
▼ Everything is Better with Friends

Using SAS in Python Applications with SASPy and Open-Source Tooling (Beyond the Basics)

▼ Setup for Part 1

Getting setup to use Google Colab with SAS OnDemand for Academics (ODA)

1. To execute code cells, you'll need credentials for the following accounts:

- Google. (If you're not already signed in, you should see a **Sign In** button in the upper right corner. You can also visit <https://accounts.google.com/signup> to create an account for free.)
- SAS OnDemand for Academics. (You can create an account for free at <https://welcome.oda.sas.com/> using an existing SAS Profile account. If you don't already have a SAS Profile account, you can create one for free using the "Don't have a SAS Profile?" link on the ODA login page.)

2. We recommend enabling line numbers using the Tools menu: **Tools -> Settings -> Editor -> Show line numbers -> Save**

3. We also recommend enabling the Table of Contents using the View menu: **View -> Table of contents**

4. To save a copy of this notebook, along with any edits you make, please use the File menu: **File -> Save a copy in Drive**

5. Looking for "extra credit"? Please let us know if you spot any typos!

▼ Connect to SAS OnDemand for Academics (ODA) and start a SAS session

Instructions:

1. Determine the Region for your ODA account by logging into <https://welcome.oda.sas.com/>. You should see a value like `Asia Pacific 1`, `Asia Pacific 2`, `Europe 1`, `United States 1`, or `United States 2` next to your username in the upper-right corner. (For more information about Regions and using Python in Jupyter Notebooks, please see the ODA documentation at https://support.sas.com/ondemand/cag_new.html#region and <https://support.sas.com/ondemand/saspy.html>.)
2. If your ODA account is associated with a Region other than `United States 1`, comment out Line 11 by adding a number sign (`#`) at the beginning of the line, and then uncomment the list of servers corresponding to your Region.

Note: As of the time of creation of this Notebook, only the Regions listed below were available. If your SAS ODA account is associated with a Region that's not listed, you will need to manually add the appropriate servers.
3. Click anywhere in the code cell, and run the cell using Shift-Enter.
4. At the prompt `Please enter the OMR user id`, enter either your SAS ODA user ID or the email address associated with your ODA account.
5. At the prompt `Please enter the password for OMR user`, enter the password for your SAS ODA account.

```
1 !pip install saspy
2
3 import saspy
4
5 sas = saspy.SASsession(
6     java='/usr/bin/java',
7     iomport=8591,
8     encoding='utf-8',
9
10    # For Region "United States 1", uncomment the line below.
11    iomhost = ['odaws01-usw2.oda.sas.com', 'odaws02-usw2.oda.sas.com', 'odaws03-usw2.oda.sas.com', 'odaws04-usw2.
12
13    # For Region "United States 2", uncomment the line below.
14    #iomhost = ['odaws01-usw2-2.oda.sas.com', 'odaws02-usw2-2.oda.sas.com'],
15
16    # For Region "Europe 1", uncomment the line below.
17    #iomhost = ['odaws01-euw1.oda.sas.com', 'odaws02-euw1.oda.sas.com'],
18
19    # For Region "Asia Pacific 1", uncomment the line below.
20    #iomhost = ['odaws01-apse1.oda.sas.com', 'odaws02-apse1.oda.sas.com'],
```

```

21
22     # For Region "Asia Pacific 2", uncomment the line below.
23     #iomhost = ['odaws01-apsel-2.oda.sas.com', 'odaws02-apsel-2.oda.sas.com'],
24
25 )
26 print(sas)

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting saspy
  Downloading saspy-4.3.2.tar.gz (9.9 MB)
    |████████████████████████████████████████| 9.9 MB 14.6 MB/s
Building wheels for collected packages: saspy
  Building wheel for saspy (setup.py) ... done
  Created wheel for saspy: filename=saspy-4.3.2-py3-none-any.whl size=9929529 sha256=ba37588fb0c94435eae91beb
  Stored in directory: /root/.cache/pip/wheels/55/a2/91/45db2a8ca68bcb2ee02c28366dc2c36d40e1af670a9ba96e12
Successfully built saspy
Installing collected packages: saspy
Successfully installed saspy-4.3.2
Using SAS Config named: default
Please enter the OMR user id: isaiah.lankham@ucop.edu
Please enter the password for OMR user : .....
SAS Connection established. Subprocess id is 164

Access Method           = IOM
SAS Config name         = default
SAS Config file         = /usr/local/lib/python3.7/dist-packages/saspy/sascfg.py
WORK Path               = /saswork/SAS_work08A100009D60_odaws01-usw2.oda.sas.com/SAS_work3BBB00009D60_odaws01-u
SAS Version             = 9.04.01M6P11072018
SASPy Version          = 4.3.2
Teach me SAS            = False
Batch                   = False
Results                 = Pandas
SAS Session Encoding    = utf-8
Python Encoding value   = utf-8
SAS process Pid value   = 40288

