PharmaSUG 2023 - Paper AP-189

You can REST easy validating Synthetic Data

Ben Howell, SAS;
Ben Bocchicchio, SAS

ABSTRACT

Synthetic data is a promising approach to replace or supplement the control arm of a clinical trial. Effective use of synthetic data could theoretically cut costs, time, and patient burden of clinical trials in half. Machine learning models are key to evaluating synthetic data for its equivalence with real data. The necessary machine learning and statistical capabilities for this purpose may not come standard in a clinical data repository (CDR) and statistical computing environment (SCE). In this case, it is necessary to integrate the CDR and SCE with a platform used to run machine learning models in order to keep sufficient electronic records for the regulatory compliance of clinical trial data during this process. Additionally, CDRs with REST APIs can be accessed remotely using simple HTTP requests from an external system. Curl commands can be called from SAS, R, Python, or the programming language of your choice. This paper demonstrates how a user with knowledge of REST APIs could run machine learning models to validate synthetic control data for use in clinical trials while keeping an audit trail of the process without requiring any knowledge of the CDR and SCE or machine learning platform. The products used in this research include PC SAS, the CDR and SCE SAS Life Science Analytics Framework, and SAS Viya. This paper is intended for individuals with knowledge of REST APIs.

INTRODUCTION

One of the largest limiting factors to progress in a clinical trial is the recruitment and selection of patients. In the cumbersome process of enrollment, 80% of trials overshoot the enrollment timeline, and 30% of phase-III trials are prematurely terminated due to patient enrollment challenges (Bhattamisra et al., 2023). When thinking about ways to remedy these enrollment issues, it's worth considering practices that reduce the number of patients needed. Incorporating synthetic data into a clinical trial to supplement or replace the control arm does just that. If previously published clinical trial data and/or real-world evidence exists for the standard of care being offered in your trial, why not use it to your advantage and leave the control arm out of your enrollment equation? Sources like routine health records, insurance claims, and patient registries can be manipulated using statistical or machine learning methods to create generative models (Thorlund et al., 2020).

When these models are used to produce synthetic data for the control arm of a clinical trial, the data sets need to be validated by evaluating the synthetic data for its equivalence to real data. You can test if the synthetic data behaves like real data with a random forest. A forest model is an ensemble of decision trees that is useful in predicting a binary target based on a number of contributing factors (Breiman, 2001). In this work, a combination of synthetic and real data is passed through a forest model that tries to predict which source each patient entry is from. In general, if the model is unable to tell whether data comes from the synthetic data set or the real data set, then the synthetic data is seen as equivalent in its behavior to the real data.

To include forest models in your clinical trial data processing, you'll need to make use of a machine learning platform. SAS Viya is a powerful tool with machine learning capabilities; however, it does not keep the necessary electronic records for regulatory compliance of clinical trial data. SAS Life Science Analytics Framework (LSAF) is a clinical data repository (CDR) and statistical computing environment (SCE) that does maintain these audit records. The first section of this paper will show how to integrate these platforms to analyze synthetic data against real data using a forest model in Viya and output the results back to LSAF while capturing an audit trail of the process.

While this integration is useful for users that understand the intricacies of the two platforms, there is an easier way to make use of the system. Starting with version 5.4.1, LSAF is equipped with REST APIs that use simple HTTP requests to access the functionality of the platform from a remote location. These requests can be made from SAS, R, Python, or the programming language of your choice. The second section of this paper will show how a PC SAS programmer can utilize REST APIs to upload a synthetic

data set to LSAF, execute the job shown in the first section to validate the synthetic data, and download the results to their PC. This process is accomplished with just a few REST API calls, and LSAF tracks each action in the audit history.

INTEGRATION OF CDR AND SCE WITH MACHINE LEARNING PLATFORM

This section will detail the integration of SAS Life Science Analytics Framework, a CDR and SCE, with SAS Viya, a machine learning platform. Figure 1 shows this process at a high level. Starting with real clinical data in LSAF, a job sends the data to Viya where it's analyzed using a forest model. Then, the results are brought back to LSAF as the output of the job.

