



SAS Cortex

Coding Version Guide



Table of Contents

Disclaimer.....	3
What is Cortex?.....	4
Fundraising Scenario.....	4
How to play Cortex in SAS OnDemand for Academics (ODA).....	6
How to play Cortex in Python Command.....	7
Install SASPy on Windows.....	8
Install SASPy on MAC.....	16
Access to Dataset.....	21
Round 1 - Game Summary	23
Round 1 - Model Description (SAS).....	23
Step 1: Data Merge	23
Step 2: Data Partition.....	24
Step 3: Model Building.....	24
Step 4: Model Comparison	25
Step 5: Scoring Data.....	25
Step 6: Export Results	25
Round 1 - Model Description (Python).....	26
Step 1: Data Merge	26
Step 2: Data Partition.....	26
Step 3: Model Building.....	27
Step 4: Model Comparison	28
Step 5: Scoring Data.....	28
Step 6: Export Results	28
Round 1 - Upload Decisions	29
Round 2 - Game Summary	35
Round 2 – Amount Model Description (SAS)	36
Step 1: Data Merge	36
Step 2: Data Partition.....	37
Step 3: Model Building.....	37
Step 4: Model Comparison	38
Step 5: Scoring Data.....	38
Step 6: Export Results	39

Round 2 – Probability Model Description (SAS).....	39
Step 1: Data Merge	40
Step 2: Data Partition.....	40
Step 3: Model Building.....	41
Step 4: Model Comparison	41
Step 5: Scoring Data.....	42
Step 6: Export Results	42
Round 2 - Amount Model Description (Python)	43
Step 1: Data Merge	43
Step 2: Data Partition.....	43
Step 3: Model Building.....	44
Step 4: Model Comparison	45
Step 5: Scoring Data.....	45
Step 6: Export Results	45
Round 2 - Amount Model Description (Python)	46
Step 1: Data Merge	46
Step 2: Data Partition.....	46
Step 3: Model Building.....	47
Step 4: Model Comparison	48
Step 5: Scoring Data.....	48
Step 6: Export Results	48
Round 2 - Upload Decisions	49

Disclaimer

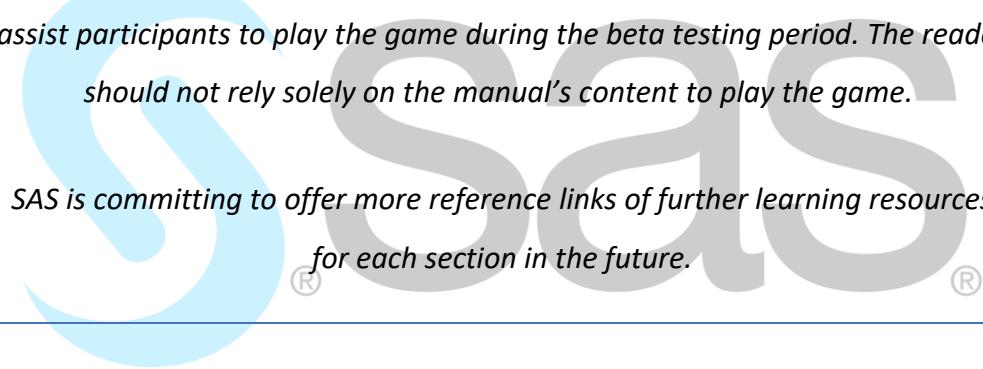
This user manual was developed by the Cortex team of SAS employees.

It is intended solely for the use of beta testing and may not be provided to any other person or entity without the express written consent of SAS Institute.

While every effort was made to ensure accuracy and completeness, neither SAS nor the report authors are able to warrant the degree of accuracy or completeness of this manual.

This user guideline was prepared on a best effort basis and is only intended to assist participants to play the game during the beta testing period. The reader should not rely solely on the manual's content to play the game.

SAS is committing to offer more reference links of further learning resources for each section in the future.

A large, semi-transparent watermark of the SAS logo is centered on the page. It features the word "SAS" in a bold, serif font, with a registered trademark symbol (®) at the end. The letters are partially obscured by a blue swoosh graphic on the left side.

What is Cortex?

SAS and ERPsim Lab at HEC Montreal partnered to develop Cortex, an analytics simulation game, which is designed to help participants apply and consolidate different data analytics concepts in realistic, business settings. Cortex currently offers a business scenario called the Fundraising Scenario.

In the Fundraising Scenario, participants will target potential donors to maximize the donations to the fundraising campaign. This simulation game provides pedagogical flexibility and is adaptable to students in both undergrad and graduate programs.

The Fundraising Scenario provides a complete experience that can be used in any teaching context with participants of varying levels of statistical knowledge and competency. Its hands-on approach helps the development of skills with tools commonly used in the industry.

The game is available for both Academic and Commercial use.



The fundraising scenario will place participants in a context where they will be working on a fundraising campaign for a foundation, which is a 12-year-old, not-for-profit charitable organization with a million members.

The foundation has decided to add a direct contact campaign to its list of marketing activities. Participants will predict how many and which individuals to target in the campaign. The objective is to fundraise the highest donation amount while managing the expenses of contacting donors.

Fundraising Scenario

Last Update date: November 27, 2020

Figure 1. Fundraising Scenario

Goal*:
Maximize the net raised funds

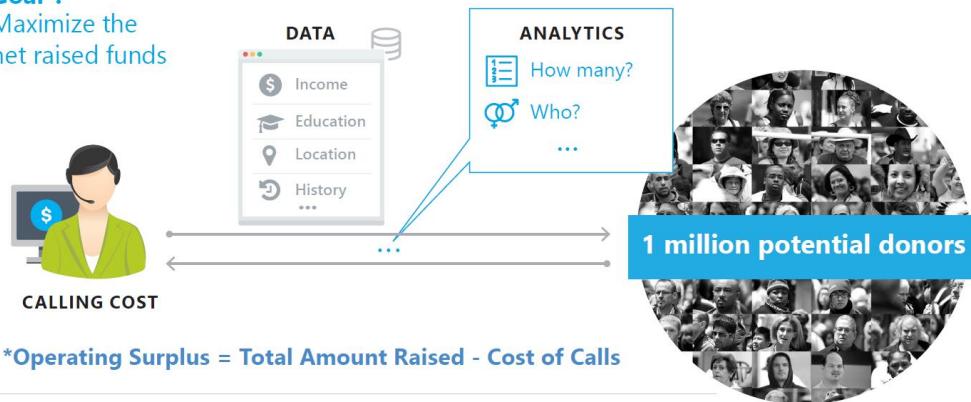


Table 1. Cost Structure

Number of contacted members	Cost per call
0 - 60,000	5\$/person
> 60,000	25\$/person

Table 2. List of variables

Variable Name	Description
ID	Member number (unique ID)
LastName	Last Name
FirstName	First Name
Woman	Sex (1=woman, 0=man)
Age	Age (years)
Salary	Annual salary in USD
Education	Highest education level
City	Type of neighborhood
SeniorList	Seniority for being on the VIP list
NbActivities	Number of participations to annual meeting
Referrals	Number of referrals
Recency	Number of years since last gift
Frequency	Number of donations
Seniority	Number of years since first donation
TotalGift	Total Donation since a member
MinGift	Minimum donation since a member
MaxGift	Maximum donation since on the VIP list
Contact	Direct solicitation this year (Only applicable to Round 2)
GaveLastYear	Whether or not the individual gave last year
AmtLastYear	Amount given last year
GaveThisYear	Whether or not the individual gave this year
AmtThisYear	Amount given this year

© ERPsim Lab, HEC Montréal, 2017-2020.

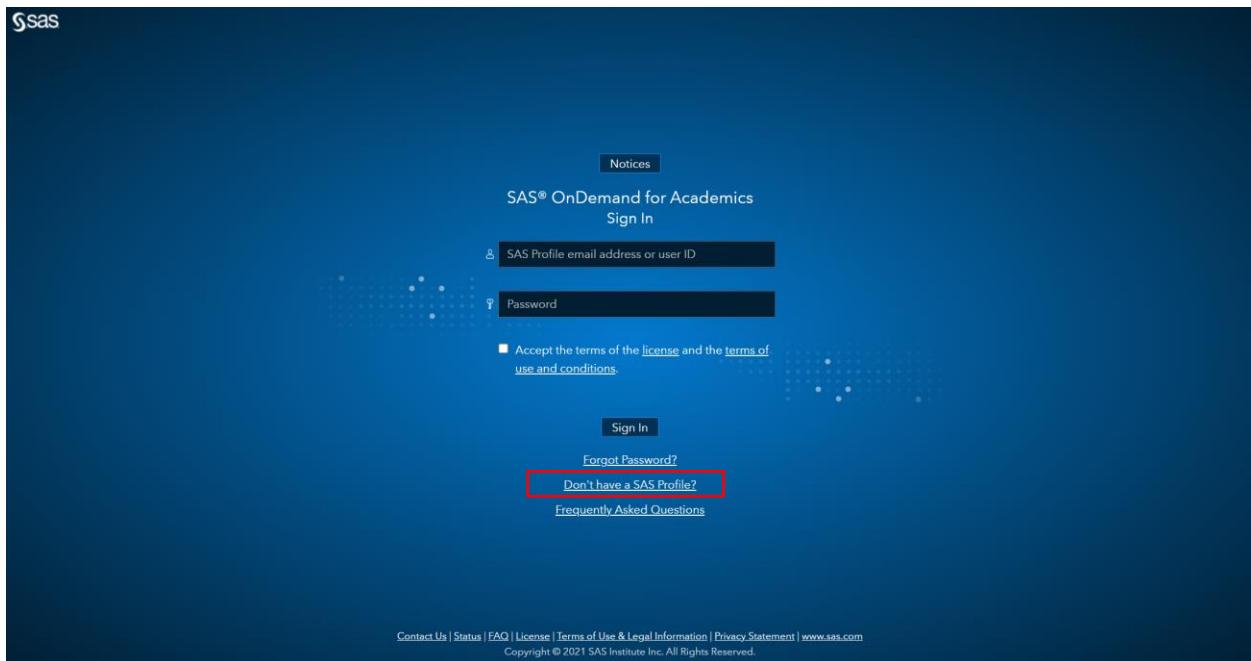
* History (HIST) dataset gives the history of 10 years leading up to, but excluding, last year.

PAGE 1

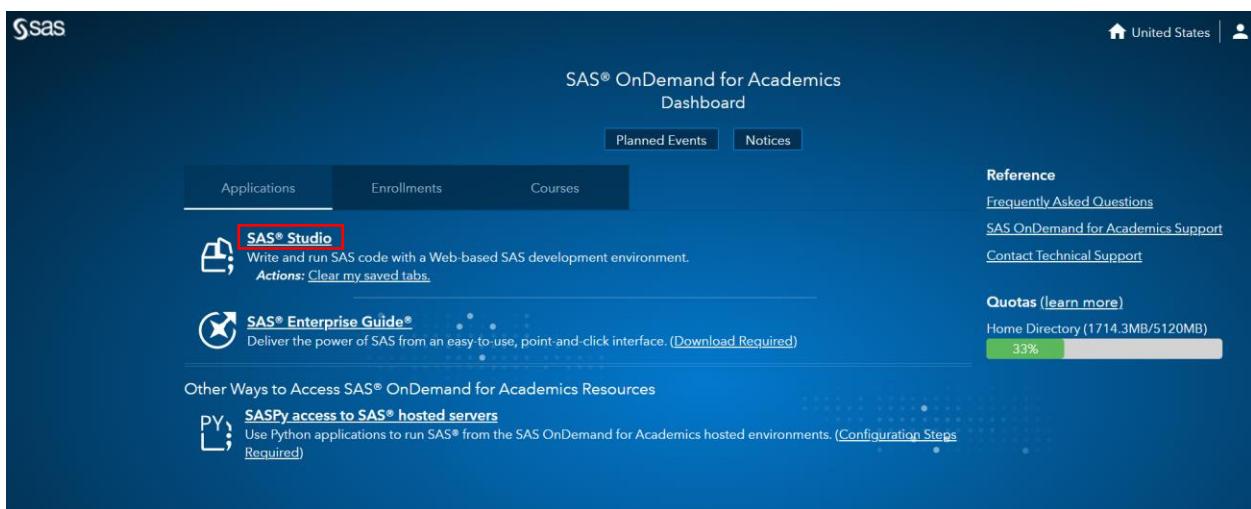
Source: https://erpsim.hec.ca/sites/default/files/cortex/GameSummary_Fundraising.pdf

How to play Cortex in SAS OnDemand for Academics (ODA)

You can play the game by coding the projection model in SAS at SAS Studio. As a first step, please create your account for ODA. To register, visit <https://odamid.oda.sas.com> and click on “**Don't have a SAS Profile?**” to create your profile and register for an account.



After you successfully create your account, you should be able to sign on the Control Center at <https://odamid.oda.sas.com>, and click “SAS Studio” to start coding.



You will be provided a baseline model of SAS code to start with, you can modify the code to achieve the goal of the game – maximizing the donation amount.

```

1 libname cortex '//home/u58555463/my_shared_file_links/u39842936';
2 libname results '/home/u58555463/results';
3
4 /*Round 1 Amount*/
5
6 /*merge dataset hist and target+rd1*/
7 data model_rdi;
8   MERGE cortex.hist cortex.target_rdi;
9   BY ID;
10  run;
11
12 data results.model_rdi; /*drop na*/
13 set model_rdi;
14 if not missing(_numeric_);
15 run;
16
17 /*data partition*/
18 proc surveyselect data=results.model_rdi rate=0.6
19   method=rsr
20   seed=1234;
21   select outall;
22   run;
23
24 data results.train_rdi results.test_rdi;
25   set donor_select;
26   if selected =1 then output results.train_rdi;
27   else output results.test_rdi;
28   run;
29
30 /*glm model*/
31 ods graphics off;
32 proc glmselect data=results.train_rdi testdata=results.test_rdi;
33 model AmtThisYear=Age Salary Seniority GameLastYear;
34

```

How to play Cortex in Python Command

You can play the game by coding the projection model in Python. You will need to first establish connection between ODA and Python Command in order to access datasets in SAS terminal, more details are listed below. We will provide the baseline model of Python code in Jupyter Notebook. Python players will be expected to install Jupyter Notebook to continue the game.

