Report for CSCI6506 Sandbox 2

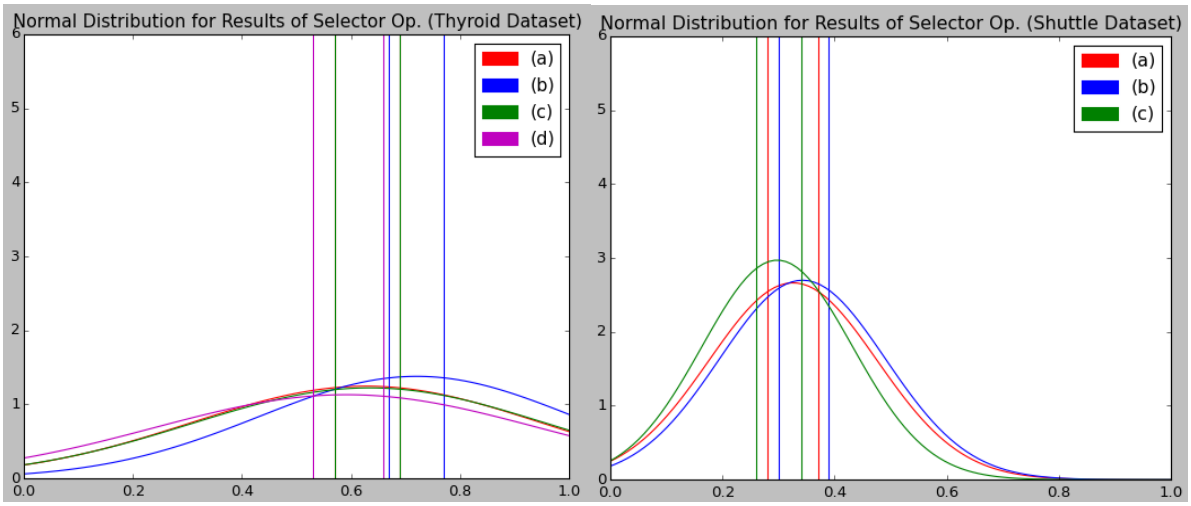
Jéssica Pauli de Castro Bonson (B00617515)

The implementation of the canonical GP for multi-class classification improved with sampling and a class-wise detection rate (DR) used the following final configurations:

Parameters	**Thyroid dataset**	**Shuttle dataset**
Population Size	200	200
Total Generations	20	20
Crossover Rate	0.1	0.1
Mutation Rate	0.9	0.9
Total Runs	50	20
Initial Program Size	32	40
Max. Program Size	64	80
Extra Registers	2	3
Fitness Function	Accuracy	(MSE+Accuracy)/2
Sampling Heuristic	$\tau/\|C\|$	$\tau/\|C\|$
Sample Size (τ)	420 (140 per class)	420 (60 per class)
Selector Operator	Proportional Selection	
Variation Op.	Crossover and Mutation	
Stop Criterion	While current_generation < generation_total	
Best Solution		
- accuracy	0.92	0.7
- DR	0.86	0.38

This values were obtained by comparing the result of sets of runs of the GP algorithm with each other, only modifying one variable per time. The comparisons were between the normal distribution of the results between each set of runs, using the metric (accuracy+DR)/2. The attribute values were normalized between -1 <= x <= 1 using the formula (x-mean)/(max-min). In the next paragraphs I will explain the details about the fitness function, the sensitivity of the parameter τ, the sampling policy, and then show the best trained programs for each dataset.

1. **Fitness Function:** To choose the best fitness function I compared, for both datasets, the normal distribution of the results of 4 fitness functions. The first two are the best performing functions from Box1, i.e. accuracy-based (a) and (MSE+accuracy/2)-based (b). The two other functions are the first ones with an additional parameter for the DR, with the same weight as the first parameter. So the function (c) is based on accuracy and DR, and (d) is based on accuracy, MSE, and DR. The following figures show the normal distribution results for these fitness functions for both datasets with a confidence interval of 80%.

