

Research Article

Heart Disease Prediction Using Machine Learning with Greater Cane Rat Algorithm-Based Hyperparameter Tuning

Khalid Fariq Hamajan ^{1,2} , Aso M. Aladdin ^{3,*} , Azad A. Ameen ⁴ 

¹ Department of Computer Science, College of Science, Charo University, Chamchamal, 46023, Iraq

² Department of Computer Science, United Skills Institute, Kalar, 46021, Iraq

³ Computer Engineering Department, Faculty of Engineering, Tishk International University, Kurdistan Region, 46001, Iraq

⁴ Department of Software Engineering, College of Engineering and Computational Science, Charo University, Chamchamal, 46023, Iraq

*Corresponding Author: Aso M. Aladdin, E-mail: aso.aladdin@chu.edu.iq

Article Info	Abstract
Article History	Cardiovascular diseases remain a leading global cause of mortality, highlighting the need for reliable and efficient tools for early risk identification. Machine learning (ML) models have shown promise for heart disease prediction; however, their performance can be highly sensitive to hyperparameter selection. Conventional hyperparameter optimisation approaches (e.g., grid search and random search) may be computationally demanding and can explore the search space inefficiently, especially when hyperparameters interact. To address this, this study investigates the Greater Cane Rat Algorithm (GCRA), a nature-inspired metaheuristic, as a hyperparameter optimisation framework for heart disease prediction. GCRA was applied to tune five classifiers, logistic regression (LR), random forest (RF), support vector machine (SVM), k-nearest neighbours (KNN), and decision tree (DT), while naïve Bayes (NB) was evaluated using default settings due to its limited tunable hyperparameters. Experiments were conducted on the UCI Heart Disease dataset under two train–test splits (80/20 and 70/30). Model performance was assessed using accuracy, precision, recall, and F1-score, and results were compared against genetic algorithm (GA) tuning as well as grid search (GS) and random search (RS). Across most models, GCRA improved performance relative to untuned baselines, with the largest gain observed for KNN, where accuracy increased from 0.8033 to 0.8689 under the 80/20 split. Overall, the findings indicate that GCRA can provide an effective and practical hyperparameter optimisation strategy for ML-based heart disease prediction under consistent evaluation settings.
Received Nov08, 2025	
Revised Feb 26, 2026	
Accepted Mar 07, 2026	
Keywords	
Hyperparameter optimization	
Metaheuristic optimisation	
Greater Cane Rat Algorithm (GCRA)	
Heart disease prediction	
Supervised machine learning	
UCI Heart disease dataset	



Copyright: © 2026 Khalid Fariq Hamajan, Aso M. Aladdin, and Azad A. Ameen. This article is an open-access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY 4.0) license.

1. Introduction

Cardiovascular diseases (CVDs) remain a major global health burden and continue to be among the leading causes of mortality. The scale of the problem has motivated sustained efforts to improve early identification of individuals at risk, especially using data-driven approaches that can support clinical

screening and decision-making. Recent work highlights that timely prediction of heart disease can assist prevention and intervention, particularly when risk factors are detected early and treatment pathways can be initiated proactively [1, 2]. Publicly available datasets, such as the UCI Heart Disease dataset, have enabled benchmarking of computational methods for heart disease prediction and have supported reproducible comparisons of machine learning (ML) models across consistent experimental settings [3, 4].

ML methods have been widely applied to heart disease prediction because they can learn non-linear patterns from demographic and clinical attributes (e.g., age, blood pressure, cholesterol, and electrocardiography-related features) and translate them into binary risk estimates [4, 5]. Classical supervised classifiers such as logistic regression (LR), random forest (RF), support vector machine (SVM), k-nearest neighbours (KNN), decision tree (DT), and naïve Bayes (NB) remain common choices in the literature because they are relatively interpretable, computationally efficient, and suitable for tabular medical data [4-7]. However, their predictive performance is often highly sensitive to hyperparameter settings, for example, the number of neighbours in KNN, regularisation strength in LR/SVM, and tree depth or ensemble size in DT/RF, meaning that default settings can yield suboptimal outcomes or unstable generalisation [5, 7, 8].

Hyperparameter optimisation (HPO) is therefore a critical step in medical prediction pipelines, yet conventional approaches such as grid and random search can be computationally expensive and inefficiently explore the search space, especially when hyperparameters interact or the space is large [7, 9, 10]. To address this limitation, recent studies increasingly apply metaheuristic optimisation methods to guide hyperparameter selection, balancing exploration (global search) and exploitation (local refinement) to reduce the risk of being trapped in local optima [11-13]. Metaheuristic-driven HPO has been reported to improve predictive performance for heart disease classification in several settings, including GA-based tuning and hybrid optimisation pipelines, where model robustness and accuracy may increase relative to baseline configurations [5, 11, 14, 15].

In this context, the present study proposes the Greater Cane Rat Algorithm (GCRA) as a metaheuristic HPO framework for heart disease prediction. GCRA is a nature-inspired optimisation algorithm that models behavioural strategies to maintain a balance between exploration and exploitation during search [16]. In this work, GCRA is applied to optimise the hyperparameters of six widely used classifiers (LR, RF, SVM, KNN, DT, and NB) on the UCI Heart Disease dataset [3, 4]. Performance is evaluated using standard

classification metrics (accuracy, precision, recall, and F1-score), supported by confusion-matrix interpretation. Experiments are conducted under two train–test splits (80/20 and 70/30) to examine sensitivity to training size, and results are further compared with three established optimisation baselines: genetic algorithm (GA), grid search (GS), and random search (RS) [9, 13, 15]. The goal is to provide a consistent, reproducible evaluation that clarifies when metaheuristic tuning yields measurable gains and how GCRA compares against widely used HPO strategies in heart disease prediction.

2. Literature Review

Recent studies have shown that machine learning (ML) can support heart disease prediction using routinely collected clinical and demographic variables, with common baselines including logistic regression (LR), random forest (RF), support vector machine (SVM), k-nearest neighbours (KNN), decision tree (DT), and naïve Bayes (NB) [1, 3, 4, 6, 17]. These methods are widely adopted because they are efficient for tabular medical datasets and can model both linear and non-linear relationships between risk factors and outcomes [1, 3, 4, 17]. However, many ML classifiers are sensitive to hyperparameter settings (e.g., K in KNN, C/ γ in SVM, or depth/ensemble size in tree-based models), meaning that default configurations may yield unstable generalisation or suboptimal performance [3, 4, 17].

Hyperparameter optimisation (HPO) is therefore an important step in the heart disease prediction pipeline. Conventional approaches such as grid search (GS) and random search (RS) are straightforward but can be computationally costly, particularly when hyperparameter interactions expand the search space or when multiple models must be tuned consistently [4, 18]. To address this, recent work increasingly uses metaheuristic and evolutionary optimisation methods to guide HPO, often reporting improved predictive performance compared with baseline configurations [7, 9, 11, 12, 19]. For example, metaheuristic tuning has been applied to SVM and RF using GA and other nature-inspired algorithms, with reported gains in accuracy and/or efficiency relative to conventional tuning strategies [2, 7, 11, 12, 19]. Other studies evaluate multiple classifiers on UCI-based benchmarks and note that the benefit of tuning is often larger for KNN and SVM than for NB, which typically has limited effective hyperparameters under common formulations [1, 4, 17]. Beyond classical ML, some studies also discuss emerging approaches (e.g., deep learning or quantum ML frameworks) for heart disease prediction, although such methods are often less directly comparable to lightweight clinical screening models due to differences in assumptions, implementation complexity, and evaluation protocols [8, 14, 15].

Despite this progress, three limitations remain common in the recent literature. First, studies frequently focus on one classifier family (e.g., SVM or RF) or one optimiser, making it difficult to compare methods under consistent evaluation conditions [4, 7, 9, 11, 12, 19]. Second, experimental protocols vary substantially (different splits, different reporting metrics, and inconsistent validation), which complicates conclusions about robustness and generalisation on small datasets such as the UCI Heart Disease dataset [1, 3, 4, 17]. Third, many papers emphasise performance improvements but report limited evidence about computational cost and statistical stability, even though metaheuristic tuning can introduce non-trivial optimisation overhead [7, 9, 19]. To provide a structured comparison of recent trends, Table 1 summarises studies published between 2024 and 2026 that use benchmark datasets (preferably the UCI Heart Disease dataset) and report ML models, tuning strategies, and performance metrics [1-3, 6, 12, 13, 17, 18, 20].

Overall, Table 1 indicates that the most frequently evaluated classifiers for UCI-style heart disease prediction are RF, SVM, LR, KNN, and DT, with NB often included as a baseline [1-3, 6, 13, 17, 20]. The most commonly used optimisation strategies include GS/RS and metaheuristics such as GA and related nature-inspired variants, with many studies reporting improved accuracy and/or F1-score after tuning [1, 2, 4, 11, 12, 17, 18]. However, the evidence is still fragmented because comparisons are often conducted under different splits, with different validation schemes and incomplete reporting of optimisation cost and statistical stability.

Motivated by these gaps, the present work evaluates the Greater Cane Rat Algorithm (GCRA) as a metaheuristic HPO framework and compares it against GA, GS, and RS using a consistent protocol across six standard classifiers (LR, RF, SVM, KNN, DT, and NB) on the UCI Heart Disease dataset [4-6, 18, 21]. In addition to split-based evaluation, the study reports cross-validation results and computational time analysis to support a more complete assessment of performance and robustness.

Overall, recent studies commonly evaluate RF, SVM, LR, and KNN on UCI-style datasets, with DT and NB often included as baselines. To improve predictive performance, various metaheuristics and evolutionary optimisation techniques have been included in several recent analyses, including GA, PSO, ACO, the FOX metaheuristic, and hybrid GA-based models for feature selection and hyperparameter tuning [6, 7, 9, 14, 15, 22]. Performance metrics have typically used accuracy, precision, recall, F1-score, specificity, and ROC-AUC [4, 5, 7, 8, 22]. This work evaluates GCRA as a metaheuristic hyperparameter optimisation framework and compares it against GA, GS, and RS under a consistent protocol.

