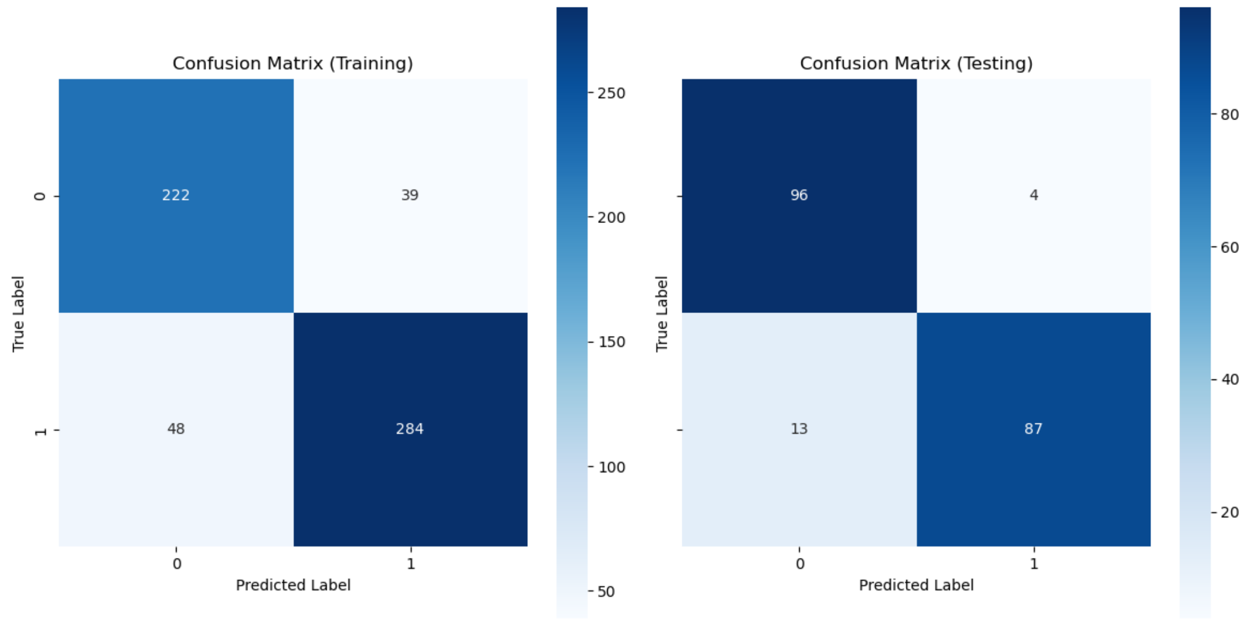
Mandar Angchekar: 386916341
Evolutionary Machine Learning – HW04

Particle Swarm Optimization for Neural Network Weight Training

A feedforward neural network with one hidden layer of 20 neurons was trained. The PSO algorithm was employed to optimize the weights and biases of the neural network. Each particle in the swarm represented a potential solution to the problem, which is a specific configuration of the neural network weights and biases.
The following parameters were utilized for the PSO optimization:

- **Cognitive Coefficient (c1):** 1.7
- **Social Coefficient (c2):** 1.3
- **Inertia Weight (w):** 0.4
- **Number of Particles:** 30
- **Number of Iterations:** 100

Particle Swarm Optimization Algorithm Confusion Matrix: HW04



The confusion matrices indicate that the model has learned to classify the majority of the instances correctly. However, there are more false negatives than false positives in the training set, and vice versa in the testing set.

Conclusion (PSO):
PSO proved to be a viable method for optimizing the weights and biases of a neural network. The optimal values led to a neural network that performed with high accuracy on

the Wisconsin Breast Cancer dataset. A more extensive hyperparameter search could be conducted to further improve the model's generalization capability.

Detailed Comparison of the Five Algorithms

1. **Backpropagation (HW00)**
2. **Evolutionary Strategies (HW01)**
3. **Genetic Algorithm (HW02)**
4. **Differential Evolution (HW03)**
5. **Particle Swarm Optimization (HW04)**

Each algorithm has its strengths and weaknesses:

- Evolutionary Strategies stand out for their high sensitivity but struggle with specificity due to higher false positives.
- Genetic Algorithm performs well across all metrics, showing a particularly strong ability to identify Class 0 instances.
- Differential Evolution demonstrates strong overall accuracy with balanced performance on both sensitivity and specificity.
- Backpropagation and Particle Swarm Optimization show very similar patterns of performance, with high accuracy and a balanced approach to sensitivity and specificity.

