COMPUTATIONAL INTELLIGENCE (MCTA 3371)

SEMESTER 1 23/24

MINI PROJECT

TITLE: INTELLIGENT HEART RISK PREDICTION USING COMPUTATIONAL INTELLIGENCE

LECTURER: DR. AZHAR BIN MOHD IBRAHIM

DR. HASAN FIRDAUS BIN MOHD ZAKI

	NAME	MATRIC NUMBER
1.	ALYA SYAFINA BINTI ABDUL RAZAK	2218084
2.	AZLIYANA SYAHIRAH BINTI AZAHARI	2210620
3.	AIN MAISARA BT. ABDULLAH	2217856
4.	ADLIN JOHANA BINTI SHAHRUL NIZAM	2111324
5.	AISYAH SOFEA OTHMAN	2115386

TABLE OF CONTENT

INTRODUCTION

Heart disease is one of the most life threatening illnesses in the world, threatening anyone's life at any time. Heart disease is defined by a wide range of complications that affect the heart and blood arteries, including a condition called coronary artery disease, heart failure, and arrhythmias. Millions of people's lives have been claimed by this disease and major disability have been the consequences. As machine learning and artificial intelligence advances, there is a promising approach to remodel the cardiovascular treatment. By using the ability of neural networks, analyzing and predicting an individual's risk of heart disease.

Neural networks 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.

The findings of this project will aid in advancing the healthcare system by providing more accurate analysis and prediction of an individual's risk of heart disease. The hospital can utilise the results to design suitable treatment for each patient based on their risk which can ultimately reduce the mortality rate due to heart disease. Moreover, these findings can empower patients by shedding new light on the severity of their illness, encouraging them 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 heart risk prediction is vital since it allows for the early identification of those who are at high risk of having a heart attack, allowing for prompt interventions that could save lives. 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.

People's frequent habit of disregarding their health has resulted in an increase in the number of heart disease patients, highlighting the urgent need for more understanding and effective preventive measures. Therefore, this heart risk prediction is an effective motivation for people to become more conscious of the consequences of their lifestyle choices on the condition of their hearts, encouraging people to adopt healthier habits. Similarly, it will help people to understand the need of living a healthy lifestyle in order to reduce their risk of heart disease.

OBJECTIVES

The goal of this study is to explore the effectiveness of neural networks in predicting heart rate variations. Neural networks have the potential to uncover intricate patterns and offer forecasts that can be valuable for medical professionals and individuals. By analyzing historical heart rate data, along with relevant health indicators, this research aims to assess the capability of neural networks in providing accurate predictions of heart rate fluctuations. The utilization of neural networks in heart rate prediction holds the promise of enhancing healthcare monitoring, enabling timely interventions, and improving overall well-being.

Using neural networks, this prediction method aims to address various crucial aspects of heart rate prediction. Heart rate prediction analysis involves estimating future heart patterns using historical data and advanced analytical approaches. The primary purpose of heart rate prediction analysis include :

1. **Health Decision Making** : Heart rate forecasts support individuals and healthcare professionals in making decisions related to health management. Predicting heart rate trends can aid in preventive measures, lifestyle adjustment, and timely medical interventions.

2. **Risk Management** : Heart rate predictions play a role in managing health risks. Individuals can take proactive measures to maintain a healthy heart rate, and healthcare providers can identify potential cardiovascular issues or variations in heart rate patterns for early inventions.

3. **Health Planning** : Accurate heart rate predictions contribute to effective health planning for individuals and healthcare organizations. Individuals can plan their fitness routine, monitor their cardiovascular health, and make lifestyle adjustments based on projected heart rate values.

