Comp 307 Assignment 2

Name: Yongbo Yu
ID: 300390526

Part 1:

Question 1(Determine and report the network architecture, including the number of input nodes, the number of output nodes, the number of hidden nodes (assume only one hidden layer is used here). Describe the rationale of your choice) :

The number of nodes in input layer is 4, because 4 different features in the iris dataset.
The number of nodes in the output layer is 3, because we have 3 different class in this dataset.
The number of nodes in the hidden layer is 2, I initialised it with 3, then I tried several different values, and find 2 and 5 nodes in the hidden layer have the similar performance. But I chose to use 2 because it can have lower computation cost, it can also save time.

Question 2 (Determine the learning parameters, including the learning rate, momentum, initial weight ranges, and any other parameters you used. Describe the rationale of your choice.):

The initial learning rate was 0.2 which follows the notes from the lecture, it did have a good performance. But I find the model converge very fast, therefore I reduce the learning rate. When I reduce it under 0.05, it will get stuck into local optimal. Therefore I set the learning rate to 0.1, which can have a nice converge period and would not stuck into local optimal.
I set momentum to 0, as we are not dealing with a very complex and this is a relatively small dataset, therefore we do not need momentum to speed up the process. Initial weight ranges was set to [-1 , 1], because from the lecture notes, it mentioned we usually want a small random value for the initial weights. Critical error was set to 0.001, because I want to minimize the error which can lead to a better performance. The classification accuracy was set to 200%, which means it will not stop the training process even the training accuracy reached 100%r to minimize the critical error. In the model I construct, the training process will stop when the error is less than 0.001.

Question 3 (Determine your network training termination criteria. Describe the rationale of your decision.) :

The network will be terminated when the error is less than the critical error which I set to 0.001. And the other condition to terminate the training process is the accuracy reach 200% but it can never happens. The reason I set the target accuracy is higher than 100% is to

minimize the critical error. The model with smaller critical error can have a better performance.

Question 4 (Report your results (average results of 10 independent experiment runs with different random seeds) on both the training set and the test set. Analyse your results and make your conclusions.) :

Run 1:

Training:

```
Mean squared error for training data: 0.001
Number of incorrect classifications: 0/75
```

Test:

```
Mean squared error for test data: 0.017
Number of incorrect classifications: 2/75
```

```
Run2:
Training:
Mean squared error for training data: 0.001
Number of incorrect classifications: 0/75

Test:
Mean squared error for test data: 0.017
Number of incorrect classifications: 2/75

Run3:
Training:
Mean squared error for training data: 0.001
Number of incorrect classifications: 0/75

Test:
Mean squared error for test data: 0.017
Number of incorrect classifications: 2/75

Run4:
Training:
Mean squared error for training data: 0.002
Number of incorrect classifications: 0/75

Test:
Mean squared error for training data: 0.018
Number of incorrect classifications: 2/75

Run5:
Training:
Mean squared error for training data: 0.001
```

```
Number of incorrect classifications: 0/75

Test:
Mean squared error for test data: 0.017
Number of incorrect classifications: 2/75

Run6:
Training :
Mean squared error for training data: 0.001
Number of incorrect classifications: 0/75

Test:
Mean squared error for test data: 0.017
Number of incorrect classifications: 2/75



Run7:
Training:
Mean squared error for training data: 0.001
Number of incorrect classifications: 0/75

Test:
Mean squared error for test data: 0.017
Number of incorrect classifications: 2/75

Run8:
Training:
Mean squared error for training data: 0.001
Number of incorrect classifications: 0/75

Test:
Mean squared error for test data: 0.017
Number of incorrect classifications: 2/75

Run9:
Training:
Mean squared error for training data: 0.001
Number of incorrect classifications: 0/75

Test:
Mean squared error for test data: 0.017
Number of incorrect classifications: 2/75

Run10:
Training:
Mean squared error for training data: 0.001
Number of incorrect classifications: 0/75

Test:
Mean squared error for test data: 0.017
```

```
Number of incorrect classifications: 2/75

The MSE value in Run4 is different from other runs. But it make
sense to me, because the initial weights is randomly generated.
The average accuracy of training is 100% and the accuracy on test
set is 97.33333%. The average MSE on test set is 0.017 which is
small as expected. The result seems to be reliable, but the
dataset I was using is small which only contains around 160
instances.
```