Table 1. Summary of recent studies (2024–2026) on heart disease prediction and hyperparameter optimisation

Ref	Author(s)	Dataset	Model(s)	Method(s) optimizer	on Metrics	Best Reported Performance
[15]	Paliwal et al. (2024)	Heart Disease Dataset	SVM	GA	Accuracy	97% accuracy
[8]	Ehtiba et al. (2024)	UCI Heart Disease (303 samples)	LR, SVM, RF, KNN	GS	Accuracy, Precision, Recall, F1-score	KNN: 92.44% accuracy, RF: 91.60%
[23]	Li (2024)	UCI Heart Disease (303)	LR, KNN, DT, ANN, DNN	Comparative ML evaluation	Accuracy, Precision	LR: 93.40% accuracy
[22]	Gupta et al. (2025)	UCI Heart Disease	RF + CNN (Ensemble)	GA-CSO feature selection + GA tuning	Accuracy, Precision, Recall, F1-score, ROC-AUC, Specificity	Best value not explicitly stated
[5]	Uddin et al. (2025)	UCI Heart Disease	RF, SVM, DT, KNN, LR	Feature selection + HPO (Hyperparameter Optimisation)	Accuracy, Precision, Recall, F1-score	RF: 93.41% accuracy, SVM: 91.21%
[6]	Hamza & Al-Ghanim (2025)	UCI Heart Disease (1,888 records)	RF, DT, SVM, KNN, NB	Hyperparameter tuning	Accuracy	RF: 96.30% accuracy
[4]	Nasution et al. (2025)	UCI Heart Disease	RF, SVM, LR, KNN	Feature selection comparison	Accuracy, Precision, Recall, F1-score	RF: 89.7% accuracy
[9]	Pathak et al. (2025)	UCI Heart Disease	DT	GS-RS (hybrid)	Accuracy, Computational efficiency	Improved accuracy & efficiency
[7]	Masbakhah et al. (2025)	UCI Heart Disease	RF	FOX metaheuristic tuning	Accuracy, Precision, Recall, F1-score	97.83% accuracy, F1 = 97.89%
[14]	Narasimhan & Victor (2025)	Cleveland Heart Dataset	RF	GA, PSO, ACO	Accuracy	GA-RF achieved highest accuracy among metaheuristics

3. Methodology

This study evaluates six machine learning classifiers, logistic regression (LR), random forest (RF), support vector machine (SVM), k-nearest neighbours (KNN), decision tree (DT), and naïve Bayes (NB), for heart disease prediction on the UCI Heart Disease dataset. The classifiers are evaluated under two train–test split ratios (80/20 and 70/30) using accuracy, precision, recall, and F1-score. GCRA is used to tune hyperparameters for LR, RF, SVM, KNN, and DT, while NB is evaluated with default settings due to limited tunable hyperparameters. Results are compared against GA, grid search (GS), and random search (RS). The proposed methodology is presented as a structured workflow, illustrated in Figure 1, comprising many stages. Figure 4 summarises the workflow: dataset selection, preprocessing, model training, hyperparameter tuning, and evaluation. Experiments were implemented in JupyterLab (Anaconda) on

Windows 11 (i5-8350U, 8 GB RAM). Thus, the main steps of the developed methodology are:

3.1. Dataset

The dataset of heart disease in this study was collected from the UCI ML Repository, which was first provided by Detrano in 1988 [3]. The UCI heart disease dataset is a well-established standard for predicting cardiovascular disease [24]. There are 303 patient records with 14 attributes of various risk factors for heart disease; The qualities encompass demographic information, including age and sex, medical test outcomes such as cholesterol and blood pressure levels, symptoms such as chest pain, and the presence or absence of heart disease, which serves as the target variable. The target variable indicates the presence of cardiac disease, represented by a value of 1, or its absence, indicated by a value of zero.

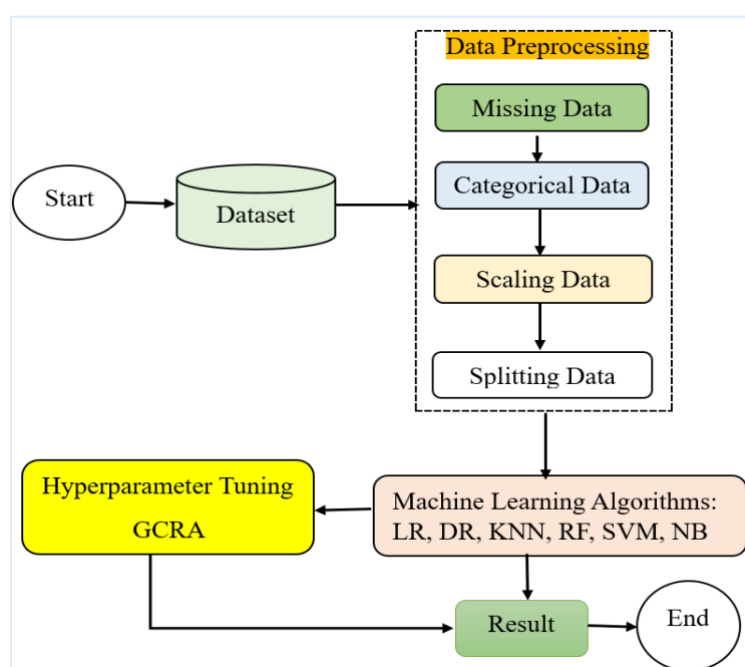


Figure 1. The architecture of the developed model

Table 2. Class distribution of the heart disease dataset

Class(target)	meaning	Samples of data	percentage
0	No heart disease	138	45.5%
1	heart disease	165	54.5%
Total		303	100%

In this study, no class balancing techniques were applied to this dataset. Random oversampling, random Under-Sampling, and Synthetic Minority Over-Sampling Technique (SMOTE) were not used, and the models were trained on the original data distribution without modifying class proportions. Table 2 indicates that the class distribution of the heart disease dataset remained unchanged and was consistent with

the applied preprocessing procedure. For the purpose of constructing supervised learning models, this variable was utilised [4, 5]. A brief description of each attribute is provided in Table 3.

Table 3. Heart disease dataset features

Attribute	Description	Data Type	Values
Age	Age of the patient	Numeric	Multiple values in the range of 28 and 77 (in years)
Sex	patient gender	Categorical	Female =0, Male =1
Cp	Type of Chest	Categorical	typical angina=0, atypical angina =1, non-anginal pain=2, asymptomatic =3
Trestbps	Blood pressure (in mm Hg)	Numeric	94-200 (in mm Hg)
Chol	Serum cholesterol level (mg/dl)	Numeric	126-564 (in mg/dl)
FBS	Blood sugar after fasting in mg/dl	Categorical	> 120 mg/dl (1 = yes; 0 = no)
Restecg	Resting electrocardiogram results	Categorical	Normal=0, Left ventricular hypertrophy =1, ST-T wave abnormality =2
Thalach	The highest heart rate achieved	Numeric	Multiple values from 71 to 202 (Low: under 50, Normal: 51–119, High: 120–180) beats/min
Exang	Exercise-induced angina	Categorical	No = 0, yes=1
Oldpeak	ST depression induced by exercise relative to rest	Numeric	between 0 and 6.2
Slope	Slope of the peak exercise ST segment	Categorical	down sloping =2, flat=1 , Upsloping =0
Ca	principal of vessel numbers	Numeric	(0–3) fluoroscopy colored
Thal	Thalassemia	Categorical	reversible defect, and the label = 2 , fixed defect= 1, Normal= 0
Target	Determination of the patient's cardiac condition	Categorical	No = 0, yes =1

3.2. Preprocessing of Data

To ensure data quality and compatibility with ML models, the following preprocessing steps are applied [4, 6, 8, 19, 25].

- **Missing Data:** Missing values, outliers, and duplicate records are common data quality issues that must be addressed during data preprocessing to ensure data quality and correctness. These are values that should be present in a dataset but are not available. Missing values occur when data is not recorded. Values that deviate far from the norm are called outliers. Duplicate records are repeated observations within the dataset. Strategies for handling missing data are generally classified into three main categories: (1) filling in missing values with zeros, ignoring data with missing values, or deleting records with missing values; (2) single imputation strategies; and (3) multiple imputation strategies [24].
- **Categorical Encoding:** Categorical variables (e.g., chest pain type, thalassemia) are transformed into numerical values using one-hot encoding.

- **Normalisation and Scaling:** Continuous attributes were standardised (z-score) to support distance- and margin-based models (particularly KNN, SVM, and LR). Tree-based models (DT, RF) do not require scaling.
- **Splitting Data:** The dataset is partitioned into 80% training and 20% testing, or into 70% training and 30% testing.

3.3. Machine Learning Algorithms

ML algorithms are fundamentally collections of directives that enable computers to learn from data, generate predictions, and improve their performance over time without explicit programming. Various algorithms generate predictions using processed data [6, 26]. This study employs the following ML classifiers: LR, KNN, SVM, DT, RF, and NB. After data preprocessing, models were trained and evaluated both with and without hyperparameter adjustment, and the evaluations were conducted independently. Figure 2 depicts the ML models employed in this research.

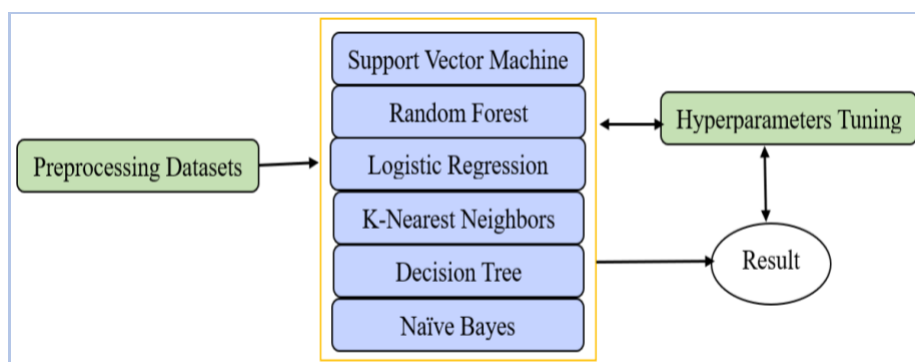


Figure 2. Represents the ML classifiers in this work

- **Logistic Regression:** It is one of the oldest and most popular methods for using statistics to solve binary classification problems, where the dependent variable is categorical [24, 27]. It uses the logistic (sigmoid) function to model the conditional probability of the positive class. This function maps linear combinations of input features to values in the [0,1] interval [25]. Maximum likelihood estimation is typically used to estimate model parameter values. LR works best when the relationship between features and the log-odds of the outcome is close to linear. This makes it a good choice for determining whether a patient has heart disease [4].
- **Decision Tree:** DTs are supervised learning algorithms used for both classification and regression tasks in ML. They construct hierarchical, tree-structured models by recursively partitioning the feature space using decision rules that maximise node purity [28]. DTs are highly interpretable and easy to visualise;

however, they are prone to overfitting, especially when grown deep without pruning. DTs are often used to predict heart disease because they are easy to interpret and can handle diverse data types [5, 27].