After you successfully open the Jupyter Notebook, you can download the Python code of the baseline model from [Github](#), you can modify the code to achieve the goal of the game – maximizing the donation amount.

```

from saspy import SASsession
sas_session = SASsession()
sas_session
Using SAS Config named: oda

#X$SAS sas_session
libname cortex '//home/u58555463/my_shared_file_links/u39842936';
libname results '/home/u58555463/results';
run;

import pandas as pd

#comment: bring to cloud sas dataset to python datafrmae(pandas) ==> take a while
data1 = sas_session.sasdata2dataframe(
    table='hist',
    libref='cortex'
)

data2 = sas_session.sasdata2dataframe(
    table='target_rdi',
    libref='cortex'
)

data1 Merge the Data
data_merge = pd.merge(data1, data2, on=["ID"], how="right")
data_merge.head()
#Deal with Missing Value
data_merge = data_merge.dropna() #comment: maybe another method for missing data, check it later
data_merge.head()

#Step2 Data Partition
#this is just a sample, you could use another library or built in funciton
from sklearn.model_selection import train_test_split
train, validation = train_test_split(data_merge, test_size=0.4) #you can change the percentage
train.head()

```

SASPy Installation and Configuration

To install the SASPy in your Python Command, you need to ensure that you meet the prerequisite:

1. Java version 1.8.0_162 or higher
2. Python 3.3 or higher
3. SASPy 3.3.4 or higher

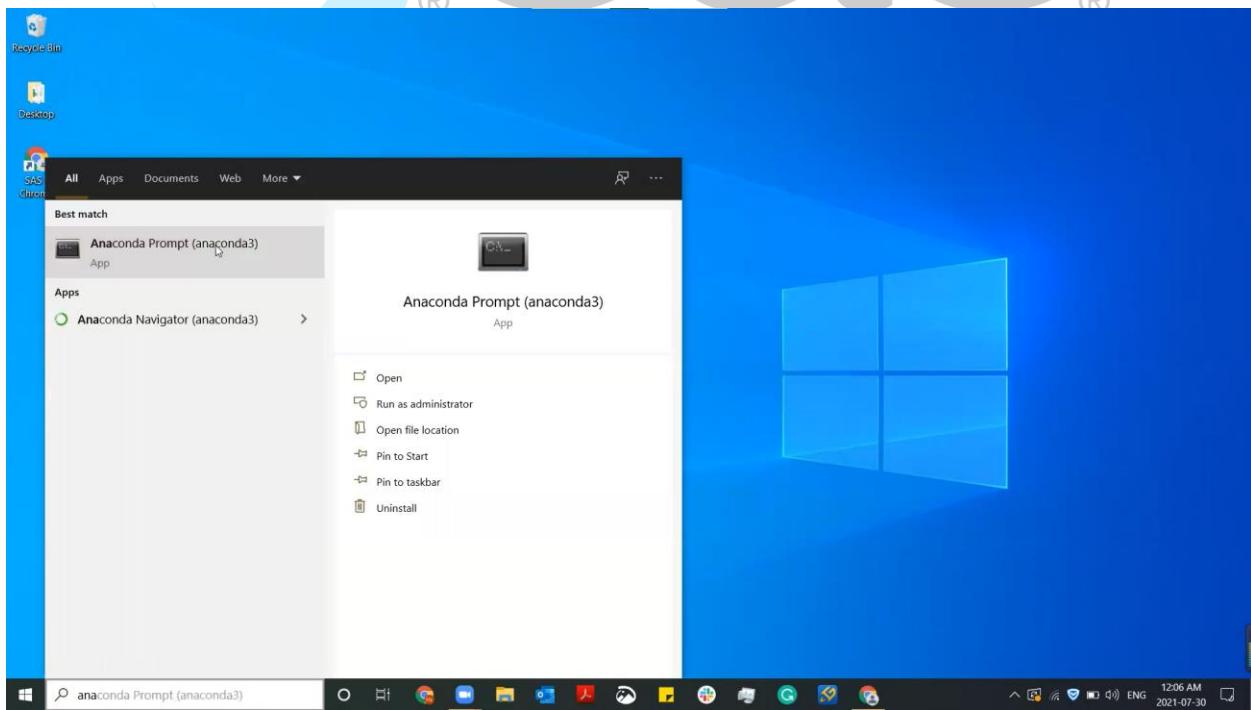
If you do not have Java installed, click [here](#) to download and install the newest version of Java.

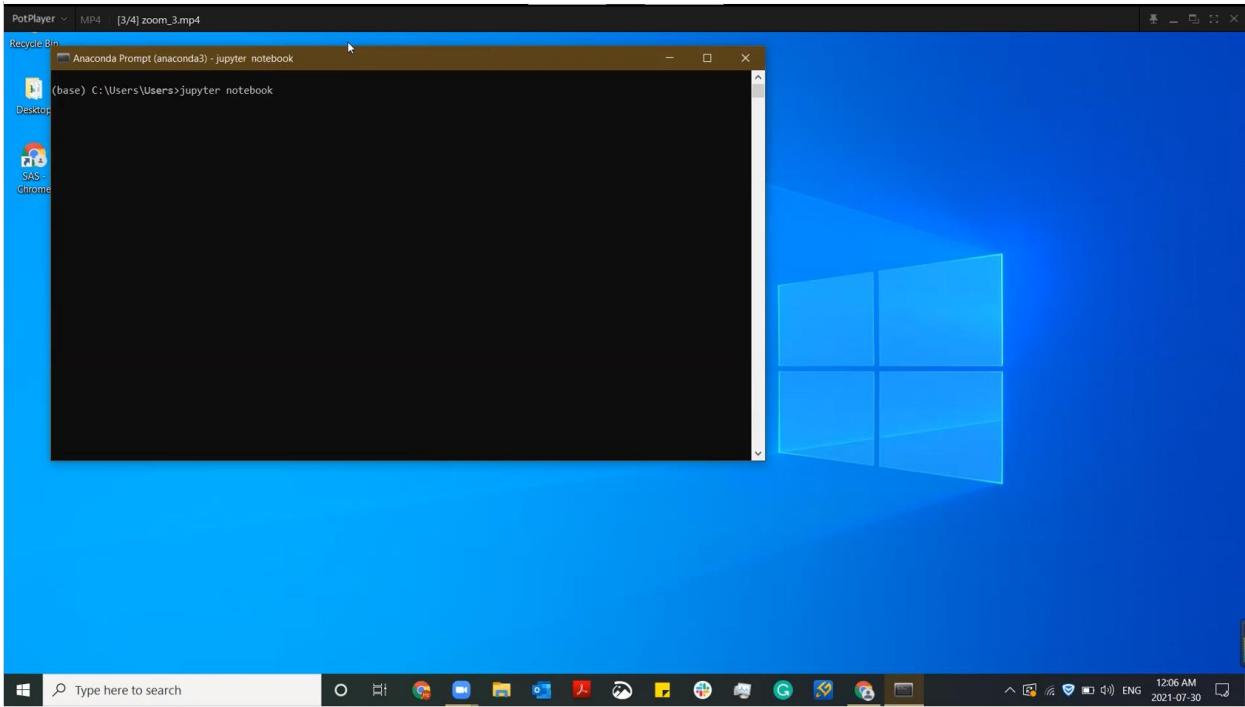
If you do not have Jupyter Notebook installed, click [here](#) to download and install the newest version of Anaconda Prompt and open Jupyter Notebook.

You can also click [here](#) to see the instructional videos of SASPy installation.

Install SASPy on Windows

You can open Jupyter Notebook from the Anaconda Prompt and create a new Python 3 Notebook in your Jupyter Notebook Web page.





A screenshot of a Jupyter Notebook interface. The title bar shows "PotPlayer MP4 [3/4] zoom_3.mp4" and the URL "localhost:8888/tree". The main area displays a file tree with the root directory "/". A context menu is open over a file named "Python 3", showing options like "Notebook", "Other", "Text File", "Folder", and "Terminal". The file tree list includes:

File	Last Modified
0	3 minutes ago
3D Objects	an hour ago
anaconda3	2 months ago
Contacts	2 months ago
Desktop	2 months ago
Documents	2 months ago
Downloads	4 months ago
Favorites	2 months ago
Links	2 months ago
Music	2 months ago
OneDrive	2 months ago
Pictures	2 months ago
Saved Games	2 months ago
Searches	2 months ago
source	2 years ago
Videos	2 months ago

The bottom of the screen shows a taskbar with various application icons and system status indicators.

Execute the code below to check your Python version, if it is higher than 3.4, PIP is already installed, if not, update your Python to so that you can use PIP to install the SASPy package.

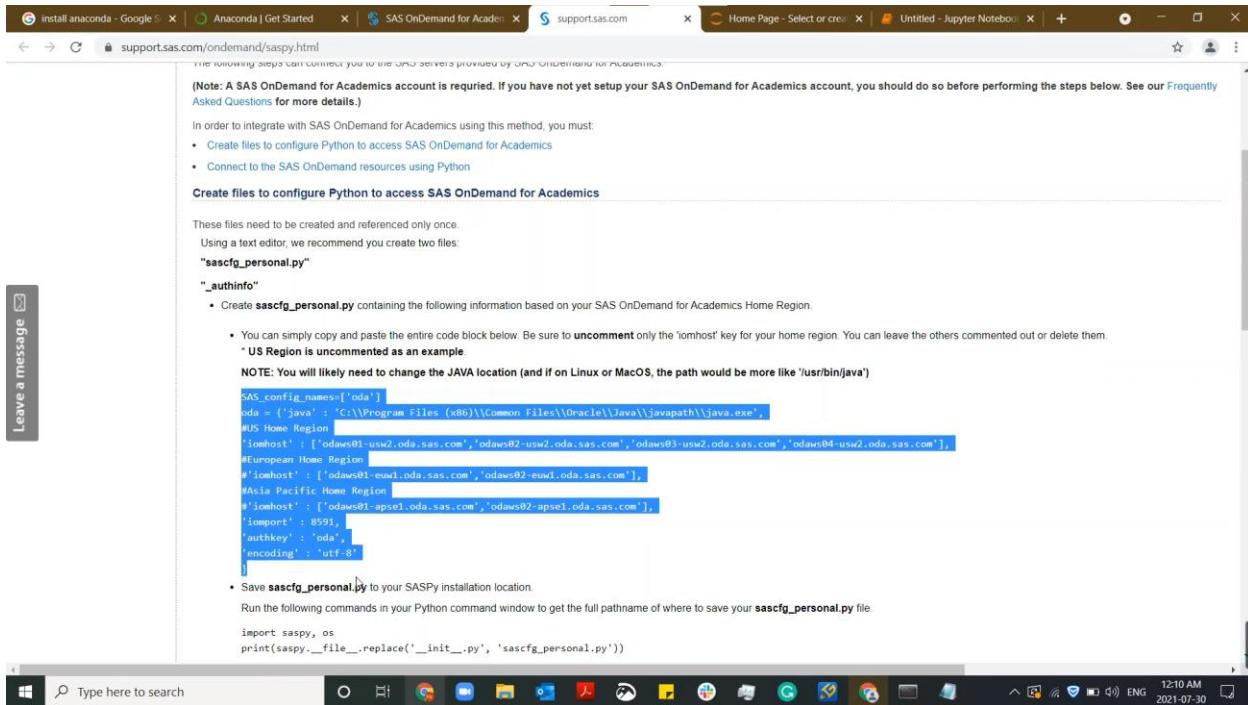
```
from platform import python_version  
print(python_version())
```

A screenshot of a Jupyter Notebook interface. The browser tab shows multiple open tabs, including 'Untitled - Jupyter Notebook'. The notebook itself has one cell containing the code: 'In [1]: from platform import python_version print(python_version())'. The output of this cell is '3.8.8'. Below the notebook, a thought bubble contains the text 'PIP is already installed.' The system tray at the bottom shows various icons and the date/time '12:08 AM 2021-07-30'.

Execute the code ***pip install saspy*** in the Jupyter Notebook to install SASPy.

A screenshot of a Jupyter Notebook interface. The browser tab shows multiple open tabs, including 'Untitled - Jupyter Notebook'. The notebook has two cells. The first cell contains the code: 'In [1]: from platform import python_version print(python_version())'. The output is '3.8.8'. The second cell contains the command: 'In [2]: pip install saspy'. The output of this cell shows the process of downloading and installing the SASPy package, including file names, sizes, and hash codes. The system tray at the bottom shows various icons and the date/time '12:09 AM 2021-07-30'.

Click [here](#) to open the SASPy support page, copy the highlighted text and paste it to a newly created text editor.



The following steps can connect you to the SAS servers provided by OnDemand for Academics.

(Note: A SAS OnDemand for Academics account is required. If you have not yet setup your SAS OnDemand for Academics account, you should do so before performing the steps below. See our Frequently Asked Questions for more details.)

In order to integrate with SAS OnDemand for Academics using this method, you must:

- Create files to configure Python to access SAS OnDemand for Academics
- Connect to the SAS OnDemand resources using Python

Create files to configure Python to access SAS OnDemand for Academics

These files need to be created and referenced only once.

Using a text editor, we recommend you create two files:

- "**sascfg_personal.py**"
- "**_authinfo**"

• Create **sascfg_personal.py** containing the following information based on your SAS OnDemand for Academics Home Region.

- You can simply copy and paste the entire code block below. Be sure to **uncomment** only the "iomhost" key for your home region. You can leave the others commented out or delete them. * US Region is uncommented as an example.

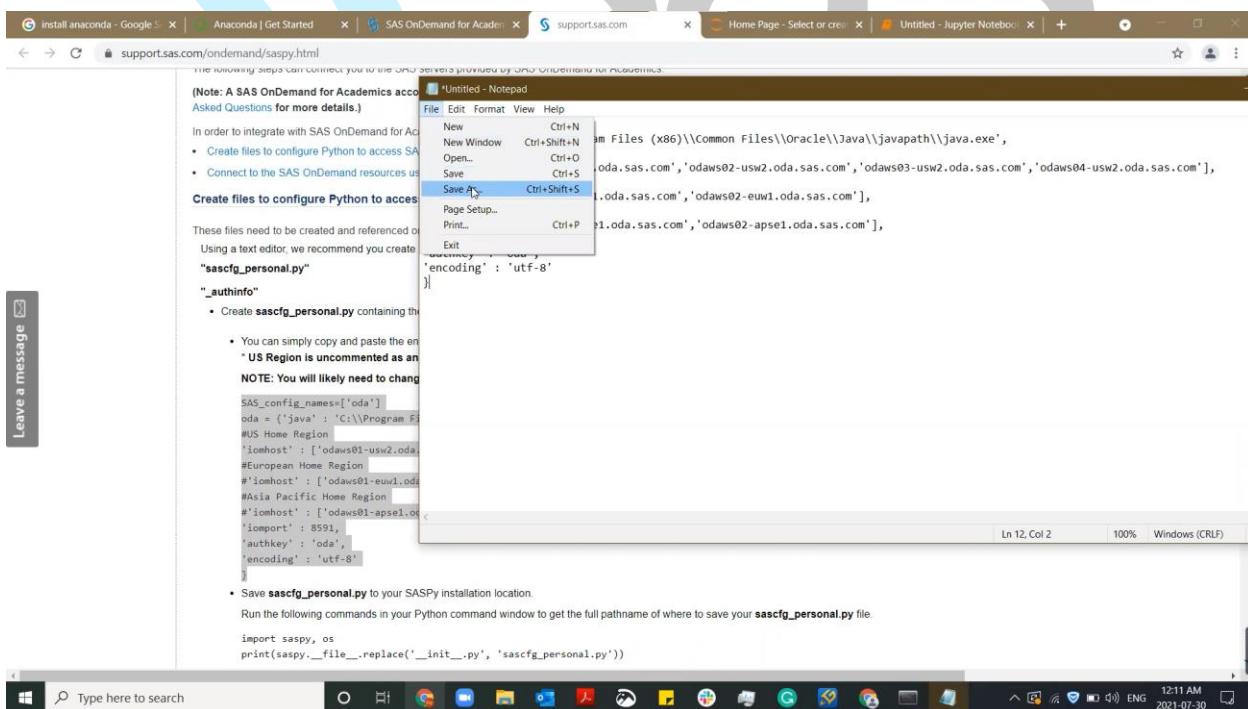
NOTE: You will likely need to change the JAVA location (and if on Linux or MacOS, the path would be more like '/usr/bin/java')

```
SAS_config_names=['oda']
oda = {'Java' : 'C:\\Program Files (x86)\\Common Files\\Oracle\\Java\\javapath\\java.exe',
#US Home Region
'iomhost' : ['odaws01-usw2.oda.sas.com', 'odaws02-usw2.oda.sas.com', 'odaws03-usw2.oda.sas.com', 'odaws04-usw2.oda.sas.com'],
#European Home Region
#iomhost' : ['odaws01-euwl.oda.sas.com', 'odaws02-euwl.oda.sas.com'],
#Asia Pacific Home Region
#iomhost' : ['odaws01-apse1.oda.sas.com', 'odaws02-apse1.oda.sas.com'],
'import' : 891,
'authkey' : 'oda',
'encoding' : 'utf-8'
```