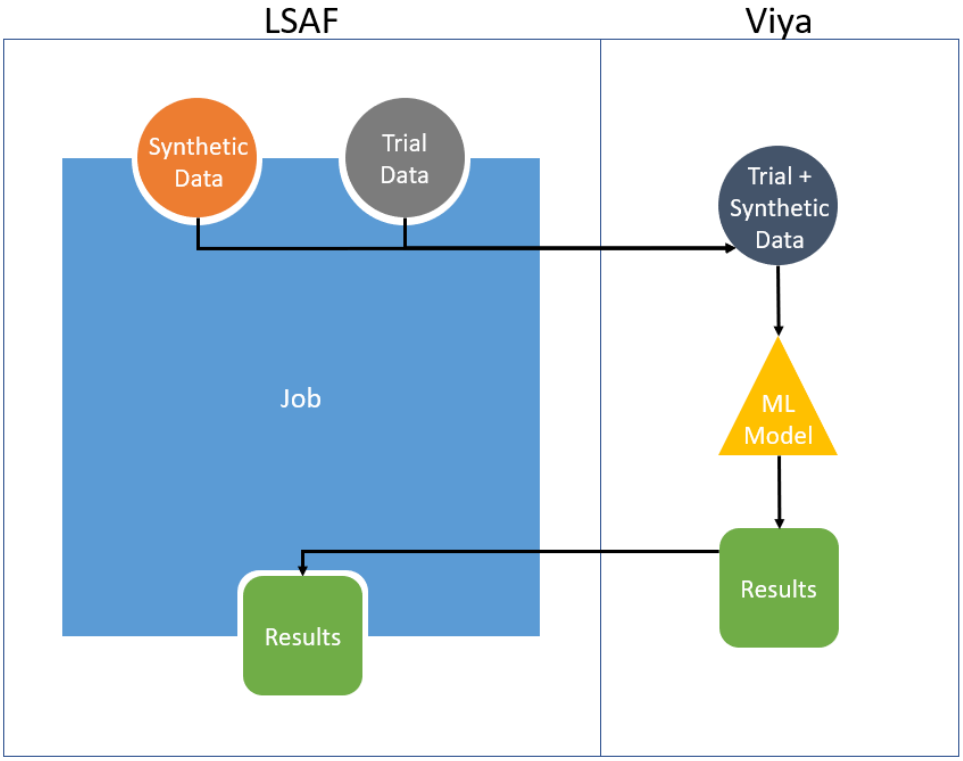


Figure 1. Data flow diagram for LSAF-Viya integration.

While this example uses a forest model, there are many other machine learning methods possible with Viya. They can be found here:
https://documentation.sas.com/doc/en/pgmsascdc/v_037/casml/titlepage.htm

LSAF OVERVIEW

LSAF is a cloud-based CDR and SCE which supports SSO with two distinct data stores: the repository and the workspace. The repository is a shared location where files are stored with permissions and privileges to control access. The workspace is a private area for each user. To edit a repository file, it must be checked out to your user workspace, which can only be done by one user at a time. Programs in the repository are executed as part of a job. Jobs can have parameters that are be passed to their program(s) through macro variables. Actions in the repository are tracked in the Audit History, including details about jobs that are run and files that are created, updated, or removed. This work uses LSAF version 5.4.1 which runs SAS 9.4 M7.

AUTHENTICATION TO VIYA FROM LSAF

Viya authentication uses a special file called an authinfo file. This file contains the hostname of the Viya environment, the port number of the Viya environment, and the Viya user's userid and password. This file can be referenced to authenticate to the Viya server as an alternative to including your Viya password in

programs. The job ‘create_auth_file.job’ executes a program that creates this file in a secure location. The Viya hostname and port are set within the program, and your Viya userid and password are job parameters that are passed to the program. Your password is a masked parameter so that the value will not appear in any logs. The password is first encoded, and then the four values are added to the authinfo file.

The filename statement that creates this file has multiple levels of security:

```
filename out "/lsafshared/SASWorkspaces/&_sddusr_./.authinfo"
permission='A::u::rw-,A::g::---,A::o::---';
```

The LSAF built-in macro variable &_sddusr_ resolves to your LSAF userid, and the path /lsafshared/SASWorkspaces/<your user id> is the physical location of your user workspace that only you have access to. Referencing the path in this way ensures that even if the job is executed in the repository, the authinfo file will be written to the workspace. Secondly, the name of the file is .authinfo. Files that start with a “.” are called dotfiles, and they are hidden by the Linux OS. This means the file is stored in your private workspace, and even you won’t be able to see the file from the UI. Lastly, for redundant security, a permission= option is used so that only the user who created the file (you) can read and write to the file, but other users are not granted any access.

