ESKİŞEHİR OSMANGAZİ ÜNİVERSİTESİ

MÜHENDİSLİK MİMARLIK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

INTRODUCTION TO HEURISTIC ALGORITHMS
FINAL PROJECT

Group-6
Ferdi İslam Yılmaz-152120191055
Yunus Emre Karagöz-152120191051
Ömer Faruk Ün-152120191001
Yiğit Efe Çoşkun-152120191093

1.Introduction

Machine learning algorithms are effective in tackling complex problems and optimizing their performance often requires careful tuning of hyperparameters. This project focuses on improving the hyperparameter selection of machine learning algorithms such as AdaBoost, SVM, Random Forest, Decision Tree, Logistic Regression, and Gradient Boosting using optimisation algorithms. The optimisation algorithms we have chosen are as follows: Whale Optimisation Algorithm, Grey Wolf Optimiser and Differential Evolution. By leveraging these advanced optimisation techniques, the research aims to improve the efficiency and effectiveness of machine learning models in tackling complex challenges in various domains.

2. Description of Algorithms

2.1. Optimization Algorithms

2.1.1. Grey Wolf Optimization

The Grey Wolf Optimization algorithm is inspired by the social hierarchy and hunting mechanisms of grey wolves. This metaheuristic algorithm mimics the collaborative behavior of wolf packs to find optimal solutions in multi-dimensional search spaces.

2.1.2. Whale Optimization

Whale Optimization is inspired by the hunting behavior of humpback whales. It is a nature-inspired algorithm that utilizes the social interactions and collective intelligence observed in whale groups to navigate complex search spaces and discover optimal solutions.

2.1.3. Differential Evolution

Differential Evolution is a stochastic, population-based optimization algorithm that iteratively explores the solution space. It employs mutation, crossover, and selection operations to evolve a population of candidate solutions toward the optimal configuration.

2.2. Machine Learning Algorithms

2.2.1. Adaboost

Adaboost (Adaptive Boosting) is an ensemble learning algorithm that aims to create a strong learner by combining weak learners. It sequentially trains weak classifiers (usually those with low success rates) and adjusts their weights by focusing on their errors.

For the AdaBoost algorithm, hyperparameters such as number of estimators, learning rate and algorithm play an important role. Number of estimators determines the number of weak learners in it and affects the complexity of the model. Using more learners can often improve performance, but may increase the training time. Learning rate controls the magnitude of each learner's influence on the previous version of the model. A low learning rate allows for more stable learning. The Algorithm parameter determines the weak learner training algorithm used. 'SAMME.R' generally performs better on models that work with probabilities. Trial and error is often used to find the optimal values of these hyperparameters. The accuracy without hyperparameter selection is 86.8%.

2.2.2. SVM (Support Vector Machine)

Support Vector Machine is a classification and regression algorithm used to classify data points into two classes. It determines the bounding hyperplane and classifies data points according to this plane.

Important hyperparameters for the Support Vector Machine (SVM) algorithm can be described as follows. The C parameter controls the regularization strength and is used to balance the cost between classification error and decision boundaries. The kernel parameter determines the function used to project data points into high-dimensional spaces. The Gamma parameter controls the domain of kernel types and affects the trade-off between overfitting and generalization. The Degree parameter is used for the "poly" kernel type and determines the polynomial degree. Higher degrees can lead to more complex decision constraints but can cause overfitting. Careful tuning of these hyperparameters affects the performance of the SVM model and trial and error methods are often used to find optimal values. The accuracy without hyperparameter selection is 88,7%.

2.2.3. Random Forest

Random Forest is a tree-based ensemble learning algorithm. Multiple decision trees are generated and their results are combined to obtain a more accurate and generalizable model.

The important hyperparameters used for the Random Forest algorithm can be described as follows: The Number of estimators parameter determines the number of trees in the forest, and generally more trees provide better generalization performance, but can increase the risk of overtraining. The Criterion parameter determines the criterion that measures the quality of node splits and can be set to "gini" or "entropy". The Max depth parameter limits the maximum depth of each tree, which is important in controlling the complexity of the model. Min samples split parameter determines the minimum number of samples required to split a node, it can help prevent overfitting. Min samples leaf parameter determines the minimum number of samples of a leaf and can contribute to make the model more generalizable. Max features parameter controls the maximum number of features to consider at each split point, usually preferred as "sqrt" or "log2". Setting these hyperparameters correctly can improve the performance of the Random Forest model, and various trial and error methods are often used to find the optimal values. Accuracy without hyperparameter selection is 98.5%.

