Paper 2712-2018

SAS® Viya™: The Beauty of REST in Action

Sean Ankenbruck and Grace Heyne Lybrand, Zencos Consulting LLC

ABSTRACT

The introduction of SAS® Viya® opened the SAS® brand and its industry leading machine learning algorithms to the open source community. With new ability to connect open source interfaces such as Lua, Python, and R as well as REST APIs to CAS, this new platform becomes a powerful tool to include in any analytical toolbox. Using SAS® Cloud Analytic Services (CAS), users can easily surface data and analytical model results to a WebApp through REST API connectivity. This capability is one key aspect Viya offers to provide real-time analytics. REST APIs provide a way to access SAS Viya over HTTP, and CAS capabilities are more efficient in communicating between front end and back end applications. Zencos will showcase SAS Viya's capabilities of leveraging the CAS server and connecting to REST APIs to surface data for real-time decision making using a case study where we score user data in real-time.

INTRODUCTION

SAS Viya is a cloud-enabled, in-memory analytics engine. It is built around the SAS Cloud Analytic Services (CAS) framework. This framework provides some extremely powerful and efficient analytics tools to developers. This paper will discuss how REST APIs have been incorporated into SAS Viya to provides users with all the powerful analytical capabilities of the SAS language regardless of your chosen programming language. We will also walk through a case study detailing the process of developing a web application to conduct real-time analytics on user submitted data. Figure 1 depicts how a web client interacts with the SAS Viya REST APIs.

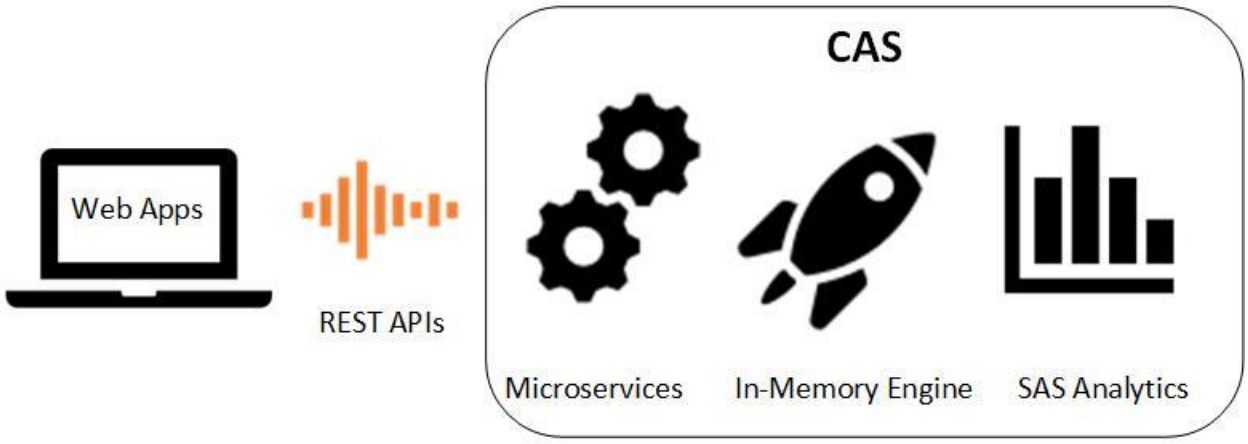


Figure 1. Representation of Web Client Using SAS Viya REST APIs.

REST APIS AND CAS ACTIONS

In this section we will discuss how SAS has integrated REST APIs into the Viya platform enabling the use of SAS analytics in any programming language that you prefer.

WHAT IS REST?

Representational State Transfer (REST) is an architectural style for distributed hypermedia systems that was first proposed by Roy Fielding in his 2000 thesis, *Architectural Styles and the Design of Network-based Software Architectures*. REST provides a set of architectural constraints that, when applied, emphasizes scalability of component interactions, generality of interfaces, independent deployment of components, and intermediary components to reduce interaction latency, enforce security, and encapsulate legacy systems.

WHAT IS A REST API?

A REST API is a defined set of instructions which developers can use to submit requests and receive responses, usually via HTTP actions (GET and POST methods for example). The modern web is an example of the REST architecture, as described by Fielding, used to provide a hypermedia interface for websites. In simplified terms, a set of rules has been compiled and is available on a web server, waiting for client applications to make requests. When an application submits a request, the API processes it, retrieves any necessary data and compiles a response. The client then receives back the response in some web standard format, usually XML or JSON. Figure 2 depicts the process flow of a RESTful API in action.