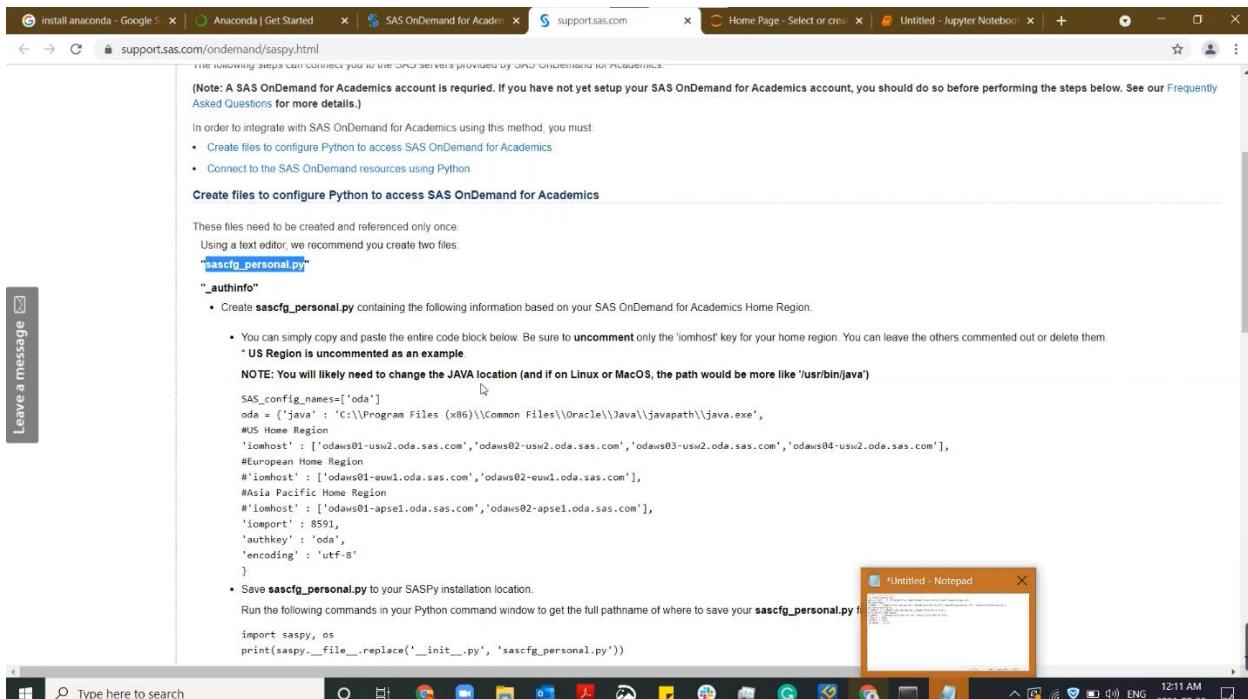
• Save **sascfg_personal.py** to your SASPy installation location.

Run the following commands in your Python command window to get the full pathname of where to save your **sascfg_personal.py** file.

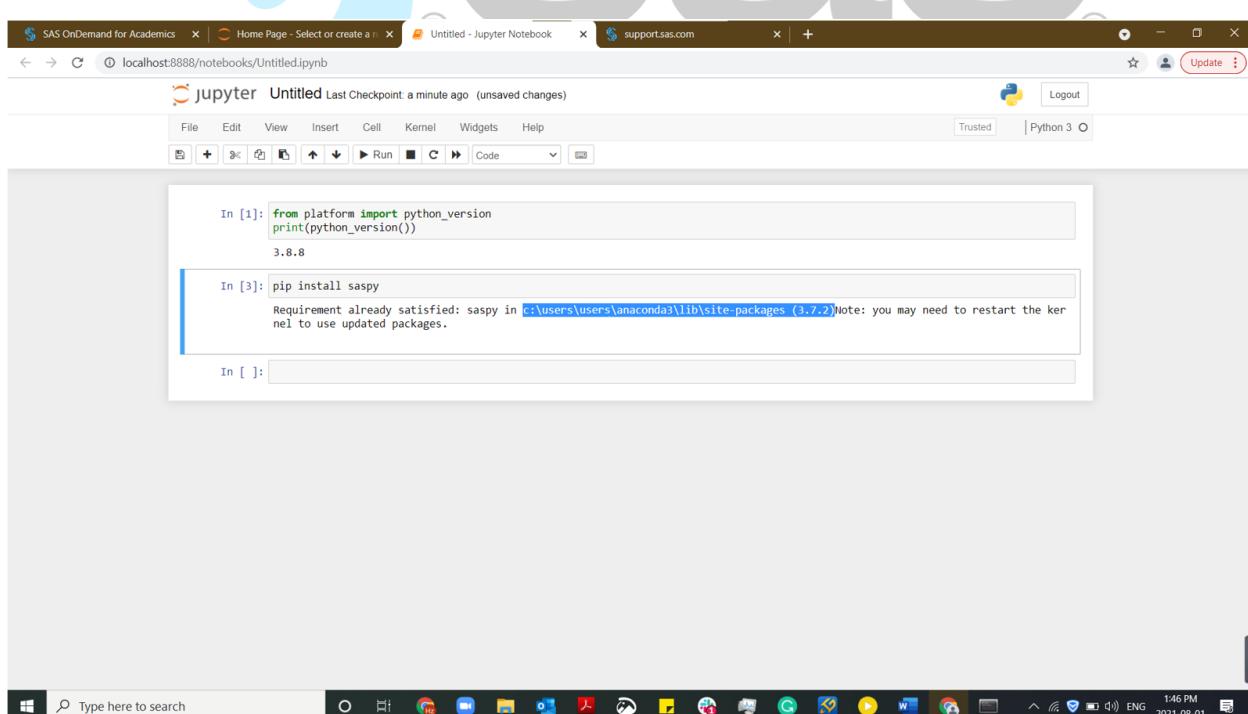
```
import saspy, os
print(saspy._file_.replace('_init__.py', 'sascfg_personal.py'))
```



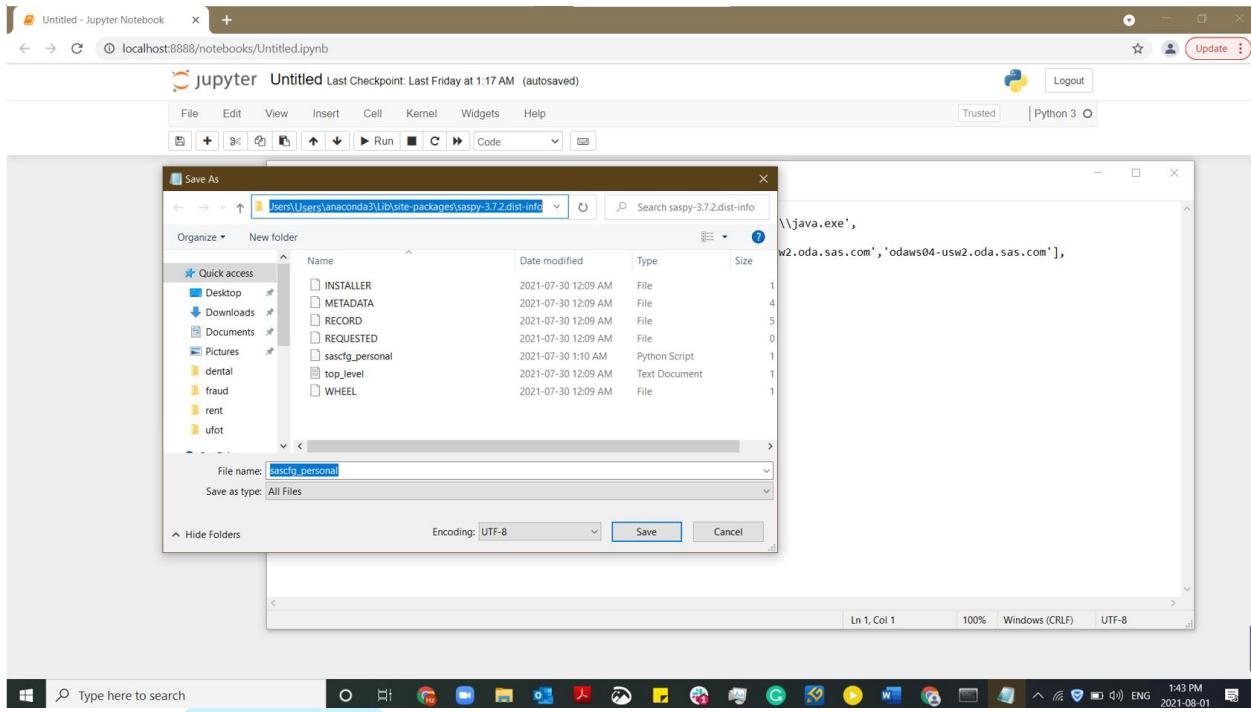
Save the text editor named ***sascfg_personal.py*** to your SASPy installation location, you can find this location by running the code ***pip install saspy*** again in the Jupyter Notebook. You need to change the file type to All Files before you save it, if the Notepad in your Windows does not allow you to do so, [download Notepad++](#) for this file creation.



The screenshot shows a web browser window with the URL support.sas.com/ondemand/saspy.html. The page contains instructions for creating a configuration file for SAS OnDemand for Academics. It includes a code snippet for `sascfg_personal.py` and a note about Java locations. In the background, a Windows taskbar is visible with a Notepad window titled "Untitled - Notepad" open, showing the same configuration file content.



The screenshot shows a Jupyter Notebook interface with a single cell containing Python code. The code imports `saspy` and prints the current Python version. The output shows "3.8.8". Another cell below shows the command `pip install saspy` being run, with a note that the requirement is already satisfied. The Jupyter interface includes a toolbar with File, Edit, View, Insert, Cell, Kernel, Widgets, Help, and a Trusted/Python 3 status bar.



Create another text editor, copy the text highlighted below and paste it to this new text editor.

_authinfo

- Create **sascfg_personal.py** containing the following information based on your SAS OnDemand for Academics Home Region
- You can simply copy and paste the entire code block below. Be sure to **uncomment** only the 'iomhost' key for your home region. You can leave the others commented out or delete them.
"US Region is uncommented as an example.

NOTE: You will likely need to change the JAVA location (and if on Linux or MacOS, the path would be more like '/usr/bin/java')

```
SAS_config_names=['oda']
oda = {'Java' : 'C:\\Program Files (x86)\\Common Files\\Oracle\\Java\\javapath\\java.exe',
#US Home Region
'iomhost' : ['odaws01-usw2.oda.sas.com','odaws02-usw2.oda.sas.com','odaws03-usw2.oda.sas.com','odaws04-usw2.oda.sas.com'],
#European Home Region
'iomhost' : ['odaws01-euw1.oda.sas.com','odaws02-euw1.oda.sas.com'],
#Asia Pacific Home Region
'iomhost' : ['odaws01-apse1.oda.sas.com','odaws02-apse1.oda.sas.com'],
'import' : 8591,
'authkey' : 'oda',
'encoding' : 'utf-8'
}
• Save sascfg_personal.py to your SASPy installation location.
```

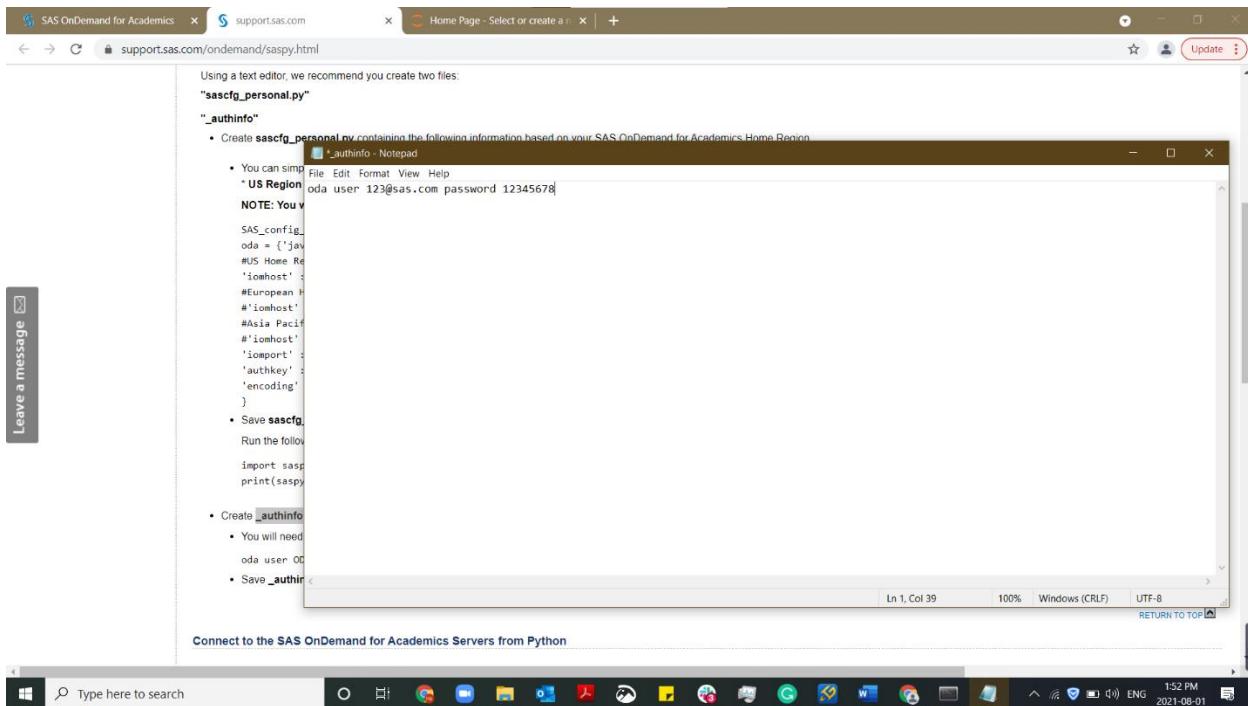
Run the following commands in your Python command window to get the full pathname of where to save your **sascfg_personal.py** file.

```
import saspy, os
print(saspy._file_.replace('_init_.py', 'sascfg_personal.py'))
```

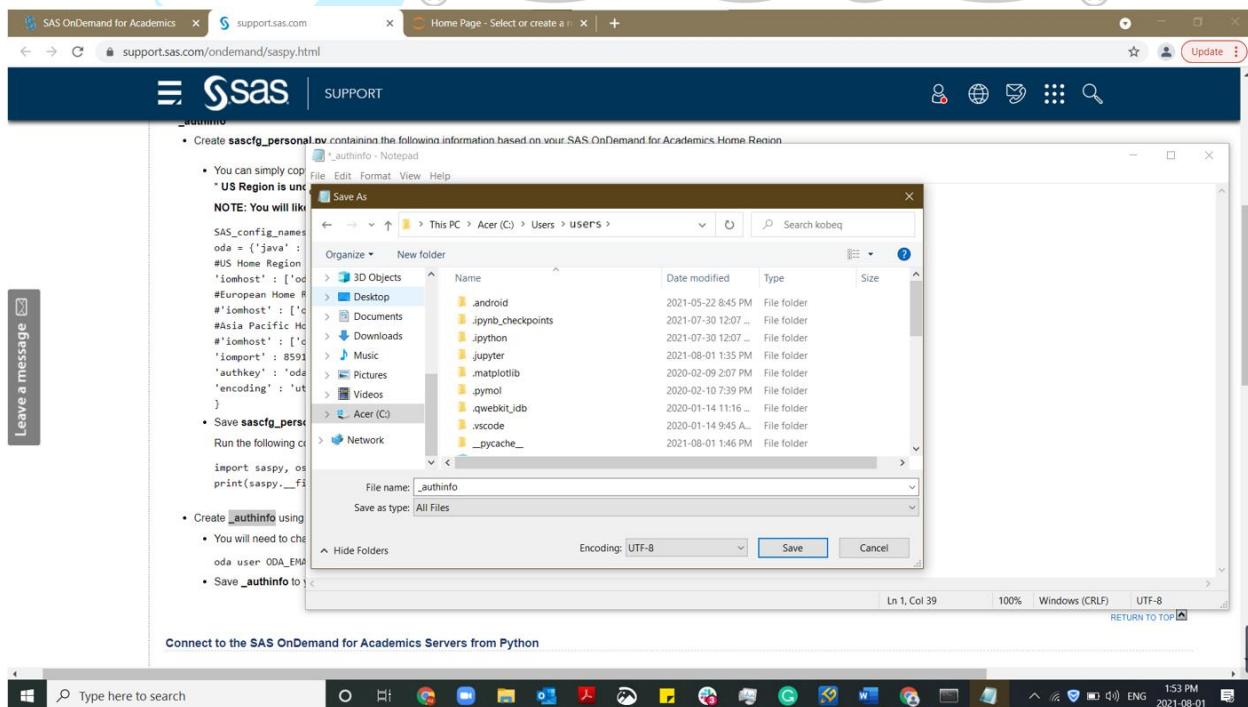
- Create **_authinfo** using the below as a template (**authinfo** if on Linux or MacOS). See the [saspy instructions](#) for more information on this topic.
 - You will need to change "ODA_EMAIL/ODA_USERNAME" and "ODA_PASSWORD" to your SAS OnDemand for Academics credentials.
 - `oda user ODA_EMAIL/ODA_USERNAME password ODA_PASSWORD`
 - Save **_authinfo** to your user's home directory `C:\Users\YOUR_USERNAME` on Windows.

