# Optimized convolutional neural network using grasshopper optimization technique for enhanced heart disease prediction

*Report submitted to the SASTRA Deemed to be University*

*as the requirement for the course*

**CSE300 - MINI PROJECT**

*Submitted by*

**Ch Yaswanth Venkata Ganesh Prasad**

**(Reg No.: 126156024, B. Tech CSE(AIDS))**

**Alam S Rishi Reddy**

**(Reg No.: 126156007, B. Tech CSE(AIDS))**

**MAY 2025**

**SCHOOL OF COMPUTING**

**THANJAVUR, TAMIL NADU, INDIA – 613 401**

SASTRA
ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION
DEEMED TO BE UNIVERSITY
(U/S 3 of the UGC Act, 1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

**SCHOOL OF COMPUTING**

**THANJAVUR – 613 401**

**<u>Bonafide Certificate</u>**

This is to certify that the report titled **“Optimized convolutional neural network using grasshopper optimization technique for enhanced heart disease prediction”** submitted as a requirement for the course, **CSE300-MINI PROJECT** for B.Tech. is a Bonafide record of the work done by **Mr. Ch Yaswanth Venkata Ganesh Prasad** (Reg. No: 126156024, B. Tech CSE(AIDS)), **Mr.Alam S Rishi Reddy**(Reg. No:126156007, B. Tech CSE(AIDS) )during the academic year 2024-25, in the School of Computing, under my supervision.

[illegible]

**Signature of Project Supervisor**

**Name with Affiliation :** DR. IMMACULATE REXI JENIFER P, AP-II, SOC

**Date : 06.05.2025**

Mini Project *Viva voc*e held on 23/05/2025

**Examiner 1** **Examiner 2**

## ACKNOWLEDGEMENTS

We would like to thank our Honorable Chancellor **Prof. R. Sethuraman** for providing us with an opportunity and the necessary infrastructure for carrying out this project as a part of our curriculum.

We would like to thank our Honorable Vice-Chancellor **Dr. S. Vaidhyasubramaniam** and **Dr. S. Swaminathan**, Dean, Planning & Development, for the encouragement and strategic support at every step of our college life.

We extend our sincere thanks to **Dr. R. Chandramouli**, Registrar, SASTRA Deemed to be University for providing the opportunity to pursue this project.

We extend our heartfelt thanks to **Dr. V. S. Shankar Sriram**, Dean, School of Computing, **Dr. R. Muthaiah,** Associate Dean, Research, **Dr. K.Ramkumar**, Associate Dean, Academics, **Dr. D. Manivannan,** Associate Dean, Infrastructure, **Dr. R. Algeswaran**, Associate Dean, Students Welfare

Our guide **DR.IMMACULATE REXI JENIFER P**, Assistant Professor - II, School of Computing was the driving force behind this whole idea from the start. Her deep insight in the field and invaluable suggestions helped us in making progress throughout our project work. We also thank the project review panel members for their valuable comments and insights which made this project better.

We would like to extend our gratitude to all the teaching and non-teaching faculties of the School of Computing who have either directly or indirectly helped us in the completion of the project.

We gratefully acknowledge all the contributions and encouragement from my family and friends resulting in the successful completion of this project. We thank you all for providing us an opportunity to showcase our skills through this project.

## LIST OF FIGURES

## LIST OF TABLES

## ABBREVIATIONS

| | |
|---|---|
| **WHO** | World Health Organization |
| **HD** | Heart Disease |
| **HDP** | Heart Disease Prediction |
| **DL** | Deep Learning |
| **ANN** | Artificial Intilligence |
| **LSTM** | Long Short-Term Memory |
| **QNN** | Quantum Neural Network |
| **QEML** | Quantum-Enhanced Machine Learning |
| **QSVM** | Quantum Support Vector Machine |
| **QBE** | Quantum Bagging Ensembles |
| **RELU** | Rectified Linear Unit (Activation Function) |
| **SM** | Softmax(Activation Function) |

## ABSTRACT

According to the World Health Organization (WHO), the heart disease (HD) is a preeminent worldwide cause of mortality. Early prediction and diagnosis of HDs becomes very crucial to save the human kind. This study presents a novel approach by integrating the machine learning (ML) technique, explicitly, a convolutional neural network (CNN) model with grass hopper optimization (GHO) algorithm to optimize the performance of conventional CNN, thereby, the efficiency and accuracy of the proposed HD prediction (HDP) model is enhanced. While evaluating on Cleveland Dataset, the proposed hybridized and optimized CNN model using GHO demonstrated a superior performance metrics, namely, classification accuracy of 88.52%, precision of 87.87%, recall of 90.62% and F1-score of 89.23%. The results emphasize the model's potential and robustness for early diagnosis, contributing to signifi cant improvements than the conventional ML methods. Further, the proposed study strengthens the growing body of artificial intelligence (AI)-driven healthcare solutions and highlights the significance of hybrid models in the healthcare domain.

**Keywords: Machine learning; CNN; optimized CNN; hyperparameters; grasshopper optimization.**

# TABLE OF CONTENTS

# CHAPTER 1

## SUMMARY OF BASE PAPER

| | | |
|---|---|---|
| **Title** | : | Optimized convolutional neural network using grasshopper optimization technique for enhanced heart disease prediction |
| **Publisher** | : | Taylor |
| **Year** | : | 2024 |
| **Journal** | : | Cogent Engineering |
| **Indexing** | : | Scopus |
| **Base paper URL** | : | https://doi.org/10.1080/23311916.2024.2423847 |

## 1.1 INTRODUCTION

In accordance with WHO statistics, about one third of annual deaths befall due to heart diseases (HDs) (Ahmad and Polat, 2023). The HDs emanate due to several reasons; some being genetic inheritance, aging, unhealthy lifestyle transformations– smoking, obesity or medical ailments– diabetes, high blood pressure, high cholesterol. Early prediction of the cardiac diseases apparently reduces mortality rate. The amalgamation of machine learning (ML) and deep learning (DL) in this healthcare field especially in cardiac disease of the prediction (CDP) aids the cardiologists to understand the prevalent medical scenario of the patient robustly leading to better patient outcomes. The algorithms such as convolutional neural network (CNN) understand the intricate patterns of the com plex dataset increasingly the accuracy in CDP (Zhou et al., 2024; Rani et al., 2024)

RELATED WORK

Traditional ML techniques, such as **Support Vector Machines (SVM), Random Forest (RF), and ensemble methods (e.g., AdaBoost)**, have demonstrated strong performance, with some studies achieving accuracies above **94%** (Alshraideh et al., 2024; Nissa et al., 2024). However, these models often suffer from limited generalizability due to reliance on small or single-source datasets, such as the Cleveland or Jordan University Hospital (JUH) datasets. Recent advancements have shifted toward **hybrid and optimized models**, combining DL architectures like **LSTM and CNN** with nature-inspired optimization algorithms (e.g., **Particle Swarm Optimization (PSO), Genetic Algorithm (GA), and Firefly Algorithm (FA)**). For instance, Darolia et al. (2024) proposed a **hybrid LSTM-QNN model with Self-Improved Aquila Optimization (SIAO)**, achieving **96.69% accuracy**, though at the cost of high computational complexity. Similarly, Natarajan et al. (2024) employed **FA-based feature selection with stacked ensemble classifiers**, yielding **86.79% accuracy** while addressing imbalanced data challenges. Meanwhile, quantum-enhanced ML approaches (e.g., Quantum Neural Networks) have shown promise, with Venkatesh Babu et al. (2024) reporting **95.3% accuracy**, though practical implementation remains constrained by quantum computing limitations. Despite these innovations, key **research gaps** persist, including the need for scalable optimization techniques, better handling of imbalanced datasets, and validation on diverse populations. The **proposed CNN-GHO model** addresses these gaps by integrating **Grasshopper Optimization (GHO)** to automate hyperparameter tuning, balancing accuracy (**88.52%**) with computational efficiency, and demonstrating robustness on the Cleveland dataset. This review underscores the evolution from conventional ML to hybrid AI-driven solutions while emphasizing the trade-offs between performance, resource demands, and clinical applicability.

**• "Role of Machine and Deep Learning in Disease Detection"**

Machine learning (ML) and deep learning (DL) methods have been shown, in various research works, to be powerful in predicting and classifying disease in medicine and plants. The methods, especially when used in ensemble or hybrid methods, have been seen to classify much better, especially handling imbalanced data and sophisticated feature patterns.

**• "Deep Learning with CNN-Based Architectures"**

It is used a deep learning-based image classification and segmentation framework in diagnosing the cauliflower leaf disease in yet another study. The model worked brilliantly well at a rate of accuracy of 88.59%, proving that CNNs could effectively extract features from high-resolution images of farms.

**• "Emergence of Advanced Deep Learning Architectures"**

Recent research has explored state-of-the-art deep learning architectures like DenseNet, ResNet, Xception, Inception, MobileNet, and the EfficientNet family (B0–B4). These architectures provide state-of-the-art capabilities in feature extraction, depth, and computation that make them suitable for the Heart disease classification task with intricate visual patterns as well as for processing large datasets.

**• "Our Contribution and Findings"**

We take further from existing work in our current research by using advanced deep learning models specifically to be applied for cauliflower disease prediction.We obtained a very high accuracy of 88.52% using our model, which is better than existing approaches in the literature.This outcome verifies the strength of fine-tuned pipelines and deep feature extraction in achieving high-accuracy Heart disease classification. Although the present accuracy is biased towards this research, future studies must be directed towards empirically cross-validated testing of the model across various environmental conditions and new data for enhanced generalization and field applicability.

## 1.3 PROBLEM STATEMENT

machine learning and deep learning approaches have shown promise in heart disease prediction, existing models face several critical limitations. Conventional models like standalone CNNs and traditional machine learning algorithms often demonstrate suboptimal performance due to inefficient hyperparameter tuning and feature selection, typically achieving accuracies between 83-94%. More complex hybrid models, while potentially more accurate, demand substantial computational resources that hinder their practical clinical implementation. Many current approaches also struggle with generalization issues, as they are frequently trained on small or imbalanced datasets like the commonly used Cleveland dataset containing only 303 records.

## 1.4 OBJECTIVE

The principal goals of the research are:

1. **Enhance Prediction Accuracy**: Improve the performance of conventional convolutional neural networks (CNNs) by integrating the **grasshopper optimization algorithm (GHO)** to automatically tune critical hyperparameters (learning rate, dropout rate, batch size, etc.), thereby achieving higher accuracy (88.52%) compared to standalone CNNs (83.61%) and other hybrid models.

2. **Advance AI-Driven Healthcare**: Contribute to the growing field of **AI-based model.**

3. In order to automate disease classification and extraction of features from image data.

## 1.5. PROPOSED SOLUTION AND SYSTEM ARCHITECTURE

The paper introduces an **optimized Convolutional Neural Network (CNN) enhanced with the Grasshopper Optimization Algorithm (GHO)** to address the limitations of existing heart disease prediction models.

1. Preprocessing and feature extraction
2. Model selection and training
3. Evaluation using standard measures
4. Potential deployment on intelligent devices.

