THE POWER TO KNOW.

SAS Cortex Coding Version Guide

SAS® Documentation

Table of Contents

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.

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.

Fundraising Scenario

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

Cortex ANALYTICS SIMULATION

Last Update date: November 27, 2020

Figure 1. Fundraising Scenario

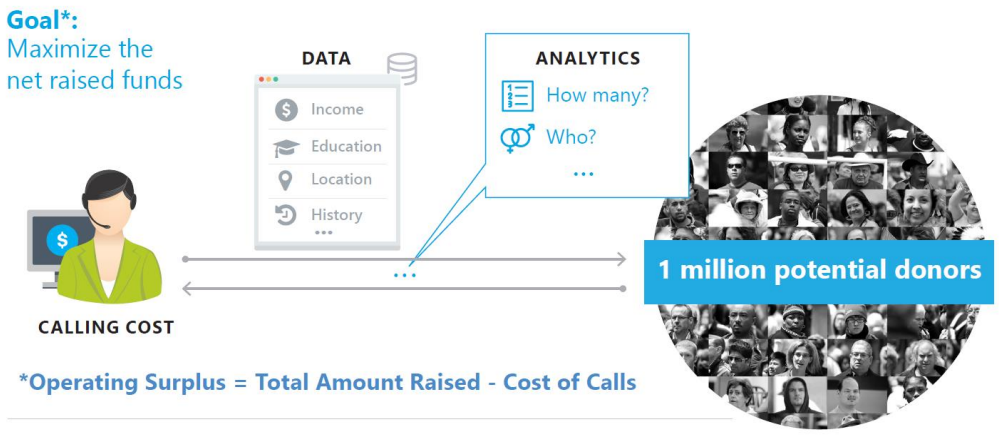


Table 1. Cost Structure

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)	ID data
LastName	Last Name	
FirstName	First Name	
Woman	Sex (1=woman, 0=man)	Socio-demographic
Age	Age (years)	
Salary	Annual salary in USD	
Education	Highest education level	
City	Type of neighborhood	
SeniorList	Seniority for being on the VIP list	History*
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	Target
AmtLastYear	Amount given last year	
GaveThisYear	Whether or not the individual gave this year	
AmtThisYear	Amount given this year	

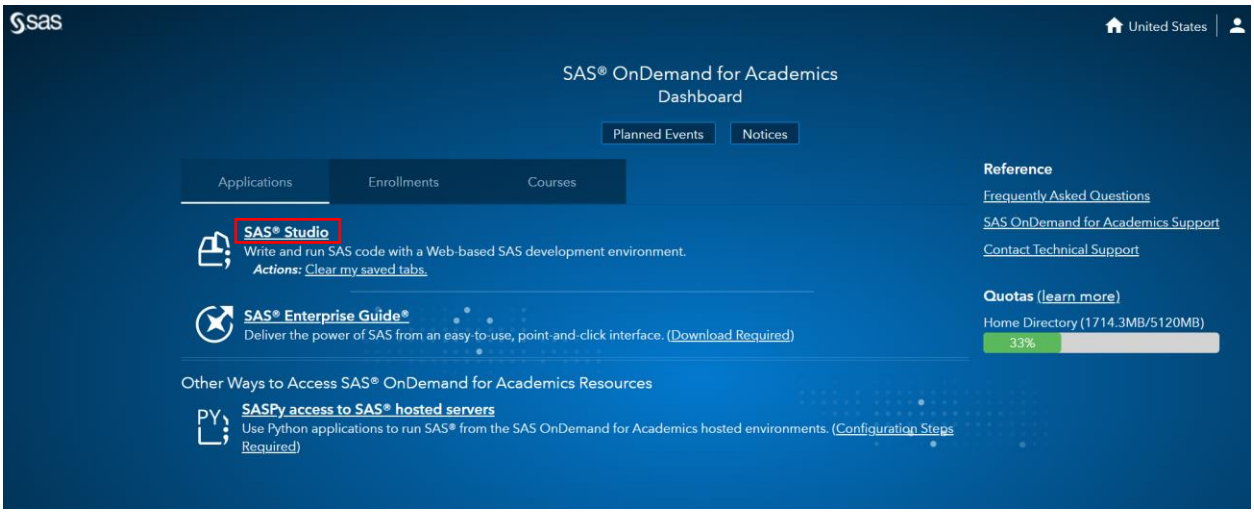
* History (HIST) dataset gives the history of 10 years leading up to, but excluding, last year.

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

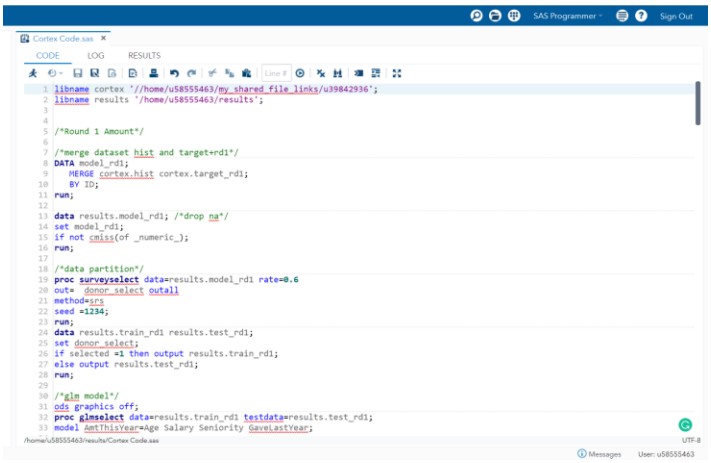
How to play Cortex in SAS OnDemand for Academics (ODA)

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.



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
```

Step1-2 Merge and DataPartition

```
%%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 dataframe(pandas) ==> take a while
data1 = sas_session.sasdata2dataframe(
table='hist',
libref='cortex'
)

data2 = sas_session.sasdata2dataframe(
table='target_rd1',
libref='cortex'
)

#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()