```

Note: This establishes a connection from Python in Google Colab to a SAS session running in SAS ODA.

▼ Install and import additional packages

```
1 # Install the rich module for colorful printing
2 !pip install rich
3
4 # We'll use IPython to display DataFrames or HTML content
5 from IPython.display import display, HTML
6
7 # We're overwriting the default print function with rich.print
8 from rich import print
9
10 # We're also setting the maximum line width of rich.print to be a bit wider (to avoid line wrapping)
11 from rich import get_console
12 console = get_console()
13 console.width = 165
```

Looking in indexes: <https://pypi.org/simple>, <https://us-python.pkg.dev/colab-wheels/public/simple/>

Collecting rich

Downloading rich-12.5.1-py3-none-any.whl (235 kB)

|██| 235 kB 14.5 MB/s

Requirement already satisfied: pygments<3.0.0,>=2.6.0 in /usr/local/lib/python3.7/dist-packages (from rich) (

Collecting commonmark<0.10.0,>=0.9.0

Downloading commonmark-0.9.1-py2.py3-none-any.whl (51 kB)

|██| 51 kB 7.0 MB/s

Requirement already satisfied: typing-extensions<5.0,>=4.0.0 in /usr/local/lib/python3.7/dist-packages (from

Installing collected packages: commonmark, rich

Successfully installed commonmark-0.9.1 rich-12.5.1

▼ Part 1. Calling SAS/STAT procedures in Python applications

▼ Section 1.1. Create titanic_sds


```
1 # I'd rather not have spaces and slashes in my SAS variable names.
2 sas.submit('options validvarname=v7;')
3
4 # Read the titanic dataset into SAS from a URL.
5 titanic_url = 'https://web.stanford.edu/class/archive/cs/cs109/cs109.1166/stuff/titanic.csv'
6 titanic_sds = sas.read_csv(file=titanic_url,table='titanic',libref='work')
7
8 # What kind of object is titanic_sds?
9 print(type(titanic_sds))
10 print('\n')
11 print(titanic_sds)
12
13 # Curious what's under the hood?
14 print('Here\'s the SAS code being used by SASPy to import the dataset:')
15 sas.teach_me_SAS(True)
16 sas.read_csv(file=titanic_url,table='titanic',libref='work')
17 sas.teach_me_SAS(False)
18 print('\n')
19
20 # Let's take a look at the data.
21 display(titanic_sds.columnInfo())
22 display(titanic_sds.head())
```

```
<class 'saspy.sasdata.SASdata'>
```

```
Libref = work
Table = titanic
Dsopts = {}
Results = Pandas
```

Here's the SAS code being used by SASPy to import the dataset:

```
filename x url "https://web.stanford.edu/class/archive/cs/cs109/cs109.1166/stuff/titanic.csv";
proc import datafile=x out=work.'titanic' dbms=csv replace; run;
```

	Member	Num	Variable	Type	Len	Pos	Format	Informat	
0	WORK.TITANIC	5.0	Age	Num	8.0	16.0	BEST12.	BEST32.	
1	WORK.TITANIC	8.0	Fare	Num	8.0	40.0	BEST12.	BEST32.	
2	WORK.TITANIC	3.0	Name	Char	54.0	48.0	\$54.	\$54.	
3	WORK.TITANIC	7.0	Parents_Children_Aboard	Num	8.0	32.0	BEST12.	BEST32.	
4	WORK.TITANIC	2.0	Pclass	Num	8.0	8.0	BEST12.	BEST32.	