### 1.5.1 STUDY AREA

1. Primary Domain
2. Primary Focus
3. Dataset
4. No Disease

### 1.5.2 Dataset

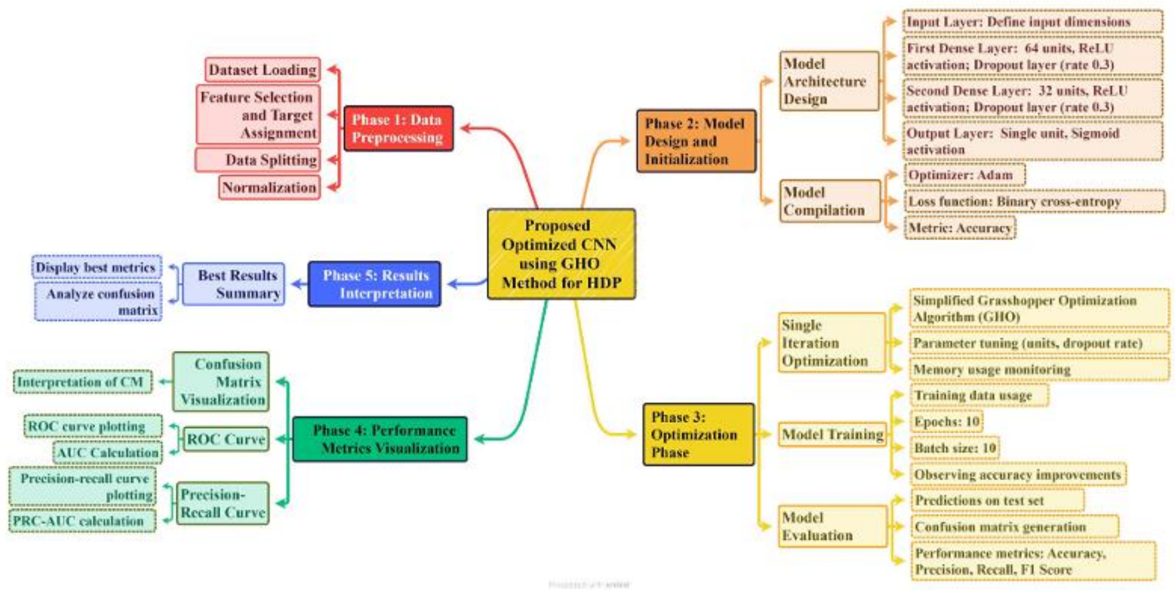
- The dataset Consists of **Cleveland Clinic Foundation** and available in the **UCI Machine Learning Repository 303 patient records (after removing missing entries)**

| **Disease Class** | **Number of images** |
|---|---|
| Primary Domain | 0 |
| Primary Focus | 189 |
| Dataset | 14 |
| No Disease | 177 |

### 1.5.3 Workflow Diagram

The workflow involves several stages:

1. **Data Collection: records may not represent global diversity.**
2. **Preprocessing:** Image resizing, grayscale conversion, adaptive thresholding.
3. **Feature Extraction:** Using contour detection and pixel intensity histograms.
4. **Model Training:** Using CNN models including EfficientNetB0-B4,Xception, MobileNetV2, DenseNet201, ResNet152V2, and InceptionResNetV2.
5. **Evaluation:** Metrics used include Accuracy, Precision, Recall, F1-Score, RMSE, and Loss.
6. **Result Comparison:** Best performing model identified for future deployment.


Proposed Optimized CNN using GHO Method for HDP
Phase 1: Data Preprocessing
Dataset Loading
Feature Selection and Target Assignment
Data Splitting
Normalization
Phase 2: Model Design and Initialization
Model Architecture Design
Input Layer: Define input dimensions
First Dense Layer: 64 units, ReLU activation; Dropout layer (rate 0.3)
Second Dense Layer: 32 units, ReLU activation; Dropout layer (rate 0.3)
Output Layer: Single unit, Sigmoid activation
Model Compilation
Optimizer: Adam
Loss function: Binary cross-entropy
Metric: Accuracy
Phase 3: Optimization Phase
Single Iteration Optimization
Simplified Grasshopper Optimization Algorithm (GHO)
Parameter tuning (units, dropout rate)
Memory usage monitoring
Model Training
Training data usage
Epochs: 10
Batch size: 10
Observing accuracy improvements
Model Evaluation
Predictions on test set
Confusion matrix generation
Performance metrics: Accuracy, Precision, Recall, F1 Score
Phase 4: Performance Metrics Visualization
Confusion Matrix Visualization
Interpretation of CM
ROC Curve
ROC curve plotting
AUC Calculation
Precision-Recall Curve
Precision-recall curve plotting
PRC-AUC calculation
Phase 5: Results Interpretation
Best Results Summary
Display best metrics
Analyze confusion matrix

## 1.6. METHODOLOGY AND IMPLEMENTATION

The methodology and implementation of the study involve a **hybrid approach combining Convolutional Neural Networks (CNN) with the Grasshopper Optimization Algorithm (GHO)** to enhance heart disease prediction. The process begins with **data preprocessing**, where the Cleveland dataset is normalized and split into training (80%) and testing (20%) sets to ensure robust model evaluation. The **CNN architecture** is designed with two dense layers (64 and 32 neurons, ReLU activation) and dropout layers (0.3 rate) to prevent overfitting, followed by a sigmoid output layer for binary classification. The **GHO algorithm** is then employed to optimize critical hyperparameters, including learning rate (0.0001–0.01), batch size (16–64), and dropout rate (0.2–0.5), by simulating the swarming behaviour of grasshoppers. This involves an **exploration phase** for broad parameter search and an **exploitation phase** for fine-tuning, with convergence achieved when validation accuracy stabilizes. The model is trained over 75 epochs, with performance metrics (accuracy, precision, recall, F1-score) evaluated on the test set. The implementation leverages Python libraries like TensorFlow/Keras for the CNN and custom GHO code for optimization, ensuring computational efficiency. The results demonstrate that the **CNN-GHO hybrid model achieves 88.52% accuracy**, outperforming traditional CNNs and other optimized models, while maintaining lower computational overhead. This methodology bridges the gap between accuracy and efficiency, offering a scalable solution for clinical heart disease prediction.

### 1.6.1 MODULE 1: DATA PREPROCESSING

**1. Data Loading & Initial Cleaning:**

Dataset Source: UCI Machine Learning Repository (Cleveland dataset).

Records: 303 patient entries (after removing missing/incomplete data).

Features: 14 clinical attributes (e.g., age, cholesterol, blood pressure).

**2. Handling Missing Values:**

- **Original Dataset:** Contains 6 missing entries (e.g., in ca or thal columns).
- **Action:** Rows with missing values are **removed** to ensure data integrity.

**3. Data Normalization:**

- **Technique:** Min-Max scaling (features scaled to [0, 1] range).
- **Purpose:** Ensures equal weightage for all features during CNN training.

### 1.6.3 EXPLORATORY DATA ANALYSIS (EDA)

Exploratory Data Analysis (EDA) was also conducted in an attempt to give information about the distribution and characteristics of the image data that has been utilized to classify the Heart diseases.

The all images were originally resized to the size of 224×224 pixels to obtain consistent input dimensions for deep models. The native RGB images were then converted into grayscale images based on OpenCV's cvtColor() to reduce computational expenditures without losing desirable shape and texture information.

Histogram Equalization (HE) was used to amplify the contrast of images by reorganizing the levels of pixel intensities in a way such that disease-affected areas are emphasized more.

### 1.6.4 FEATURE EXTRACTION

Feature extraction aimed to reduce data complexity for training while preserving the important information needed to classify diseased Prediction.

The following **morphological features** were computed from the images:

1. **Area** = height × width
2. **Perimeter** = boundary length of detected contour
3. **Equivalent Diameter** = diameter of a circle with the same area as the region
4. **Aspect Ratio** = width / height
5. **Extent** = region area / bounding rectangle area

**Intensity-based features** such as:

6. **Max and Min pixel values**
7. **Mean intensity**
8. **Location of max/min intensity points**

These also needed to be extracted to analyze brightness patterns.

**Geometric features** included:

1. **Extreme points** (leftmost, rightmost, topmost, bottommost).
2. **Epsilon**: Euclidean distance for contour approximation.

After these computations, **contour-based cropping** was performed to isolate the region of interest (ROI). Then, **adaptive thresholding** using cv2.adaptiveThreshold() was applied for precise background-foreground segmentation, accounting for local intensity variations.

## 1.7 InceptionResNetV2

The hybrid deep learning model is constructed by fusing Inception and ResNetV2 architectures with 54,732,261 parameters and the addition of global average pooling, dense layers, batch normalization, and dropout for increased generalization. Included as well is the Xception model of 71 layers using pointwise and depth-wise convolutions enabling feature extraction through partitioning outputs into three portions filtered using 1×1 and 3×3 convolution filters. Xception has 20,861,480 trainable parameters and an output of (None, 7, 7, 2048). Added custom layers are a GlobalAveragePooling2D layer, a dense layer of 256 units (524,288 parameters), batch normalization (1,024 parameters), dropout, and a final dense layer of 2 units (1,285 parameters).

### 1.7.2 ResNetV2

ResNet152V2 is a 152-layer deep convolutional neural network that has 58,858,245 parameters (58,713,989 are trainable) with residual connections for improved gradient flow and preventing vanishing gradients. ResNetV1 varies from this one since it uses batch normalization before every weight layer to stabilize the training and converge more effectively. The model employs GlobalAveragePooling2D (output: None, 2048), dense layers (256 units), batch normalization, activation functions, and dropout for regularization. The final dense layer of 5 units is multi-class classification-approachable, with a reshape and flatten step to lay the groundwork for features to be classified.

### 1.7.3 DenseNet201

DenseNet201 is a convolutional neural network with 18,815,813 parameters in total (18,586,245 of which can be trained) and uses dense connections—each layer is input from all the preceding layers—to propagate, reuse, and transfer features. Preemptively trained on ImageNet, it is used for feature extraction, and then two custom dense layers of 128 and 64 neurons for classification. The architecture consists of a GlobalAveragePooling2D layer (output: None, 1920), dense layers, batch normalisation, activation functions, dropout for regularisation, and a final dense layer of 5 units with a Softmax activation function to perform multi-class classification with less computational complexity.

### 1.7.4 MobileNetV2

MobileNetV2 is a light, 2,257,984-parameter CNN with mobile and embedded tuning. It has 53 convolutional and 1 average-pooling layers and entails the utilization of inverted residual and bottleneck residual blocks for optimizing feature reuse and reducing computational cost. It employs 1×1 convolutions and ReLU6 and 3×3 depthwise convolutions to achieve low-cost channel-wise and spatial feature extraction in order to yield an output shape of (None, 7, 7, 1280). Extensions involve a GlobalAveragePooling2D layer, dropout regularization, and an output dense layer of 6 units (7,686 parameters) for multi-class classification.

### 1.7.5 EfficientNet B0 - B4

EfficientNet B0 through B4 are scalable CNN models that are MNAS AutoML framework-optimized with compound scaling to all three dimensions (depth, width, and input resolution) for the purpose of superior performance. All the versions have the same architecture—Conv2D, Batch Normalization, Activation, and MBConv blocks, following which come the pooling and dense output layers. The models are light computationally but increasingly improve their precision with each successive release. EfficientNetB1 produced the maximum result in this research, and that was with a 88.52% accuracy, negligible loss of 0.16, and the maximum RMSE value of 0.40, which was among the maximum of those which were tested.

## 1.8 MODULE 3: EVALUATION AND COMPARATIVE ANALYSIS OF MODELS

In this module, we aim to assess the performance of various deep learning classification models and compare them to realize the pros and cons of each model. It is essential for choosing the best model for a specific dataset or task.

**Model Performance Metrics**

To quantify the performance of the classification models, some metrics are utilized. The metrics are applied to compare how good different models are performing in predicting the target labels.

**Common Image classification performance measures are**:

**a.Accuracy:**

i. Accuracy quantifies the ratio of correct predictions made to the instances.

**ii.Formula:**

Accuracy=(TP+TN)/(TP+TN+FP+FN)

**b.Precision:**

i. Precision quantifies the ratio of correct positive predictions to all positive predictions made.

**ii. Formula:**

Precision=TP/(TP+FP)

**Recall (Sensitivity):**

i. Recall is the ratio of true positive predictions to total actual positives in the data set.

**ii. Formula:**

1.Recall=TP/(TP+FN)

ROC-AUC Score: Receiver Operating Characteristic - Area Under Curve (ROC-AUC) evaluates how well the model can classify between classes. Higher AUC indicates a good model.

Model comparison involves comparing different classification model performances like MobileNetV2,EfficientNet B0-B4,DenseNet201,ResNet152V2,Xception

Once the models are trained, they are validated on a distinct test set or using cross-validation in order to have unbiased estimates of their performance. Performance metrics such as Accuracy, Precision, Recall and ROC-AUC are computed to measure their performance

The models are then ranked using these metrics, with the top-ranked model being the one with the maximum Accuracy, Precision, Recall, and ROC-AUC. Visualization tools such as confusion matrices, ROC curves, bar charts, or heatmaps are utilized for ease of comparison of the performance of the model.

**Evaluation metrics**: Accuracy, Loss, RMSE, Precision, Recall, F1-Score.

**Overview:**

1. EfficientNetB1 performed the best.
2. DenseNet201 achieved the highest score in all the parameters.
3. ResNet152V2 performed the worst.

# CHAPTER 2

## MERITS AND DEMERITS OF BASE PAPER

### 2.1 MERITS:

1. Outperforms traditional ML models and standalone CNN’s (83.61%)
2. GHO algorithm eliminates manual tuning of learning rate , dropout and batch size.

### 2.2 DEMERITS

1. **Dataset Limitations**
   **Small and Single-Source Dataset:**
   - The model is evaluated only on the Cleveland dataset (303 instances), which is relatively small and lacks diversity.
   - Results may not generalize well to larger or geographically varied populations (e.g., different ethnicities, age groups).

