UN PET Lab Competition: Competition

Welcome! This is a short walkthrough to give you a step through the elements of the competition. Any action you perform with data libraries here will impact your ranking and score! Please use sandbox environment shared prior! Once you experiment and come up with a good strategy, then you can start here on the competition notebook and copy in any of the cells you think might be helpful.

Prerequisites: Proxy Installation and Quick Start

Step 1: Download installation and start scripts

We need these scripts to install proxy for connecting to the Hackathon API server running in Oblvious enclave. Two scripts oblv-install.sh and oblv-start.sh are downloaded in step 1.

```
!wget https://api.oblivious.ai/oblv-ccli/scripts/oblv-install.sh
!wget https://api.oblivious.ai/oblv-ccli/scripts/oblv-start.sh
!chmod +x ./oblv-install.sh
!chmod +x ./oblv-start.sh
```

```
--2022-11-11 04:27:33--  https://api.oblivious.ai/oblv-ccli/scripts/oblv-ins
Resolving api.oblivious.ai (api.oblivious.ai)... 3.14.98.130, 3.20.100.31
Connecting to api.oblivious.ai (api.oblivious.ai)|3.14.98.130|:443... connec
HTTP request sent, awaiting response... 200 OK
Length: 201 [text/x-sh]
Saving to: ‘oblv-install.sh.9’

oblv-install.sh.9   100%[===================>]     201  --.-KB/s    in 0s

2022-11-11 04:27:33 (50.3 MB/s) - ‘oblv-install.sh.9’ saved [201/201]

--2022-11-11 04:27:34--  https://api.oblivious.ai/oblv-ccli/scripts/oblv-sta
Resolving api.oblivious.ai (api.oblivious.ai)... 3.20.100.31, 3.14.98.130
Connecting to api.oblivious.ai (api.oblivious.ai)|3.20.100.31|:443... connec
HTTP request sent, awaiting response... 200 OK
Length: 207 [text/x-sh]
Saving to: ‘oblv-start.sh.9’

oblv-start.sh.9     100%[===================>]     207  --.-KB/s    in 0s
```

Step 2: Install the OBLV on Datalore

We've downloaded two scripts to this notebooks file system in the `/data/notebook_files/` folder. The first is `oblv-install.sh` which will help us to download and install the proxy on Linux, specifically `Ubuntu:22.04` which this notebook is running in. We will introduce the second script, `oblv-start.sh`, in the next subsection.

To run the installation script, first open a terminal. To do this, on the top panel on this window click `Tools >> Terminal`. Your screen will be split and you will have a terminal on the right-hand side. You will already be in the `/data/notebook_files/` so you can go ahead and execute the script:

```
./oblv-install.sh
```

That should have installed the `oblv` proxy. You can test this by running `oblv --help` and you should see the following output:

```
Configuration file stored at: "/home/user/.config/oblv/oblv_config.yaml"

oblv 0.4.0
Oblivious Software Ltd. <oblivious.ai>
Oblivious client app for encrypted connection to secure enclave

USAGE:
    oblv <SUBCOMMAND>

FLAGS:
    -h, --help       Prints help information
    -V, --version    Prints version information

SUBCOMMANDS:
    connect      Connect to enclave
    help         Prints this message or the help of the given subcommand(s)
    keygen       Generate public/private rsa key pair
    reconnect    Reconnect to a previously connected enclave
```

Step 3: Run the Proxy as a Background Task

Now that we've installed proxy, we can connect to the enclave. Now, to make a connection you both authenticate yourself **and** the attestation of the enclave. To do this, the `oblv` cli proxy needs access to your public/private key pair which you were emailed (or will be, depending on

needs access to your public/private key pair which you were emailed (or will be, depending on when you are reading this).

These keys should have been labelled `oblv_public.der` and `oblv_private.der` and the keys the contents will be unique for you. If you look at the sidebar on the left, and click the paper clip icon, you can upload those keys as files to the `/data/notebook_files/`. In principle, it doesn't matter what you call these files, but we've hardcoded `oblv_public.der` and `oblv_private.der` into the script `oblv-start.sh` for your convenience.

Note: The `oblv-start.sh` script is only a template for you to fill in based on the details you received in an email from the organisers. You will have to copy in the PCR codes (there are 3 of these) and the URL hosting the enclave. The script is a single command and there is no reason at all this needs to be a script and not run directly - however, in testing we found some people had issues pasting into the Datalore terminal in the web console, so we put it into a script to make life easier for you.

