MINI PROJECT

HEART DISEASE PREDICTION USING SOFT COMPUTING

MCTA 3371

COMPUTATIONAL INTELLIGENCE

SECTION 2

INSTRUCTOR

AHMAD JAZLAN BIN HAJA MOHIDEEN & AMIR AKRAMIN BIN SHAFIE

SEMESTER 1 24/25

TEAM MEMBERS

NAME	MATRIC NO.
MUHAMMAD ZAMIR FIKRI BIN MOHD ZAMRI	2212515
MUHAMMAD HAFIZ HAMZAH BIN FANSURI	2212803
ABDUL HADI BIN ZAWAWI	2210739
SYABAB ALHAQQI BIN JAAFAR	2211117

TABLE OF CONTENT

INTRODUCTION

The growing prevalence of cardiovascular diseases (CVD) has made heart risk prediction an essential tool in healthcare. With advancements in technology, there is an increasing focus on developing accurate and efficient methods to assess an individual's risk of heart disease. This report explores the current methodologies for heart risk prediction, including traditional risk scoring systems and emerging technologies like machine learning and wearable devices. The objective is to evaluate these approaches, identify challenges and propose recommendations for improving predictive accuracy and accessibility.

We use ANN and GA to implement our prediction. Artificial Neural Networks (ANN) are based on the complex build of the human brain, and have shown extraordinary ability to understand and recognise hidden connections and patterns among large and diverse datasets. The neural network model will be evaluated using relevant health data from heart risk prediction dataset. Genetic algorithms (GA) are a class of optimization algorithms inspired by the principles of natural selection and genetics. They are particularly beneficial in solving complex optimization problems where traditional methods struggle.

The findings of this project will help people with an advanced healthcare system by providing more accurate analysis and prediction of an individual's risk of heart disease. These findings can encourage patients to live healthier lifestyles and follow prescribed medication. Prediction models can drive patients to take proactive efforts to reduce possible threats and enhance their overall cardiovascular health by presenting them with information about their particular risk of heart disease.

This prediction tool is important in the healthcare system because it helps prioritize resources and actions for those that are most needed, hence optimizing spending and increasing efficiency. Therefore, the importance of heart risk prediction cannot be taken lightly as it represents a proactive approach to addressing the global burden of heart disease, which continues to pose a significant public health challenge worldwide.

OBJECTIVES

1. **Develop a Predictive Model**:
    - Classify people into high-risk or low-risk groups for heart disease using an Artificial Neural Network (ANN) based on certain health criteria.
2. **Promote Early Detection**:
    - Give patients and medical professionals a tool that helps them recognize possible heart disease risks early so that prompt actions may be made.
3. **Facilitate User Interaction**:
    - Create a system that allows users—patients or healthcare professionals—to enter their medical data and get real-time, useful estimations.

METHODOLOGY

1. Data processing

1.1. Loading dataset:

1.1.1. The (readtable) function loads data from the heart_attack_prediction_dataset_Modified.xlsx file.

1.1.2. The column names are preserved using detectImportOptions with VariableNamingRule set to 'preserve'.

1.2. Feature Extraction:

1.2.1. Key health parameters such as Age, Cholesterol, HeartRate, Smoking, FamilyHistory, Diabetes, and Obesity are extracted.

1.2.2. BloodPressure values are split into systolic and diastolic components to compute the Mean Arterial Pressure (MAP).

1.3. Data Cleaning:

1.3.1. Rows with missing values (NaN) are removed to ensure data integrity.

1.4. Feature Normalization:

1.4.1. Features are standardized by subtracting the mean and dividing by the standard deviation for consistent scaling.

2. Splitting the dataset

2.1. The dataset is divided into training (70%) and testing (30%) subsets using MATLAB's cvpartition function.

2.2. This ensures random yet reproducible splits for model evaluation.

3. Training the ANN

3.1. Defining the ANN Architecture:

3.1.1. A feedforward neural network is generated using MATLAB's (fitcnet) function, optimized for binary classification.

3.1.2. Hyperparameters (e.g., number of hidden neurons) are determined via a Genetic Algorithm (GA).

3.2. Hyperparameter Optimization:

3.2.1. The GA minimizes the objective function, which evaluates the model accuracy on the test data.

3.3. Model Training:

3.3.1. The optimal architecture is trained on the training set with backpropagation to adjust weights iteratively.

4. Testing and Evaluation

4.1. Model Testing:

4.1.1. Predictions are generated for the test set, rounded to binary outputs (0 or 1).

4.2. Performance Metrics:

4.2.1. Model accuracy is computed as the ratio of correctly classified cases.

4.2.2. A confusion matrix visualizes true positives, true negatives, false positives, and false negatives.

4.3. Residual Analysis:

4.3.1. Regression residuals are plotted to analyze prediction errors.

4.4. Visualization:

4.4.1. The neural network architecture and confusion matrix are visualized for interpretability.

5. User Interaction Through GUI

5.1. Designing the GUI:

5.1.1. A MATLAB App Designer GUI allows user interaction with fields for health parameters, dropdown menus for categorical inputs, and a "Predict Risk" button.