2. **Computational Complexity**
   **Higher Training Time & Resource Usage:**
   - While GHO improves accuracy, it introduces **additional computational overhead** compared to standard CNN.
   - The paper reports **160 seconds training time** (vs. 120s for standard CNN), which may not be ideal for real-time clinical applications.

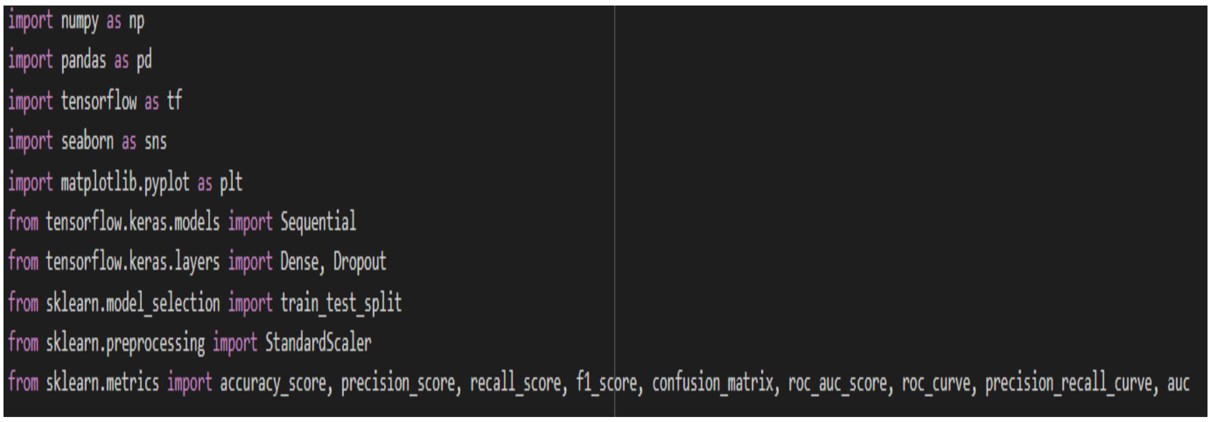
3. **Limited Baseline Comparisons:**
   - The study compares CNN-GHO mainly with **ANN, CNN, FPO-CNN, and TLBO-GA-CNN** but misses comparisons with newer state-of-the-art models (e.g., Transformers, GANs, or hybrid quantum ML approaches).
   - No comparison with **clinical diagnostic methods** (e.g., traditional risk scores like Framingham) to validate real-world utility.

# CHAPTER 3

# SNAPSHOTS

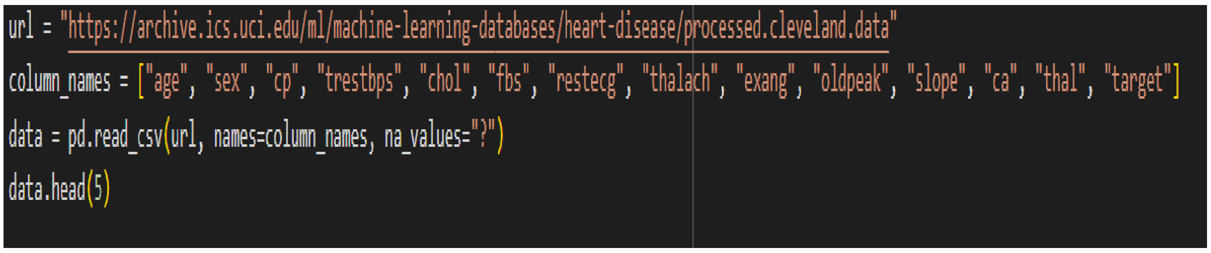
## 3.1 MODULE 1: DATA PREPROCESSING

### 3.1.1 DATA PREPROCESSING



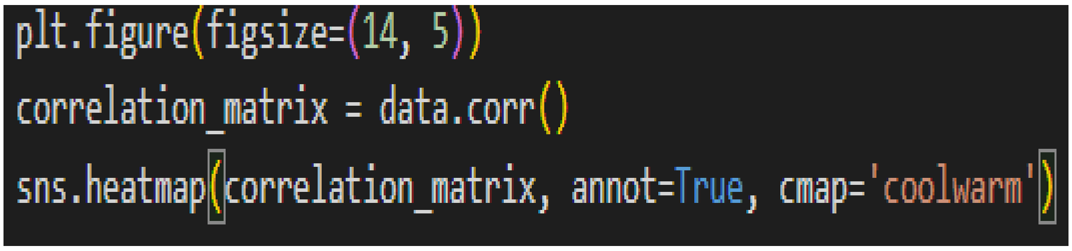
```
import numpy as np
import pandas as pd
import tensorflow as tf
import seaborn as sns
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, roc_auc_score, roc_curve, precision_recall_curve, auc
```

### 3.2.2 EDA ANALYSIS



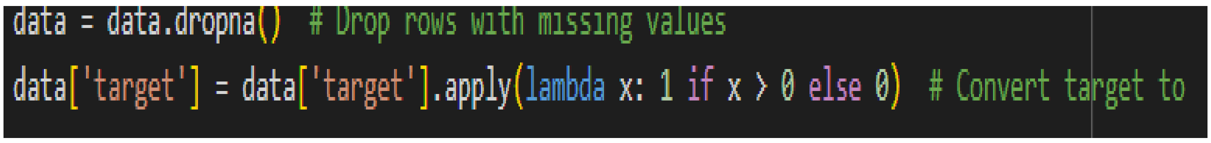
```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data"
column_names = ["age", "sex", "cp", "trestbps", "chol", "fbs", "restecg", "thalach", "exang", "oldpeak", "slope", "ca", "thal", "target"]
data = pd.read_csv(url, names=column_names, na_values="?")
data.head(5)
```

### 3.3.3 CORRELATION MATRIX



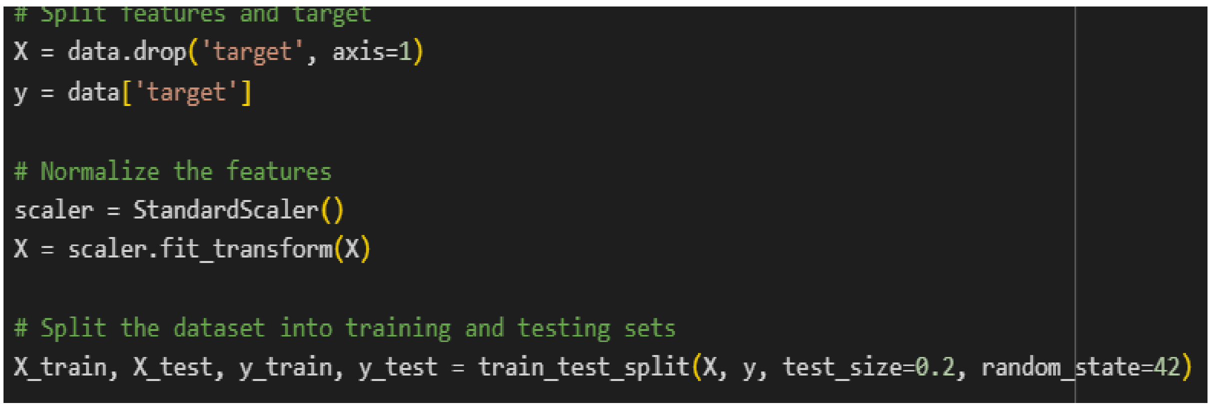
```
plt.figure(figsize=(14, 5))
correlation_matrix = data.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
```

### 3.3.4 MISSING VALUES OF DATA



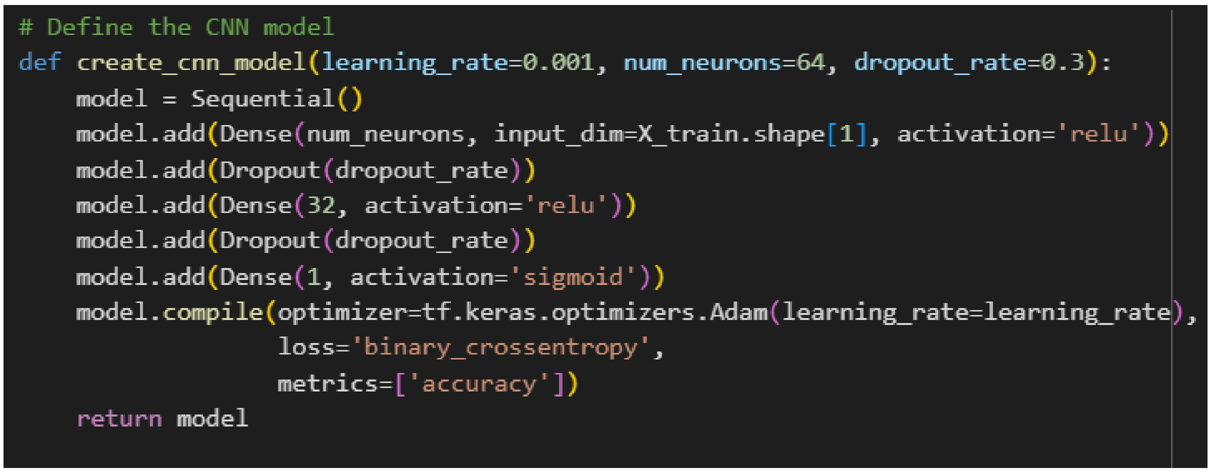
```
data = data.dropna()  # Drop rows with missing values
data['target'] = data['target'].apply(lambda x: 1 if x > 0 else 0)  # Convert target to
```