[RETURN TO TOP](#)

You will need to change ***ODA_EMAIL/ODA_USERNAME*** and ***ODA_PASSWORD*** to your ODA credentials, for example, if your ODA email is 123@sas.com and your ODA password is 12345678, your text editor will look like this.

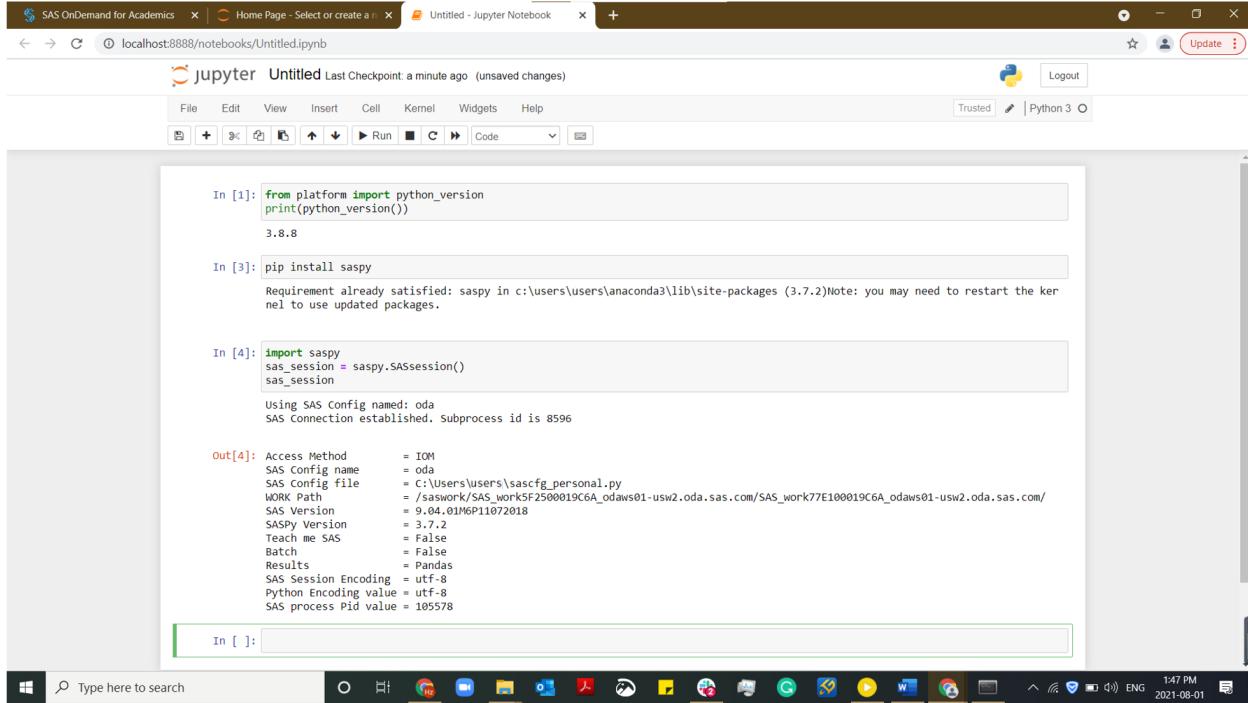


Save ***_authinfo*** to your user's home directory *C:\Users\YOUR_USERNAME* on Windows. You need to change the file type to All Files before you save it.



From a Python prompt or from another Python interface, like Jupyter Notebook, use the following commands to confirm a connection to ODA. ***This step should be performed each time that you want to connect to hosted SAS servers.***

```
import saspy  
sas_session = saspy.SASsession()  
sas_session
```



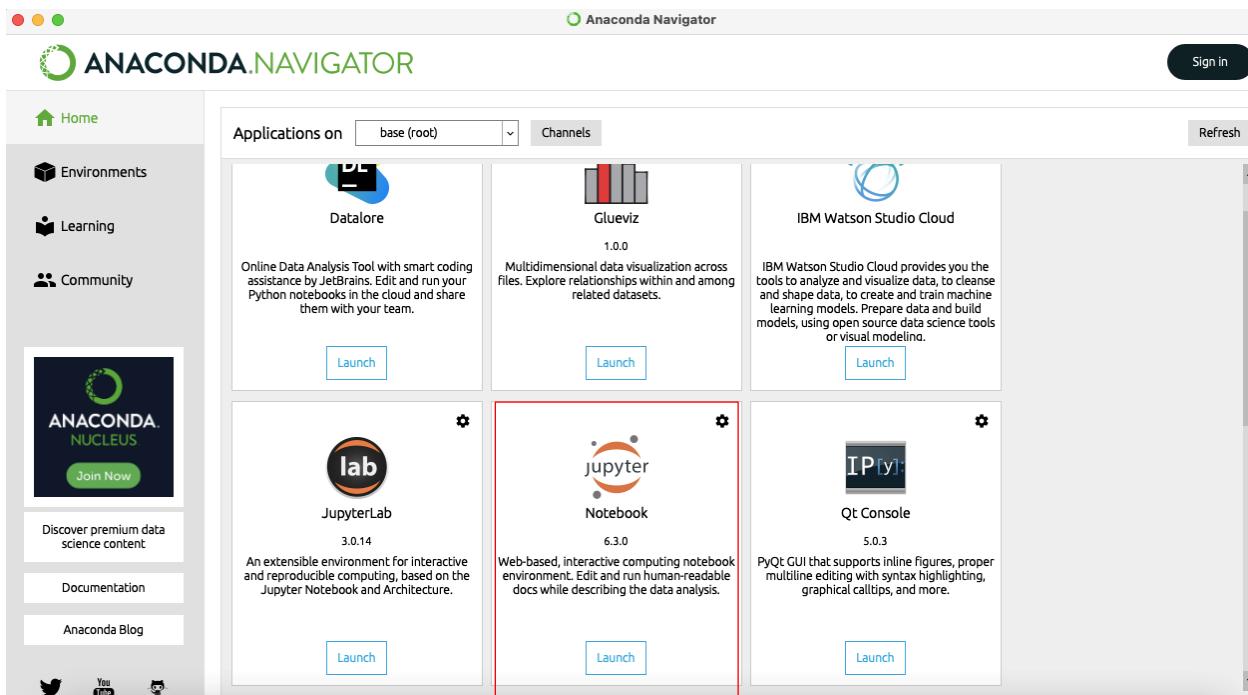
The screenshot shows a Jupyter Notebook running on a Windows desktop. The notebook has four cells:

- In [1]: `from platform import python_version`
`print(python_version())`
3.8.8
- In [3]: `pip install saspy`
Requirement already satisfied: saspy in c:\users\users\anaconda3\lib\site-packages (3.7.2)
Note: you may need to restart the kernel to use updated packages.
- In [4]: `import saspy`
`sas_session = saspy.SASsession()`
Using SAS Config named: oda
SAS Connection established. Subprocess id is 8596
- Out[4]:
Access Method = IOM
SAS Config name = oda
SAS Config file = C:\Users\users\sascfg_personal.py
WORK Path = /saswork/SAS_workSF2500019C6A_odaws01-usw2.oda.sas.com/SAS_work77E100019C6A_odaws01-usw2.oda.sas.com/
SAS Version = 9.04.01M6P11072018
SASPY Version = 3.7.2
Teach me SAS = False
Batch = False
Results = Pandas
SAS Session Encoding = utf-8
Python Encoding value = utf-8
SAS process Pid value = 105578

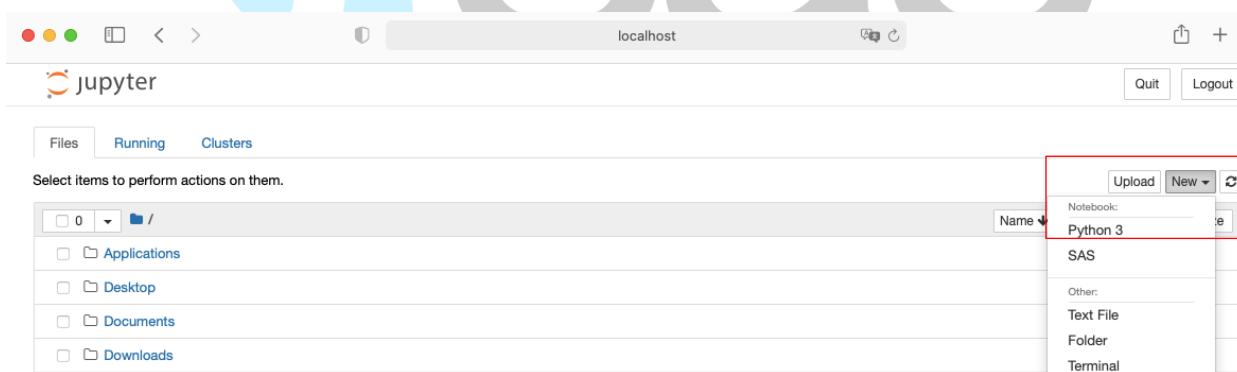
Note: If you encounter, "***None of the requested encryption algorithms are supported by both peers: AES***", please confirm you are running the correct version of Java (1.8.0_162 or greater) and Contact Us for additional assistance.

Install SASPy on MAC

Download Anaconda [here](#). Launch Jupyter Notebook from the Anaconda Navigator.

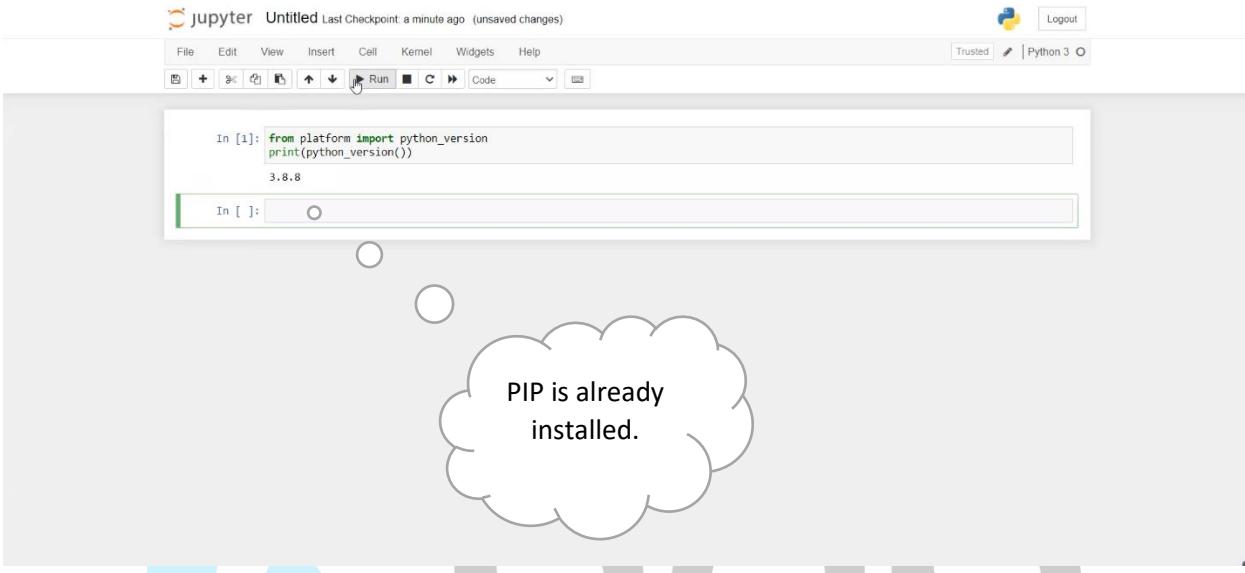


Create a new Python 3 Notebook in your Jupyter Notebook Web page.



Execute the code below to check your Python version, if it is higher than 3.4, PIP is already installed, if not, update your Python to so that you can use PIP to install the SASPy package.

```
from platform import python_version  
print(python_version())
```



Execute the code ***pip install saspy*** in the Jupyter Notebook to install SASPy.

```
In [1]: from platform import python_version  
print(python_version())  
3.8.8  
  
In [2]: pip install saspy  
Collecting saspy  
  Using cached saspy-3.7.2-py3-none-any.whl  
Installing collected packages: saspy  
Successfully installed saspy-3.7.2  
Note: you may need to restart the kernel to use updated packages.
```

You will need to create 2 configuration files in total (*sascfg_personal.py* & *.authinfo*) and this step is for *sascfg_personal.py* file.

1. Run the following commands in your Jupyter Notebook to get the full pathname of where to save your *sascfg_personal.py* file. Copy the path you got.

```
import saspy, os  
print(saspy.__file__.replace('__init__.py', 'sascfg_personal.py'))
```

```
In [3]: import saspy, os  
print(saspy.__file__.replace('__init__.py', 'sascfg_personal.py'))  
  
/Users/ /opt/anaconda3/lib/python3.8/site-packages/saspy/sascfg_personal.py
```

2. Open your Terminal and start a new window



3. Type Command **vim FULLPATH** in the Terminal to create *sascfg_personal.py* at the path got in step 1. You will need to change **FULLPATH** to your unique path got from Step 1.

For example, given the path achieved in previous picture, you will enter:

```
vim /users/username/opt/anaconda3/lib/python3.8/site-packages/saspy/sascfg_personal.py
```

```
● ○ ● ● Username -bash - 80x24  
Last login: Sun Aug 1 17:07:46 on ttys000  
  
The default interactive shell is now zsh.  
To update your account to use zsh, please run `chsh -s /bin/zsh`.  
For more details, please visit https://support.apple.com/kb/HT208050.  
(base) Username MacBook-Air:~ Username $ vim /Users/ Username/ opt/anaconda3/lib/py  
thon3.8/site-packages/saspy/sascfg_personal.py
```

4. Press “*a*” to enter edit mode

5. Copy and paste the grey code block below. Be sure to uncomment only the 'iomhost' key for your home region. ***US Region is uncommented as an example.** You can leave the others commented out or delete them.