#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()
```

SASPy Installation and Configuration

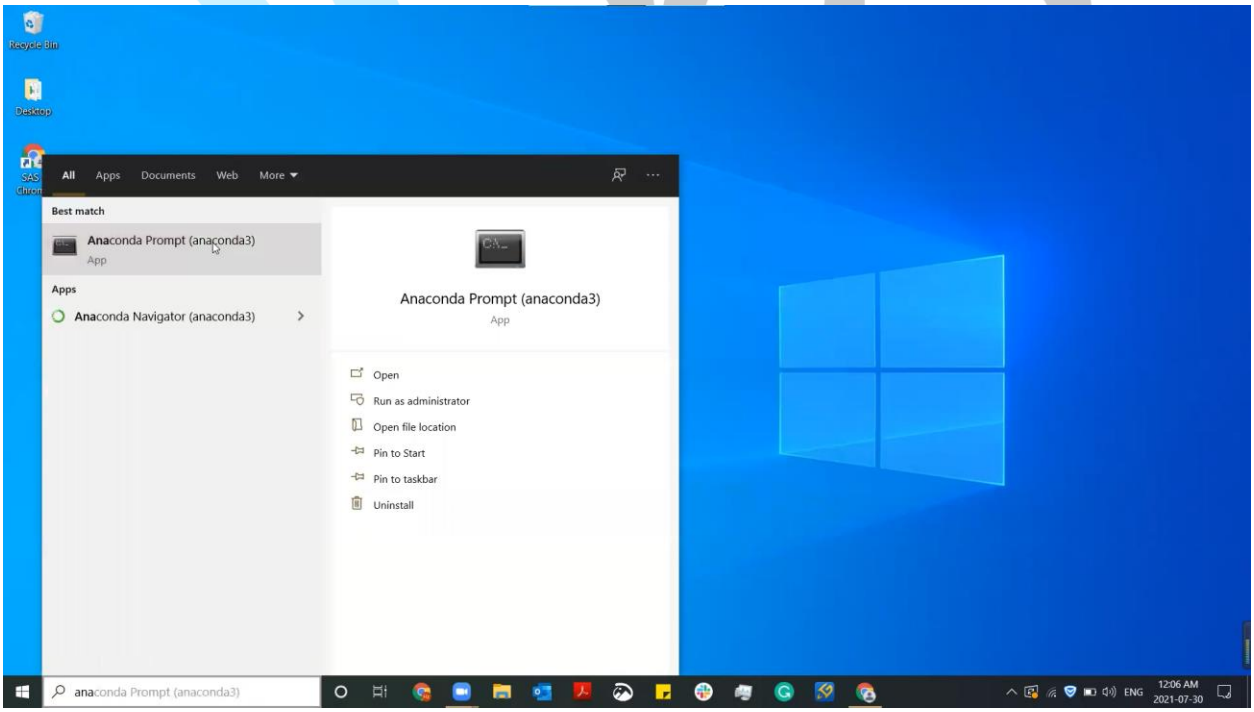
To install the SASPy in your Python Command, you need to ensure that you meet the prerequisite:

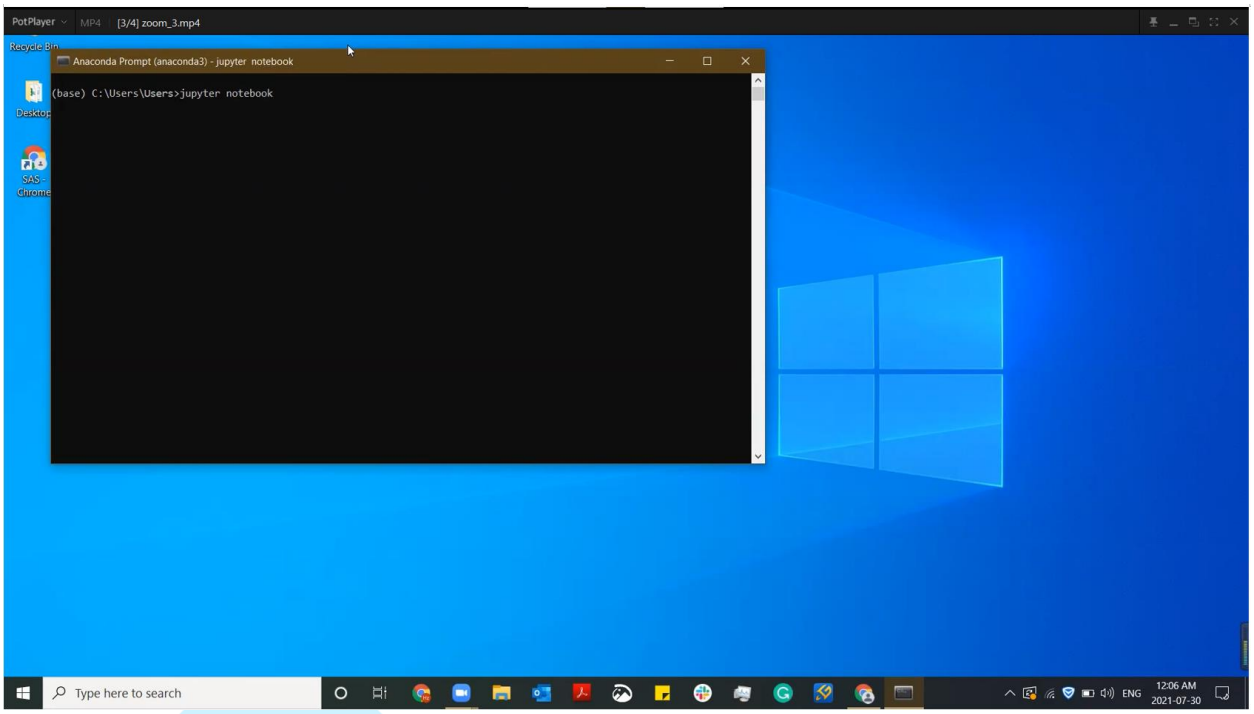
1. Java version 1.8.0_162 or higher
2. Python 3.3 or higher
3. SASPy 3.3.4 or higher

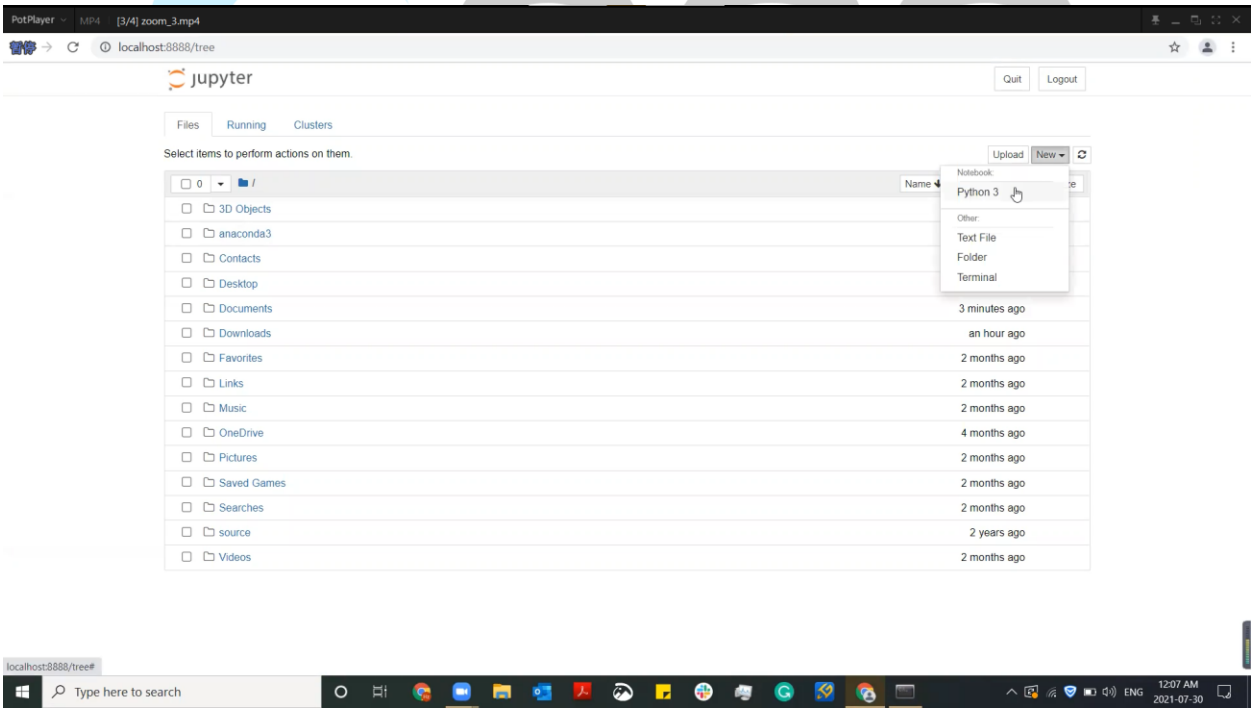
If you do not have Java installed, click here to download and install the newest version of Java.
If you do not have Jupyter Notebook installed, click here to download and install the newest version of Anaconda Prompt and open Jupyter Notebook.

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.

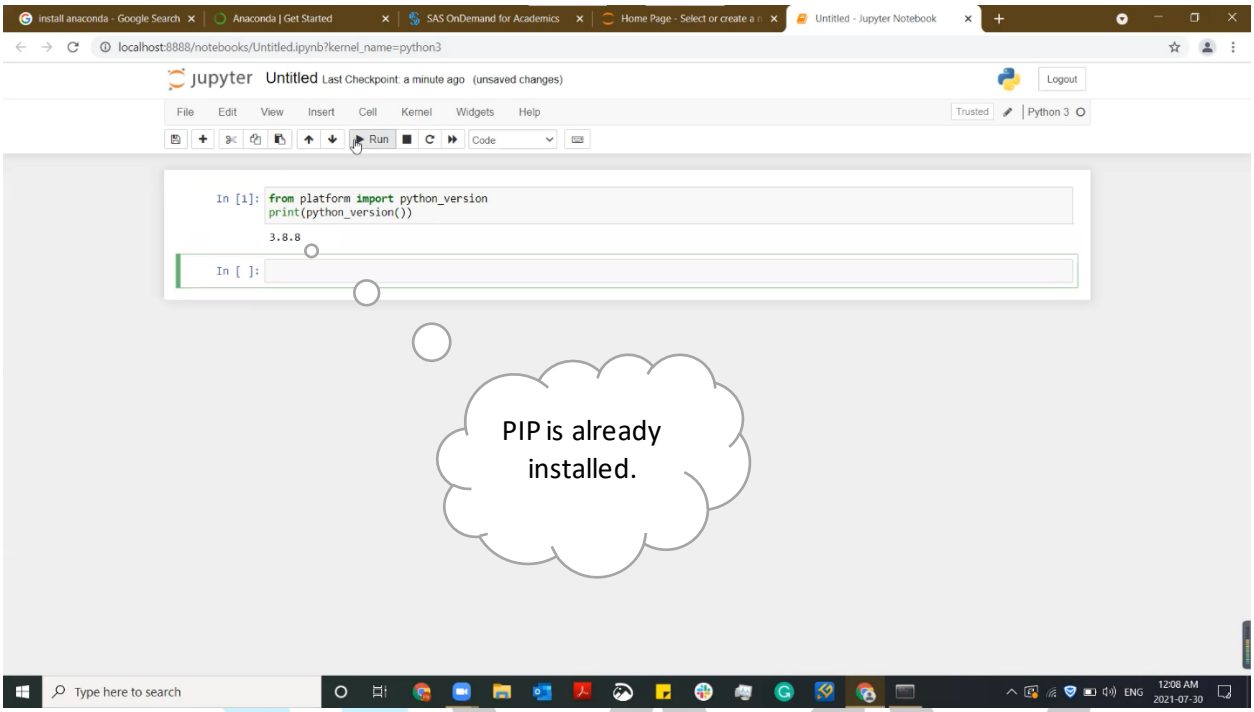



PotPlayer
MP4
[3/4] zoom_3.mp4
Recycle Bin
Desktop
SAS - Chrome
Anaconda Prompt (anaconda3) - jupyter notebook
(base) C:\Users\Users>jupyter notebook
Type here to search
ENG
12:06 AM
2021-07-30

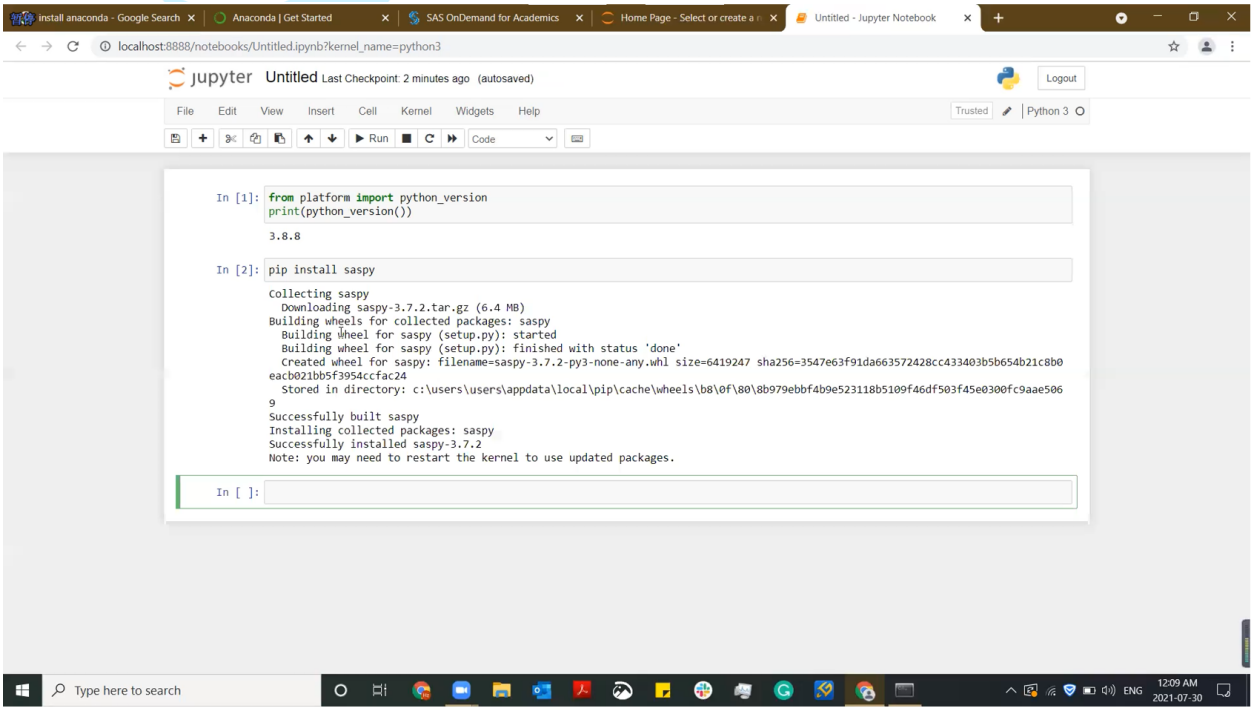

PotPlayer
MP4
[3/4] zoom_3.mp4
localhost:8888/tree
jupyter
Quit
Logout
Files
Running
Clusters
Select items to perform actions on them.
Upload
New
Notebook:
Python 3
Other:
Text File
Folder
Terminal
3D Objects
anaconda3
Contacts
Desktop
Documents
3 minutes ago
Downloads
an hour ago
Favorites
2 months ago
Links
2 months ago
Music
2 months ago
OneDrive
4 months ago
Pictures
2 months ago
Saved Games
2 months ago
Searches
2 months ago
source
2 years ago
Videos
2 months ago
localhost:8888/tree#
Type here to search
ENG
12:07 AM
2021-07-30

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.

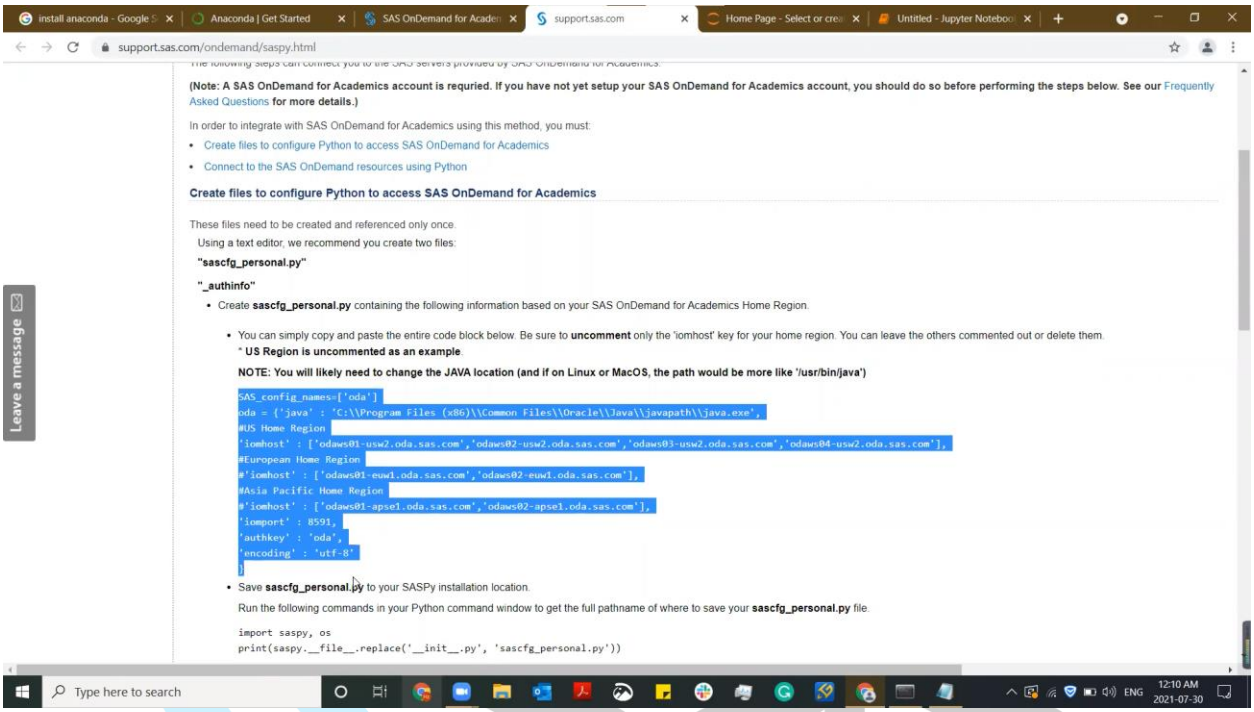
```
from platform import python_version
print (python_version())
```

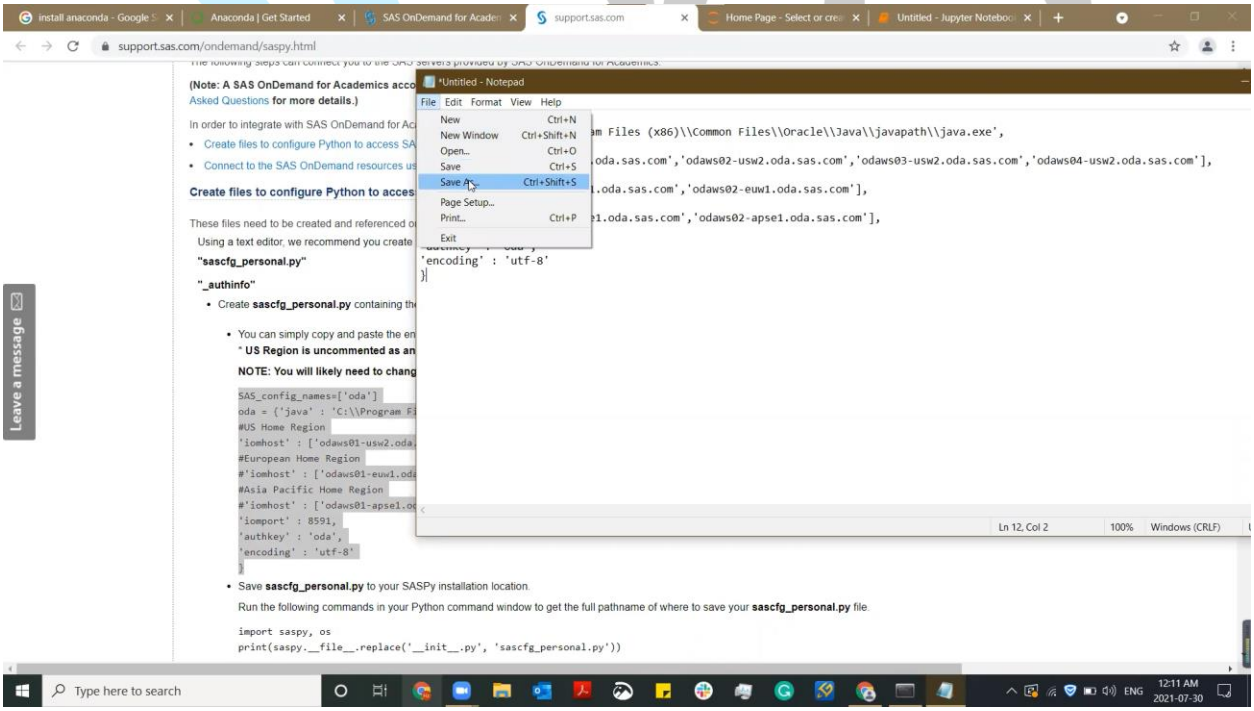


Execute the code ***pip install saspy*** in the Jupyter Notebook to install SASPy.

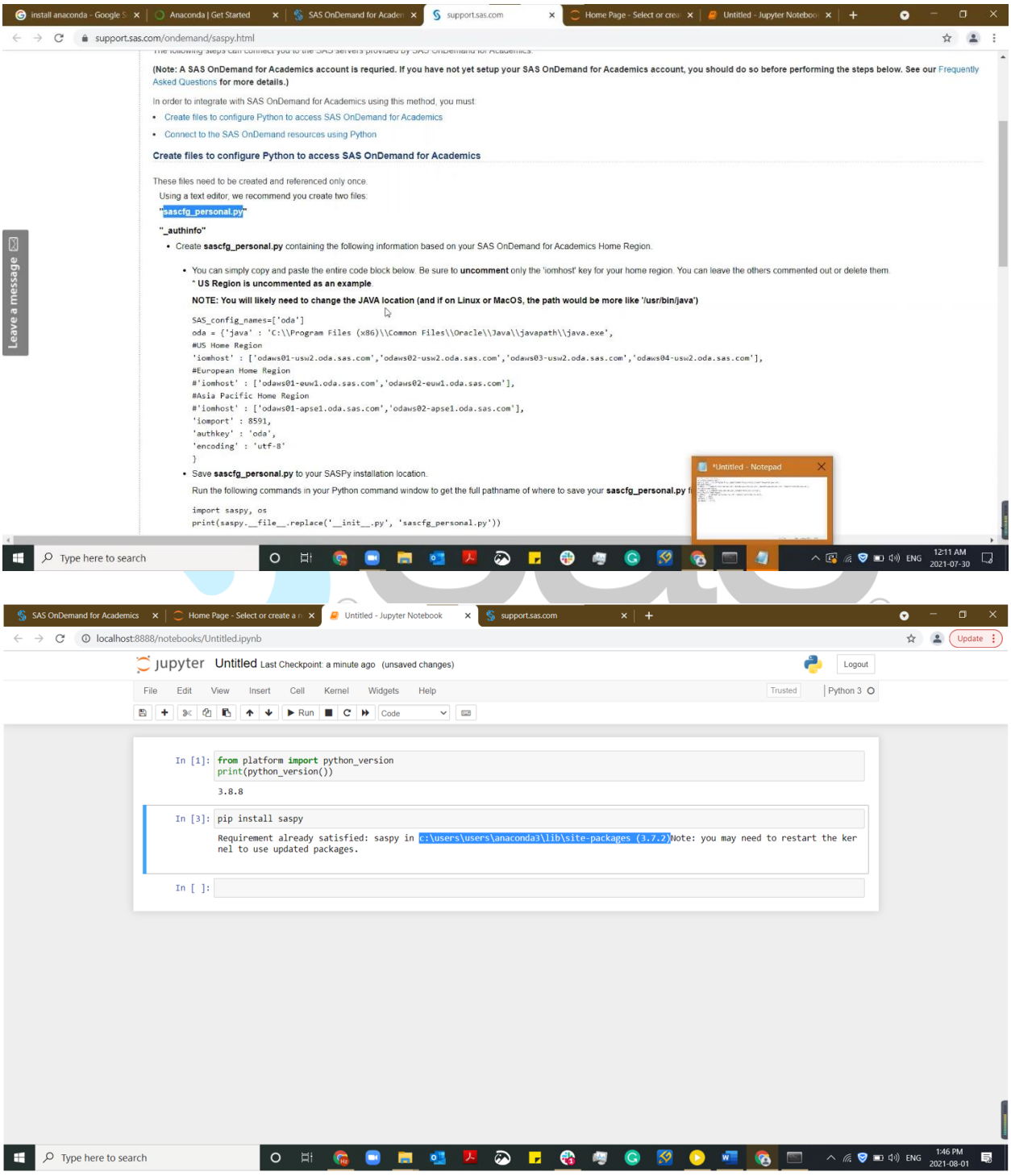


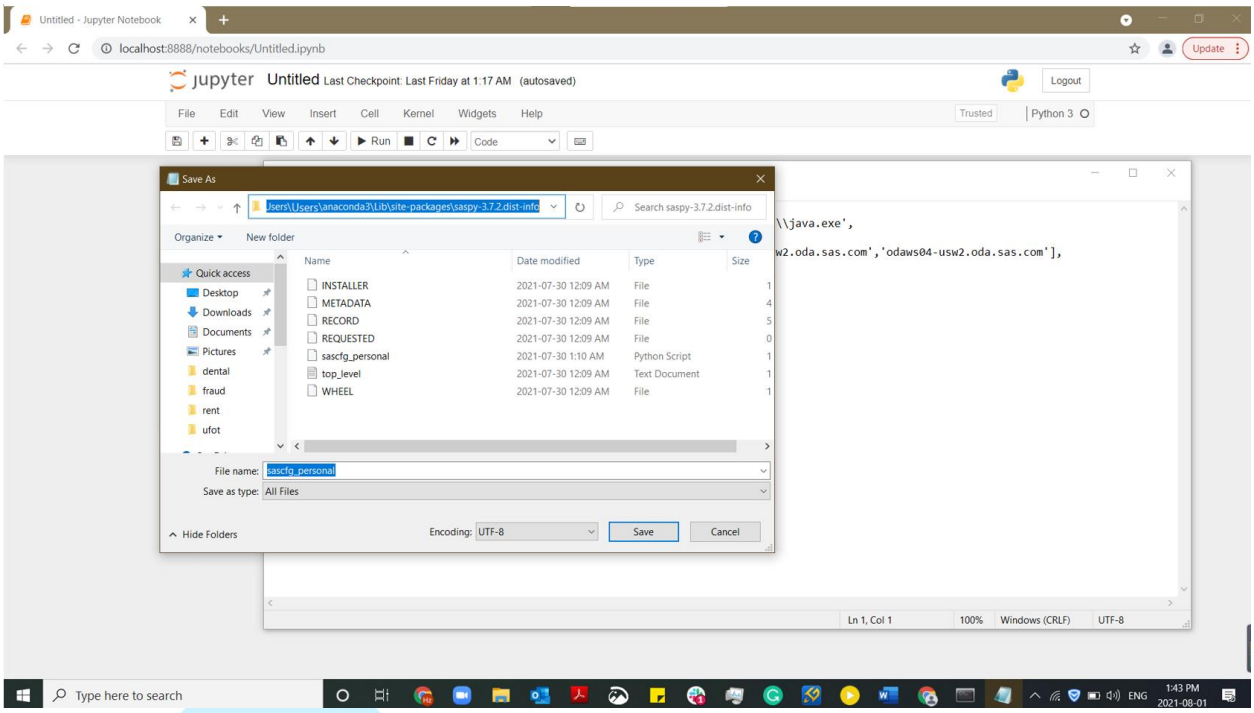
Click here to open the SASPy support page, copy the highlighted text and paste it to a newly created text editor.



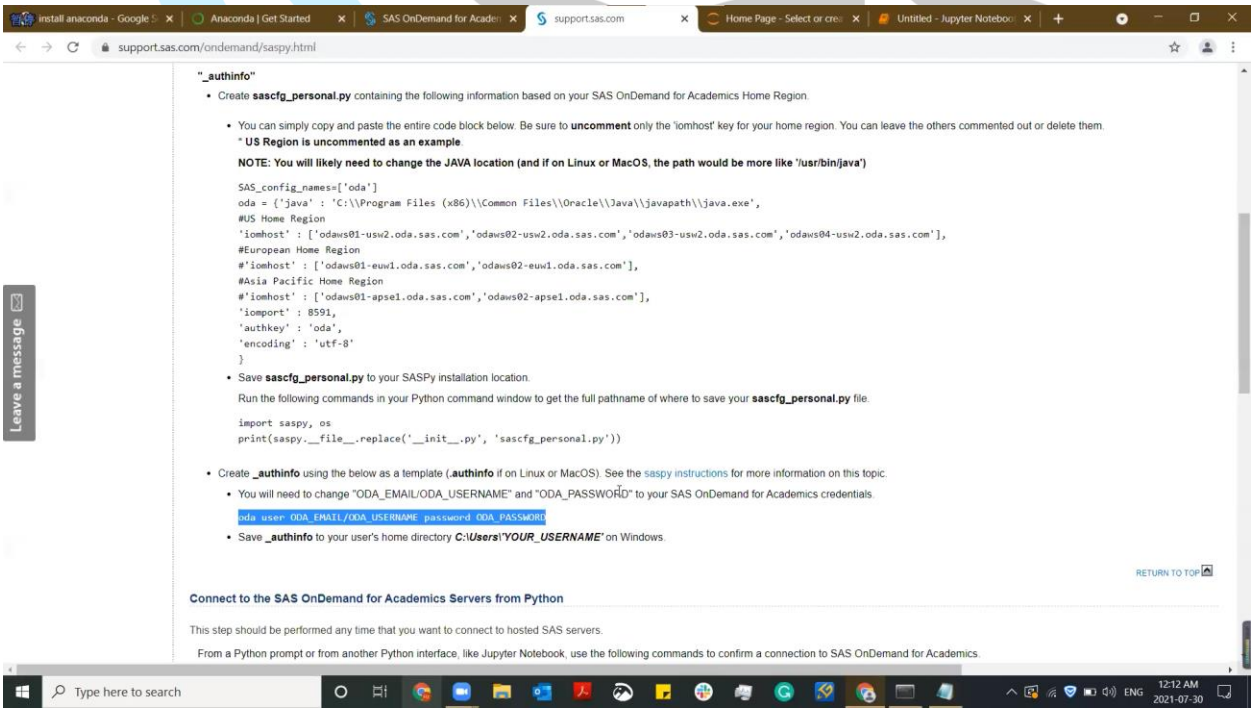


Save the text editor named ***sascfg_personal.py*** to your SASPy installation location, you can find this location by running the code ***pip install saspy*** again in the Jupyter Notebook. You need to change the file type to All Files before you save it, if the Notepad in your Windows does not allow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

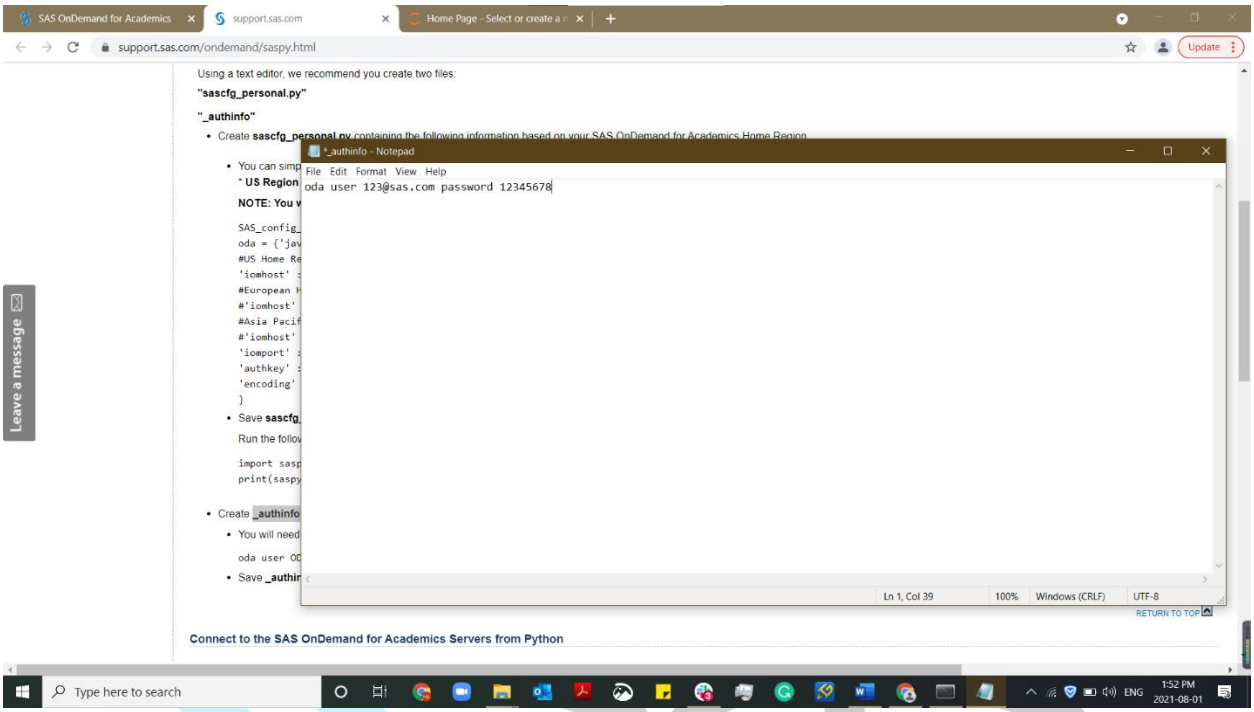




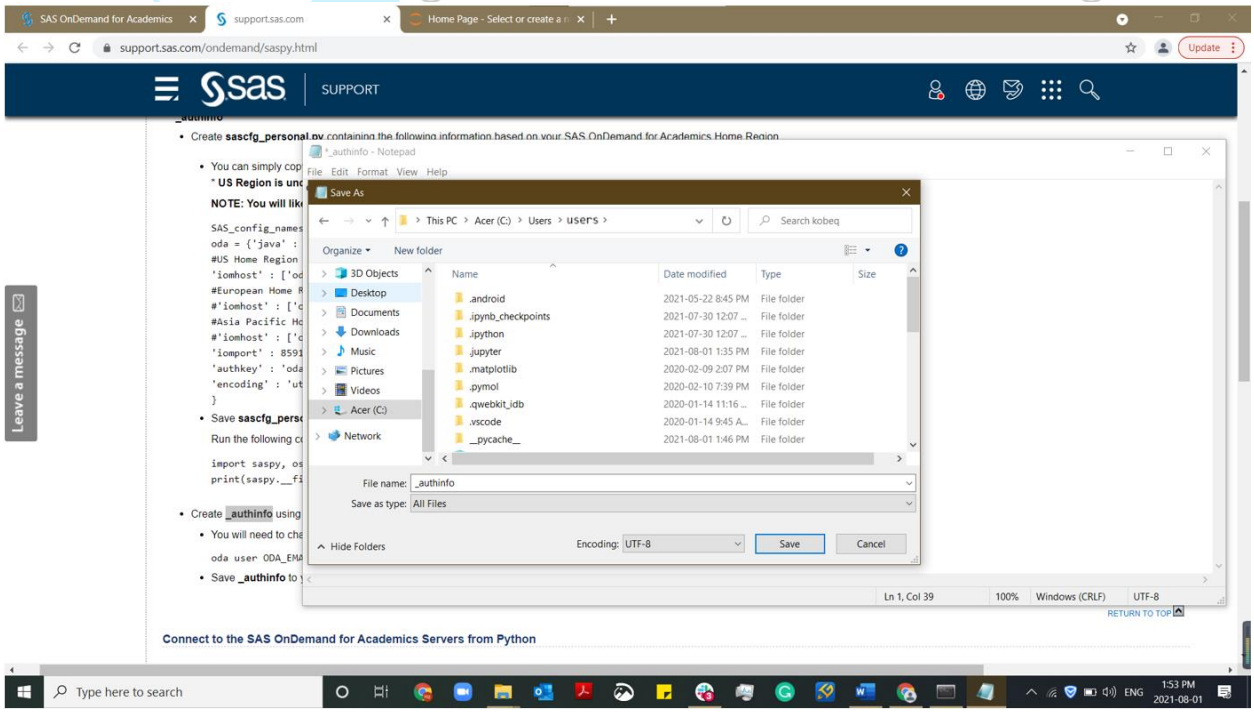
Create another text editor, copy the text highlighted below and paste it to this new text editor.



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.

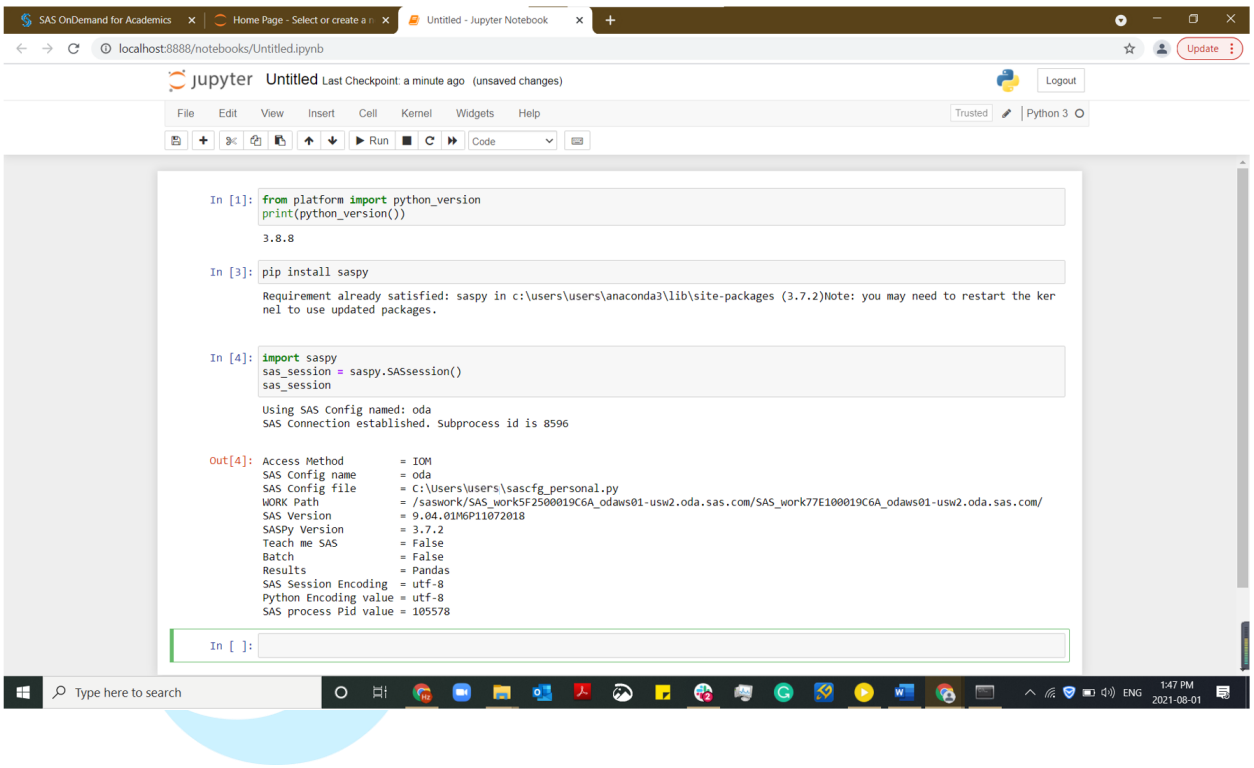


Save ***_authinfo*** to your user's home directory *C:\Users\'YOUR_USERNAME'* on Windows. You need to change the file type to All Files before you save it.



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.***

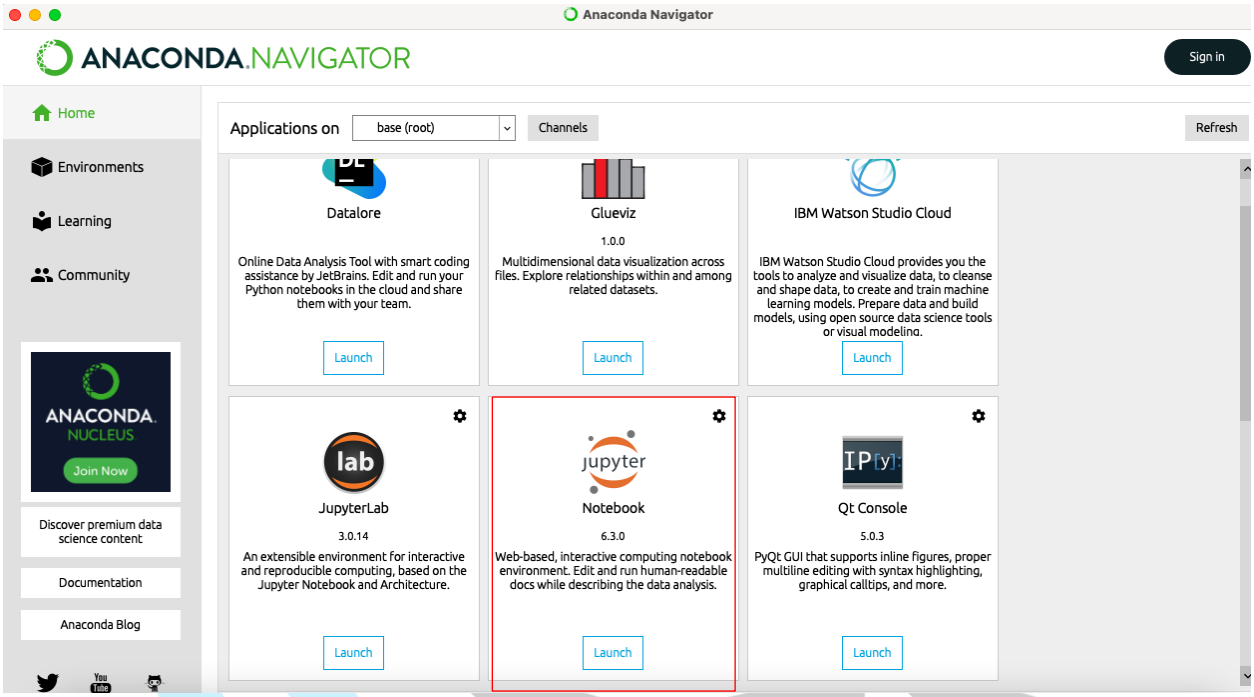
```
import saspy
sas_session = saspy.SASsession()
sas_session
```



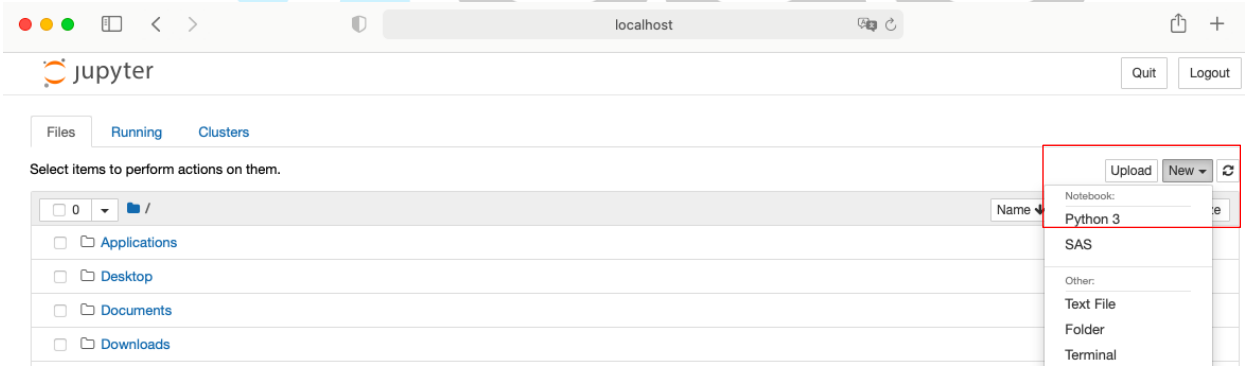
Note: If you encounter, **"*None of the requested encryption algorithms are supported by both peers: AES*"**, please confirm you are running the correct version of Java (1.8.0_162 or greater) and Contact Us for additional assistance.

Install SASPy on MAC

Download Anaconda here. Launch Jupyter Notebook from the Anaconda Navigator.

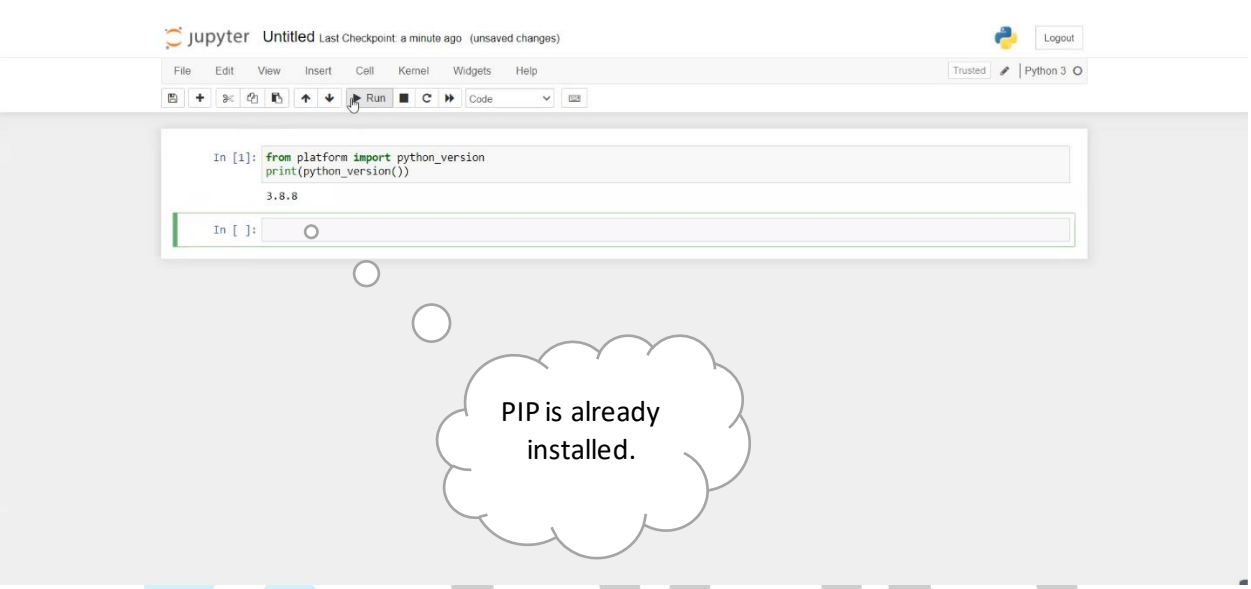


Create a new Python 3 Notebook in your Jupyter Notebook Web page.



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'))
```

