLIES THE VALUES

Our goal is to develop a dashboard that allows a user to enter values that are used by a scoring algorithm to produce a risk score. There are many moving parts, and several steps must be followed to complete our task. I'll first outline each of the steps below, and then we will walk through each step in turn. Here are our steps:

1. Extract score code from a SAS Visual Statistics model.
2. Develop a stored process that feeds data to the score code.
3. Create data that contains parameter values for our model and move this data into SAS Visual Analytics.
4. Create a SAS Visual Analytics Dashboard capable of passing parameters to our stored process.

EXTRACT SCORE CODE

After creating a predictive model in SAS Visual Statistics, you can extract the SAS score code with just a few clicks. Figure 1 shows the SAS Visual Statistics interface with the option for exporting score code selected.

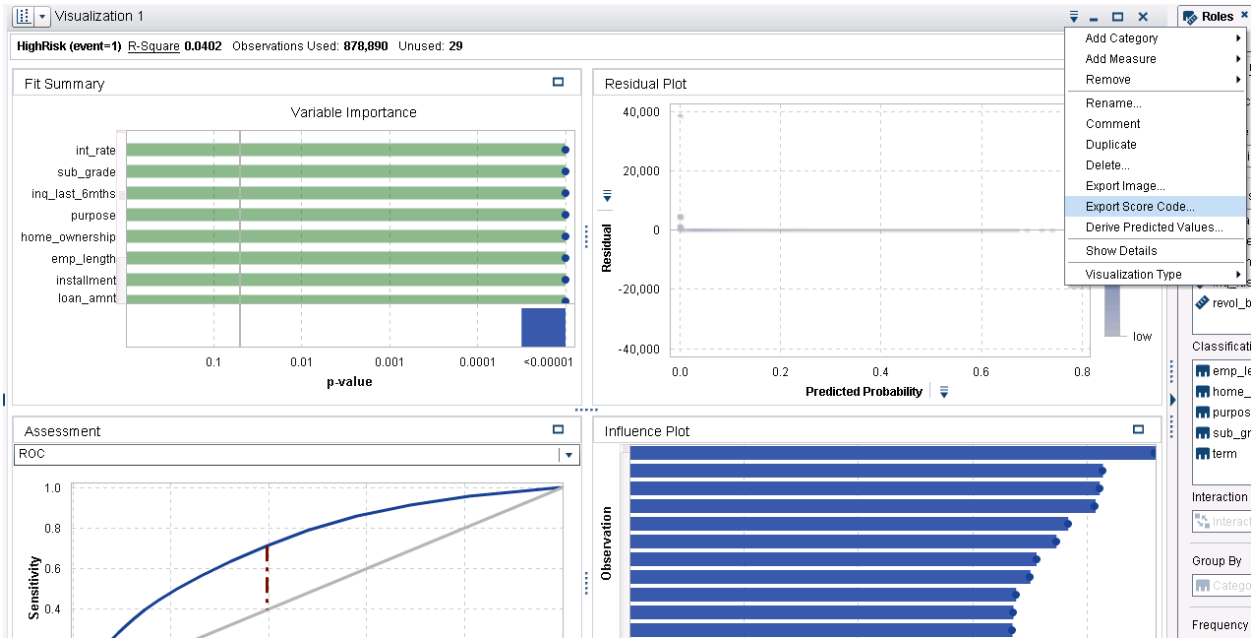


Figure 1. Export Score Code

DEVELOP STORED PROCESS

SAS Visual Analytics enables you to pass parameter values to a SAS Stored Process. We are going to take advantage of this capability by passing values we specify in our dashboard to a SAS Stored Process that contains our scoring algorithm. Before we get into the details of the stored process, it is important to understand how you would use the score code in a typical SAS session. You simply wrap your score code in a DATA step:

```
data scored_loan_data;
  set mydata.loans;

 /* SAS Score Code Generated by SAS Visual Statistics goes here */
run;
```

This code example assumes that you have a data set named “loans” in a library named “mydata”. It further assumes that your “loans” table contains the same predictor variables your score code is expecting. Running this code will produce a table named “scored_loan_data” in your SAS WORK library. The table will be identical to the “loans” table except that it will also contain a risk score for each observation under the new column “P_HighRisk”.