NOTE: You will likely need to change the JAVA location. For MAC users, your Java path is usually '/usr/bin/java'

```
SAS_config_names=['oda']
oda = {'java' : '/usr/bin/java',
#US Home Region
'iomhost' : ['odaws01-usw2.oda.sas.com','odaws02-usw2.oda.sas.com','odaws03-
usw2.oda.sas.com','odaws04-usw2.oda.sas.com'],
#European Home Region
#'iomhost' : ['odaws01-euw1.oda.sas.com','odaws02-euw1.oda.sas.com'],
#Asia Pacific Home Region
#'iomhost' : ['odaws01-apse1.oda.sas.com','odaws02-apse1.oda.sas.com'],
'iomport' : 8591,
'authkey' : 'oda',
'encoding' : 'utf-8'
}
```

6. Hit "**Esc**" and then type **:wq** and then hit "**Enter**" save and quit.

```
SAS_config_names=['oda']
oda = {'java' : '/usr/bin/java',
#US Home Region
'iomhost' : ['odaws01-usw2.oda.sas.com','odaws02-usw2.oda.sas.com','odaws03-usw2
.oda.sas.com','odaws04-usw2.oda.sas.com'],
#European Home Region'iomhost' : ['odaws01-euw1.oda.sas.com','odaws02-euw1.oda.s
as.com'],
#Asia Pacific Home Region
#'iomhost' : ['odaws01-apse1.oda.sas.com','odaws02-apse1.oda.sas.com'],
'iomport' : 8591,
'authkey' : 'oda',
'encoding' : 'utf-8'
}
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
:wq
```

You will need to create 2 configuration files in total (***sascfg_personal.py*** & ***.authinfo***) and this step is for ***.authinfo*** file.

1. Open Terminal and type command ***vim ~/.authinfo***

```
● ● ● Username bash - 80x24
Last login: Tue Jul  6 18:36:45 on ttys000
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) Username MacBook-Air:~ Username $ vim ~/.authinfo
```

2. Press “***a***” to enter edit mode.
3. Type in the following text, use the coding block below as a template. You will need to change ***ODA_EMAIL/ODA_USERNAME*** and ***ODA_PASSWORD*** to your SAS OnDemand for Academics credentials

```
oda user ODA_EMAIL/ODA_USERNAME password ODA_PASSWORD
```

```
● ● ● Username vim ~/.authinfo - 80x24
oda user ODA_Email/ODA_Username password ODA_password
```

4. Press “***Esc***” and then type **:wq** and then hit “***Enter***” to save the file and quit.

Open Python prompt or from another Python interface, like Jupyter Notebook. Use the following commands to confirm a connection to ODA. *This step should be performed each time that you want to connect to hosted SAS servers.*

```
import saspy
sas_session = saspy.SASsession()
sas_session
```

```
In [7]: import saspy
sas_session = saspy.SASsession()
sas_session

Using SAS Config named: oda
SAS Connection established. Subprocess id is 12442
```

```
Out[7]: Access Method      = IOM
SAS Config name        = oda
SAS Config file       = /Users/      /opt/anaconda3/lib/python3.8/site-packages/saspy/sascfg_personal.py
WORK Path             = /saswork/SAS_work19620000F37A_odaws02-usw2.oda.sas.com/SAS_workAB480000F37A_odaws02-usw2.od
a.sas.com/
SAS Version          = 9.04.01M6P11072018
SASPy Version        = 3.7.2
Teach me SAS         = False
Batch                = False
Results              = Pandas
SAS Session Encoding = utf-8
Python Encoding value = utf-8
SAS process Pid value = 62330
```

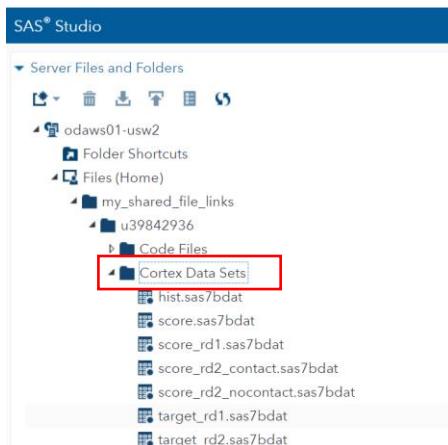
Note: If you encounter, "***None of the requested encryption algorithms are supported by both peers: AES***", please confirm you are running the correct version of Java (1.8.0_162 or greater) and Contact Us for additional assistance.

Access to Dataset

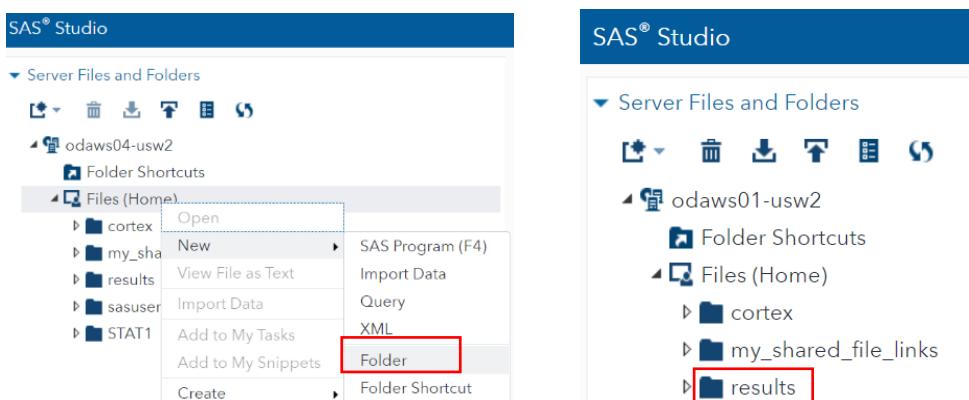
In order to access the dataset in the SAS terminal, you need to follow these steps:

1. Sign on the Control Center at <https://odamid.oda.sas.com>.
2. Look for the Enroll in a course link in the "Enrollments" section near the bottom of the page. Click this link to start the enrollment.
3. Enter the course code: 83fd3afd-8964-4f5b-a612-7b665ed69104

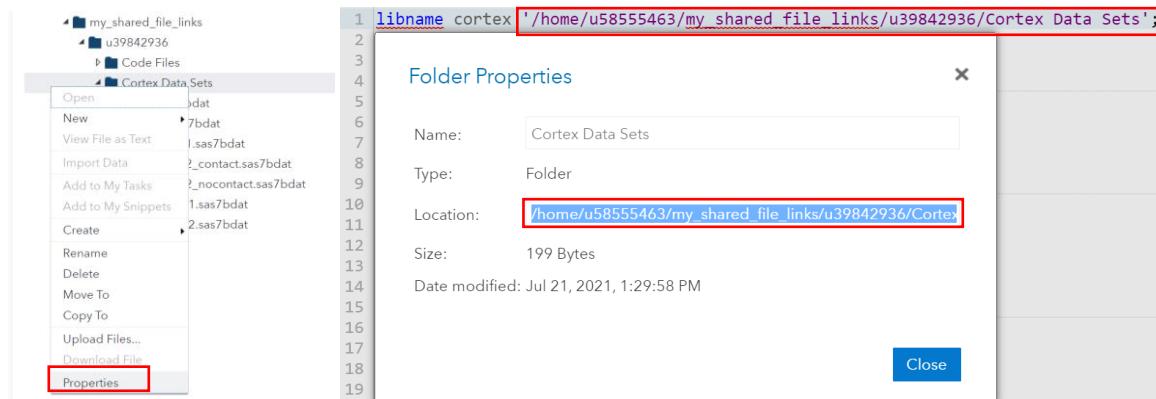
Then you can open your SAS Studio interface and open the “my_shared_file_links” folder, you can see all the cortex dataset in the folder named “Cortex Data Sets” under the “u39842936” folder.



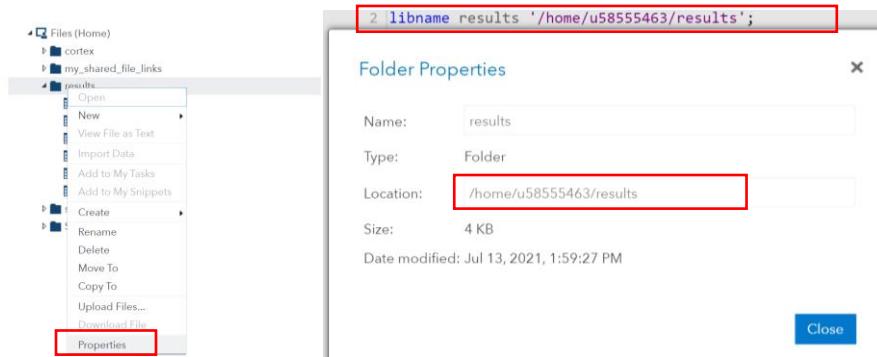
After that, you need to create a folder named “results”. This is important because you do not have the authority to do any edit/change to the datasets in that folder “u39842936”. Please restore any interim datasets to the 'work' or 'results' library.



Copy the path of “u39842936” folder and create “cortex” library by code: `libname cortex '/path';`



Copy the path of “results” folder and create “results” library by code: `libname results '/path';`; then you can restore any interim dataset to the “results” folder&library.



[Link for Reference](#)

You can then start coding in the SAS Studio.

If you want to use Python Command for coding, you need to connect your SAS terminal to the Python Command using SASPy, then you need to add this code `%%SAS sas_session` and create “cortex” library in SAS language to access datasets in SAS terminal. You don’t need to create “results” library in Jupyter Notebook.

```
In [ ]: ┌─ from saspy import SASsession
          sas_session = SASsession()
          sas_session

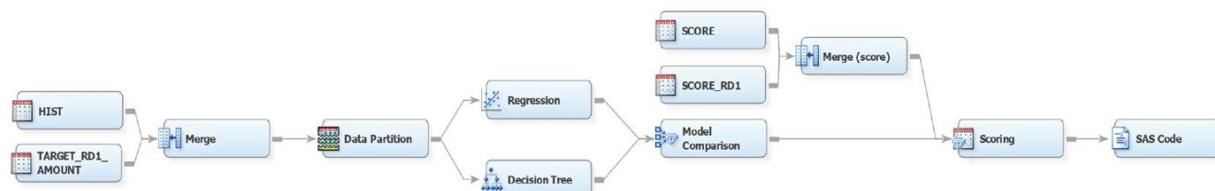
In [ ]: ┌─ %%SAS sas_session
          libname cortex '/home/u58555463/my_shared_file_links/u39842936/Cortex Data Sets';
```

Round 1 - Game Summary

In Round 1, your mission is to predict the amount that each member would give next year to the foundation. You would then use this prediction to call the most valuable members (i.e., members who are predicted to give the most to the foundation).

Round 1 - Model Description (SAS)

There are 6 steps for game completion: Data Merge, Data Partition, Model Building, Model Comparison, Scoring Data and Export Results. Check [Data Modification](#) for more SAS resources.



Step 1: Data Merge

The first step is to merge two datasets into one for better modification. You need to deal with missing values after merging the datasets so that you proceed can to the next step — Data Partition.

```
/* merge dataset hist and target_rd1*/
DATA model_rd1;
  MERGE cortex.hist cortex.target_rd1;
  BY ID;
run;

data results.model_rd1; /*drop na*/
set model_rd1;
if not cmiss(of _numeric_);
run;
```

You can use other way to deal with missing values

[Link for Reference](#)

Step 2: Data Partition

The second step is data partition, this will help the model to increase its performance, please note that this is a default model, you can change the training and validation percentage for a better model performance.

```
/* data partition*/
proc surveyselect data=results.model_rdl rate=0.6
out= donor select outall
method=srs
seed =1234;
run;

data results.train_rdl results.test_rdl;
set donor select;
if selected =1 then output results.train_rdl;
else output results.test_rdl;
run;
```

[Link for Reference](#)

Step 3: Model Building

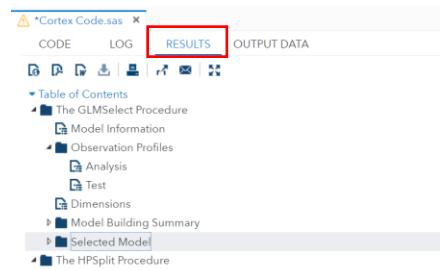
In this step, [linear regression](#) and [decision tree](#) models are used for analysis, you can also choose other model methods (i.e., [logistic regression](#), [HP forest](#), [gradient boosting](#)) to predict the amount donated. We recommend you perform a descriptive analysis to select independent variables for prediction.

```
/* glm model*/
ods graphics off;
proc glmselect data=results.train_rdl testdata=results.test_rdl;
model AmtThisYear=Age Salary Seniority GaveLastYear;
title 'Regression of donation this year'
'Predictors';
code file='/home/u58717790/Results/regression_1.sas';
run;

/* decision tree model*/
proc arboretum data= results.train_rdl;
target AmtThisYear / level=interval;
input Age Salary / level=interval;
input GaveLastYear Seniority/level=nominal;
score data=results.test_rdl role=valid OUT=_NULL_ OUTFIT=results.DT_stat_rdl;
code file='/home/u58717790/Results/decisiontree_1.sas';
quit;
```

Step 4: Model Comparison

When comparing models, you can click RESULTS tab for comparison, SAS will compute statistics automatically, you can then choose [different criteria](#) to determine which model fits better for prediction. In the default model, linear regression model is selected for scoring.



Step 5: Scoring Data

Scoring new data to compute predictions for an existing model is a fundamental stage in the analytics life cycle, after you score out the dataset, you can export the model result for upload.

```
/* scoring the data*/
DATA score_rdl;
  MERGE cortex.score cortex.score_rdl;
  BY ID;
run;

data results.score_rdl; /*drop na*/
set score_rdl;
if not cmiss(of _numeric_);
run;

data results.result_rdl (keep= id p_amtthisyear);
set results.score_rdl;
%include '/home/u58717790/Results/regression_1.sas';
run;
```

Step 6: Export Results

Following code will help you to export the model result into a CSV file. You can sort the data by predicted amount in the descending order before export the file, this will help you determine number of donors you want to contact.

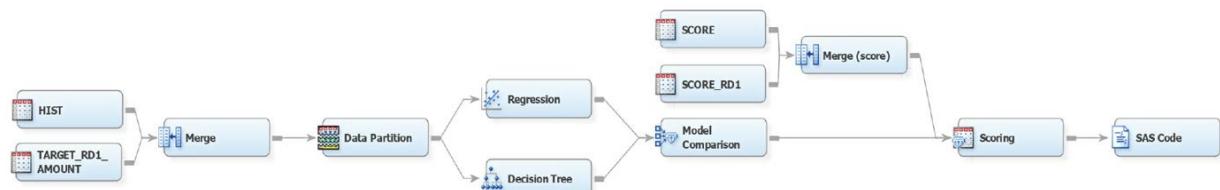
```
/* Export data*/
PROC SORT DATA=results.result_rdl;
  BY descending p_amtthisyear;
RUN;

proc export data=results.output_rdl
outfile="/home/u58717790/Results/Round1_Output.csv" dbms=csv
replace;
run;
```

Round 1 - Model Description (Python)

NOTE: You need to install **saspy**, **pandas**, **numpy**, and **sklearn** python packages to run the baseline model in your python command.

There are 6 steps for game completion: Data Merge, Data Partition, Model Building, Model Comparison, Scoring Data and Export Results.



