How to configure Python and SASPy

Michael McCarthy PE, Austin Energy, Austin TX USA

ABSTRACT

SASPy is an interface between SAS® and Python that can generate SAS® code from Python3 scripts. The SASPy module enables Python objects and syntax to access SAS data. Thus, it's possible to use Python to perform machine learning techniques on existing SAS data sets using packages like Scikit-learn, SciPy, and NumPy. This paper presents a step-by-step tutorial showing the configuration of the connection from BASE SAS® to an Anaconda Distribution of Python3 running on Windows 10.

INTRODUCTION

SASPy provides Python access to your exiting archive of BASE SAS® data sets, making it possible for you to share insights and collaborate with users on the Python platform. If you already have Python and Jupyter Notebooks installed, just "pip" the SASPy package. If you don't, then go to https://www.python.org/downloads/ and install Python and pip packages as you need them. Note, there are several free open source Python distributions that will perform setup and configuration during install. This can potentially save a lot of time and frustration as there is likely older version(s) of Python on your PC configured to run with existing applications, and it's important to avoid a changing the configuration of these Python installations. Be sure to install Python3 since this will give you access Scikit-learn, SciPy, and NumPy. If you enjoy the Jupyter format, you can work in the Python3 Kernel as well as the SAS®Kernel. The Jupyter platform will allow you to program in either Python3 or BASE SAS®.

METHODOLOGY – INSTALLATION INSTRUCTIONS

The first thing to do is install Python3. I'm installing the Anaconda3 version that comes with pandas and Jupyter Notebooks installed and configured for Windows 10. Choosing a distribution is helpful since the majority of the Python configuration details are handled automatically:

Anaconda Installation:

1.) Go to https://www.anaconda.com/download/
2.) Select Python 3.6 Version (64 Bit Graphic Installer)
3.) Open the Anaconda3.exe installer > Agree to Terms > Select Just Me (Installs as App)
4.) When you are done you'll see a folder called Anaconda3. It has installed Anaconda Navigator, Anaconda DOS Prompt, Jupyter Notebook, and Spyder
5.) You're done installing the file server. This file server is running locally on your PC, but you can access all the installed tools from a browser (except - Anaconda DOS Prompt. Do this from your App list).
6.) Note that you now have a local install of Python and these tools under your user profile on this PC. Since it's just installed for your profile, you were not required to be an Admin on your PC. Note that the files are actually installed in C:\Users\YourWindowsName\AppData\Local\Continuum
7.) Go to the Anaconda DOS prompt and type Python
    (base) C:\Users\YourWindowsName>python
    You should see…
    Python 3.6.3 |Anaconda custom (64-bit)| (default, Oct 15 2017, 03:27:45) [MSC v.1900 64 bit (AMD64)] on win32
    Type "help", "copyright", "credits" or "license" for more information.
8.) Note: You may have other globally installed versions of Python on your PC (i.e. ArcGIS uses Python). These are separate from this install. When you install packages, please make sure you are using the Anaconda DOS