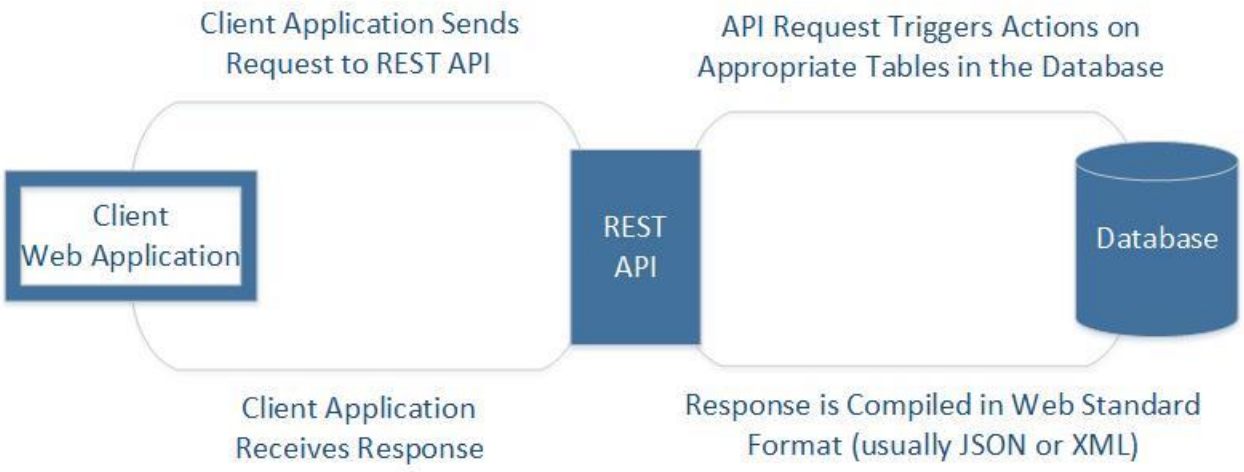


Figure 2. Representation of the Request / Response Lifecycle of a REST API

SAS VIYA REST APIS

Rest APIs provide a way to access the capabilities of SAS Viya using standard, open HTTP network protocols. This has opened the door to allow users to access SAS resources using many different open source technologies including: Java, Lua, Python, JavaScript, R, and other programming languages. The only requirement is that the language have a standard library capable of executing HTTP requests and the ability to parse and generate web standardized data, in XML or JSON.

UNDERSTANDING CAS ACTIONS

Cloud Analytical Services (CAS) is the centerpiece to the SAS Viya framework. When you want to establish a connection to the CAS server, you must create a session. Once this session has been established it is then possible to make use of the many CAS actions available to you. A CAS action is, in its simplest form, a task. These tasks can be anything from loading data, to transforming data, to performing advanced analytics. These actions fall into one of four categories:

- Statistics
- Analytics
- System
- Data Mining and Machine Learning

CAS actions and action sets are a game changer. It is now possible to conduct your analytics using the reliable features of the SAS language from the programming language of your choice! From data manipulation, to statistics, to advanced machine learning techniques, there is an action available for just about any situation.

WHAT IS PYTHON SWAT?

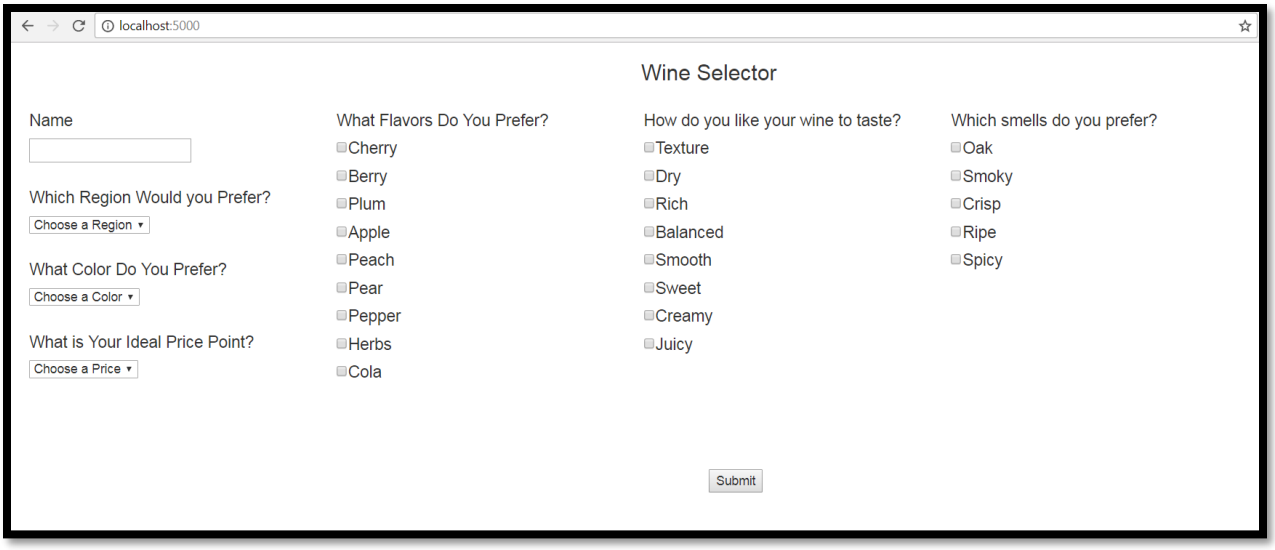
The SAS Scripting Wrapper for Analytics Transfer (SWAT) package is the Python client to SAS Cloud Analytic Services. The Python SWAT package provides direct access to CAS, using Python conventions and data structures to execute CAS actions. It allows you to load data into memory, apply CAS actions, summarize, model and score the data all from within Python. The format and structure of data and functions while using the package is like that of the popular Pandas library.