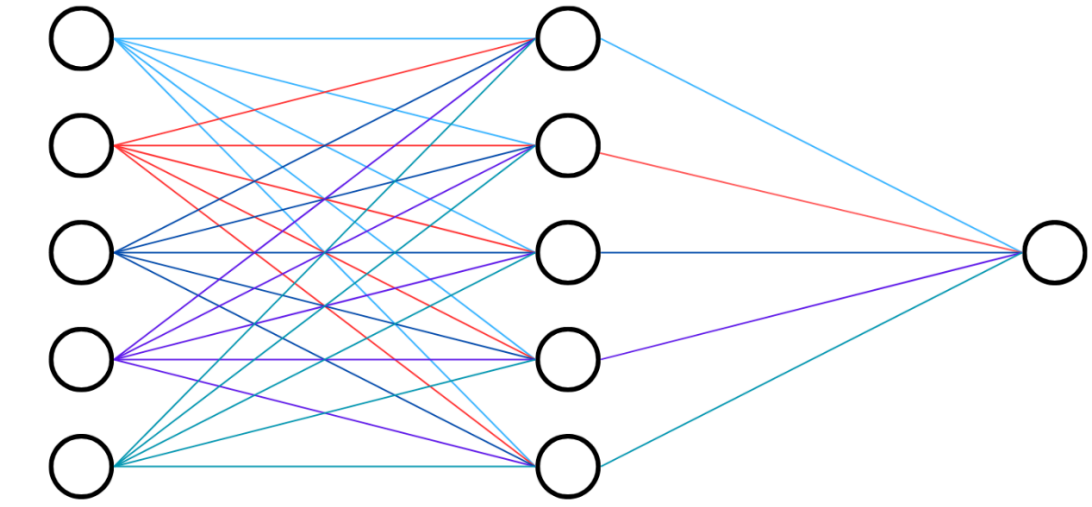
4. **Health Analysis** : Heart rate predictions help enhance the overall understanding of cardiovascular health and related factors. Analyzing past heart rate data can provide insights into lifestyle influences, stress levels, physical activity, and other significant factors affecting heart rate patterns.

METHODOLOGY

We obtained the heart rate prediction dataset from Kaggle. The dataset is a synthetic creation generated using CHATGPT to simulate a realistic experience. In this approach, neural networks are used instead of fuzzy logic . The following are the reasons why neural network are used :

1. **Continuous Learning :** With more data, neural networks can keep learning and getting better, which lets them adapt to new patterns and adjustment in medical conditions over time. This is very important in medical institutions where new patient data and study findings keep coming out.
2. **Scalability :** Neural networks are capable of effectively adapting to the quantity of the dataset, making them well-suited for processing the large volumes of medical data commonly found in studies involving the prediction of heart.
3. **Complex pattern capture :** Neural networks excel in capturing complex patterns and relationships in data. Heart risk prediction requires the examination of numerous correlations among diverse risk indicators, including blood pressure, cholesterol levels, family history, and heart rate.

The following diagram is the neural network model that will be implemented in our prediction system. The design of this neural network has 3 layers.



Input Layer : The initial layer of the neural network is the input layer, tasked with receiving input data. This specific input layer comprises 5 neurons, allowing it to accommodate five distinct input values. Within the input layer, the activation function takes the form of a linear function, a direct linear transformation is applied to the input values, devoid of any introduced non-linearities. Consequently, the output from each neuron in the input layer is a linear combination of its respective inputs.

Hidden Layer : The hidden layers within a neural network play a crucial role in processing incoming data and extracting key information. In this instance, the hidden layer consists of five neurons, and its activation function is defined by the ReLu function, expressed as $y = \max(0, X)$. The utilization of the ReLu activation function is that introduces the property of non-linearity to deep learning and solves the vanishing gradients issues. It interprets the positive part of its argument and is one of the most popular activation functions in deep learning. The choice facilitates the extraction of significant features and intricate relationships from input data as it traverses through the hidden layer.

Output Layer : The final layer of the neural network is the output layer, tasked with delivering the desired output derived from the information processed in the hidden layers. In this illustration, the output layer consists of one neuron, producing a single output value. The activation function employed in the output layer is characterized by the sigmoid function, defined as $y=1/(1+ e-x)$.

The forward method executes the neural networks forward feed forward, involving matrix multiplication and the application of the ReLu and sigmoid activation function at the hidden and output layers. In the backward pass, backpropagation is employed to adjust the neural network's weight. This process entails computing the disparity between the anticipated and actual output, applying the derivative of the ReLu and sigmoid function to the error, and subsequently propagating the error backward through the network. With a learning rate set as 0.5, the neural network undergoes training through a loop. In each iteration, the train method is called to further refine the network's weight.

IMPLEMENTATION DETAILS

Python is used in the implementation of the heart risk prediction system. This algorithm estimates the risk of a heart attack based on heart rate, cholesterol, age, sex, and diabetes. Getting a sample of data is the first stage. The neural network system needs this data in order to be trained. The elements are entered into the system's database in array format. Subsequently, the entered data is normalized to lie between 0 and 1.

