Getting Started: API access to SAS® Visual Data Mining and Machine Learning using Python

Quick Start

Requirements

To use Python with SAS Cloud Analytic Services (CAS), the client machine that runs Python must meet the following requirements:

- Use 64-bit Linux or 64-bit Windows.
- Use a 64-bit version of Python, such as the Anaconda platform from Continuum Analytics.
- Use Python version 2.7.x or 3.4+. The Python package from SAS, SWAT, is compatible with both versions.
- The SAS Scripting Wrapper for Analytics Transfer (SWAT) package is available for download from https://github.com/sassoftware/python-swat. Information for installing is available from a README that is included in the download.
- Create an authinfo file with your *Project Data Sphere* credentials. See Create an Authinfo File in *Client Authentication Using an Authinfo File*.
- The full Python code sample outlined below can be found here and run using Jupyter Notebook: VDMML_PYTHON_API_ACCESS.txt

Connect and Start a Session

To enable a Python program to work with SAS Cloud Analytic Services, you must establish a connection with the server.

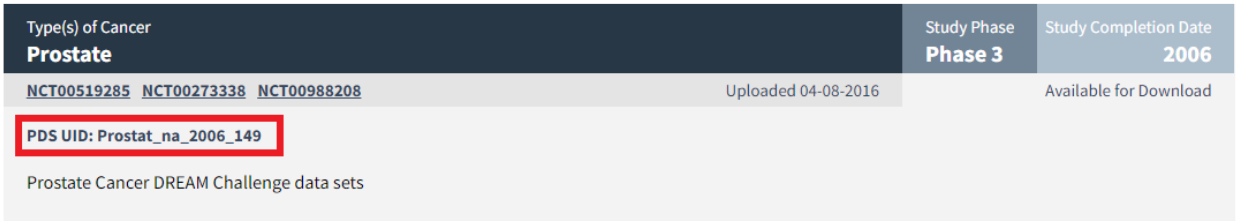
To run the following code in your Python client, download the certificates file: trustedcerts.pem
Replace /path/to/ trustedcerts.pem with the absolute path to the downloaded file, trustedcerts.pem.

```
import os, swat

# Set the SSL/TLS certificate:

os.environ["CAS_CLIENT_SSL_CA_LIST"] = '/path/to/ trustedcerts.pem'

# Connect to CAS:

s = swat.CAS("https://mpmprodvdmml.ondemand.sas.com/cas-shared-
default-http/", 443)

# Print the server status to make sure we're connected:

print(s.serverstatus())
```

The `s` variable now holds the connection to CAS, which can be used to load data and run CAS actions.

Load a *Project Data Sphere* Data Set

The *Project Data Sphere* data sets are automatically available to the platform's SAS tools, including connections to CAS through Python. First, identify the Unique Dataset ID for the data set you are interested in using. You can find this ID on the Access Data page for the data set.



This example will use the Prostate Cancer DREAM Challenge data sets, which has a Unique Dataset ID of Prostat_na_2006_149.

```
# List all files in the caslib:

res = s.table.fileInfo(allFiles=True, caslib="Prostat_na_2006_149")

print(res)
```

After listing the files, let's look for the training data files:

```
# List all files in the dataFiles_859 subdirectory:

res = s.table.fileInfo(allFiles=True, caslib="Prostat_na_2006_149",
path="dataFiles_859")

print(res)
```

Let's load one of the tables into CAS:

```
# Create an output table to load the file into:

core_train = s.CASTable('core_train', replace=True, caslib='CASUSER')

s.table.loadTable(sourceCaslib="Prostat_na_2006_149",
casOut=core_train, path="dataFiles_859/CoreTable_training.csv")

# View the first 5 rows:

print(core_train.head())
```

Notice we were able to run the head() function on the CASTable object. The SWAT package mimics much of the API of the Pandas package so that using CAS should feel familiar to current Pandas users.

Run a CAS Action to Perform Data Analysis

Let's run a frequency analysis on the STUDYID and DEATH columns in this table:

```
# Load the freqTab action set:

s.loadactionset('freqTab')

# Create a two-way cross tabulation table:
```

```
res = s.freqTab.freqTab(table=core_train, includeMissing=True,
tabulate=[{'vars':{'DEATH', 'STUDYID'}}])

print(res)
```

In this example we used the "freqTab" action set. Many other data mining and machine learning action sets are available on this platform through SAS Viya. A comprehensive list can be found here.

Save a CAS table to CASUSER

Changes to CAS tables are performed in-memory and will not persist once the user's session ends. You can save CAS tables to a SAS data set on the filesystem:

```
# Save the CAS table to a .sashdat file in the CASUSER caslib:

core_train.table.save(caslib="CASUSER", name="core_train.sashdat",
replace=True)
```

Alternatively, you can save your changes to a CSV file:

```
# Save the CAS table to a CSV file in the CASUSER caslib:

core_train.table.save(caslib="CASUSER", name="core_train.csv",
replace=True)
```

Finishing Up

Once you are done working with a CAS table and have saved any changes you want to keep, you should drop it from CAS memory to save system resources:

```
# Drop the table from memory:

core_train.table.dropTable()
```

Documentation

Python Programming for SAS

Getting Started with SAS® Viya® 3.4 for Python

Additional Videos

Getting Started with the Python Interface of SAS Viya

Getting Started: API access to SAS® Visual Data Mining and Machine Learning using R

Quick Start

Requirements

To use R with SAS Cloud Analytic Services (CAS), the client machine that runs R must meet the following requirements:

- Use 64-bit Linux or 64-bit Windows.
- Use a 64-bit version of R.
- Use R 4.0.5 or later.
- See the full R example below here.
- Create an authinfo file with your Project Data Sphere credentials. See Create an Authinfo File in *Client Authentication Using an Authinfo File*.
- The full R code sample outlined below can be found here: VDMML_R_API_ACCESS.txt

Install Packages

Install the required packages. These packages have additional dependencies that are automatically installed from CRAN when you run install.packages().

