Genetic Algorithm for Data mining and Classification of Data and rules

**AHMED THOLAAL - 17020343**

**1 INTRODUCTION**

In this paper, I will be explaining my attempts on solving the three problems given with three data set. The task was to solve a set of classification problems using any of the evolutionary intelligence covered on the course.

The data set 1 and data set 2 are binary string representation while data set 3 has real number values. The aim of the assignment is to use algorithms developed by the inspiration from nature such as Simple genetic algorithm to show that it is possible to detect patterns in data using an existing data set of rules or training dataset and classify a given data set. And use this dataset to accurately classify and predict the future data sets called test sets.

Firstly, I constructed simple genetic algorithm program using the object-oriented java programming language. Simple genetic algorithm is used because it can evolve overtime to classify data when correctly set up. Then it is amended so that its fitness function is modified such that it can classify the provided data. In this assignment, parameters such as mutation rate, population size is adjusted to see the best values for these parameters.

**BACKGROUND RESEARCH**

Data mining is the analysis and exploration of huge amount of data to identify meaningful pattern, rules, aims and to predict future outcome (MicroStrategy, 2019). “Data mining “ that word was given because searching for large-scale data for pattern is similar to mining a mountain for a valuable mineral ore since both method require either shifting through massive amount of material, or intelligently probing it to find where the value resides (Alexander, 2019). In industries there are many cases where data mining technique is used. In customer-oriented business, data mining technique is used to create a buying profile of their customers so that they can better target their customer with more relevant and customized promotion.

Traditionally, certain questions were too complex and lengthy to come up with a solution, however by using data mining technique it is possible answer these questions with in a reasonable time frame because they search databases for hidden patterns, finding predictive information that experts may miss because it lies outside their expectation. Data mining is also used at medical care industry where it is used to uncover hidden relationship between illnesses and known drug treatments and identify trends that help pinpoint which drugs are the most effective for what type of patients (Alexander, 2019).

Even though data mining is useful, it is not all time one hundred percent accurate as there are some occasions where it can also sometimes predict wrong results. This is due to quality of the data gathered.

There are many methods used in data mining, this paper will try to resolve this problem using Genetic Algorithm.

**GENETIC ALGORITHM**

In 1970s, John Holland came up with the search heuristic that is inspired by Charles Darwin’s theory of evolution. In this algorithm, its emphasis on the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation (Mallawaarachchi, 2019).

Such as it happens in nature, genetic algorithm also uses processes that occur in nature,

they are:

* Selection
* Crossover
* Mutation.

Genetic Algorithm pseudo code is as follows:

*INITIALISE* population with random candidate solution

*EVALUATE* each candidate solution

*WHILE* termination condition is not true *DO*

**SELECT** individuals for the next generation

**RECOMBINE** pairs of parents

**MUTATE** the resulting offspring

**EVALUATE** each candidate solution.

END

**Initializing and Encoding**

One of the reason genetic algorithm is popular for its robustness since it works not only on single solution at time but multiple solutions concurrently. To facilitate this, genetic algorithm has population of individuals. In genetic algorithm, population is the set of individuals. The initial population is created randomly and aach individual of the population is characterized by set of parameters known as Genes and these genes joined into a string to form a chromosome. Each chromosome is a possible solution (Mallawaarachchi, 2019). A chromosome can be encoded with either alphabets, binary numbers or real values numbers. It really depends on the optimization problem the user is working.

In the case of data set 1 and data set 2, genetic algorithm can randomly create individuals by giving random binary digits as genes to the chromosome. For instance, if the population size is 100, then 100 individuals (chromosomes) will be created randomly giving random binary digits to its genes.

Once the population is initialized, then the next phase is Selection.

**Fitness Function**

Fitness function is one of the most important components of genetic algorithm. The fitness function takes an individual and determines the quality of the solution or how well it fulfils whatever the criteria the algorithm is optimizing for(Kiely, 2019).Each individuals fitness score will be evaluated and set using this function to determine whether they should be allowed to reproduce and make offspring.

**Selection**

The selection process selects individual or solutions from the mating pool directed by the survival of the fittest concept of natural genetic systems to mate and to create new children for the next generation (Maulik and Bandyopadhyay, 2000). The type of the selection used in this genetic algorithm is fitness proportionate selection, in which every individual can become a parent however the probability of becoming a parent depends on its fitness score. The fitter the individual the better chance it has, to become a parent (Tutorialspoint.com, 2019). One of the most popular two selection method is tournament selection and roulette-wheel selection.