CASE STUDY: SCORING USER DATA IN REAL-TIME

For this case study, we developed a web application using the Python SWAT package. The application provided a seamless flow of data between the Python client - built in Flask - and our CAS Server - hosted on Amazon AWS. Furthermore, we built in the functionality to score data in real-time using CAS actions and return these results to the user. The sample application allows users to select various preferences related to wine. When the user submits their preferences, the application scores this information and returns to the user a suggested variety of wine.

CREATING THE USER INTERFACE

The first step in the process was to create a web application to capture use data. There are no specific requirements for developing a web application, but it must be able to capture the data and store it in a format that Python can transmit to SAS Viya. We chose to build our application in Python Flask due to its ease of use and rapid prototyping abilities. Display 1 is a sample of the user interface for our wine selector application.



Display 1: User Interface for Sample Web Application.

PARSING THE DATA

Using the Pandas library, we collected the user responses and stored them in a DataFrame. The attributes that users could select from are the exact variables that we used as inputs into our decision tree model. Once the data was in a format that we could upload to Viya, we then established a CAS session via the SWAT package.

ESTABLISHING A CONNECTION

Once the data was stored in memory on the Python web server, the next step was to establish a connection with the CAS server. The server must be up and running and the user must be an authorized user in the CAS system. The following parameters are necessary to connect:

- Hostname
- CAS Port
- Username
- Password

```
conn = swat.CAS(casHost, casPort, userName, passWord)

CAS('ip', 5570, user, protocol='cas', name='session-1', session='sessionID')
```

Once the connection was established, we had access to all the available CAS action sets that have been developed for SAS Viya.

UPLOADING USER RESPONSES

At this point in the process, the dataframe containing user responses was still in memory on the Python web server. Before we could use this data in the SAS Viya environment, it was necessary to load it to the CAS server. The SWAT package has various functions that allow for interactions between the Python language and SAS Viya, and the format is very similar to what you may be familiar with from the Pandas library. To upload a dataframe just call the upload_frame() function. This function takes a dataframe in memory on the Python client and uploads it as a CAS table to the CAS server. Here is an example of the upload_frame() function:

```
conn.upload_frame(user_responses,casout=conn.CASTable(name='user_responses'))
```

THE MODEL

For this case study, we generated data based on trends we found on a popular wine review website. We compiled a list of some of the most prominent terms used to describe wines and used them as inputs in a decision tree model to predict wine variety. The resulting model was used to generate score code that we integrated with our web application to score user responses and provide a suggested variety based on the user's preferences. The modeling process is depicted in Figure 3 below.

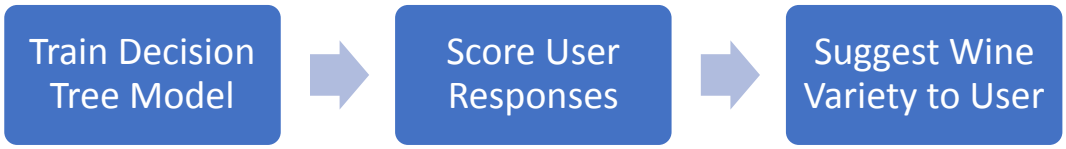


Figure 3: Case Study Modeling Process

TRAIN DECISION TREE MODEL

The model was built and trained using the `dtreeTrain` action in the `decisionTree` action set:

```
conn.loadactionset("decisionTree")
conn.decisionTree.dTreeTrain(
      casOut = {"name":"tree_model"},
      inputs = [{vars}],
      modelId = "DT_wine_variety",
      table = {"caslib":"public", "name": "wines_model_data"},
      target = "variety"
)
```

The `casOut` parameter specifies where to output the tree model. This tree model table can then be used as an input to the scoring function.