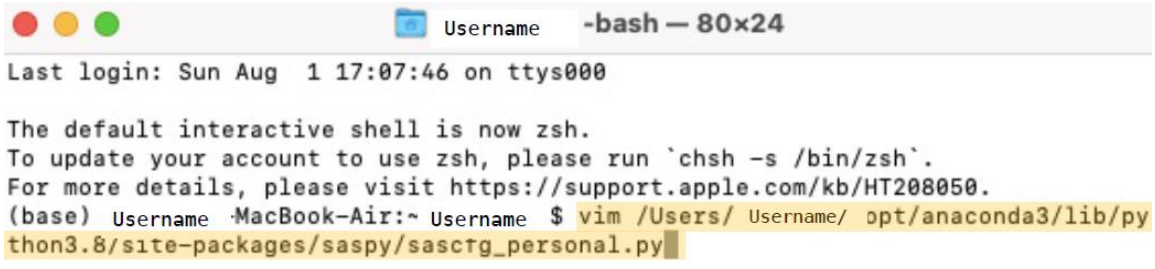
```
In [3]: import saspy, os
        print(saspy.__file__.replace('__init__.py', 'sascfg_personal.py'))

/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. <u>You will need to change ***FULLPATH*** to your unique path got from Step 1.</u>

   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



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.

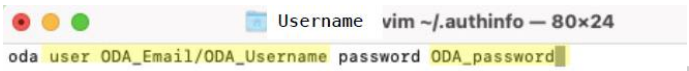
You will need to create 2 configuration files in total (***sascfg_personal.py*** **&** ***.authinfo***) and this step is for ***.authinfo*** file.

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

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
```