Roulette-wheel selection was introduced in 1975 by Holland and is based on concept of proportionality. The area of the roulette-wheel is divided as per the fitness scores of the individual; the fitness value of each individual in a population corresponds to the area of the roulette-wheel proportions. Then the wheel is spun, an individual marked by the roulette-wheel pointer is then selected (Abd Rahman et al., 2016). The probability of getting chosen is directly proportional to the fitness score. Bigger fitness score means bigger area in the wheel allocated to the individual and better chance to get selected. There are some disadvantages involved in roulette-wheel selection, it cannot handle negative values due to proportionality concept and when population converges, it loses selection pressure (Abd Rahman et al., 2016).

One of the advantages of tournament selection is it allows negative values and its ability to either handle maximization or minimization problem without any structural changes. In tournament selection two individuals are selected by random. Then fitness of these two are compared and the one with the more fitness score is chosen for mating (Abd Rahman et al., 2016). This method ensures that there could be some weak scoring members ensuring the randomness in the population and maintaining good diversity.

**Crossover**

Crossover is the step after the selection process. Once parents are selected, they are now ready for recombination or crossover. This operator is used to combine genetic material from two parents to create new offspring. There are many types of crossovers. In this paper I focus on single point crossover, hence a single point is randomly selected by random number generator and this point is the point where parents will swap their tails to mix the genes and create new offspring. In the case of dataset 1 and dataset 2, both use binary values as genes hence these genes will be mixed to create hopefully better fitter individuals then the parents. The idea is that the next generation is expected to be fitter than the current generation eventually will be replaced by the next generation.

**Mutation**

The aim of mutation is to restore lost or unexplored genetic material into the population to prevent the premature convergence of the genetic algorithm to suboptimal solutions (Srinivas and Patnaik, 1994). Mutation modifies the value of each ‘gene’ of an individual with the mutation probability (Pm) to get a new solution. Like in crossover, mutation also has many types of operators such as bit Flip mutation, Random Resetting, swap mutation and scramble mutation etc. Compare to crossover probability, mutation probability is usually very small because too high mutation rate means more exploitation rate and in turns prevents population to converge to any optimum solution. Mutation rate really depends on the specific problem. By adjusting mutation rate one can identify the best value for the specific problem.

**Evaluate**

As seen in the above GA pseudocode, in the evaluation phase of genetic algorithm, determines which solutions to be eliminated and remove from the population and which solutions to be kept for the next generation. When keeping individual for the next generation, both fitness score and diversity of the population is considered. There are fitness-based selection and age-based selection. In this paper, fitness-based selection is used. Which means offspring tend to replace least fit solutions in the population. As the fitter solutions occupy, usually old less fit individuals are kicked out from the population and do not get the chance to reproduce. The average fitness increases usually for each next generation until termination condition is met.

**Termination condition**

The genetic algorithm will keep running until the termination condition is met. There may be several reasons for termination of the algorithm such as finding satisfied solution that meet certain minimum criteria or algorithm has run maximum number of times that programmer set it to. Another reason could be the highest-ranking individuals’ fitness has reached a plateau such that successive iterations no longer produces better results (Yoshida et al., 2008). Once the algorithm stops, the final population is returned by the program.

The mentioned operators such as selection, crossover, mutation etc. will be used in this assignment. The adjustments made will be discussed in the coming sections.

**Classification**

Classification is technique to categorize the data into a desired and distinct number of classes where label can be assigned to each class (Gahukar, 2019).

Binary classifiers classify with only two different classes in some case it is either 1 or 0 as the output. Classification is used in many fields such as spam email detection. Genetic algorithms are used to discover classification rules. It is in the form IF X THEN Y, where X is antecedent of the rule and is form from a conjunction of conditions and Y is the consequent of the rule, that is the predicted class (Robu et al, 2011).

**EXPERIMENTATION**

Since both dataset 1 and dataset2 are very similar, I will be dividing my discussion about mutation, population size, using wildcard, number of rules used, into both of these sections. In data set 1, changing mutation rate, using wildcard vs not using it, will be discussed. In the dataset 2, effect of changing population size, number of rules used, and number of generations used will be discussed.