Once the keys are in place, go ahead and hit `./oblv-start.sh` in the terminal. This will run the proxy to listen to port `3031` and forward it securely end-to-end into the enclave. Encryption is performed on the fly, so you can just send requests to `localhost:3031` an interact with the enclave as if it is running in locally.

The Remote-Execution Client

Installation with Pip

Now that we've the proxy connection sorted, we can send traffic to the `localhost:3031` and it will get directed to the enclave with secure end-to-end encryption and attestation. But we actually want to use the DP libraries locally, and then serialize them into JSON to be sent as REST API calls. Having to write this yourself would just be clunky... so instead we provided a small client library which allows you to pass in the classes of the DP libraries and the serialization/deserialization and request handling will be taken care for you. Let's go ahead and install that in the next cell:

```
# to be replaced with normal pip install when ready
!pip install dp-serial -U
```

```
# also update pandas
!pip install pandas==1.5.1
```

```
Collecting dp-serial
  Downloading dp_serial-0.1.5-py3-none-any.whl (7.1 kB)
Collecting pandas==1.5.1
  Downloading pandas-1.5.1-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_
     |████████████████████████████████| 12.2 MB 35.5 MB/s
Collecting opendp==0.6.1
  Downloading opendp-0.6.1-py3-none-any.whl (19.4 MB)
     |████████████████████████████████| 19.4 MB 46.2 MB/s
Collecting numpy==1.22.3
  Downloading numpy-1.22.3-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_
     |████████████████████████████████| 16.8 MB 30.5 MB/s
Collecting requests==2.28.1
  Downloading requests-2.28.1-py3-none-any.whl (62 kB)
     |████████████████████████████████| 62 kB 3.1 MB/s
Collecting scikit-learn>=1.1.2
  Downloading scikit_learn-1.1.3-cp38-cp38-manylinux_2_17_x86_64.manylinux20
     |████████████████████████████████| 31.2 MB 29.9 MB/s
Collecting diffprivlib==0.6.0
  Downloading diffprivlib-0.6.0-py3-none-any.whl (174 kB)
     |████████████████████████████████| 174 kB 64.0 MB/s
```

Running the Client

Once the library has been installed, we can create a Client object to send and receive our requests with. This is agnostic of the enclaves (like it would work if you were sending the serialized models to a normal server too). So we simply parameterize the client with the REST API endpoint, so it knows what to speak to. In our case, this is `localhost:3031`:

```
#import required libs
from dp_serial.client.client import Client
import numpy
import pandas as pd

COMPETITION_URL = "http://localhost:3031"

competition_enclaves = Client(COMPETITION_URL)

# the column names of the remote dataset
COLNAMES = ['urban', 'hhsize', 'head_gender', 's2_q10_mktcentre', 's2_q19_floo
       's2_q20_wallmat', 's2_q21a_powergrid', 'ownasset_0', 'ownasset_1',
```

```
        'ownasset_2', 'ownasset_3', 'ownasset_4', 's3_q3a_attendchurch',
        's3_q7_visitother', 's53_q6_hunger', 's53_q6_staple',
        's5_q38_worryfood', 's5_q42a_meat', 's5_q42b_eggs',
        's5_q2_receivedfood', 's5_q4_rent', 's7_q2_sendgift',
        's9_q4_hospitalvisit', 's9_q11_medsoutstock', 's2_q15b_schoolchildren',
        's2_q15c_youngchildren', 's2_q21_noadults', 's2_q23_anyphone',
        's2_q24_numchild', 's2_q2_age', 's4_q12_num_ent', 's5_q1',
        's5_q39a_hungryadult', 's5_q39b_hungrychild', 's5_q3a_food',
        's5_q3b_personal', 's5_q3c_durables', 's5_q3d_services', 's5_q3e_comms',
        's5_q3f_housing', 's5_q3g_utilities', 's5_q3h_transport',
        's5_q3i_medical', 's5_q40a_skippedadult', 's5_q40b_skippedchild',
        's5_q41a_nofoodadult', 's5_q41b_nofoodchild', 'wave',
        'sum_s4_q19_earnings', 'sum_s4_q19_laidoff', 'sum_s4_q20_profits',
        'sum_s4_q20_salestrend', 'sum_s4_q20c_survive', 's7_q1_receivegifthh',
        's4_q1_agactivity', 'y_financial_actions', 'y_financial_help', 'y_retur

# get the train_x and test_x from the remote server
train_x = pd.read_csv("https://api.oblivious.ai/datasets/X_train.csv")
test_x = pd.read_csv("https://api.oblivious.ai/datasets/X_test.csv")
```

```
train_x.head()
```