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
```

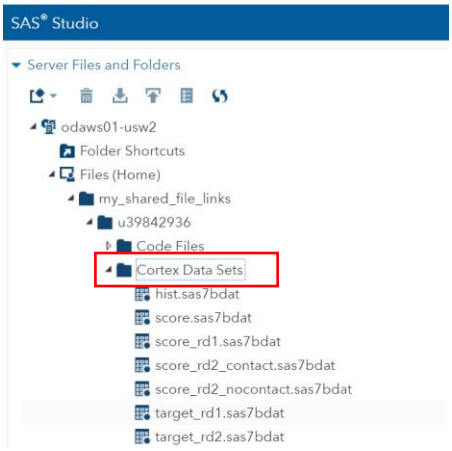
Note: If you encounter, "***None of the requested encryption algorithms are supported by both peers: AES***", please confirm you are running the correct version of Java (1.8.0_162 or greater) and Contact Us for additional assistance.

Access to Dataset

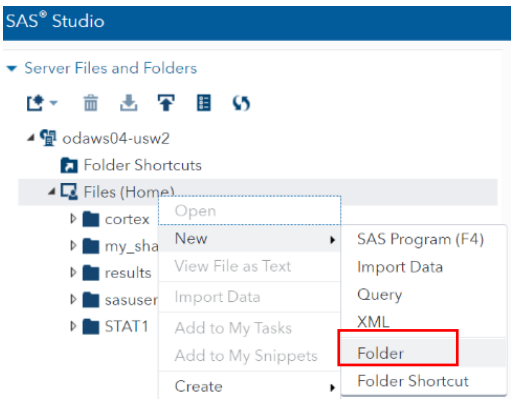
In order to access the dataset in the SAS terminal, you need to follow these steps:

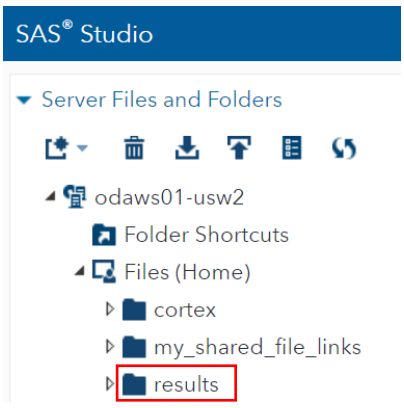
1. Sign on the Control Center at https://odamid.oda.sas.com.
2. Look for the Enroll in a course link in the "Enrollments" section near the bottom of the page. Click this link to start the enrollment.
3. Enter the course code: 83fd3afd-8964-4f5b-a612-7b665ed69104

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.

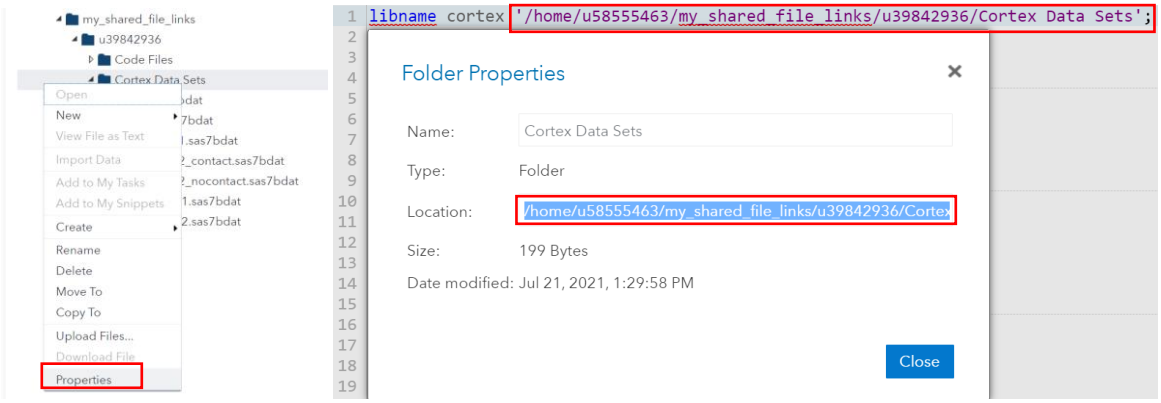


After that, you need to create a folder named “results”. This is important because you do not have the authority to do any edit/change to the datasets in that folder “u39842936”. Please restore any interim datasets to the 'work' or 'results' library.

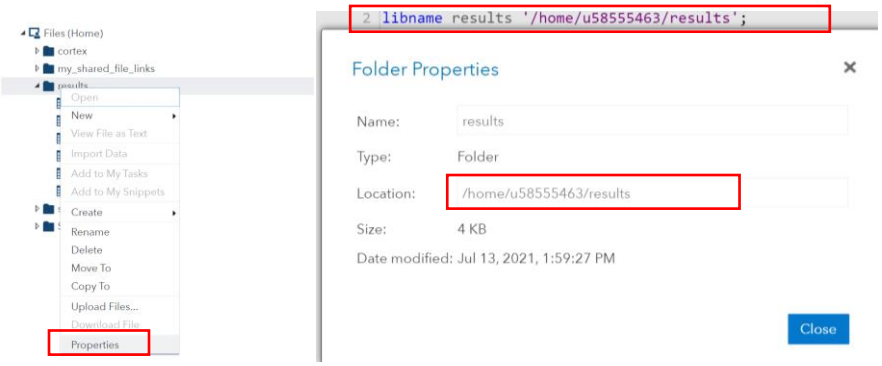




Copy the path of "u39842936" folder and create "cortex" library by code: libname cortex '/***path***';



Copy the path of "results" folder and create "results" library by code: libname cortex '/***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
```