Step 1: Data Merge

The first step is to merge two datasets into one for better modification. You need to deal with missing values after merging the datasets so that you can proceed to the next step — Data Partition.

```
#Step1 Merge the Data
data_merge = pd.merge(data1, data2, on=["ID"], how="right")
data_merge.head()
#Deal with Missing Value
data_merge = data_merge.dropna() #comment: maybe another method for missing data, check it later
data_merge.head()
```

Step 2: Data Partition

The second step is data partition, this will help the model to increase its performance, please note that this is a default model, you can change the training and validation percentage for a better model performance.

```
#Step2 Data Partition
#this is just a sample, you could use another Library or bulit in function
from sklearn.model_selection import train_test_split
train, validation = train_test_split(data_merge, test_size=0.4) # you can change the percentage
train.head()
```

Step 3: Model Building

In this step, linear regression and decision tree models are used for analysis, you can also choose other model methods (i.e., logistic regression, HP forest, gradient boosting) to predict the amount donated. We recommend you perform a descriptive analysis to select independent variables for prediction.

```
from sklearn import linear_model

#comment: it's numpy array
X_train = train[['Age', 'Salary','Seniority', 'AmtLastYear']]
Y_train = train['AmtThisYear']
X_valid = validation[['Age', 'Salary','Seniority', 'AmtLastYear']]
Y_valid = validation['AmtThisYear']

regr = linear_model.LinearRegression()
regr.fit(X_train,Y_train)
regr_predict=regr.predict(X_valid)
```

```
#you can change the criteria

import numpy as np
from sklearn import metrics
#MAE
print(metrics.mean_absolute_error(Y_valid,regr_predict))
#MSE
print(metrics.mean_squared_error(Y_valid,regr_predict))
#RMSE
print(np.sqrt(metrics.mean_squared_error(Y_valid,regr_predict)))

19.718080643457622
11071.03733901477
105.21899704433021
```

```
from sklearn.tree import DecisionTreeRegressor

X_train = train[['Age', 'Salary','Seniority', 'AmtLastYear']]
Y_train = train['AmtThisYear']
X_valid = validation[['Age', 'Salary','Seniority', 'AmtLastYear']]
Y_valid = validation['AmtThisYear']

DT_model = DecisionTreeRegressor(max_depth=5).fit(X_train,Y_train)
DT_predict = DT_model.predict(X_valid) #Predictions on Testing data
```

```
#you can change the criteria
#MAE
print(metrics.mean_absolute_error(Y_valid,DT_predict))
#MSE
print(metrics.mean_squared_error(Y_valid,DT_predict))
#RMSE
print(np.sqrt(metrics.mean_squared_error(Y_valid,DT_predict)))

19.7086695390796
11259.624229034016
106.11137652972944
```

Step 4: Model Comparison

When comparing models, you need to choose a criterion and calculate it manually to determine which model fits better for prediction since Python will not compute the statistical results automatically. In the default model, we compare the MSE of two model, and the linear regression model is selected for scoring.

Step 5: Scoring Data

Scoring new data to compute predictions for an existing model is a fundamental stage in the analytics life cycle, after you score out the dataset, you can export the model result for upload.

```
scoring_data = pd.merge(data3, data4, on=["ID"], how="right")
scoring_data = scoring_data.dropna()
scoring_data.head()

X = scoring_data[['Age', 'Salary', 'Seniority', 'AmtLastYear']]
regr_predict_end=regr.predict(X)
scoring_data['Prediction'] = regr_predict_end

scoring_data.sort_values(by=['Prediction'], inplace=True, ascending=False)
scoring_data.head()
```

Step 6: Export Results

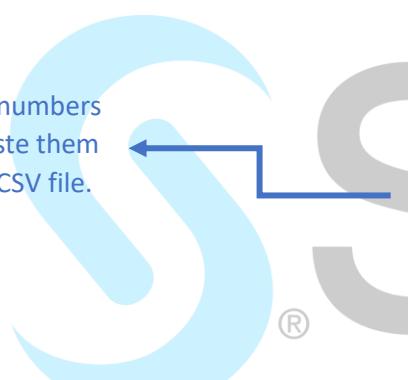
Following code will help you to export the model result into a CSV file. You can sort the data by predicted amount in the descending order before export the file, this will help you determine number of donors you want to contact.

```
Result= scoring_data[['ID', 'Prediction']]
Result.to_csv('Round1_Output.csv', index=False)
```

Round 1 - Upload Decisions

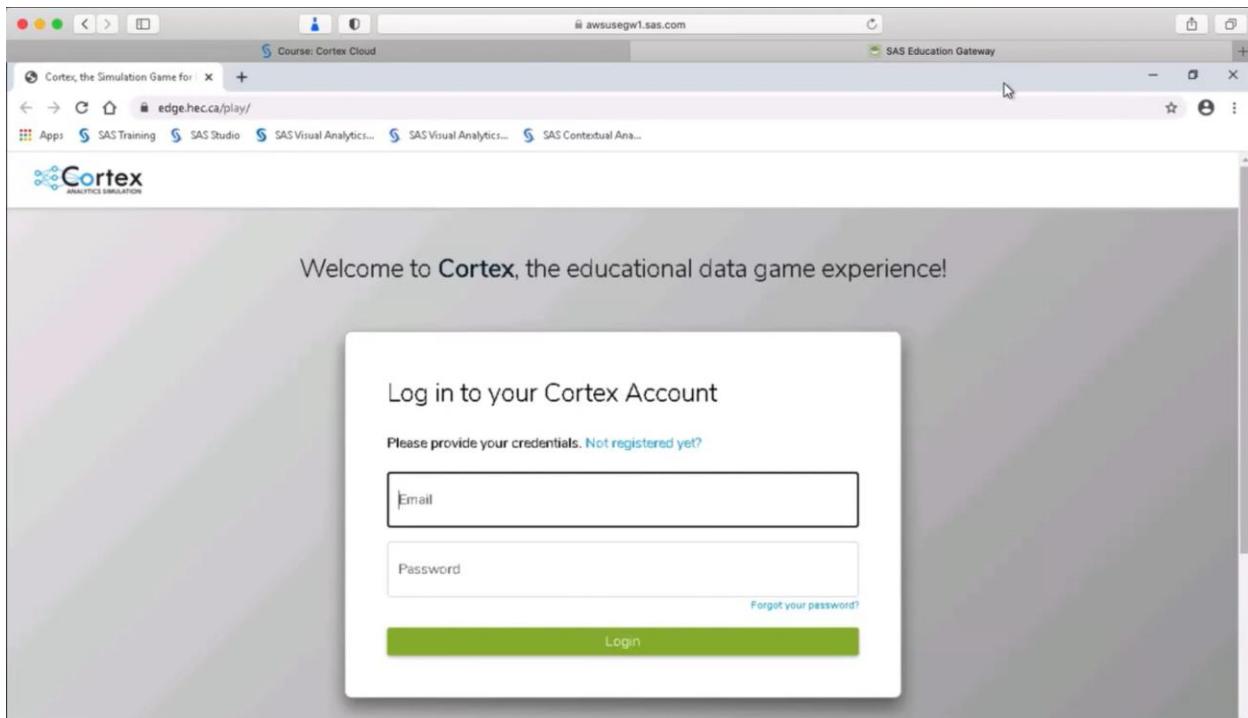
After you export the result, you need to create a new CSV file, select the number of people you want to contact in the file exported, and copy the ID list ONLY to the new CSV file. You need to **SAVE** and **CLOSE** the new CSV file before uploading it to the game leaderboard.

Select ID numbers
ONLY, paste them
to a new CSV file.

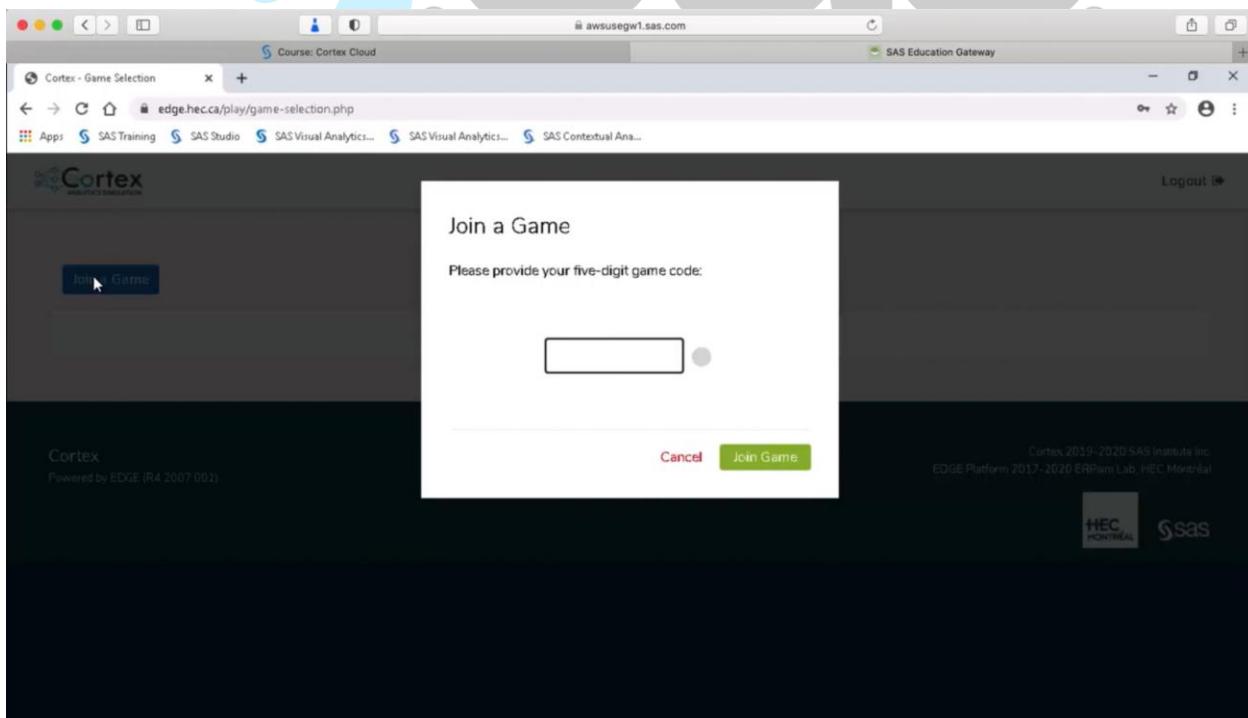


A	B	C	D	E	F	G	H
1 ID	Predicted						
2 2614873	30.75581						
3 2193356	30.70765						
4 2616449	30.64838						
5 2208025	30.5039						
6 2572522	30.46109						
7 2232518	30.43434						
8 2017025	30.37588						
9 2918876	30.37588						
10 2978270	30.34913						
11 2433265	30.33308						
12 2297238	30.31702						
13 2723669	30.28905						
14 2637077	30.25775						
15 2946923	30.24211						
16 2104314	30.22565						
17 2497239	30.14376						
18 2111312	30.13346						
19 2384884	30.10217						
20 2110812	30.03138						
21 2148211	30.02603						
22 2211631	30.02603						
23 2011207	29.99061						
24 2580510	29.98526						
25 2261728	29.95769						
26 2750743	29.95728						
27 2229345	29.93506						
28 2211997	29.92518						
29 2889345	29.87661						
30 2211063	29.85399						
31 2032546	29.85358						
32 2076486	29.83421						
33 2813830	29.83299						
34 2158899	29.83008						
35 2611528	29.82845						

Then you need to log in to the Leaderboard: <https://edge.hec.ca/play/>



Click **Join a Game** button and enter the Five-Digit Game Code **NF02N** that you should have received from our email.



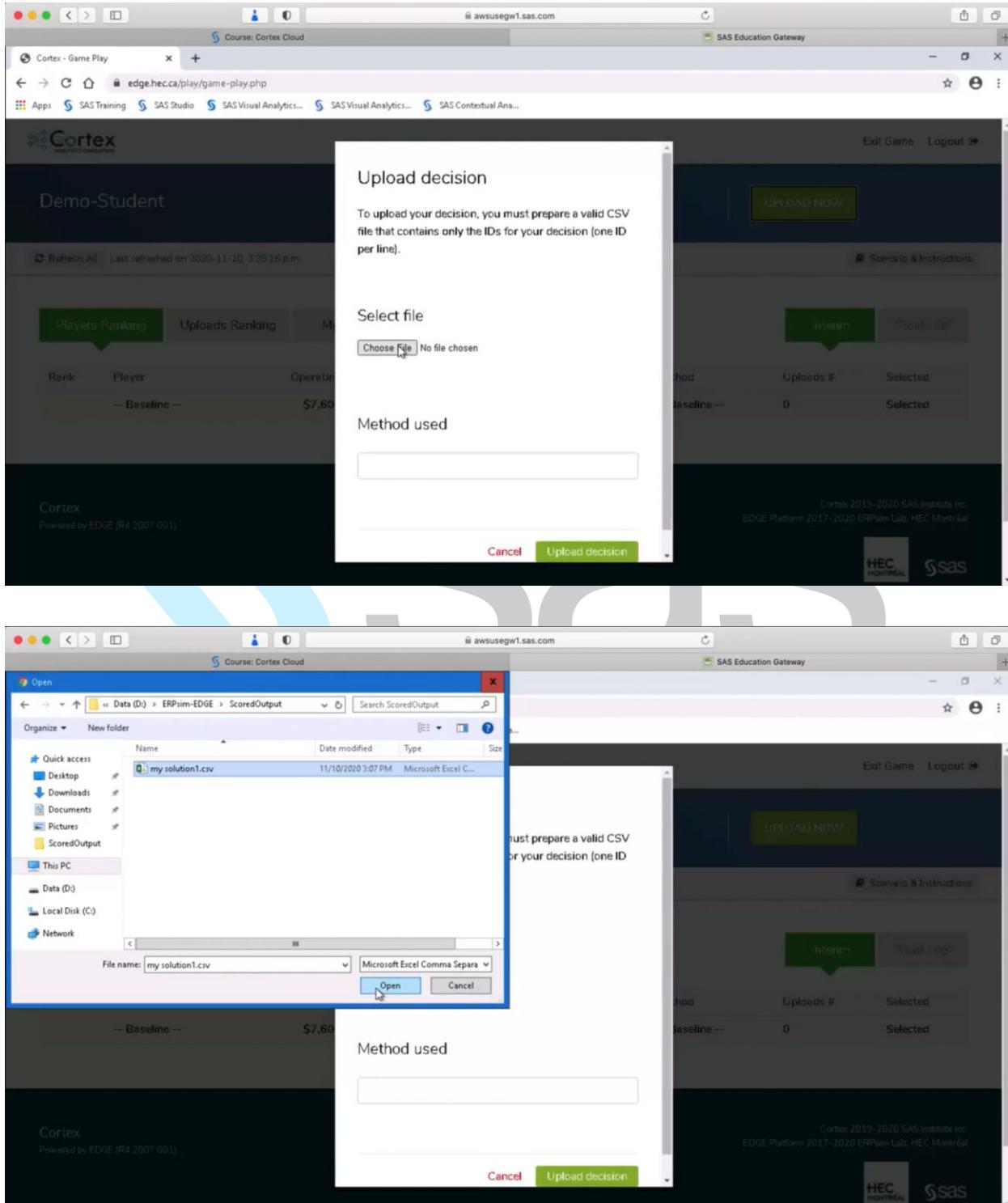
Click **Play** button to start the game.

The screenshot shows a web browser window titled "Course: Cortex Cloud" at "edge.hec.ca/play/game-selection.php". The page features a "Join a Game" button and a list of users, including "Demo-Student". A prominent blue "Play" button is located on the right side of the user list. The footer contains the Cortex logo, copyright information for SAS Institute Inc., and logos for HEC Montréal and SAS.

Click **UPLOAD NOW** button to upload CSV file.

The screenshot shows a web browser window titled "Course: Cortex Cloud" at "edge.hec.ca/play/game-play.php". The page displays a "Demo-Student" profile with a message "Game is running" and "1 player". A large green "UPLOAD NOW" button is prominently displayed. Below it is a table for "Players Ranking" and "Uploads Ranking". The table includes columns for Rank, Player, Operating surplus, Expenses, Donors contacted, Method, Uploads #, and Selected. A "Baseline" row is shown. At the bottom, there are tabs for "Interim" and "Real Life". The footer contains the Cortex logo, copyright information for SAS Institute Inc., and logos for HEC Montréal and SAS.

Click **Choose File** button to select your new CSV file, you can name your upload under the **Method used**, and click **Upload decision** button to complete your upload.