Concept Check 1.1

- Try this, and see what happens: Comment out Line 17 so that `sas.teach_me_SAS` is no longer turned off, and then rerun the code cell above.
- True or False: Commenting out Line 17, so that `sas.teach_me_SAS` is left on, has no effect on subsequent code.
- Fun Fact: The SASdata object `titanic_sds` does not represent a dataset kept in memory in our local Google Colab session. Instead, it's a pointer to normal SAS dataset file kept on disk in the remote SAS ODA session.

```
2      1.0      3.0      MISS. Laina Heikkinen      female      20.0      0.0
```

Solution: False! If we don't turn "Teach Me SAS" off, Lines 21-22 won't execute.

```
4      0.0      3.0      Mr William Henry Allen      male      35.0      0.0
```

▼ Section 1.2. Create titanic_partitions

```
1 # Make sure sas.teach_me_SAS is set to False.
```

```

2 sas.teach_me_SAS(False)
3
4 # Let's partition the dataset into subsets for training and testing a predictive model.
5 titanic_partitions = titanic_sds.partition(seed=42, singleOut=False)
6
7 # And then print some information about each subset.
8 print(type(titanic_partitions))
9 print('\n')
10 print(titanic_partitions)
11 print('\n')
12 print(f'{titanic_partitions[0].obs()} observations for training')
13 print(f'{titanic_partitions[1].obs()} observations for test')

```

```
<class 'tuple'>
```

```

(Libref = work
Table   = titanic1_train
Dsopts  =
Results = Pandas
, Libref = work
Table   = titanic1_score
Dsopts  =
Results = Pandas
)

```

```

621 observations for training
266 observations for test

```

Concept Check 1.2

- Try this, and see what happens: Try using `sas.teach_me_SAS` to see the SAS code generated by the `partition` method on Line 5.
- True or False: In order to make sure `titanic_partitions` is created as expected (without a Traceback, a.k.a., a run-time error), Line 5 needs to be run with `sas.teach_me_SAS` turned off.
- Fun Fact: A `tuple` is like a fixed-length `list`. Whereas a `list` can be changed in place, and can grow and shrink in size, a `tuple` is immutable. This means any attempt to change a `tuple` actually results in a new `tuple` being created, instead. (Other examples of immutable Python objects include `int` and `str`.)

Solution: False! If we don't turn "Teach Me SAS" off, Lines 8-13 will result in Traceback errors since the variable `titanic_partitions` won't have been created.

▼ Section 1.3. Create `titanic_model`

```
1 # Make sure sas.teach_me_SAS is set to False (just in case).
2 sas.teach_me_SAS(False)
3
4 # Now let's create an object that will allow us to access SAS/STAT procedures.
5 sas_stat = sas.sasstat()
6
7 # We'll also set our response variables and explanatory variables.
8 outcome = "survived(event='1')"
9 covariates = 'pclass sex age siblings_spouses_aboard parents_children_aboard fare'
10
11 # We're now ready to use PROC LOGISTIC to train a model and score the test dataset.
12 titanic_model = sas_stat.logistic(
13     data = titanic_partitions[0],
14     cls = 'sex',
15     model = f'{outcome} = {covariates}',
16     stmtpassthrough = f'score data={titanic_partitions[1].table} out=scored fitstat',
17     procopts = 'plots=none',
18 )
19
20 # Let's check the SAS log to see how everything went.
21 display(titanic_model.LOG)
22
23 # What else is in this titanic_model object?
24 print(dir(titanic_model))
```


63		The SAS System	Monday, Septemb
408	options nosource;		
536			
64		The SAS System	Monday, Septemb
537			
538	%macro proccall(d);		
539	proc logistic data=work.'titanic1_train'n plot=all plots=none ;		
540	class sex;		
541	model survived(event='1') = pclass sex age siblings_spouses_aboard parents_children_aboard fare;		
542	score data=titanic1_score out=scored fitstat ;		
543	run; quit; %mend;		
544	%mangobj(log0001,logistic,'titanic1_train'n);		
548			
549			
550			
65		The SAS System	Monday, Septemb
551			
	[
	'ASSOCIATION',		
	'CLASSLEVELINFO',		
	'CONVERGENCESTATUS',		
	'FITSTATISTICS',		
	'GLOBALTESTS',		
	'LOG',		
	'MODELANOVA',		
	'MODELINFO',		
	'NOBS',		