# **Result and Parameter testing**

**Data Set 1**

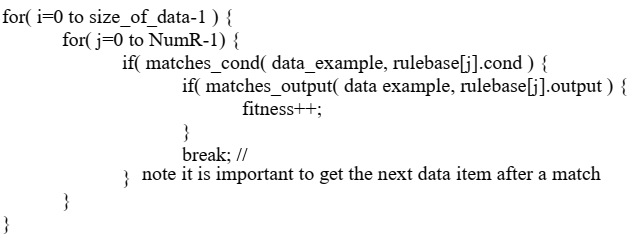
The first task is the classification problem and the approach to solve this problem is to use one of the most popular evolutionary computing algorithms, genetic algorithm. The “data1.txt” file contains 32 rows of bit strings. The first 5 bits of the row represents input variable and the 6th bit of each row represents the predicted variable as the output. The input variable is the condition and the output variable is the type. In this case the output is either 1 or 0.

10 rules are used for dataset 1 which means gene length of an individual is ((5 in the condition +1 in the output) \*10) = 60. Therefore, gene length is 60.

Since the input data is binary the individuals in the population genes are also in binary. The algorithm initializes the population of individuals with random selection of 0 and 1s.

**Fitness Function**

For the data set 1 and data set 2, fitness is scored by using the following algorithm.



As seen in the algorithm, population of individuals gene, in this case rules are compared against the conditions within each row of the data set to see if they match. When comparing rules or gene of the individual with the dataset, when they match, the individual’s fitness score will get increase. Once rule is matched with the row in the data set then the loop will break, and the algorithm will stop comparing that row of the data set with the rest of the rules in the individual so it again start from the comparison part and will try to compare another row of the data from the dataset.

**Wildcards**

**“#”** represents the **wildcard**, which is mixed with the genes of the individual during initial population creation and in mutation phase of the algorithm as without wildcards it will be very difficult to classify larger set of data and it facilitates smaller set of rules to classify much bigger set of data. Wildcard generalises the rule for instance in this case one rule may be able to match more than a row in the dataset.

In the algorithm when the fitness algorithm checks for rule bits against data set row, if a “#” wildcard encounters in the solution’s gene, it will be counted as a success regardless of what bit is in the data set row at that position. With this setup for example the now the algorithm can use a single rule to match multiple data set rows. For example, a rule in the genome may be “0##10”, it will then be able to match the following 2 rows (for example if exists) “01010” and “00110” of the dataset.

In this assignment and in the dataset 1 and 2, the algorithm executed 10 times. Then average fitness of the best fitness score of each generation is Obtained. I will be tweaking and adjusting the parameters and test out different parameters in order to identify the effect and identify optimum search parameters.

The control subject for the experiment has the following parameters: -

Mutation rate = 0.02 or 2%

Wildcard = is on.

Selection = Tournament selection

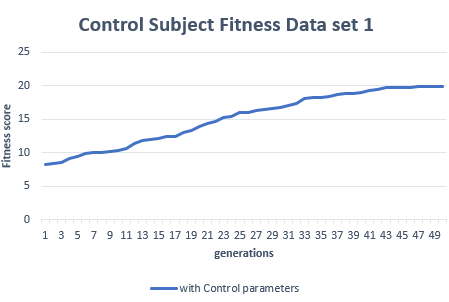
Population Size = 50

Number of Generations = 50

Number of Rules = 10

Gene length = 60

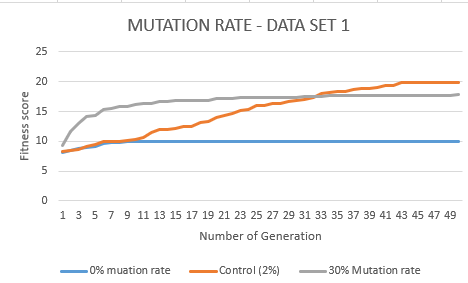
Figure 1 is the graph of control subject with the above parameters.



**Figure 1.0, Control subject with default parameters.**

The control subject is very important part of an experiment so I can compare different parameter change to the default or control subject and get a more clear picture.

As seen in the figure 1.0 graph, the rules with the mentioned default parameters, after 50 generation and population size of 50, it was only able to match only 20 data rows from the data set 1. Therefore, with the original parameters the algorithm only was able to classify about 62% of the dataset 1.