	urban	hhsize	head_gender	s2_q10_mktcentre	s2_q19_floormat	s2_q20_wallmat	s2_q21a_powergrid	ownas
0	0	6	1	-1	3	16	0	0
1	0	6	1	-1	3	-1	0	0
2	0	2	0	0	2	16	1	0
3	0	2	0	0	2	-1	1	0
4	0	3	0	0	2	11	1	0

5 rows × 55 columns

```
test_x.head()
```

	urban	hhsize	head_gender	s2_q10_mktcentre	s2_q19_floormat	s2_q20_wallmat	s2_q21a_powergrid	ownas
0	0	0	0	-1	-1	16	1	1
1	1	1	0	1	2	14	1	0
2	1	6	0	1	2	7	1	0
3	0	5	0	0	2	7	1	0

4	0	7	0	-1	2	14	0	0

5 rows × 55 columns

Original Dataset

The original dataset of the hackathon is the panel study: "Socio-economic impact of COVID-19 on refugees" in Kenya from May 2020 to June 2022. It was conducted through a collaboration between the UNHCR, the World Bank, the Kenya National Bureau of Statistics and the University of California, Berkeley.

The dataset consists of eight waves distant by two to four months of Rapid Response Phone Survey with refugee households in Kenya. Its goal is originally to track the socioeconomic impacts of the COVID-19 pandemic and the recovery from it in order to inform a targeted response.

More details and explanations can be found on the UNHCR catalog here.

Challenge Dataset

The dataset available in the challenge has been curated to a smaller dataset of 58 columns and 11498 rows. Three of these columns are the differentially private columns (y) to query and predict on.

Data description

We describe here the resulting columns in x (the features) and y (the values to predict).

x features

Column name	Type	Question	Possible responses
hhsize	Categories	What is the number of people in the household	['1' < '2' < '3' < '4' ... '12' < '13' < '14' < '15+'] with 0 to 14 as ordinal values
head_gender	Categories	What is the gender of the household head	[0: 'Male' , 1: 'Female']
s2_q2_age	Numeric	What is the age of the household head	Float value
s2_q15b_schoolchildren	Numeric	What is the proportion of school-aged children (5-17	Float value

		years) in the household	
s2_q15c_youngchildren	Numeric	What is the proportion of young children (0-4 years) in the household	Float value
s2_q21_noadults	Numeric	How many other adults (18 and above) members are there in the household	Float value
s2_q24_numchild	Numeric	What is the proportion of children (0-17 years old) in the household	Float value
urban	Categories	Do you live an urban area or rural/refugee camp area	[0: 'Rural (or refugee camp)', 1: 'Urban']
s2_q10_mktcentre	Boolean	Is the current residence located within a town/trading center	1 or 0
s2_q19_floormat	Categories	What is the predominant floor material of the main dwelling unit	['Dung' < 'Vinyl Or Asphalt Strips' < 'Cement' < 'Earth/Sand' < 'Wood Planks/Shingles' < 'Palm/Bamboo' < 'Parquet Or Polished Wood' < 'Ceramic Tiles' < 'Carpet'] with 0 to 8 as ordinal values
s2_q20_wallmat	Categories	What is the predominant wall material of the main dwelling unit	['No Walls' < 'Mud/Cow Dung' < 'Cane/Palm /Trunks' < 'Grass/Reeds' < 'Cardboard' < 'Bamboo With Mud'< 'Stone With Mud'< 'Corrugated Iron Sheets' < 'Reused Wood' < 'Tent/Canvas' < 'Wood Planks/Shingles' < 'Stone With Lime/Cement' < 'Uncovered Adobe' < 'Covered Adobe' < 'Cement' < 'Cement Blocks'

