
**FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY**

# BITI 3143 EVOLUTIONARY COMPUTING

GROUP'S PROJECT: FEATURE SELECTION USING GENETIC ALGORITHM (GA)

**LECTURER'S NAME:**

TS. DR. ZERATUL IZZAH BINTI MOHD YUSOH

**GROUP'S LEADER:**

QUEK YAO JING (B031810136)

**GROUP MEMBER:**

| | |
|---|---|
| MEOR AMIRUL ASHRAF BIN JAMALULAIL | (B031810468) |
| MUHAMAD NUR BIN MUHAMAD ZAIDI | (B031810314) |
| AKMALUNNISAK BINTI DARUS | (B031810143) |

**INDIVIDUAL TASK DISTRIBUTION**

a) Programmer (Fitness function Algorithm Designer & Programmer)

Quek Yao Jing (Leader)

- Code C++ Genetic Algorithm (GA)
- Working on Python ML code on fitness function evaluation
- Finding the best evaluation metrics for the fitness function
- Fitness function algorithm experimentation
- C++ programming coding comment
- Report writing (fitness function)

b) Documentation (Report and Analysis)

Meor Amirul Ashraf

- Algorithm design
- Graph plotting and result interpreting
- Report writing (detail description on the GA's design)

Muhamad Nur

- Graph plotting and result interpreting
- Report writing (the analysis of the result)

Akmalunnisak

- Algorithm Design
- Report writing (description of the problem to solve and instances)
- Report compilation

## DESCRIPTION OF THE PROBLEM TO SOLVE

Human Activity Recognition (HAR) attempts to acknowledge a person's behaviour through a series of impressions about themselves and the environment around them. Recognition may be determined by focus on the knowledge gathered from specific sources such as environment or body-worn sensors. It is a challenging problem because there is no clear analytical way to relate the sensor data to specific actions in a general way. It is technically challenging because of the massive volume of sensor data collected and the classical use of hand-crafted features and heuristics from these data in developing predictive models.

The Human Activity Recognition database was built from recordings of 30 study participants performing activities of daily living (ADL) while carrying a waist positioned smartphone with inertial sensors embedded in it. The goal is to identify the actions under one of the six activities that have been carried out by the human which are walking, walking upstairs, walking downstairs, sitting, standing, laying. The need for specific field expertise becomes the challenge of existing solutions. Therefore, to find an experimental planning that can optimize the algorithm for this problem, we implement the feature selection using Genetic Algorithm (GA).

## DESCRIPTION OF THE PROBLEM'S INSTANCES

Feature selection is the method of selecting the most suitable inputs for a product. These techniques can be used to identify and remove unneeded, irrelevant, and redundant features which do not contribute to or reduce the predictive model's accuracy. By using genetic algorithms, it helped the algorithm to operate on a population of individuals to produce better approximations. At each generation, a new population is created by selecting individuals according to their level of fitness in the problem domain and recombining them together using operators borrowed from natural genetics.

The offspring might also undergo bit-flip mutation to get a new solution. Meanwhile, to make sure the result was more reliable, we came out with several test plans to produce a better result. The testing plan will be using the same parameters excluded for the population sizes and maximum generation. The mutation and crossover will remain the same as it is a sensitive parameter for this project. The population size also needs to be minimum 30. Based on the testing plan, we will choose the best one and run it several times to get the analysis.

**DETAIL DESCRIPTION ON THE GA's DESIGN**

a) Chromosome Representation (Binary)

A chromosome representation is necessary to describe each attribute in the GA population. For this project, we will have 100 genes in the binary form. The binary form will use 1 and 0 to represent the availability of the feature.

b) i) Fitness function

Fitness function is one of the important evaluation metrics used to rate the fitness of the chromosome. We will need to keep track of the best fitness as this can also be categorized as Design Problem or Engineering Problem which we want to get the best fitness available and also the average fitness to understand the behavior of the algorithm. Average fitness denoted as:

$$\frac{\Sigma Fitness}{Population\ Size}$$

The problem that we are required to solve is to use a machine learning algorithm named as a support vector machine (SVM) to classify the human activity, which can be categorized into walking, walking upstairs, walking downstairs, sitting, standing, laying. In this case, this is considered as multi-class classification problems in the machine learning domain.