2.2.4. Decision Tree

Decision Tree is a tree structure used in classification and regression tasks. It reaches the result by making decisions according to the features in the dataset.

The important hyperparameters used for the Decision Tree algorithm can be explained as follows: Criterion parameter determines the criterion used for splitting a node. Splitter controls how the nodes are split. Max depth limits the maximum depth of the tree, important to avoid overfitting. Min samples split determines the minimum number of samples needed to split a node. Min samples leaf determines the minimum number of samples of a leaf. Setting these hyperparameters appropriately affects the performance of the Decision Tree model. Trial and error is often used to find optimal values. Accuracy without hyperparameter selection is 97.0%.

2.2.5. Logistic Regression

Logistic Regression is a statistical method that examines the effect of independent variables on one or more dependent variables. It is used in classification problems and gives a probability value between 0 and 1 through the output sigmoid function.

Some important hyperparameters used for the Logistic Regression algorithm can be explained as follows: Penalty determines the type of regulation. C controls the strength of the regularization. Higher values of C mean less regularization and can lead to a better fit of the model to the training data. Solver determines the algorithm used to solve the optimization problem. Setting these hyperparameters correctly can improve the performance of the Logistic Regression model. Various trial and error methods are often used to find the optimal values.Accuracy without hyperparameter selection is 80.9%.

2.2.6. Gradient Boosting

Gradient Boosting is an ensemble learning algorithm that builds a strong model by combining weak learners. It works by minimizing the gradient on the error function.

The important hyperparameters used for the Gradient Boosting algorithm can be described as follows: Learning rate controls the magnitude of the contribution of each tree addition. Number of estimators determines the number of weak learners (usually decision trees) to be used. Max depth limits the maximum depth of each tree, important to control overlearning. Min samples split determines the minimum number of samples needed to split a node. Min samples leaf sets the minimum number of samples of a leaf. If the learning rate is low, the model may be more robust and have high generalization ability, but it may be necessary to add more trees. Other parameters can also affect the complexity of the model and the training time, preventing overfitting. Various trial and error methods are often used to find the optimal values of these hyperparameters.Accuracy without hyperparameter selection is 95.6%.

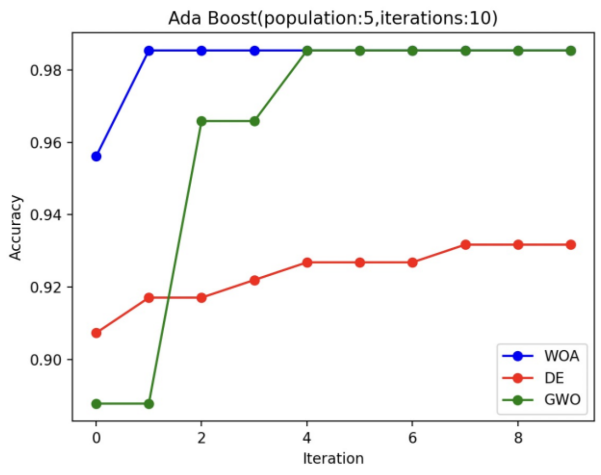
3. Dataset

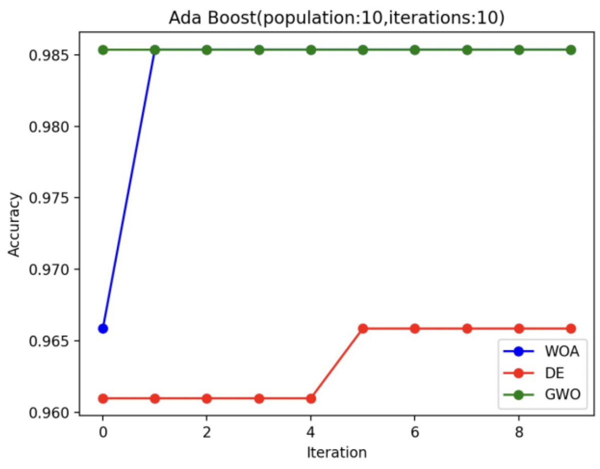
In our project, we used a heart disease dataset with 14 attributes, which contain various information about the health status of the patients. The dataset contains the labels that will be used for the learning process. The "target" field refers to the presence of heart disease in the patient. It is integer valued 0 = no disease and 1 = disease. In our project, we trained 6 different models for prediction.