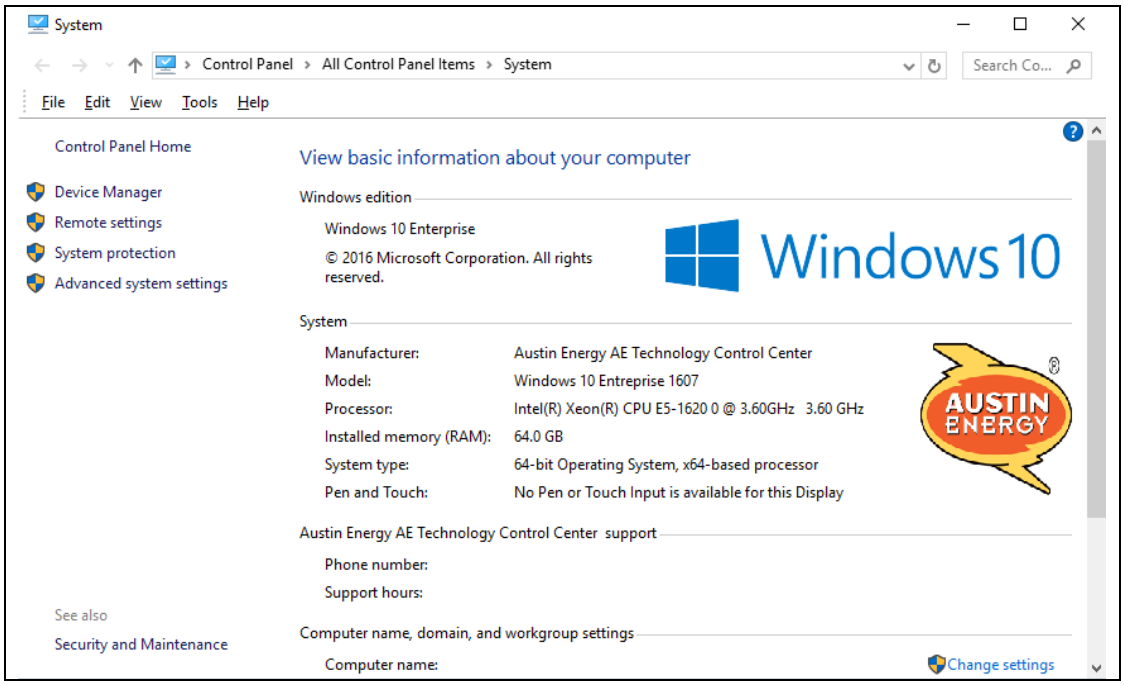
Prompt. The advantage of the Anconda3 version is that comes with pandas and Jupyter Notebooks installed and configured.

Now we're ready to install the SASPy package:

1.) SASPy provides Python access to all of the features that your SAS license allows
2.) Go to the Anaconda DOS Prompt > Type pip install saspy
3.) You will see some info about the install in your Anaconda DOS Prompt window. When it's done, navigate to C:\Users\YOURWINDOWSNAME\AppData\Local\Continuum\anaconda3\Lib\site-packages. As you install packages using the Python "pip" command you will see them in this directory. Note that you must import the packages using the import "package name" command in you Python script. Sometimes the package name is different (e.g. Sci-Kit Learn is called sklearn) so you may need to check the site-packages directory to see what things are actually named.
4.) Packages can launch or interact with other programs on you PC. To enable these, you will need to update the Java Path (or called "Path"). This "Path" is actually a list of important locations Java needs to know about. Java needs to know these locations since it's invoked from some Python packages (like saspy) to run other PC programs (like SAS). You can update your "Path" by following the instructions on the next page

Almost done…How to set the path and variables in Windows 8 and Windows 10:

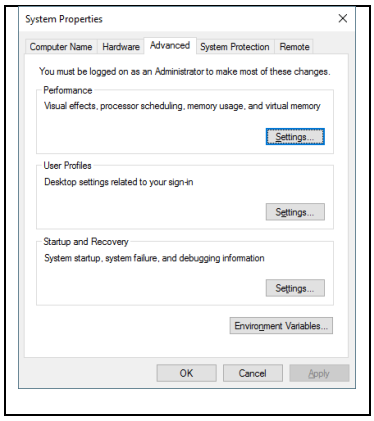
1.) From the Desktop, right-click the very bottom left corner of the screen to get the Power User Task Menu.



2.) From the Power User Task Menu, click **System**. It should look like:
3.) Click the **Advanced System Settings** link in the left column. You will need Admin privileges or IT to update this PATH

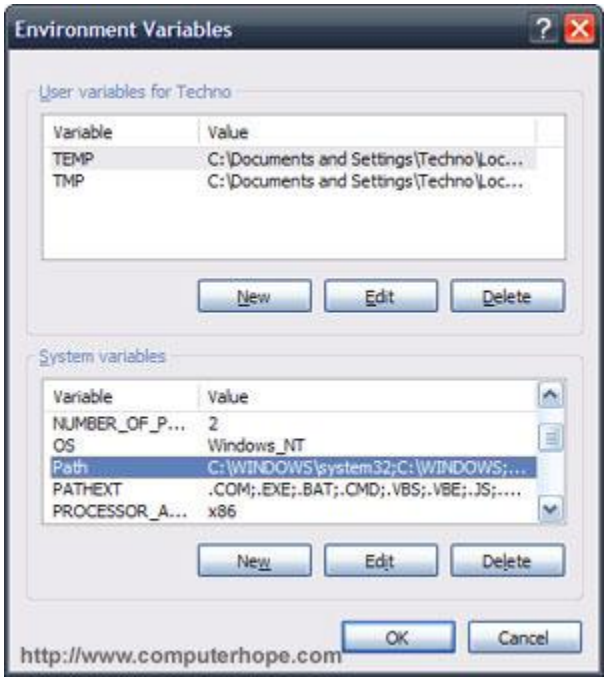
Note: In Windows 10, you may need to scroll down to the *Related settings* section and click the **System info** link. In the System window that opens, click the **Advanced system settings** link in the left column.

