Assignment 2
Feature Selection

Artificial Intelligence

Course Instructor: Shahela Saif

Submitted by:

Haadiya Sajid	Musa Haroon
21i-1216	21i-1206

Section: P (BS-SE)

Date: 31/03/2024

Spring 2024

Department of Software Engineering

FAST – National University of Computer & Emerging Sciences

Islamabad Campus

Table of Contents

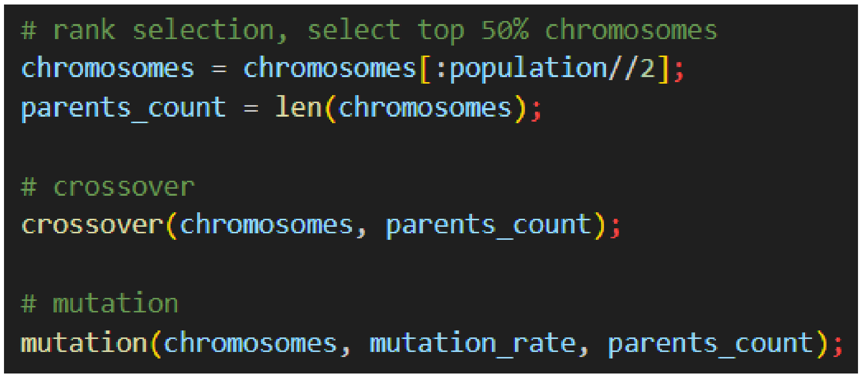
Genetic Algorithm

Chromosomes

The chromosome representation in this Genetic Algorithm (GA) is a binary string, where each bit represents the presence (1) or absence (0) of a feature in the dataset. The length of the chromosome is equal to the number of features in the dataset.

Genetic Operators

The GA uses two genetic operators: crossover and mutation. Prior to applying these, selection guides the application of genetic operators by determining which individuals contribute to the next generation.



```
# rank selection, select top 50% chromosomes
chromosomes = chromosomes[:population//2];
parents_count = len(chromosomes);

# crossover
crossover(chromosomes, parents_count);

# mutation
mutation(chromosomes, mutation_rate, parents_count);
```

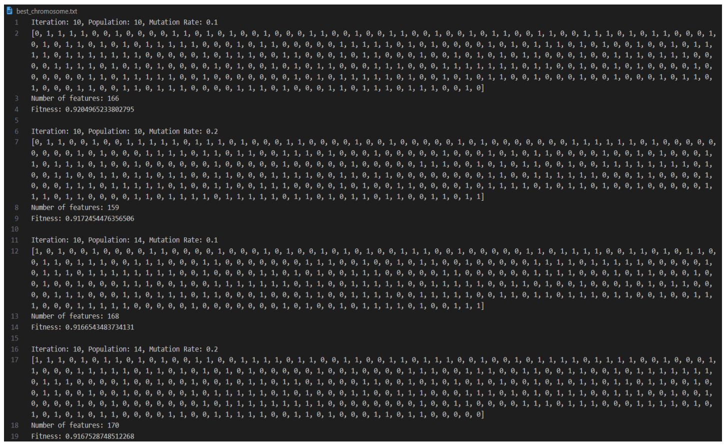
GA.*py*

- The **selection** takes place to determine which chromosomes are eligible to be parents for the next generation. This is done on the basis of **rank selection**, whereby the top 50% fittest chromosomes are selected to be parents.
- The **crossover** function in **`operators.py`** takes two parent chromosomes and combines them to create **two** new child chromosomes. It does this by selecting a random crossover point and then concatenating the first part of the first parent with the second part of the second parent for the first child, and vice versa for the second child.
- The **mutation** function in **`operators.py`** introduces random changes in the chromosome. It does this by selecting a random index in the chromosome and flipping its value (0 to 1 or 1 to 0). The **mutation rate** determines the probability of a mutation occurring. By introducing mutation in the chromosomes, we are greatly reducing the probability of getting stuck at a local maxima in terms of fitness.

Feature selection

The feature selection process is performed in the **`GA.py`** file. The Genetic Algorithm (GA) is used to select the best features for the model. The impact of feature selection on model performance can be seen in the fitness values of the chromosomes (sets of features). The chromosomes with higher fitness values are selected for the next generation, which means they contribute to a better model performance.

We ran the GA on different parameters, changing **mutation rate** and **population size** in each run. Generally, we would select the minimum number of features which are giving us the highest accuracy. In this case, the second-lower number of features (166) is giving us the highest accuracy **(0.9204)**.