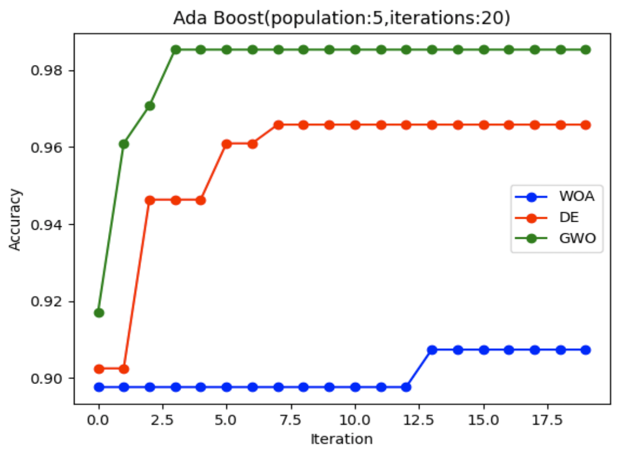
4. Results

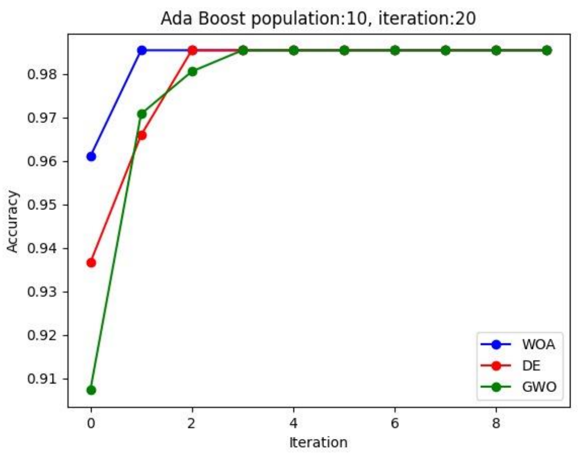
4.1. Adaboost Hyperparameter Tuning

For the Adaboost model we chose the hyperparameters as n_estimators, learning_rate and algorithm. The dimension was set to three.We ran each optimization algorithm at 4 different population sizes and iterations and generated the following tables.