The system will go through incremental training to modify the weights between neural network layers before beginning the prediction phase. The neural network's inputs are the normalized values for age, sex, cholesterol, heart rate, and diabetes.The program will compute the estimated risk of a heart attack once these inputs are subjected to forward propagation using set activation functions. Next, in order to get the mean absolute squared loss, the algorithm will compare the real heart risk with the projected heart risk. The weights of the neural network are then updated using this loss value. To get the most appropriate weight values, this process is repeated for each sample set of data.

When the neural network has completed its training, the program is prepared to generate predictions. Utilizing forward propagation, the program generates the patient's estimated heart attack risk based on input parameters such as age, sex, cholesterol level, heart rate, and diabetes.

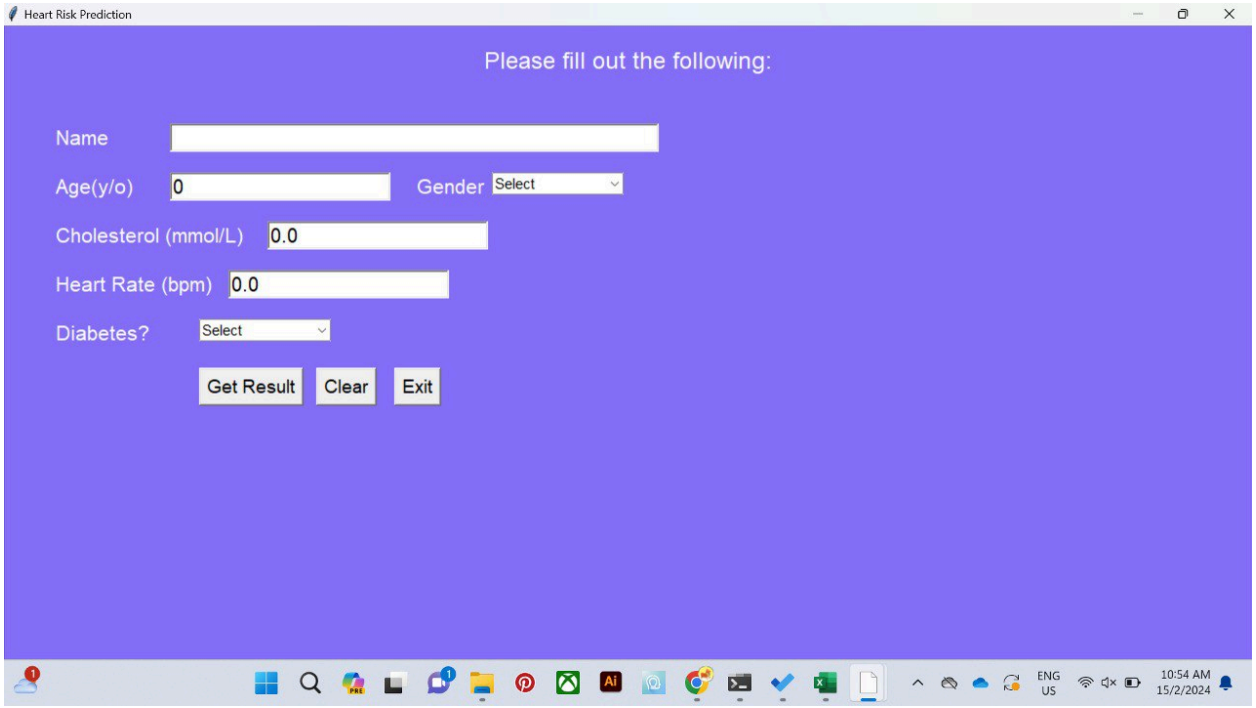
SIMULATION

Using google colab, a cloud based platform provided by Google that allows users to run Python codes in a Jupyter Notebook environment. First of all, we need to upload a file to the system by clicking ‘file upload’ in google colaboratory allowing users to visualize the inputs. The code should be executed by the user to facilitate the file uploading and display the input. Ni direct input is required from the user. The files are in comma-separated values(csv) format, as it is the only type compatible with the utilized code.