- **Random Forest:** RF is an ensemble learning algorithm that builds several DTs during training and combines their predictions to make the final prediction [4]. For classification tasks, RF uses majority voting, while for regression, it computes the average of individual tree predictions [27]. By introducing randomness through bootstrapped sampling and feature subset selection, RF improves predictive accuracy and effectively reduces overfitting compared to a single DT [26].
- **KNN:** It is a non-parametric, instance-based learning algorithm that assigns class labels based on the majority vote of the k closest training samples in the feature space [4, 29]. Unlike ensemble methods such as RF, which build multiple DTs during training, KNN performs no explicit model training and makes predictions directly from the local neighbourhoods of the query instance. KNN works best for classification tasks where the decision boundaries are not straight lines. This is because it naturally adapts to the data distribution [25].
- **Naïve Bayes:** It is a probabilistic classification algorithm based on Bayes' theorem with the simplifying assumption of conditional independence among features [29]. Although this assumption rarely holds in real-world data, NB is computationally efficient, requires minimal parameter tuning, and often achieves competitive performance in medical classification tasks, including heart disease diagnosis [6, 27, 28].
- **SVM:** It is a supervised learning algorithm that is great for classification and regression tasks, especially on small to medium-sized datasets. SVM's goal is to find the best separating hyperplane that maximises the margin between the classes. Kernel functions enable it to model non-linear decision boundaries in the input space [10, 15, 29].

3.4. Hyperparameters Metaheuristic Algorithms Tuning

ML models frequently exhibit tendencies toward overfitting and underfitting when applied to limited datasets [6]. The issue can be addressed through hyperparameter tuning, exploring various Permutations of hyperparameter selections and evaluating the model's performance on both the training and test datasets [29]. When the performance on the training data significantly exceeds that on the testing data, it is referred to as overfitting. Conversely, if the model's performance on the training data is markedly inferior to its performance on the test data, it is termed underfitting [15]. Hyperparameter adjustment facilitates the identification of the optimal model configuration for variance, also known as overfitting and underfitting,

by balancing bias, thereby ensuring robust generalisation [10]. Hyperparameter tuning is the process of selecting parameter values for an ML algorithm to improve its accuracy and performance. The parameters' values dictate the algorithm's performance and the training-testing procedure [5]. Table 4 presents the parameters of the ML methods employed in this investigation before optimisation.

Numerous methods exist for hyperparameter optimisation of ML models, including metaheuristic algorithms. This study employs the GCRA metaheuristic to identify the optimal parameter set and assess the models using that set. By using GCRA for hyperparameter tuning, ML models can achieve higher accuracy, precision, recall, and F1 scores than conventional methods. The algorithms regulate search diversity, prevent premature convergence, and ensure stability in optimised models. This novel adaptation has not been explored in prior heart disease prediction studies, making this a new contribution to the field. For each classifier, GCRA was applied to optimise critical hyperparameters, as shown in Table 5. Hyperparameters were sampled within predefined bounds, and candidate solutions were evaluated using the fitness function. Until a certain number of generations have passed, this process will continue. The optimal hyperparameter combination is identified by analysing the fitness score from the previous generation.

Hyperparameter initialisation ranges and tuned values were as follows: for RF, the number of trees, `n_estimators`, was searched in [10,100] and tuned to 26; for KNN, the number of neighbours, `K` was searched in [1,15] and tuned to 7; for DT, number of trees the `max_depth` was searched in [1,15] and tuned to 6. For SVM, the regularisation parameter `C` was searched in [0.1,10] and tuned to 0.153132, while for LR, `C` (inverse regularisation strength) was searched in [0.01,100] and tuned to 0.01, indicating stronger regularisation. In this study, NB is typically employed with its default hyperparameter settings; therefore, all experiments were conducted with these settings.

Table 4. Parameters of ML algorithms

MLA	Parameter
RF	Number of trees(<code>n_estimators</code>), maximum tree depth (<code>max_depth</code>)
KNN	Number of neighbours (<code>n_neighbours</code>), weighting scheme (<code>weights</code>)
DT	Maximum tree depth (<code>max_depth</code>), splitting criterion (<code>criterion</code>)
SVM	<code>C</code> (regularisation), kernel coefficient (<code>gamma</code>)
LR	<code>C</code> (regularisation strength), solver (<code>solver</code>)
NB	None

Table 5. Hyperparameter tuning of ML classifier for GCRA

MLA	Parameter	Range	After tuning the parameter
RF	Number of trees	[10, 100]	n_estimators = 26
KNN	Number of neighbours (n_neighbors)	[1, 15]	K=7
DT	Maximum tree depth	[1, 15]	max_depth=6
SVM	C (regularisation)	[0.1,10]	C=0.153132
LR	C (regularisation strength)	[0.01, 100]	C=0.01
NB	None	default	Default

The search space for hyperparameters was defined based on literature reviews and pilot tests. Constraints were used for both RF and DT models to prevent overfitting; meanwhile, hyperparameters for KNN, LR, and SVM models addressed underfitting and overfitting. The GCRA performed an implicit sensitivity analysis, resulting in performance improvements across different data split ratios in split-test evaluations, indicating the robustness and generalisation capability of the obtained hyperparameters.

3.4.1. GCRA Model

The GCR is a territorial animal consisting of a dominant male, several females, and their offspring. It uses its nose for fighting and leaves trails for essential supplies. The GCRA balances exploitation and exploration and is mathematically modelled using this adaptable behaviour [16, 30].

The optimisation methods of GCRA start with the generation of a matrix randomly, also known as the population x matrix in equation 1. The algorithm was run 30 times in an experiment of 1,000 iterations, with a population size of 30. Each run used a different random seed, and reported results are averaged across runs unless stated otherwise. The population of the GCRA is randomly generated by choosing the greatest value as the upper bound (UB) and the smallest value as the lower bound (LB) [16, 30]

$$x = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d-1} & x_{1,d} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,d-1} & x_{2,d} \\ \vdots & \vdots & x_{i,j} & \vdots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,d-1} & x_{n,d} \end{bmatrix} \quad (1)$$

Here, x is the whole population GCRA of the matrix, with i^{th} position and j^{th} dimension in the individual rat $x_{i,j}$ is stochastically generated at the start of the algorithm and implemented as shown in equation 2. Furthermore, the population group's size represents the n symbol, and the d symbol denotes the population's dimension.

$$x_{i,j} = LB_j + rand \times (UB_j - LB_j) \quad (2)$$

The alpha male rat, known for leading the group and prioritising refuge or food, modifies the

placements of other rats, as outlined in equation 3.

$$x_{i,j}^{new} = 0.7 * \frac{(x_{i,j} + x_{k,j})}{2} \quad (3)$$

where the new i^{th} GCR's status is indicated by (x_i) , The current GCR's position is indicated by $(x_{i,j})$, $x_{i,j}^{new}$ significant values in the j^{th} dimension, and the alpha male on the j^{th} dimension is represented by $x_{k,j}$. The objective function's current value indicates that the alpha male's fittest function value is indicated by $f_{x,i}$. A random number C is defined within the problem space. r is used to model the result of a plentiful source of food, which subsequently necessitates more in detail and is specified using equation 6, where equation 7 defines the coefficient α , which simulates a declining source of food and forces the quest for recent sources of foods or shelter; equation 8 defines the coefficient β , this prompts the GCR to go to the breeding area's other abundant sources of foods [16, 30].

The value of the ρ parameter, which determines whether the GCRA enters the exploration or exploitation phase, is set during a wet season.

$$x_{i,j}^{new} = x_{i,j} + c \times (x_{k,j} - r \times x_{i,j}) \quad (4)$$

A thorough parametric analysis is used to carefully adjust ρ to 0.5. The control parameter ρ determines the phase, and when a random value is less than ρ , an exploration occurs and goes to equations (4) and (5); in other words, exploitation occurs in equations (9). Exploration updates promote diversification by moving candidates using random interactions and coefficients, whereas exploitation updates intensify the search by refining candidates around the current best (alpha) solution. After each update, solutions are clipped to stay within the predefined bounds (LB, UB), ensuring that all hyperparameter values are valid. To further illustrate, see Figure 3, which shows pseudo-code for the heart disease dataset for testing and training.

$$x_i = \begin{cases} x_{i,j} + c \times (x_{i,j} - \alpha \times x_{k,j}), & f_i^{new} < f_i \\ x_{i,j} + c \times (x_{m,j} - \beta \times x_{k,j}), & \text{otherwise} \end{cases} \quad (5)$$

$$r = F_{x,k} - C_{iter} \times \left(\frac{F_{x,k}}{Max_{iter}} \right) \quad (6)$$

$$\alpha = 2 \times r \times rand - r \quad (7)$$

$$\beta = 2 \times r \times \mu - r \quad (8)$$

$$x_{i,j}^{new} = x_{i,j} + C \times (x_{k,j} - \mu \times x_{m,j}) \quad (9)$$

Here, C_{iter} It is a new iteration, and the highest iteration is called Max_{iter} .

```

#inputs:
#x: attributes of Dataset (No=14)
# targets of the dataset indicating heart disease or not (0=No, 1=yes)
# test_size: proportion of the dataset (0.20 for 20%) used for testing
# random state=42
#outputs:
# x_train, x_test: Attribute subset to training and testing
#y_train, y_test: target labels.
# Preprocess the data
# y_categorical: categorical(y)
# Scale the dataset
# X_scaled: scaler.fit_transform(X)
# Split the dataset into training and testing sets
# ML Classifiers
Function: train_test_split (X_scaled, y, test_size=0.2, random_state=42)
    return accuracy_score (y_test, predictions)
GCRA implementation:
    for l in range(max_iterations)
        calculate r, a m, B, and c
        if np. random. Rand () > gr_rho
            exploration
        Update GCR location according to equ (4)
        Else Exploration
        Update GCR location according to equ (9)
        End if
        Determine the fitness of the search agent to the current position
        Find the latest GCR location. Based on equ (5)
        Election of the recent hyperparameter
        End for
Return: x_train, x_test, y_train, y_test
End

```

Figure 3. Represents the pseudo-code dataset for testing and training

4. Results and Discussions

This section compares ML classifiers such as KNN, NB, SVM, RF, LR, and DT without hyperparameters against the GCRA metaheuristic as hyperparameters for these metrics (confusion matrix, precision, accuracy, recall, and F1-score), and compares GCRA with the GA metaheuristic and two methods, GS, and RS, then tested on five-fold cross-validation.

4.1. Evolution and Performance Metrics

The evaluation of an ML model's performance is essential to assess its efficacy, reliability, and capacity to generalise to data that has not yet been observed. Performance metrics provide quantitative measures for comparing models and assessing how well they predict or classify outcomes. Among these metrics, the most popular ones for assessing model performance are confusion, accuracy, precision, recall, and F1-score.

The performance of a classification model can be conceptualised using a confusion matrix, a tabular representation. It presents the total number of accurate and incorrect predictions, broken down by class [5].

In Figure 4, it is structured as a square table, with each row representing the actual class and each column indicating the anticipated class. Truthful positives are denoted by the letters TP, while true negatives are denoted by the letters TN, false positives by the letters FP, and false negatives by the letters FN. The matrix is composed of these four components.