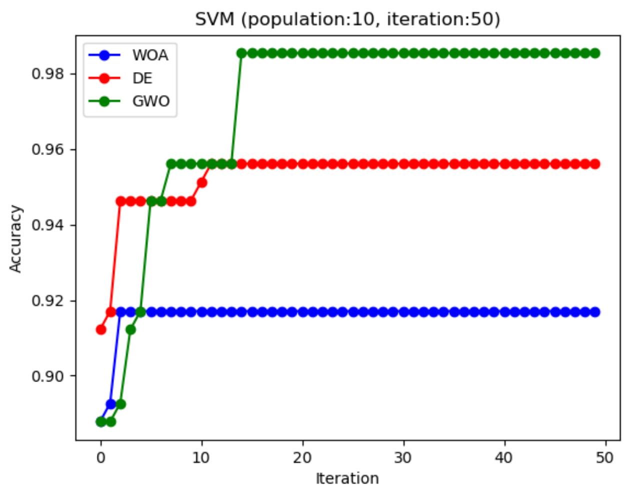


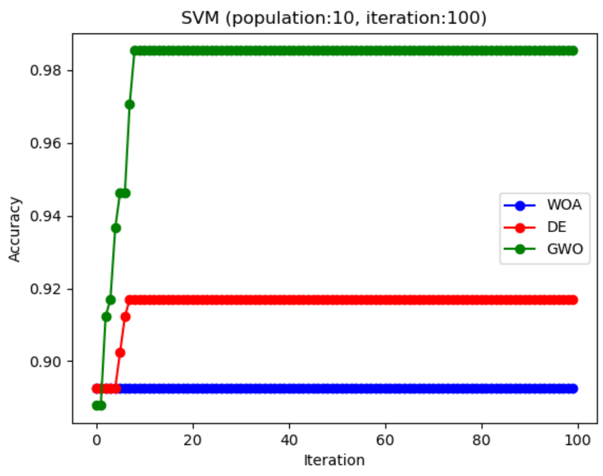


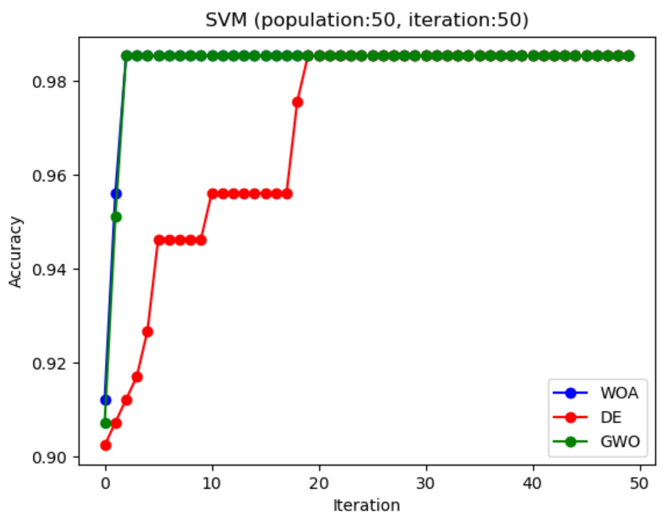


4.2. SVM Hyperparameter Tuning

For the SVM model, we chose the hyperparameters C, kernel, gamma and degree. The dimension was set to four. We ran each optimization algorithm at 3 different population sizes and iterations and generated the following tables.