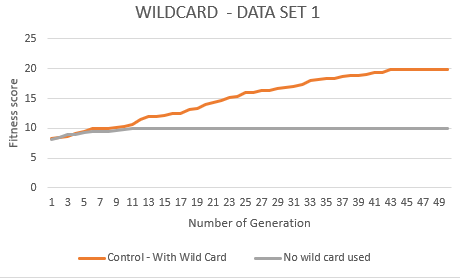


**Figure 2.0, Comparing different mutation rate, High vs Low vs Control**

Figure 2.0 show the effects of changing mutation rate. The control is again used in this one which is 2%, it shows that the fitness score of the population increases rapidly and stop at fitness score 20. Upon testing many times with different parameters, I found about 2% the control to be the best value for this task. It is because this value provides the best range for exploration of the search space by mutation operator.

When high mutation rate 30% is used, that means there is 1/3 chance of mutation, and the maximum fitness score gained was about 17 and it stayed at that rate from about 19th generation onwards. No improvement in the solutions after that mark. The reason for this is, too high mutation means individuals are mutated too often and creating too many randomness in the population. This in turns prevent population to converge to a optimum solution and it keeps searching random areas of the search space. Too high mutation rate does not satisfy our task.

When 0% mutation rate was chosen, the fitness score maxed at 10, and it did not find any further solution in the search space. This is because the population is now converged to a local optimum. NO further random exploration in the search space is possible due to mutation being 0% hence no further fitness increasement.



**Figure 3, Comparing wildcard use vs no wildcard use**

As seen in the figure 3.0, the control subject or orange line represents the using of wildcard while the grey line represents the one without the wild card. As mentioned, wild card is represented in the system using “#” symbol.

The fitness of the control subject (with wildcard) is much higher than the one without the wildcard.

In control subject where wild card is used, it shows that fitness increases in the slope then stops at 20. The one without the wildcard, shows that fitness increases for a while then stuck at fitness score 10. This is because wild card enables generalisation and allows for matching more than one rule. For instance,

If wildcard is not used in this program, suppose individuals has 60 bits, the maximum fitness score even the fittest individual can have is 10. But with wildcard even the number of rules used is much less it can match many more rows in the dataset as discussed in the above sections regarding wildcard.

**Data Set 2**

Data set 2 is very similar to data set 1, both datasets are using the same classifier program and same code for the genetic algorithm. The differences are, it has 64 rows instead of 32 and each row had 7 bits instead of 6 bits. The first 6 bit from the 7 bit is for the condition and the last bit is for the type. Therefore, now the gene length of an individual is (6+1) \*10 = 70 genes. This was the only difference and the rest is same. I will be testing the following parameters they are population size, number of rules used, and number of generations used. The mutation rate and wildcard has been discussed in the dataset1 part and needs no repetition.

The original parameters used for the control subject is as following: -

Mutation rate = 0.02 or 2%

Wildcard = is on.

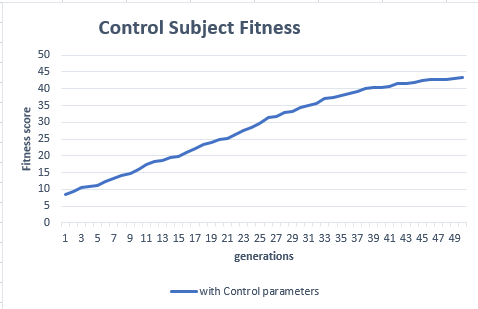
Selection = Tournament selection

Population Size = 50

Number of Generations = 50

Number of Rules = 10

Gene length = 70



**Figure 4.0, Control subject with original parameters**

As used in the dataset 1 the same parameters have been used for control subject. Only minor difference is the length of the gene size of an individual, now it is 70.

As seen in figure 4.0, as the generations are increased, the fitness also increases rapidly, and it stopped at about fitness score 44. This means with original control subject parameters as mentioned above, 44 rows were able to match from the dataset. However, when I tested with increased number of generations, I was able to get much higher fitness score. Additionally, when tested with increased population size, I was able to get higher number of fitness score. All 64 rows were able to match when the population size and number of generations increased both together to about 500. I will be discussing them below in details.