In this example, the data we scored already exists. With our stored process, we are going to include code that creates the table we want to score on-the-fly. The values for each variable in our table will be determined by the values that were read in by the SAS Visual Analytics user. We will use macro variables in a DATA step to create our input table. Here is an example:

```
data WORK.Single Loan Applicant;
     Loan ID=&Loan_ID;
     HighRisk="0";
     loan amnt=&loan amt;
     int rate=&int rate;
     installment=&installment;
     inq last 6mths=&inq6;
     revol bal=&revolving;
     term="&term";
     sub grade="&sub grade";
     emp length="&emp length";
     home ownership="&home_ownership";
     purpose="&Purpose";
run;
```

When the user supplies values from the SAS Visual Analytics report, the macro variables will resolve and a table named "Single_Loan_Applicant" will be generated. There are two variables in the table that I want to draw your attention to. The first is the "Loan_ID" variable. This variable was not used by our model and is not absolutely necessary. We include this variable as a way of distinguishing individual observations. The next variable is "HighRisk". Without this variable in our table, our score code will not run. You can set the value to "0" or "1". It will not affect the risk score.

The next step in our stored process is to score the table we just created. We will use the same technique we looked at previously and we will wrap our score code in a DATA step:

```
data Single Loan Scored;
     set WORK.Single_Loan_Applicant;

/* SAS Score Code Generated by SAS Visual Statistics goes here */

run;
```

After scoring our "Single_Loan_Applicant" table, we will have a risk score in our "Single_Loan_Scored" table. Once again, our risk score is located in the new column "P_HighRisk". From here, we need a way to display the result to the SAS Visual Analytics user. A simple PROC PRINT step would do the trick, but it would be a bit clunky. Instead, we're going to use SAS code to build HTML output. The final portion of our SAS code pulls the score from the "P_HighRisk" column and displays it in HTML:

```
data  null ;
  set Single Loan Scored;
  PctHighRisk = PUT (P HighRisk, PERCENT7.3);
  call symput("myscore",pctHighRisk);
 run;


data  null ;
file  webout;
put "<!DOCTYPE html5>";
put "<html>";
```

```
put "<head>";
put "<style>";
put "body {";
put "    background-color: Silver;";
put "}";
put "</style>";
put "</head>";
put "<body>";
put "<center>";
put "<h1>The Risk Score is: &myscore</h1>";
put "</center>";
put "</body>";
put "</html5>";
run;
```

We are now done with the code portion of our project. The next step is to register our code as a stored process using SAS® Enterprise Guide, or SAS® Management Console. When we register the stored process, we need to create input prompts for each of the macro variables we defined in our first DATA step. Measures should be registered with a prompt type of "Numeric" where the user enters values. Categories should be registered with the Prompt Type "Text" where the user selects values from a static list. For more information about registering a stored process, see the *SAS 9.4 Stored Processes: Developer's Guide.*

CREATE PARAMETER DATA

In our SAS Visual Analytics Dashboard we are going to use text input boxes and drop-down lists to make selections that will populate a list table with a single row of data. The values we populate in the table are the input values we want to pass to our stored process for scoring. We use text input boxes for numeric variables, and drop-down lists for our category variables. Using drop-down lists with valid selections ensures that we're passing valid data to our stored process. In SAS Visual Analytics you populate the values for a drop-down list with data you loaded into the SAS® Visual Analytics Designer. We will need to load a table to the SAS® LASR Analytic Server that contains all of our available category variables. You can use your preferred method to load this table. In this example, we will name this table "Parameters". Our next challenge is to build a dashboard capable of passing vales to our stored process. Figure 2 shows an example of how your dashboard might look.

CREATE SAS VISUAL ANALYTICS DASHBOARD

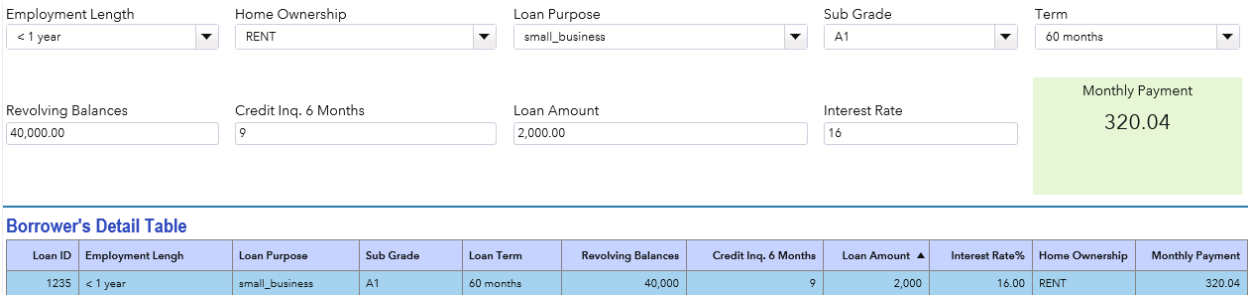


Figure 2. Interactive SAS Visual Analytics Dashboard

In the example, we are using drop-down lists and text input boxes to populate values in a list table. The variables assigned to the roles in the list table are calculated items where the value of the calculation is equal to the value of the parameter. Once our selections are made, we can click the list table to pass all

of the values to our stored process for scoring. The risk score will be displayed in a new browser tab. To build our dashboard, we will need to complete the following steps in SAS Visual Analytics Designer:

1. **Build the skeleton of our dashboard** by dragging over enough text input boxes for all of the measures, and enough drop-down lists for all of the categorical variables that feed our model. We will also need to include a list table in our dashboard.
2. **Define parameters** for each of the variables that feeds our stored process. It is a good idea to give our parameter the same name as its corresponding variable.
3. **Create calculated items** for all of the variables that feed our stored process. The value of the calculation is equal to the parameter we created for the same variable. To create this calculation, we simply drag our parameter into the empty calculation box. For an illustration of this part of the step, refer to Figure 3. In Figure 3 we are creating a calculated item for Loan Amount where the value of the calculation is equal to the value we give our parameter.

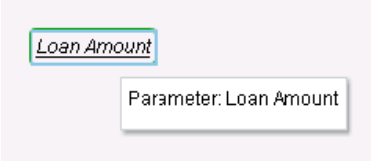


Figure 3. Example Calculation

4. **Assign data** to the list table, text input, and drop-down list objects. For all of our drop-down lists and text input boxes we will assign the parameters we defined in step 2. Our numeric parameters are assigned to text input boxes, and categorical parameters are assigned to drop-down lists. While assigning the parameters to the drop-down lists, we will use the Roles tab to assign the corresponding variable from the "Parameters" table to the "Category" Role. Figure 4 Illustrates this step.

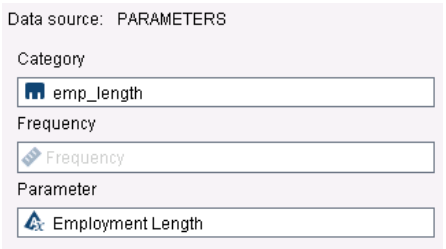


Figure 4. Example Drop-Down List

Next, we will populate the list table with the calculated items we defined in Step 3. Once this is finished, we should be able to read in text and make selections in our drop-down lists to dynamically alter the values in the list table. The final step in our design process involves passing the values in the list table to our stored process for scoring.

Link the list table to the stored process. We accomplish this by right-clicking the list table and selecting "Add Link > External Link". Then, we choose the option to "Link to Stored Process". Next, we navigate to the stored process we previously created. SAS Visual Analytics will automatically detect the parameters we used in the DATA step. A box will be displayed asking us to map the values in the list table to the parameter values in our stored process. We will map the calculated items in our list table to the corresponding target macro variables our stored process.

Figure 5 illustrates how we set up the mapping.

Figure 5. Mapping Values in the List Table to the Stored Process

After linking our list table to the stored process, we should have a functioning interface for producing on on-the-fly risk scores. We should ensure that everything is working correctly and double-click the list table in the dashboard. If we've done everything correctly, a new tab will open with a risk score. We should try making changes to the text input and drop-down lists to see how the risk score is affected.

UPLOAD AND SCORE A CSV TABLE

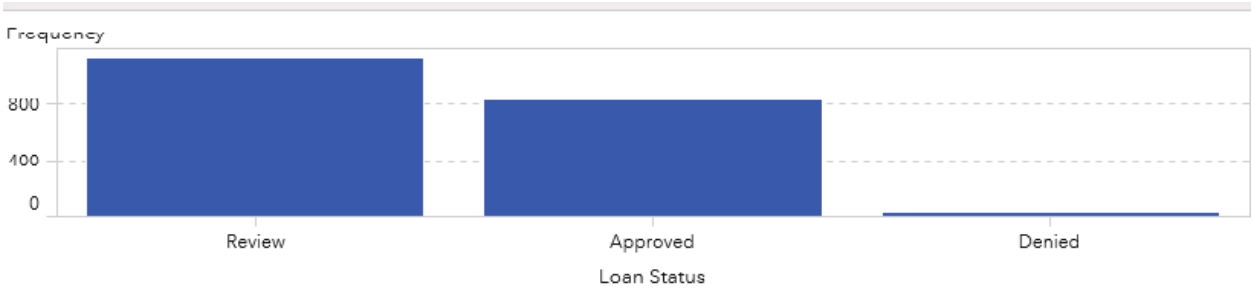
In the previous sections of this paper we focused on developing an interface to produce one risk score at a time. In this next section, we will look at how to create an interface in SAS Visual Analytics that allows you to upload and score an entire table, and then immediately view the results in your dashboard. This time, our task involves the creation of two SAS Stored Processes. We will call the first stored process "Upload_Form_STP". This stored process will be surfaced as a report object in SAS Visual Analytics Designer. Its purpose will be to surface an HTML form with a "browse" button for uploading data. The second stored process we create will be called "Score_Table". This stored process will pull the table from the first stored process, and then score it with our SAS Visual Statistics score code. The final step in this stored process will be a SAS DATA step that places the table into memory allowing it to be viewed in our SAS Visual Analytics Dashboard.