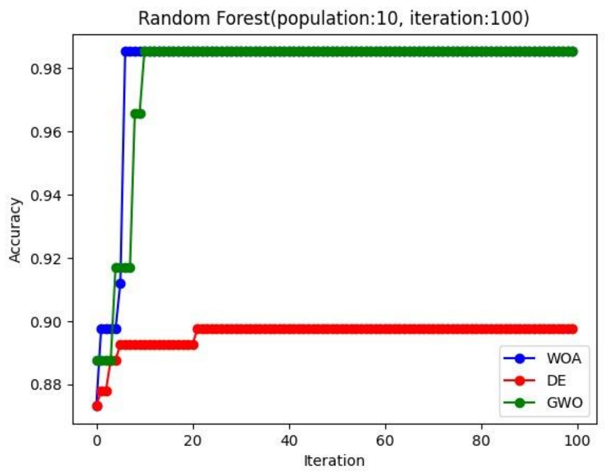


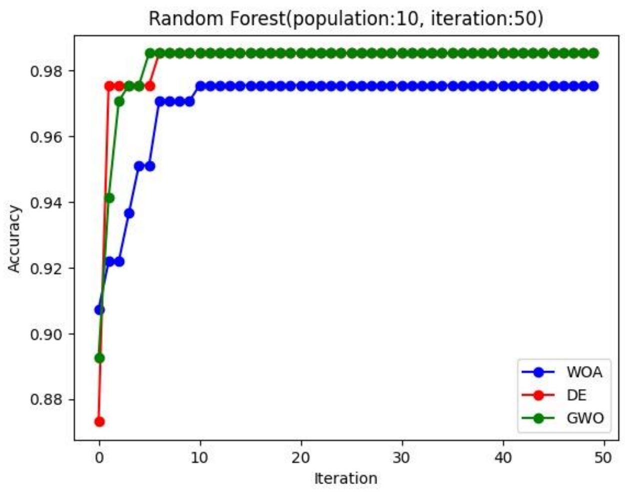


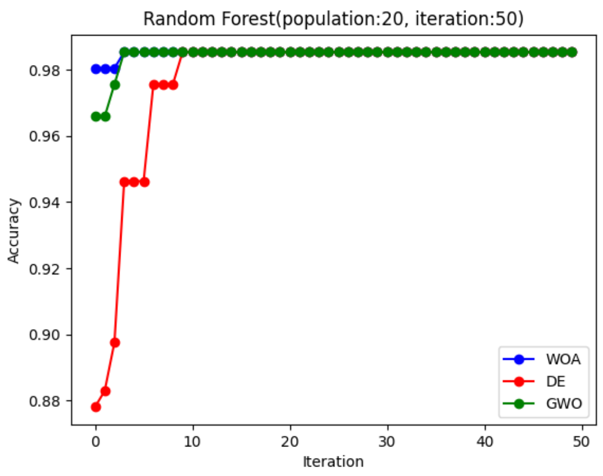


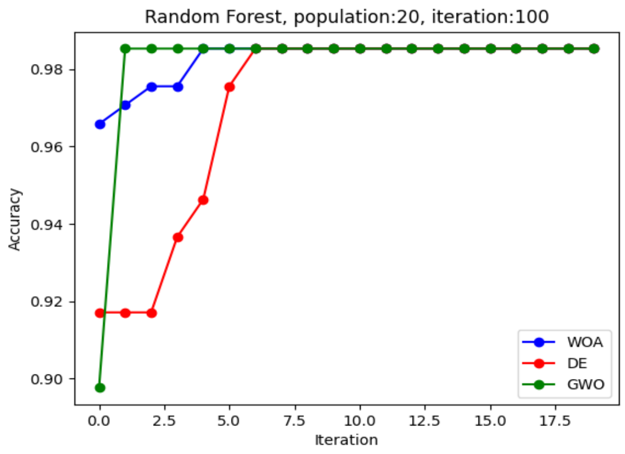
4.3. Random Forest Hyperparameter Tuning

For the random forest model, we chose the hyperparameters as n_estimators, criterion, max_depth, min_samples_split, min_samples_leaf and max_features. The dimension was set to six.We ran each optimization algorithm at 4 different population sizes and iterations and generated the following tables.