Table 6. Nomenclature of symbols

Symbol	Definition
x	population GCRA of the matrix
n	population group's size
$x_{i,j}$	i^{th} position and j^{th} dimension in the individual rat
d	Dimension of solution vector (number of tuned hyperparameters) , while $d \geq 1$
$x_{k,j}$	alpha male on the j^{th} dimension of GCR
$x_{i,j}^{new}$	new i^{th} GCR's status is indicated by (x_i) , s
$f_{x,i}$	objective function's current value indicates that the alpha male's fittest function value.
c	Random coefficient controlling step magnitude, value of c between (0,1)
r	Term modelling food availability/pressure over iterations. defined via the referenced equations
α	Coefficient/term used in the movement model (food/leader influence), appears in the update equations.
β	Coefficient/term used in the movement model (breeding/alternate resources), appears in the update equations.
ρ	Control parameter that selects exploration vs. exploitation, set to 0.5
μ	Random scaling factor used in exploration (if included in implementation)
$rand$	Random number for switching phases (between (0,1), exploration occurs if $rand \leq 0.5$, else if exploitation occurs
C_{iter}	It is a new iteration, start with 1
$Max_{iter.}$	The maximum iteration is defined.
f_i	Fitness (objective) function
C	Regularisation strength
$n_estimators$	Number of trees
LB/UB	Lower/upper bounds of decision variables
max_depth	maximum tree depth
$n_neighbours$	Number of neighbours

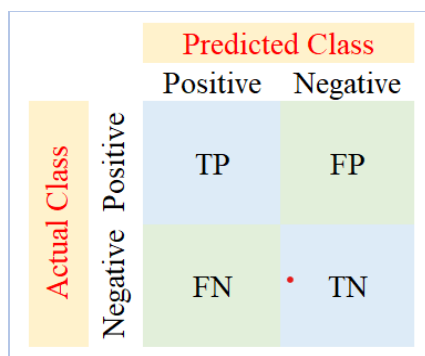


Figure 4. Confusion Matrix Standard

The predominant measures for assessing model performance include confusion metrics, accuracy, precision, recall, and F1-score [4, 15, 17, 29].

- **Accuracy:** This is the percentage of all samples that were correctly classified as either diseased (true positives) or not diseased (true negatives). It is a general performance metric, but it may not be accurate when used on datasets with uneven class distributions.

Precision: proportion of predicted positive cases that are truly positive: $\text{Precision} = TP / (TP + FP)$.

Recall (Sensitivity): proportion of actual positive cases correctly identified: $\text{Recall} = TP / (TP + FN)$.

F1-score: harmonic mean of precision and recall: $F1 = 2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$.

$$\text{Accuracy} = \frac{TP+TN}{TP+FN+TN+FP} \quad (10)$$

- **Precision:** It measures the proportion of predicted positive instances that are truly positive. It is calculated using the following formula:

$$\text{Precision} = \frac{TP}{TP+FP} \quad (11)$$

- **Recall (Sensitivity):** This is the proportion of correctly identified instances of disease among all actual cases of disease. It indicates how well the model identifies disease. A low level of recall in clinical situations is even more dangerous, as failure to detect a disease in patients often has severe consequences.

$$\text{Recall} = \frac{TP}{TP+FN} \quad (12)$$

- **F1-Score:** It is given by the calculation of the harmonic mean of precision and recall. The use of F-Score in cases of imbalanced classes is particularly important because other evaluation criteria do not give equal weight to precision and recall.

$$\text{F1-Score} = 2 \left(\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \right) \quad (13)$$

LR, SVM, KNN, NB, DT, and RF are among the well-established ML models evaluated in this part. The performance of the suggested model is benchmarked against these models. In this presentation, the results are derived from trials that utilised ML algorithms to forecast the presence of cardiac disease. Both studies, conducted simultaneously, compared training models without hyperparameter tuning and models with hyperparameter tuning utilising the GCRA metaheuristic.

GCRA is applied as a minimisation procedure; therefore, maximising a performance metric is achieved by minimising its negative value. The algorithm optimises a single metric depending on the experimental configuration:

- **Accuracy-based tuning:**

$$f(x) = - \text{Accuracy}(x) \quad (14)$$

- **F1-based tuning:**

$$f(x) = -f1(x) \quad (15)$$

Those formulations enable GCRA to maximise the chosen metric while remaining consistent with the minimisation framework. In real-world applications, accuracy is a suitable evaluation metric when the class distribution is roughly balanced. On the other hand, the F1-score is often used when a stricter balance between precision and recall is needed, especially in medical decision-making applications where the risk of false negatives is likely to be much higher.

4.1.1. Performance Under 80% Training and 20% Testing

In this section, detailed performance is presented for ML classifiers (KNN, SVM, RF, LR, DT, and NB) with 80% training and 20% testing, covering accuracy, precision, recall, and F1-score. This section is divided into two parts: a comparison of GCRA with hyperparameters against GCRA without hyperparameters, and a comparison of GCRA against GA, GS, and RS.

4.1.1.1. Comparison of GCRA With vs Without Hyperparameters

The experimental results in Table 7 clearly show that including GCRA for hyperparameter tuning significantly improved the performance of almost all the ML classifiers tested. The greatest improvement was shown by the KNN model, whose average accuracy increased from 0.8033 to 0.8689 and its F1-score from 0.7997 to 0.8676, indicating that GCRA identified the best combination of k (the number of neighbours) and distance metric values to improve generalisation and stability of KNN. The RF classifier showed a significant improvement, with accuracy increasing from 0.8197 to 0.8525 and precision from 0.847 to 0.8699, supporting the hypothesis that GCRA successfully fine-tuned the number of estimators and tree depth, thereby reducing overfitting.

The SVM itself achieved moderate but consistent improvements, with its accuracy increasing from 0.8197 to 0.8361, and corresponding gains in both precision and recall, reflecting GCRA's ability to efficiently adjust both the kernel parameters and the strength of regularisation. LR and DT similarly benefited from GCRA optimisation, although the latter showed a quite striking increase in accuracy from 0.7049 to 0.7869, demonstrating GCRA's strong capability to improve performance and control overfitting in simple, rule-based models.

Overall, the results shown in Table 7 primarily demonstrate the effectiveness of GCRA as a strong hyperparameter optimisation technique for machine learning models; models that use GCRA tend to improve their performance metrics across both complex and simple models. Thus, these results demonstrate the main aim of using GCRA: to serve as a strong model that can improve classification performance regardless of the chosen model's complexity.

The confusion matrix further evidences this in Figures 5 and 6, where models optimised by GCRA have fewer misclassifications and stronger diagonal dominance.

Table 7. Comparison of ML classifiers without hyperparameters with GCRA as a hyperparameter by using 80% training and 20% testing

Model	Accuracy		Precision		Recall		F1 Score	
	without GCRA	with GCRA	without GCRA	with GCRA	without GCRA	with GCRA	without GCRA	with GCRA
KNN	0.8033	0.8689	0.8126	0.874	0.8033	0.8689	0.7997	0.8676
NB	0.8197	0.8197	0.8262	0.8262	0.8197	0.8197	0.8172	0.8172
SVM	0.8197	0.8361	0.8346	0.8473	0.8197	0.8361	0.8154	0.8331
RF	0.8197	0.8525	0.847	0.8699	0.8197	0.8525	0.8132	0.849
LR	0.8033	0.8197	0.8126	0.847	0.8033	0.8197	0.7997	0.8132
DT	0.7049	0.7869	0.7053	0.792	0.7049	0.7869	0.702	0.7839

* NB has default parameters

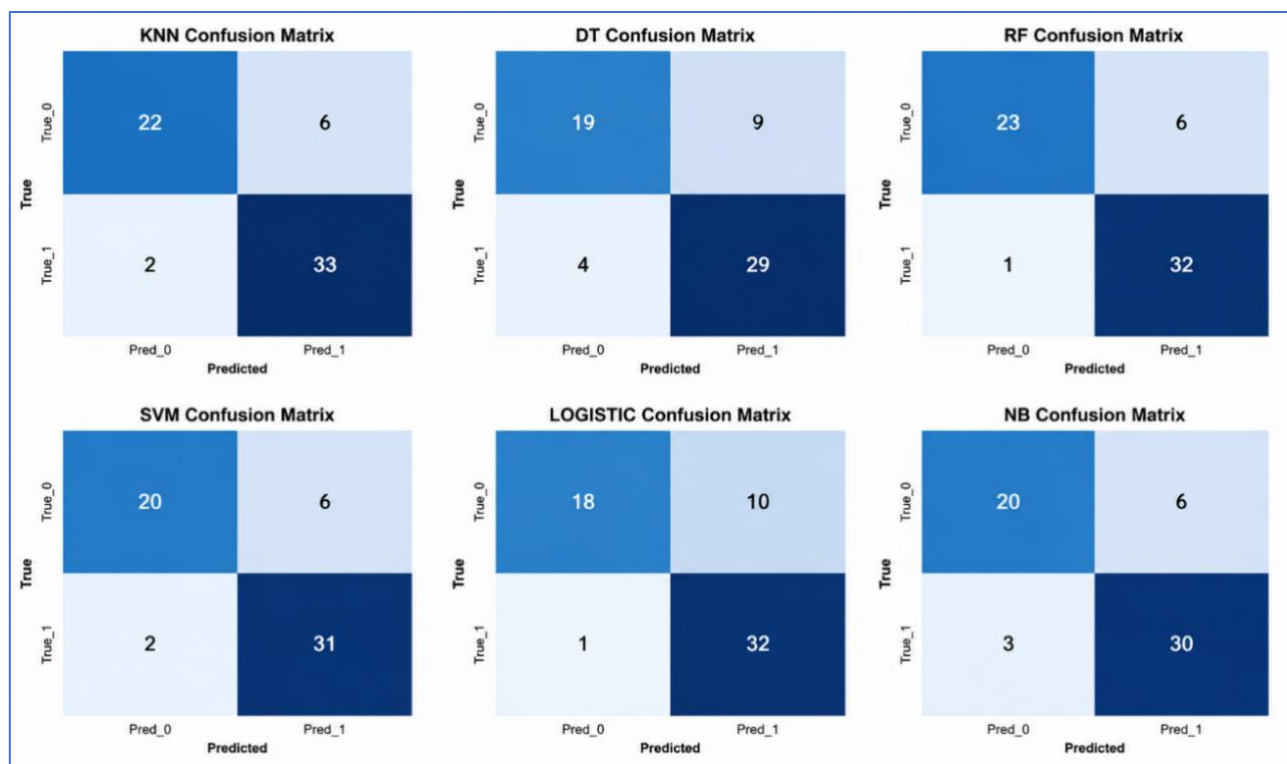


Figure 5. The results of all the confusion matrix ML classifiers with GCRA hyperparameters for 80% training and 20% testing