			< 'Bricks' < 'Plywood'] with 0 to 17 as ordinal values
s2_q21a_powergrid	Boolean	Is the home physically connected to the electricity grid	1 or 0
s2_q23_anyphone	Numeric	In total, how many mobile phone numbers do all members of the household use	Integer value
ownasset_0	Boolean	Did the household own none of the following: Radio/Mattress/Charcoal /Refrigerator before March 2020	1 or 0
ownasset_1	Boolean	Did the household own a Radio before March 2020	1 or 0
ownasset_2	Boolean	Did the household own a Mattress before March 2020	1 or 0
ownasset_3	Boolean	Did the household own a Charcoal Jiko before March 2020	1 or 0
ownasset_4	Boolean	Did the household own a Refigerator before March 2020	1 or 0
s3_q5a_attendchurch	Categories	In the past 14 days, has any member of the household attended religious service personally	['Yes, both this week and last week', 'Yes, this week (in the past 7 days)', 'Yes, last week (between 14 to 7 days ago)', 'No'] with 0 to 3 as ordinal values
s3_q7_visitother	Boolean	In the past 14 days did anyone visit the household house, or did any household member visit another house	1 or 0

s5_q1	Numeric	How much, in total, did the household consume from its own agricultural or pastoral production in past 7 days	Float value in Ksh
s53_q6_hunger	Boolean	Did any adult of the household went hungry because of lack of food in the last 30 days	1 or 0
s53_q6_staple	Boolean	The household couldnt access to staple food in the last month at any time	1 or 0
s5_q38_worryfood	Boolean	Did the responder worry that the household would not have enough food in the last 30 days	1 or 0
s5_q39a_hungryadult	Numeric	What is the number of days where adults in the household have gone to bed hungry in the past 7 days	Float value
s5_q39b_hungrychild	Numeric	What is the number of days where children in the household have gone to bed hungry in the past 7 days	Float value
s5_q42a_meat	Boolean	Did any of the meals that the household ate yesterday include Meat or fish	1 or 0
s5_q42b_eggs	Boolean	Did any of the meals that the household ate yesterday include Eggs	1 or 0
s5_q40a_skippedadult	Numeric	What is the number of days where adults in the household skipped meals in the past 7 days	Float value

		in the past 7 days	
s5_q40b_skippedchild	Numeric	What is the number of days where children in the household skipped meals in the past 7 days	Float value
s5_q41a_nofoodadult	Numeric	What is the number of days where adults in the household have gone entire day without food in the past 7 days	Float value
s5_q41b_nofoodchild	Numeric	What is the number of days where children in the household have gone entire day without food in the past 7 days	Float value
s4_q1_agactivity	Boolean)	In the past 14 days, have any household members performed any agricultural or pastoral activities	1 or 0
s4_q12_num_ent	Numeric	In the last months, other than farming, what is the number of non-agricultural self-employed enterprises run by members of household	Float value
sum_s4_q19_earnings	Numeric	What is the total earnings/revenues of all self-employed entreprises of the household since the last survey	Float value
sum_s4_q19_laidoff	Numeric	What is the total number of employees laid off in all self-employed entreprises of the household since the last survey	Float value
		What is the total sum profit of all self-employed	

sum_s4_q20_profits	Numeric	entreprises of the household since the last survey	Float value
sum_s4_q20_salestrend	Numeric	What is the total earnings / revenue of all self-employed entreprises of the household in a typical current 2-weeks period	Float value
sum_s4_q20c_survive	Numeric	At the current scale of operations, what is the number of weeks the household would survive without selling additional assets or getting additional assistance / loans to continue business operations (aggregated for all self-employed entreprises of the household)	Float value
s5_q3a_food	Numeric	What is the total household expenditure in Groceries / Food: Includes all meat, fish, eggs, dairy, oils, fats, vegetables, fruit, sugar, products and drinks	Float value in Ksh
s5_q3b_personal	Numeric	What is the total household expenditure in household and personal items: Includes soap, cleaning agents, toilet paper/tissues, air freshener, shoe polish, insecticide, matches, candles, toiletries, cosmetics	Float value in Ksh
		What is the total household expenditure in Assets / Durables : Includes vehicles	