### 3.3.5 SPLITTING FEATURES AND TARGET



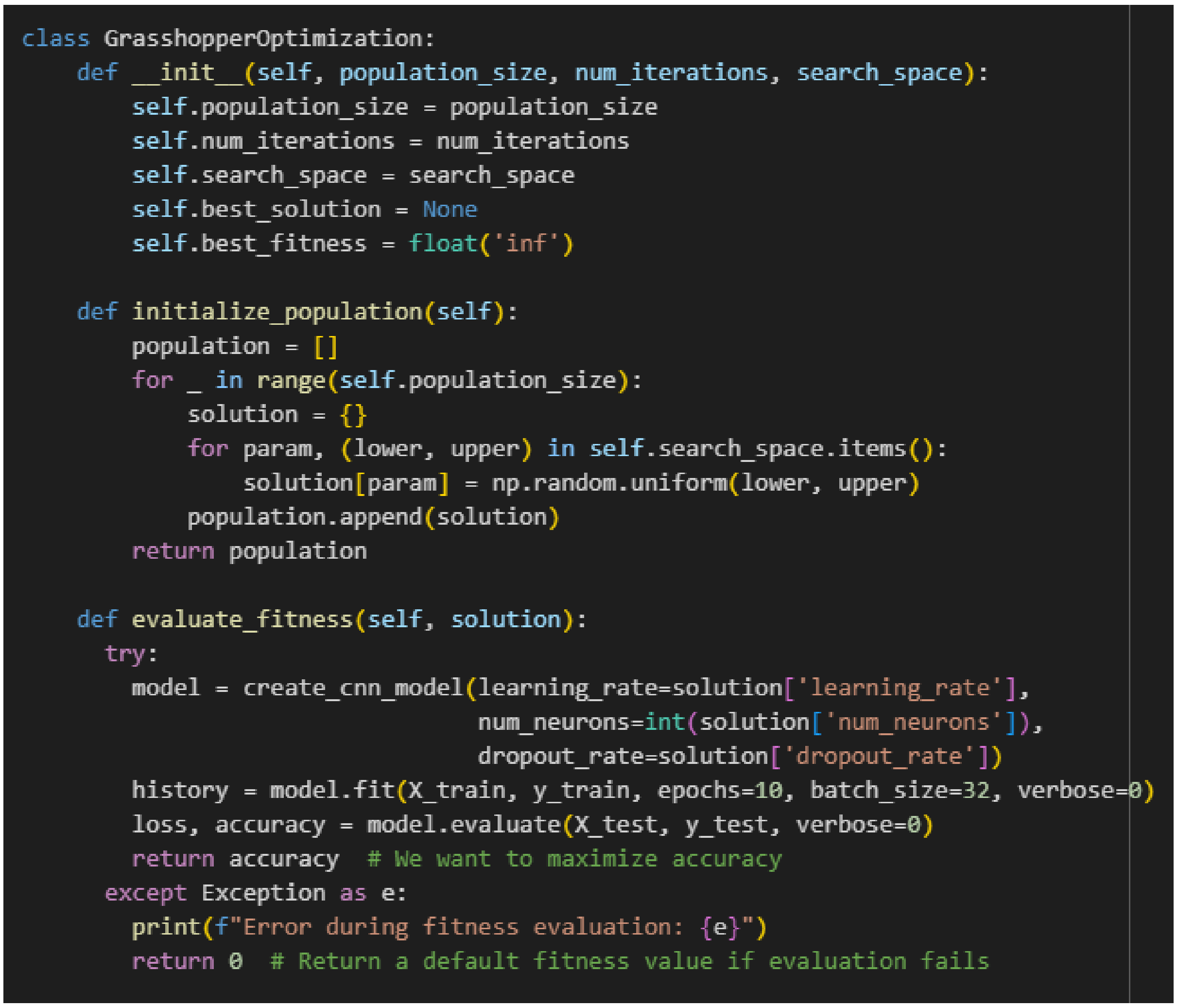
```
# Split features and target
X = data.drop('target', axis=1)
y = data['target']

# Normalize the features
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

### 3.3.6 DEFINE CNN MODEL

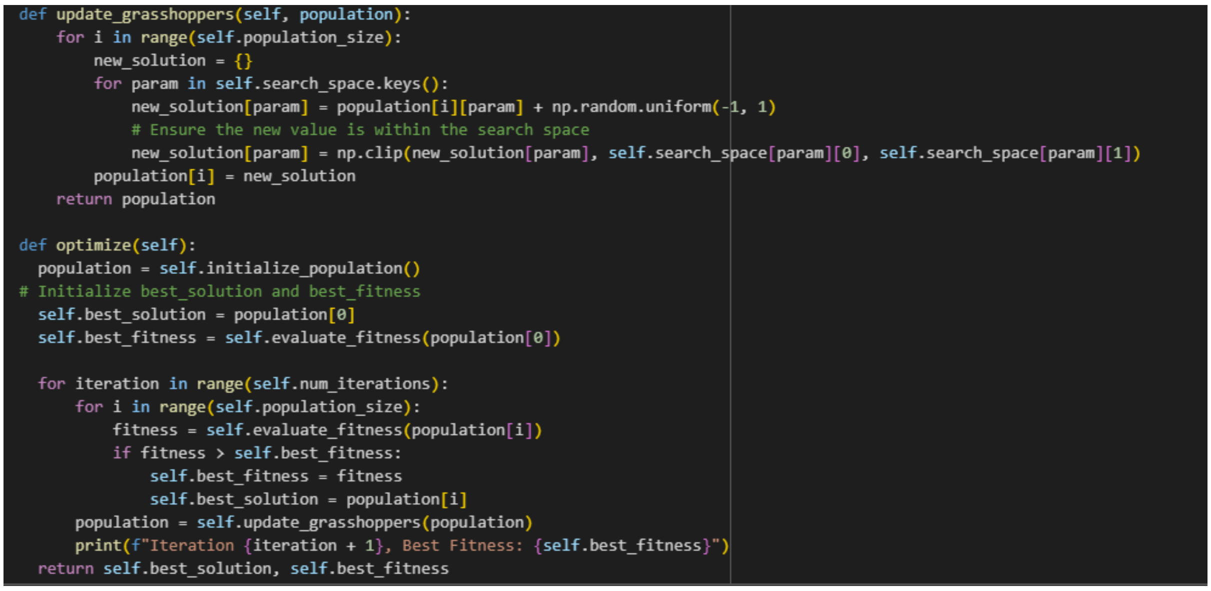


```python
# Define the CNN model
def create_cnn_model(learning_rate=0.001, num_neurons=64, dropout_rate=0.3):
    model = Sequential()
    model.add(Dense(num_neurons, input_dim=X_train.shape[1], activation='relu'))
    model.add(Dropout(dropout_rate))
    model.add(Dense(32, activation='relu'))
    model.add(Dropout(dropout_rate))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=learning_rate),
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
    return model
```

### 3.3.7 GRASSHOPPER OPTIMIZATION

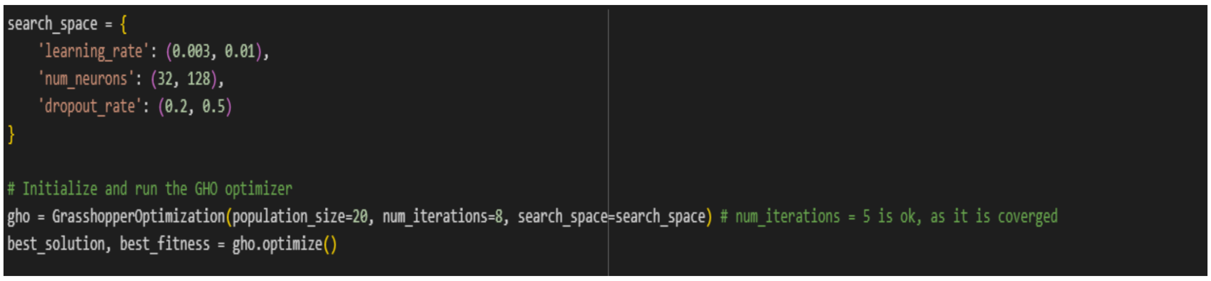


```
class GrasshopperOptimization:
    def __init__(self, population_size, num_iterations, search_space):
        self.population_size = population_size
        self.num_iterations = num_iterations
        self.search_space = search_space
        self.best_solution = None
        self.best_fitness = float('inf')

    def initialize_population(self):
        population = []
        for _ in range(self.population_size):
            solution = {}
            for param, (lower, upper) in self.search_space.items():
                solution[param] = np.random.uniform(lower, upper)
            population.append(solution)
        return population

    def evaluate_fitness(self, solution):
      try:
        model = create_cnn_model(learning_rate=solution['learning_rate'],
                                 num_neurons=int(solution['num_neurons']),
                                 dropout_rate=solution['dropout_rate'])
        history = model.fit(X_train, y_train, epochs=10, batch_size=32, verbose=0)
        loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
        return accuracy  # We want to maximize accuracy
      except Exception as e:
        print(f"Error during fitness evaluation: {e}")
        return 0  # Return a default fitness value if evaluation fails
```

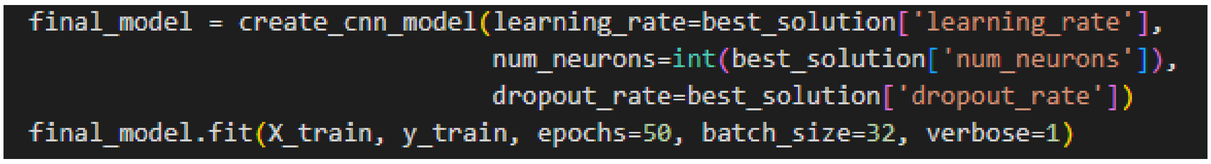


```python
def update_grasshoppers(self, population):
    for i in range(self.population_size):
        new_solution = {}
        for param in self.search_space.keys():
            new_solution[param] = population[i][param] + np.random.uniform(-1, 1)
            # Ensure the new value is within the search space
            new_solution[param] = np.clip(new_solution[param], self.search_space[param][0], self.search_space[param][1])
        population[i] = new_solution
    return population

def optimize(self):
  population = self.initialize_population()
# Initialize best_solution and best_fitness
  self.best_solution = population[0]
  self.best_fitness = self.evaluate_fitness(population[0])

  for iteration in range(self.num_iterations):
      for i in range(self.population_size):
          fitness = self.evaluate_fitness(population[i])
          if fitness > self.best_fitness:
              self.best_fitness = fitness
              self.best_solution = population[i]
      population = self.update_grasshoppers(population)
      print(f"Iteration {iteration + 1}, Best Fitness: {self.best_fitness}")
  return self.best_solution, self.best_fitness
```

## 3.3.7 GHO OPTIMIZER

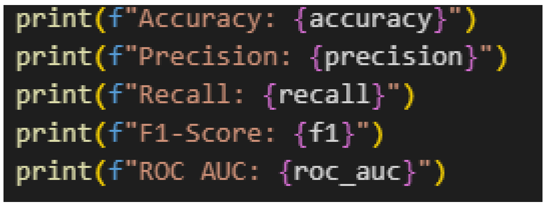


```python
search_space = {
    'learning_rate': (0.003, 0.01),
    'num_neurons': (32, 128),
    'dropout_rate': (0.2, 0.5)
}

# Initialize and run the GHO optimizer
gho = GrasshopperOptimization(population_size=20, num_iterations=8, search_space=search_space) # num_iterations = 5 is ok, as it is coverged
best_solution, best_fitness = gho.optimize()
```

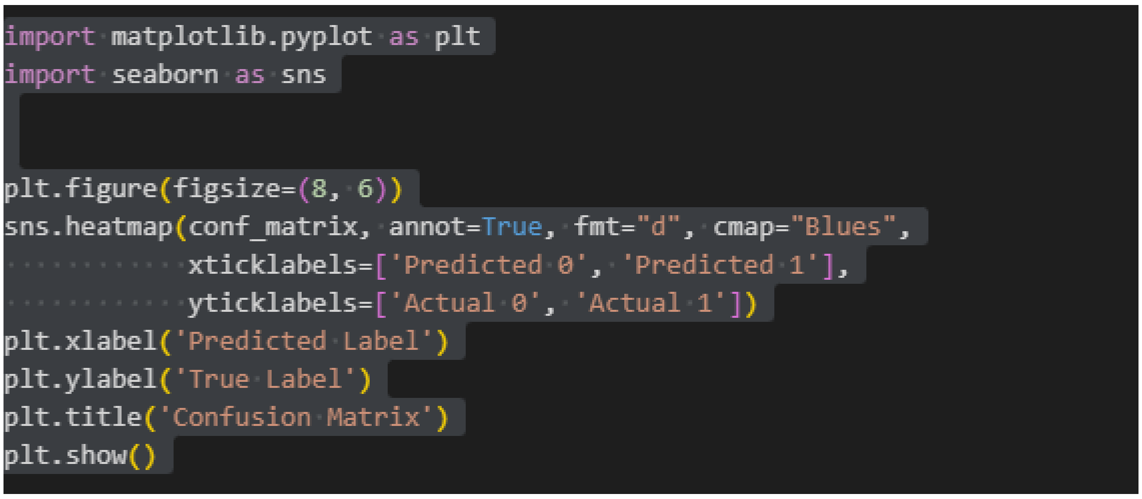