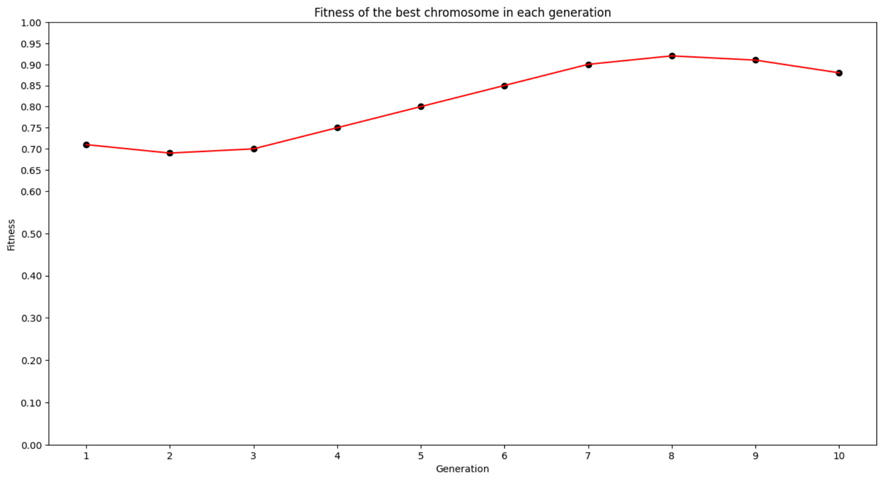


```
Iteration: 10, Population: 10, Mutation Rate: 0.1
[0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1,
0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1,
1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0,
0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0,
1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0]
Number of features: 166
Fitness: 0.9204965233802795

Iteration: 10, Population: 10, Mutation Rate: 0.2
[0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1,
1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1,
0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,
1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1]
Number of features: 159
Fitness: 0.9172454476356506

Iteration: 10, Population: 14, Mutation Rate: 0.1
[1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0,
0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1,
0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0,
0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0,
0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1,
1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1]
Number of features: 168
Fitness: 0.9166543483734131

Iteration: 10, Population: 14, Mutation Rate: 0.2
[1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1,
1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0,
0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1,
0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0,
1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0]
Number of features: 170
Fitness: 0.9167528748512268
```

The features selected by each run of the GA are represented by the chromosome variable in the **`GA.py`** file. The best chromosome after each run is saved in the **`best_chromosome.txt`** file by the **`save_best_chromosome()`** function from the **`processing.py`** file, which also saves information about the iterations, population size, and mutation rate that were used to obtain the best chromosome.

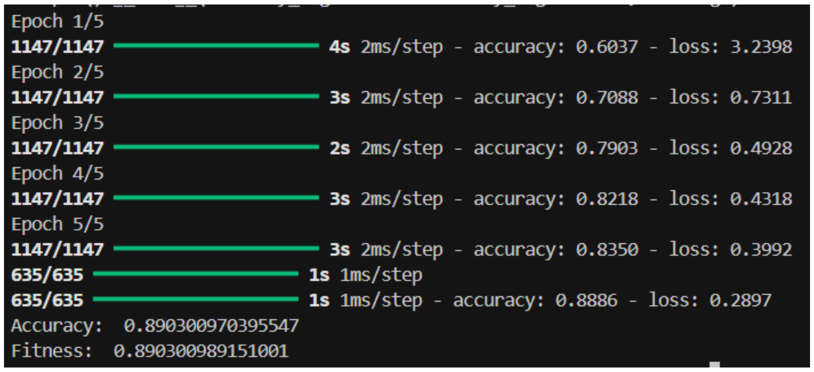
Graphical Representation

After selecting the best run, we then plotted the graph of the fitnesses of the best chromosomes in each generation of that run using **matplotlib**



Performance Evaluation

After performing feature selection, we got the features that give us the best accuracy. We then trained a model on those features and evaluated its performance.



```
Epoch 1/5
1147/1147 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.6037 - loss: 3.2398
Epoch 2/5
1147/1147 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.7088 - loss: 0.7311
Epoch 3/5
1147/1147 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.7903 - loss: 0.4928
Epoch 4/5
1147/1147 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8218 - loss: 0.4318
Epoch 5/5
1147/1147 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8350 - loss: 0.3992
635/635 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step
635/635 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.8886 - loss: 0.2897
Accuracy:  0.890300970395547
Fitness:  0.890300989151001
```

Terminal after training the model

Fitness

The fitness value of the model is also evaluated by **`model.evaluate(X_test, y_test).`** **`X_test`** is the dataset on which we test the model and **`y_test`** are its labels.

Accuracy

The accuracy of the model is calculated by comparing the predicted labels with the actual labels. The predictions were calculated by **`model.predict(X_test)`**. These predictions were then categorized to 1 and 0 depending on whether the prediction was **< 0.5** or not, and then matched with **`y_test`** to calculate the accuracy.

The performance evaluation results for the final model are presented in the **`model.py`** file.