How to Participate?

* Submit a zip file of your python package for symbolic regression using scientific approaches to [Easychair??]
* Unzip the package will be a folder containing all the python files. Inside the folder, there should be a python program file called main.py. When we evaluate your program, we will execute the following command line argument:

python main.py \

--input\_eq\_name PATH\_TO\_THE\_INPUT\_SYMBOLIC\_EQUATION \

--time\_limit 3600

--output\_filename ../PATH\_TO\_THE\_OUTPUT\_FILE \ # save your predicted expression into this file

Here, input\_eq\_name contains the file name storing the ground-truth equation and other information such as the type and amount of noise used during evaluation, etc. The ground-truth equation is used by the oracle to generate the output [will be discussed later]. The input file will be encrypted using a private key that only the organizers know, and the actual structure of the input file will not be released to prevent participants from directly reading the ground-truth equation.

Time\_limit is used to let your program know the time limit of the evaluation. The unit is in seconds. Your program can design different strategies for competitions with different time limits. However, **your program will be killed once the time limit has been reached**, whether the program decides to quit itself or not.

The output\_filename designates the file you will need to write the symbolic equation you have found into.

* Your python package will need to access the oracle by calling the evaluate method from an object of the class Equation\_evaluator. In your main.py, you will need to initialize an Equation\_evaluator object. You can do this in the following way. First in the header of main.py:

from symbolic\_equation\_evaluator import \*

Then in the main program, execute:

Eq\_eval = Equation\_evaluator(input\_eq\_name)

Where the input\_eq\_name denotes the input file name passed from the command line argument.

* Equation\_evaluator provides you with this method:

def evaluate(self, X)

This method can be used as the oracle. When queried by the input X, it returns noisy estimations of . Here, The datatype of X is numpy.ndarray. It is a matrix. The first dimension corresponds to the batch size and the second dimension corresponds to number\_of\_variables. Basically, each row of X represents one input to the symbolic expression. The output of evaluate will be a vector, where the i-th output is the noisy estimation of .

* Other useful functions of the Equation\_evaluator class are:

def get\_nvars(self)

This function returns the number of input variables of the symbolic equation.

def get\_function\_set(self)

This function returns the set of operators possibly used in the symbolic equation. An example output looks like:

{‘sin’: 1, ‘cos’: 1, ‘+’: 2, ‘-‘ : 2, ‘\*’ : 2, ‘/’: 2, ‘const’: 0}

This output means the ground-truth equation may contain +, -, \*, / as binary operators, sin and cos functions as unary operators and real-valued constants. The numbers as the values of the dictionary denote the arity of the operators. The full set of operators we consider are [xxxxxxxxxxxxx]. Each competition track will use a subset of all operators.

* We give you symbolic\_equation\_evaluator\_public.py and an example input file for you to debug your symbolic regressor. The content in the python and input files should be self-explanatory. symbolic\_equation\_evaluator\_public.py is an example implementation of symbolic\_equation\_evaluator.py that we will use in the evaluation. However, the actual implementation of symbolic\_equation\_evaluator.py and the content in the input file will be different from the ones we provide to you, and the input file will be encrypted using keys only available to us.

DO NOT TRY TO FIGURE OUT WAYS TO DECODE THE GROUNDTRUTH EQUATIONS BY STUDYING THESE EXAMPLE FILES.

YOU SHOULD ONLY USE get\_nvars, get\_function\_set, and evaluate from the Equation\_evaluator object.

During the evaluation, we will first replace the symbolic\_equation\_evaluator.py in the zip folder with our own secret version. We will also manually check if your program calls methods other than the three suggested above. We will not include your submission in the evaluation if we find you use any side information other than that provided by the three methods.

* You will need to write the symbolic equation that you found into the output file. The output file contains only one line, the pre-order of the equation you have found. For example, the following line can be one valid equation in the output file:

[('+','binary'), ('\*', 'binary'), ('0.1', 'const'), ('X1', 'var'), ('sin', 'unary'), ('X2', 'var')]

This represents the equation 0.1\*X1 + sin(X2). You need to write ‘binary’ behind all binary operators and unary behind all unary operators (there are only these two types). Constants need to be followed by a string const, and variables followed by a string var.

Here is some explanation on what is the preorder traversal of an equation.

* Your python package can depend on other packages available for installation via pip. Provide the required list using requirement.txt