In our feature selection task, this can be considered as Constraint Optimization Problem (COP) in the Evolutionary Algorithm domain. We have a constraint which number of attributes cannot be lower than 2. On the other hand, we want to minimize the number of attributes while maximizing the f1-mean score. Below are some of the experimental algorithm designs.

ii) F1-Score, F1-Mean Evaluation Experimentation

There are various ways to evaluate the fitness of the algorithm based on the incoming attributes used, such as Precision, Recall, and F1-score. In this project, we will use f1-score as our fitness function, which denoted as:

$$f1 - score = 2 * \left( \frac{Precision * Recall}{Precision + Recall} \right)$$

But since we have six activities altogether, we will need to get the mean of f1-score, which denoted as:

$$f1 - mean = \frac{\Sigma f1 - score}{n.of\ attributes}$$

$$f1 - mean \left( inverted \right) = \frac{1}{f1 - mean}$$

Hence, we will be able to obtain the f1-mean of the algorithm. We then use this fitness function as the evaluation metric of our algorithm. In this algorithm, the best value of the f1-mean will be one. The smaller the value, the better the result.

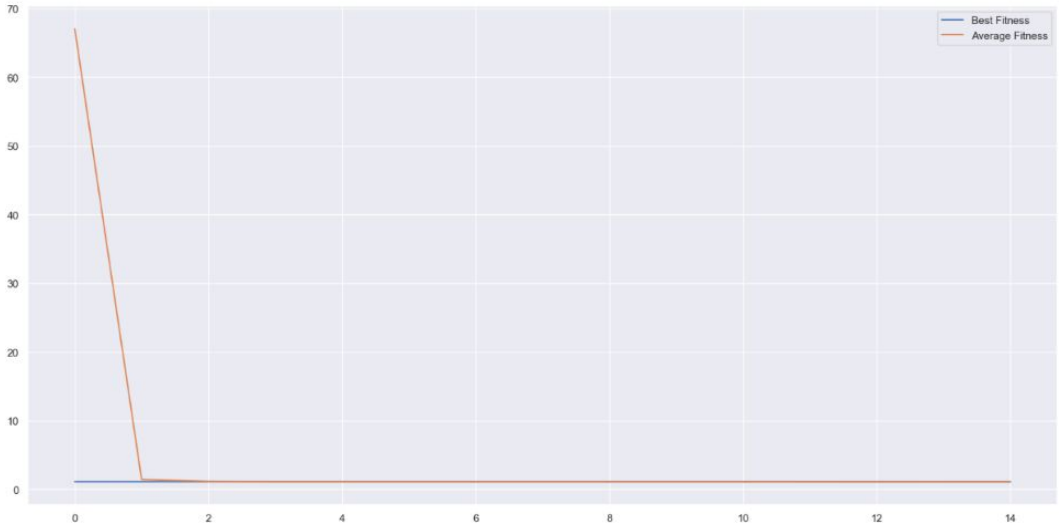



Figure 1 shows the graph and the results for running from generation 0 to 15 with f1-mean(inverted) as the fitness function.

| Gene | Population Size | Cross-over probability | Mutation Probability | Max-Generation | Max-Mutation |
|---|---|---|---|---|---|
| 100 | 30 | 0.8 | 0.2 | 15 | 3 |

Table 1 shows the parameter setting for figure above

Based on Figure 1, the training and evaluation were tested with parameters under Table 1 As we can see from the graph above, we can obtain a high score of around generation 7, and the generation afterward tends to remain the same until generation 200.

Based on the graph, we can make a few assumptions and hypotheses. Based on the observations on the cross-over and mutation frequency, the chromosome indeed tried with a lot of combination, and was able to obtain good results. So, how many minimum attributes are required to obtain the results?

Based on the experimentation, 50 attributes, and 60 attributes, both can obtain an accuracy of around 0.97. Hence, my next approaches are to minimize the attributes used when evaluating the fitness function.

iii) F1-Mean, Attributes Minimization, Weight

As discussed in (ii), F1-Mean alone is not sufficient to act as the evaluation metrics as we want to minimize the features used in machine learning. Therefore, we will need to include the details and information of the number of attributes and the weight of the importance of the variable in the algorithm. A new algorithm is used, which denoted as:

$$fitness = \left(\frac{Total\ Attributes\ used}{Total\ Attributes} * attribute\ weight\right) + \left(1 - \left(F1 - Mean * f1 - weight\right)\right)$$

In the formula above, we fixed the *attribute weight* and *f1 weight* as 0.4 and 0.6 respectively. The weight here means how vital is the variable in our proposed solution algorithm. In our case, we give more priority towards the f1-mean. By using this formula, the graph will be the difference from 4.2.1 where it will be the inversion where the lower the value, the better the results are.