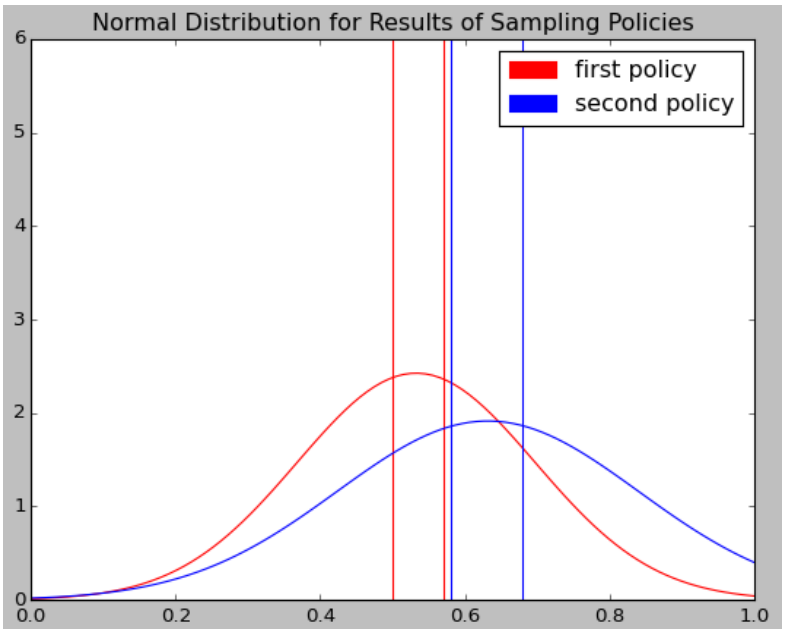


For the Thyroid dataset, (a) and (c) got the same distribution values. The only statistically relevant result is that, for this confidence level the (d) function was the worse one, and (a), (b), and (c) are statistically similar. Due to these results, the Shuttle dataset wasn't tested with function (d), and the results for this dataset also didn't show any relevant statistically difference between (a), (b), and (c). Besides that, (a) produced the best solution for the Thyroid dataset and (c) the best one for the Shuttle dataset.

2. **Parameter τ:** This value affected mainly the Shuttle dataset, since it has a lot of variance between the samples per classes and, for example, a sample of 50 elements per class is too few for class 1, with thousands of data items, so it may decrease the accuracy for this class. But it is already bigger than the total data for classes 2, 3, 6, and 7, so a even higher value would unbalance the sample, and may decrease the DR metrics. For higher values of τ, the accuracy increased and the DR decreased as shown in the table below. By the results it seems that this parameter affects the DR metric way more than the accuracy metric.

	τ = 140	**τ = 420**
Accuracy (avg; std-dev)	0.367; 0.198	0.418; 0.273
DR (avg; std-dev)	0.256; 0.09	0.233; 0.067

3. **Sampling policies:** Besides the data cardinality, an issue that is also addressed by the sampling policy is the training data distribution per class. The second heuristic (τ/|C|) for sampling policy gave much better results for the DR metric, but a bit lower ones for the accuracy metric, since it sampled a balanced amount of samples per class, without taking the class weights in account. Also, the second policy produced better results for the metric (accuracy+DR)/2, since the high increase in the DR compensated the small decrease in the accuracy. The sample subset is replaced at each generation. A comparison between the data distribution of the outputs (accuracy+DR)/2 metric) for a set of runs with the same parameters for the Shuttle dataset with a 85% confidence interval is shown below, confirming at a 0.15 level that the second policy is better than the first one for the evaluation metric being used.



