In charge of Black Box testing: Animesh KC

The primary testing approach used is equivalence classes
Fitness is measured in mean absolute error

Note for gplearn, functions such as division, square root, and logarithm are protected such that normally invalid inputs have valid outputs, albeit sacrificing the interpretability of these inputs in the process.
Equivalence Classes
gpn refers to the library used by gplearn for certain functions, namely protected functions such as division. The functions addition, subtraction, and

Input Condition	Regular Classes	Invalid Classes
Addition: x1 + x2	x0, x1 E R (1)	
Subtraction: x1 - x2	x0, x1 E R (2)	
multiplication: x1*x2	x0, x1 E R (3)	
division: gpn.div2(x1,x2)	x0 E R, x1 < -0.001 (4) x0 E R, x1 > 0.001 (5)	x0 E R, -0.001 <= x1 <= 0.001 (6)
square root: gpn.sqrt1(x1)	x0 >=0 (7)	x0 < 0 (8)
logarithm: gpn.log1(x1)	x0 >= 0.001 (9)	x0 < 0.001 (10)
absolute: gpn.abs1(x1)	x0 E R (11)	
maximum: gpn.max2(x1, x2)	x0, x1 E R (12)	
minimum: gpn.min2(x1, x2)	x0, x1 E R (13)	
sine: gpn.sin1(x1), x1 in radians	x0 E R (14)	

cosine: gpn.cos1(x1), x1 in radians	x0 E R (15)	
tangent: gpn.tan1(x1), x1 in radians	x0 E R (16)	
inverse: gpn.inv1(x1)	x0 < -0.001 (17) x0 > 0.001 (18)	-0.001 <= x0 <= 0.001 (19)

Restrictions:
Maximum of 2 variables at a time

ID & Description	Input and test domain	Expected R^2 compared to Test Data	Actual R^2 compared to test data
1 Testing the sinusoidal functions together along with the basic subtraction to join the terms (Classes 2, 14, 15)	z = gpn.cos1(x0) - gpn.sin1(x1) -10 <= x0 <= 10 -10 <=x1 <= 10	>=0.9	1.0
2 Testing the tangent function (Classes 1, 16)	z = gpn.tan1(x0 + x1) -10 <= x0 <= 10 -10 <= x1 <= 10	>=0.9	1.0
3 Testing division and inverse where the denominator is below 0. Testing addition to join terms. (Classes 1, 4, 17)	z = gpn.div2(x0, x1) + gpn.inv1(x1) -10 <= x0 <= 10 -10 <= x1 < -0.001	>=0.9	1.0
4 Testing division and inverse where the denominator is above 0. Testing multiplication to join terms (Classes 3, 5, 18)	z = gpn.div2(x0, x1) * gpn.inv1(x1) -10 <= x0 <= 10 0.001 < x1 <= 10	>=0.9	1.0
5	z = gpn.sqrt1(x0 + x2)	>=0.9	1.0

Testing Square Root of positive numbers (Class 7)	0 <= x0 <= 5 0 <= x1 <= 5		
6 Testing logarithms of positive numbers (Class 9)	z = gpn.log1(x0 / x1) 0.001 <= x0 <= 10 0.001 <= x1 <= 1	>=0.9	1.0
7 Testing absolute value function (Class 11)	z = gpn.abs1(5*x0 - 2* x2) -10 <= x0 <= 10 -10 <= x1 <= 10	>=0.9	0.998
8 Testing max and min functions (Classes 12 and 13)	z = gpn.max2(x0, 5*x1) * gpn.min2(4*x0, x1) -10 <= x0 <= 10 -10 <= x1 <= 10	>=0.9	0.998
9 Testing Invalid class 6	z = gpn.div2(8*x0, x1) -10 <= x0 <= 10 -0.001 <=x1 <=0.001	<0.5	-0.07
10 Testing Invalid class 8	z = gpn.sqrt1(x0 - 2x1) -10 <= x0 < 0 0 < x1 <= 10	<0.5	-0.19
11 Testing Invalid class 10	z = gpn.log1(x0 * x1) -10 <= x0 < 0 0 < x1 <= 10	<0.5	-0.19
12 Testing Protected class 19	z = gpn.inv1(x0 - x1) -10 <= x0 <= 10 -0.001 <= x1 <= 0.001	<0.5	0.05