PUSH DATA TO VIYA FOR MACHINE LEARNING AND RETURN OUTPUTS

The job ‘validate_synthetic_data.job’ creates a SAS session in LSAF which submits code via a CAS session into Viya, runs a machine learning procedure, and brings the outputs back to LSAF. A forest model is the machine learning technique used in this example, but any of the techniques supported by Viya could be used in its place for different applications. The job takes two data sets as input – the synthetic data set, and the real trial data to compare it to. These data sets are combined, and the resulting comparison data set contains a new variable, “source,” that notes each observation as “synthetic” or “real.”

Next, the comparison data set must be passed to Viya. This job uses the authinfo file created in the previous job to make the connection to Viya. The following options create the connection:

```
options cashost="&viyaServer" casport=&casPort
authinfo="/lsafshared/SASWorkspaces/&_sddusr_./.authinfo";
```

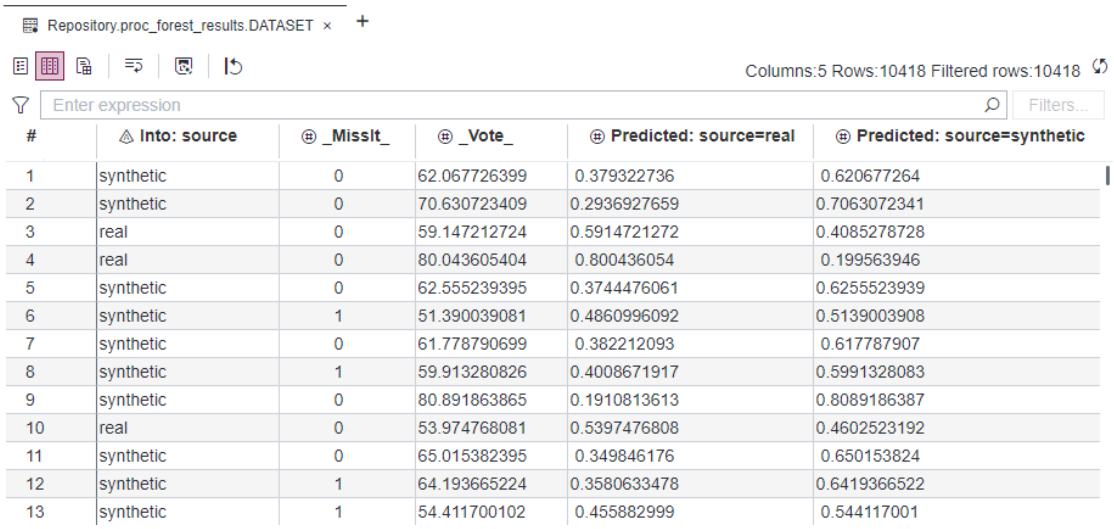
The path to the authinfo file is referenced the same way it is assigned in the filename statement so that the workspace file can be accessed from the repository job. Once this connection is made, Viya caslibs are assigned to SAS librefs in the LSAF SAS session so that they can be referenced. Then you can use a simple data step to transfer files between LSAF and Viya. The following step loads the combined data set, compare.sas7bdat, from a local LSAF SAS library called ‘lsaf’ to a CAS table in the ‘casuser’ CAS library in Viya:

```
data casuser.compare;
   set lsaf.compare;
run;
```

Now that the data is in a CAS table, the following FOREST procedure builds a model to predict the source of each observation based on the input variables:

```
proc forest data=casuser.compare;
   input <<list of variables>>;
   target source/level=nominal;
   output out=casuser.proc_forest_results;
run;
```

The output statement specifies the CAS table where the results are written. Since this portion of the code is processed by the Viya server, the output table must be a CAS table in Viya. The table can easily be brought back to LSAF from Viya with a data step, the same way that the compare data set was loaded to Viya from LSAF. An example of this output data set is shown in Figure 2:



Repository.proc_forest_results.DATASET

Columns:5 Rows:10418 Filtered rows:10418

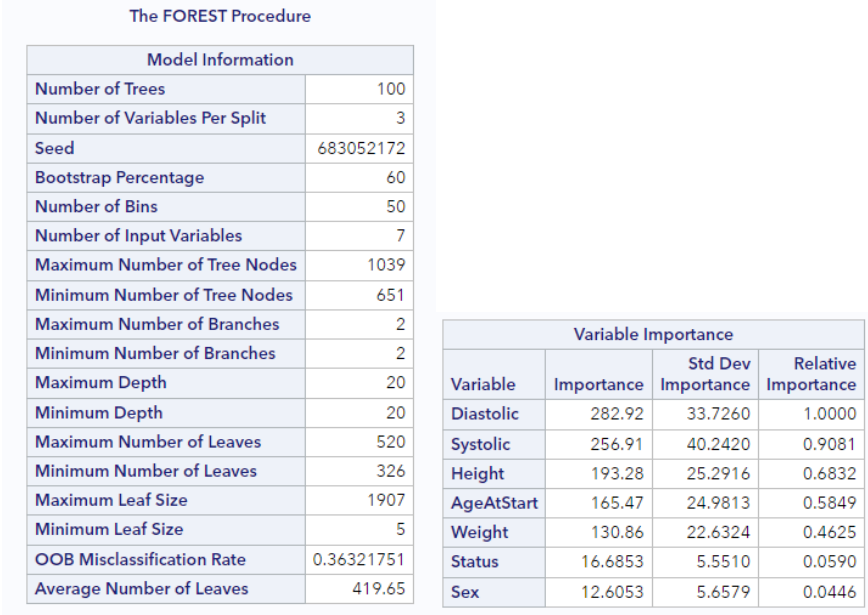
Enter expression

Filters...

#	Into: source	_MisIt_	_Vote_	Predicted: source=real	Predicted: source=synthetic
1	synthetic	0	62.067726399	0.379322736	0.620677264
2	synthetic	0	70.630723409	0.2936927659	0.7063072341
3	real	0	59.147212724	0.5914721272	0.4085278728
4	real	0	80.043605404	0.800436054	0.199563946
5	synthetic	0	62.555239395	0.3744476061	0.6255523939
6	synthetic	1	51.390039081	0.4860996092	0.5139003908
7	synthetic	0	61.778790699	0.382212093	0.617787907
8	synthetic	1	59.913280826	0.4008671917	0.5991328083
9	synthetic	0	80.891863865	0.1910813613	0.8089186387
10	real	0	53.974768081	0.5397476808	0.4602523192
11	synthetic	0	65.015382395	0.349846176	0.650153824
12	synthetic	1	64.193665224	0.3580633478	0.6419366522
13	synthetic	1	54.411700102	0.455882999	0.544117001

Figure 2. Synthetic data validation output data set.

There is also an ODS statement that creates a summary of the forest model analysis in an html file, Viya_Procedure_Output.htm, shown in Figure 3. Information is included about the forest model and the relative importance of the variables in making predictions of the source:



The FOREST Procedure

Model Information	
Number of Trees	100
Number of Variables Per Split	3
Seed	683052172
Bootstrap Percentage	60
Number of Bins	50
Number of Input Variables	7
Maximum Number of Tree Nodes	1039
Minimum Number of Tree Nodes	651
Maximum Number of Branches	2
Minimum Number of Branches	2
Maximum Depth	20
Minimum Depth	20
Maximum Number of Leaves	520
Minimum Number of Leaves	326
Maximum Leaf Size	1907
Minimum Leaf Size	5
OOB Misclassification Rate	0.36321751
Average Number of Leaves	419.65

Variable Importance			
Variable	Importance	Std Dev Importance	Relative Importance
Diastolic	282.92	33.7260	1.0000
Systolic	256.91	40.2420	0.9081
Height	193.28	25.2916	0.6832
AgeAtStart	165.47	24.9813	0.5849
Weight	130.86	22.6324	0.4625
Status	16.6853	5.5510	0.0590
Sex	12.6053	5.6579	0.0446

Figure 3. Parts of the ODS output from the synthetic data validation job.