```python
final_model = create_cnn_model(learning_rate=best_solution['learning_rate'],
                               num_neurons=int(best_solution['num_neurons']),
                               dropout_rate=best_solution['dropout_rate'])
final_model.fit(X_train, y_train, epochs=50, batch_size=32, verbose=1)
```

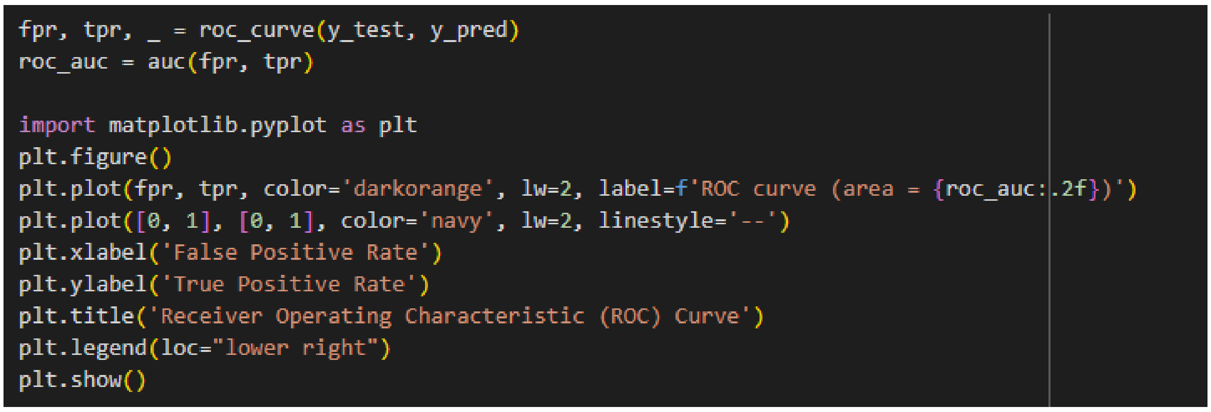
## 3.3.8 EVALUATION METRIX



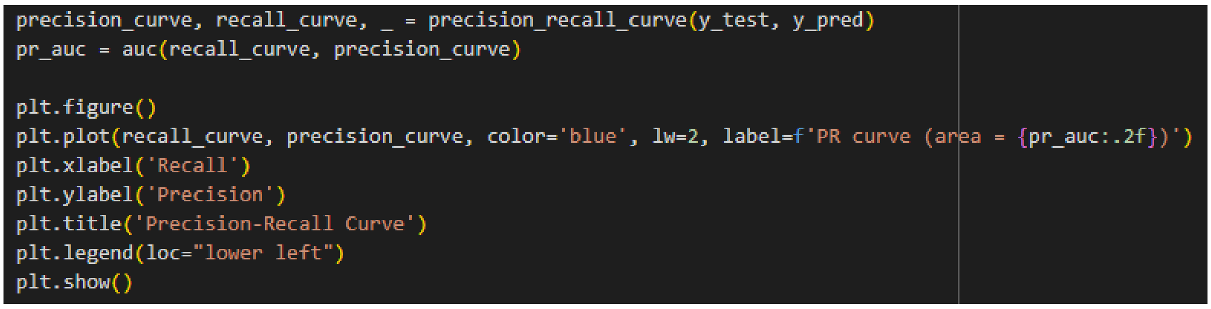
```python
print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1-Score: {f1}")
print(f"ROC AUC: {roc_auc}")
```



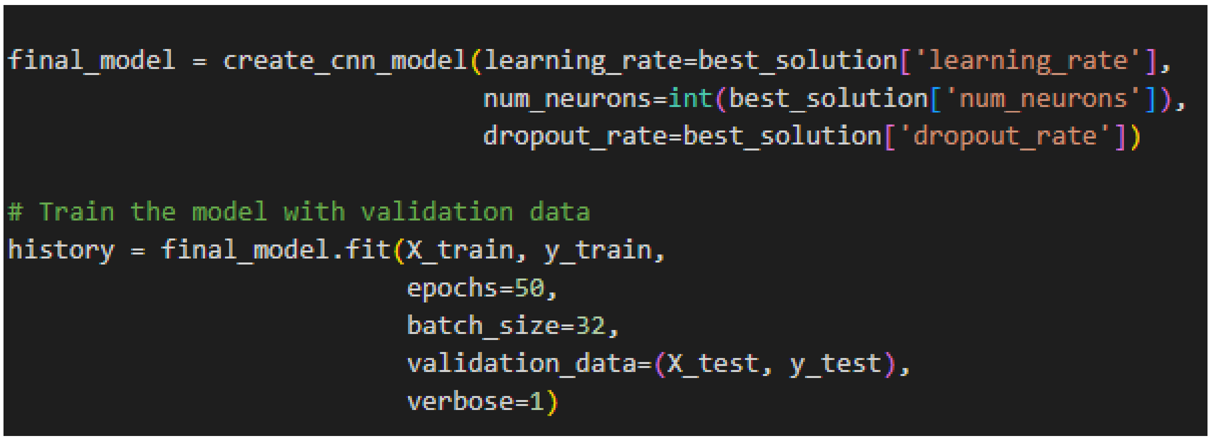
```python
import matplotlib.pyplot as plt
import seaborn as sns


plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",
            xticklabels=['Predicted 0', 'Predicted 1'],
            yticklabels=['Actual 0', 'Actual 1'])
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```



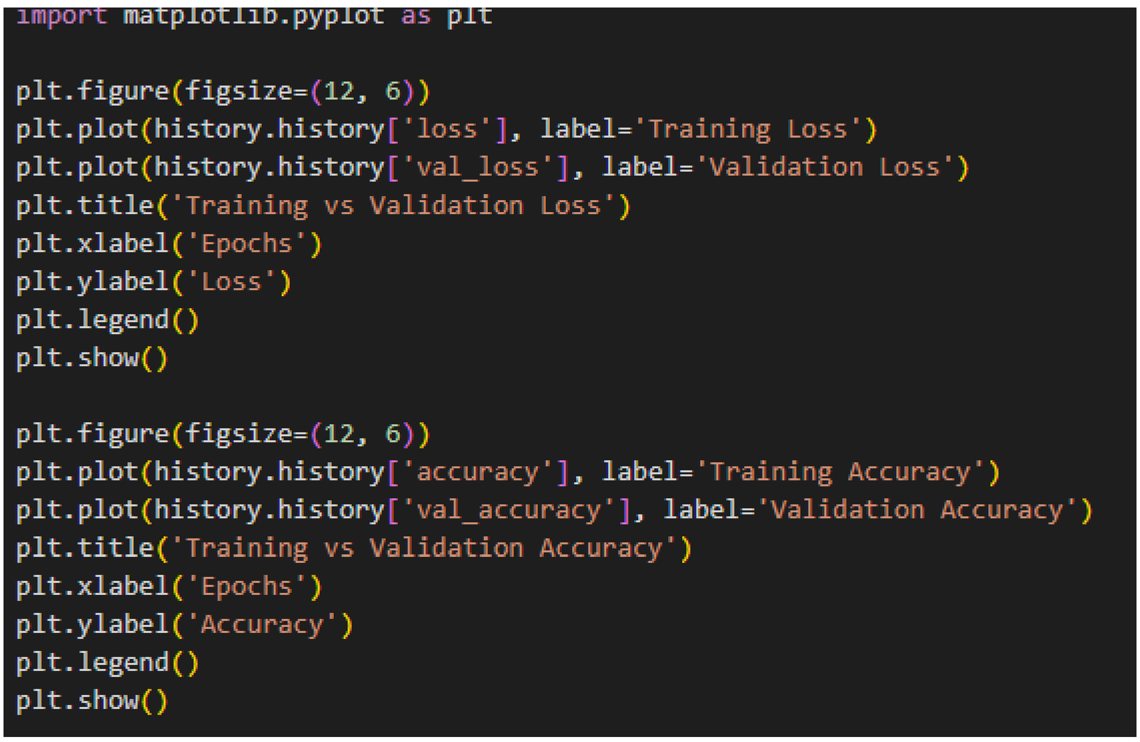
```python
fpr, tpr, _ = roc_curve(y_test, y_pred)
roc_auc = auc(fpr, tpr)

import matplotlib.pyplot as plt
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```



```python
precision_curve, recall_curve, _ = precision_recall_curve(y_test, y_pred)
pr_auc = auc(recall_curve, precision_curve)

plt.figure()
plt.plot(recall_curve, precision_curve, color='blue', lw=2, label=f'PR curve (area = {pr_auc:.2f})')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.legend(loc="lower left")
plt.show()
```



```python
final_model = create_cnn_model(learning_rate=best_solution['learning_rate'],
                               num_neurons=int(best_solution['num_neurons']),
                               dropout_rate=best_solution['dropout_rate'])

# Train the model with validation data
history = final_model.fit(X_train, y_train,
                          epochs=50,
                          batch_size=32,
                          validation_data=(X_test, y_test),
                          verbose=1)
```

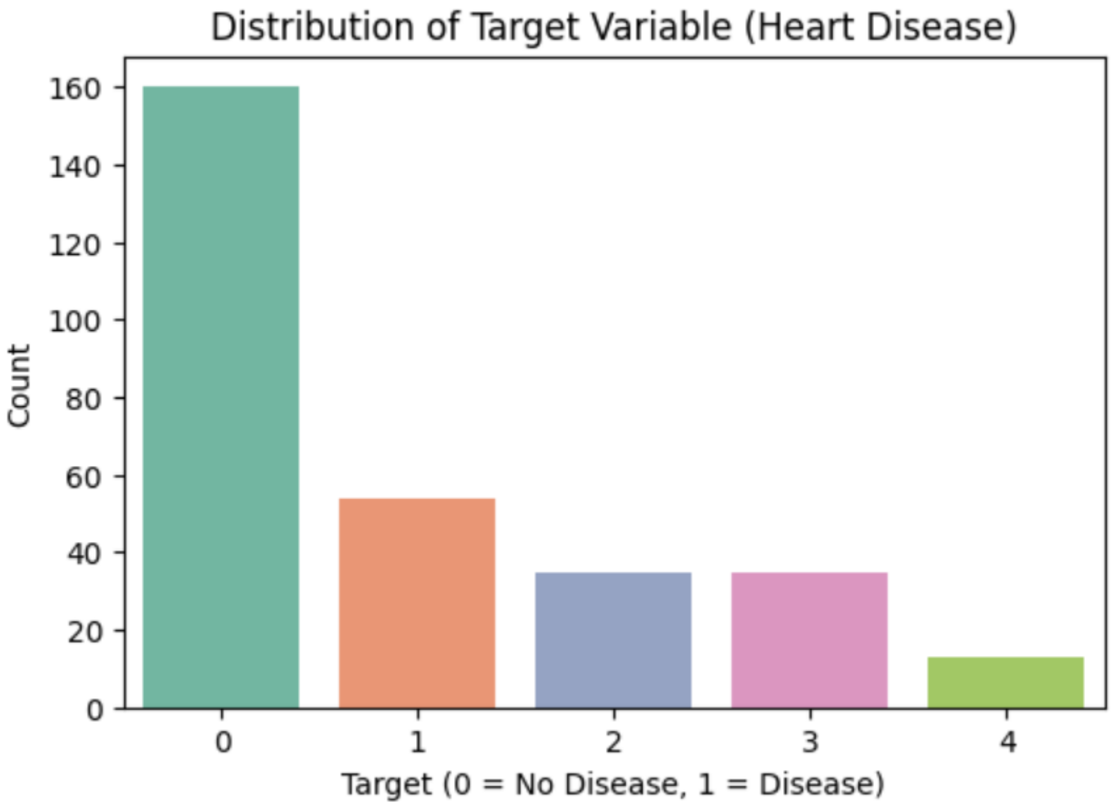


```python
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training vs Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

plt.figure(figsize=(12, 6))
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training vs Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