5.2. User Input:

5.2.1. Inputs are normalized using the same standardization parameters as the training data.

5.2.2. Blood pressure is split and MAP is calculated dynamically.

5.3. Processing and Prediction:

5.3.1. TThe trained ANN predicts the heart disease risk as "Low" or "High," displayed in the GUI.

5.3.2. For high risk, the GUI includes a suggestion for medical consultation.

5.4. Error Handling:

5.4.1. Validations are included for blood pressure format and input completeness, with error messages displayed via (uialert).

CODING DETAILS: ARTIFICIAL NEURAL NETWORK (ANN) AND GENETIC ALGORITHM (GA)

1. Initial Setup

```
clear all
close all
clc
```

- Clear all variables from memory (clear all),
- Close all open figures (close all),
- Clear the command window (clc).

2. Load and Preprocess Data

```
filename = 'heart_attack_prediction_dataset_Modified.xlsx'; % specify your file name
opts = detectImportOptions(filename);
opts.VariableNamingRule = 'preserve';
data = readtable(filename, opts);
```

- filename specifies the name of the Excel file you want to load (heart_attack_prediction_dataset_Modified.xlsx).
- detectImportOptions(filename) automatically detects how the data should be read, including column headers and data types.
- opts.VariableNamingRule = 'preserve'; ensures that the variable names from the file are preserved exactly as they are.
- readtable(filename, opts) reads the data from the file and stores it in a table called data, where each row is a sample and each column is a feature.

3. Extracting Columns from the Data

```
Age = data{:, 1};
Cholesterol = data{:, 3};
BloodPressure = data{:, 4};
HeartRate = data{:, 5};
Diabetes = data{:, 6};
FamilyHistory = data{:, 7};
Smoking = data{:, 8};
Obesity = data{:, 9};
```

- The data{:, 1} extracts all the rows of the first column, and similarly for other columns.
- Each column is assigned to a variable, such as Age, Cholesterol, BloodPressure, and so on, representing various features of the data, like age, cholesterol level, blood pressure, and so on.

4. **Blood Pressure Preprocessing**

```
bp_values = BloodPressure;
map_values = zeros(length(BloodPressure), 1);
for i = 1:length(bp_values)
   bp_split = strsplit(bp_values{i}, '/');
   systolic = str2double(bp_split{1});
   diastolic = str2double(bp_split{2});
   map_values(i) = (2 * diastolic + systolic) / 3;
end
```

The BloodPressure variable contains blood pressure values in a string format (e.g.,'120/80').

The for loop goes through each blood pressure value:

- **strsplit(bp_values{i}, '/')** splits the string '120/80' into two parts: '120' (systolic) and '80' (diastolic).
- **str2double()** converts those strings into numbers.
- The **mean arterial pressure (MAP)** is calculated using the formula:

MAP=(2*Diastolic+Systolic)/3

- These MAP values are stored in the map_values array.

5. **Feature Matrix Preparation**

input = [Age Cholesterol map_values...

HeartRate Diabetes FamilyHistory Smoking Obesity];

output = data{:, 25};

- The input matrix is created by combining multiple features into one matrix. These features (age, cholesterol, MAP, heart rate, diabetes status, etc.) will be used as the input to the neural network.
- output is the target variable (the heart attack risk) extracted from column 25 in the dataset. This variable will be what the model tries to predict.

6. **Data Partitioning**

cv = cvpartition(size(input, 1), 'HoldOut', 0.3);

idx = cv.test;

XTrain = input(~idx, :);

YTrain = output(~idx);

XTest = input(idx, :);

YTest = output(idx);

- cvpartition(size(input, 1), 'HoldOut', 0.3): Splits the data into training (70%) and testing (30%) sets.
- cv.test gives the indices of the testing set.
- XTrain and XTest hold the input features for the training and testing sets, respectively.

- YTrain and YTest hold the corresponding output labels (heart attack risk) for training and testing.

7. **Standardization the Input Features**

```
meanX = mean(XTrain);

stdX = std(XTrain);

XTrain = (XTrain - meanX) ./ stdX;

XTest = (XTest - meanX) ./ stdX;
```

- This standardizes the features so that they have a mean of 0 and a standard deviation of 1. Standardization helps neural networks learn better.
- meanX is the mean of the training data, and stdX is the standard deviation.
- The training and test data are both standardized using these values.