The forest model tries to predict the source of each observation based on the other variables in the data set. The choice between “synthetic” and “real” is a binary target, and the OOB Misclassification Rate is a percentage of how often the model incorrectly assigns observations to a source. In most applications of a forest model, a low misclassification rate would indicate a good model. However, in this instance, a misclassification rate of 50% would mean that the model is completely incapable of discerning one source from the other. In other words, the synthetic control data is equivalent to or behaves the same as the real trial data. If the misclassification rate is not close to 50%, then you might have to revisit your synthetic data generation. In this example run, the misclassification rate was 36%, so there is room for improvement but our focus was primarily on the process and not on the results for our example data set.

To summarize, the combination of ‘create_auth_file.job’ and ‘validate_synthetic_data.job’ enables you to connect to Viya from LSAF, take advantage of machine learning methods possible with Viya, and store

the outputs in LSAF. Since this process is taking place in the repository, versioning can be enabled on these data sets, and every action is tracked in the Audit History. Audit records captured from a run of 'validate_synthetic_data.job' are shown in Figure 4:

Object	Name	Location	Action	Acted On By (Display Name)	Mode	Date Acted On
SAS Data Set	proc_forest_results.sas7bdat	/LSAF-Viya/Remote_Access/Outputs	Created	behowe (Ben Howell)		Mar 27, 2023, 4:50:41 PM GMT-04:00
Folder	Outputs	/LSAF-Viya/Remote_Access	File added	behowe (Ben Howell)		Mar 27, 2023, 4:50:41 PM GMT-04:00
SAS Data Set	proc_forest_results.sas7bdat	/LSAF-Viya/Remote_Access/Outputs	Associated	behowe (Ben Howell)		Mar 27, 2023, 4:50:41 PM GMT-04:00
File	Validation_Summary.htm	/LSAF-Viya/Remote_Access/Outputs	Checked in	behowe (Ben Howell)		Mar 27, 2023, 4:50:41 PM GMT-04:00
File	Validation_Summary.htm	/LSAF-Viya/Remote_Access/Outputs	Permissions created	behowe (Ben Howell)		Mar 27, 2023, 4:50:41 PM GMT-04:00
File	Validation_Summary.htm	/LSAF-Viya/Remote_Access/Outputs	Owner changed	behowe (Ben Howell)		Mar 27, 2023, 4:50:41 PM GMT-04:00
File	Validation_Summary.htm	/LSAF-Viya/Remote_Access/Outputs	Created	behowe (Ben Howell)		Mar 27, 2023, 4:50:41 PM GMT-04:00
Folder	Outputs	/LSAF-Viya/Remote_Access	File added	behowe (Ben Howell)		Mar 27, 2023, 4:50:41 PM GMT-04:00
File	Validation_Summary.htm	/LSAF-Viya/Remote_Access/Outputs	Associated	behowe (Ben Howell)		Mar 27, 2023, 4:50:41 PM GMT-04:00
File	validate_synthetic_data.lst	/LSAF-Viya/Remote_Access/Jobs	Checked in	behowe (Ben Howell)		Mar 27, 2023, 4:50:41 PM GMT-04:00
File	validate_synthetic_data.lst	/LSAF-Viya/Remote_Access/Jobs	Checked out	behowe (Ben Howell)		Mar 27, 2023, 4:50:41 PM GMT-04:00
File	validate_synthetic_data.log	/LSAF-Viya/Remote_Access/Jobs	Checked in	behowe (Ben Howell)		Mar 27, 2023, 4:50:41 PM GMT-04:00
File	validate_synthetic_data.log	/LSAF-Viya/Remote_Access/Jobs	Checked out	behowe (Ben Howell)		Mar 27, 2023, 4:50:41 PM GMT-04:00
SAS Data Set	compare.sas7bdat	/LSAF-Viya/Remote_Access/Data	Checked in	behowe (Ben Howell)		Mar 27, 2023, 4:50:41 PM GMT-04:00
SAS Data Set	compare.sas7bdat	/LSAF-Viya/Remote_Access/Data	Checked out	behowe (Ben Howell)		Mar 27, 2023, 4:50:41 PM GMT-04:00
Job	validate_synthetic_data.job	/LSAF-Viya/Remote_Access/Jobs	Ran	behowe (Ben Howell)		Mar 27, 2023, 4:50:33 PM GMT-04:00

Attribute	From	To
Size		768 KB
Copy to workspace		Allow all
Comment		Checked in on completion of job: /LSAF-Viya/Remote_Access/Jobs/validate_synthetic_data.job

Figure 4. Audit History records for the job run.