Concept Check 1.3

- Try this, and see what happens: Change LOG on Line 21 to any of the object names in the list that was output by Line 24.
- True or False: The Python object `titanic_model` has 13 sub-objects nested inside of it.
- Fun Fact: The strings `"survived(event='1')"` and `'survived(event="1")'` produce identical behavior in Python.

Solution: True! There are 13 items in the list printed out by Line 24.

▼ Section 1.4. Create train_auc and test_auc

```
1 # But how does the model perform?
2 display(titanic_model.ASSOCIATION)
3 display(titanic_model.SCOREFITSTAT)
4
5 # Let's pull out the AUC value for training.
6 train_auc_srs = titanic_model.ASSOCIATION.loc[titanic_model.ASSOCIATION['Label2'] == 'c', 'nValue2']
7 print('\n')
8 print(type(train_auc_srs))
9 print('\n')
10 print(train_auc_srs)
11 train_auc = train_auc_srs.iloc[0]
12
13 # Let's also pull out the AUC value for test.
14 test_auc = titanic_model.SCOREFITSTAT['AUC'][0]
15
16 # And, finally, let's compare them.
17 print('\n')
18 print(f'Training AUC: {train_auc:.4f}, Test AUC: {test_auc:.4f}')
```

	Label1	cValue1	nValue1	Label2	cValue2	nValue2
0	Percent Concordant	86.5	86.532990	Somers' D	0.731	0.730761
1	Percent Discordant	13.5	13.456928	Gamma	0.731	0.730834
2	Percent Tied	0.0	0.010082	Tau-a	0.339	0.338866



Concept Check 1.4

- Short Answer: How would you access specific cells of a dataset in SAS?
- Fun Facts:
 - A DataFrame in Python is a tabular data structure with rows and columns, similar to a SAS data set. The columns of a DataFrame are called `series`.
 - While SAS datasets are typically accessed from disk and processed row-by-row, DataFrames are loaded into memory all at once. This means values in DataFrames can be randomly accessed (like on Line 6 above), but it also means the size of DataFrames can't grow beyond available memory.
 - Problem solving always involves trade-offs. Python provides flexible access to in-memory data, whereas SAS makes it possible to work with datasets whose size exceeds available memory.

Solution: We can subset a SAS dataset to specific rows and columns with KEEP, DROP, and WHERE dataset options, or in a SQL query with SELECT, WHERE, and HAVING clauses. To pull out a single value, we might store it in a macro variable with CALL SYMPUTX in a DATA step, or an INTO expression in PROC SQL.

▼ Section 1.5. Alternate ways to create train_auc

```
1 # We could simplify getting the training AUC by just hard-coding its row index.
2 train_auc = titanic_model.ASSOCIATION['nValue2'][3]
3 print(train_auc)
```

0.86538030693402

```
1 # Or we can create an index on the Label2 column and get the best of both worlds.
2 indexed_association = titanic_model.ASSOCIATION.set_index('Label2')
3 train_auc = indexed_association['nValue2']['c']
4 print(train_auc)
```

0.86538030693402

Concept Check 1.5

- Multiple choice: What do you think is the best way to pull out the AUC value from the ASSOCIATION attribute of `titanic_model`?
 - A. Using `loc` to subset on the value of a column.
 - B. Using column labels and integer row indices.
 - C. Using `set_index` to create a non-integer index.
- Fun Fact: This exercise would seem to contradict the often-quoted aphorism "There should be one-- and preferably only one -- obvious way to do it" from the [Zen of Python](#).

Solution: The authors prefers option C (`set_index`), but there are merits to all three methods.