Question 5 ((optional/bonus) Compare the performance of this method (neural networks) and the nearest neighbour methods.) :

In the KNN model I construct in assignment I, when K = 3, I got the highest accuracy which is 96% on the Iris dataset. The accuracy of the neural network model is 97.33% which is slightly higher than the KNN model. For this dataset, neural network can have a better performance, but it does not mean that neural network method is always better than KNN. We need to use the more suitable methods for different problem.

Part 2 :

Question 1: (Determine a good terminal set for this task)

I used ECJ, more specific I am using tutorial4 from ECJ package. I used X as terminal set, which is the independent variable in the function. And I also construct a constant class as terminal set, it is a random value and used as a terminal node who does not have any children.

Question 2: (Determine a good function set for this task)

There are four different function nodes in the function set which are add, substract, multiply and protected divide.
The multiply, add and subtraction is already implemented in the package, and I only implemented the protect division myself, which will return 1 when the denominator is zero.

Question 3: (Construct a good fitness function and describe it using plain language (and mathematical formula, or other formats you think appropriate))

In this assignment I used Mean Square Error(MSE) as the fitness function. MSE is a measure of the quality of an estimator.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

The formula is shown above, it will sum up all the square of the difference between the predicted value and the expected value, then it will take the mean value of the summed result as the MSE value.

Question 4: (Describe the relevant parameter values and the stopping criteria you used.)

The number of generations was set to 80, after the 80 generations finished then the program stops.(Because I found the performance of the model did not increase much after 40 generations therefore 80 should be big enough for our model.)
The population was set to 2048.
The crossover rate was set to 0.8 the mutation rate was set to 0.1 and the reproduction rate was set to 0.1.
Tournament size was set to 7 which means we randomly select 7 individuals in the pools and use the individual with the best fitness.
Max depth = 17(program size), grow for subtree mutation is 5 (as many as the minimal depth), and the half builder is 6(minimal depth is 2 by the way).

(There are several important parameters in GP model. In ECJ we can simply change the parameters by write the parameter's name and the value we want in the tutorial4.params, which will automatically overwrite the value in the other packages like koza and simple packages when we run the GP model.)
After all these, I would like to mention I do change the acceptable error for making decisions that when it will decide it is a hit. It could be changed in the multiRegression class, I set that down to 0.005, which make can make the model more accurate.

```
#parameters
generations = 101
    # Subsidiary pipelines:
pop.subpop.0.species.pipe.num-sources = 3
pop.subpop.0.species.pipe.source.0 = ec.gp.koza.CrossoverPipeline
pop.subpop.0.species.pipe.source.0.prob = 0.8
pop.subpop.0.species.pipe.source.1 = ec.breed.ReproductionPipeline
pop.subpop.0.species.pipe.source.1.prob = 0.1
pop.subpop.0.species.pipe.source.2 = ec.gp.koza.MutationPipeline
pop.subpop.0.species.pipe.source.2.prob = 0.1
```

Stopping criteria:
When generation reaches the maximum generation value which I set to 80.
The standardized error was eliminated or adjusted value increase up to 1.(Which means the function that generate by GP is near optimal)

Question 5: (List three different best programs evolved by GP and the fitness value of them (you need to run your GP system several times with different random seeds and report the best programs of the runs))

a) :