8. **Objective Function for Genetic Algorithm**

```
  function accuracy = objectiveFunction(params, XTrain, YTrain, XTest, YTest)

  hiddenLayerSizes = round(params(1)); % Number of neurons in hidden layer

  Mdl = fitcnet(XTrain, YTrain, 'LayerSizes', hiddenLayerSizes);

  YPred = predict(Mdl, XTest);

  accuracy = -sum(YPred == YTest) / length(YTest) * 100; % Negative accuracy for minimization

end
```

- This function is used for optimizing the neural network’s hyperparameters (in this case, the number of neurons in the hidden layer) using Genetic Algorithm (GA).
- params is an input parameter that will be adjusted by GA.

- The function trains a neural network with the given number of neurons and computes the accuracy of the predictions.
- The negative accuracy is returned so that the GA can minimize this value.

9. **Running Genetic Algorithm to Optimize Hyperparameters**

options = optimoptions('ga', 'Display', 'iter', 'PopulationSize', 20, 'MaxGenerations', 10);

bounds = [5; 50]; % Bounds for hidden layer size

[optimalParams, optimalAccuracy] = ga(@(params) objectiveFunction(params, XTrain, YTrain, XTest, YTest), 1, [], [], [], [], bounds(1), bounds(2), [], options);

optimoptions('ga') sets up the **Genetic Algorithm** with options:

- 'PopulationSize' sets how many configurations will be tested in each generation.
- 'MaxGenerations' specifies how many generations the GA should run.

bounds specifies the range of neurons to test (between 5 and 50 neurons).

The GA tries different configurations of hidden layer sizes and optimizes the number of neurons by minimizing the error.

10. **Train the Final Neural Network with Optimal Hyperparameters**

optimalHiddenLayerSizes = round(optimalParams(1));

Mdl = fitcnet(XTrain, YTrain, 'LayerSizes', optimalHiddenLayerSizes);

- The best number of neurons (found by the GA) is stored in optimalHiddenLayerSizes.
- The neural network is then trained again using this optimal configuration (fitcnet).

11. **Prediction and Evaluation**

```
YPred = predict(Mdl, XTest);

accuracy = sum(YPred == YTest) / length(YTest) * 100;
```

- predict(Mdl, XTest) uses the trained model (Mdl) to make predictions on the test data (XTest).
- The accuracy is calculated as the percentage of correct predictions.

12. Visualization of Results

```
confusionchart(YTest, YPred);
```

- A confusion matrix is displayed, which shows the performance of the model by comparing the true values (YTest) with the predicted values (YPred).

13. Saving the Model and Parameters

```
save('trained_network.mat', 'net');
save('standardization_params.mat', 'meanX', 'stdX');
```

- The trained neural network (net) and the standardization parameters (meanX, stdX) are saved into .mat files for later use.

GRAPHICAL USER INTERFACE (GUI)

1. **Create the Main Figure Window**

```
fig = uifigure('Name', 'Heart Attack Risk Prediction', 'Position', [100, 100, 500, 600]);
```

- This creates a UI figure window where the user will interact with the inputs and see the output.
- The 'Name' parameter specifies the title of the window as 'Heart Attack Risk Prediction'.
- 'Position' specifies the position and size of the window (starting from [100, 100] with a size of 500x600 pixels).

2. **Create Input Fields for Age, Cholesterol, Blood Pressure, etc.**

Age Input:

```
uilabel(fig, 'Text', 'Age:', 'Position', [50, 520, 150, 22]);
ageField = uieditfield(fig, 'numeric', 'Position', [200, 520, 100, 22]);
```

- uilabel creates a label to display text ('Age:').
- uieditfield creates an editable field where users can input their age. The 'numeric' type ensures that only numbers can be entered.

Cholesterol Input:

```
uilabel(fig, 'Text', 'Cholesterol (mg/dL):', 'Position', [50, 480, 150, 22]);
cholesterolField = uieditfield(fig, 'numeric', 'Position', [200, 480, 100, 22]);
```

- Similarly, this label and field let the user input their cholesterol value in mg/dL.

Blood Pressure Input:

```
uilabel(fig, 'Text', 'Blood Pressure (Systolic/Diastolic):', 'Position', [50, 440, 250, 22]);
bpField = uieditfield(fig, 'text', 'Position', [300, 440, 100, 22]);
```

- This field allows the user to input their blood pressure in the format Systolic/Diastolic (e.g., 120/80).

Heart Rate Input:

```
uilabel(fig, 'Text', 'Heart Rate (bpm):', 'Position', [50, 400, 150, 22]);
heartRateField = uieditfield(fig, 'numeric', 'Position', [200, 400, 100, 22]);
```

- An editable field for the user to enter their heart rate in beats per minute.

Diabetes, Family History, Smoking, and Obesity Inputs:

```
diabetesMenu = uidropdown(fig, 'Items', {'Yes', 'No'}, 'Position', [200, 360, 150, 22]);
familyHistoryMenu = uidropdown(fig, 'Items', {'Yes', 'No'}, 'Position', [300, 320, 150, 22]);
smokingMenu = uidropdown(fig, 'Items', {'Yes', 'No'}, 'Position', [200, 280, 150, 22]);
obesityMenu = uidropdown(fig, 'Items', {'Yes', 'No'}, 'Position', [200, 240, 150, 22]);
```

- uidropdown creates dropdown menus for binary options, where the user selects either 'Yes' or 'No'.