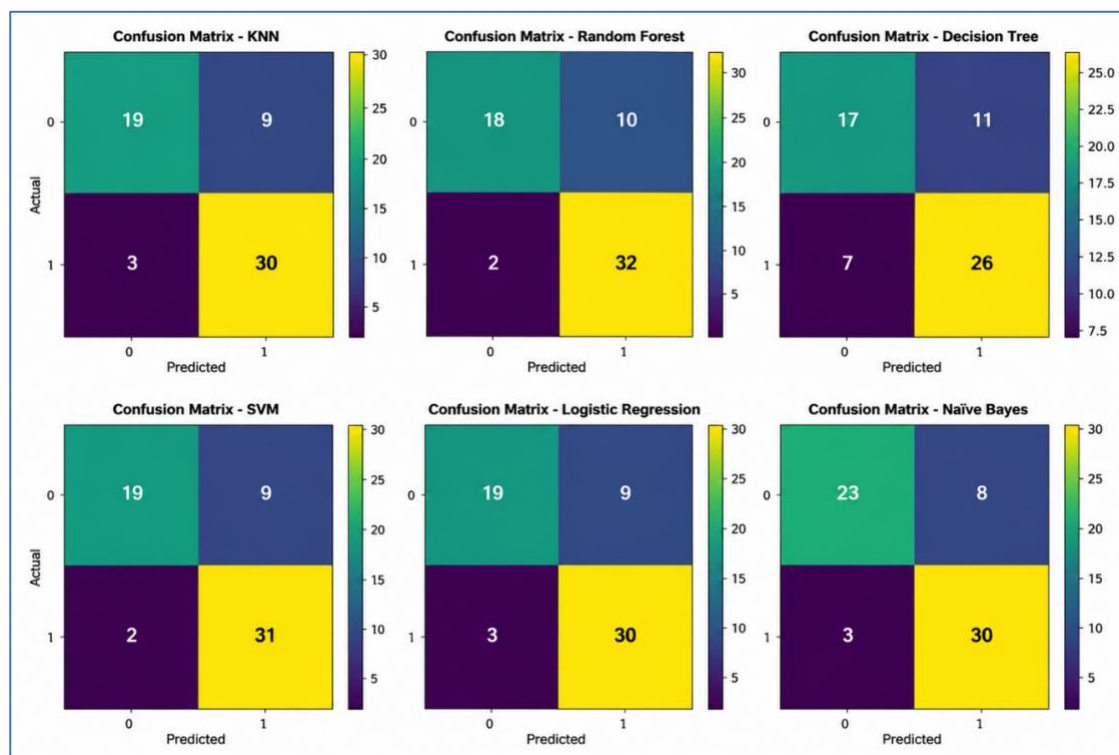


Figure 6. The results of all the confusion matrix ML classifiers without hyperparameters for 80% training and 20% testing

4.1.1.2. Comparison of GCRA with GA, GS, and RS

Table 9 compares the performance of six classifiers (KNN, SVM, RF, LR, DT, and NB) across four tuning methods (GCRA, GA, GS, and RS) using an 80% training and 20% test split. Overall, the results clearly show that GCRA consistently outperforms the other methods across most models and evaluation metrics.

GCRA yields the highest accuracy across all classifiers. The highest accuracy value is acquired by the KNN Classifier using GCRA (0.8689), followed by RF (0.8525) and SVM (0.8361). The remaining methods, GA, GS, and RS, produce less accurate values with greater variability, thereby ensuring that the hyperparameter optimisation subsets chosen by GCRA include more important tuning. Precision values are also consistently higher across GCRA. RF with GCRA achieves the highest precision at 0.8699, though KNN is not far behind at 0.874, as is SVM at 0.8473. Thus, it appears that the GCRA improves the models' ability to reduce false positives compared with other approaches. Although GA sometimes produces very high recall values (e.g., 0.9394 for KNN and SVM), this comes at the cost of lower precision and accuracy. GCRA provides a more balanced recall, with notable performance for DT (0.8197) and LR (0.8525), indicating better generalisation rather than bias toward the positive class. The F1-score, which balances precision and recall, further highlights the advantage of GCRA. The highest F1-scores are mostly achieved

with GCRA, particularly for KNN (0.7999) and DT (0.7997). This confirms that GCRA offers a better trade-off between precision and recall than GA, GS, and RS.

Across all classifications and metrics considered, GCRA is clearly superior to all other methods. Though GA and GS perform well occasionally on recall, they exhibit an imbalance in other metrics. RS has the poorest overall performance across most classifications. This suggests that GCRA is a better approach to hyperparameter optimisation, thereby achieving better classification performance and greater overall ML model robustness.

4.1.2. Performance under 70% Training and 30% Testing

In this section, a comprehensive performance analysis is conducted using a 70% training and 30% test split for the six considered ML classifiers. The evaluation is conducted based on accuracy, precision, recall, and F1-score. Furthermore, the results include comparisons between GCRA with optimised hyperparameters and its non-optimised counterpart, as well as comparisons with GA, GS, and RS.

4.1.2.1. Comparison of GCRA With vs Without Hyperparameters

The 70/30 train-test split, as shown in Table 8, yielded the same overall performance pattern; however, the absolute metric values were lower due to the smaller training data set. Through the KNN classifier, there was again a considerable gain in performance: accuracy went up from 0.7912 to 0.8242, and the F1 score increased from 0.7894 to 0.8230. The RF model has been stabilising in its gains: its accuracy improved from 0.8022 to 0.8132, indicating GCRA's growing ability to maintain the bias-variance trade-off even with fewer training samples. The SVM's performance was slightly but steadily better across all evaluated metrics, further affirming GCRA's applicability in non-linear optimisation.

LR's accuracy increased from 0.7582 to 0.8022, indicating that hyperparameter tuning is essential for helping the model converge and fit well. In the same way, the DT model showed a significant improvement from 0.7253 to 0.7802, whereas NB showed no change in metrics due to its limited tunable parameters. In total, the 70/30 split further supports the view that the GCRA approach is highly efficient and effective, particularly in terms of generalisation capability and overall efficacy. Apart from this, based on results from the smaller data set, this strategy successfully improved the performance of most classifiers, including KNN, RF, SVM, LR, and DT. Therefore, it can be concluded that this strategy is an effective technique for optimising different ML approaches.

In the general pattern holds for both single splits (80/20 and 70/30): scores are usually higher in the

80/20 split because there is more training data. Tuning helps KNN the most in both settings, and RF and SVM improve as well. However, performance drops slightly at the 70/30 split (especially for KNN and RF), indicating that the models are sensitive to smaller training sizes. LR and DT consistently improve when tuned across both splits, whereas NB remains the same in both cases, as expected since it has few tunable hyperparameters.

The confusion matrices in Figures 7 and 8 also support these numerical enhancements, indicating improved classification accuracy and reduced false predictions when GCRA was used.

Table 8. Comparison of ML classifiers without hyperparameters with GCRA as a hyperparameter by using 70% training and 30% testing

Model	Accuracy		Precision		Recall		F1 Score	
	without GCRA	with GCRA	without GCRA	with GCRA	without GCRA	with GCRA	without GCRA	with GCRA
KNN	0.7912	0.8242	0.7925	0.8252	0.7912	0.8242	0.7894	0.823
NB	0.7912	0.7912	0.7925	0.7925	0.7912	0.7912	0.7894	0.7894
SVM	0.7912	0.8022	0.7911	0.8028	0.7912	0.8022	0.7903	0.8009
RF	0.8022	0.8132	0.8028	0.8133	0.8022	0.8132	0.8009	0.8124
LR	0.7582	0.8022	0.7581	0.8137	0.7582	0.8022	0.7567	0.7975
DT	0.7253	0.7802	0.728	0.7823	0.7253	0.7802	0.7259	0.7778

* NB has default parameters

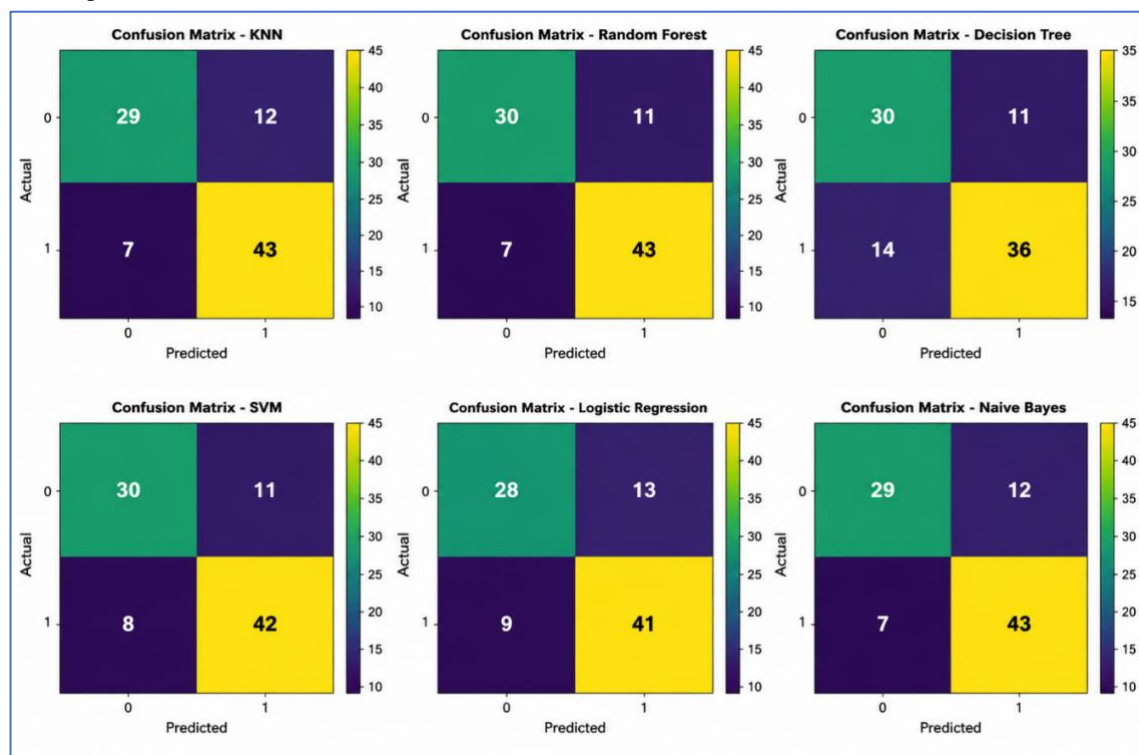


Figure 7. The results of all the confusion matrix ML classifiers without hyperparameters for 70% training and 30% testing