s5_q3c_durables	Numeric	(car, boat, bike, motorbike, handcarts, etc.), furniture (bed, chair, vases, mirror, etc.), kitchen and other equipment (cutlery, pots, pans, plates, etc.), electronic equipment, (lamps, mobile phone, television, etc.), tools and livestock	Float value in Ksh
s5_q3d_services	Numeric	What is the total household expenditure in Local Services: Includes maize grinding, haircuts, prepared meals eaten outside the home, bicycle repair, recreation, etc	Float value in Ksh
s5_q3e_comms	Numeric	What is the total household expenditure in Communication: Includes airtime, internet and other phone expenses	Float value in Ksh
s5_q3f_housing	Numeric	What is the total household expenditure in Housing: Includes rent, mortgage, home maintenance, and repairs	Float value in Ksh
s5_q3g_utilities	Numeric	What is the total household expenditure in Energy and Utilities: Includes electricity, water, firewood, charcoal, kerosene	Float value in Ksh
s5_q3h_transport	Numeric	What is the total household expenditure in Transport: Includes petrol, tolls, taxi/bus/matatu/boda/piki /train/flight fare, hotels stays (NOT including	Float value in Ksh

		medical reasons)	
s5_q3i_medical	Numeric	What is the total medical expenses: Includes consultation fees, medicines, hospital costs, lab test costs, ambulance costs, and related transport (in Ksh)	Float value in Ksh
s7_q1_receivegifthh	Boolean	In the past 14 days, did anyone in the household receive a gift / assistance of money or goods from someone outside the household?	1 or 0
s7_q2_sendgift	Boolean	In the past 14 days, did anyone in the household give or send money or goods to someone outside the household?	1 or 0
s9_q11_medsoutstock	Categories	In the last week, has anyone in the household been unable to buy medicine?	['No, get it for free' < 'No' < 'Yes'] with 0 to 2 as ordinal values
wave	Numeric	Panel wave (id the the questionary)	Integer value
s5_q2_receivedfood	Boolean	Did you receive food as a gift in past 7 days	1 or 0
s5_q4_rent	Boolean	Do you rent your accommodation?	1 or 0
s9_q4_hospitalvisit	Boolean	In the past 30 days, did any member visit hospital for any reason?	1 or 0

For boolean columns, 1 is for 'Yes' and 0 is for 'No'.

NaNs values were replaced by -1.

y (train_y: differentially private, test_y: to predict)

Column name	Type	Question	Possible responses
y_financial_actions	Boolean	In the past 14 days, did anyone in the household take out a new loan to use on or sell any livestock or other household assets to generate income [Vehicles, Furniture, kitchen or electronic equipment, tools] ?	1 or 0
y_financial_help	Boolean	In the past 14 days, did anyone in this household receive a gift / assistance of money or goods from a government program, a non-governmental organization/community group or an individual politician/government official?	1 or 0
y_return	Boolean	Does the household not plan to return to their home country in the foreseeable future ?	1 or 0

1 is for 'Yes' and 0 is for 'No'.

Test-Train split

The test set is 20% of stratified sampling from the 8 waves of the questionnaire. This means that in each wave, 20% of the rows were randomly sampled to be in the test set (test_x and test_y) and the 80% remaining rows are in the train set (train_x, train_y).

As a participant, you have full access to train_x and use differential privacy to understand and modelise its relation with train_y. Based on your model, you then use test_x to predict test_y.

Using the client with SmartNoise SQL

This is probably the simplest interaction you can do with the client. Just call the below function with the SQL query you wish to execute and the epsilon and delta budget per step of execution.

There are 2 steps to executing an SQL query which you should always do. SQL queries may take multiple steps and you may spend more privacy budget than you intended to. So the first step is free and simply returns the epsilon and delta that *would be* applied if you decide to execute it for real. The second then executes the call for you.

The table name you are selecting from is always `comp.comp` and the column names will be as they are documented in the competition notebook. We'll just use the names `col1` and `col2` in the sandbox for demonstration purposes.

```
#Getting SQL query epsilon, delta estimate from API Server
estimate = competition_enclaves.sql_privacy_estimate("SELECT COUNT(y_return) as

print(estimate)
```

```
[1.0,0.00014999500000001387]
```