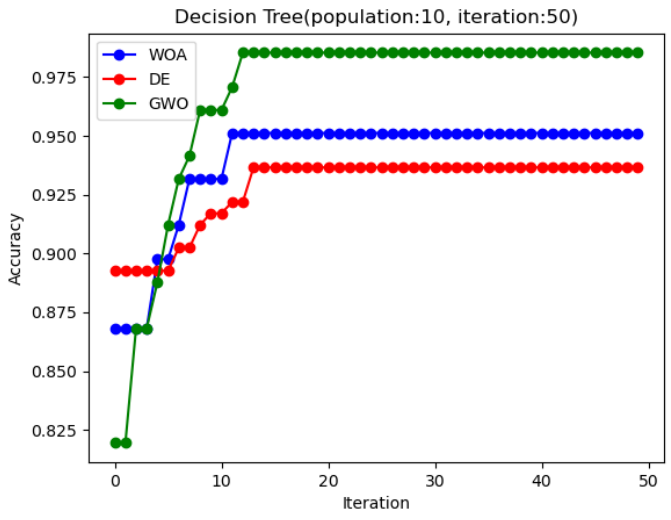


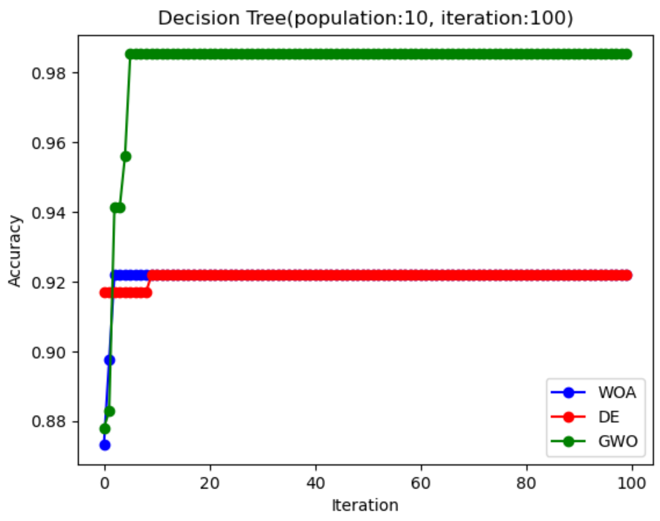


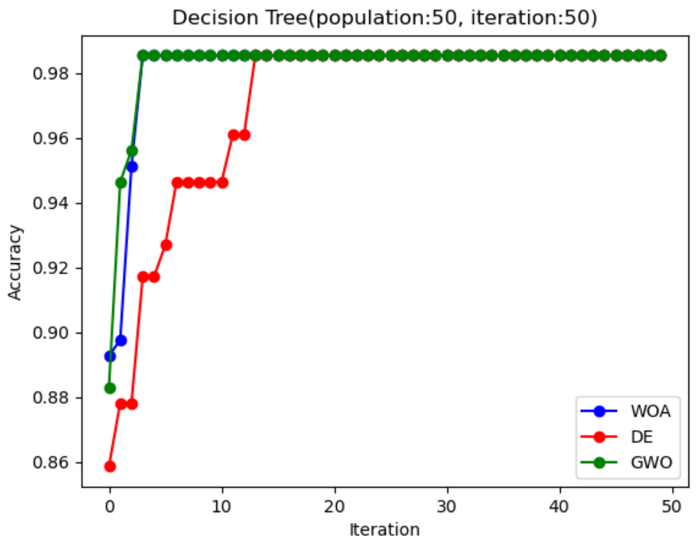


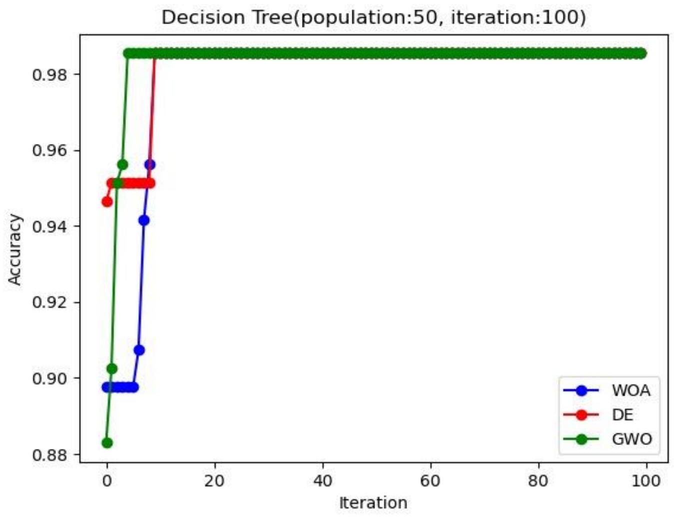
4.4. Decision Tree Hyperparameter Tuning

For the decision tree model, we chose the hyperparameters as criterion, splitter, max_depth, min_samples_split and min_samples_leaf. The dimension was set to five. We ran each optimization algorithm at 4 different population sizes and iterations and generated the following tables.