This control subject will be used to compare parameter changes in this dataset as well. The following 10 rules were the best match and was able to match with the dataset 2 rows:

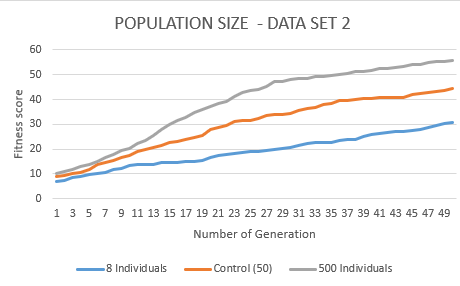
Rule 1: 000011 = 1, Rule 2: 0#1#00 = 0,

Rule 3: 10#010 = 0, Rule 4: ##0#00 = 0

Rule 5: ##001# = 0, Rule 6: 01##01 = 0

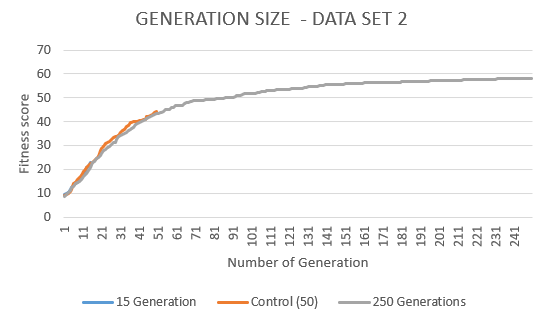
Rule 7: #11000 = 1, Rule 8: 111#11 = 1

Rule 9: ###### = 1, Rule 10: 0#101# = 0



**Figure 5.0, High population vs control subject vs low population on dataset 2.**

Figure 5.0 shows the comparison between high and low population. The sizes compared were 8, 50 and 500. As seen in the grey line with 500 population size, the greater the number of populations the higher the chance of getting fitter solutions and performance of the algorithm is much higher than those with 50 and 8 population. Since individuals represents solutions in Genetic Algorithm, the more individuals mean the more solutions in the search space. The rate of increment of the slope of the 500-population size is also much higher than the other 2. Initially three of them had very similar fitness score but as it progressed to for example 17th generation, the 500 population one gain much higher fitness score of about 35, while 23 by control subject about 13 by population size 8. The reason is there is wider selection of individual for mating pool during selection phase and number of fitter individuals is higher as the population gets bigger. One another observation is that the initial fitness of the bigger population is also higher than the low population size ones. Therefore, if the three algorithms are ran for the same short time period, the probability of getting higher and fitter individual in bigger population is more than the smaller ones for the same period of time. However, the bigger the population size the higher the resources needed by the program.

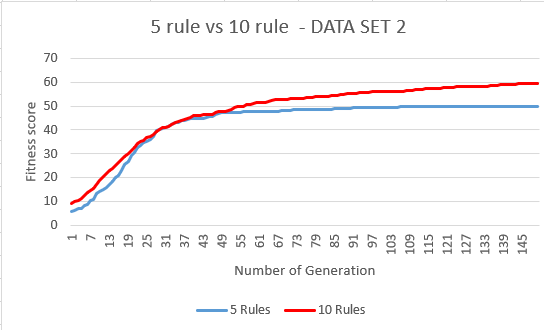


**Figure 6.0, Generation size comparison and its effect on fitness score.**

Figure 6.0 show the effect of changing generation size to fitness score. Three sizes 15 generation, control subject that is 50 generation and maximum number as 250 generation are compared. The idea is to see how much fitness is gained just by adding more generation to the algorithm.

With 15 generation, the solutions best fitness score achieved is 23. While the control subject with 50 generations achieved 44 fitness score. At 250 generation the algorithm was able to find population best fitness score as 58. The lesser the generation the fewer the fitness score. On the other hand, the more the generation means the more population go through evolution circle such as selection, crossover and mutation, hence there is more chance for getting fitter individuals from each passing generation.

At about 200th generation the solutions almost converge. It is very important to tweak the value so that the best value for generation can be obtain. In this case, at about 250th generation it converges and no need to take very high value for number of generation as it will increase the computation load.



**Figure 7.0, Comparing 5 rules to 10 rules**

5 rules were unable to match more than 50 data items in the dataset 2 file as seen in the figure 7.0. However, 10 rules were able to match about all rules in the dataset 2 file as seen in the figure 7.0. Both lines slope was very steep initially and the steepness slowed down at about 50th generation. At about 75th generation, 5 rules almost converged at fitness score 50 and the slope was stagnant from this point onwards. However, 10 rules were able to match more at this point and slowly it matches all the rows in the dataset.