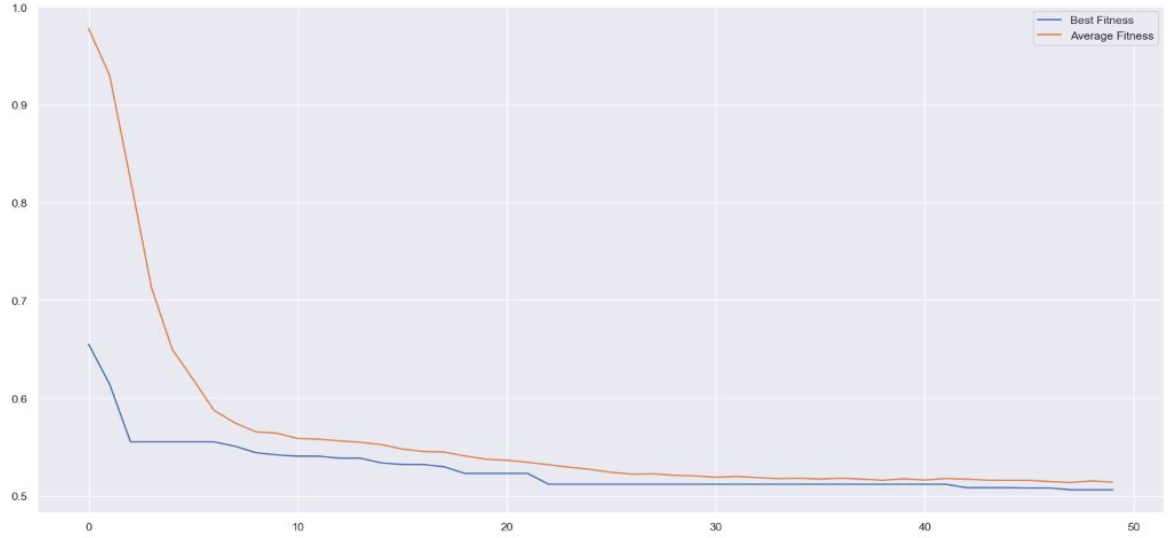



Figure 2 shows the graph and the results for running from generation 0 to 50 with the new fitness function

.

| Gene | Population Size | Cross-over Probability | Mutation Probability | Max Generation | Max Mutation |
|---|---|---|---|---|---|
| 100 | 30 | 0.8 | 0.2 | 50 | 3 |

Table 2 shows the parameter setting for figure 2

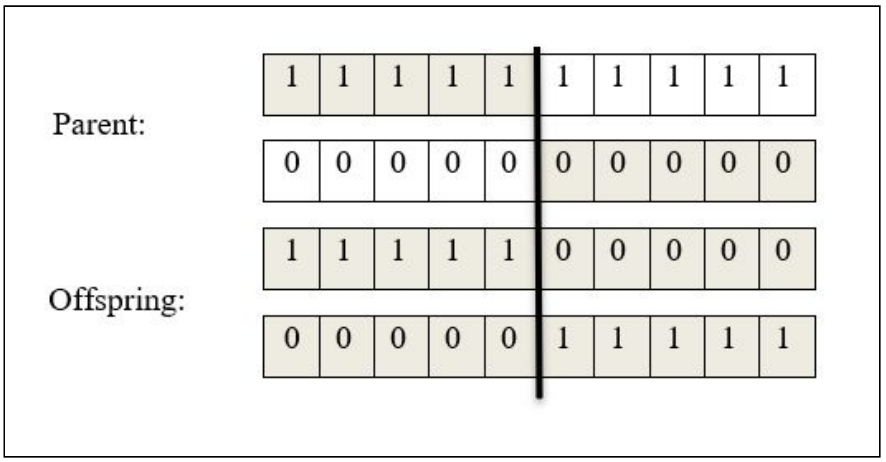
In retrospect, we found that by using the new fitness function, the number of attributes activated had significantly reduced and allowed for faster execution of the Machine Learning training time.

c) Parent selection (Tournament)

Parent Selection is the process of selecting parents which mate and recombine to create offspring for the next generation. We choose tournament selection based on the population size. The best performing chromosome will proceed the next round until there are only 2 left at the end and they will be selected as parents. This scenario will allow the best chromosome to reproduce multiple times because we want to prevent diversity.

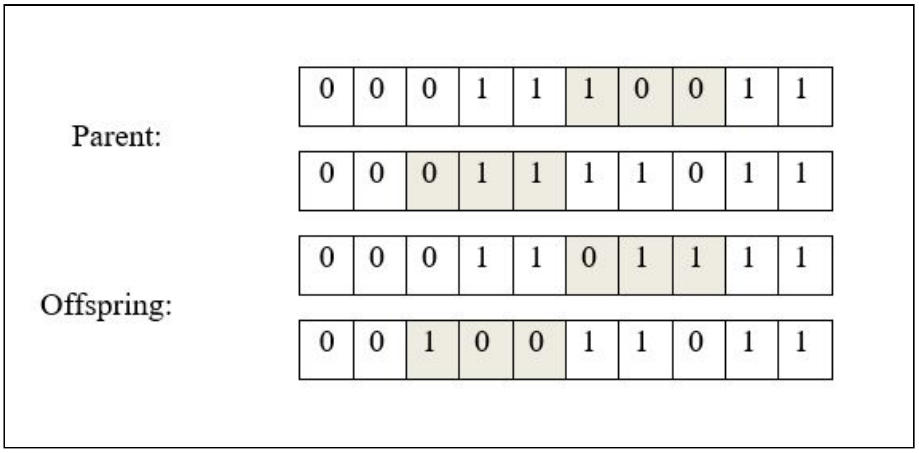
d) Cross-Over (1-point crossover)