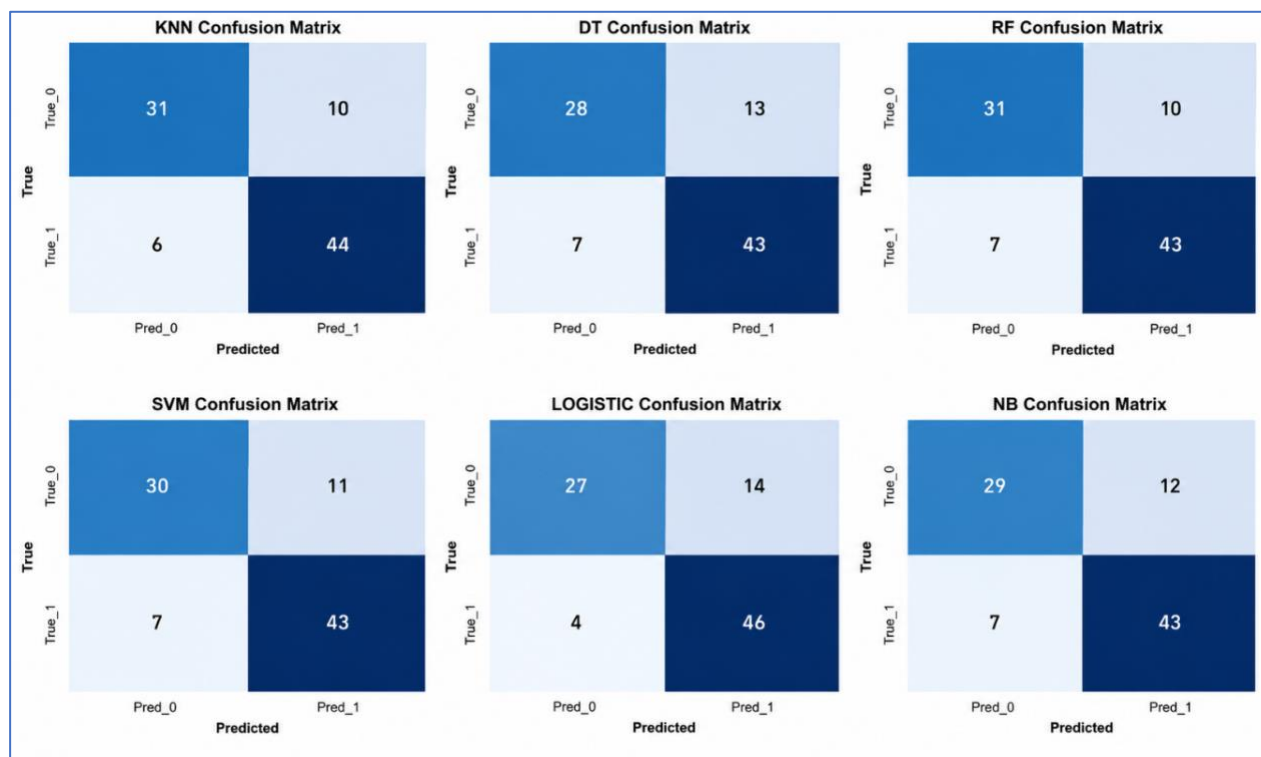


Figure 8. The results of all the confusion matrix ML classifiers with GCRA hyperparameters for 70% training and 30% testing

Table 9. Comparison of GCRA with GA, GS, and RS under 80% training and 20% testing

Models	Accuracy				Precision				Recall				F1-Score			
	GCRA	GA	GS	RS	GCRA	GA	GS	RS	GCRA	GA	GS	RS	GCRA	GA	GS	RS
KNN	0.8689	0.8197	0.8033	0.8197	0.874	0.7750	0.8126	0.8262	0.8689	0.9394	0.8033	0.8197	0.7999	0.8493	0.7997	0.8172
SVM	0.8361	0.8033	0.8197	0.8197	0.8473	0.7561	0.8346	0.8346	0.8199	0.9394	0.8197	0.8197	0.8172	0.8378	0.8154	0.8154
RF	0.8525	0.7869	0.8361	0.7869	0.8699	0.7500	0.8583	0.7992	0.8361	0.9091	0.8361	0.7869	0.8154	0.8219	0.8313	0.7819
LR	0.8197	0.7869	0.8033	0.8033	0.847	0.7500	0.8126	0.8126	0.8525	0.9091	0.8033	0.8033	0.8132	0.8219	0.7997	0.7997
DT	0.7869	0.7541	0.7049	0.7049	0.792	0.7500	0.7053	0.7053	0.8197	0.8182	0.7049	0.7049	0.7997	0.7826	0.7020	0.7020
NB	0.8199	0.8197	0.8197	0.8197	0.8263	0.7895	0.8262	0.8262	0.7869	0.9091	0.8197	0.8197	0.702	0.8451	0.8172	0.8172

* NB has default parameters

Table 10. Comparison of GCRA with GA, GS, and RS under 70% training and 30% testing

Models	Accuracy				Precision				Recall				F1-Score			
	GCRA	GA	GS	RS	GCRA	GA	GS	RS	GCRA	GA	GS	RS	GCRA	GA	GS	RS
KNN	0.8242	0.7582	0.7473	0.7582	0.8252	0.7625	0.7502	0.7625	0.8242	0.7523	0.7423	0.7523	0.823	0.7531	0.7422	0.7531
SVM	0.8022	0.8132	0.8132	0.8132	0.8028	0.8214	0.8214	0.8214	0.8024	0.806	0.806	0.806	0.8009	0.8078	0.8078	0.8078
RF	0.8132	0.8077	0.8077	0.8077	0.8133	0.8128	0.8128	0.8128	0.8132	0.8017	0.8017	0.8017	0.8124	0.8025	0.8025	0.8025
LR	0.8025	0.8022	0.8022	0.8022	0.8137	0.8085	0.8085	0.8085	0.8022	0.797	0.797	0.797	0.7975	0.799	0.799	0.799
DT	0.7802	0.7253	0.7253	0.7363	0.7823	0.7301	0.7301	0.7399	0.7802	0.7213	0.7213	0.7307	0.7778	0.7217	0.7217	0.7314
NB	0.7914	0.7912	0.7912	0.7912	0.7925	0.7937	0.7937	0.7937	0.7912	0.7837	0.7837	0.7837	0.7894	0.7861	0.7861	0.7861

* NB has default parameters

4.1.2.2. Comparison of GCRA with GA, GS, and RS

Table 10 presents a comparative assessment of GCRA with GA, GS, and RS across six classifiers, with 70%-30% training and testing splits. Table 10 shows that GCRA outperforms all classifiers in most cases. For KNN, RF, and DT classifiers, GCRA achieves better accuracy, precision, recall, and F1 scores across all optimisation techniques. For the LR model, GCRA outperforms all with better precision (0.8137), a slight increase in accuracy, and improved recall. However, its F1-score is low at 0.7975, although it is very close to those obtained by other optimisation techniques, 0.799. For SVM classifiers, all optimisation techniques yield almost identical results, because performance is relatively stable with proper hyperparameters.

4.1.3. Naïve Bayes Performance Stability Under GCRA-Based Tuning

Therefore, NB has default parameters, so it offers few continuous, model-specific hyperparameters to tune in this setup, since its parameters are primarily determined directly from the training data under fixed probabilistic assumptions. Across Tables 7-11, the NB classifier shows largely stable performance and minimal sensitivity to GCRA-based tuning. In both the 80/20 and 70/30 splits (Tables 6 and 7), applying GCRA does not change NB's accuracy, precision, recall, or F1-score, which is consistent with the fact that NB has limited effective hyperparameters in the adopted configuration. Consequently, GCRA had no meaningful search space to optimise for NB, and the model's performance remained unchanged.

When NB is compared across different optimisation strategies (Tables 8 and 9), NB classifiers show a visible improvement in all categories under GCRA compared to other optimisation techniques. Its precision is not at its best; however, it is very close to those obtained by other optimisation techniques, 0.7925 and 0.7937. The reported differences are generally small and reflect minor trade-offs between precision and recall rather than systematic gains from hyperparameter optimisation, reinforcing NB's robustness but also indicating restricted room for improvement through metaheuristic tuning in this study.

4.2. Five-Fold Cross Validation

5-fold cross-validation is a common method of assessment in machine learning to estimate how a model will perform on new data. First, the dataset is split into five parts of almost equal size (folds). The model is then trained and tested five times. During each training session, four folds (about 80% of the data) are used for training, and the remaining fold (about 20%) is held out for testing; the test fold changes in each iteration so that each fold is used for evaluation exactly once. Performance is typically summarised

by reporting the mean and standard deviation (Mean $\pm$ SD) across the five runs [24, 28].

Table 11. Five-fold stratified cross-validation results (mean $\pm$ SD) for ML classifiers with and without GCRA-based hyperparameter tuning

Model	Accuracy		Precision		Recall		F1- score	
	With GCRA	Without GCRA	With GCRA	Without GCRA	With GCRA	Without GCRA	With GCRA	Without GCRA
KNN	0.8051 $\pm$ 0.0945	0.7985 $\pm$ 0.0798	0.7914 $\pm$ 0.1095	0.7918 $\pm$ 0.0994	0.8909 $\pm$ 0.0407	0.8727 $\pm$ 0.0332	0.8359 $\pm$ 0.0729	0.8278 $\pm$ 0.0598
NB	0.8084 $\pm$ 0.0780	0.8084 $\pm$ 0.0780	0.8247 $\pm$ 0.1110	0.8247 $\pm$ 0.1110	0.8424 $\pm$ 0.0449	0.8424 $\pm$ 0.0449	0.8300 $\pm$ 0.0607	0.8300 $\pm$ 0.0607
SVM	0.8316 $\pm$ 0.0577	0.8217 $\pm$ 0.0597	0.8074 $\pm$ 0.0685	0.8098 $\pm$ 0.0793	0.9152 $\pm$ 0.0136	0.8909 $\pm$ 0.0550	0.8569 $\pm$ 0.0432	0.8459 $\pm$ 0.0473
RF	0.8218 $\pm$ 0.0671	0.8152 $\pm$ 0.0373	0.8304 $\pm$ 0.0844	0.8132 $\pm$ 0.0624	0.8545 $\pm$ 0.0332	0.8667 $\pm$ 0.0346	0.8409 $\pm$ 0.0533	0.8371 $\pm$ 0.0252
LR	0.8217 $\pm$ 0.0396	0.8416 $\pm$ 0.0501	0.7865 $\pm$ 0.0590	0.8195 $\pm$ 0.0601	0.9333 $\pm$ 0.0254	0.9152 $\pm$ 0.0254	0.8519 $\pm$ 0.0246	0.8638 $\pm$ 0.0385
DT	0.7489 $\pm$ 0.0494	0.7289 $\pm$ 0.0732	0.7599 $\pm$ 0.0574	0.7482 $\pm$ 0.0743	0.7939 $\pm$ 0.0254	0.7636 $\pm$ 0.0498	0.7759 $\pm$ 0.0368	0.7553 $\pm$ 0.0594

Table 11 summarises the five-fold stratified cross-validation results (mean $\pm$ SD) for all classifiers, comparing performance with and without GCRA in Table 5 of this work (e.g., KNN $k = 7$, RF $n_estimators = 26$, DT $max_depth = 6$, SVM $C = 0.153132$, LR $C = 0.01$; NB uses defaults). Using cross-validation gives a more reliable picture than relying on a single train–test split. Overall, adding GCRA improves performance across several modes, most notably SVM and DT, primarily by boosting recall. KNN also benefits, but the improvements are smaller. RF improves slightly, though the gains come with a small precision-recall trade-off. LR increases recall but at the cost of lower precision and reduced overall Accuracy and F1-score, indicating a stronger emphasis on sensitivity. The reported standard deviations also show that stability varies across classifiers, reinforcing the need for cross-validation to make fair and consistent comparisons. LR did not consistently improve under cross-validation, suggesting sensitivity to the selected tuning objective and split variability.