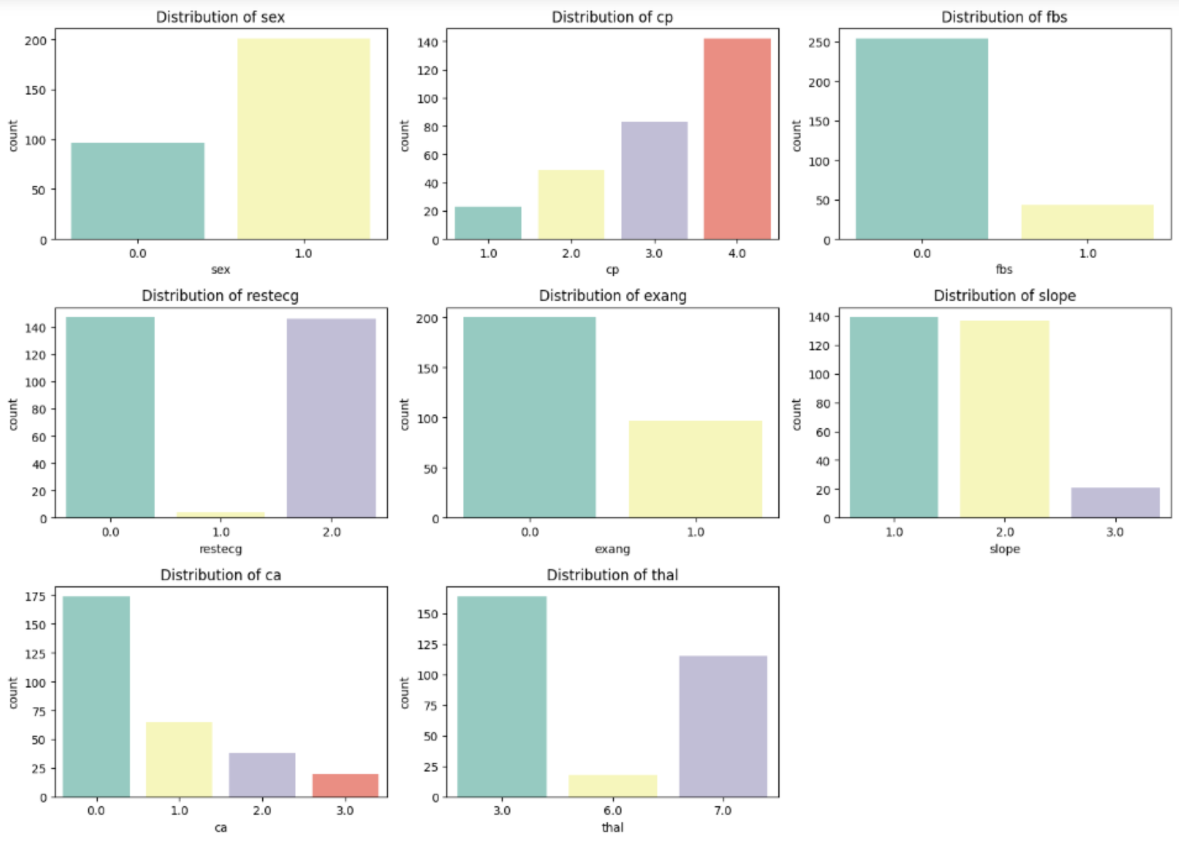
# CHAPTER 4

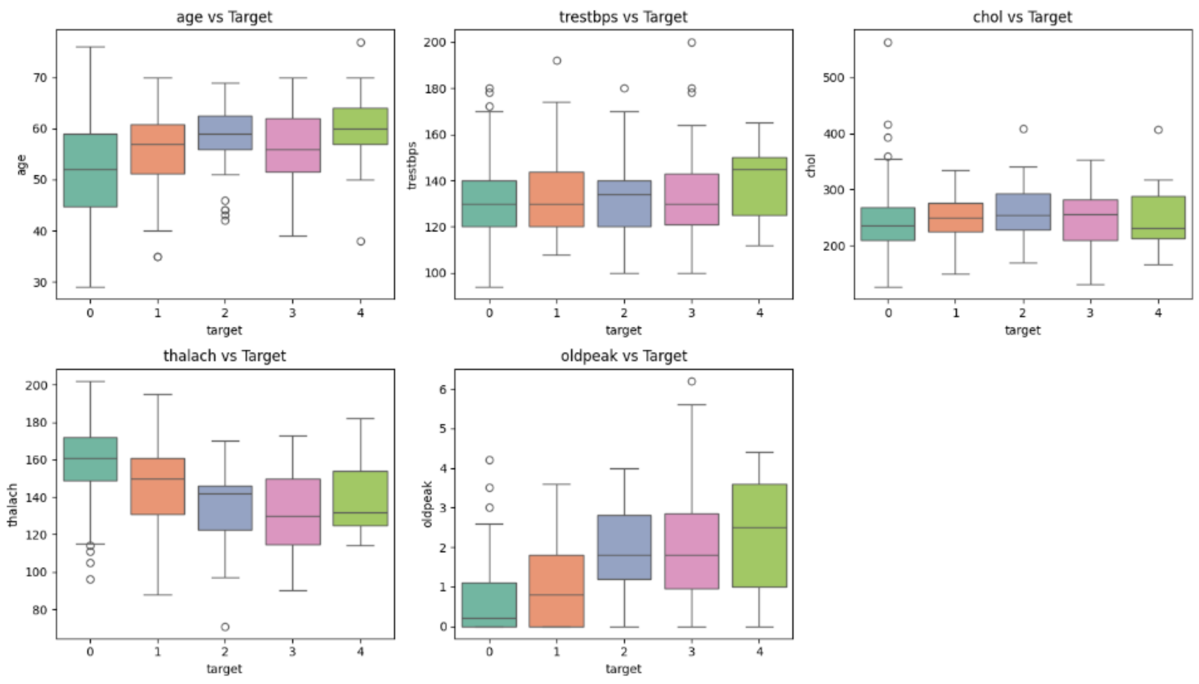
# SNAPSHOTS

## 4.1 DATA PREPROCESSING

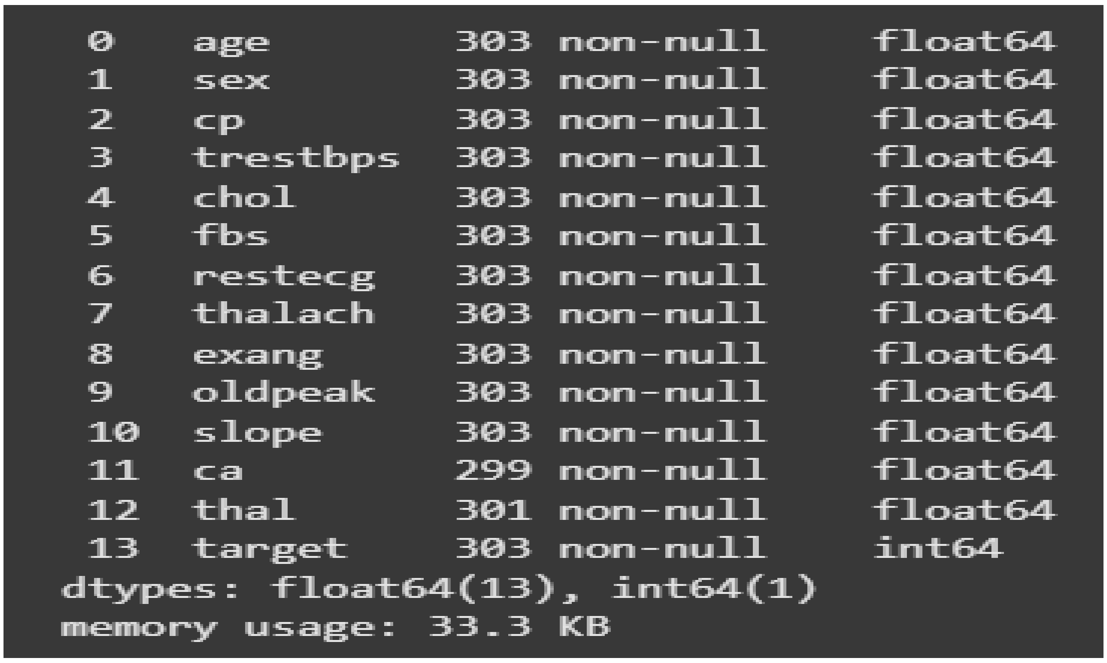


## 4.2 Exploratory Data Analysis (EDA)



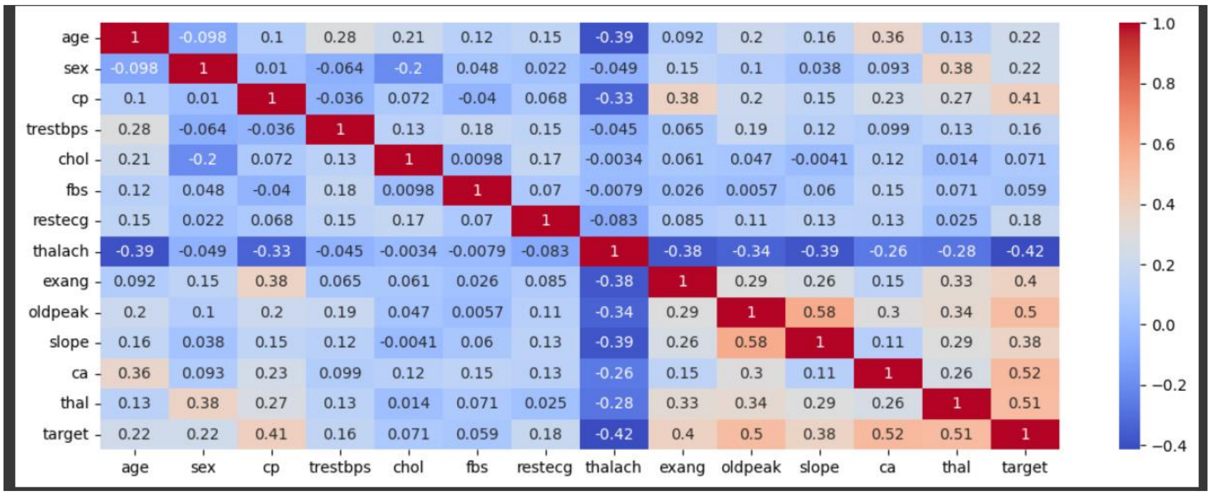


## 4.3 FEATURES



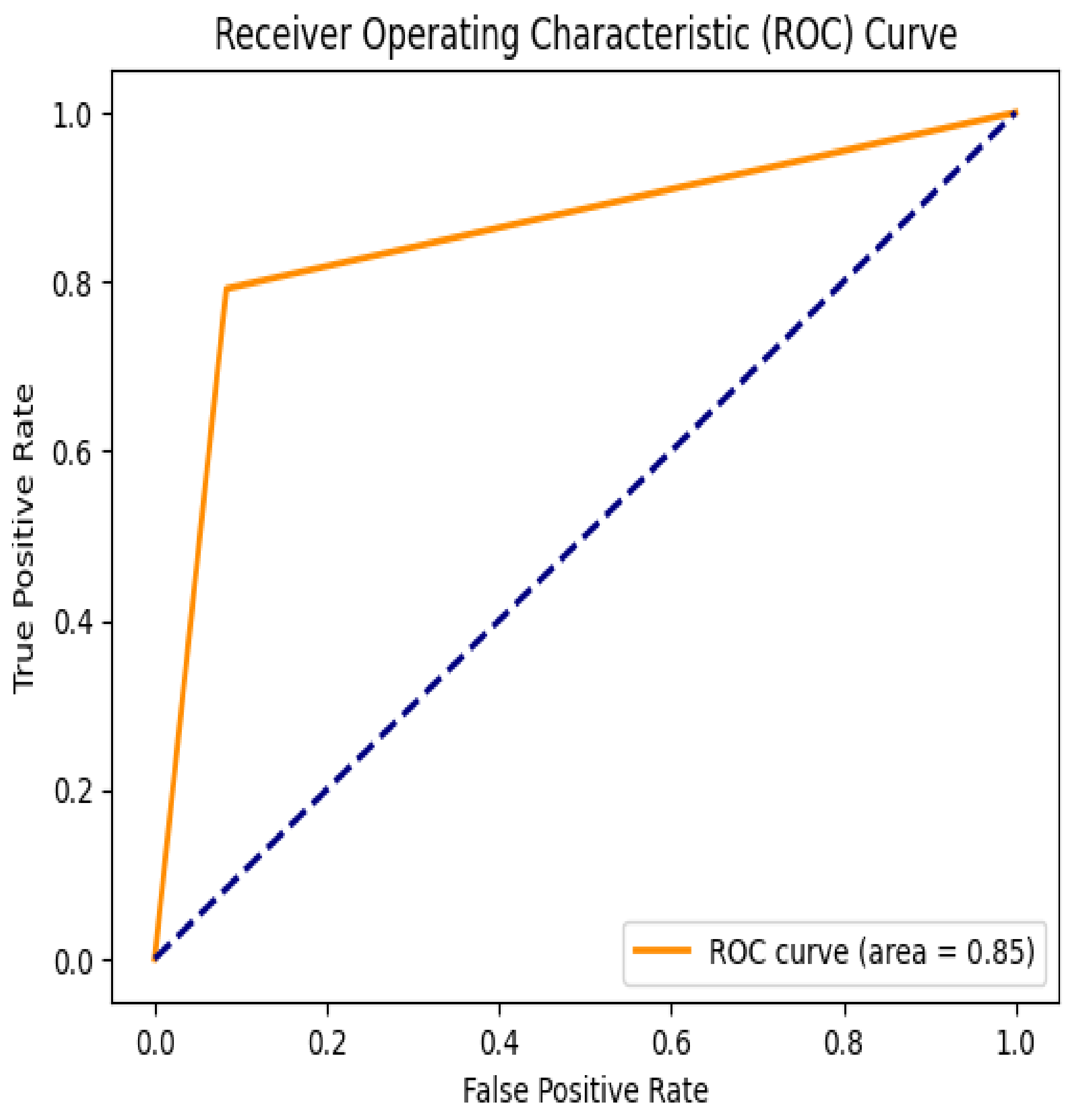
```
  0    age        303 non-null    float64
  1    sex        303 non-null    float64
  2    cp         303 non-null    float64
  3    trestbps   303 non-null    float64
  4    chol       303 non-null    float64
  5    fbs        303 non-null    float64
  6    restecg    303 non-null    float64
  7    thalach    303 non-null    float64
  8    exang      303 non-null    float64
  9    oldpeak    303 non-null    float64
  10   slope      303 non-null    float64
  11   ca         299 non-null    float64
  12   thal       301 non-null    float64
  13   target     303 non-null    int64
 dtypes: float64(13), int64(1)
 memory usage: 33.3 KB
```