Crossover is a genetic operator used to separate the chromosome or chromosomes programming from one generation to the next generation. In this project, we used one-point crossover to get a new offspring. By selecting a random crossover point, the first half point will be inherited from the father and the second half from the mother. The tails of these two parents will be exchanged to get a new offspring.




e) Mutation (Bit-flip)

Mutation may be defined as a small random tweak in the chromosome, to get a new solution. For this project we use bit flip mutation to select one or more random bits and flip them. This operator is the same like the crossover, but we select from random bit and crossover select from random point.




f) Termination Strategy

A genetic algorithm is run over a number of generations until the termination criteria are reached. If a termination condition has been defined for an analysis, the condition will stop the analysis. The termination criteria for this project is dependent on the maximum generation (MAX_GENERATION) parameter and will be changed several times according to the testing plan.

g) Measurement Indices

The measurement index is measured by our algorithm performance. We decided to use generation-based performance indices which are Best of Current Population (BCP) to record the best fitness value for each generation and Average of Current Population (ACP) to record the average fitness value. The BCP dgn ACP will be recorded into bestfitness.txt and averagefitness.txt that auto generate by the algorithm. The record of BCP and ACP for each run then will be illustrated in order to see clearly the evolving of the chromosome.

**RESULT ANALYSIS**

After the algorithm has been designed and implemented in the feature selection, we run the algorithm seven times with different parameters to observe and record the average and best fitness to evaluate the algorithm. Below are the parameters for the algorithm's variables.

| GENE | Population Size | Cross-over probability | Mutation probability | Max generation | Max mutation |
|---|---|---|---|---|---|
| 100 | 30 | 0.8 | 0.2 | 10 | 3 |

| GENE | Population Size | Cross-over probability | Mutation probability | Max generation | Max mutation |
|---|---|---|---|---|---|
| 100 | 30 | 0.8 | 0.2 | 20 | 3 |

| GENE | Population Size | Cross-over probability | Mutation probability | Max generation | Max mutation |
|---|---|---|---|---|---|
| 100 | 35 | 0.8 | 0.2 | 10 | 3 |

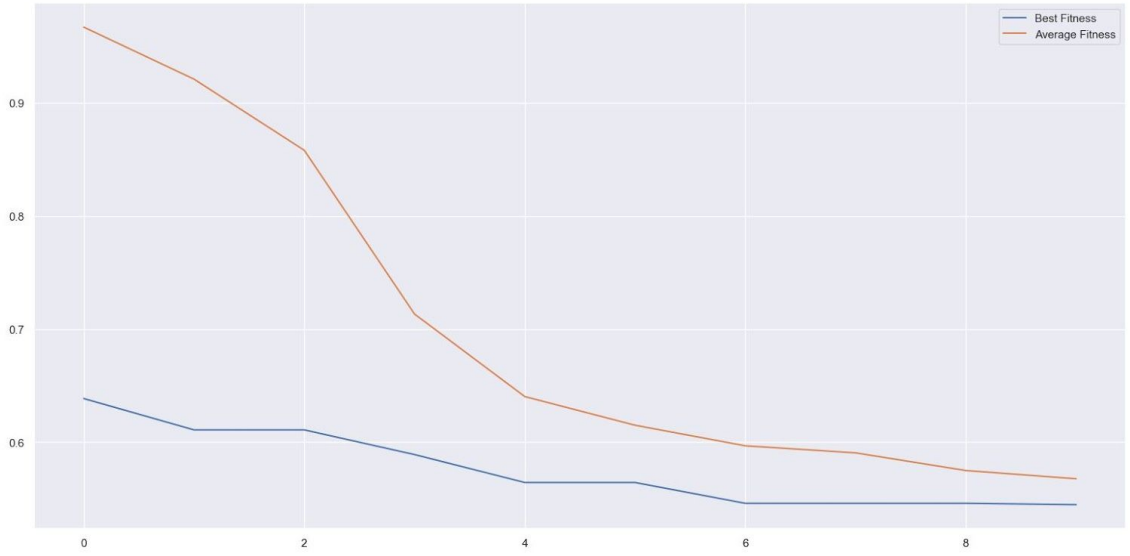
| GENE | Population Size | Cross-over probability | Mutation probability | Max generation | Max mutation |
|---|---|---|---|---|---|
| 100 | 35 | 0.8 | 0.2 | 20 | 3 |

| GENE | Population Size | Cross-over probability | Mutation probability | Max generation | Max mutation |
|---|---|---|---|---|---|
| 100 | 40 | 0.8 | 0.2 | 10 | 3 |