SCORE USER RESPONSES

We described how we acquired user responses in the section titled, *Uploading User Responses*. Once the user data and decision tree model was loaded to the CAS server, we scored the user responses to determine a wine variety that closely matched their preferences. The scoring was accomplished with the `dtreeScore` action in the `decisionTree` action set:

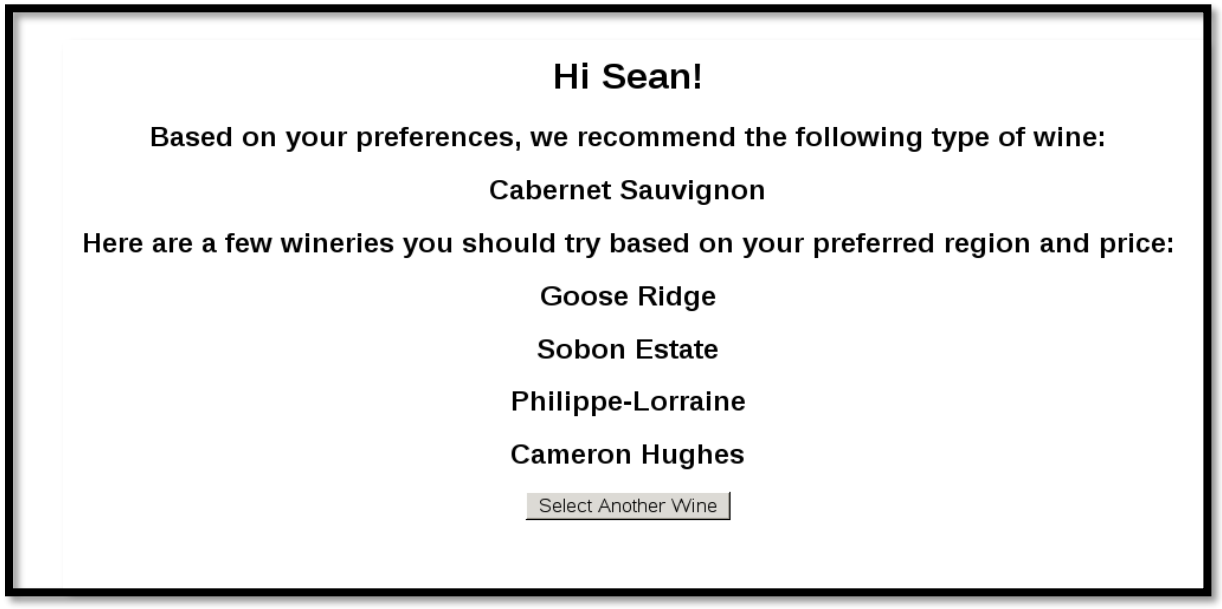
```
conn.decisionTree.dtreeScore(
      casOut ={"name":"scored_wine_data"},
      modelId = "_DT_wine_variety",
      modelTable = {"name":"tree_model"},
      table = {"caslib":"public","name":"user_responses", "vars":[{vars}]
      target = "variety"

)
```

This action creates a score table on the CAS server with the predicted probability for each of the wine varieties based on the user preferences. The variety with the greatest probability is the suggestion that we wanted to return to the user, but to do that we had to integrate this scoring code into our custom wine application.

SUGGEST WINE VARIETY TO USER

We had to figure out a way to retrieve the predicted wine variety from CAS and provide that prediction back to our user via the web application. This is where the SAS Viya platform truly shines. We took this decision tree code and integrated it directly into the web application. When a user clicks submit, the data is transmitted in real-time to the CAS server, scored, and then the results are retrieved and sent to our users in the form of a prediction of a wine variety that they should try. Thanks to a little bit of customization – and creativity – we also decided to provide the users with a few wineries that make their suggested variety based on the region and price preferences they selected. The results page can be seen in Display 2.



Display 2: User Interface Results Page

CONCLUSION

The introduction of SAS Viya has opened the SAS brand and its industry leading machine learning algorithms to the open source community. In this paper, we demonstrated how simple it was to integrate a scoring model into a custom web application written entirely in Python. The business impact of these new methodologies is astounding as it will allow more developers, analysts, and data scientists from various industries to integrate world class analytical software into applications written in their language of choice. There is no longer a need to choose one over another, instead it is possible to harness the relative strengths of each language and combine the results in ways that were previously unattainable. We think that this relationship between SAS and the open source community is an integral part of the future of analytics.

REFERENCES

Fielding, Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine, 2000.

RECOMMENDED READING

- SAS® Viya Programming Guide
- SAS® Cloud Analytic Services Action Sets
- Python SWAT Programming Guide
- Python Flask Programming Guide

CONTACT INFORMATION

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

Sean Ankenbruck
Zencos Consulting
sankenbruck@zencos.com
www.zencos.com

Grace Heyne Lybrand
Zencos Consulting
glybrand@zencos.com
www.zencos.com