A key limitation is that the reported performance is split-dependent: results can change noticeably with a different random train–test partition. Therefore, metrics from a single (or few) splits may overestimate generalisation, and conclusions about robustness should be interpreted cautiously.

4.3. Time Process Classification

Table 12 shows the average time (in minutes) taken to optimise each classifier using the 80/20 and 70/30 test split configuration. Overall, the trend for GCRA is a steady decline in computational time compared to GA across all classifiers in both test split configurations. For instance, the computational time for SVM, RF, and DT using the 80/20 test split is lower than with the 70/30 test split (i.e., 15.00 minutes vs 17.50 minutes, 7.00 minutes vs 8.00 minutes). The trend is the same for all classifiers using the 70/30

test split. GCRA achieves lower computational time than Lagrange Elementary Optimization (LEO) for most classifiers, particularly LR, NB, and DT, although LEO is faster with SVM and RF. RF and SVM are the most computationally expensive classifiers for both test split configurations, while NB and KNN are the least time-consuming classifiers to optimise using both test split configurations. These trends show that the GCRA achieves a positive trade-off between optimisation efficiency and computational cost, depending on the classifier type. To obtain the following results, this research was implemented in JupyterLab version 4.3.4 of Anaconda on Windows 11 64-bit on a laptop equipped with an Intel Core i5-8350U processor operating at 1.90 GHz and 8 gigabytes of Random Access Memory (RAM).

Table 12. Average optimisation time per classifier (minutes) 80/20, and 70/30 testing

s	80/20			70/30		
	GCRA	GA	LEO	GCRA	GA	LEO
KNN	1.50	1.70	1.60	1.60	1.80	1.70
SVM	15.00	17.50	14.4	14.00	12.60	13.6
LR	5.00	5.50	6.0	5.20	5.70	6.5
NB	1.00	1.10	1.2	1.00	1.10	1.2
DT	7.00	8.00	7.5	8.40	9.40	8.90
RF	10.00	11.00	9.0	11.50	12.70	10.0

4.4. Statistical Significance

The P-value helps determine whether the difference in performance between two groups is significant. A P-value of less than 5% indicates there exists a statistically significant performance difference, whereas a P-value of 5% or more or equal indicates two algorithms do not differ much from each other [31-33].

As shown in Table 13, the p-values for the paired t-tests between GCRA and GA and between GCRA and LEO, using an 80/20 training/test split. Paired t-tests were computed across $n = 30$ independent runs per classifier, using the performance metric values obtained in each run (paired by run index and identical data split). In general, most p-values are below the significance threshold ($p < 0.05$), indicating that the performance difference between the two classifiers is statistically significant in favour of GCRA across most evaluation criteria and metrics. When comparing the performance of GCRA and GA, the table shows that Accuracy, Recall, and F1-score are statistically significant in favour of GCRA across KNN, SVM, LR, NB, and DT classifiers. Also, in some cases of precision, namely KNN and RF classifiers, the p-values are greater than 0.05, indicating non-significant differences. Similarly, when comparing the performance of the classifiers GCRA and LEO, the table shows that most p-values are less than 0.05, especially for SVM, LR,

NB, and RF. Also, in some cases with DT and KNN classifiers, the p-values are slightly greater than 0.05, indicating differences of marginal or no statistical significance. Thus, the results of the paired t-test indicate that the proposed method outperforms the other classifiers.

Table 13. Paired t-test GCRA Vs GA, and Leo using 80 % training, 20% testing

Classifiers	GCRA-GA				GCRA - LEO			
	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
KNN	0.0492	0.0990	0.00705	0.00494	0.0660	0.0627	0.0719	0.0699
SVM	0.030	0.0912	0.01195	0.00206	0.00110	0.00186	0.00036	0.00069
LR	0.0340	0.0970	0.00566	0.00087	0.0003	0.0052	0.0052	0.00015
NB	0.0002	0.0368	0.01222	0.01431	0.0002	0.00012	0.0075	0.0033
DT	0.0328	0.0420	0.0015	0.0171	0.0549	0.0522	0.0589	0.0561
RF	0.0656	0.1199	0.00730	0.00065	0.0055	0.0005	0.0115	0.0099

4.5. Discussion

Overall, the results indicate that GCRA can improve performance for classifiers with tunable, interacting hyperparameters (especially KNN, RF, SVM, and DT), while NB shows limited sensitivity due to its constrained hyperparameter space. GCRA is used for hyperparameter tuning. This tuning spans various ML classifiers; improvements were observed in several areas. Confusion matrices were used to summarise classification outcomes and derive accuracy, precision, recall, and F1-score. The equations for these metrics are presented below. These confirm that GCRA enhances model generalisation. The algorithm efficiently explores the search space and exploits it to identify optimal parameter combinations.

The impact of GCRA varies across models, as certain algorithms respond more favourably than others; consequently, its effectiveness is model-dependent. These algorithms have adjustable, continuous hyperparameters. KNN, RF, SVM, and DT are examples. These models greatly benefit from optimisation. The benefit is larger than probabilistic models see, for example, NB. The latter algorithms have limited tunable parameters. Moreover, the training-to-testing ratio could shift from 80/20 to 70/30. There was a slight drop in absolute performance at the time of the shift. Yet, improvements with GCRA stayed consistent. This means that the algorithm is stable across data splits.

Of all the models tested, KNN and RF showed significant performance improvements. It was a good gain. It seems that GCRA's adaptive search mechanism is beneficial for the algorithms. These algorithms depend on multiple, interdependent hyperparameters. There has also been some enhancement in the

evaluation metrics. GCRA not only improves accuracy but also maintains a good balance between false negatives and false positives. This provides classification performance that can be relied on. Therefore, based on performance results, GCRA tuned with hyperparameters performed better than the non-tuned GCRA. The tuned algorithm achieved higher accuracy and more stable performance because the optimised hyperparameters enabled a better fit between the model and the data. On the other hand, the GCRA without hyperparameters relies on default settings that are much weaker and less consistent than those with hyperparameters. Overall, the results have provided a clear indication of the GCRA with hyperparameters.

GCRA optimises a single performance metric depending on the specific implementation. In one configuration, the objective function is defined as the negative classification accuracy, thereby enabling GCRA to maximise accuracy; in another, it is defined as the negative F1-score, resulting in the maximisation of the F1-score. Model performance is evaluated using five-fold cross-validation, and a separate validation set is used during hyperparameter tuning. Consequently, hyperparameter tuning was performed on the training data only (via cross-validation), and the test set was reserved for final evaluation.

In particular, in the area of healthcare decisions, the problem of false negatives is even more important because withholding treatment from a patient with undiagnosed coronary disease could result in unnecessarily high morbidity and mortality. For these reasons, recall is viewed as more important than accuracy. When dealing with small datasets, such as one containing only 303 samples, randomly dividing the data into training and test sets can significantly affect the model's performance. Different splits may include different samples in both sets, leading the model to learn distinct patterns during each run and to be evaluated on different data points. Consequently, performance metrics such as accuracy may vary considerably across iterations, leading to instability and unreliable estimates of real-world performance.

The paper assesses the performance of GCRA as a hyperparameter optimisation technique for traditional ML models, highlighting deficiencies and limitations in the search space due to only a few tunable hyperparameters for the given tuning definition. Even for some models, such as NB, where the number of continuous hyperparameters is zero or low, the benefits of GCRA, as a global optimisation approach, are diminished. The evaluation without optimisation overheads may also be platform- and budget-dependent, as the authors point out, due to the optimisation approach's stopping criteria and computational budget constraints. The paper has not explored evaluation areas, such as overheads from repeated runs, robustness across multiple datasets, and testing for significance at the repeated-evaluation

level, all of which are equally important for the confirmation process.

Another limitation of this paper is that it is conducted using only the UCI heart disease dataset, which consists of 303 samples. While it is one of the most popularly used benchmarks in the field, its modest size may limit the general applicability of the results reported. In other words, these results are sensitive to the specific data division and may not accurately reflect in larger or more heterogeneous clinical populations. The research's major constraint is that it was conducted in a controlled environment, potentially exaggerating the clinical performance measures. Reliance on a single dataset and a constant evaluation protocol may lead to overly optimistic accuracy and recall estimates due to sampling and dataset biases. While optimised models show improved performance in experiments, this does not guarantee generalisation; careful interpretation of clinical reliability is necessary until verified by independent cohorts.

5. Conclusions and Future Work

This study evaluated the Greater Cane Rat Algorithm (GCRA) as a metaheuristic hyperparameter tuning strategy for heart disease prediction using six machine learning classifiers (KNN, RF, SVM, LR, DT, and NB) on the UCI Heart Disease dataset. Across split-based experiments, GCRA improved performance for most tunable models, with the strongest gains observed for KNN and DT. Under the 80/20 train–test split, KNN accuracy increased from 0.8033 (default settings) to 0.8689 (GCRA-tuned), and its F1-score improved from 0.7997 to 0.8676. RF also improved with tuning, increasing accuracy from 0.8197 to 0.8525 and precision from 0.8470 to 0.8699. Under the 70/30 split, the same overall pattern was observed: KNN accuracy increased from 0.7912 to 0.8242, RF accuracy increased from 0.8022 to 0.8132, and DT accuracy increased from 0.7253 to 0.7802. In contrast, NB remained largely unchanged across “with vs without tuning” comparisons, as expected given its limited effective hyperparameter space in the configuration used. Comparisons against GA, GS, and RS indicate that GCRA often provides a stronger balance across accuracy/precision/recall/F1 than competing methods, particularly for models that are sensitive to interdependent hyperparameters.

Despite these improvements, the study has clear limitations. Results are based on a single, relatively small benchmark dataset (303 samples), and performance estimates can be split-sensitive. Future work should therefore (i) validate the tuned models on external cohorts or additional heart disease datasets, (ii) adopt nested cross-validation to avoid optimistic bias in hyperparameter selection, and (iii) extend

evaluation metrics to include ROC-AUC and calibration measures to reflect clinical decision requirements better. Overall, the findings support GCRA as a practical hyperparameter optimisation approach for ML-based heart disease prediction, particularly for models with continuous or strongly interacting hyperparameters.

Data Availability Statement: The dataset analysed in this study is publicly available in the University of California, Irvine (UCI) Machine Learning Repository. Additional processed data are available from the corresponding author on reasonable request.

Funding: The authors declare that no funding was received for this research.

Declaration of Competing Interest: The authors declare that they have no known competing interests.