Best Individual of Run:
Subpopulation 0:
Evaluated: true
Fitness: Standardized=7.031250000466827E-12 Adjusted=0.9999999999929687 Hits=20
Tree 0:
((x - (x * x)) - (((((((0.556695 * 0.032513) * (x - x)) / (x * (0.082981 * x))) - (0.655646 + x)) - ((x * x) - ((x * x) / (0.738944 - x)))) * (x - x)) - (x / ((x - (x * x)) * x)))) * (x + ((((0.801267 * (x / ((x - (0.683361 * (0.302983 * 0.022744))) * x))) * (x - x)) / (x * (x - (x * x)))) - (x * x)))

b)

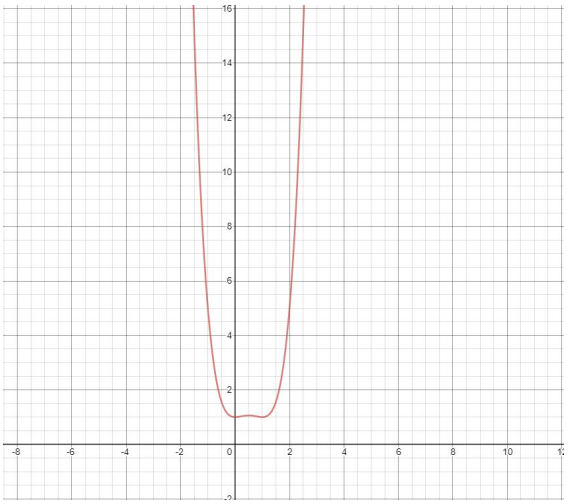
Best Individual of Run:
Subpopulation 0:
Evaluated: true
Fitness: Standardized=7.031250000466827E-12 Adjusted=0.9999999999929687 Hits=20
Tree 0:
((((x * x) * (x - 0.870560)) - (x * x)) / (0.515893 * ((x - x) / ((((x - (x * 0.187389)) - (((x - 0.241080) + (0.408200 * ((x - x) / ((((x - (x * 0.708515)) - ((x * x) - (x * 0.565184))) / 0.908803) + 0.287500)))) - (x * 0.559248))) / 0.171298) + 0.323309)))) + ((((((x * x) - ((x - (x / x)) + (0.712238 * ((x - x) / ((((x + x) + ((x - 0.017929) * (0.343666 + x))) / 0.102824) + 0.748472))))) - (x - x)) - x) * x) * x)

c) :

Best Individual of Run:
Subpopulation 0:
Evaluated: true
Fitness: Standardized=7.031250000466827E-12 Adjusted=0.9999999999929687 Hits=20
Tree 0:
((x * x) + (x / x)) + ((((((x * 0.530641) - (x * x)) - 0.900086) * ((x * x) - (x * x))) - (x * x)) * ((x + x) - (x * x)))

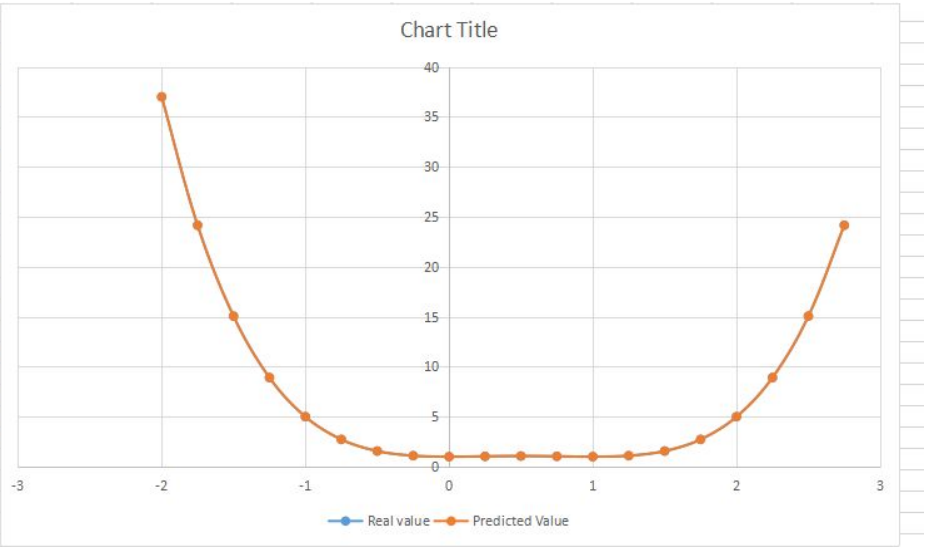
Question 6: ((optional, bonus) Analyse one of the best programs to reveal why it can solve the problem in the task.)