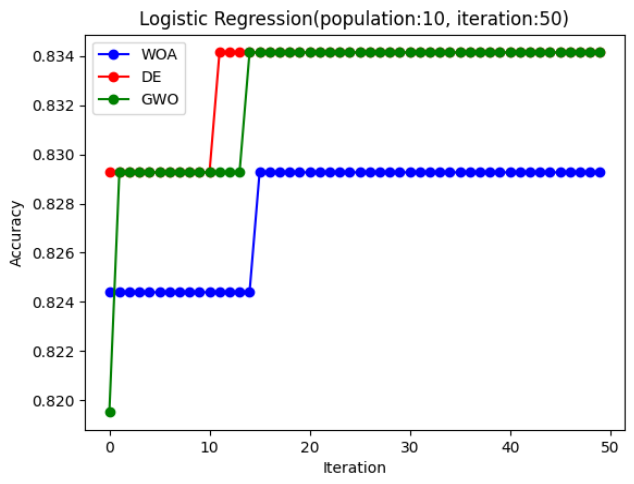


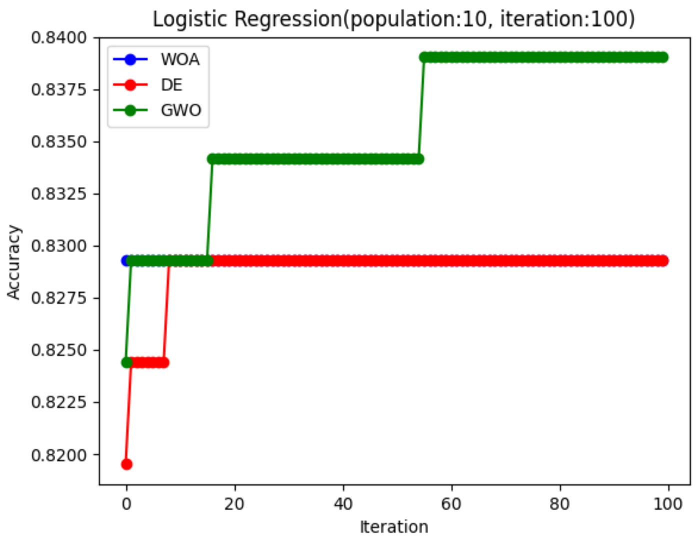


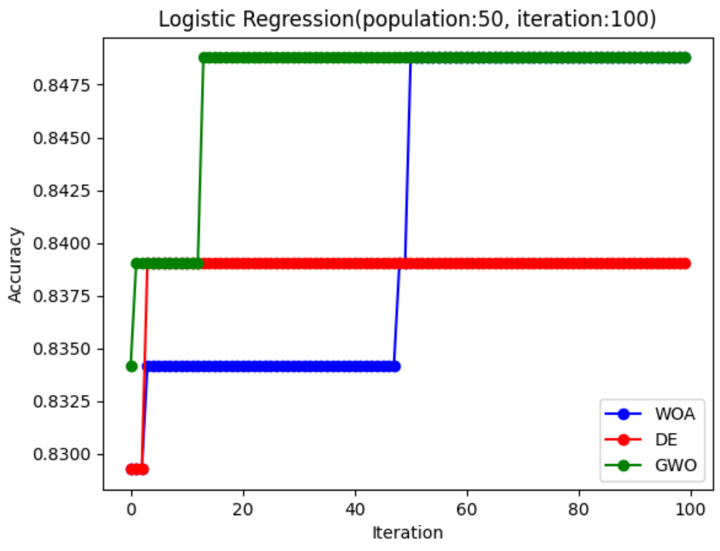


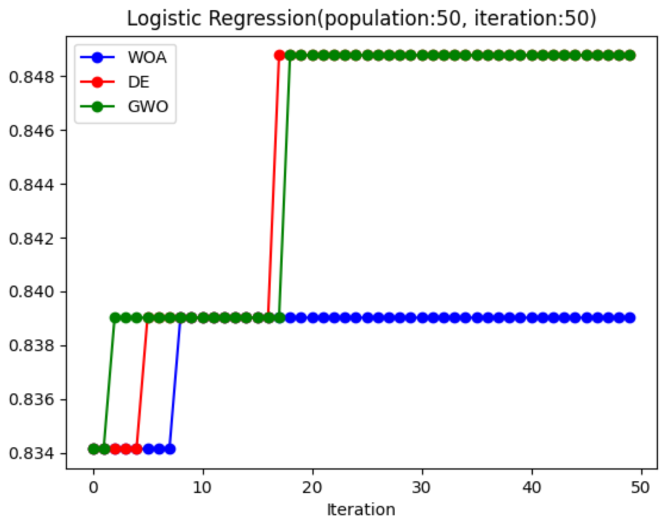
4.5. Logistic Regression Hyperparameter Tuning

For the logistic regression model, we chose the hyperparameters as penalty, C and solver. The dimension was set to three. We ran each optimization algorithm at 4 different population sizes and iterations and generated the following tables.