outputted function of error guessing case 1:

```
(add(add(log(add(inv(div(sin(X0), mul(X0, 0.952))),neg(abs(X0)))), neg(inv(div(sin(X1), mul(X0,
0.952))))), min(add(log(add(min(mul(-0.020, X1), cos(X1)), neg(add(log(add(min(inv(div(sin(X1),
mul(X0, 0.952))), cos(X1)), div(sin(X1), add(log(add(min(mul(-0.020, X1), cos(X1)),
neg(add(log(cos(X1)), neg(0.952))))), add(tan(tan(sin(X1))), neg(inv(div(sin(X1), neg(div(X1,
```

0.794)))))))))), neg(0.952))))), add(tan(tan(sin(X1))), neg(inv(div(sin(X1), neg(div(X1, 0.794))))))), div(neg(cos(tan(X1))), inv(mul(add(X1, X1), log(X1)))))), div(neg(cos(inv(mul(add(X1, X1), log(X1))))), inv(mul(add(X1, X1), log(X1)))))

Error guessing Notes:
Do sinusoidals with coefficients
do complex functions within functions: e.g. logarithms within sinusodials

ID & Description	Input and test domain	Expected R^2 compared to Test Data	Actual R^2 compared to test data
1 Testing trigonometric functions within themselves	gpn.sin1(x0 + gpn.cos1(x1)) -10 <= x0 <= 10 -10 <=x1 <= 10	>=0.9	0.39
2 Testing division with embedded functions Choose a domain that will not cause division by zero or logarithm error	gpn.div2(gpn.log1(5*x0 +8), gpn.log1(6*x1 - 2)) -1 <= x0 <= 10 2 < x1 < 10	>=0.9	0.9
3 Testing square root with embedded functions	gpn.sqrt1(gpn.cos1(x0) * gpn.inv1(gpn.sin1(x1)) 0 <= x0 <= 10 0 <= x1 <= 10	>=0.9	0.81
4 Testing maximum and minimum within themselves	gpn.max2(gpn.min2(gpn. .cos1(x0), gpn.sin1(x1)), gpn.tan1(x0/x1)) 0 <= x0 <= 10 0 <= x1 <= 10	>=0.9	1.0
5 testing absolute with embedded functions	gpn.abs1(inv1(x0) * log1(8*x1)) 0.001 < x0 <= 10 0 < x1 <= 10	>=0.9	0.62
6	z = gpn.div2(x0, x1)	1.0	-27.14

Testing full assortment of functions together	+gpn.cos1(2x0) - gpn.sin1(3x1) * gpn.max1(6x0 +10, 2x1 - 8) - gpn.min1(x1 + 5, 3x0) + gpn.inv1(x1) - gpn.tan1(x0) * gpn.log1(x0) + gpn.sqrt1(x0) * gpn.abs1(-2x0) 0.001 <= x0 <= 10 -10 <=x1 < -0.001		

1			
1 Testing Regular classes 1, 2, 3, 4, 7, 9, 11, 12, 13, 14, 15, 16, 17 Testing classes with an airity of 2 separately.	z = gpn.div2(x0, x1) +gpn.cos1(2x0) - gpn.sin1(3x1) * gpn.max1(6x0 +10, 2x1 - 8) - gpn.min1(x1 + 5, 3x0) + gpn.inv1(x1) - gpn.tan1(x0) * gpn.log1(x0) + gpn.sqrt1(x0) * gpn.abs1(-2x0) 0.001 <= x0 <= 10 -10 <=x1 < -0.001	1.0	-4.359568