The 5 rules used was:

Rule 1: 01##1# = 1

Rule 2: 0####0 = 0

Rule 3: #10### = 0

Rule 4: 10#0## = 0

Rule 5: ###### = 1

The reason why with 5 rules it was able to match so much was, because it has superfluous rules such as “###### = 1” and “0####0=0” and “#10### =0”. The “###### = 1” rule can match almost all conditions with type =1 in the data set hence it covers almost half of the data set. The remining 2 rules also have many wildcards hence they will also be able to match another data item with 0s. This is the reason with even 5 rules it was able to match 50 data rows in the data file. This happened in this case but in another case if by chance the five rules contain both “######=0 and ######=1”, they can match almost all the items in the dataset.

I can deduce from this that the bigger the dataset the lesser efficient the 5 rules compare to 10 rules because 10 rules will be searching for more in the search space.

However, by chance less number of rules may show similar behaviour as 10 rules. To make 5 rules more efficient more efficient we can tweak population size to higher number and change mutation rate.

# Conclusion

In conclusion, the genetic algorithm is a very powerful evolutionary algorithm technique which can be used in the data mining cases to classify rules in the given data. It can classify both binary values and real number values.

The effectiveness of genetic algorithm can be achieved by “fine tuning” the algorithm which is by adjusting and tweaking the parameter of the algorithm such as mutation, population size and number of generations used.

Very high mutation rate, few generations and low population size is not suitable for the algorithm as seen in the above graphs comparison. In order to fine tune the algorithm, it is better to use a control subject with original parameters, so effect of changing parameter can be better observed.

Introducing wildcard to the population increases the rate of matching by many times compare to the population without the wildcards because they allow to generalize rules. Also 10 rules vs 5 rules were observed on dataset 2. Even though 5 rules matches most of the data item, it is less efficient than 10 rules as, 10 rules can search bigger search space for solution.

# REFERENCES

Abd Rahman, R., Ramli, R., Jamari, Z. and Ku-Mahamud, K. (2016). Evolutionary Algorithm with Roulette-Tournament Selection for Solving Aquaculture Diet Formulation. *Mathematical Problems in Engineering*, 2016, pp.1-10.

Maulik, U. and Bandyopadhyay, S. (2000). Genetic algorithm-based clustering technique. *Pattern Recognition*, 33(9), pp.1455-1465.

Srinivas, M. and Patnaik, L. (1994). Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(4), pp.656-667.

Yoshida, H., Sordo, M., Vaidya, S. and Jain, L. (2008). *Advanced computational intelligence paradigms in healthcare*. Springer, p.161.

MicroStrategy. (2019). Data Mining Explained. [online] Available at: https://www.microstrategy.com/us/resources/introductory-guides/data-mining-explained [Accessed 27 Jun. 2019].

Alexander, D. (2019). *Data Mining*. [online] Laits.utexas.edu. Available at: https://www.laits.utexas.edu/~anorman/BUS.FOR/course.mat/Alex/ [Accessed 27 Jun. 2019].

Mallawaarachchi, V. (2019). *Introduction to Genetic Algorithms — Including Example Code*. [online] Medium. Available at: https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3 [Accessed 27 Jul. 2019].

Kiely, P. (2019). *Introduction to Genetic Algorithms*. [online] FloydHub Blog. Available at: https://blog.floydhub.com/introduction-to-genetic-algorithms/ [Accessed 27 Jul. 2019].

Tutorialspoint.com. (2019). *Genetic Algorithms - Parent Selection*. [online] Available at: https://www.tutorialspoint.com/genetic\_algorithms/genetic\_algorithms\_parent\_selection.htm [Accessed 28 Jul. 2019].

Gahukar, G. (2019). *Classification Algorithms in Machine Learning…*. [online] Medium. Available at: https://medium.com/datadriveninvestor/classification-algorithms-in-machine-learning-85c0ab65ff4 [Accessed 28 Jul. 2019].

Robu, Raul & Stefan, Holban. (2011). A genetic algorithm for classification. Recent Researches in Computers and Computing - International Conference on Computers and Computing, ICCC'11. 52-56.

**Click below link to find the assignment code in java which was coded using NetBeans and the graphs:**