The screenshot shows a web browser window with two tabs: "Cortex - Game Play" and "SAS Education Gateway". The main content area displays the Cortex game interface for a user named "Demo-Student". The interface includes a navigation bar with "Players Ranking", "Uploads Ranking", and "My Uploads" tabs. The "Players Ranking" tab is active, showing a table with columns: Rank, Player, Operating surplus, Expenses, Donors contacted, Method, Uploads #, and Selected. A row for "John Doe" is highlighted in green, indicating the user's current position. Below the table, there are tabs for "Interim" and "Real Life". A modal dialog box titled "Upload decision" is open in the center. It contains instructions: "To upload your decision, you must prepare a valid CSV file that contains only the IDs for your decision (one ID per line)". It has a "Choose File" button with the path "my solution1.csv" and a text input field containing "my 1st upload". At the bottom of the dialog are "Cancel" and "Upload decision" buttons, with the latter being the target of a mouse cursor.

In the main leaderboard: **Players Ranking** tab, you can check the result of the model you choose to upload as well as results uploaded by other players.

The screenshot shows the same web browser window and Cortex game interface as the previous one, but the "Players Ranking" tab is now active. The main content area displays the game results. The "Game is running" status bar at the top right shows "1 player". The "Players Ranking" table is displayed with the following data:

Rank	Player	Operating surplus	Expenses	Donors contacted	Method	Uploads #	Selected
1	John Doe	\$8,594,815.00	\$549,975.00	69999	my 2nd model	2	<input checked="" type="radio"/>
-- Baseline --							
		\$7,602,655.00	\$0.00	0	-- Baseline --	0	Selected

In the **My Uploads** tab, you can see results of all your uploads in the chronological order.

The screenshot shows the Cortex Analytics Simulation interface. At the top, there are tabs for "Players Ranking", "Uploads Ranking", and "My Uploads". The "My Uploads" tab is currently selected, indicated by a green background and white text. Below the tabs is a table with the following data:

Rank	Player	Operating surplus	Expenses	Donors contacted	Method	Uploads #	Selected
1	John Doe	\$7,602,720.00	\$150.00	30	my 1rst upload	1	<input type="radio"/>
-- Baseline --		\$7,602,655.00	\$0.00	0	-- Baseline --	0	Selected

In the **Uploads Ranking** tab to see your uploads' result based on the Operating Surplus ranking. You can then **Select** one of your uploads with the highest operating surplus and upload to the main Leaderboard, where other players can also see your result.

The screenshot shows the Cortex Analytics Simulation interface. The "Uploads Ranking" tab is currently selected, indicated by a green background and white text. Below the tabs is a table with the following data:

Rank	Player	Operating surplus	Expenses	Donors contacted	Method	Uploads #	Selected
1	John Doe	\$7,602,720.00	\$150.00	30	my 1rst upload	1	<input type="radio"/>
-- Baseline --		\$7,602,655.00	\$0.00	0	-- Baseline --	0	Selected

Round 2 - Game Summary

In Round 2, the foundation still needs help to increase the net amount of the donations. To do so, the effectiveness of the modelling approach should be improved.

One way to improve is to adopt a two-stage modelling approach as below:

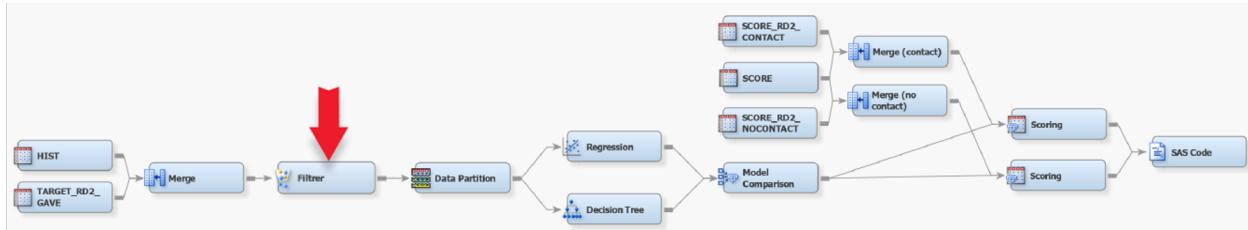
1. Fit a model to determine the ***probability P*** that an individual will give
2. Keep only the data of those who gave, fit a model for ***M (the amount gave)***
3. Use both models to make predictions on the population
4. Compute **$P*M$** to determine the '***expected donation***' of each individual

There are many approaches to 2-stage modelling, but in most cases these steps are required to calculate the value of the uplift (hence the uplift modelling):

- Predict the expected value if a person receives a treatment (here called or contacted)
- Predict the expected value if a person does not receive a treatment (here not called or not contacted)
- Compute the ***difference between the two*** (i.e., the uplift generated by the treatment or targeted action: here the call). The higher the difference is, the treatment (i.e. Contact/ call) is more effective.

Round 2 – Amount Model Description (SAS)

There are 6 steps for game completion: Data Merge, Data Partition, Model Building, Model Comparison, Scoring Data and Export Results. Check [Data Modification](#) for more SAS resources.



Step 1: Data Merge

The first step is to merge two datasets into one for better modification. Keep only the data of those who gave ("Gavethisyear" = 1). You need to deal with missing values after merging the datasets so that you proceed can to the next step — Data Partition.

```
/* merge dataset hist and target_rd2*/
DATA model_rd2_amt;
  MERGE cortex.hist cortex.target_rd2;
  BY ID;
  if Gavethisyear=1;
run;

data results.model_rd2_amt; /*drop na*/
set model_rd2_amt;
if not cmiss(of _numeric_);
run;
```

You can use other way to deal with missing values

[Link for Reference](#)

Step 2: Data Partition

The second step is data partition, this will help the model to increase its performance, please note that this is a default model, you can change the training and validation percentage for a better model performance.

```
/* data partition*/
proc surveymselect data=results.model_rd2_amt rate=0.6
out= donor_select outall
method=srs
seed =1234;
run;

data results.train_rd2_amt results.test_rd2_amt;
set donor_select;
if selected =1 then output results.train_rd2_amt;
else output results.test_rd2_amt;
run;
```

[Link for Reference](#)

Step 3: Model Building

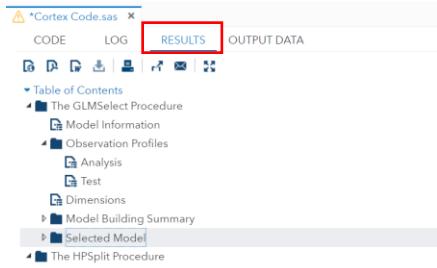
In this step, [linear regression](#) and [decision tree](#) models are used for analysis, you can also choose other model methods (i.e., [logistic regression](#), [HP forest](#), [gradient boosting](#)) to predict the amount donated. We recommend you perform a descriptive analysis to select independent variables for prediction.

```
/* glm model*/
ods graphics off;
proc glmselect data=results.train_rd2_amt testdata=results.test_rd2_amt;
model amtThisYear=Age Salary Seniority GaveLastYear contact/ selection=none;
title 'Regression of donation this year';
'Predictors';
code file='/home/u58717790/Results/regression_2_amt.sas';
run;

/* decision tree model*/
proc arboretum data= results.train_rd2_amt;
target AmtThisYear / level=interval;
input Age Salary / level=interval;
input GaveLastYear Seniority contact/level=nominal;
score data=results.test_rd2_amt role=valid OUT=_NULL_ OUTFIT=results.DT_stat_rd2_amount;
code file='/home/u58717790/Results/decisiontree_2_amt.sas';
quit;
```

Step 4: Model Comparison

When comparing models, you can click RESULTS tab for comparison, SAS will compute statistics automatically, you can then choose [different criteria](#) to determine which model fits better for prediction. In the default model, linear regression model is selected for scoring.



Step 5: Scoring Data

Scoring new data to compute predictions for an existing model is a fundamental stage in the analytics life cycle. You need to predict the amount of donation next year given the donor is contacted and not contacted. After that, you can export the model result for upload.

```
/* predict amtnextyear given contacted */
DATA contact_rd2;
  MERGE cortex.score cortex.score_rd2_contact;
  BY ID;
run;

data results.contact_rd2;
/*drop na*/
set contact_rd2;
if not cmiss(of _numeric_);
run;

data results.amtcontact (keep= id p_amtthisyear rename=(p_amtthisyear=AmtContact));
set results.contact_rd2;
%include '/home/u58717790/Results/regression_2_amt.sas';
run;
```

Predict donation
next year given the
donor is contacted

```

/* predict amtnextyear given not contacted */
DATA nocontact_rd2;
  MERGE cortex.score cortex.SCORE_RD2_NOCONTACT;
  BY ID;
run;

data results.nocontact_rd2; /*drop na*/
set nocontact_rd2;
if not cmiss(of _numeric_);
run;

data results.amtnoncontact (keep= id p amtthisyear rename=(p amtthisyear=AmtNoContact));
set results.nocontact_rd2;
%include '/home/u58717790/Results/regression 2 amt.sas';
run;

```

Predict donation
next year given the
donor is not
contacted

Step 6: Export Results

Merge the two different predictions (amount given contacted and amount given not contacted) by ID, and then export as CSV file. Following code will help you to export the model result into a CSV file.

```

/* Export data */

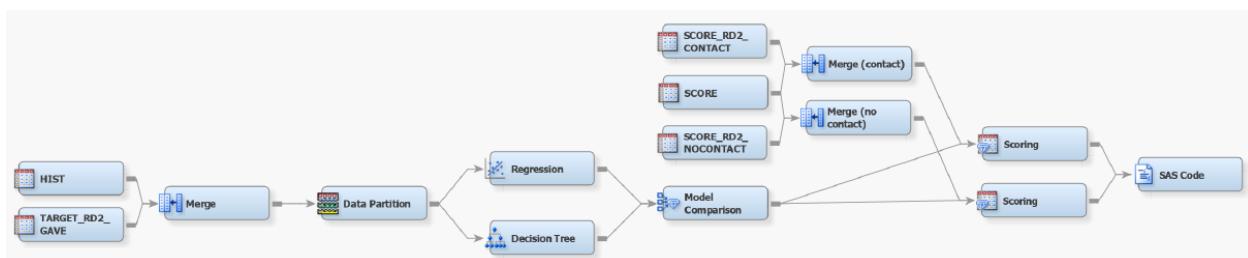
DATA results.rd2_output_amt;
  MERGE results.amtcontact results.amtnoncontact ;
  BY ID;
run;

proc export data=results.rd2_output_amt
outfile="/home/u58717790/Results/Round2 Output amt.csv" dbms=csv
replace;
run;

```

Round 2 – Probability Model Description (SAS)

There are 6 steps for game completion: Data Merge, Data Partition, Model Building, Model Comparison, Scoring Data and Export Results. Check [Data Modification](#) for more SAS resources.



Step 1: Data Merge

The first step is to merge two datasets into one for better modification. You need to deal with missing values after merging the datasets so that you proceed can to the next step — Data Partition.

```
/* merge dataset hist and target_rd2*/
DATA model_rd2_prob;
  MERGE cortex.hist cortex.target_rd2;
  BY ID;
run;

data results.model_rd2_prob; /*drop na*/
set model_rd2_prob;
if not cmiss(of _numeric_);
run;
```

You can use other
way to deal with
missing values

[Link for Reference](#)

Step 2: Data Partition

The second step is data partition, this will help the model to increase its performance, please note that this is a default model, you can change the training and validation percentage for a better model performance.

```
/* data partition*/
proc surveyselect data=results.model_rd2_prob rate=0.6
out= donor_select outall
method=srs
seed =1234;
run;

data results.train_rd2_prob results.test_rd2_prob;
set donor_select;
if selected =1 then output results.train_rd2_prob;
else output results.test_rd2_prob;
run;
```

[Link for Reference](#)

Step 3: Model Building

In this step, [linear regression](#) and [decision tree](#) models are used for analysis, you can also choose other model methods (i.e., [logistic regression](#), [HP forest](#), [gradient boosting](#)) to predict the amount donated. We recommend you perform a descriptive analysis to select independent variables for prediction.

```
/* glm model*/
ods graphics off;
proc glmselect data=results.train_rd2_prob testdata=results.test_rd2_prob;
model gaveThisYear=Age Salary Seniority GaveLastYear contact;
title 'Regression of donation this year '
'Predictors';
code file='/home/u58717790/Results/regression_2_prob.sas';
run;

/* decision tree model*/
proc arboretum data= results.train_rd2_prob;
target GaveThisYear / level=nominal;
input Age Salary / level=interval;
input GaveLastYear Seniority contact/level=nominal;
score data=results.test_rd2_prob role=valid OUT=NULL OUTFIT=results.DT_stat_rd2_prob;
code file='/home/u58717790/Results/decisiontree_2_prob.sas';
quit;
```

Step 4: Model Comparison



When comparing models, you can click RESULTS tab for comparison, SAS will compute statistics automatically, you can then choose [different criteria](#) to determine which model fits better for prediction. In the default model, linear regression model is selected for scoring.

A screenshot of the SAS Results interface. At the top, there are tabs labeled CODE, LOG, RESULTS (which is highlighted with a red box), and OUTPUT DATA. Below the tabs is a toolbar with various icons. The main area shows a hierarchical Table of Contents. Under 'Table of Contents', there are sections for 'The GLMSelect Procedure', 'Model Information', 'Observation Profiles' (which has 'Analysis' and 'Test' as sub-sections), 'Dimensions', 'Model Building Summary', 'Selected Model' (which is highlighted with a blue box), and 'The HPSplit Procedure'. A horizontal scrollbar is visible at the bottom of the content area.

Step 5: Scoring Data

You need to predict the *probabilities of donating* next year given the donor is contacted and not contacted. After that, you can export the model result for upload.

```
/* predict givennextyear given contacted */
DATA contact_rd2;
  MERGE cortex.score cortex.score_rd2_contact;
  BY ID;
run;

data results.contact_rd2; /*drop na*/
set contact_rd2;
if not cmiss(of _numeric_);
run;

data results.predcontact (keep= id p_gavethisyear rename=(p_gavethisyear=PContact));
set results.contact_rd2;
%include '/home/u58717790/Results/regression_2_prob.sas';
run;

/* predict givennextyear given not contacted */

DATA nocontact_rd2;
  MERGE cortex.score cortex.SCORE_RD2_NOCONTACT;
  BY ID;
run;

data results.nocontact_rd2; /*drop na*/
set nocontact_rd2;
if not cmiss(of _numeric_);
run;

data results.prednoncontact (keep= id p_gavethisyear rename=(p_gavethisyear=PNonoContact));
set results.nocontact_rd2;
%include '/home/u58717790/Results/regression_2_prob.sas';
run;
```

Predict probability
of donate next
year given the
donor is contacted

Predict probability
of donate next year
given the donor is
NOT contacted.

Step 6: Export Results

Merge the two different predictions (probability given contacted and probability given not contacted) by ID, and then export as CSV file.

```
/* Export data*/

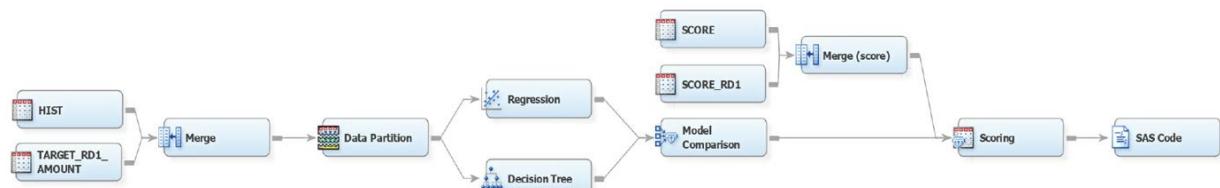
DATA results.rd2_output_prob;
  MERGE results.predcontact results.prednoncontact;
  BY ID;
run;

proc export data=results.rd2_output_prob
outfile="/home/u58717790/Results/Round2 Output prob.csv" dbms=csv
replace;
run;
```

Round 2 - Amount Model Description (Python)

NOTE: You need to install **saspy**, **pandas**, **numpy**, and **sklearn** python packages to run the baseline model in your python command.