4.) In the System Properties window, click on the **Advanced** tab, then click the **Environment Variables** button near the bottom of that tab.



5.) In the Environment Variables window (pictured below), highlight the **Path** variable in the "System variables" section and click the **Edit** button. Add or modify the path lines with the paths you want the computer to access. Each different directory is separated with a semicolon as shown below.

C:\Program Files;C:\Winnt;C:\Winnt\System32



Note: You can edit other environment variables by highlighting the variable in the "System variables" section and clicking **Edit**. If you need to create a new environment variable, click **New** and enter the variable name and variable value. To view and set the path in the Windows command line, use the path command.

6.) We need to add the directory that contains the file **sspiauth.dll** to our Path variable. This file is located at ~\SASHome\SASFoundation\9.4\core\sasext. Search for this folder in your SASHome directory. Click Edit and add the path to this directory to the list.

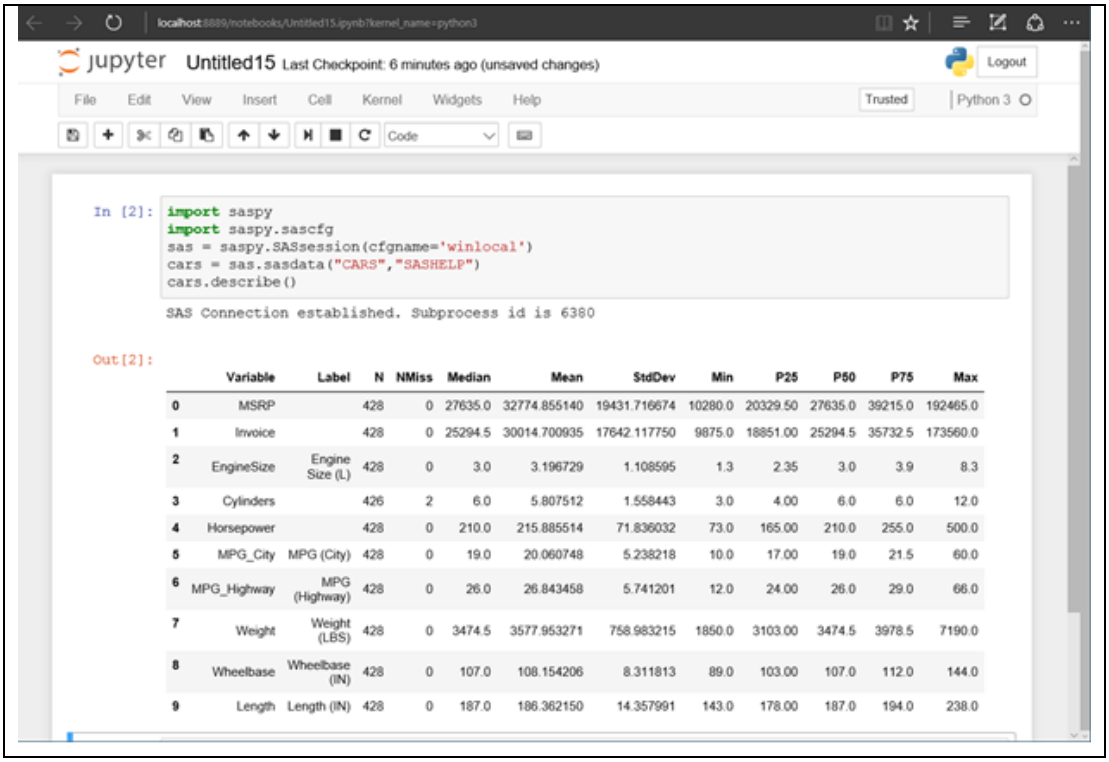
7.) If you open the cmd line in DOS and type path (i.e. > path) you will see something like (our edit is in red):