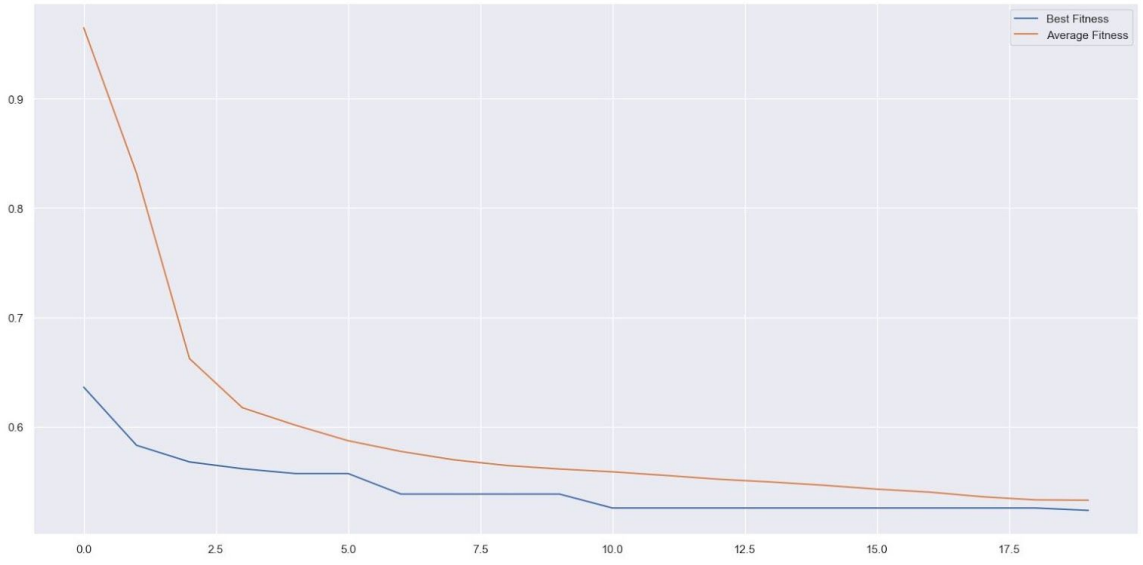
| GENE | Population Size | Cross-over probability | Mutation probability | Max generation | Max mutation |
|---|---|---|---|---|---|
| 100 | 40 | 0.8 | 0.2 | 20 | 3 |

| GENE | Population Size | Cross-over probability | Mutation probability | Max generation | Max mutation |
|---|---|---|---|---|---|
| 100 | 50 | 0.8 | 0.2 | 25 | 3 |

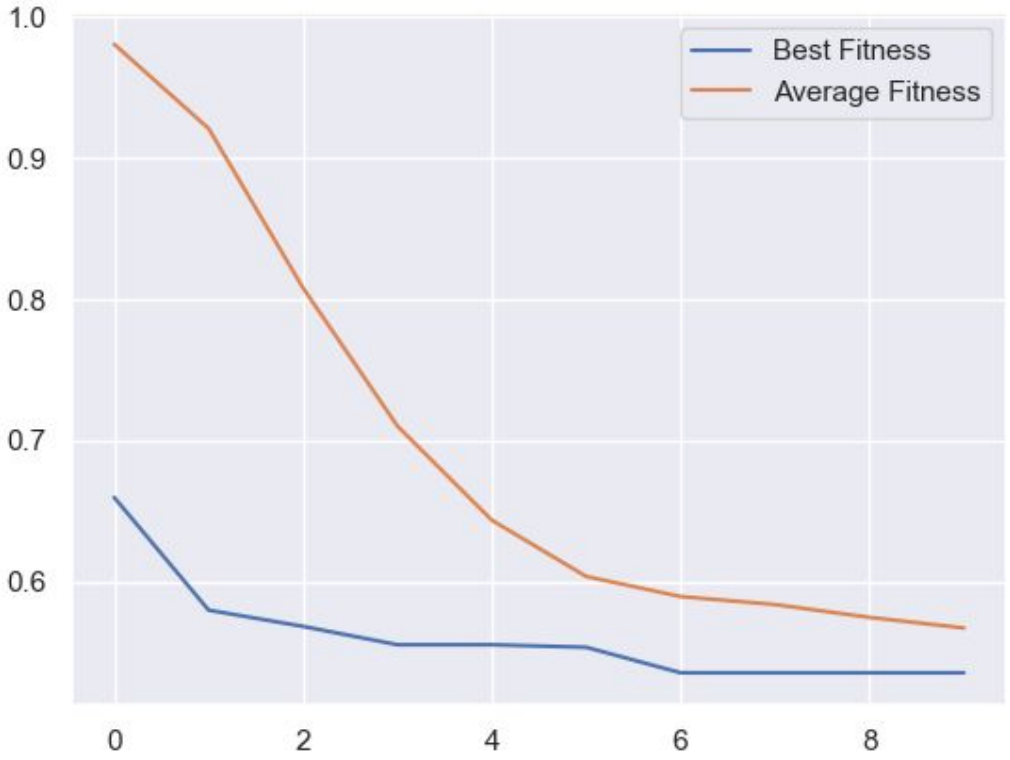
Based on the table above, we decided to keep the parameters value for gene, crossover probability, mutation probability and max mutation the same while changing the population size and max generation. Below are the plotted graphs resulting from the implementation of the algorithm that display the average and best fitness over generation.