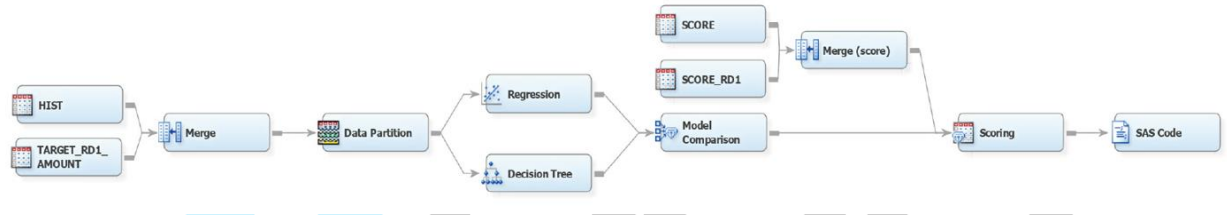
```
In [ ]:
%%SAS sas_session

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.



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_rd1 rate=0.6
out=  donor_select outall
method=srs
seed =1234;
run;

data results.train_rd1 results.test_rd1;
set donor_select;
if selected =1 then output results.train_rd1;
else output results.test_rd1;
run;
```

Link for Reference

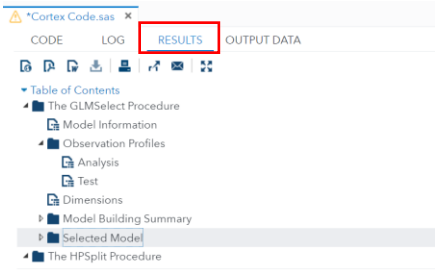
Step 3: Model Building

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_rd1 testdata=results.test_rd1;
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_rd1;
target AmtThisYear / level=interval;
input Age Salary / level=interval;
input GaveLastYear Seniority/level=nominal;
score data=results.test_rd1 role=valid OUT=_NULL_ OUTFIT=results.DT_stat_rd1;
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_rd1;
   MERGE cortex.score cortex.score_rd1;
   BY ID;
run;

data results.score_rd1; /*drop na*/
set score_rd1;
if not cmiss(of _numeric_);
run;


data results.result_rd1 (keep= id p_amtthisyear);
set results.score_rd1;
%include '/home/u58717790/Results/regression_1.sas';
run;
```

Step 6: Export Results

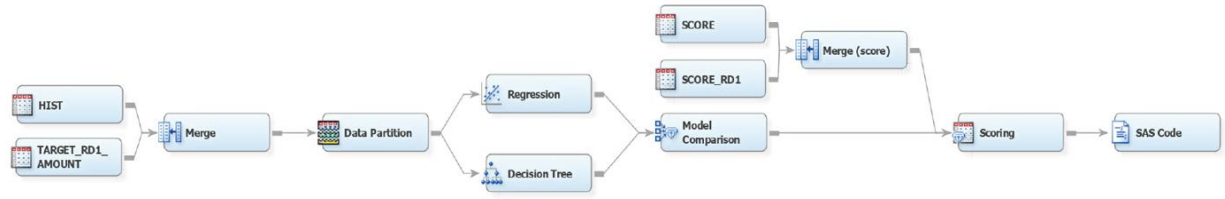
Following code will help you to export the model result into a CSV file. You can sort the data by predicted amount in the descending order before export the file, this will help you determine number of donors you want to contact.

```
/* Export data*/
PROC SORT DATA=results.result_rd1;
    BY descending p_amtthisyear;
RUN;

proc export data=results.output_rd1
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 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
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

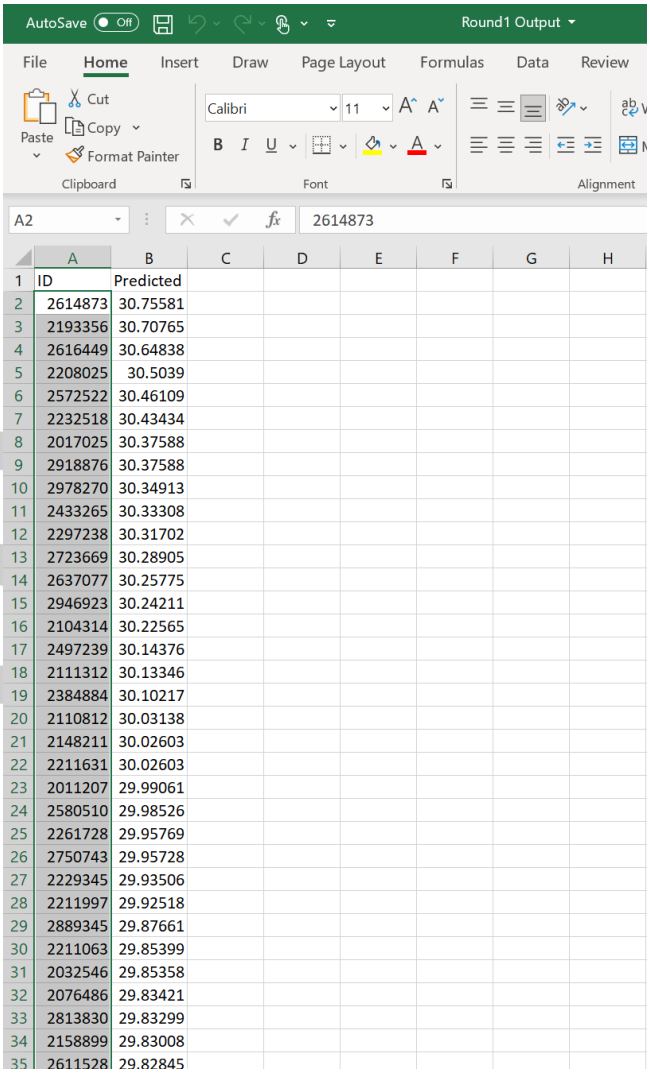
Following code will help you to export the model result into a CSV file. You can sort the data by predicted amount in the descending order before export the file, this will help you determine number of donors you want to contact.

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

Round 1 - Upload Decisions

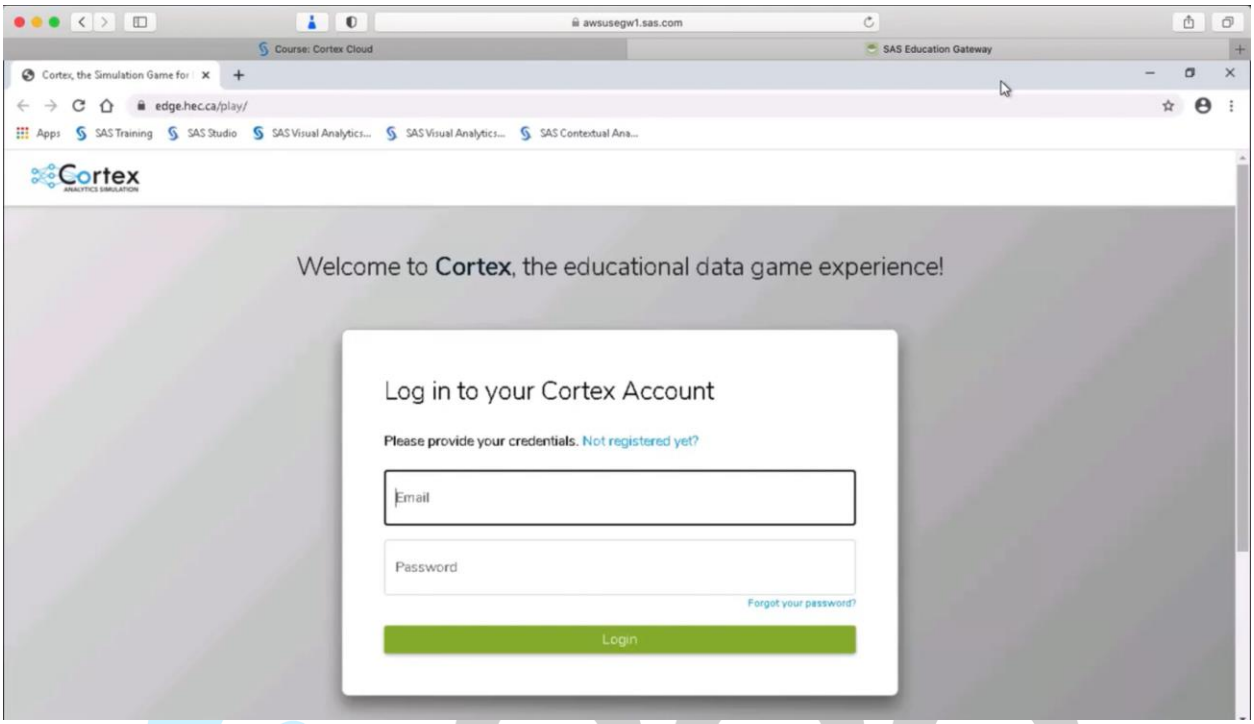
After you export the result, you need to create a new CSV file, select the number of people you want to contact in the file exported, and copy the ID list ONLY to the new CSV file. You need to **SAVE** and **CLOSE** the new CSV file before uploading it to the game leaderboard.