The best program I chose is the third one. And I tried plot the graph with the equation generated by GP.



The graph of the function generated by GP is shown above.
GP generated function :
((x * x) + (x / x)) + ((((((x * 0.530641) - (x * x)) - 0.900086) * ((x * x) - (x * x))) - (x * x)) * ((x + x) - (x * x)))
We can also simplify this equation, I just used sympy(Python) to simplify this equation down to y= x**3*(x - 2) + x**2 + 1. Then I put the real data and the predicted data into excel and the result is shown below:

X	Real value	Predicted Value
-2	37	37
-1.75	24.16016	24.16015625
-1.5	15.0625	15.0625
-1.25	8.91016	8.91015625
-1	5	5
-0.75	2.72266	2.72265625
-0.5	1.5625	1.5625
-0.25	1.09766	1.09765625
0	1	1
0.25	1.03516	1.03515625
0.5	1.0625	1.0625
0.75	1.03516	1.03515625
1	1	1
1.25	1.09766	1.09765625
1.5	1.5625	1.5625
1.75	2.72266	2.72265625
2	5	5
2.25	8.9101	8.91015625
2.5	15.0625	15.0625
2.75	24.16016	24.16015625



From the table we can see the predicted value is nearly the same value with the data provided. And we can not see any difference from two trend lines.
Therefore I think the model is near optimal.

Part 3:

Question 1: Determine a good terminal set for this task

For this dataset I am using ten terminal nodes as my terminal sets. They are x1,x2,x3,x4,x5,x6,x7,x8,x9 each nodes represent one feature in the dataset.

Question 2: Determine a good function set for this task.

As we are dealing with a more complex dataset, I added more function nodes into the function set. It includes add, sub, multiply, protected division, sin, cos, tan, square, square root, log, exponential. Some of the function set requires two children and some of them just require one child like sin and cos, which is specified in the tutorial4.params file.(will be used as argument in the program)

Question 3:Construct a good fitness function and describe it using plain language (and mathematical formula, or other formats you think appropriate).

I changed the fitness function to error rates. Because we are dealing with a classification problem, therefore I am using the error rates to determine the performance of the model. The formula of the fitness function is shown below.

$$\text{Error Rate} = \frac{Number of Instances - Number of Hits(Correct classified Instances)}{Number of Instances} = \frac{Number of Incorrect Classified}{Number of Instances}$$

Question 4:Describe the relevant parameter values and the stopping criteria you used

The generation was set to 80.(Increase the number of generations from 50 to 80 because we are dealing with a more complex problems comparing to the one I did in part 2, therefore I decided to let the program evolve more to have a better performance. Most programs can converge around 50 generations, therefore I set it to 80.)
The population size was 1024 that in each generation it will generate 2048 individuals.
(I did this because I want to increase the diversity, by initialising more individuals.)
The Mutation rate was 0.05. The reproduction rate was 0.05. The crossover rate is 0.9.(Try several times found this value can lead to a high performance)
The tournament size was 7.
The maximum tree depth was set to 17.
The minimum mutation depth was decreased from 5 to 2.
The rate of hal-grow-half-ramp is 0.5

When the program reaches 100 generations it will terminate the program.
The standardized error reaches 0 or adjusted reaches 1.
When adjusted reaches 1 means, the model is perfectly suitable for the data we feed in.
Otherwise the program will be forced stop when the gernations reaches 100.