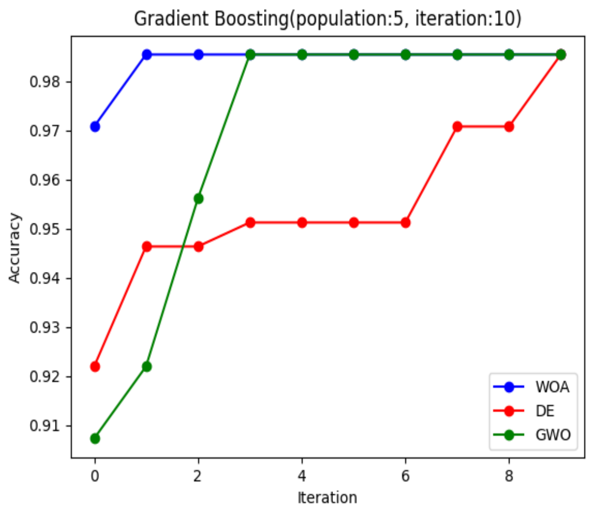


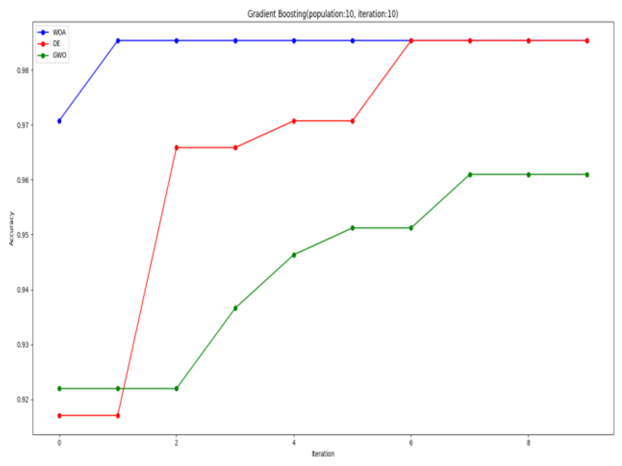


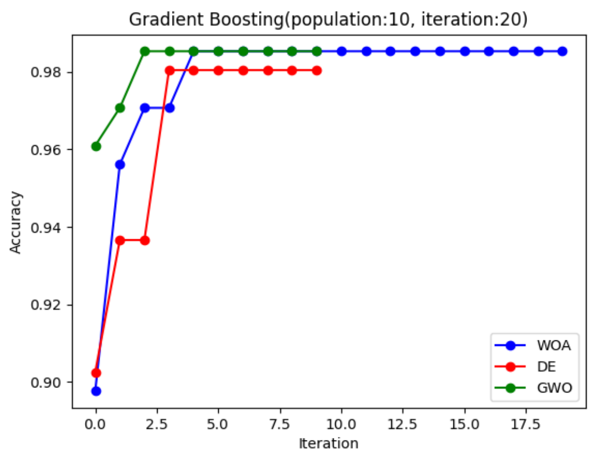


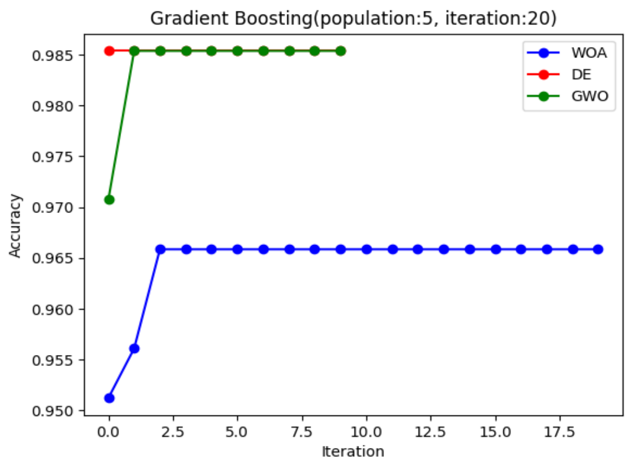
4.6. Gradient Boosting Hyperparameter Tuning

For the gradient boosting model, we chose the hyperparameters as learning_rate, n_estimators, max_depth, min_samples_split and min_samples_leaf. The dimension was set to five. We ran each optimization algorithm at 4 different population sizes and iterations and generated the following tables.









5. Conclusion

In this study, we explored the application of three nature-inspired optimization algorithms, namely WOA (Whale Optimization Algorithm), GWO (Grey Wolf Optimizer), and DE (Differential Evolution), for hyperparameter tuning in the context of heart disease detection using six different machine learning models. Our findings highlight the efficiency of these algorithms in achieving optimal hyperparameter configurations with notable improvements in model accuracy.

All 3 optimization algorithms achieved 98.5% accuracy for AdaBoost, SVM, Random Forest, Decision Tree and Gradient Boosting with sufficient population size and number of iterations. For logistic regression, it reached 84.7% accuracy. And these results are higher than the accuracy without hyperparameter selection.

The results show that, against the common assumption, the use of high iteration and population sizes is not necessary for our specific problem and dataset. The selected algorithms show solid performance at lower iteration and population sizes, demonstrating their effectiveness in our application domain. Moreover, models trained with hyperparameters determined through the optimization process exhibited higher accuracy rates compared to the default settings. This observation underlines the importance of our chosen hyperparameter tuning approach in improving the predictive capabilities of machine learning models.

In conclusion, our study not only contributes insights into the effectiveness of WOA, GWO and DE algorithms for hyperparameter optimization, but also highlights the importance of specific parameter configurations in achieving superior model performance. The results suggest a promising pathway for the development of reliable and accurate models for heart disease detection and demonstrate the potential impact of nature-inspired optimization algorithms in medical applications.

6. References

https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset
https://www.sciencedirect.com/science/article/pii/S0965997816300163
https://www.sciencedirect.com/science/article/pii/S0965997813001853
https://www.frontiersin.org/articles/10.3389/fbuil.2020.00102#:~:text=Differential%20evolution%20(DE)%20is%20a,explore%20very%20large%20design%20spaces.
https://www.kaggle.com/code/tanshihjen/eda-classification-heartdisease#Predictive-Analytics:-Classification