Research Involving Human/Animal Participants: This article does not involve any studies conducted by the authors on animals or human participants.

References

- [1] J. Raghunath and S. Kiran, "Investigation and comparison of feature selection techniques with hyper parameter tuning for predicting proactive cardiovascular disease," *Journal of Theoretical and Applied Information Technology*, vol. 103, no. 8, pp. 3335-3348, 2025. [Online]. Available: <https://www.jatit.org/volumes/Vol103No8/22Vol103No8.pdf>.
- [2] J. Joseph and K. Kartheeban, "Visualizing the full spectrum optimization of K-nearest neighbors from data preprocessing to hyperparameter tuning and K-fold validation for cardiovascular disease prediction," *Informatica*, vol. 49, no. 2, 2025, doi: <https://doi.org/10.31449/inf.v49i2.7774>.
- [3] R. Detrano *et al.*, "International application of a new probability algorithm for the diagnosis of coronary artery disease," *The American Journal of Cardiology*, vol. 64, no. 5, pp. 304-310, 1989, doi: [https://doi.org/10.1016/0002-9149\(89\)90524-9](https://doi.org/10.1016/0002-9149(89)90524-9).
- [4] N. Nasution, M. A. Hasan, and F. B. Nasution, "Predicting heart disease using machine learning: An evaluation of logistic regression, random forest, SVM, and KNN models on the UCI heart disease dataset," *IT Journal Research and Development*, vol. 9, no. 2, pp. 140-150, 2025, doi: <https://doi.org/10.25299/itjrd.2025.17941>.
- [5] N. Uddin, M. S. Hossain, M. Z. Mahmud, R. Ferdous, and M. A. Talukder, "HPML-CVD: Hyperparameter tuned machine learning model to predict cardiovascular disease," *Biomedical Materials & Devices*, vol. 4, no. 1, pp. 861-875, 2026, doi: <https://doi.org/10.1007/s44174-025-00325-1>.
- [6] N. R. Hamza and F. J. Al-Ghanim, "Performance comparison of machine learning algorithms in heart disease prediction with enhanced accuracy through hyper parameter tuning," *Journal of Al-Qadisiyah for Computer Science and Mathematics*, vol. 17, no. 2, pp. 60-71, 2025, doi: <https://doi.org/10.29304/jqscm.2025.17.22180>.
- [7] A. Masbakhah, U. Sa'adah, and M. Muslikh, "Heart disease classification using random forest and Fox algorithm as hyperparameter tuning," *Journal of Electronics, Electromedical Engineering, and Medical Informatics*, vol. 7, no. 4, pp. 964-976, 2025, doi: <https://doi.org/10.35882/jeeemi.v7i4.932>.
- [8] F. O. E. Ehtiba, S. F. Elfaitouri, H. S. B. Abdelmula, and A. Elghirani, "Optimization of heart disease prediction using supervised learning with hyperparameter optimization methods," *Journal of Academic Research*, vol. 28, no. 2, pp. 15-26, 2024, doi: <https://doi.org/10.65540/jar.v28i2.672>.

- [9] A. K. Pathak, M. Chaubey, and M. Gupta, "Randomized-grid search for hyperparameter tuning in decision tree model to improve performance of cardiovascular disease classification," doi: <https://doi.org/10.48550/arXiv.2411.18234>.
- [10] N. Chandrasekhar and S. Peddakrishna, "Enhancing heart disease prediction accuracy through machine learning techniques and optimization," *Processes*, vol. 11, no. 4, p. 1210, 2023, doi: <https://doi.org/10.3390/pr11041210>.
- [11] A. Mansoori, M. Zeinalnezhad, and L. Nazarimanesh, "Optimization of tree-based machine learning models to predict the length of hospital stay using genetic algorithm," *Journal of Healthcare Engineering*, vol. 2023, p. 9673395, 2023, doi: <https://doi.org/10.1155/2023/9673395>.
- [12] A. Bahrami, M. Rakhshaninejad, R. Ghousi, and A. Atashi, "Enhancing machine learning performance in cardiac surgery ICU: Hyperparameter optimization with metaheuristic algorithm," *PLOS ONE*, vol. 20, no. 2, p. e0311250, 2025, doi: <https://doi.org/10.1371/journal.pone.0311250>.
- [13] Y. A. Ali, E. M. Awwad, M. Al-Razgan, and A. Maarouf, "Hyperparameter search for machine learning algorithms for optimizing the computational complexity," *Processes*, vol. 11, no. 2, p. 349, 2023, doi: <https://doi.org/10.3390/pr11020349>.
- [14] G. Narasimhan and A. Victor, "A hybrid approach with metaheuristic optimization and random forest in improving heart disease prediction," *Scientific Reports*, vol. 15, p. 10971, 2025, doi: <https://doi.org/10.1038/s41598-024-73867-x>.
- [15] S. Paliwal, S. Praveen, M. A. Alam, and J. Ahmed, "Optimizing heart disease prediction models using genetic algorithms: A metaheuristic approach," *Journal of Theoretical and Applied Information Technology*, vol. 102, no. 9, pp. 3868-3881, 2024. [Online]. Available: <https://www.jatit.org/volumes/Vol102No9/14Vol102No9.pdf>.
- [16] J. O. Agushaka, O. Akinola, A. E. Ezugwu, and O. N. Oyelade, "A novel binary greater cane rat algorithm for feature selection," *Results in Control and Optimization*, vol. 11, p. 100225, 2023, doi: <https://doi.org/10.1016/j.rico.2023.100225>.
- [17] M. Rahardi, B. P. Asaddulloh, A. Aminuddin, F. F. Abdulloh, I. Saifudin, and F. P. Kusumawijaya, "Optimizing machine learning models for class imbalance in heart disease prediction," *Engineering, Technology & Applied Science Research*, vol. 15, no. 3, pp. 23599-23604, 2025, doi: <https://doi.org/10.48084/etasr.10407>.
- [18] M. P. Rodriguez and M. Nafea, "Centralized and federated heart disease classification models using UCI dataset and their Shapley-value based interpretability," doi: <https://doi.org/10.48550/arXiv.2408.06183>.
- [19] A. A. Maulani, S. Winarno, J. Zeniarja, R. T. E. Putri, and A. N. Cahyani, "Comparison of hyperparameter optimization techniques in hybrid CNN-LSTM model for heart disease classification," *Sinkron: Jurnal dan Penelitian Teknik Informatika*, vol. 8, no. 1, pp. 455-465, 2024, doi: <https://doi.org/10.33395/sinkron.v9i1.13219>.
- [20] A. M. Aladdin and T. A. Rashid, "LEO: Lagrange elementary optimization," *Neural Computing and Applications*, vol. 37, no. 19, pp. 14365-14397, 2025, doi: <https://doi.org/10.1007/s00521-025-11225-2>.
- [21] H. S. Abdulla, A. A. Ameen, S. I. Saeed, I. A. Mohammed, and T. A. Rashid, "MRSO: Balancing exploration and exploitation through modified rat swarm optimization for global optimization," *Algorithms*, vol. 17, no. 9, p. 423, 2024, doi: <https://doi.org/10.3390/a17090423>.
- [22] I. Gupta, A. Bajaj, M. Malhotra, V. Sharma, and A. Abraham, "Heart disease prediction using a hybrid feature selection and ensemble learning approach," *IEEE Access*, vol. 13, pp. 111926-111937, 2025, doi: <https://doi.org/10.1109/ACCESS.2025.3583757>.
- [23] H. Li, "Enhancing cardiovascular disease prediction with machine learning: A comparative study using the UCI heart disease dataset," presented at the Proceedings of the 2nd International Conference on Data Analysis and Machine Learning (DAML 2024), 2025, doi: <https://doi.org/10.5220/0013516000004619>.
- [24] A. A. Hassan, R. K. Muhammed, T. Y. Mawlood, A. M. Aladdin, B. A. Hassan, and T. A. Rashid, "Addressing missing data in machine learning: A comparative insights on imputation techniques and classification accuracy," presented at the Proceedings of the 2025 International Conference on Next Generation Information System Engineering (NGISE), 2025, doi: <https://doi.org/10.1109/NGISE64126.2025.11085273>.
- [25] M. N. Uddin and R. K. Halder, "An ensemble method based multilayer dynamic system to predict cardiovascular disease

- using machine learning approach," *Informatics in Medicine Unlocked*, vol. 24, p. 100584, 2021, doi: <https://doi.org/10.1016/j.imu.2021.100584>.
- [26] S. Celik, B. Doganli, M. U. Sasmaz, and U. Akkucuk, "Accuracy comparison of machine learning algorithms on World Happiness Index data," *Mathematics*, vol. 13, no. 7, p. 1176, 2025, doi: <https://doi.org/10.3390/math13071176>.
- [27] T. Hassan, M. Majeed, and M. Ahmad, "Optimizing SVM performance through combinatorial hyperparameter tuning and model selection," *International Journal Bioautomation*, vol. 29, no. 2, pp. 117-144, 2025, doi: <https://doi.org/10.7546/ijba.2025.29.2.000981>.
- [28] T. Y. Mawlood, A. A. Hassan, R. K. Muhammed, A. M. Aladdin, T. A. Rashid, and B. A. Hassan, "Improving cardiovascular disease prediction through stratified machine learning models and combined datasets," *UHD Journal of Science and Technology*, vol. 9, no. 1, pp. 149-168, 2025, doi: <https://doi.org/10.21928/uhdjst.v9n1y2025.pp149-168>.
- [29] G. V. R. Ram, P. Jayanth, M. H. Kumar, S. Mundody, and R. M. R. Guddeti, "A meta-heuristic approach for optimizing neural network model for heart disease prediction," in *Proceedings of the 35th Conference of Open Innovations Association FRUCT*, Tampere, Finland, Apr. 24–26, 2024, pp. 825–830.
- [30] J. O. Agushaka, A. E. Ezugwu, A. K. Saha, J. Pal, L. Abualigah, and S. Mirjalili, "Greater cane rat algorithm (GCRA): A nature-inspired metaheuristic for optimization problems," *Heliyon*, vol. 10, no. 11, p. e31629, 2024, doi: <https://doi.org/10.1016/j.heliyon.2024.e31629>.
- [31] R. M. D. Omer, N. K. Al-Salihi, T. A. Rashid, A. M. Aladdin, M. Mohammadi, and J. Majidpour, "Artificial cardiac conduction system: Simulating heart function for advanced computational problem solving," doi: <https://doi.org/10.48550/arXiv.2404.02907>.
- [32] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80-83, 1945, doi: <https://doi.org/10.2307/3001968>.
- [33] A. A. H. Amin, A. M. Aladdin, D. O. Hasan, S. R. Mohammed-Taha, and T. A. Rashid, "Enhancing algorithm selection through comprehensive performance evaluation: Statistical analysis of stochastic algorithms," *Computation*, vol. 11, no. 11, p. 231, 2023, doi: <https://doi.org/10.3390/computation11110231>.