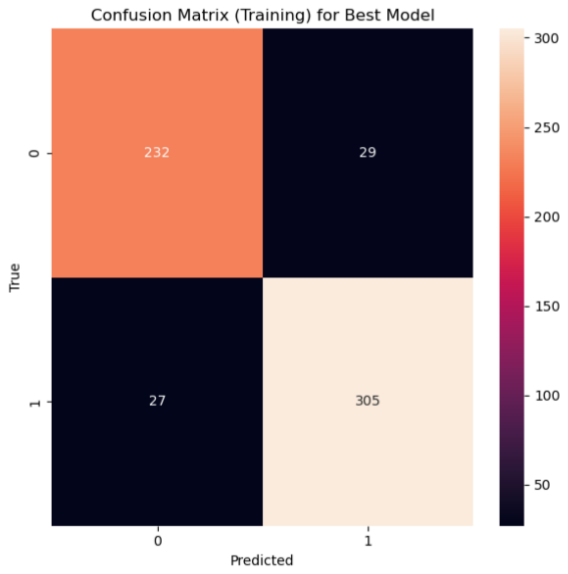
Choosing the "best" algorithm depends on the specific requirements of the task. For instance, if minimizing false negatives is crucial (such as in the dataset I used i.e. medical diagnoses where missing a positive case can be very detrimental), Evolutionary Strategies or Genetic Algorithm might be preferred. For general use where a balance is needed, Differential Evolution might be the most suitable choice.

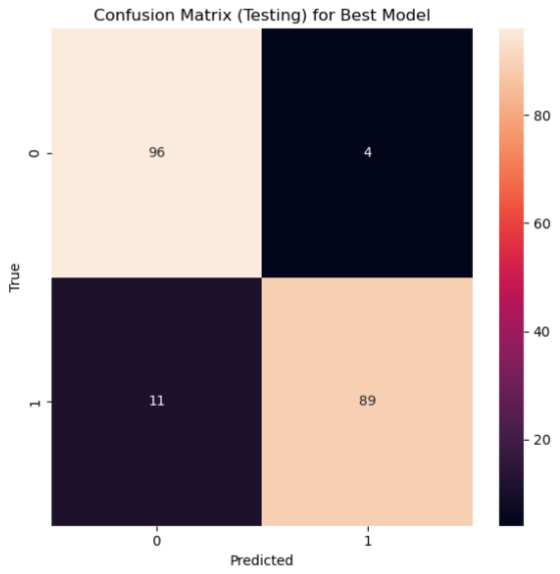
Finally, it's important to note that while training performance is informative, the testing performance is a better indicator of how the algorithms would perform on unseen data. Evolutionary Strategies and Genetic Algorithm show strong testing performance, which suggests good generalization from training to testing.

The comparison of the algorithms is based on their performance metrics derived from confusion matrices for both training and testing datasets displayed on the next pages:

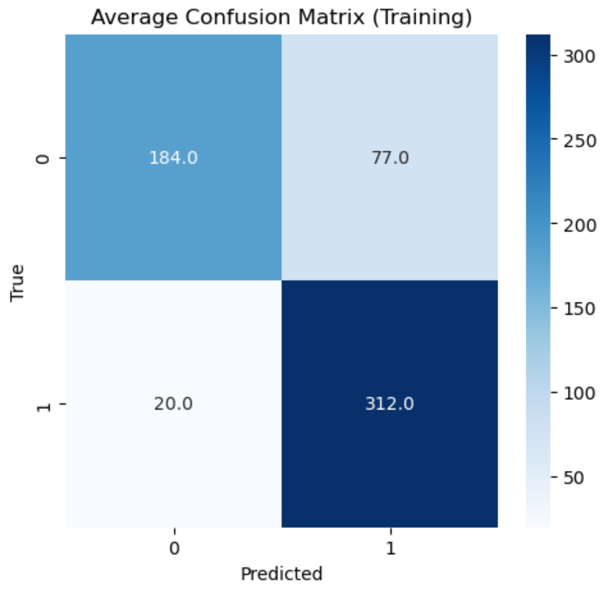
Mandar Angchekar: 386916341
Evolutionary Machine Learning – HW04

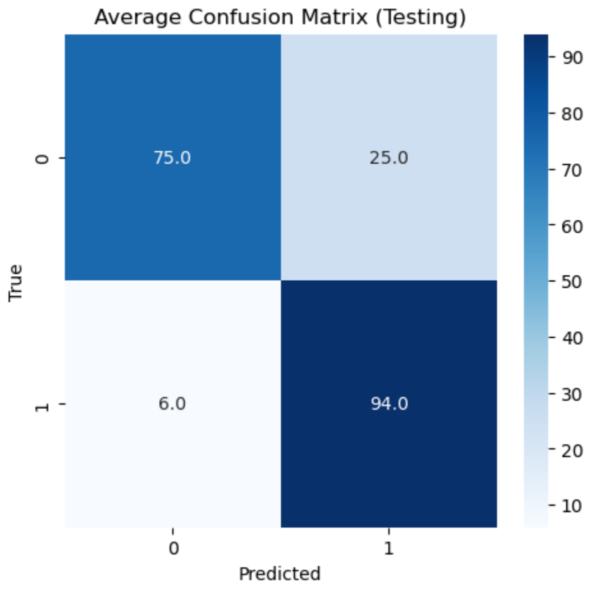
Backpropagation Algorithm Confusion Matrix: HW00