## 4.4 CORRELATED MATRIX



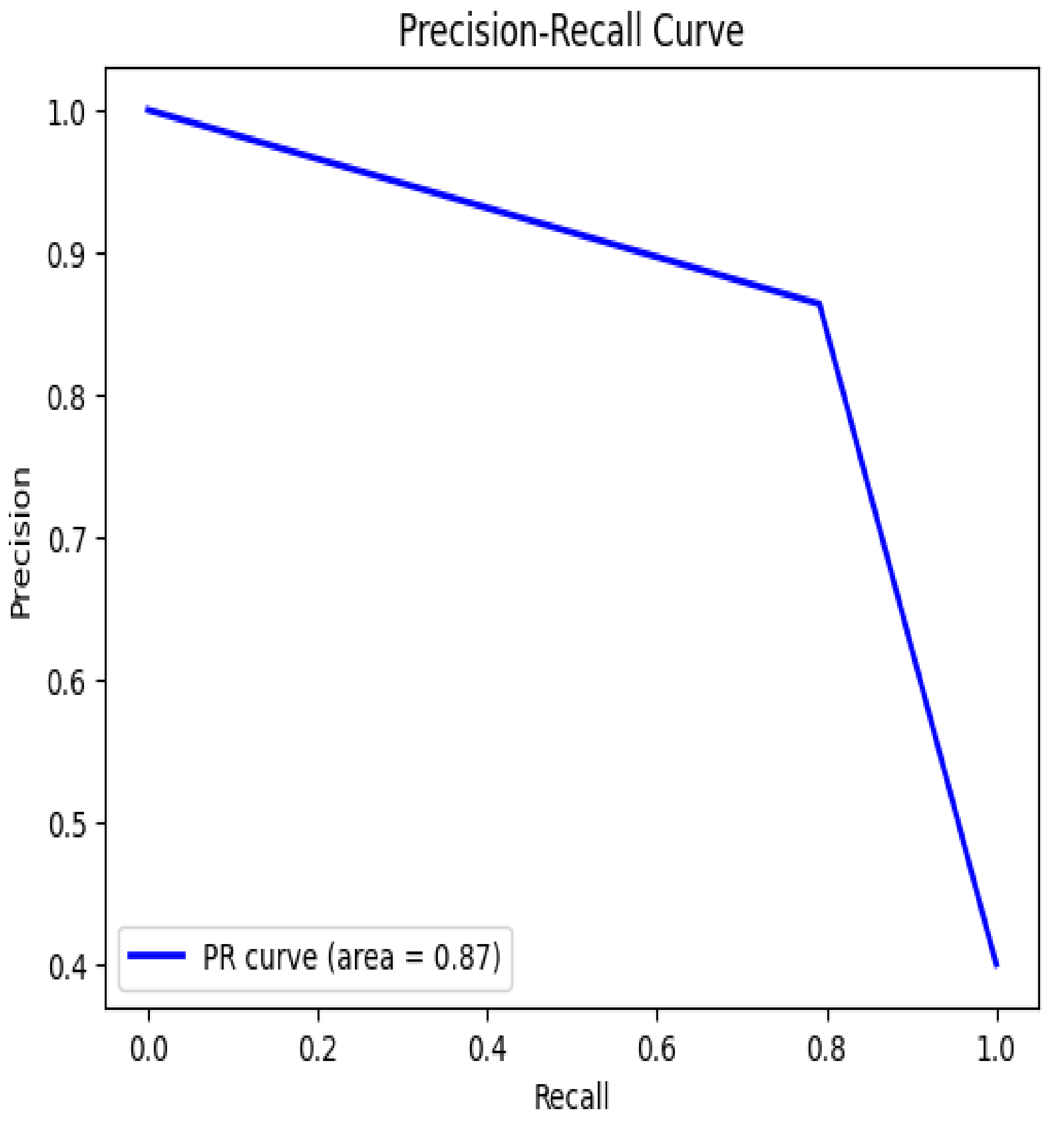
| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| age | 1 | -0.098 | 0.1 | 0.28 | 0.21 | 0.12 | 0.15 | -0.39 | 0.092 | 0.2 | 0.16 | 0.36 | 0.13 | 0.22 |
| sex | -0.098 | 1 | 0.01 | -0.064 | -0.2 | 0.048 | 0.022 | -0.049 | 0.15 | 0.1 | 0.038 | 0.093 | 0.38 | 0.22 |
| cp | 0.1 | 0.01 | 1 | -0.036 | 0.072 | -0.04 | 0.068 | -0.33 | 0.38 | 0.2 | 0.15 | 0.23 | 0.27 | 0.41 |
| trestbps | 0.28 | -0.064 | -0.036 | 1 | 0.13 | 0.18 | 0.15 | -0.045 | 0.065 | 0.19 | 0.12 | 0.099 | 0.13 | 0.16 |
| chol | 0.21 | -0.2 | 0.072 | 0.13 | 1 | 0.0098 | 0.17 | -0.0034 | 0.061 | 0.047 | -0.0041 | 0.12 | 0.014 | 0.071 |
| fbs | 0.12 | 0.048 | -0.04 | 0.18 | 0.0098 | 1 | 0.07 | -0.0079 | 0.026 | 0.0057 | 0.06 | 0.15 | 0.071 | 0.059 |
| restecg | 0.15 | 0.022 | 0.068 | 0.15 | 0.17 | 0.07 | 1 | -0.083 | 0.085 | 0.11 | 0.13 | 0.13 | 0.025 | 0.18 |
| thalach | -0.39 | -0.049 | -0.33 | -0.045 | -0.0034 | -0.0079 | -0.083 | 1 | -0.38 | -0.34 | -0.39 | -0.26 | -0.28 | -0.42 |
| exang | 0.092 | 0.15 | 0.38 | 0.065 | 0.061 | 0.026 | 0.085 | -0.38 | 1 | 0.29 | 0.26 | 0.15 | 0.33 | 0.4 |
| oldpeak | 0.2 | 0.1 | 0.2 | 0.19 | 0.047 | 0.0057 | 0.11 | -0.34 | 0.29 | 1 | 0.58 | 0.3 | 0.34 | 0.5 |
| slope | 0.16 | 0.038 | 0.15 | 0.12 | -0.0041 | 0.06 | 0.13 | -0.39 | 0.26 | 0.58 | 1 | 0.11 | 0.29 | 0.38 |
| ca | 0.36 | 0.093 | 0.23 | 0.099 | 0.12 | 0.15 | 0.13 | -0.26 | 0.15 | 0.3 | 0.11 | 1 | 0.26 | 0.52 |
| thal | 0.13 | 0.38 | 0.27 | 0.13 | 0.014 | 0.071 | 0.025 | -0.28 | 0.33 | 0.34 | 0.29 | 0.26 | 1 | 0.51 |
| target | 0.22 | 0.22 | 0.41 | 0.16 | 0.071 | 0.059 | 0.18 | -0.42 | 0.4 | 0.5 | 0.38 | 0.52 | 0.51 | 1 |

## 4.5 ROC CURVE

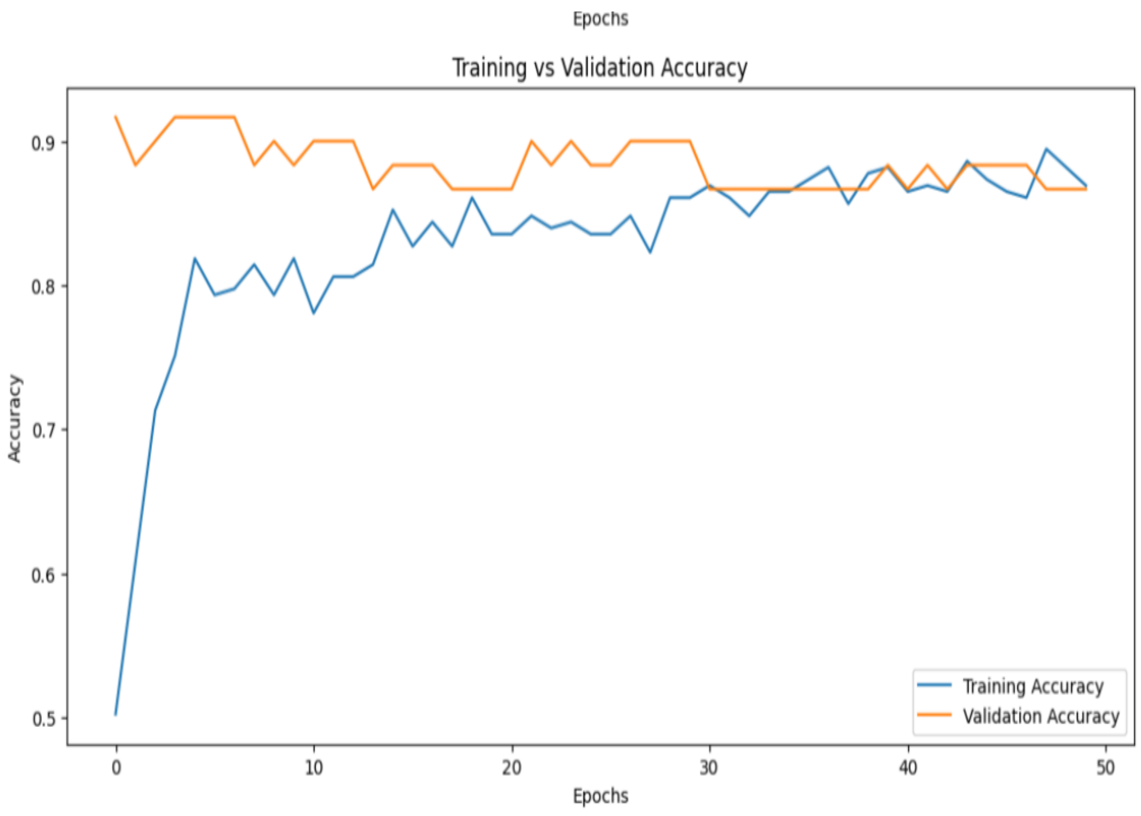


Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.85)

## 4.6 TRAINING VS VALIDATION LOSS

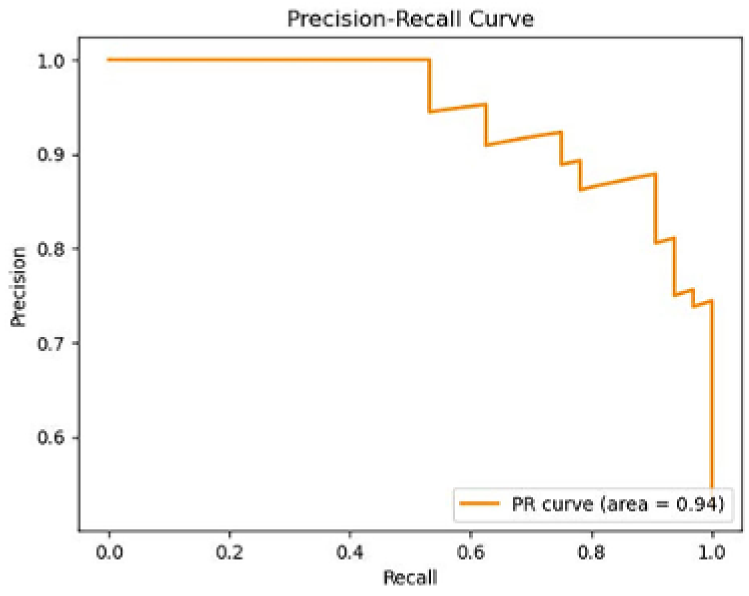


## 4.7 TRAINING VS VALIDATION ACCURACY



## 4.3 COMPARATIVE ANALYSIS

Training



## Testing

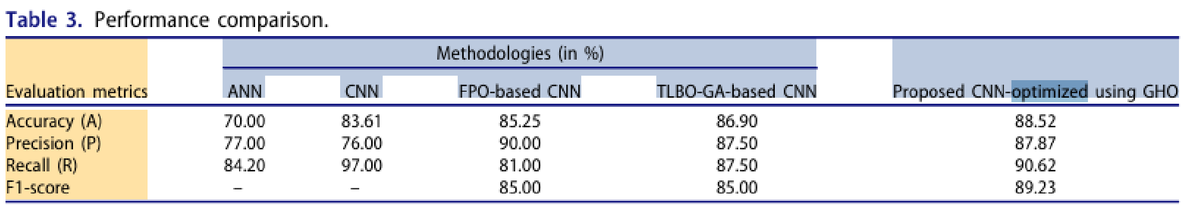


**Table 3.** Performance comparison.

| Evaluation metrics | Methodologies (in %) | | | | |
|---|---|---|---|---|---|
| | ANN | CNN | FPO-based CNN | TLBO-GA-based CNN | Proposed CNN-optimized using GHO |
| Accuracy (A) | 70.00 | 83.61 | 85.25 | 86.90 | 88.52 |
| Precision (P) | 77.00 | 76.00 | 90.00 | 87.50 | 87.87 |
| Recall (R) | 84.20 | 97.00 | 81.00 | 87.50 | 90.62 |
| F1-score | – | – | 85.00 | 85.00 | 89.23 |