Use the link for the latest zip file from the SAS Scripting Wrapper for Analytics Transfer (SWAT) GitHub to install R-SWAT below.

Run the following code sample in your R client:

```
# Install required packages:

install.packages('httr')

install.packages('jsonlite')

install.packages('devtools')

install.packages('dplyr')

# Update the link below to use the latest zip file from the R-SWAT
releases:

install.packages('https://github.com/sassoftware/R-
swat/releases/download/v1.6.3/R-swat-1.6.3+vb21030-win-64.tar.gz',
repos=NULL, type='file',INSTALL_opts="--no-multiarch")

# Load the packages to be used:

library(devtools)

library(swat)
```

```
library(dplyr)

library(httr)

library(jsonlite)
```

Connect and Start a Session

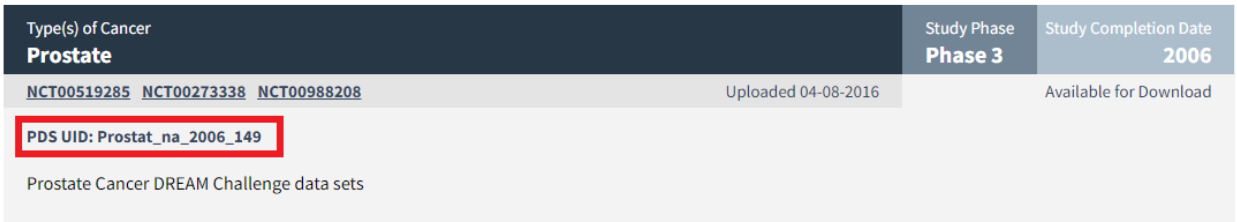
To enable an R program to work with SAS Cloud Analytic Services, you must establish a connection with the server.

```
# Connect to CAS:

s <- CAS("https://mpmprodvdmml.ondemand.sas.com/cas-shared-default-http/", 443, protocol='https')

# Print the server status to make sure we're connected:

print(cas.builtins.serverStatus(s))
```

The `s` variable now holds the connection to CAS, which can be used to load data and run CAS actions.

Load a *Project Data Sphere* Data Set

The Project Data Sphere data sets are automatically available to the platform's SAS tools, including connections to CAS through Python. First, identify the Unique Dataset ID for the data set you are interested in using. You can find this ID on the Access Data page for the data set.



This example will use the Prostate Cancer DREAM Challenge data sets, which has a Unique Dataset ID of Prostat_na_2006_149.

```
# List all files in the caslib:

res = cas.table.fileInfo(s, allFiles=TRUE,
caslib="Prostat_na_2006_149")

print(res)
```

After listing the files, let's look for the training data files:

```
# List all files in the dataFiles_859 subdirectory:

res = cas.table.fileInfo(s, allFiles=TRUE,
caslib="Prostat_na_2006_149", path="dataFiles_859")

print(res)
```

Let's load one of the tables into CAS:

```
# Load the table into CAS:

res = cas.table.loadTable(s, sourceCaslib="Prostat_na_2006_149",
casOut=list(name='core_train', replace=TRUE, caslib='CASUSER'),
path="dataFiles_859/CoreTable_training.csv")

print(res)

# Create a CASTable to reference the table we just loaded into CAS:

core_train = defCasTable(s, 'core_train', caslib='CASUSER')

# View the first 5 rows:

head(core_train)
```

Run a CAS Action to Perform Data Analysis

Let's run a frequency analysis on the STUDYID and DEATH columns in this table:

```
# Load the freqTab action set:

loadActionSet(s, 'freqTab')

# Create a two-way cross tabulation table:

res = cas.freqTab.freqTab(s, table=core_train, includeMissing=TRUE,
tabulate=c(vars=c('DEATH', 'STUDYID')))

print(res)
```

In this example we used the "freqTab" action set. Many other data mining and machine learning action sets are available on this platform through SAS Viya. A comprehensive list can be found here.

Save a CAS table to CASUSER

Changes to CAS tables are performed in-memory and will not persist once the user's session ends. You can save CAS tables to a SAS data set on the filesystem:

```
# Save the CAS table to a .sashdat file in the CASUSER caslib:

cas.table.save(s, table=core_train, caslib="CASUSER",
name="core_train.sashdat", replace=TRUE)
```

Alternatively, you can save your changes to a CSV file:

```
# Save the CAS table to a CSV file in the CASUSER caslib:

cas.table.save(s, table=core_train, caslib="CASUSER",
name="core_train.csv", replace=TRUE)
```

Finishing Up

Once you are done working with a CAS table and have saved any changes you want to keep, you should drop it from CAS memory to save system resources:

```
# Drop the table from memory:

cas.table.dropTable(s, name='core_train', caslib='CASUSER')
```

Documentation

R Programming for SAS Viya