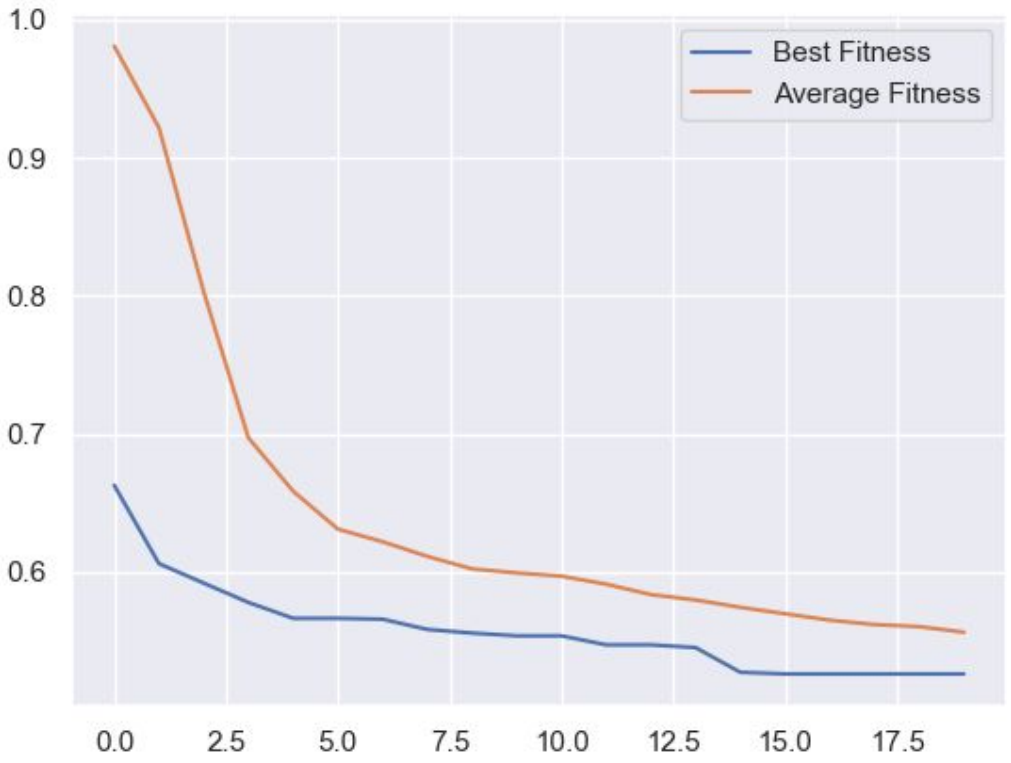

For the first run, we decided to set the parameter value for the population size and max generation to 30 and 10 respectively. The graph shows that the two lines which are the average and best fitness started to converge slowly over generation. The average fitness started to go below 0.6 and stayed between 0.5 and 0.6 fitness values after sixth generation while the best fitness already achieved below 0.6 and stayed between 0.5 and 0.6 before third generation.