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

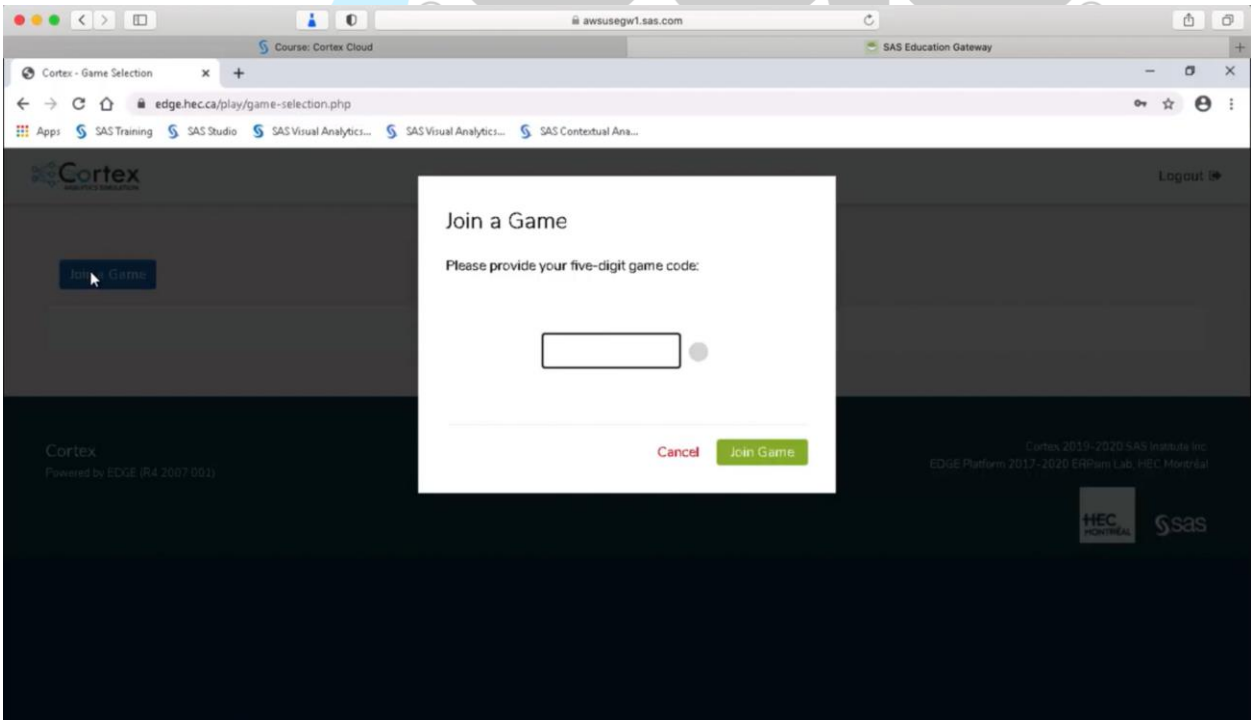


	A	B
1	ID	Predicted
2	2614873	30.75581
3	2193356	30.70765
4	2616449	30.64838
5	2208025	30.5039
6	2572522	30.46109
7	2232518	30.43434
8	2017025	30.37588
9	2918876	30.37588
10	2978270	30.34913
11	2433265	30.33308
12	2297238	30.31702
13	2723669	30.28905
14	2637077	30.25775
15	2946923	30.24211
16	2104314	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

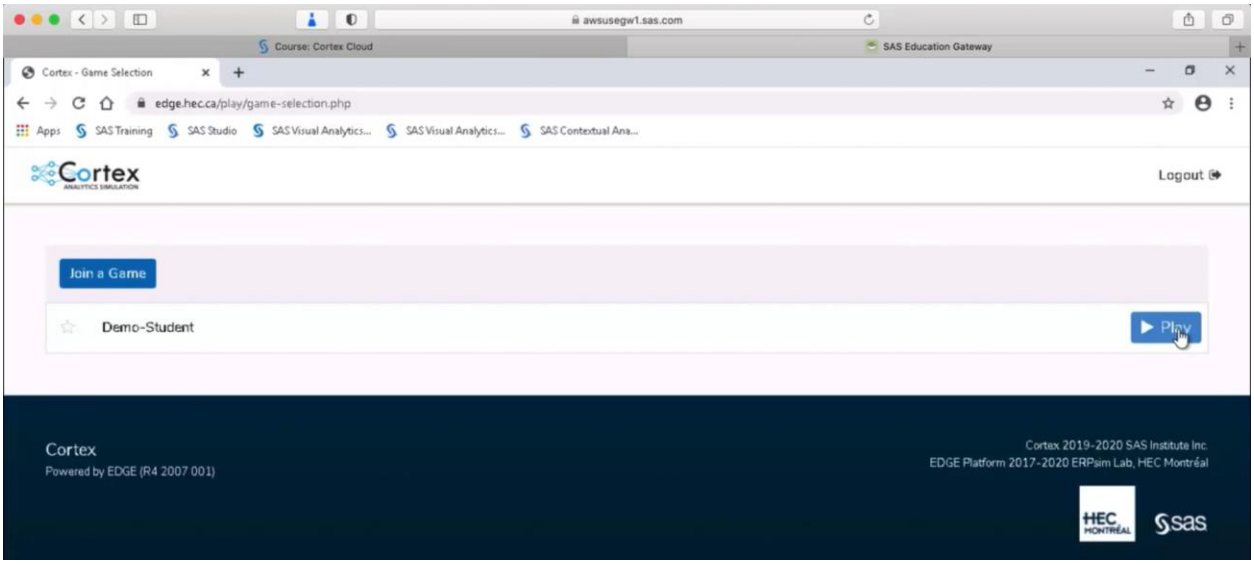
Then you need to log in to the Leaderboard: https://edge.hec.ca/play/



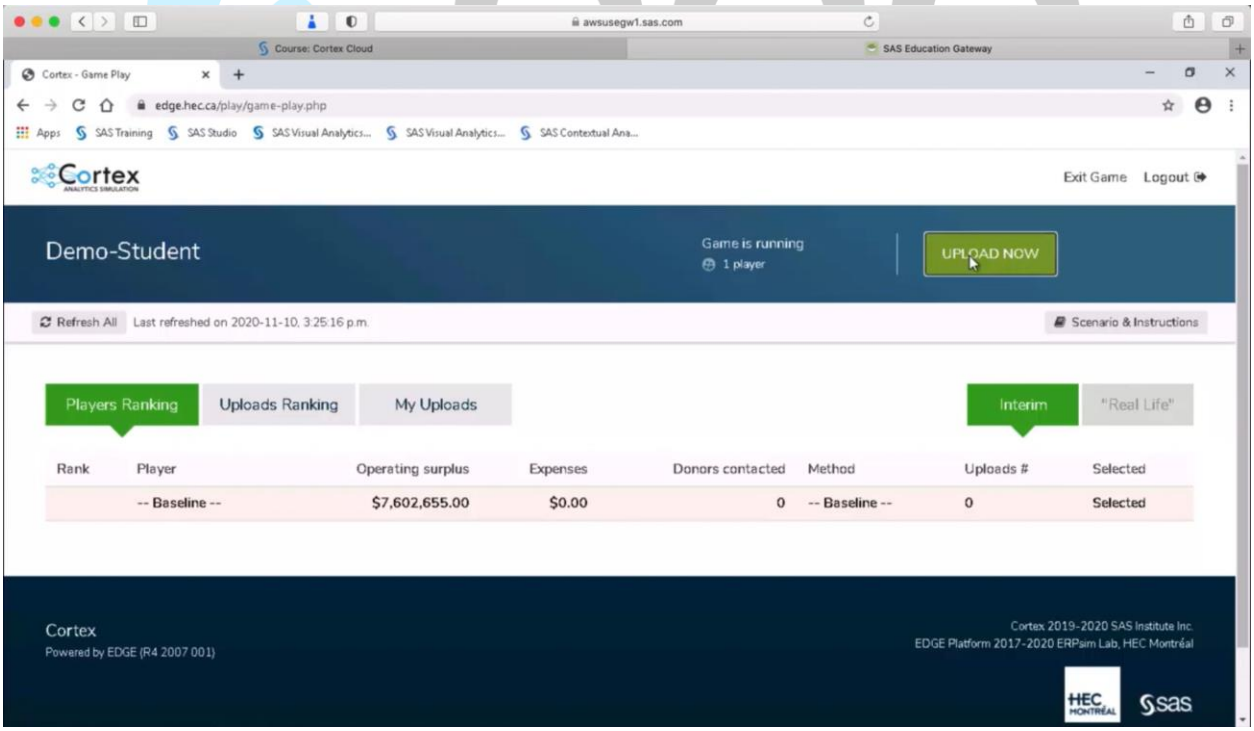
Click **Join a Game** button and enter the Five-Digit Game Code **NF02N** that you should have received from our email.



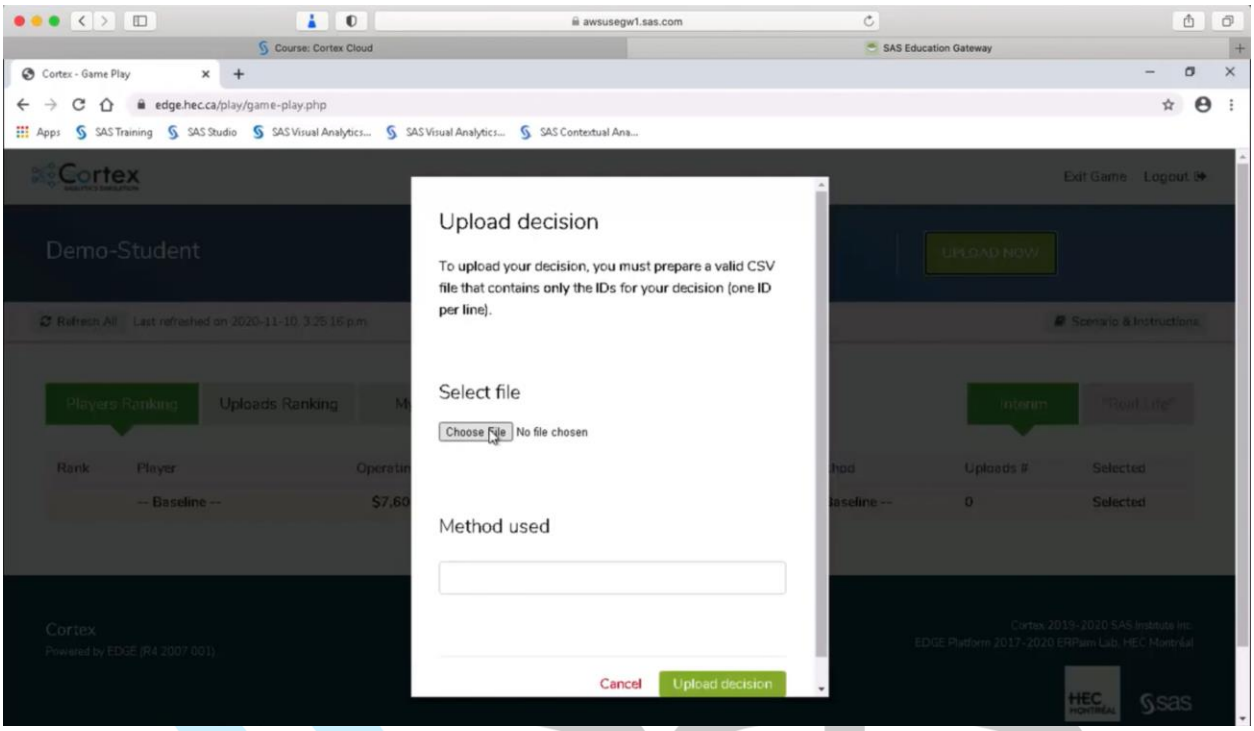
Click **Play** button to start the game.

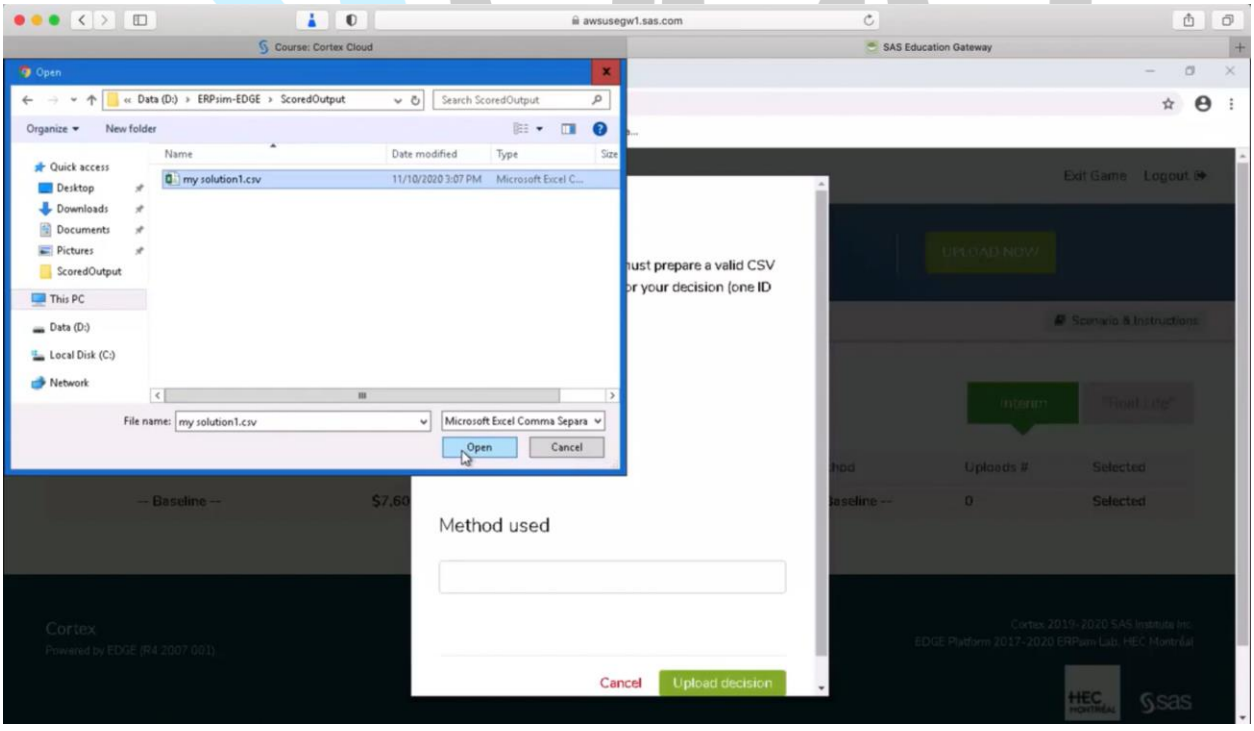


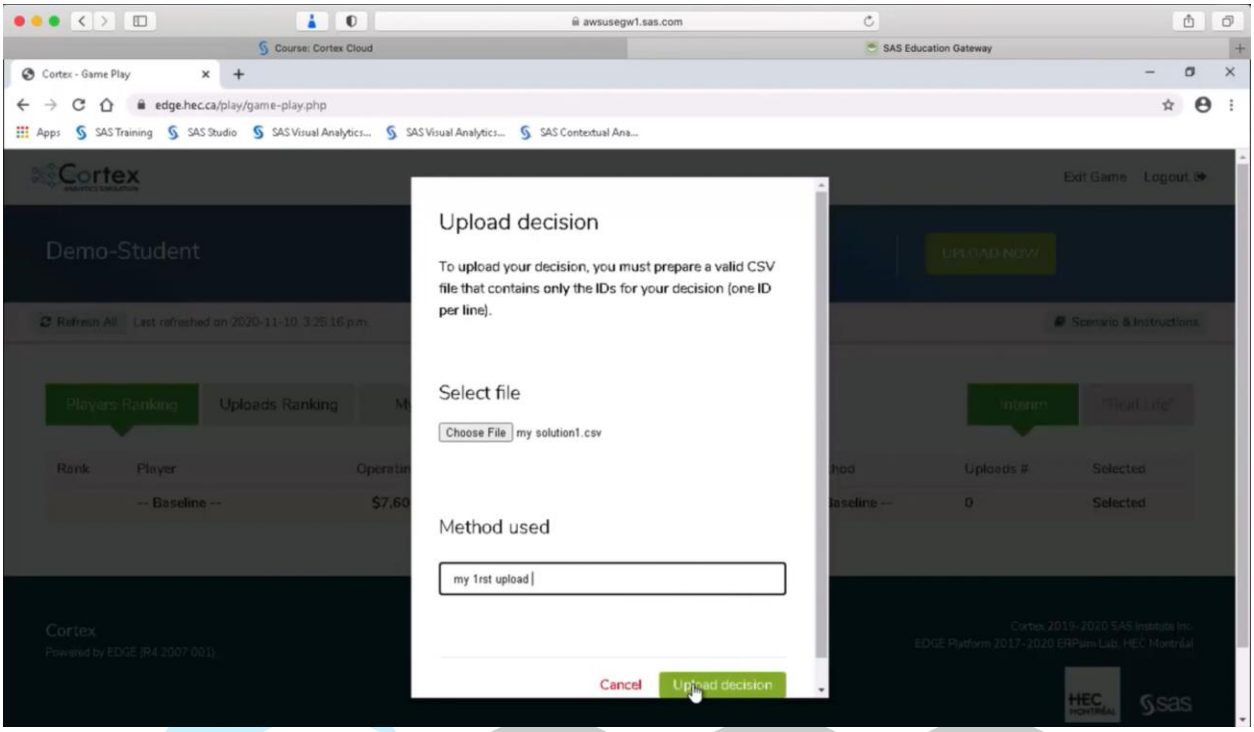
Click **UPLOAD NOW** button to upload CSV file.