3. **Create the 'Predict Risk' Button**

```
predictButton = uibutton(fig, 'Text', 'Predict Risk', 'Position', [200, 150, 100, 40], ...
   'ButtonPushedFcn', @(btn, event) predictRisk());
```

- A button labeled 'Predict Risk' is created.
- When clicked, it triggers the predictRisk function (explained later).

4. **Callback Function: predictRisk**

This function will run when the user clicks the 'Predict Risk' button.

Try-Catch Block for Error Handling:

```
try
   % Get inputs from the fields
   age = ageField.Value;
   cholesterol = cholesterolField.Value;
   bp = bpField.Value;
   heartRate = heartRateField.Value;
   diabetes = strcmp(diabetesMenu.Value, 'Yes');
   familyHistory = strcmp(familyHistoryMenu.Value, 'Yes');
   smoking = strcmp(smokingMenu.Value, 'Yes');
   obesity = strcmp(obesityMenu.Value, 'Yes');
```

- Inputs are taken from the fields and stored in variables.
- For dropdowns (diabetesMenu, familyHistoryMenu, etc.), strcmp checks if the selected option is 'Yes' (converted to a logical true/false value).

Blood Pressure Parsing:

```
bp_split = strsplit(bp, '/');
if numel(bp_split) ~= 2
   error('Invalid blood pressure format. Use "Systolic/Diastolic".');
end
systolic = str2double(bp_split{1});
diastolic = str2double(bp_split{2});
if isnan(systolic) || isnan(diastolic)
   error('Invalid blood pressure values.');
end
map = (2 * diastolic + systolic) / 3;
```

- The blood pressure input (Systolic/Diastolic) is split using strsplit to extract systolic and diastolic values.
- str2double converts the split string values into numeric values.
- The mean arterial pressure (MAP) is calculated using the formula: $MAP=(2\times Diastolic+Systolic)/3$

Feature Vector Preparation:

```
features = [age, cholesterol, map, heartRate, diabetes, familyHistory, smoking, obesity];
```

- The features (age, cholesterol, blood pressure, heart rate, etc.) are grouped into a feature vector for prediction.

Load the Trained Model and Standardization Parameters:

```
load('trained_network.mat', 'net');
load('standardization_params.mat', 'meanX', 'stdX');
```

- The trained neural network (net) and standardization parameters (meanX, stdX) are loaded from .mat files that were saved earlier. These are used to make predictions and to standardize the input features.

Standardize the Input Features:

```
features = (features - meanX) ./ stdX;
```

- The input features are standardized using the previously loaded mean and standard deviation values.

Neural Network Prediction:

```
risk = net(features);
riskBinary = round(risk);
```

- The net(features) call uses the trained neural network to predict the risk.
- The result is rounded to a binary value (1 for high risk, 0 for low risk).

Display the Predicted Risk Level:

```
if riskBinary == 1
   riskLevel = 'High';
else
   riskLevel = 'Low';
end
uialert(fig, sprintf('Predicted Risk Level: %s', riskLevel), 'Prediction Result');
```

- The risk is displayed as either 'High' or 'Low' based on the predicted binary value.
- A UI alert is shown to the user with the prediction result.

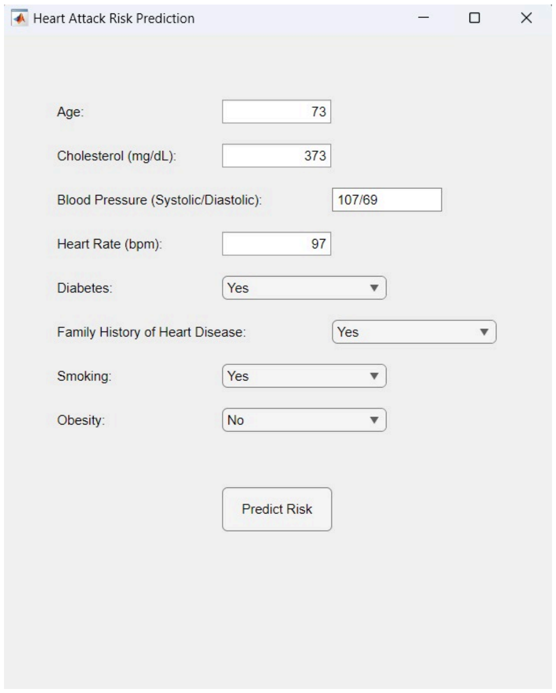
5. **Error Handling**

```
catch ME
   uialert(fig, sprintf('Error: %s', ME.message), 'Error');
end
```

- If any errors occur during input processing, an error message is shown in an alert box to the user.