PATH=C:\Oracle\product\11.2.0\client_X64\bin;C:\ProgramData\Oracle\Java\javapath;C:\windows\system32;C:\windows;C:\windows\System32\Wbem;C:\windows\System32\WindowsPowerShell\v1.0\;C:\Program Files\ServiceNow;C:\Program Files\ServiceNow;F:\SAS\SASHome\SASFoundation\9.4\ets\sasexe;F:\SAS\SASHome\Secure\ccme4;F:\SAS\SASHome\SASFoundation\9.4\core\sasext;C:\Users\YOURWINDOWSNAME\AppData\Local\Microsoft\WindowsApps;

8.) Now update the sascfg.py file using IDLE. This file will be located in C:\Users\WINDOWSUSERNAME\AppData\Local\Continuum\anaconda3\Lib\site-packages\saspy. Right click and choose Open with IDLE (Python editor).

```
…scroll down to the config names below

SAS_config_names=['winlocal','winiomwin','winiomIWA']

…scroll down to the path files below

# build out a local classpath variable to use below for Windows clients
cpW  =
"F:\\SAS\\SASHome\\SASDeploymentManager\\9.4\\products\\deploywiz__94490__prt__xx__sp0__1\\deploywiz\\sas.svc.connection.jar"
cpW += ";F:\\SAS\\SASHome\\SASDeploymentManager\\9.4\\products\\deploywiz__94490__prt__xx__sp0__1\\deploywiz\\log4j.jar"
cpW +=
";F:\\SAS\\SASHome\\SASDeploymentManager\\9.4\\products\\deploywiz__94490__prt__xx__sp0__1\\deploywiz\\sas.security.sspi.jar"
cpW += ";F:\\SAS\\SASHome\\SASDeploymentManager\\9.4\\products\\deploywiz__94490__prt__xx__sp0__1\\deploywiz\\sas.core.jar"
cpW += ";C:\\ProgramData\\Anaconda3\\Lib\\site-packages\\saspy\\java\\saspyiom.jar"

…cpW is aliased as winlocal (see below)

        winlocal = {'java'     : 'java',
                    'encoding'  : 'windows-1252',
                    'classpath' : cpW
    }
```

9.) Test your connection by opening Anaconda > Jupyter > and type the following in a Jupyter “In” box:



	Variable	Label	N	NMiss	Median	Mean	StdDev	Min	P25	P50	P75	Max
0	MSRP		428	0	27635.0	32774.855140	19431.716674	10280.0	20329.50	27635.0	39215.0	192465.0
1	Invoice		428	0	25294.5	30014.700935	17642.117750	9875.0	18851.00	25294.5	35732.5	173560.0
2	EngineSize	Engine Size (L)	428	0	3.0	3.196729	1.108595	1.3	2.35	3.0	3.9	8.3
3	Cylinders		426	2	6.0	5.807512	1.558443	3.0	4.00	6.0	6.0	12.0
4	Horsepower		428	0	210.0	215.885514	71.836032	73.0	165.00	210.0	255.0	500.0
5	MPG_City	MPG (City)	428	0	19.0	20.060748	5.238218	10.0	17.00	19.0	21.5	60.0
6	MPG_Highway	MPG (Highway)	428	0	26.0	26.843458	5.741201	12.0	24.00	26.0	29.0	66.0
7	Weight	Weight (LBS)	428	0	3474.5	3577.953271	758.983215	1850.0	3103.00	3474.5	3978.5	7190.0
8	Wheelbase	Wheelbase (IN)	428	0	107.0	108.154206	8.311813	89.0	103.00	107.0	112.0	144.0
9	Length	Length (IN)	428	0	187.0	186.362150	14.357991	143.0	178.00	187.0	194.0	238.0

GOING FURTHER

It's even possible to add the SAS Kernel to your Jupyter Notebook and program in SAS from your Jupyter project:

1.) Go to your Anaconda DOS prompt and type "pip install sas_kernel". It will install the sas_kernel in C:\Users\YOURWINDOWSNAME\AppData\Local\Continuum\anaconda3\Lib\site-packages
2.) Once the install is complete, type "jupyter kernelspec list" you should see:

```
(base) C:\Users\YOURWINDOWSNAME>jupyter kernelspec list
Available kernels:
  sas       C:\Users\YOURWINDOWSNAME\AppData\Roaming\jupyter\kernels\sas
  python3   C:\Users\YOURWINDOWSNAME\AppData\Local\Continuum\anaconda3\share\jupyter\kernels\python3
```

3.) Now, from Jupyter create a new Code Cell. Select Cell Type 'SAS'. Type PROC setinit; run; CNTRL+RETURN. You will see Jupyter connect to the kernel, start a process, and return all of you licensing information. You can now use Jupyter to run BASE SAS using your PC.
4.) Note that Jupyter can only be connected to a single kernel at a given time, so you cannot have mixed Cells in you Jupyter notebook.

CONNECT SASPy TO YOUR SAS DATA

The following example demonstrates how to connect Python to local SAS data sets

```
#Import a SAS data set into python
import saspy
sas=saspy.SASsession(cfgname='winlocal')
# Python uses \\ notation for windows filepaths
sas.saslib("MYLIB", "BASE", "C:\\Users\\MYUSERNAME\\Desktop\\MachineLearning")
PYTHONDATA = sas.sasdata("TCADWCAD","MYLIB")

#Import a SAS data set into pandas
import pandas
PANDADATA= pandas.read_sas('C:/Users/MYUSERNAME/Desktop/MachineLearning/TCADWCAD.sas7bdat')
print(PANDADATA)
```

PERFORM MACHINE LEARNING ON A SAS DATA SET USING PYTHON

The data set in this example consists of public tax appraisal data. Here, I'm using Python packages sklearn, scipy, and numpy to train a model which predicts the Appraised Value based upon the sqft, year built, floors and zip code. I then apply the model to a small sample of test data. The SAS data set used to train the model is loaded using the statement:

TRAIN_X = pandas.read_sas('C:/Users/ MYUSERNAME /Desktop/MachineLearning/TRAIN.sas7bdat')

```
# Python3 MACHINE LEARNING EXAMPLE SCSUGS 2018
# Written by Michael McCarthy, PE
#
# IMPORT LIBRARIES NEEDED FOR MACHINE LEARNING
import pandas
# IMPORT NumPy
import numpy
# IMPORT SCIKIT-LEARN
import sklearn
# IMPORT scipy
import scipy
# IMPORT JOBLIB FOR PIPELINING
import joblib
# Import a SAS data set into python
import saspy
sas=saspy.SASsession(cfgname='winlocal')
```

```
# Python uses \\ notation for windows filepaths
sas.saslib("MYLIB", "BASE", "C:\\Users\\MYUSERNAME\\Desktop\\MachineLearning")
# DATA PREP ON SAS - Either run SAS Kernel in Jupyter or use BASE SAS editor
# libname dataprep 'C:\Users\MYUSERNAME\Desktop\MachineLearning';
# data dataprep.TRAIN (KEEP = STATECODE FLOORS YRBLT SQFT ZIP );
# set dataprep.TCADWCAD;
#   format ZIP 6.;
#  FLOORS = MAXFLOOR; YRBLT = MAXYRBLT; SQFT = TOTALSQFT; ZIP=STREETZIP;
#  if missing(MAXFLOOR)OR missing(MAXYRBLT) OR (TOTALSQFT LT 200) OR (TOTALSQFT GT 9000) OR ZIP LT 70000 OR ZIP GT 80000 Return;
#  if STATE_CD = 'A1' then output;
#  run;
# Pandas uses / notation for windows filepaths - Make sure no missing values in data set
TRAIN_X = pandas.read_sas('C:/Users/ MYUSERNAME /Desktop/MachineLearning/TRAIN.sas7bdat')
TRAIN_Y = pandas.read_sas('C:/Users/ MYUSERNAME /Desktop/MachineLearning/TRAIN.sas7bdat')

# You could also import csv directly into pandas
#TRAIN_X = pandas.read_csv("C:/Users/ MYUSERNAME /Desktop/MachineLearning/TRAIN.csv")
#TRAIN_Y = pandas.read_csv("C:/Users/ MYUSERNAME /Desktop/MachineLearning/TRAIN.csv")

# BUILD TRAINING DATA SETS
X = TRAIN_X[['FLOORS','YRBLT','SQFT','ZIP']].as_matrix()
print(X)
Y = TRAIN_Y['VALUE'].as_matrix()
print(Y)
# SPLIT THE DATA 70-30 INTO TRAINING AND TEST DATA SETS
# YOU MUST IMPORT INDIVIDUALLY MODELS FROM THE SCIKIT_LEARN PACKAGE
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.3, random_state=0)
# FIT REGRESSION MODEL
from sklearn import ensemble
model = ensemble.GradientBoostingRegressor(
   n_estimators = 10, # How many decision trees to build
   learning_rate=0.3, # how much each decision tree influences the model
   max_depth = 4, # How deep each individual decision tree can be
   min_samples_leaf = 10, # How often a value appears in the training set before it's used to make a decision
   max_features=0.1, # Number of features considered to create a branch in the decision tree
   loss = 'huber' # Controls how the Cost function is considered while the model learns
)
model.fit(X_train,Y_train) #CALL Sci-Kit Learn and train the model
# SAVE OUTPUT MODEL USING ANOTHER SCIKIE LEARN FUNCTION
joblib.dump(model,'trained_house_model.pkl')

#CHECK ACCURACY OF MODEL USING MEAN ABSOLUTE ERROR FUNCTION FOR EACH Y CALL PREDICTION
# FIND THE ERROR RATE ON THE TRAINING DATA SET
from sklearn.metrics import mean_absolute_error
mse = mean_absolute_error(Y_train,model.predict(X_train))
print("Training data set Mean Absolute Error: %.4f" % mse)

# FIND THE ERROR RATE ON THE TEST DATA SET
mse = mean_absolute_error(Y_test,model.predict(X_test))
print("Test data set Mean Absolute Error: %.4f" % mse)

# LOAD THE GRADIENT BOOSTING MODEL AND APPLY TO NEW DATA
model = joblib.load('trained_house_model.pkl')

# IMPORT REAL DATA
REAL_X  =  pandas.read_csv("C:/Users/MYUSERNAME/Desktop/MachineLearning/SCSUGS.csv",dtype={"FLOORS":  int,"YRBLT":  int,"SQFT":  int,"ZIP":  int
},low_memory=False)
# CONVERT TO MATRIX
X = REAL_X[['FLOORS','YRBLT','SQFT','ZIP']].as_matrix()
print(X)
# Pass in values from a data set we want to model - Call Model.predict
Predicted_Home_Values = model.predict(X)
print(Predicted_Home_Values)
predicted_value = Predicted_Home_Values[0]
print("The estimated value is ${:.2f}".format(predicted_value))
mylist = REAL_X.ID.tolist()

# OUTPUT PREDICTED HOUSE PRICES
for n in mylist:
   Predicted_Home_Values = model.predict(X)
   predicted_value = Predicted_Home_Values[n-1]
   print("The estimated value is ${:.2f}".format(predicted_value))
```

CONCLUSION

The instructions presented in this paper should get SASPy up and running on Python3 with Jupyter Notebooks. I chose to install the Anaconda distribution since the Python package installation and configuration is automated. This paper presents a step-by-step tutorial showing the configuration of the connection from BASE SAS® to an Anaconda Distribution of Python3 running on Windows 10. In this paper I demonstrated the use of Python3 to perform machine learning on an existing SAS data set using packages Scikit-learn, SciPy, and NumPy. Happy coding!

ACKNOWLEDGEMENTS

Special thanks to John Trowbridge, PE (DABI Engineering Supervisor, Austin Energy), Liz Jambor (DABI Manager, Austin Energy), Victor Carr (Division Manager of Information Systems, Austin Energy) and Debbie Kimberly (VP Customer Energy Solutions, Austin Energy) for their continued support of our SAS Data Analytics Server.

REFERENCES

SASPy Documentation
https://sassoftware.github.io/saspy/

Paper 2822-2018 A Basic Introduction to SASPy and Jupyter Notebooks
Jason Phillips, PhD, University of Alabama
https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2018/2822-2018.pdf

CONTACT INFORMATION

I value your comments. You can contact the author at:

Michael McCarthy, PE
Austin Energy, Customer Energy Solutions, Austin TX USA
michael.mccarthy@austinenergy.com