Once again, there are many moving parts, and several steps that must be followed to complete our task. I'll first outline each of the steps below, and then we will walk through each step in turn. Here are our steps:

1. Upload a scored data set into the LASR Analytic Server and build a report.
2. Develop a stored process that builds an HTML form for uploading data.
3. Develop a stored process that accepts data from the first stored process and scores it.
4. Add the stored process object to our dashboard.

UPLOAD SCORED DATA SET INTO THE LASR ANALYTIC SERVER – BUILD REPORT

After our table is scored, we want immediate access to the results. To enable this, we need to build a report "template" from some already-scored data. For a refresher on how to score data with SAS Visual Statistics score code, scroll back up to the top of page 2. Once our data is scored, uploaded, and available in SAS Visual Analytics Designer, we can build a report. Figure 6 provides an example report. In the example, custom categories based on risk score are used to classify whether a loan has a status of Approved, Denied, or Review.



LoanID	purpose	Risk of Default	Loan Status
1052616	car	12.13%	Review
1043030	car	1.03%	Approved
1059497	car	4.74%	Approved
1051325	car	11.99%	Review
1047639	car	8.07%	Review
1047667	car	1.10%	Approved

Figure 6. Example Report for Scored Data

DEVELOP STORED PROCESS THAT BUILDS HTML FORM

With only a few modifications, the code sample below is all that is needed to build our first stored process "Upload_Form_STP". This stored process surfaces an HTML form with a browse button that allows us to upload a table and then pass that table to our second stored process for scoring. We will be writing the file to _webout. Notice green highlighting below. Using this code is how we call the stored process that does the scoring. Our next step will be to build the "Score_Table" stored process. When we move this code into a new environment, we'll need to modify the http server name as well as the path to the stored

process that does the scoring. Once those modifications are made, we will register our code as a stored process in either SAS Enterprise Guide or SAS Management Console.

```
data  null ;
  format infile $char256.;
  input;
  infile = resolve(_infile_);
  file  webout;
  put infile;
cards4;
<html5>
<BODY>
<form action="http://sasva.demo.sas.com/SASStoredProcess/do"
enctype="multipart/form-data" method="post">
<input type="hidden" name="_program" value="/Shared
Data/SP/BulkLoanScore">
<th>Choose a file to upload:</th>
<td><input type="file" name="myfile"></td>
<td colspan="2" align="right"><input type="submit" value="SUBMIT"
onclick="msg()"></td>

</form>
</html5>

;;;;
;;;;
run;
```