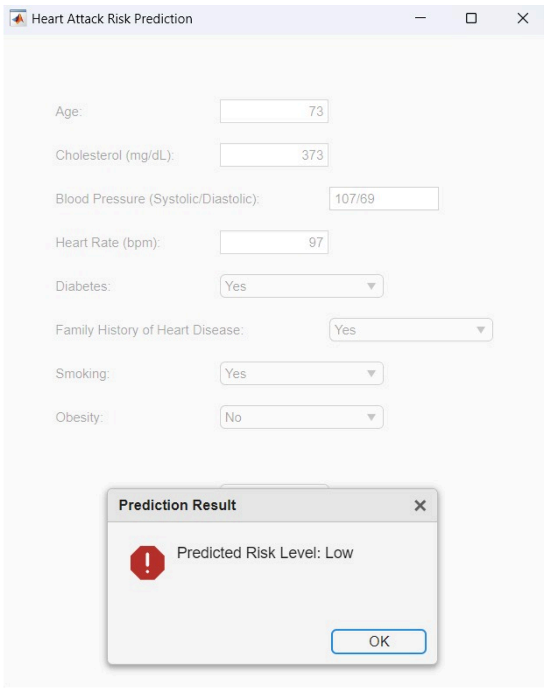
RESULT:

- SIMULATION



Heart Attack Risk Prediction

Age:	73
Cholesterol (mg/dL):	373
Blood Pressure (Systolic/Diastolic):	107/69
Heart Rate (bpm):	97
Diabetes:	Yes
Family History of Heart Disease:	Yes
Smoking:	Yes
Obesity:	No