▼ Section 1.6. Additional Exercises

For practice, we recommend the following:

1. Run the code in the cell below (taken from Section 1.4).
2. Modify the code to instead pull the `Percent Concordant` value out of `titanic_model.ASSOCIATION`.
3. Use the `display` method to view the contents of `titanic_model.PARAMETERESTIMATES`.
4. Finally, pull a single value out of `titanic_model.PARAMETERESTIMATES`. (E.g., you might try getting the parameter estimate for the variable `Age`, which you can do using any of the three methods discussed above.)

```

1 display(titanic_model.ASSOCIATION)
2 train_auc_srs = titanic_model.ASSOCIATION.loc[titanic_model.ASSOCIATION['Label2'] == 'c', 'nValue2']
3 print(train_auc_srs)

```

	Label1	cValue1	nValue1	Label2	cValue2	nValue2
0	Percent Concordant	86.5	86.532990	Somers' D	0.731	0.730761
1	Percent Discordant	13.5	13.456928	Gamma	0.731	0.730834
2	Percent Tied	0.0	0.010082	Tau-a	0.339	0.338866
3	Pairs	89270	89270.000000	c	0.865	0.865380

3 0.86538

Name: nValue2, dtype: float64

```

1 display(titanic_model.ASSOCIATION)
2 train_percent_concordant_srs = titanic_model.ASSOCIATION.loc[titanic_model.ASSOCIATION['Label1'] == 'Percent Concordant', 'nValue2']
3 print(train_percent_concordant_srs)
4
5 display(titanic_model.PARAMETERESTIMATES)
6 age_parameter_estimate = titanic_model.PARAMETERESTIMATES.loc[titanic_model.PARAMETERESTIMATES['Variable'] == 'Age', 'Estimate']
7 print(age_parameter_estimate)

```



	Label1	cValue1	nValue1	Label2	cValue2	nValue2	
0	Percent Concordant	86.5	86.532990	Somers' D	0.731	0.730761	
1	Percent Discordant	13.5	13.456928	Gamma	0.731	0.730834	
2	Percent Tied	0.0	0.010082	Tau-a	0.339	0.338866	
3	Pairs	89270	89270.000000	c	0.865	0.865380	

0 86.53299
Name: nValue1, dtype: float64

Variable	ClassVal0	DF	Estimate	StdErr	WaldChiSq	ProbChiSq	_ESTTYPE_
----------	-----------	----	----------	--------	-----------	-----------	-----------

▼ Notes and Resources

1. For more about the `pandas` package, including the methods used above, see the following:

- <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.loc.html>
- <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.iloc.html>
- <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop.html>
- https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.set_index.html
- <https://pandas.pydata.org/docs/reference/api/pandas.Series.html>

2. For more about the `rich` package, see <https://rich.readthedocs.io/>

3. For more about the `saspy` package, including the methods used above, see the following:

- <https://sassoftware.github.io/saspy/api.html#saspy.sasdata.SASdata.columnInfo>
- <https://sassoftware.github.io/saspy/api.html#saspy.sasdata.SASdata.head>
- <https://sassoftware.github.io/saspy/api.html#saspy.sasdata.SASdata.obs>
- <https://sassoftware.github.io/saspy/api.html#saspy.sasdata.SASdata.partition>
- <https://sassoftware.github.io/saspy/api.html#saspy.SASsession.lastlog>

- https://sassoftware.github.io/saspy/api.html#saspy.SASsession.read_csv
- <https://sassoftware.github.io/saspy/api.html#saspy.sasstat.SASstat>
- <https://sassoftware.github.io/saspy/api.html#saspy.sasstat.SASstat.logistic>

4. For more information on built-in Python data structures, such as tuples, see <https://jakevdp.github.io/WhirlwindTourOfPython/06-built-in-data-structures.html>.
5. For more information on f-strings (i.e., Python strings like `f'{outcome} = {covariates}'`), see <https://realpython.com/python-f-strings/>.
6. We welcome follow-up conversations. You can connect with us on LinkedIn or email us at isaiah.lankham@gmail.com and matthew.t.slaughter@gmail.com
7. If you have a GitHub account (or don't mind creating one), you can also chat with us on Gitter at <https://gitter.im/saspy-bffs/community>