REMOTE ACCESS TO CDR USING REST APIS

The LSAF jobs in the previous section work together to validate synthetic data using a forest model. Once a job is in place, it can always be run from within LSAF. However, you might be a statistician without extensive knowledge of the LSAF and Viya platforms who wants to validate a synthetic data set. This section will show an example of how you can access the functionality of LSAF remotely with just your data and a few REST API calls to validate synthetic data. This process is summarized in Figure 5:

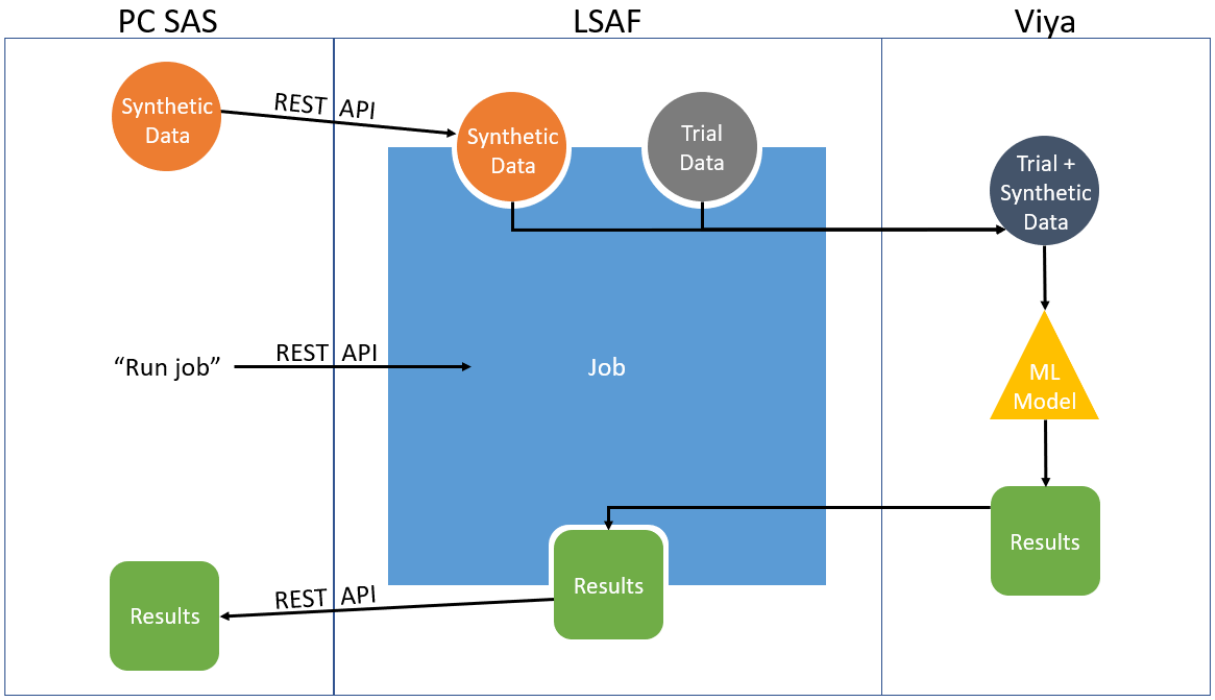


Figure 5. Data flow diagram for validation of synthetic data from PC SAS using REST APIs.

In this example, REST APIs are called from PC SAS to upload synthetic data to LSAF, remotely execute the LSAF job to validate the data using a forest model and download the results to your local computer. Each REST API has a corresponding macro program named %rest_<<action>>. Note that these programs could be implemented using any programming language that supports HTTP requests.