Now that you know what the [epsilon, delta] cost would be, you can choose to get the result of the query:

```
train_x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9199 entries, 0 to 9198
Data columns (total 55 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   urban                   9199 non-null   int64
 1   hhsize                  9199 non-null   int64
 2   head_gender             9199 non-null   int64
 3   s2_q10_mktcentre        9199 non-null   int64
 4   s2_q19_floormat         9199 non-null   int64
 5   s2_q20_wallmat          9199 non-null   int64
 6   s2_q21a_powergrid       9199 non-null   int64
 7   ownasset_0              9199 non-null   int64
 8   ownasset_1              9199 non-null   int64
 9   ownasset_2              9199 non-null   int64
 10  ownasset_3              9199 non-null   int64
 11  ownasset_4              9199 non-null   int64
 12  s3_q5a_attendchurch     9199 non-null   int64
 13  s3_q7_visitother        9199 non-null   int64
 14  s53_q6_hunger           9199 non-null   int64
```

```
test_x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2299 entries, 0 to 2298
Data columns (total 55 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   urban                   2299 non-null   int64
 1   hhsize                  2299 non-null   int64
 2   head_gender             2299 non-null   int64
 3   s2_q10_mktcentre        2299 non-null   int64
 4   s2_q19_floormat         2299 non-null   int64
 5   s2_q20_wallmat          2299 non-null   int64
 6   s2_q21a_powergrid       2299 non-null   int64
 7   ownasset_0              2299 non-null   int64
 8   ownasset_1              2299 non-null   int64
 9   ownasset_2              2299 non-null   int64
 10  ownasset_3              2299 non-null   int64
 11  ownasset_4              2299 non-null   int64
 12  s3_q5a_attendchurch     2299 non-null   int64
 13  s3_q7_visitother        2299 non-null   int64
 14  s53_q6_hunger           2299 non-null   int64
```

```
#Running SQL query using Smartnoise SQL

# only run _if_ you are sure you want to spend the budget to get the answer
query_result = competition_enclaves.sql("SELECT COUNT(y_financial_help) as fin_

print(query_result)
```

```
   fin_help
0      7001
1      2197
```

Hurray! We just learned something about the sensitive data via an SQL command (assuming col1 was public and col2 was sensitive, for example).

Seems simple enough, right? The art of the competition, if you were to use the DP-SQL interface, is working out what questions to ask and how much budget you should spend on them - but mastery of that is left up to you.

Using the client with SmartNoise Synth

The next, similarly straight forward, interface is that of SmartNoise Synth for synthetic data. To use this, you need to specify what model you would like to use and the privacy budget you

would like to apply. There are some advanced additional data you can add, but we'll get to that shortly.

⚠ **Warning:** Using high epsilon and delta values may significantly increase the time taken to generate date with SmartNoise Synth In some cases time may exceed 10 minutes!

Use params select_cols, and multiply_mat to select specific columns/transform and reduce time taken

```
#Generating Data with MST Synthesizer
cols_to_select = ["head_gender", "s2_q21a_powergrid", "y_return"]
mat = numpy.array([[0.001,0.1,0.001], [0.01,0.1,0.02], [0.41,0.1,0.3]])

# only run _if_ you are sure you want to spend the budget to get the answer
mwem_synthetic_data = competition_enclaves.synth("DPCTGAN", 1, 0.0001, select_c

print(mwem_synthetic_data)
```

```
             0         1         2
0     0.000023  0.009803  0.606196
1     0.000001  0.000452  0.412466
2     0.000001  0.001847  0.374232
3     0.000012  0.023726  0.090268
4     0.000002  0.001296  0.668312
...        ...       ...       ...
9995  0.000036  0.008575  0.318388
9996  0.000003  0.003247  0.379928
9997  0.000037  0.003859  0.718180
9998  0.000016  0.003107  0.189361
9999  0.000008  0.002826  0.918117

[10000 rows x 3 columns]
```

Of course, you do not need to use `MWEM` as your preferred synthetic data. You can choose from `[MWEM, DPCTGAN, MST, PATECTGAN]` and then the second parameter is the epsilon you would like to spend and the second is the delta, as was the case for the SQL queries.