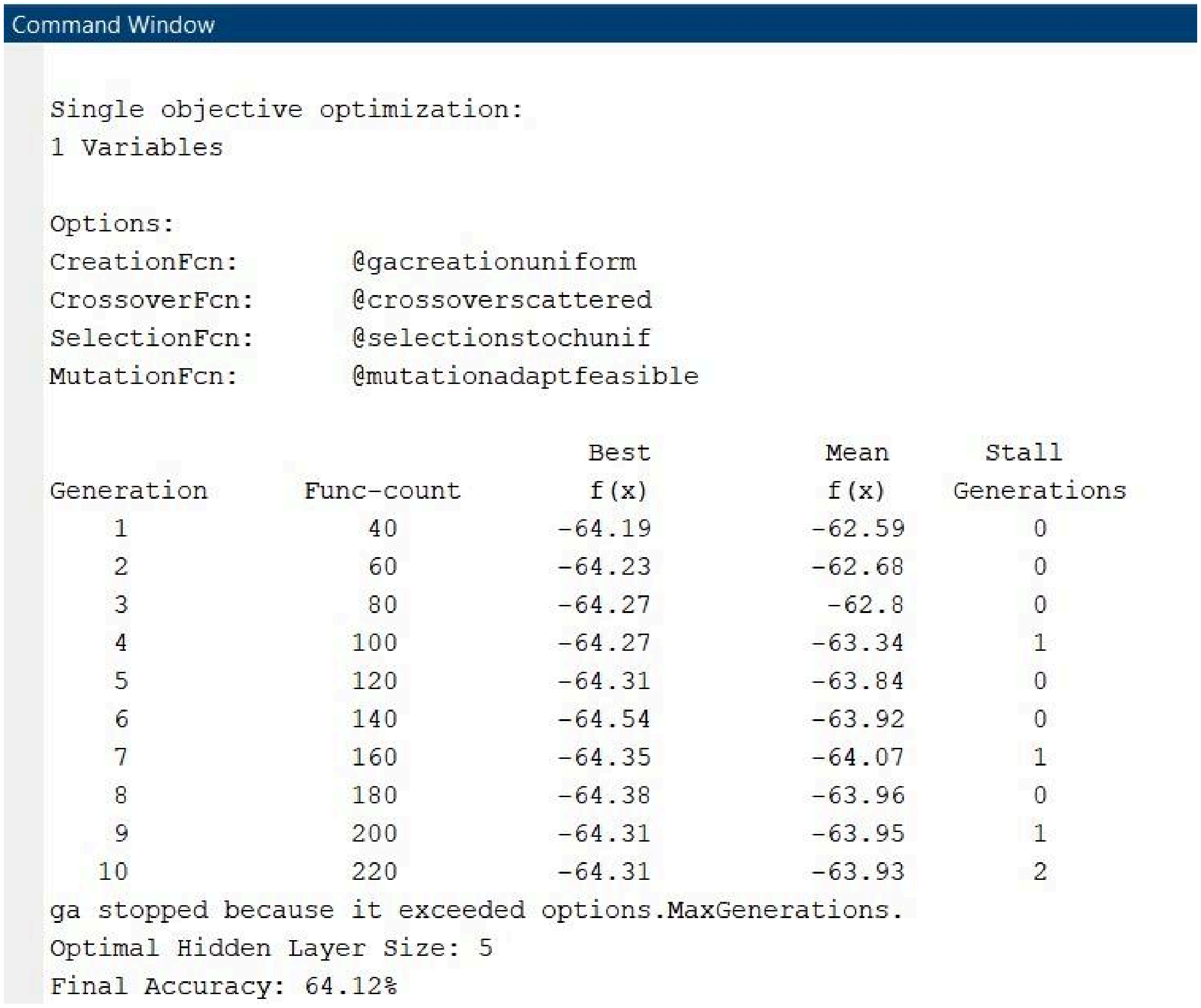
Heart Attack Risk Prediction

Age:	73
Cholesterol (mg/dL):	373
Blood Pressure (Systolic/Diastolic):	107/69
Heart Rate (bpm):	97
Diabetes:	Yes
Family History of Heart Disease:	Yes
Smoking:	Yes
Obesity:	No

Prediction Result

Predicted Risk Level: Low

- EVALUATION



```
Single objective optimization:
1 Variables

Options:
CreationFcn:       @gacreationuniform
CrossoverFcn:      @crossoverscattered
SelectionFcn:      @selectionstochunif
MutationFcn:       @mutationadaptfeasible

                                    Best           Mean      Stall
Generation      Func-count          f(x)           f(x)    Generations
    1              40             -64.19         -62.59        0
    2              60             -64.23         -62.68        0
    3              80             -64.27          -62.8        0
    4             100             -64.27         -63.34        1
    5             120             -64.31         -63.84        0
    6             140             -64.54         -63.92        0
    7             160             -64.35         -64.07        1
    8             180             -64.38         -63.96        0
    9             200             -64.31         -63.95        1
   10             220             -64.31         -63.93        2
ga stopped because it exceeded options.MaxGenerations.
Optimal Hidden Layer Size: 5
Final Accuracy: 64.12%
```

- VALIDATION

The validation process was conducted to assess the effectiveness of the hybrid Genetic Algorithm-Neural Network (GA-NN) model in predicting heart disease risk. The Genetic Algorithm optimised the neural network by determining the optimal hidden layer size, which was identified as 5 after 10 generations. This optimisation enhanced the model's performance by minimizing the objective function to -64.38, as shown in the GA optimisation process. The final accuracy achieved was 64.12%.

To evaluate the model's predictions, a confusion matrix was used, displaying true negatives (1667), false positives (23), false negatives (920), and true positives (18). The model's accuracy, precision, recall, and F1-score were calculated from these values. Accuracy, defined as the proportion of correct predictions, was 64.12%. Precision, representing the proportion of predicted positive cases that were actual positives, was relatively low due to the high number of false negatives. Similarly, the recall (sensitivity) indicated that only a small fraction of actual positive cases were correctly identified, suggesting room for improvement in reducing false negatives. Overall, the GA-NN hybrid system demonstrated moderate performance but requires further optimization to enhance its sensitivity and precision for more reliable heart disease risk prediction.

Conclusion

This project demonstrates a heart attack risk prediction model that effectively combines Genetic Algorithms (GAs) with Artificial Neural Networks (ANNs) to produce precise and understandable forecasts. By determining the ideal hidden layer size, GA's neural network architecture optimization greatly improves the model's performance and raises the classification accuracy of heart attack risk. The ANN-based method produces robust predictions, which guarantees that intricate correlations between lifestyle and medical characteristics are recorded.

A Graphical User Interface (GUI) is incorporated into the model to enhance its usability in addition to its technical capabilities. Users without technical knowledge may easily interact with the system thanks to this functionality. Users may rapidly obtain a risk assessment by entering personal health information, which makes the model useful for practical uses like early health screenings or teaching resources.

To further increase the precision of predictions, future model improvements may concentrate on adding other information, such eating patterns, levels of physical activity, or genetic markers. In order to improve the model's generalisability and guarantee wider application across various demographics and geographical areas, the dataset should be expanded to include a more representative and diverse population. Prediction powers might also be further improved by using cutting-edge strategies like ensemble approaches or deep learning.

To sum up, this project demonstrates how important health issues may be addressed by fusing user-centric design with machine learning methods. This model has the potential to be a useful tool for heart disease risk management and preventive healthcare with further refinement.