REST API AUTHENTICATION

REST APIs communicate with LSAF through HTTP requests. In PC SAS, these requests are made using the HTTP procedure. The first call that must be made is to logon to LSAF. The following code shows an example of PROC HTTP used in the %rest_logon macro:

```
proc http
   url="https://<<Your LSAF instance>>.ondemand.sas.com/lsaf/api/logon"
   method="POST"
   webusername="<<Insert Userid>>"
   webpassword="<<Insert Password>>"
   AUTH_BASIC
   headerout=resphdrs
   out=resp;
run;
```

This code tries to logon to the url using the webusername and webpassword. The headerout= and out= options specify file names to write the response header and response, respectively. If the logon is unsuccessful, the response will contain errors or warnings that explain why. If the logon is successful, the response header will contain an authentication token. The token can be thought of as a session id, and it will expire after 30 minutes of inactivity. The token is a required input for any other REST API calls.

In the %rest_logon macro, once the token is received, its value is saved to a text file called token.txt and a global macro variable “&token” is updated with its value:

```
data _null_;
   file "&local_folder./Authentication/token.txt";
   infile resphdrs dlm=' ' firstobs=2 obs=2;
   length item $13 value $100;
   input item value;
   value=strip(value);
   put value;
   call symputx("token",value,'G');
run;
```

At the start of any program using REST API macros, the first macro called is %rest_createtoken. This macro will check to see if the token.txt file exists. If it does, it will save the value to the “&token” macro variable. If it doesn’t, %rest_logon is executed to create a new token. The following code defines the %rest_createtoken macro:

```
%macro rest_createtoken;
   option nonotes;
   %if %sysfunc(fileexist("&local_folder./Authentication/token.txt")) %then
   %do;
      %put &SYSMACRONAME: Using stored token.;
      data _null_;
         infile "&local_folder./Authentication/token.txt";
         length value $100;
         input value;
         value=strip(value);
         call symputx("token",value,'G');
      run;
   %end;
   %else %do;
```

```
      %put &SYSMACRONAME: Creating new token.;
      %rest_logon;
   %end;
   option notes;
%mend rest_createtoken;
```

GENERAL STRUCTURE OF REST API MACRO PROGRAMS

The general structure of REST API macros is as follows:

1. Make the REST API request (using the authentication token macro variable “&token”).
2. Capture the response.
3. If the initial request failed due to authentication, logon and try again with the new token.
4. If the request still fails, errors are written to the log and/or specified data sets.
5. If the request is successful, results are written to the log and/or specified data sets.

This structure accounts for the fact that tokens expire after 30 minutes of inactivity. If the token saved in token.txt is expired, %rest_createtoken still assigns the value to the “&token” macro variable. Each macro program will try once to re-authenticate and make the request again, and after that the response will be captured.

RUN LSAF JOBS FROM PC SAS

A PC SAS programmer can utilize REST API macros to upload a synthetic data set to LSAF, execute the job ‘validate_synthetic_data.job’ introduced above with the newly uploaded data as input, and download the results. Note that this job requires an authinfo file to be present in the user’s workspace for connection to Viya. If this file is not in place, REST API macros can be used in a similar way to the code shown below to execute the job ‘create_auth_file.job’ and create this file.

The first REST API macro used is %rest_createtoken to create the authentication token macro variable. Then, the synthetic data set ‘synthetic_data.sas7bdat’ is uploaded to LSAF with the %rest_rep_uploadfile macro:

```
%rest_rep_uploadfile(
   lsaf_path=&lsaf_folder/Data/synthetic_data.sas7bdat,
   local_path=&local_folder/Data/synthetic_data.sas7bdat,
   lsaf_overwrite=true);
```

The path “&lsaf_folder/Data” is the LSAF repository location where the job looks for input data sets, and this data set will be overwritten if it already exists. Once the data is uploaded, another REST API macro tells LSAF to run the job ‘validate_synthetic_data.job’:

```
%rest_rep_runjob(
   lsaf_path=&lsaf_folder/Jobs/validate_synthetic_data.job);
```