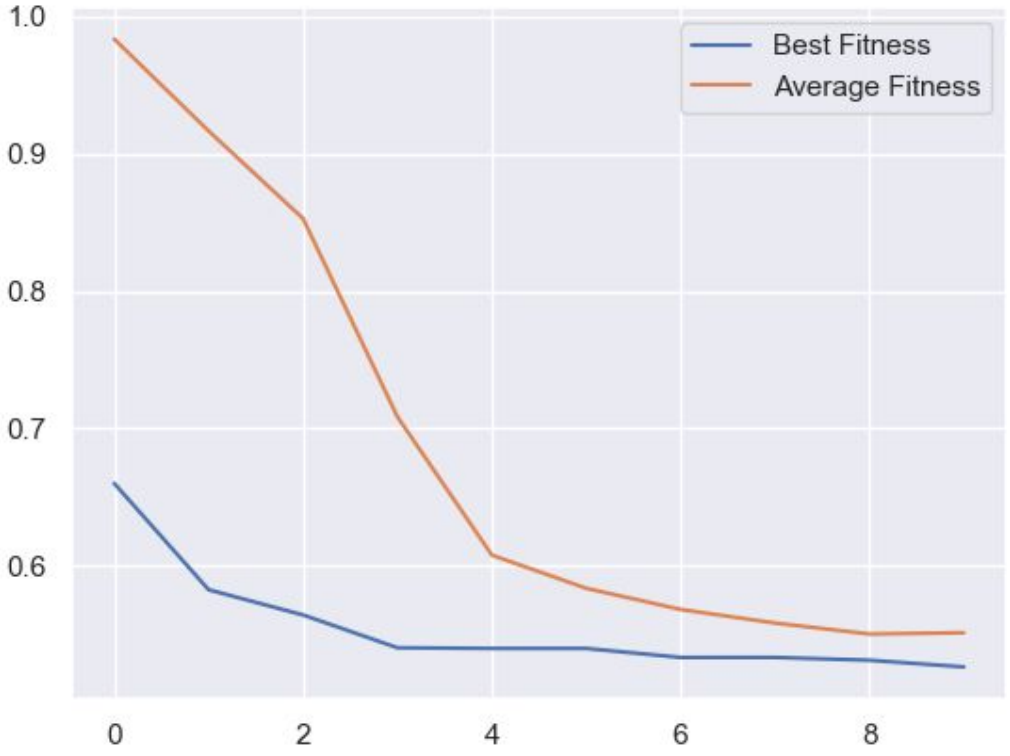
On the second run, we only increase the max generation which is from 10 to 20. The result that we get is the average fitness started to go below 0.6 and stay between 0.5 and 0.6 after the third generation while the best fitness is achieved below 0.6 and stayed between 0.5 and 0.6 before the second generation. The result gained is better compared to the first run because the best fitness is achieved by the earlier generation.



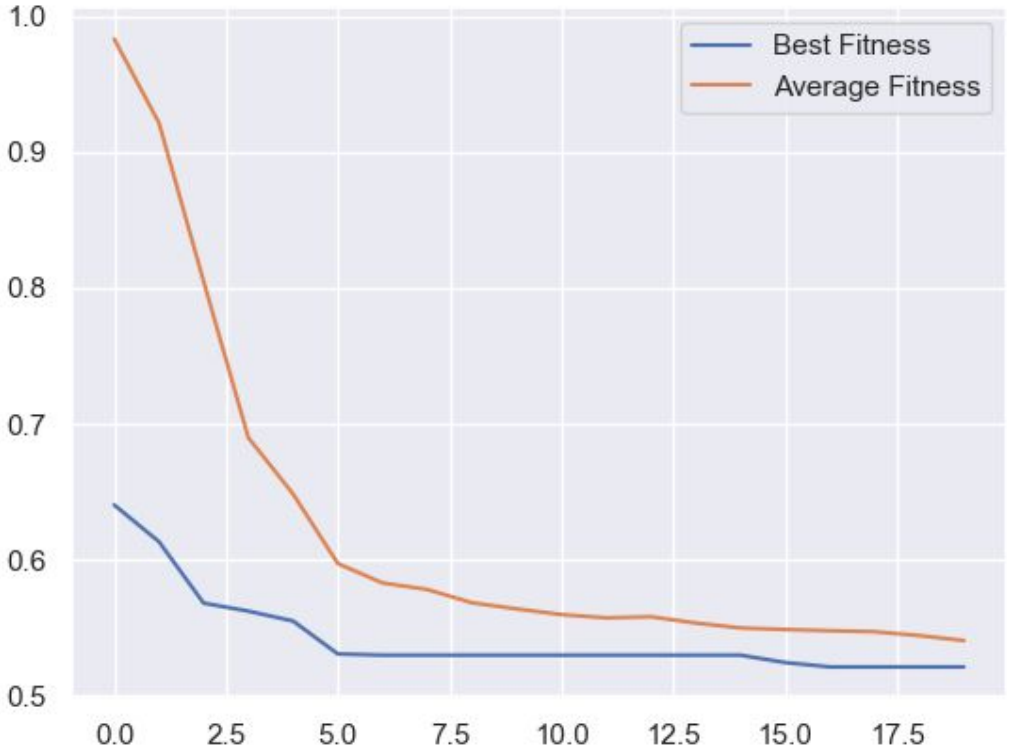

On the third run, we increased the population size to 35 and decreased the max generation back to 10 like the first run. We achieved the result of the average fitness started to go below 0.6 and stay between 0.5 and 0.6 after fifth generation while the best fitness already achieved below 0.6 and stayed between 0.5 and 0.6 before the first generation. The result is not good compared to the previous run which is the average and best fitness achieved by the earlier generation. This result shows that increasing the population size and decreasing the max generation will not give better results.