Question 5: Describe your main considerations in splitting the original data set into a training set training.txt and a test set test.txt.

Based on the information given in the handout, there are 699 instances. There are 458 benign instances and 241 malignant instances. I decide to use 30% of the whole dataset as my test set and the rest of the dataset using as training set. The other concern is to balancing my training data to build a robust model. But as the dataset given is very unbalanced and this question is not very complex. So I just randomly chose 30% malignant instances and 30% benign instances from the original dataset, and build the test set with this chosen instances. And the rest used as my training set.

Question 6: Report the classification accuracy (average accuracy over 10 independent experiment runs with different random seeds) on both the training set and the test set.

Average Training Rate = 97.42%
Average Test Rate = 95.238%

Run 1:
Best Individual of Run:
Subpopulation 0:
Evaluated: true
Fitness: Standardized=0.028056112224448898 Adjusted=0.9727095516569201 Hits=485
Tree 0:
(square(rlog(rlog(x5 * x3) * x5)) - sin(x3)) - cos(((((exp(square(rlog(rlog(x2) * x5))) * (rlog(x5 * rlog(x2)) - sin(x3))) - sin(x3)) - cos(((x5 - tan(square(x5 * x5))) * tanh(cos((((x5 * tanh(rlog(x2))) - sin(x3)) - cos(sqrt(x5))) * tanh((sin(x3) - sin(x3)) - sqrt(sqrt(x5)))))) - ((square(rlog(rlog(x5 * x3) * x5)) - sin(x3)) - cos(sqrt(x5))))) * tanh(rlog(x2))) * tanh(((exp(square(rlog(rlog(x2) * x5))) * square(rlog(rlog(exp(square(rlog(rlog(x2) * x5)))) * x5))) - sin(x3)) - exp(cos(x5))))

Performance of Best Individual on Testing Set:

Fitness: Standardized=0.035 Adjusted=0.9661835748792271 Hits=193

Run 2

Best Individual of Run:
Subpopulation 0:
Evaluated: true
Fitness: Standardized=0.018036072144288578 Adjusted=0.9822834645669293 Hits=490
Tree 0:
exp(cube(rlog(square(tan(sin(x5)) * square(square(rlog(x3) * rlog(x5)) + rlog(rlog(cos(square(exp(x2)))))))) * square((square(rlog(x3) * square(cos(square(exp(x2))))) * square(rlog(square(rlog(x3)) * square(tan(sin(x5)))))) * square(rlog(x5)))) / x5) * (((((((rlog(x3) * square(cos(square(exp(x2))))) * square(square(rlog(x3) * square(cos(square(exp(x2))))) + rlog(rlog(cos(square(exp(x2))))))) + (x5 * (square(x3 + square(rlog(x5))) * (((square(rlog(x3)) * rlog(x3)) * square(rlog(x5))) * square(rlog(x5)))))) * square(square(rlog(cos(square(exp(x2)))) * square(rlog(x5))) + rlog(square(rlog(x3))))) * (rlog(x3) * (rlog((rlog(x3) * (square(rlog(x3)) * square(rlog(x5)))) + (tan(sin(rlog(sqrt(x2)))) * square(square(rlog(x3) * rlog(x5)) + rlog(rlog(cos(square(exp(x2)))))) * square(x5)))) - sin(x4)) - tan(sin(x5)))

Performance of Best Individual on Testing Set:

Fitness: Standardized=0.03 Adjusted=0.970873786407767 Hits=194

Run3

Best Individual of Run:
Subpopulation 0:
Evaluated: true
Fitness: Standardized=0.03006012024048096 Adjusted=0.9708171206225682 Hits=484
Tree 0:
((cube(sin(x5)) / ((x5 / (cube(rlog(x3 * x5) - (x5 + x5)) + x5)) + x5)) - (sin((rlog(x3 * rlog(x2)) - cube(sin(((sin(cube(x5)) - rlog(x5 / (x5 + x5))) - x3) / ((rlog(x5) - (rlog(x3 * (x5 + x5)) - cube(sin(x5)))) + (x5 + x5))))) + (((x5 + x5) - cube(cube(sin(x5)))) + rlog(cube(rlog(sin(x5)) - cube(x5))))) - sqrt(exp(x2)))) / (((rlog(x2) * (cube(rlog(x3 * x5) - cube(sin(x5))) + rlog(cube(sin(x5)) / rlog((((rlog(x5 / (x5 + x5)) * cube(x5 / (x5 + x5))) - x5) + rlog(x5 / (x5 + x5))) / ((x5 + x5) + x5))))) - sin(x5)) + rlog(x5 / (x5 + x5)))

Performance of Best Individual on Testing Set:

Fitness: Standardized=0.015 Adjusted=0.9852216748768474 Hits=197

Run 4

Best Individual of Run:
Subpopulation 0:
Evaluated: true
Fitness: Standardized=0.03406813627254509 Adjusted=0.9670542635658915 Hits=482

Tree 0:
rlog(rlog(sqrt(exp(square(rlog(x5))) * exp(rlog(x2))) * rlog(x3))) *
(sqrt(exp(cube(sqrt(((rlog(x2) * (exp(rlog(x2)) * x3)) * rlog(x5)) * square(rlog(x1 * x5))))) *
exp(rlog(square(rlog(x5))))) * sqrt(sqrt(exp(exp(x2) * (x3 * x5)) * exp(exp(rlog(x2))))))

Performance of Best Individual on Testing Set:

Fitness: Standardized=0.64 Adjusted=0.9697560975609756 Hits=192

Run 5:
Best Individual of Run:
Subpopulation 0:
Evaluated: true
Fitness: Standardized=0.026052104208416832 Adjusted=0.9746093750000001 Hits=486
Tree 0:
((((x3 * rlog(x3)) * ((((rlog(x3) * x3) * (rlog(x5 + rlog(((rlog(x3) * rlog(x5)) * rlog(x5)) * rlog(x5)))
* x5)) * x5) * ((rlog(x3) * x5) * rlog(rlog(x3) * x5)))) - sin(x2)) * x5) - sin(tan(cos(rlog(x2))))

Performance of Best Individual on Testing Set:

Fitness: Standardized=0.015 Adjusted=0.9852216748768474 Hits=197

Run 6:
Best Individual of Run:
Subpopulation 0:
Evaluated: true
Fitness: Standardized=0.026052104208416832 Adjusted=0.9746093750000001 Hits=486
Tree 0:
square(rlog(x3) * (rlog(x3) * square(((rlog(x3) * rlog(x5)) - cos(sin(cos(x3)) -
rlog(rlog(rlog(x5)) - x2))) * square(rlog(x5))))) - cos(rlog((rlog(rlog(x2)) * ((rlog(rlog(rlog(x5 -
x2) - x2)) - rlog((x5 - x2) - x2)) * (x5 * x3))) - x2))

Performance of Best Individual on Testing Set:

Fitness: Standardized=0.02 Adjusted=0.9803921568627451 Hits=196

Run 7:
Best Individual of Run:
Subpopulation 0:
Evaluated: true
Fitness: Standardized=0.02404809619238477 Adjusted=0.9765166340508806 Hits=487
Tree 0:

rlog(((rlog(x5 * rlog(x3)) * rlog(x3)) * rlog(x3)) * (rlog((rlog(x3) * x5) * (exp(rlog(x3) * (rlog(x2) * (x5 * exp(rlog(rlog(x2) * x5) * (rlog(x2) * x5))))) * (rlog(x3) * x5))) * ((rlog(x5 * ((x5 - exp(cos(rlog(x2)))) * (rlog(x3) * x5))) * (rlog(x3) * (rlog(x5 * rlog(x3)) * ((rlog(x5 * (rlog(x5 * rlog(x3)) * (rlog(x3) * x5))) * (x2 * rlog(rlog(x5 * rlog(x3)) * (rlog(x3) * x5)))) * ((rlog(x3) * x5) * x5))))) * (rlog(x5 * rlog(x3)) * (rlog(x3) * x5)))))