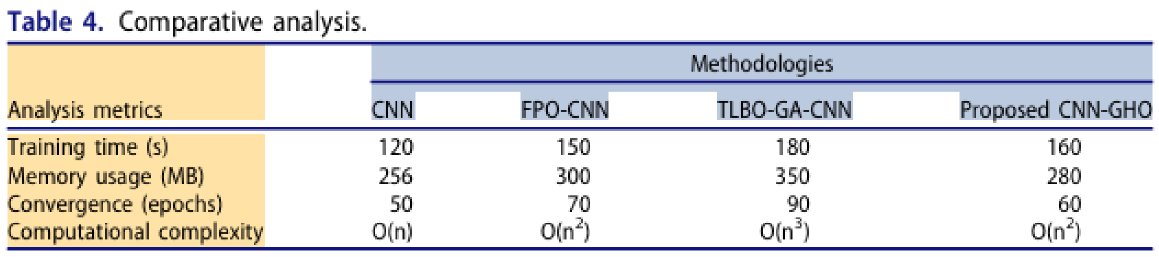
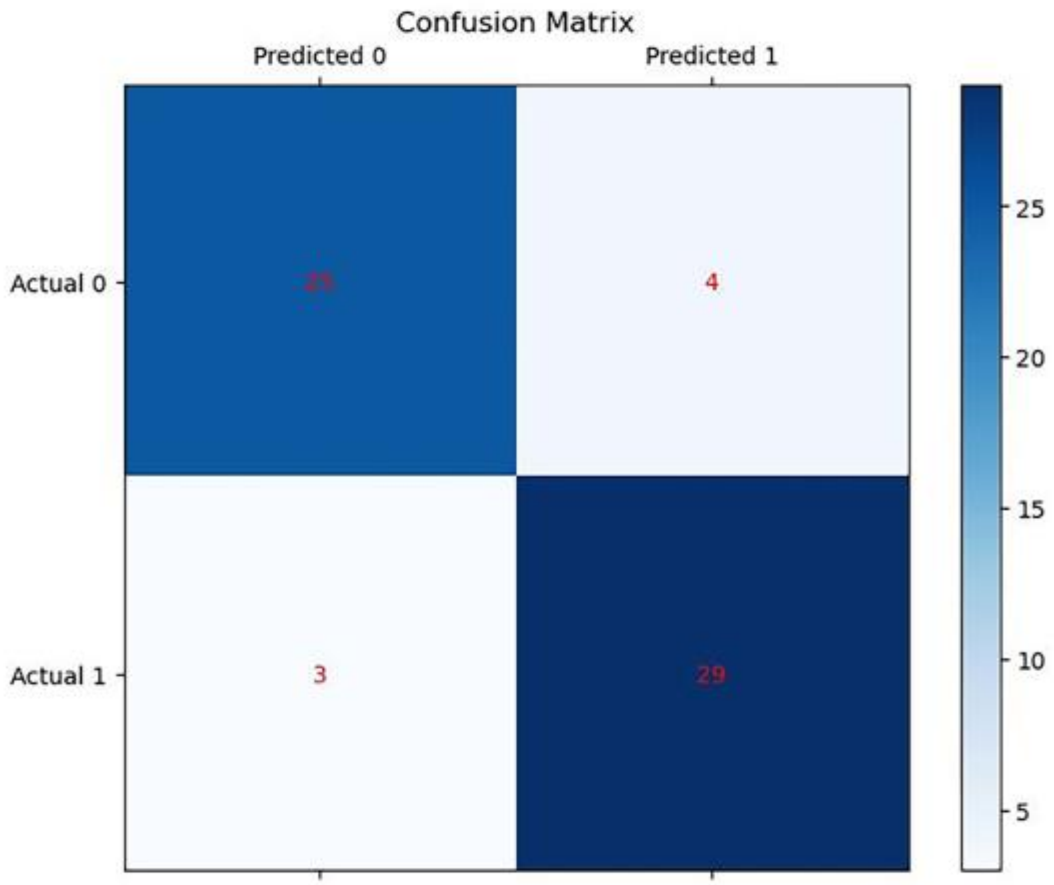
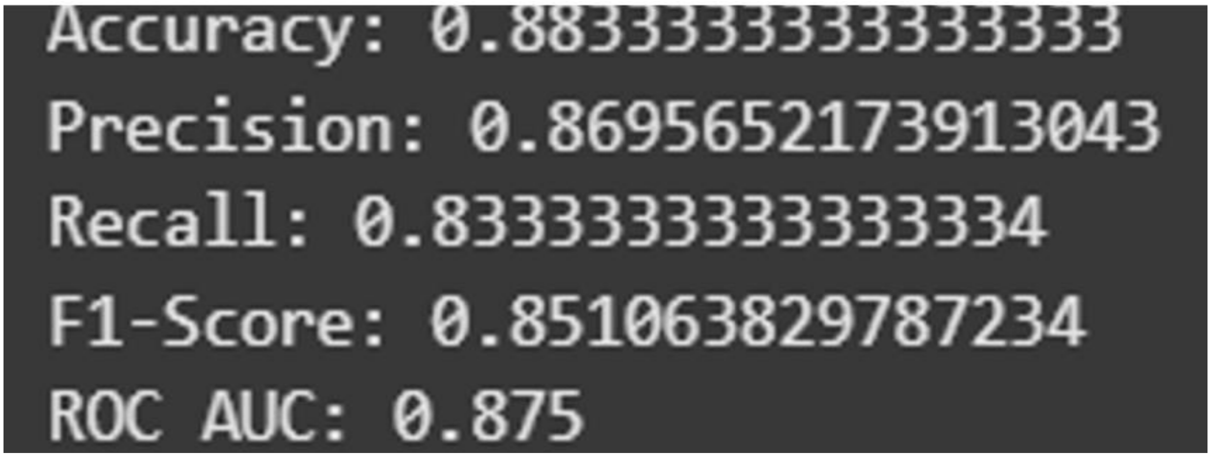


**Table 4.** Comparative analysis.

| Analysis metrics | Methodologies | | | |
|---|---|---|---|---|
| | CNN | FPO-CNN | TLBO-GA-CNN | Proposed CNN-GHO |
| Training time (s) | 120 | 150 | 180 | 160 |
| Memory usage (MB) | 256 | 300 | 350 | 280 |
| Convergence (epochs) | 50 | 70 | 90 | 60 |
| Computational complexity | O(n) | O($n^2$) | O($n^3$) | O($n^2$) |

## 4.4. Performance Metrics





```
Accuracy: 0.8833333333333333
Precision: 0.8695652173913043
Recall: 0.8333333333333334
F1-Score: 0.851063829787234
ROC AUC: 0.875
```

# CHAPTER 5

## 5.1 CONCLUSION

The proposed CNN model optimized using GHO method successfully demonstrated the efficacy for HDPs. The proposed hybrid CNN-GHO model showed the significant improvement in the evaluation metrics, includes Pattern Recognition, Information Retrieval, Image Processing and Machine Learning.

## 5.2 FUTURE PLANS

The proposed study introduces a hybrid model combining a Convolutional Neural Network (CNN) with the Grasshopper Optimization Algorithm (GHO) to enhance heart disease prediction. Key features of this approach include the optimization of critical hyperparameters such as learning rate, batch size, dropout rate, and the number of neurons in dense layers, ensuring a balance between model complexity and performance. The GHO algorithm fine-tunes these parameters by mimicking the swarming behavior of grasshoppers, effectively exploring and exploiting the solution space for optimal results. The model was evaluated on the Cleveland dataset, achieving an accuracy of 88.52%, precision of 87.87%, recall of 90.62%, and an F1-score of 89.23%. Additional performance metrics, including ROC curves (AUC-ROC = 0.94) and precision-recall curves (AUC-PRC = 0.94), further validate its robustness. Comparative analysis with other methods, such as ANN, conventional CNN, and FPO-based CNN, demonstrates the superiority of the proposed hybrid model.

# CHAPTER 6

# REFERENCES

1. Ahmad, A. A., & Polat, H. (2023). Prediction of heart disease based on machine learning using jellyfish optimization algorithm. Diagnostics (Basel, Switzerland), 13(14), 1–17. https://doi.org/10.3390/diagnostics13142392.

2. Al Maruf, A., Golder, A., Al Numan, A., Haque, M. M., & Aung, Z. (2024). Prediction of heart disease and heart failure using ensemble machine learning models. In S. K. Udgata, S. Sethi, & X. Z. Gao (Eds.), Intelligent systems. ICMIB 2023. Lecture notes in networks and systems (Vol. 728), 481–492.

3. Armoundas, A. A., Narayan, S. M., Arnett, D. K., Spector Bagdady, K., Bennett, D. A., Celi, L. A., Friedman, P. A., Gollob, M. H., Hall, J. L., Kwitek, A. E., Lett, E., Menon, B. K., Sheehan, K. A., & Al-Zaiti, S. S. (2024). Use of artifi cial intelligence in improving outcomes in heart disease: A scientific statement from the American heart associ ation. Circulation, 149(14).

4. Asgarov, E. (2024). A comprehensive analysis of machine learning techniques for heart disease prediction. OALib, 11(04), 1–16. Article No. e11490. https://doi.org/10.4236/ oalib.1111490.

5. Bhatt, C. M., Patel, P., Ghetia, T., & Mazzeo, P. L. (2023). Effective heart disease prediction using machine learn ing techniques. Algorithms, 16(2), 88. https://doi.org/10. 3390/a16020088.

6. Ram Kumar, R. P., & Polepaka, S. (2020). Performance com parison of random forest classifier and convolution neural network in predicting heart diseases. Proceedings of the Third International Conference on Computational Intelligence and Informatics. Advances in Intelligent Systems and Computing (Vol. 1090, pp. 683–691).

7. Dhilsath Fathima, M., Manikandan, M., Raviyathu Ammal, M. S. S., Kiruthika, K., Deepa, J., & Singh, P. K. (2024). Enhanced ensemble classifiers for heart disease predic tion. In S. Kumar, K. Balachandran, J. H. Kim, & J. C. Bansal (Eds.), Fourth congress on intelligent systems. CIS 2023. Lecture notes in networks and systems (Vol. 869), 131–141. Springer. https://doi.org/10.1007/978-981-99 9040-5_9.

8. Natarajan, K., Vinoth Kumar, V., Mahesh, T. R., Abbas, M., Kathamuthu, N., Mohan, E., & Annand, J. R. (2024). Efficient heart disease classification through stacked ensemble with optimized firefly feature selection. International Journal of Computational Intelligence Systems, 17(1), 1–14. article no. 174https://doi.org/10.1007/s44196-024-00538-0.

9. Zhou, C., Dai, P., Hou, A., Zhang, Z., Liu, L., Li, A., & Wang, F. (2024). A comprehensive review of deep learning based models for heart disease prediction. Artificial Intelligence Review, 57(10), 263. https://doi.org/10.1007/ s10462-024-10899-0.

10. Jawalkar, A. P., Swetcha, P., Manasvi, N., Sreekala, P., Aishwarya, S., Kanaka Durga Bhavani, P., & Anjani, P. (2023). Early prediction of heart disease with data ana lysis using supervised learning with stochastic gradient boosting. Journal of Engineering and Applied Science, 70(1), 122. https://doi.org/10.1186/s44147-023-00280-y.

# CHAPTER 7

# APPENDIX – BASE PAPER

| | | |
|---|---|---|
| **Title** | : | Optimized convolutional neural network using grasshopper optimization technique for enhanced heart disease prediction |
| **Author** | : | Sanjeeva Polepaka, R. P. Ram Kumar, Deepthi Palakurthy, Vanam Manasa, Akuthota Saritha, Saurav Dixit, Abhishek Chhetri & Myasar Mundher Adnan |
| **Publisher** | : | Nature Portfolio |
| **Year** | : | 2024 |
| **Journal** | : | Cogent Engineering |
| **Indexing** | : | Scopus |
| **Base paper URL** | : | https://doi.org/10.1080/23311916.2024.2423847 |