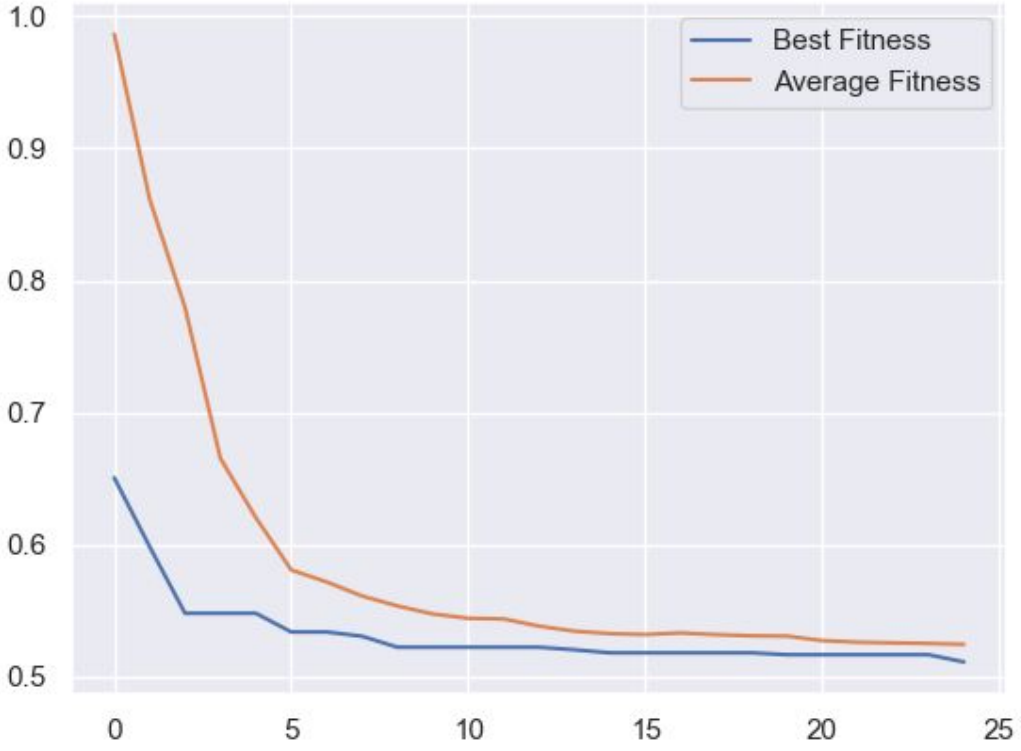

For the fourth run, we did not change population size and only increased the max generation to 20. We achieved the best fitness between 0.5 and 0.6 before the second generation while the average fitness stayed between 0.5 and 0.6 after the tenth generation. The result was fine. It is not good or bad compared to the previous run but it still can be improved.




On the fifth run, we increased the population size to 40 while decreasing the max generation to 10 back. The result that we get is better compared to the fourth run. The best fitness stayed between 0.5 and 0.6 before the first generation while the average fitness is before the fifth generation. At this point, it is not proven that increasing the population size or max generation will lower the value of the fitness below 0.5 but it somehow affects the time given for the algorithm to achieve best or average fitness by earlier generation.




For the sixth run, we did not change the population size and increase the max generation to 40. The line for average fitness stayed between 0.5 and 0.6 after the fifth generation while the best fitness before the second generation. As mentioned before, the fitness value will not go below 0.5 no matter how we change the value for population size or max generation.




On the seventh or final run, we decided to increase the population size by 10 which is from 40 to 50 and the max generation by 5 which is from 20 to 25. The result is still the same which is the average and best fitness will stayed between 0.5 and 0.6 and not get lower than that. It seems that the algorithm managed to achieve the best and average fitness between 0.5 and 0.6 from the fifth generation since the fifth run. It means that the parameter that being used on the fifth run might be the minimum value needed for the algorithm to achieve the best and average fitness between 0.5 and 0.6 and earlier than the fifth generation.

Based on the observation on all the runs that we did, it seems that changing the population size and max generation might not give the value for the best and average fitness lower but make the algorithm give the result by earlier generation. Changing the max generation might give better result since the algorithm is given more chance to evolve and produce better result.