Performance of Best Individual on Testing Set:

Fitness: Standardized=0.05 Adjusted=0.8560606060606061 Hits=170

Run 8:
Best Individual of Run:
Subpopulation 0:
Evaluated: true
Fitness: Standardized=0.03006012024048096 Adjusted=0.9708171206225682 Hits=484
Tree 0:
(exp((exp(cos(x5)) * (rlog(x3) * ((rlog(x3) * rlog(x5)) * exp(rlog((((sqrt(x2 * x3) + sqrt(x2 * x3)) * cos(x5)) * rlog(x2 * (rlog(x3) * rlog(x5)))) * rlog(x5)))))) * x5) - exp(cos(rlog((sqrt(x2 * x3) * rlog(rlog(x5) * x3)) * x5)))) - exp(cos(cos(sin(x5 * x5))))

Performance of Best Individual on Testing Set:

Fitness: Standardized=0.015 Adjusted=0.9852216748768474 Hits=197

Run 9:
Best Individual of Run:
Subpopulation 0:
Evaluated: true
Fitness: Standardized=0.02404809619238477 Adjusted=0.9765166340508806 Hits=487
Tree 0:
(rlog(rlog(x3 * x5)) - sin(sin(rlog(square(((tan(x5 + exp(rlog(rlog(x3)))) * sin(tan(x5))) / exp(square(rlog(x5 * (rlog(x5 * x5) * x5))))) + ((tan(x5 + x5) * sin(x5 * x5)) / exp(square(rlog(rlog(x5 * x5) * x5) * rlog(x5)))))) / (sin(tan(x5)) * x5))) + ((tan(x5 + exp(rlog(rlog(x3)))) * sin(tan(x5))) / exp(square(rlog(exp(square(rlog(x5 * (rlog(x5 * x5) * x5))))))))

Performance of Best Individual on Testing Set:

Fitness: Standardized=0.02 Adjusted=0.9803921568627451 Hits=196

Run 10:

Best Individual of Run:
Subpopulation 0:
Evaluated: true
Fitness: Standardized=0.02404809619238477 Adjusted=0.9765166340508806 Hits=487
Tree 0:
(((rlog(x3) - sin(exp(x5))) * square(rlog((rlog(square(rlog(rlog(square(rlog(sin(exp(x5)))))))) - ((rlog(rlog(x3) * (x5 * sin(rlog(x2)))) * x5) / x3)) * x2) * square(rlog(x5 * x5)))) - sin(exp(rlog(rlog(x3)) * square(rlog(x2) * sin(square(exp(x5)))))) * square(rlog(((rlog(x3) - (rlog(x3) / rlog(rlog(x5 * x3)))) * x2) * ((rlog(x3) - (((sin(exp(rlog(rlog(x3)) * square(sin(rlog(x2))))) * square(rlog(x5 * x5))) * x5) / rlog(rlog(x5 * x3)))) * rlog(sin(exp(x5))))) * rlog((rlog(x5) - ((x1 * x5) / x3)) * x2))

Performance of Best Individual on Testing Set:

Fitness: Standardized=0.185 Adjusted=0.8438818565400843 Hits=163

Question 7 : List three best programs evolved by GP and the fitness value of them.