APPENDIX

Full Coding

1. **MATLAB coding**

```
clear all
close all
clc

%% Load and preprocess data
filename = 'heart_attack_prediction_dataset_Modified.xlsx'; % specify your file name
opts = detectImportOptions(filename);
opts.VariableNamingRule = 'preserve';
data = readtable(filename, opts);

% Preprocess features
Age = data{:, 1};
Cholesterol = data{:, 3};
BloodPressure = data{:, 4};
HeartRate = data{:, 5};
Diabetes = data{:, 6};
FamilyHistory = data{:, 7};
Smoking = data{:, 8};
Obesity = data{:, 9};


% Blood pressure preprocessing (format: systolic/diastolic)
bp_values = BloodPressure;
map_values = zeros(length(BloodPressure), 1);
for i = 1:length(bp_values)
    bp_split = strsplit(bp_values{i}, '/');
    systolic = str2double(bp_split{1});
    diastolic = str2double(bp_split{2});
    map_values(i) = (2 * diastolic + systolic) / 3;
end



% Prepare feature matrix
input = [Age  Cholesterol map_values ...
    HeartRate Diabetes FamilyHistory Smoking Obesity ];
output = data{:, 25};

% Partition data for training and testing
cv = cvpartition(size(input, 1), 'HoldOut', 0.3);
idx = cv.test;
XTrain = input(~idx, :);
YTrain = output(~idx);
XTest = input(idx, :);
YTest = output(idx);

% Standardize data
meanX = mean(XTrain);
stdX = std(XTrain);
XTrain = (XTrain - meanX) ./ stdX;
XTest = (XTest - meanX) ./ stdX;

% Transpose for compatibility
XTrain = XTrain';
YTrain = YTrain';
XTest = XTest';
YTest = YTest';

%% Genetic Algorithm for Feature Selection
numFeatures = size(XTrain, 1);
options = optimoptions('ga', 'Display', 'iter', 'PopulationSize', 20, 'MaxGenerations', 10);
```

```
[selectedFeatures, ~] = ga(@(features) gaObjective(features, XTrain, YTrain, XTest,
YTest), ...
   numFeatures, [], [], [], [], zeros(1, numFeatures), ones(1, numFeatures), [], options);

% Use selected features
selectedFeatures = logical(round(selectedFeatures));
XTrain = XTrain(selectedFeatures, :);
XTest = XTest(selectedFeatures, :);

%% Train and visualize neural network
hiddenLayerSizes = 10; % Number of neurons
net = feedforwardnet(hiddenLayerSizes);

% Set training options
net.trainFcn = 'trainlm';           % Levenberg-Marquardt
net.trainParam.epochs = 100;        % Number of epochs
net.trainParam.showWindow = true;   % Display training progress

% Train the network
[net, tr] = train(net, XTrain, YTrain);

% Display network diagram
view(net);

% Evaluate performance
YPredTrain = net(XTrain);
YPredTest = net(XTest);
YPredTestBinary = round(YPredTest);

trainAccuracy = sum(round(YPredTrain) == YTrain) / length(YTrain) * 100;
testAccuracy = sum(YPredTestBinary == YTest) / length(YTest) * 100;

fprintf('Training Accuracy: %.2f%%\n', trainAccuracy);
fprintf('Testing Accuracy: %.2f%%\n', testAccuracy);
```

```
% Plot performance
figure;
plotperform(tr);

% Plot regression
figure;
plotregression(YTrain, YPredTrain, 'Training', YTest, YPredTest, 'Testing');

% Plot confusion matrix
figure;
%% Save the trained neural network and standardization parameters
% Save the trained network
save('trained_network.mat', 'net');

% Save standardization parameters
save('standardization_params.mat', 'meanX', 'stdX');
plotconfusion(YTest, YPredTestBinary, 'Test Data');
```