You can also request to only use a subset of the columns so that you are not spreading your privacy budget over columns which are not important in your eyes (DP synthetic data in very high dimensions would typically require a lot of epsilon to be accurate). To do this, simply pass in the column names you wish to synthesize like this:

```
#Generating Data with PATCTGAN Synthesizer

#Matrix to multiply data with
mat = numpy.array([[0.001,0.1,0.001]])

# only run _if_ you are sure you want to spend the budget to get the answer
patectgan_synthetic_data = competition_enclaves.synth("PATECTGAN", 1, 0.00001,

print(patectgan_synthetic_data)

# now use your newly generated synthetic data anyway you like!
```

```
             0
0     1.052215
1     1.182738
2    -0.119510
3     1.318730
4    -0.333501
...        ...
9995  0.860115
9996  0.331400
9997  0.571410
9998  1.456395
9999  0.589440

[10000 rows x 1 columns]
```

⚠ **Warning:** Some synthetic methods are slow, but it totally depends on how many columns you select, the synthetic data model used, etc. If for any reason there is a timeout, your score should be unaffected as all queries are performed with exception handling.

Using the client with `DiffPrivLib`

For `DiffPrivLib`, on the author's advice, we've restricted functionality to executing pipelines which may hold one or many models to be applied sequentially. These follow the `DiffPrvLib` docs exactly, but before applying your pipeline on the data, you pass it to the client and it will be remotely executed for you and will return the trained model for you. Let us see this in action:

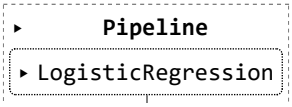
```
from sklearn.pipeline import Pipeline
from diffprivlib import models

#Diffprivlib LR Pipeline
lr_pipe = Pipeline([
    ('lr', models.LogisticRegression(data_norm=50))
])

# train the model and get the resulting trained model

# only run _if_ you are sure you want to spend the budget to get the answer
trained_model = competition_enclaves.diffprivlib(lr_pipe, y_column="y_financia
print(trained_model)
```

```
Pipeline(steps=[('lr',
                 LogisticRegression(accountant=BudgetAccountant(spent_budget
                                    data_norm=50))])
```

```
from sklearn.pipeline import Pipeline
from diffprivlib import models

#Diffprivlib LR Pipeline
lr_pipe = Pipeline([
    ('lr', models.LogisticRegression(data_norm=50))
])

# train the model and get the resulting trained model

# only run _if_ you are sure you want to spend the budget to get the answer
trained_modela = competition_enclaves.diffprivlib(lr_pipe, y_column="y_return"
print(trained_modela)
```

```
Pipeline(steps=[('lr',
                 LogisticRegression(accountant=BudgetAccountant(spent_budget
                                    data_norm=50))])
```

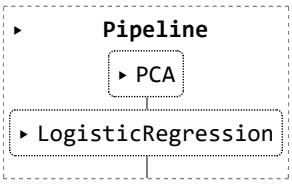
```
trained_modela
```



Equally, we can do a more complicated pipeline involving scaling, pca and training a logistic regression model:

```
# #Preparing Diffprivlib SPLR Pipeline
# splr_pipe = Pipeline([
#     #('scaler', models.StandardScaler(epsilon=0.001, bounds=([0, 0, 0, 0, 0,
#                                             #   [1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
#     # you might not need a scalar here, it's just an example
#     ('pca', models.PCA(2, epsilon=0.001, data_norm=00, centered=True)),
#     ('lr', models.LogisticRegression(data_norm=50, epsilon=0.001))
# ])

# # train the model and get the resulting trained model

# # only run _if_ you are sure you want to spend the budget to get the answer
# trained_model = competition_enclaves.diffprivlib(splr_pipe, y_column="y_retu


# # NOTE: Please keep epsilon values small in case the pipeline training takes
```

```
trained_model
```




```
from sklearn.pipeline import Pipeline
from diffprivlib import models

#Diffprivlib LR Pipeline
lr_pipe = Pipeline([
    ('lr', models.LogisticRegression(data_norm=50))
])
```

```
# train the model and get the resulting trained model

# only run _if_ you are sure you want to spend the budget to get the answer
trained_modelb = competition_enclaves.diffprivlib(lr_pipe, y_column="y_financi

print(trained_modelb)
```

```
Pipeline(steps=[('lr',
                 LogisticRegression(accountant=BudgetAccountant(spent_budget
                                    data_norm=50))])
```

All of the accepted `DiffPrivLib` pipelines are accepted so check out the docs directly. Any model submitted not from `DiffPrivLib` will be rejected.

Using the client with OpenDP

The last framework available is OpenDP. This is in one respect the most flexible, but from another respect the most complicated for someone who is starting out for the first time.