There are 6 steps for game completion: Data Merge, Data Partition, Model Building, Model Comparison, Scoring Data and Export Results.



Step 1: Data Merge

The first step is to merge two datasets into one for better modification. You need to deal with missing values after merging the datasets so that you can proceed to the next step — Data Partition.

```
#Step1 Merge the Data
data_merge = pd.merge(data1, data2, on=["ID"], how="right")
data_merge = data_merge.loc[(data_merge['GaveThisYear'] == 1)]

#Deal with Missing Value
data_merge = data_merge.dropna() #comment: you could use another method to deal with NA
data_merge.head()
```

Step 2: Data Partition

The second step is data partition, this will help the model to increase its performance, please note that this is a default model, you can change the training and validation percentage for a better model performance.

```
#Step2 Data Partition
#this is just a sample, you could use another library or bulit in funciton
from sklearn.model_selection import train_test_split
train, validation = train_test_split(data_merge, test_size=0.4) # you can change the percentage
train.head()
```

Step 3: Model Building

In this step, linear regression and decision tree models are used for analysis, you can also choose other model methods (i.e., logistic regression, HP forest, gradient boosting) to predict the amount donated. We recommend you perform a descriptive analysis to select independent variables for prediction.

```
from sklearn import linear_model

X_train = train[['Age', 'Salary', 'Seniority', 'GaveLastYear', 'Contact']]
Y_train = train['AmtThisYear']
X_valid = validation[['Age', 'Salary', 'Seniority', 'GaveLastYear', 'Contact']]
Y_valid = validation['AmtThisYear']

regr = linear_model.LinearRegression()
regr.fit(X_train,Y_train)
regr_predict=regr.predict(X_valid)
```

```
#you can change the criteria

import numpy as np
from sklearn import metrics
#MAE
print(metrics.mean_absolute_error(Y_valid,regr_predict))
#MSE
print(metrics.mean_squared_error(Y_valid,regr_predict))
#RMSE
print(np.sqrt(metrics.mean_squared_error(Y_valid,regr_predict)))

69.17860894186053
68533.36200899993
261.7887736496734
```

```
from sklearn.tree import DecisionTreeRegressor

X_train = train[['Age', 'Salary', 'Seniority', 'GaveLastYear', 'Contact']]
Y_train = train['AmtThisYear']
X_valid = validation[['Age', 'Salary', 'Seniority', 'GaveLastYear', 'Contact']]
Y_valid = validation['AmtThisYear']

DT_model = DecisionTreeRegressor(max_depth=5).fit(X_train,Y_train)
DT_predict = DT_model.predict(X_valid) #Predictions on Testing data
print(DT_predict)

[ 55.1316166  63.68324351  54.83150079 ... 113.67346939  55.1316166
 63.68324351]
```

```
#you can change the criteria
#MAE
print(metrics.mean_absolute_error(Y_valid,DT_predict))
#MSE
print(metrics.mean_squared_error(Y_valid,DT_predict))
#RMSE
print(np.sqrt(metrics.mean_squared_error(Y_valid,DT_predict)))

68.62514956955992
68426.50239010017
261.58459891610624
```

Step 4: Model Comparison

When comparing models, you need to choose a criterion and calculate it manually to determine which model fits better for prediction since Python will not compute the statistical results automatically. In the default model, we compare the MSE of two model, and the linear regression model is selected for scoring.

Step 5: Scoring Data

Scoring new data to compute predictions for an existing model is a fundamental stage in the analytics life cycle, after you score out the dataset, you can export the model result for upload.

```
#Predict amount give given contact

scoring_data_contact = pd.merge(data3, data4, on=["ID"], how="right")
scoring_data_contact = scoring_data_contact.dropna()
scoring_data_contact.head()

X = scoring_data_contact[['Age', 'Salary', 'Seniority', 'GaveLastYear', 'Contact']]
regr_predict_contact=regr.predict(X)
scoring_data_contact['Prediction'] = regr_predict_contact

scoring_data_contact= scoring_data_contact[['ID', 'Prediction']]
scoring_data_contact = scoring_data_contact.rename({'Prediction': 'AmtContact'}, axis=1)
scoring_data_contact.head()

#Predict amount give given not contact

scoring_data_nocontact = pd.merge(data3, data5, on=["ID"], how="right")
scoring_data_nocontact = scoring_data_nocontact.dropna()
scoring_data_nocontact.head()

X = scoring_data_nocontact[['Age', 'Salary', 'Seniority', 'GaveLastYear', 'Contact']]
regr_predict_nocontact=regr.predict(X)
scoring_data_nocontact['Prediction'] = regr_predict_nocontact

scoring_data_nocontact= scoring_data_nocontact[['ID', 'Prediction']]
scoring_data_nocontact = scoring_data_nocontact.rename({'Prediction': 'AmtNoContact'}, axis=1)
scoring_data_nocontact.head()

result = pd.merge(scoring_data_contact, scoring_data_nocontact, on=["ID"], how="right")
result.sort_values(by=['ID'], inplace=True)
result.head()
```

Step 6: Export Results

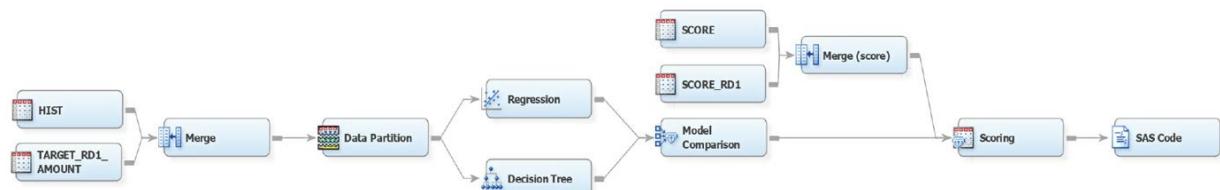
Following code will help you to export the model result into a CSV file. You can sort the data by predicted amount in the descending order before export the file, this will help you determine number of donors you want to contact.

```
Result.to_csv('Round2_Output_amt.csv', index=False)
```

Round 2 - Amount Model Description (Python)

NOTE: You need to install *saspy*, *pandas*, *numpy*, and *sklearn* python packages to run the baseline model in your python command.

There are 6 steps for game completion: Data Merge, Data Partition, Model Building, Model Comparison, Scoring Data and Export Results.



Step 1: Data Merge

The first step is to merge two datasets into one for better modification. You need to deal with missing values after merging the datasets so that you can proceed to the next step — Data Partition.

```
#Step1 Merge the Data
data_merge = pd.merge(data1, data2, on=["ID"], how="right")
data_merge.head()
#Deal with Missing Value
data_merge = data_merge.dropna() #comment: you could use another method to deal with NA
data_merge.head()
```

Step 2: Data Partition

The second step is data partition, this will help the model to increase its performance, please note that this is a default model, you can change the training and validation percentage for a better model performance.

```
#Step2 Data Partition
#this is just a sample, you could use another library or bulit in funciton
from sklearn.model_selection import train_test_split
train, validation = train_test_split(data_merge, test_size=0.4) # you can change the percentage
train.head()
```

Step 3: Model Building

In this step, linear regression and decision tree models are used for analysis, you can also choose other model methods (i.e., logistic regression, HP forest, gradient boosting) to predict the amount donated. We recommend you perform a descriptive analysis to select independent variables for prediction.

```
from sklearn import linear_model

#comment: it's numpy array ==> categorical variables
X_train = train[['Age', 'Salary', 'Seniority', 'GaveLastYear', 'Contact']]
Y_train = train['GaveThisYear']
X_valid = validation[['Age', 'Salary', 'Seniority', 'GaveLastYear', 'Contact']]
Y_valid = validation['GaveThisYear']

regr = linear_model.LinearRegression()
regr.fit(X_train,Y_train)
regr_predict=regr.predict(X_valid)
```

```
#you can change the criteria

import numpy as np
from sklearn import metrics
#MAE
print(metrics.mean_absolute_error(Y_valid,regr_predict))
#MSE
print(metrics.mean_squared_error(Y_valid,regr_predict))
#RMSE
print(np.sqrt(metrics.mean_squared_error(Y_valid,regr_predict)))

0.3118823356074323
0.15618097384416432
0.39519738592779724
```

```
from sklearn.tree import DecisionTreeRegressor

X_train = train[['Age', 'Salary', 'Seniority', 'GaveLastYear', 'Contact']]
Y_train = train['GaveThisYear']
X_valid = validation[['Age', 'Salary', 'Seniority', 'GaveLastYear', 'Contact']]
Y_valid = validation['GaveThisYear']

DT_model = DecisionTreeRegressor(max_depth=5).fit(X_train,Y_train)
DT_predict = DT_model.predict(X_valid) #Predictions on Testing data
print(DT_predict)

[0.16870329 0.28120045 0.17655114 ... 0.28120045 0.15828309 0.15828309]
```

```
#you can change the criteria
#MAE
print(metrics.mean_absolute_error(Y_valid,DT_predict))
#MSE
print(metrics.mean_squared_error(Y_valid,DT_predict))
#RMSE
print(np.sqrt(metrics.mean_squared_error(Y_valid,DT_predict)))

0.3094375110885283
0.1552100484136524
0.39396706513825797
```

Step 4: Model Comparison

When comparing models, you need to choose a criterion and calculate it manually to determine which model fits better for prediction since Python will not compute the statistical results automatically. In the default model, we compare the MSE of two model, and the linear regression model is selected for scoring.

Step 5: Scoring Data

Scoring new data to compute predictions for an existing model is a fundamental stage in the analytics life cycle, after you score out the dataset, you can export the model result for upload.

```
#Predict prob of give given contact

scoring_data_contact = pd.merge(data3, data4, on=["ID"], how="right")
scoring_data_contact = scoring_data_contact.dropna()
scoring_data_contact.head()

X = scoring_data_contact[['Age', 'Salary', 'Seniority', 'GaveLastYear', 'Contact']]
regr_predict_contact=regr.predict(X)
scoring_data_contact['Prediction'] = regr_predict_contact

scoring_data_contact= scoring_data_contact[['ID', 'Prediction']]
scoring_data_contact = scoring_data_contact.rename({'Prediction': 'ProbContact'}, axis=1)
scoring_data_contact.head()
```

```
#Predict prob of give given not contact

scoring_data_nocontact = pd.merge(data3, data5, on=["ID"], how="right")
scoring_data_nocontact = scoring_data_nocontact.dropna()
scoring_data_nocontact.head()

X = scoring_data_nocontact[['Age', 'Salary', 'Seniority', 'GaveLastYear', 'Contact']]
regr_predict_nocontact=regr.predict(X)
scoring_data_nocontact['Prediction'] = regr_predict_nocontact
scoring_data_nocontact= scoring_data_nocontact[['ID', 'Prediction']]
scoring_data_nocontact = scoring_data_nocontact.rename({'Prediction': 'ProbNoContact'}, axis=1)
scoring_data_nocontact.head()
```

```
result = pd.merge(scoring_data_contact, scoring_data_nocontact, on=["ID"], how="right")
result.sort_values(by=['ID'], inplace=True)
result.head()
```

Step 6: Export Results

Following code will help you to export the model result into a CSV file. You can sort the data by predicted amount in the descending order before export the file, this will help you determine number of donors you want to contact.

```
Result.to_csv('Round2_Output_prob.csv', index=False)
```

Round 2 - Upload Decisions

Copy and paste these five columns into a new CSV file.

	A	B	C	D	E	F	G	H
1	ID	PContact	PNoContact					
2	2000001	0.36622	0.156038					
3	2000010	0.350215	0.140033					
4	2000015	0.397774	0.187593					
5	2000016	0.514806	0.304624					
6	2000017	0.464418	0.254236					
7	2000020	0.372183	0.162001					
8	2000029	0.482034	0.271852					
9	2000033	0.508128	0.297947					
10	2000038	0.371945	0.161763					
11	2000039	0.479945	0.269763					
12	2000041	0.374053	0.163872					
13	2000044	0.354092	0.14391					
14	2000047	0.355683	0.145501					
15	2000048	0.358991	0.148809					
16	2000050	0.35061	0.140429					
17	2000053	0.348137	0.137955					
18	2000054	0.387792	0.177611					
19	2000056	0.365499	0.155317					
20	2000057	0.497774	0.287592					
21	2000062	0.381978	0.171796					
22	2000063	0.483542	0.273361					
23	2000064	0.390385	0.180203					
24	2000066	0.376888	0.166706					
25	2000067	0.393756	0.183575					
26	2000069	0.334106	0.123924					
27	2000070	0.485404	0.275222					
28	2000078	0.479068	0.268886					
29	2000081	0.403616	0.193434					
30	2000083	0.399815	0.189634					
31	2000085	0.360897	0.150716					
32	2000088	0.409312	0.19913					
33	2000090	0.455577	0.245395					
34	2000092	0.505388	0.295206					
35	2000094	0.344085	0.133904					
36	2000096	0.39193	0.181748					

	A	B	C	D	E	F	G
1	ID	AmtContact	AmtNoContact	PContact	PNoContact	Expected Donations	
2	2000001	90.90781652	90.90781652	0.36622015	0.156038452	=B2*D2-C2*E2	
3	2000010	82.08431557	82.08431557	0.350215173	0.140033474		
4	2000015	78.22825441	78.22825441	0.397774397	0.187592699		
5	2000016	79.22104693	79.22104693	0.51480568	0.304623982		
6	2000017	49.27271395	49.27271395	0.464418022	0.254236323		
7	2000020	48.8660076	48.8660076	0.372182882	0.162001184		

Create a new column names “Expected Donations” and calculate it.

	A	B	C	D	E	F	G
1	ID	AmtContact	AmtNoContact	PContact	PNoContact	Expected Donations	
2	2000001	90.90781652	90.90781652	0.36622015	0.156038452	=B2*D2-C2*E2	
3	2000010	82.08431557	82.08431557	0.350215173	0.140033474		
4	2000015	78.22825441	78.22825441	0.397774397	0.187592699		
5	2000016	79.22104693	79.22104693	0.51480568	0.304623982		
6	2000017	49.27271395	49.27271395	0.464418022	0.254236323		
7	2000020	48.8660076	48.8660076	0.372182882	0.162001184		

Sort all columns by descending amount of Expected Donations.

The screenshot shows a Microsoft Excel spreadsheet with data in rows 1 through 18. The columns are labeled A through L. Row 1 contains the headers: ID, AmtContact, AmtNoContact, PContact, PNoContact, and Expected Donations. Rows 2 through 18 contain data points. A cursor is in cell A2. A 'Sort' dialog box is open over the spreadsheet, with the following settings:

- Add Level:** Not selected.
- Delete Level:** Not selected.
- Copy Level:** Selected.
- Options...**: Not selected.
- My data has headers:** Selected.
- Column:** Sort by 'Expected Donations'.
- Sort On:** Cell Values.
- Order:** Largest to Smallest.

Select the number of people ID you want to contact and keep this information in the CSV file for upload.

ONLY keep ID
numbers in the
CSV file for upload.

The screenshot shows a Microsoft Excel spreadsheet with data in rows 1 through 18. The columns are labeled A through I. Row 1 contains the headers: ID, AmtContact, AmtNoContact, PContact, PNoContact, and Expected Donations. Rows 2 through 18 contain data points. A large blue arrow points from the text "ONLY keep ID numbers in the CSV file for upload." to the first column (ID) of the data table.

Then you can follow the same steps mentioned in Round 1 to upload your model results.

Dataset Modification:

Data Exploration: Click [here](#) to learn how to explore the data and the variables in the dataset.

Model Comparison: Click [here](#) to learn the criteria for model comparisons and pick the model performed best for upload.

Removing Variables: Click [here](#) to learn how to remove some of the variables from your models.

Transform Variables: Click [here](#) to learn how to transform variables to make them more symmetrical can improve performance.

Replacing Missing Variables: Click [here](#) to learn how to deal with missing variables in the dataset.