When the job has been submitted, you’ll need to wait until the job makes the connection between LSAF and Viya and brings the results back to LSAF before executing the next step in PC SAS. A job receives a submission id when it is submitted, and this is part of the response to the REST API request to run a job. The submission id is used to check the status of the job with another REST API macro, %rest_job_getsubmissionstatus, which makes a REST API request to return the status of that submission. The following code checks the status every 2 seconds and does not proceed until the job has completed:

```
%macro wait;
   %TryAgain:
      data _null_;
         call sleep(2,1);
      run;
```

```
    %rest_job_getsubmissionstatus(lsaf_jobsubmission_id=&jobSubmissionId);

    %if ("&jobStatusType"^="COMPLETED") %then %goto TryAgain;
%mend wait;
%wait;
```

Once the job has completed, the submission status determines if the job was successful. If the job had warnings or errors, the log is downloaded from LSAF to a local file using another REST API macro, %rest_rep_downloadfile. The contents of the log are printed to the PC SAS log for ease of viewing with the following data step:

```
data _null_;
    infile "&local_folder/Outputs/log.txt" truncover;
    input job_log $500.;
    if index(job_log,"WARNING:")>0 then put "WARNING: " job_log=;
    else if index(job_log,"ERROR:")>0 then put "ERROR: " job_log=;
    else put job_log=;
run;
```

However, if the job was successful, the %rest_rep_downloadfile macro is used to download the synthetic data validation results:

```
%rest_rep_downloadfile(
    lsaf_path=&lsaf_folder/Outputs/proc_forest_results.sas7bdat,
    local_path=&local_folder/Outputs/proc_forest_results.sas7bdat);

%rest_rep_downloadfile(
    lsaf_path=&lsaf_folder/Outputs/Validation_Summary.htm,
    local_path=&local_folder/Outputs/Validation_Summary.htm);
```

In summary, a user with active LSAF and Viya accounts can run a PC SAS program to upload synthetic data to LSAF, use machine learning to validate their data against trial data, and download the results, all while keeping audit records in LSAF of the entire process. The Audit History entries from the remote run of the job using REST APIs are just as thorough as when the job is run from within LSAF. Some of the entries even note that the action was a result of a REST API request (Figure 6). These records are essential for validating synthetic data in a compliance-friendly way.

Object	Name	Location	Action	Acted On By (Display Name)	Mode	Date Acted On
File	Validation_Summary.htm	/LSAF-Viya/Remote_Access/Outputs	Downloaded	behowe (Ben Howell)		Mar 27, 2023, 10:39:09 AM GMT-04:00
SAS Data Set	proc_forest_results.sas7bdat	/LSAF-Viya/Remote_Access/Outputs	Downloaded	behowe (Ben Howell)		Mar 27, 2023, 10:39:09 AM GMT-04:00

Attribute	From	To
Client used		Public REST API
Destination IP		10.225.127.70
File source		Repository

Figure 6. Audit History records from REST API download requests.

CONCLUSION

This paper proposes a method to incorporate machine learning into clinical trial data processing by integrating a machine learning platform with a CDR and SCE to validate synthetic data sets. This work also provides an example of how programmers without LSAF and Viya skills can still take advantage of machine learning functionality with proper electronic records for regulatory compliance using REST APIs. However, this process is just an example of one remote environment interacting with LSAF and Viya to do one type of machine learning analysis. There is great potential for more remote environments and different machine learning techniques to be used in a similar fashion to open the door for machine learning use in clinical trials.

REFERENCES

Bhattamisra, S.K.; Banerjee, P.; Gupta, P.; Mayuren, J.; Patra, S.; Candasamy, M. 2023. “Artificial Intelligence in Pharmaceutical and Healthcare Research.” *Big Data and Cognitive Computing*, 7(1):10. https://doi.org/10.3390/bdcc7010010

Breiman, L. 2001. “Random Forests.” *Machine Learning*, 45:5–32. https://doi.org/10.1023/A:1010933404324

Thorlund, K.; Dron, L.; Park, J.J. H.; Mills, E. J. 2020. “Synthetic and External Controls in Clinical Trials - A Primer for Researchers.” *Clinical Epidemiology*, 12:457–467. https://doi.org/10.2147/CLEP.S242097

CONTACT INFORMATION

Your comments and questions are valued and encouraged. If you are interested in learning more or seeing a demonstration of the code, contact the author at:

Ben Howell
Ben.Howell@sas.com