DEVELOP STORED PROCESS THAT SCORES AND UPLOADS THE TABLE TO MEMORY

Our second stored process, “Score_Table”, has a few responsibilities. It pulls the table from the first stored process using the _WEBIN file reference. Next, it uses INFILE and DATA step statements to read in the CSV file and produce a SAS table. After the SAS table is created, our score code wrapped in a DATA step scores the data, produces a scored data set, and finally, uploads that data set into memory. The various steps in the code are noted in the comments:

```
/* Step 1. Read in CSV file and generate SAS data set */

%let csv_FILE=%sysfunc(pathname(&_WEBIN_FILEREF));


DATA WORK.Loan_Applicants;
 LENGTH
        Loan_ID            8
        HighRisk           8
        loan_amnt          8
        int_rate           8
        installment        8
        inq_last_6mths     8
```

```
        revol bal           8
        term              $ 9
        sub grade         $ 2
        emp length        $ 9
        home ownership    $ 8
        purpose           $ 18 ;

    INFILE "&csv_FILE"
        LRECL=91
        ENCODING="WLATIN1"
        TERMSTR=CRLF
        DLM=','
        MISSOVER
        DSD
           firstobs=2;
     INPUT
        Loan ID           : ?? BEST8.
        HighRisk          : ?? BEST1.
        loan amnt         : ?? BEST5.
        int rate          : ?? COMMA5.
        installment       : ?? COMMA7.
        inq last 6mths    : ?? BEST2.
        revol bal         : ?? BEST7.
        term              : $CHAR9.
        sub grade         : $CHAR2.
        emp length        : $CHAR9.
        home ownership    : $CHAR8.
        purpose           : $CHAR18. ;

RUN;

/* Step 2. Score SAS LOAN_APPLICANTS data set */


data Loan Applicants Scored;
  set work.loan_applicants;
 /* SAS Score Code generated by SAS Visual Statistics goes here */

RUN;

/* Step 3. Replace table in memory */

LIBNAME VALIBLA SASIOLA  TAG=hps  PORT=10010 HOST="sasva.demo.sas.com"
SIGNER="http://sasva.demo.sas.com:80/SASLASRAuthorization" ;


                /* Drop existing LOAN APPLICANTS_SCORED table */
proc datasets lib=valibla nodetails nolist;
  delete Loan_Applicants_Scored;
run;
               /* Place LOAN APPLICANTS_SCORED In-Memory */
data VALIBLA.Loan Applicants Scored();
     set Work.Loan_Applicants_Scored (  );
```

```
      run;

/* Step 4. Surface a duplicate version of the HTML form generated in
the first stored process, "UPLOAD_FORM_STP" */


data  null ;
  format infile $char256.;
  input;
  infile = resolve(_infile_);
  file  webout;
  put infile;
cards4;
<html5>
<BODY>
<form action="http://sasva.demo.sas.com/SASStoredProcess/do"
enctype="multipart/form-data" method="post">
<input type="hidden" name="_program" value="/Shared
Data/SP/BulkLoanScore">
<th>Choose a file to upload:</th>
<td><input type="file" name="myfile"></td>
<td colspan="2" align="right"><input type="submit" value="SUBMIT"
onclick="msg()"></td>

</form>
</html5>
;;;;
run;
```

After registering the code above as a SAS Stored Process, we're ready to complete our dashboard.

FINISH THE DASHBOARD

In the first part of this section we created a report to display the information from our scored data set. In our final step, we are going to add our stored process "Upload_Form_STP" to that report. In SAS Visual Analytics Designer, on a new tab, we will add the stored process object. Figure 7 shows what our stored process will look like:

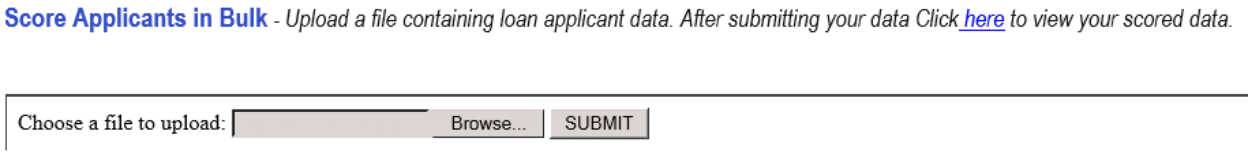


Figure 7. Example Report for Scored Data

We can use the text object in SAS Visual Analytics Designer to provide instructions for our end users. In this example, we used a section link to link the word "here" to the tab we created to display information from our scored data set. Requiring the end user to click a link to view the results is strategic as it forces

refresh of the report before it is displayed. Since our scoring process uploads a new version of the table, it is important that our report is refreshed. We are now ready to test our dashboard. We use the browse button to upload a CSV file, and then click submit. Once the data is submitted and scored, the box that contains the path to our file will clear. Once the file is cleared, we can click the link to view our results.

CONCLUSION

The SAS platform has both powerful analytics and exceptional visualization capabilities. Through integration of the SAS Stored Process, we are able operationalize analytics in an easy-to-use interface, putting the power of analytics into the hands of those who need it. Figure 8 shows the finished product (a dashboard capable of interactive, individualized scoring, as well as an option for scoring in bulk):

Figure 7. Example Report for Scored Data

ACKNOWLEDGMENTS

Thanks to Bruce Mills and Joe Flynn for their contributions to this project.

RECOMMENDED READING

- *Base SAS Procedures Guide*
- *SAS 9.4 Stored Processes: Developer's Guide*

CONTACT INFORMATION

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

Eli Kovick
100 SAS Campus Drive
Cary, NC 27513
SAS Institute Inc.
Eli.Kovick@sas.com
http://www.sas.com