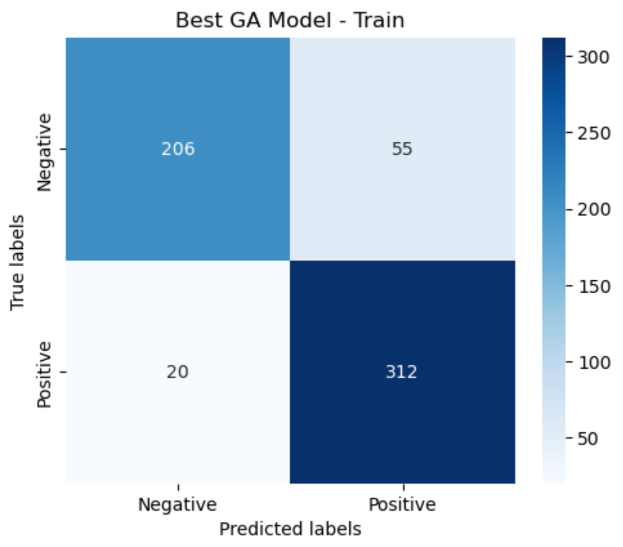
Evolutionary Strategies Confusion Matrix: HW01

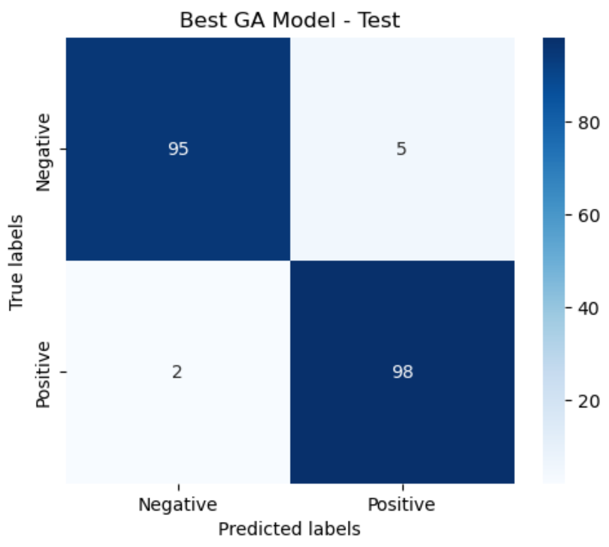




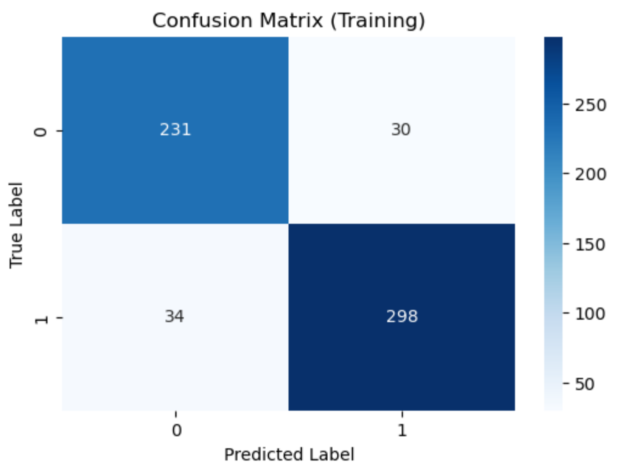
Mandar Angchekar: 386916341
Evolutionary Machine Learning – HW04

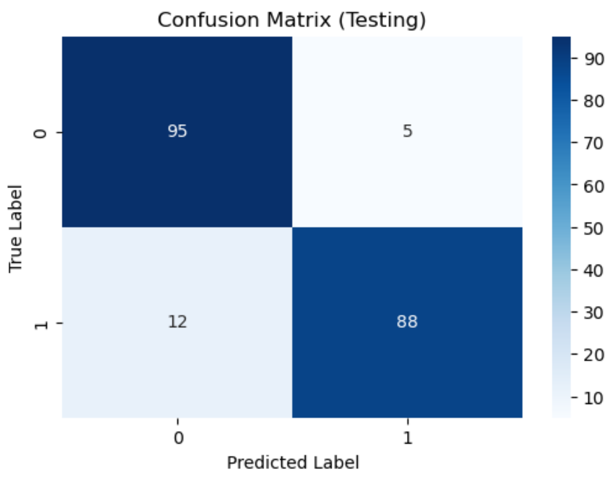
Genetic Algorithm Confusion Matrix: HW02





Differential Evolution Algorithm: HW03


Confusion Matrix (Training)
True Label
Predicted Label


Confusion Matrix (Testing)
True Label
Predicted Label