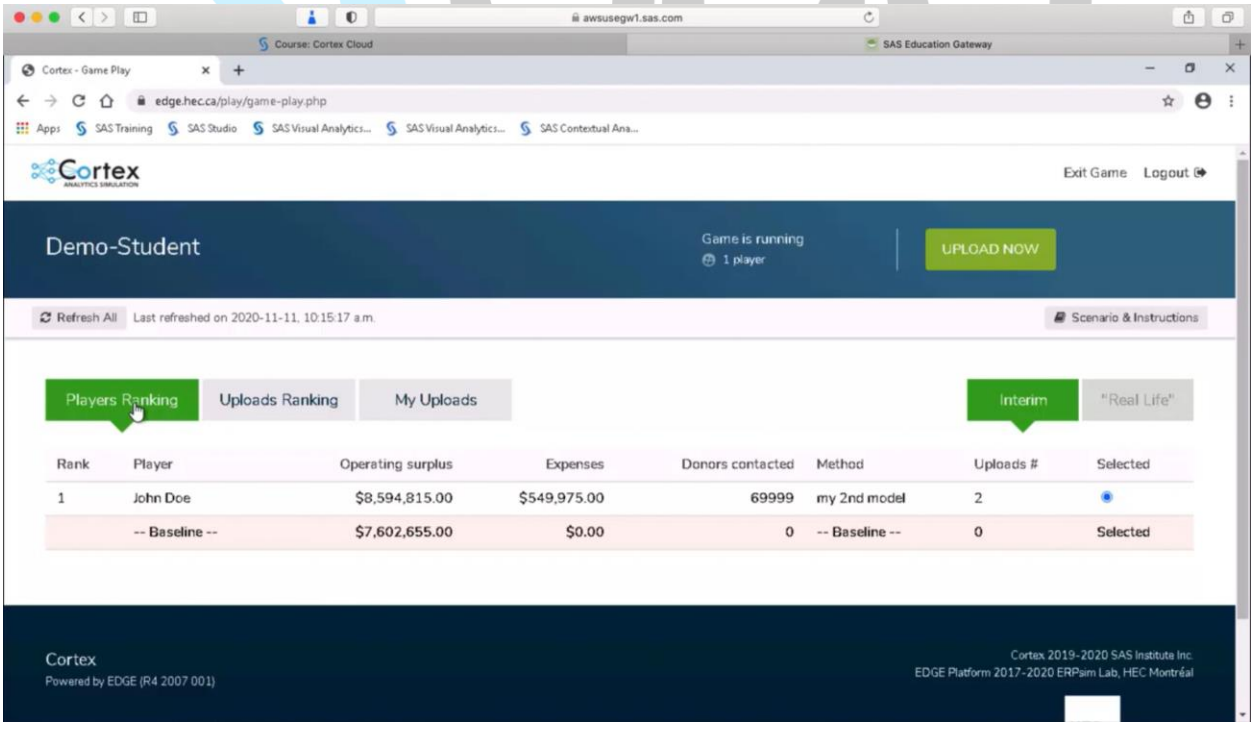
Click **Choose File** button to select your new CSV file, you can name your upload under the **Method used**, and click **Upload decision** button to complete your upload.



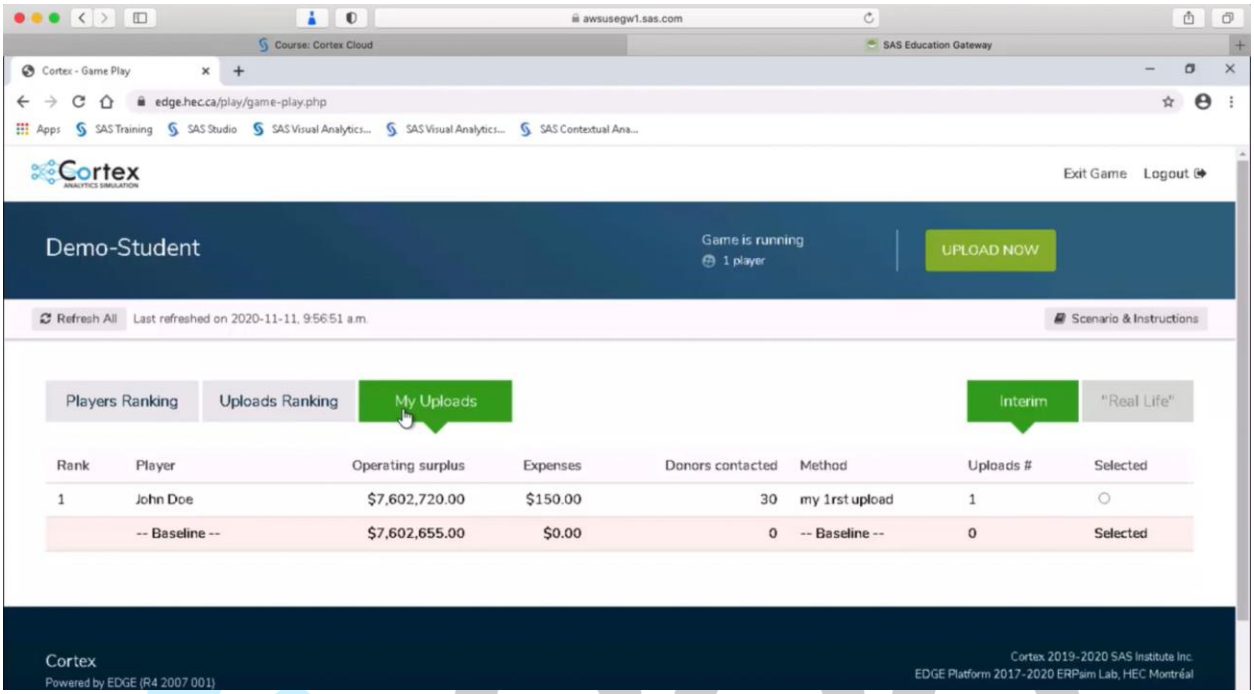




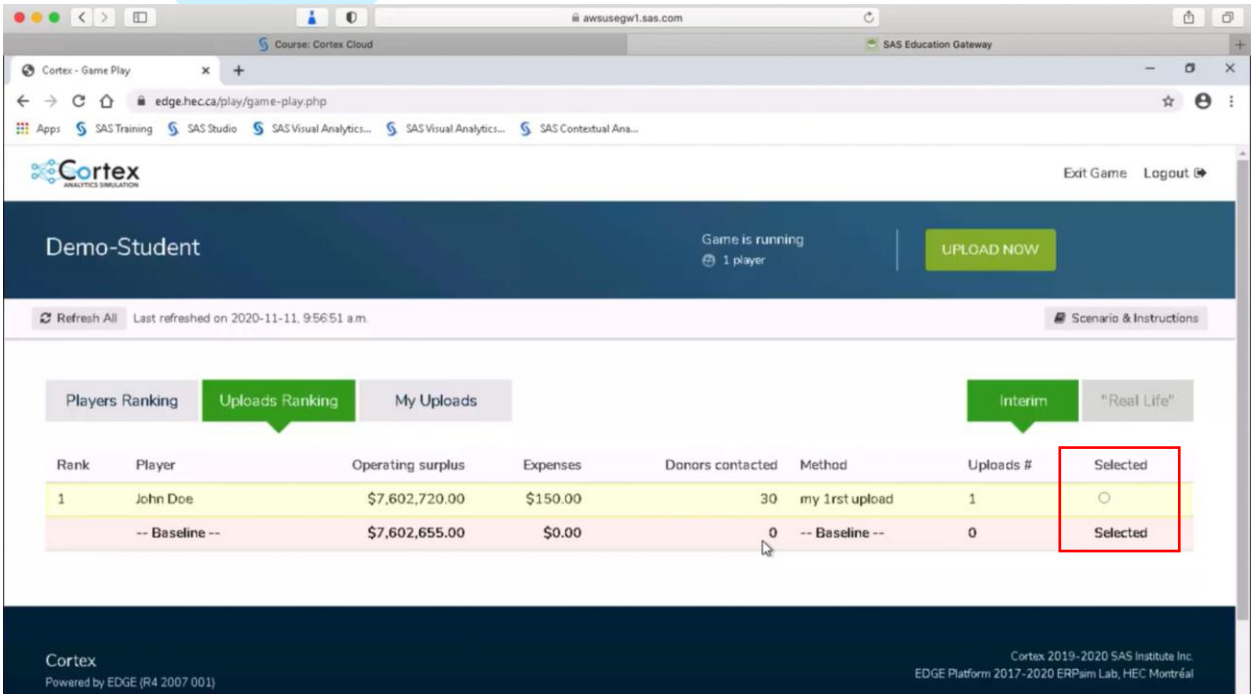
In the main leaderboard: **Players Ranking** tab, you can check the result of the model you choose to upload as well as results uploaded by other players.



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



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.



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:

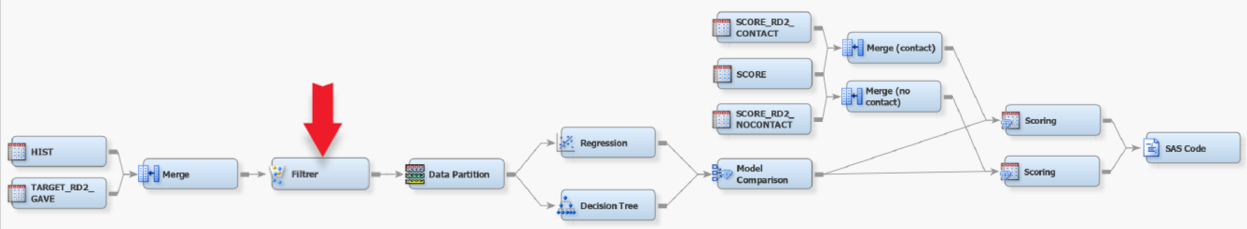
1. Fit a model to determine the ***probability P*** that an individual will give
2. Keep only the data of those who gave, fit a model for ***M (the amount gave)***
3. Use both models to make predictions on the population
4. Compute ***P*M*** to determine the ***'expected donation'*** of each individual

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.



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 surveyselect 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

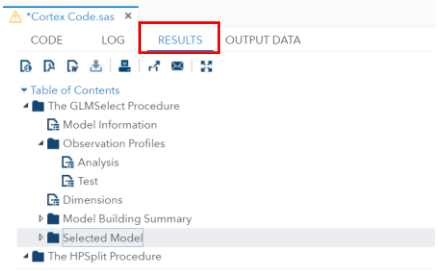
Step 3: Model Building

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 donation next year given the donor is contacted

```
/* 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 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

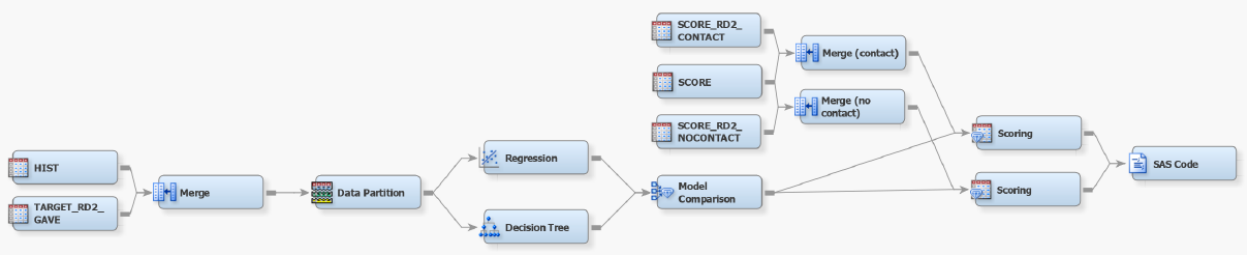
Step 6: Export Results

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)

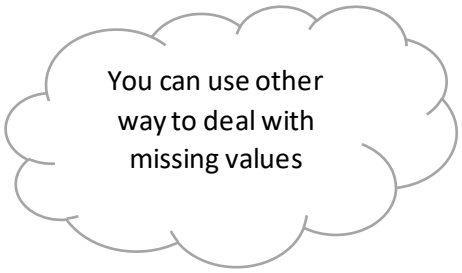
There are 6 steps for game completion: Data Merge, Data Partition, Model Building, Model Comparison, Scoring Data and Export Results.



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;
```



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

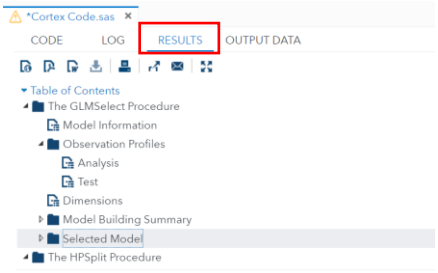
Step 3: Model Building

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.