2. **GUI coding**

```
function SimpleHeartRiskGUI
  % Create the main figure window
  fig = uifigure('Name', 'Heart Attack Risk Prediction', 'Position', [100, 100, 500, 600]);

  % Age
  uilabel(fig, 'Text', 'Age:', 'Position', [50, 520, 150, 22]);
  ageField = uieditfield(fig, 'numeric', 'Position', [200, 520, 100, 22]);

  % Cholesterol
  uilabel(fig, 'Text', 'Cholesterol (mg/dL):', 'Position', [50, 480, 150, 22]);
  cholesterolField = uieditfield(fig, 'numeric', 'Position', [200, 480, 100, 22]);

  % Blood Pressure
  uilabel(fig, 'Text', 'Blood Pressure (Systolic/Diastolic):', 'Position', [50, 440, 250, 22]);
```

```
    bpField = uieditfield(fig, 'text', 'Position', [300, 440, 100, 22]);

    % Heart Rate
    uilabel(fig, 'Text', 'Heart Rate (bpm):', 'Position', [50, 400, 150, 22]);
    heartRateField = uieditfield(fig, 'numeric', 'Position', [200, 400, 100, 22]);

    % Diabetes
    uilabel(fig, 'Text', 'Diabetes:', 'Position', [50, 360, 150, 22]);
    diabetesMenu = uidropdown(fig, 'Items', {'Yes', 'No'}, 'Position', [200, 360, 150, 22]);

    % Family History
    uilabel(fig, 'Text', 'Family History of Heart Disease:', 'Position', [50, 320, 250, 22]);
    familyHistoryMenu = uidropdown(fig, 'Items', {'Yes', 'No'}, 'Position', [300, 320, 150,
22]);

    % Smoking
    uilabel(fig, 'Text', 'Smoking:', 'Position', [50, 280, 150, 22]);
    smokingMenu = uidropdown(fig, 'Items', {'Yes', 'No'}, 'Position', [200, 280, 150, 22]);

    % Obesity
    uilabel(fig, 'Text', 'Obesity:', 'Position', [50, 240, 150, 22]);
    obesityMenu = uidropdown(fig, 'Items', {'Yes', 'No'}, 'Position', [200, 240, 150, 22]);

    % Predict Risk Button
    predictButton = uibutton(fig, 'Text', 'Predict Risk', 'Position', [200, 150, 100, 40], ...
        'ButtonPushedFcn', @(btn, event) predictRisk());

    % Callback function for risk prediction
    function predictRisk()
    try
        % Get inputs from the fields
        age = ageField.Value;
        cholesterol = cholesterolField.Value;
        bp = bpField.Value;
        heartRate = heartRateField.Value;
        diabetes = strcmp(diabetesMenu.Value, 'Yes');
        familyHistory = strcmp(familyHistoryMenu.Value, 'Yes');
        smoking = strcmp(smokingMenu.Value, 'Yes');
        obesity = strcmp(obesityMenu.Value, 'Yes');

        % Process blood pressure input
        bp_split = strsplit(bp, '/');
        if numel(bp_split) ~= 2
            error('Invalid blood pressure format. Use "Systolic/Diastolic".');
        end
        systolic = str2double(bp_split{1});
        diastolic = str2double(bp_split{2});
        if isnan(systolic) || isnan(diastolic)
            error('Invalid blood pressure values.');
        end
        map = (2 * diastolic + systolic) / 3;

        % Prepare feature vector
        features = [age, cholesterol, map, heartRate, diabetes, familyHistory, smoking,
obesity];

        % Load the trained neural network and standardization parameters
        load('trained_network.mat', 'net'); % Replace with your saved network
        load('standardization_params.mat', 'meanX', 'stdX');

        % Standardize the features
        if size(features, 2) ~= size(meanX, 2)
            error('Feature vector size does not match the standardization parameters.');
        end
        features = (features - meanX) ./ stdX;

        % Ensure the features are a column vector
        features = features';
        % Predict using the neural network
        risk = net(features);
        riskBinary = round(risk);

        % Display risk level
        if riskBinary == 1
            riskLevel = 'High';
        else
            riskLevel = 'Low';
        end
        uialert(fig, sprintf('Predicted Risk Level: %s', riskLevel), 'Prediction Result');

    catch ME
        uialert(fig, sprintf('Error: %s', ME.message), 'Error');
    end
end
end
```

3. **GA objective**

```
function errorRate = gaObjective(selectedFeatures, XTrain, YTrain, XTest, YTest)
    selectedFeatures = logical(round(selectedFeatures));
    XTrainReduced = XTrain(selectedFeatures, :);
    XTestReduced = XTest(selectedFeatures, :);
    net = feedforwardnet(10); % Define a feedforward network
    net = train(net, XTrainReduced, YTrain);
    YPred = net(XTestReduced);
    YPredBinary = round(YPred);
    errorRate = sum(YPredBinary ~= YTest) / length(YTest); % Minimize error rate
end
```

References

1. *Heart attack Risk Prediction Dataset*. (2024, May 11). Kaggle. https://www.kaggle.com/datasets/iamsouravbanerjee/heart-attack-prediction-dataset
2. Fang, J., Luncheon, C., Ayala, C., Odom, E., & Loustalot, F. (2019). Awareness of heart attack symptoms and response among adults — United States, 2008, 2014, and 2017. *MMWR Morbidity and Mortality Weekly Report*, *68*(5), 101–106. https://doi.org/10.15585/mmwr.mm6805a2
3. Fukuoka, Y., & Oh, Y. J. (2021). Perceived Heart Attack Likelihood in Adults with a High Diabetes Risk. *Heart & Lung*, *52*, 42–47. https://doi.org/10.1016/j.hrtlng.2021.11.007
4. Chrysant, S. G., & Chrysant, G. S. (2014). A healthy lifestyle could reduce the onset of first heart attack by 80%. *Journal of Clinical Hypertension*, *17*(3), 168–171. https://doi.org/10.1111/jch.12466
5. Shahmohamadi, E., Sedaghat, M., Rahmani, A., Larti, F., & Geraiely, B. (2023). “Recognition of heart attack symptoms and treatment-seeking behaviors: a multi-center survey in Tehran, Iran.” *BMC Public Health*, *23*(1). https://doi.org/10.1186/s12889-023-15826-1