Program 1:

Run 6:
Best Individual of Run:
Subpopulation 0:
Evaluated: true
Fitness: Standardized=0.026052104208416832 Adjusted=0.9746093750000001 Hits=486
Tree 0:
square(rlog(x3) * (rlog(x3) * square(((rlog(x3) * rlog(x5)) - cos(sin(cos(x3)) - rlog(rlog(rlog(x5)) - x2))) * square(rlog(x5))))) - cos(rlog((rlog(rlog(x2)) * ((rlog(rlog(rlog(x5 - x2) - x2)) - rlog((x5 - x2) - x2)) * (x5 * x3))) - x2))

Performance of Best Individual on Testing Set:

Fitness: Standardized=0.02 Adjusted=0.9803921568627451 Hits=196

Program 2:

Run 9:
Best Individual of Run:
Subpopulation 0:
Evaluated: true
Fitness: Standardized=0.02404809619238477 Adjusted=0.9765166340508806 Hits=487

Tree 0:
(rlog(rlog(x3 * x5)) - sin(sin(rlog(square(((tan(x5 + exp(rlog(rlog(x3)))) * sin(tan(x5))) / exp(square(rlog(x5 * (rlog(x5 * x5) * x5))))) + ((tan(x5 + x5) * sin(x5 * x5)) / exp(square(rlog(rlog(x5 * x5) * x5) * rlog(x5))))))) / (sin(tan(x5)) * x5))) + ((tan(x5 + exp(rlog(rlog(x3)))) * sin(tan(x5))) / exp(square(rlog(exp(square(rlog(x5 * (rlog(x5 * x5) * x5))))))))

Performance of Best Individual on Testing Set:

Fitness: Standardized=0.02 Adjusted=0.9803921568627451 Hits=196

Performance of Best Individual on Testing Set:

Fitness: Standardized=0.135 Adjusted=0.8810572687224669 Hits=173

Program 3:

Run 2
Best Individual of Run:
Subpopulation 0:
Evaluated: true
Fitness: Standardized=0.018036072144288578 Adjusted=0.9822834645669293 Hits=490
Tree 0:
exp(cube(rlog(square(tan(sin(x5)) * square(square(rlog(x3) * rlog(x5)) + rlog(rlog(cos(square(exp(x2)))))))) * square((square(rlog(x3) * square(cos(square(exp(x2))))) * square(rlog(square(rlog(x3)) * square(tan(sin(x5)))))) * square(rlog(x5)))) / x5) * (((((((rlog(x3) * square(cos(square(exp(x2))))) * square(square(rlog(x3) * square(cos(square(exp(x2))))) + rlog(rlog(cos(square(exp(x2))))))) + (x5 * (square(x3 + square(rlog(x5))) * (((square(rlog(x3)) * rlog(x3)) * square(rlog(x5))) * square(rlog(x5)))))) * square(square(rlog(cos(square(exp(x2)))) * square(rlog(x5))) + rlog(square(rlog(x3))))) * (rlog(x3) * (rlog((rlog(x3) * (square(rlog(x3)) * square(rlog(x5)))) + (tan(sin(rlog(sqrt(x2)))) * square(square(rlog(x3) * rlog(x5)) + rlog(rlog(cos(square(exp(x2)))))))) * square(x5)))) - sin(x4)) - tan(sin(x5)))

Performance of Best Individual on Testing Set:

Fitness: Standardized=0.03 Adjusted=0.970873786407767 Hits=194

Question 8: Analyse one of best programs to reveal why it can solve the problem in the task.

Take the second program as example.
(rlog(x3) * (rlog(x3) * square(((rlog(x3) * rlog(x5)) - cos(sin(cos(x3)) - rlog(rlog(rlog(x5)) - x2))) * square(rlog(x5))))) - cos(rlog((rlog(rlog(x2)) * ((rlog(rlog(rlog(x5 - x2) - x2)) - rlog((x5 - x2) - x2)) * (x5 * x3))) - x2))

The function generated by GP is shown above. It requires values for several features, then the result from this formula will determine which class this data should be. For example if the result from this function is less than 0, then it will be classified as benign otherwise it will be classified as malignant.
The function represent the relationships between features and the class. With this program we can classified the data based on these nine features. But mathematically it is hard to create this function, but with GP it can automatically generates functions for this problem.