In OpenDP, we create pipelines of transformations (like clipping values, selecting columns, etc) and measurements (calculations on the transformed dataset). We typically use the right shift operator, `>>`, to concatenate these transformations and measurements together from end-to-end.

The measurement is the part of the pipeline which applies differential privacy. So if there is no measurement as part of the pipeline, then the remote execution will be rejected. Also, same as the other methods, if the epsilon or delta far exceeds the capped epsilon/delta permitted per query, it will also fail.

🛑 **Important:** When using `OpenDP` you must import transformers and measurements as `import dp_serial.opendp_logger.trans as trans` and `import dp_serial.opendp_logger.meas as meas` respectively. OpenDP does not natively keep a track of the abstract syntax tree (`AST`) of the pipeline and hence it would not be parsable. This `opendp.logger` within `dp_serial` wraps every method in OpenDP such that it can store and export the AST required for remote execution.

```
import dp_serial.opendp_logger.trans as trans
import dp_serial.opendp_logger.meas as meas
import dp_serial.opendp_logger.comb as comb
```

```
pipeline = comb.make_pureDP_to_fixed_approxDP(
    trans.make_split_dataframe(separator=",", col_names=COLNAMES) >>
    trans.make_select_column(key="y_financial_actions", TOA=str) >>
    trans.make_cast(TIA=str, TOA=int) >>
    trans.make_impute_constant(0) >>
    trans.make_clamp(bounds=(0, 1)) >>
    trans.make_bounded_sum((0, 1)) >>
    meas.make_base_discrete_laplace(scale=1.)
)

eps, delta = pipeline.map(1)
```

```
print(eps, delta)

# only run _if_ you are sure you want to spend the budget to get the answer
```

```
1.0 0.0
```

```
opendp_result = competition_enclaves.opendp(pipeline)
```

```
print(opendp_result)
```

```
     0
0  162
```

```
trained_model.predict(test_x)
```

```
trained_modela.predict(test_x)
```

```
trained_modelb.predict(test_x)
```

Can I only use these outputs?

Yes and no. You absolutely do not need to use the above functions and stop there. You have many ways in which you can be strategic in your actions before and after. By really understanding `train_x` via visualization, dimensionality reduction, etc you can hopefully make more informed queries, spending less epsilon and delta. Equally, when you get an output, for example, some synthetic data, then you have the opportunity to figure out how to use that output to make informed predictions on `test_x`.

```
test_x[trained_model]
```

```
KeyError: Pipeline(steps=[('lr',
                LogisticRegression(accountant=BudgetAccountant(spent_budget
                                   data_norm=50))])
```

```
import pandas as pd
```

```
df = pd.DataFrame({'y_financial_actions':[trained_modelb], 'y_financial_help':
                   'y_return':[trained_modela]})
```

```
test_x[df.y_financial_actions]
```

```
KeyError: "None of [Index([(LogisticRegression(accountant=BudgetAccountant(s|
```

```
# import pandas as pd
# #Submission schema -> df = pd.DataFrame({'y_financial_actions':[1,0,1,1,....(
# #Please use test_x for no of rows
# res = competition_enclaves.submit_predictions_comp(submission=df)
```

```
import pandas as pd
#df = pd.DataFrame({'id':[1,2,3,4,5], 'labels':[0, 0, 0, 0, 1]})
#res = competition_enclaves.submit_predictions_comp(submission=df)
```

```
dataf = pd.DataFrame({'id':[test_x.index], 'labels':[df.y_financial_actions,df
```

```
ValueError: All arrays must be of the same length
```

```
df.y_financial_actions
```

```
pred = trained_model.predict(test_x)
```

```
preda = trained_modela.predict(test_x)
```

```
predb = trained_modelb.predict(test_x)
```

```
dataf = pd.DataFrame({'id':[test_x.index], 'labels':['predb']})
```

```
res = competition_enclaves.submit_predictions_comp(submission=df)
```

```
#Stats functions to check team status
competition_enclaves.get_total_epsilon()
```

```
'36.012'
```

```
competition_enclaves.get_total_delta()
```

```
'0.0012799750000000694'
```

```
competition_enclaves.get_accuracy()
```

```
'0.0'
```

```
competition_enclaves.get_score()
```

```
'0.0'
```