Algorithm info:Class Distributions: defaultdict(<type 'int'>, {0: 11478, 1: 13, 2: 39, 3: 2155, 4: 809, 5: 4, 6: 2}), for a total of 14500 samples

4. **Best Classifier:** The next table show the values of the best found solutions, according to the metric (DR+accuracy)/2. The code for each solution is also shown below. Since the Shuttle dataset had very unbalanced data per class, it was much harder to find a solution that performed well on both. The solution was able to have a good accuracy in the biggest class (class 1, 11478 items in testset), and reasonable ones in some smaller classes (1, 2, and 4, with 13, 39 and 809 items each). But it totally missed the classes 4, 6 and 7. 6 and 7 are actually very small (4 and 2 items each), but class 4 has 809 items so probably it would be possible to find a solution that included class 4 with more trials. Both solutions were obtained at generation 13th, they also have the same length of instruction (38), with 2 and 5 introns each.

Best Solution	**Thyroid dataset**	**Shuttle dataset**
Accuracy (testset)	0.92	0.7
DR (testset)	0.86	0.38
Recall per class (testset)	[0.86, 0.78, 0.92]	[0.86, 0.87, 0.59, 0.0, 0.36, 0.0, 0.0]
Fitness	0.88	0.39
Generation	13	13

r[0] = r[0] - r[0] r[4] = r[4] / r[4] r[2] = r[2] - r[3] r[0] = r[0] * i[1] r[0] = r[0] + r[3] r[1] = r[1] - i[4] r[1] = r[1] / i[18] r[4] = r[4] + r[3] r[1] = r[1] + i[10] r[0] = r[0] + r[3] r[0] = r[0] + i[13] r[3] = r[3] - i[13] r[4] = r[4] / r[3] r[0] = r[0] / i[16] r[2] = r[2] - i[12]	r[0] = r[0] + r[1] r[4] = r[4] - r[0] r[5] = r[5] + r[7] r[6] = r[6] * i[4] r[3] = r[3] / i[1] r[7] = r[7] + i[5] r[1] = r[1] - i[0] r[2] = r[2] - r[5] r[1] = r[1] + r[3] r[9] = r[9] - i[1] r[4] = r[4] * i[8] r[3] = r[3] - r[7] r[9] = r[9] * i[3] r[2] = r[2] + r[3] r[2] = r[2] * i[1]

r[1] = r[1] - i[3] r[3] = r[3] - r[4] r[1] = r[1] / i[5] r[3] = r[3] / r[0] r[1] = r[1] / r[2] r[2] = r[2] + i[3] r[2] = r[2] + r[0] r[2] = r[2] / r[3] r[1] = r[1] / r[4] r[1] = r[1] + r[3] r[2] = r[2] + i[0] r[3] = r[3] + i[8] r[0] = r[0] - r[1] r[2] = r[2] / r[2] r[3] = r[3] / r[0] r[1] = r[1] * r[3] r[0] = r[0] * i[18] r[0] = r[0] - r[4] r[3] = r[3] - r[4] r[3] = r[3] * r[1] r[4] = r[4] - r[4] r[2] = r[2] - r[3] r[4] = r[4] * i[2]	r[9] = r[9] - i[6] r[8] = r[8] + r[8] r[3] = r[3] - i[5] r[5] = r[5] + r[5] r[6] = r[6] + r[4] r[0] = r[0] + i[3] r[1] = r[1] * i[2] r[5] = r[5] / i[6] r[3] = r[3] / i[1] r[3] = r[3] / i[2] r[1] = r[1] * r[5] r[3] = r[3] / i[0] r[4] = r[4] + r[7] r[5] = r[5] / i[5] r[3] = r[3] * r[5] r[2] = r[2] * r[4] r[3] = r[3] * i[8] r[9] = r[9] + r[4] r[9] = r[9] - i[8] r[0] = r[0] - i[8] r[4] = r[4] + r[8] r[1] = r[1] * i[1] r[6] = r[6] * i[8]
Code for best program (Thyroid dataset)	***Code for best program (Shuttle dataset)***