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 givenextyear 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 probability of donate next year given the donor is contacted

```
/* predict givenextyear 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=PNoContact));
set results.nocontact_rd2;
%include '/home/u58717790/Results/regression_2_prob.sas';
run;
```

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

Step 6: Export Results

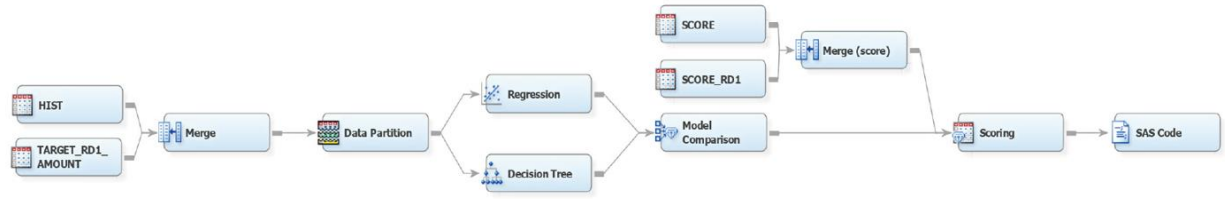
Merge the two different predictions (probability given contacted and probability given not contacted) by ID, and then export as CSV file.

```
/* 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

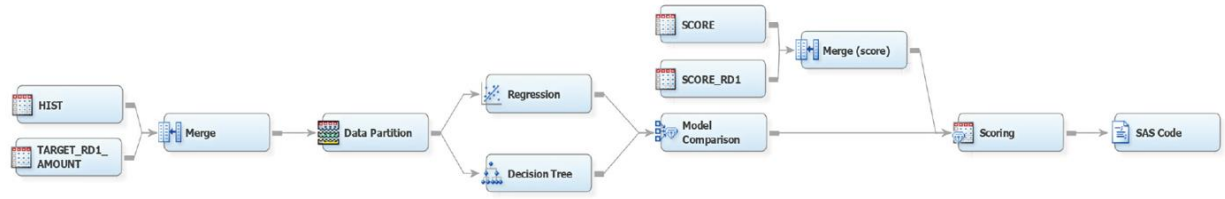
Following code will help you to export the model result into a CSV file. You can sort the data by predicted amount in the descending order before export the file, this will help you determine number of donors you want to contact.

```
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()
```

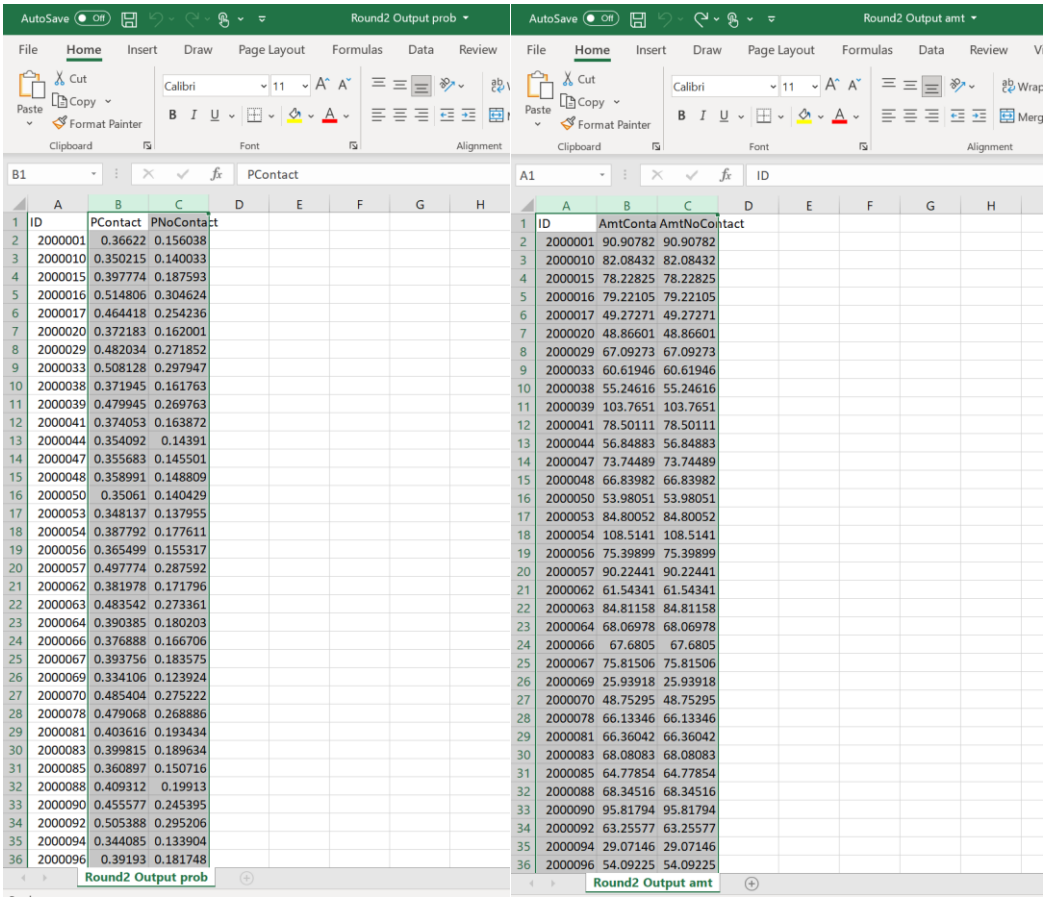
Step 6: Export Results

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

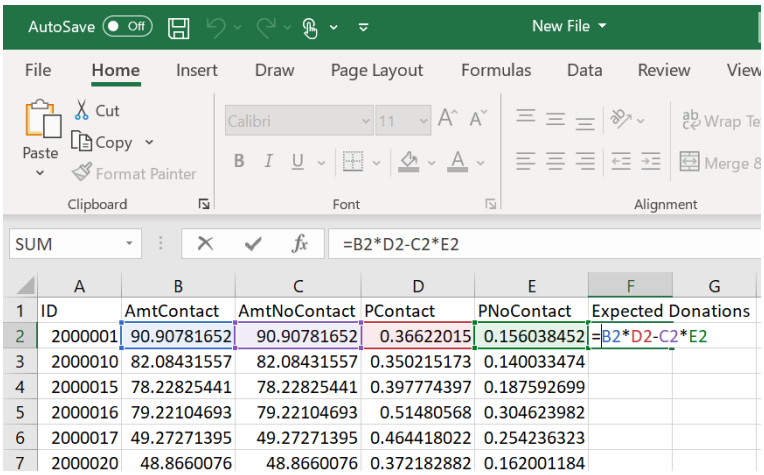
Copy and paste these five columns into a new CSV file.



ID	PContact	PNoContact
2000001	0.36622	0.156038
2000010	0.350215	0.140033
2000015	0.397774	0.187593
2000016	0.514806	0.304624
2000017	0.464418	0.254236
2000020	0.372183	0.162001
2000029	0.482034	0.271852
2000033	0.508128	0.297947
2000038	0.371945	0.161763
2000039	0.479945	0.269763
2000041	0.374053	0.163872
2000044	0.354092	0.14391
2000047	0.355683	0.145501
2000048	0.358991	0.148809
2000050	0.35061	0.140429
2000053	0.348137	0.137955
2000054	0.387792	0.177611
2000056	0.365499	0.155317
2000057	0.497774	0.287592
2000062	0.381978	0.171796
2000063	0.483542	0.273361
2000064	0.390385	0.180203
2000066	0.376888	0.166706
2000067	0.393756	0.183575
2000069	0.334106	0.123924
2000070	0.485404	0.275222
2000078	0.479068	0.268886
2000081	0.403616	0.193434
2000083	0.399815	0.189634
2000085	0.360897	0.150716
2000088	0.409312	0.19913
2000090	0.455577	0.245395
2000092	0.505388	0.295206
2000094	0.344085	0.133904
2000096	0.39193	0.181748

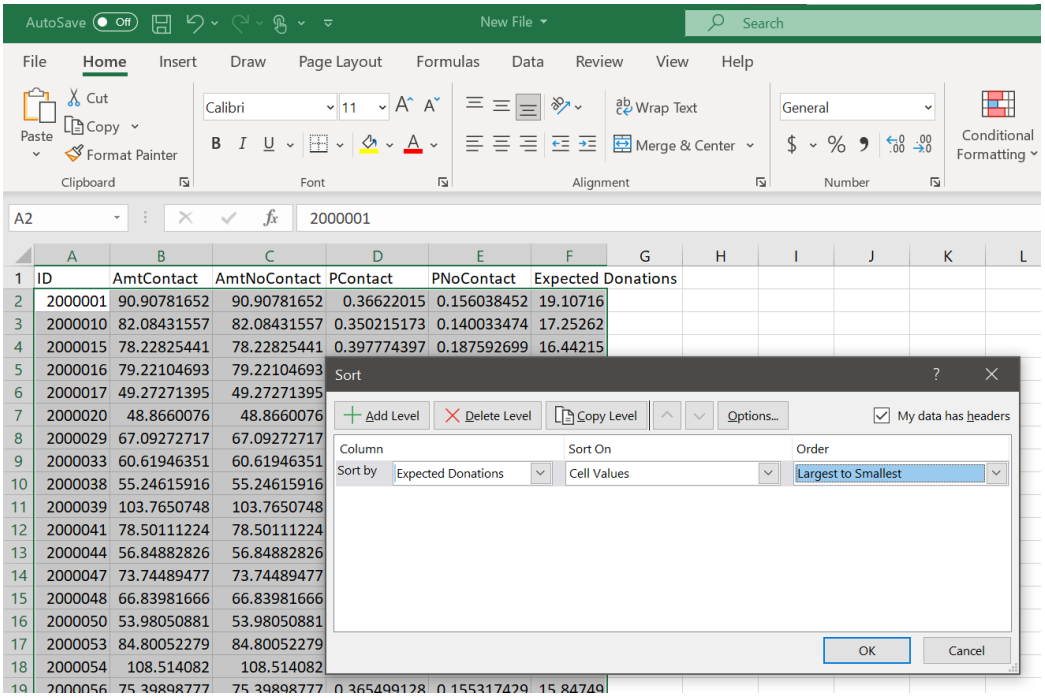
ID	AmtConta	AmtNoContact
2000001	90.90782	90.90782
2000010	82.08432	82.08432
2000015	78.22825	78.22825
2000016	79.22105	79.22105
2000017	49.27271	49.27271
2000020	48.86601	48.86601
2000029	67.09273	67.09273
2000033	60.61946	60.61946
2000038	55.24616	55.24616
2000039	103.7651	103.7651
2000041	78.50111	78.50111
2000044	56.84883	56.84883
2000047	73.74489	73.74489
2000048	66.83982	66.83982
2000050	53.98051	53.98051
2000053	84.80052	84.80052
2000054	108.5141	108.5141
2000056	75.39899	75.39899
2000057	90.22441	90.22441
2000062	61.54341	61.54341
2000063	84.81158	84.81158
2000064	68.06978	68.06978
2000066	67.6805	67.6805
2000067	75.81506	75.81506
2000069	25.93918	25.93918
2000070	48.75295	48.75295
2000078	66.13346	66.13346
2000081	66.36042	66.36042
2000083	68.08083	68.08083
2000085	64.77854	64.77854
2000088	68.34516	68.34516
2000090	95.81794	95.81794
2000092	63.25577	63.25577
2000094	29.07146	29.07146
2000096	54.09225	54.09225

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

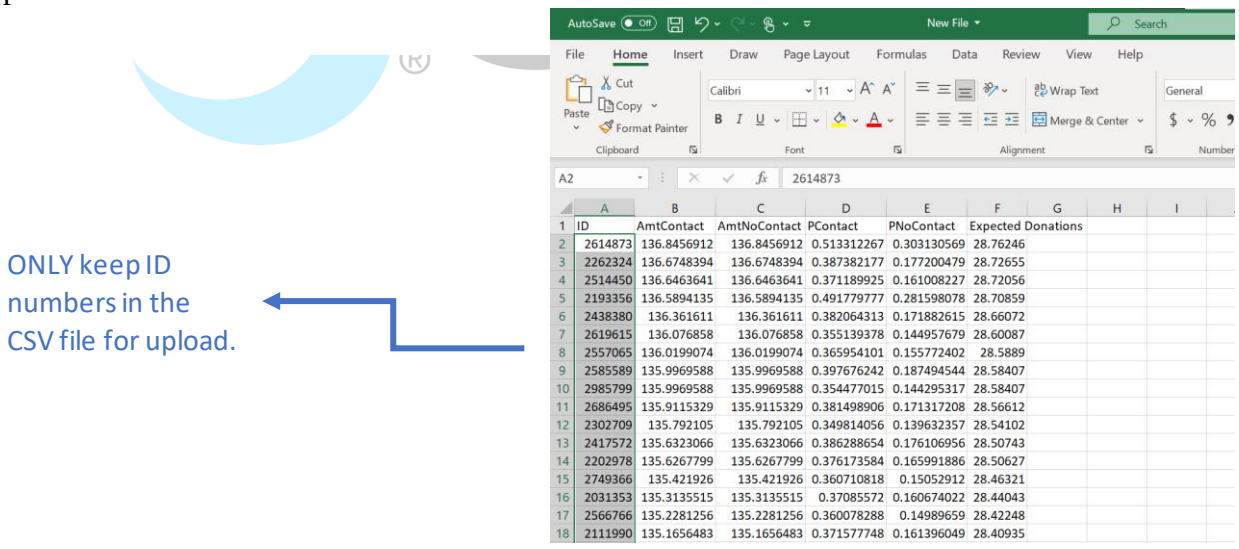


ID	AmtContact	AmtNoContact	PContact	PNoContact	Expected Donations
2000001	90.90781652	90.90781652	0.36622015	0.156038452	=B2*D2-C2*E2
2000010	82.08431557	82.08431557	0.350215173	0.140033474	
2000015	78.22825441	78.22825441	0.397774397	0.187592699	
2000016	79.22104693	79.22104693	0.51480568	0.304623982	
2000017	49.27271395	49.27271395	0.464418022	0.254236323	
2000020	48.8660076	48.8660076	0.372182882	0.162001184	

Sort all columns by descending amount of Expected Donations.



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



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