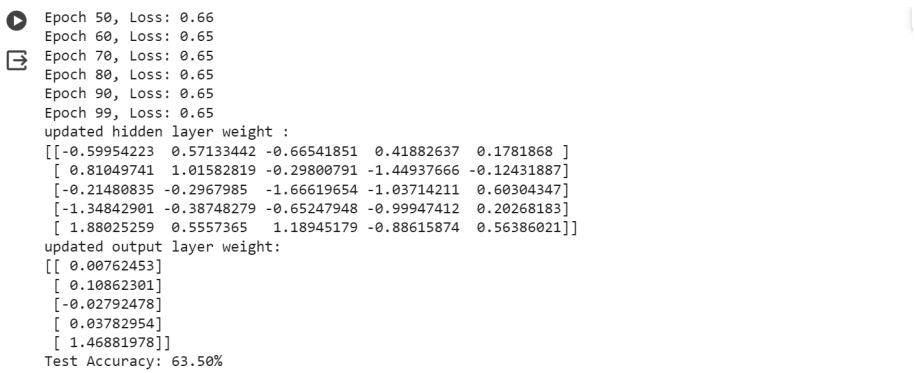
Next, users are required to initiate input prompts, providing data such as age, heart rate, blood pressure, and cholesterol. This input is then utilized by the neural network to predict heart rate values based on its learned patterns and training.

```
HEART RISK PREDICTION
For this prediction please input numbers
REFERENCE
Male= 1  Female=0
Diabetes
No=0 Yes=1
Please input Age Sex Cholesterol Heart_Rate Diabetes in order separated by spaces:
```

Based on our Graphical User Interface, users have to key in their name, age, gender, cholesterol level, heart rate and diabetes. Users have to select gender and either they have diabetes or not.



EVALUATION



```
Epoch 50, Loss: 0.66
Epoch 60, Loss: 0.65
Epoch 70, Loss: 0.65
Epoch 80, Loss: 0.65
Epoch 90, Loss: 0.65
Epoch 99, Loss: 0.65
updated hidden layer weight :
[[-0.59954223  0.57133442 -0.66541851  0.41882637  0.1781868 ]
 [ 0.81049741  1.01582819 -0.29800791 -1.44937666 -0.12431887]
 [-0.21480835 -0.2967985  -1.66619654 -1.03714211  0.60304347]
 [-1.34842901 -0.38748279 -0.65247948 -0.99947412  0.20268183]
 [ 1.88025259  0.5557365   1.18945179 -0.88615874  0.56386021]]
updated output layer weight:
[[ 0.00762453]
 [ 0.10862301]
 [-0.02792478]
 [ 0.03782954]
 [ 1.46881978]]
Test Accuracy: 63.50%
```

In this simulation, the loop iterates 200 times. An epoch signifies a full iteration through the entire training dataset, where the model processes the complete dataset, generates predictions, computes the error, and adjusts the weights. Conducting multiple epochs is typically essential to improve the model's capacity to generalize patterns from the data.

Loss represents the measure of how well the model’s predictions match the actual values in the raining dataset. In the simulation, we can see positive progress in the training process as the value loss decreases progressively. Decreasing loss shows the difference between the predicted output and the true target values. The goal during training is to minimize the loss, indicating that the model is improving in making accurate predictions.

VALIDATION

The ability of the neural network to identify complex data and recognize output patterns is the significance of our research. The results we found can help us better comprehend the trend and create a data system that is more accurate. Neural networks can be trained to forecast future data based on historical data since they are predictive analytics as well. For this project, the prediction on the error or loss declines gradually, whilst the test accuracy reached at 63.50%. This means that the test prediction is substantial.

A few variables may have an impact on the data. The training data comes first. the quality, quantity, and diversity of training material that neural networks are trained on. The accuracy of the heart attack risk increases with training data volume. The second is the length of training. The performance of the network can be impacted by how long it is trained. If the training time is too long, the network will memorise data without making good generalisations, and insufficient training data will prevent the network from capturing the output pattern. Furthermore, the chance of having a heart attack is a reflection of what people expect to happen to their health. Due to the subjectivity of their beliefs and opinions, the heart attack risk prediction would be affected.

There are certain drawbacks in the model and system that make it less successful than other heart attack prediction models. The system's drawback is that, while we create the system to only anticipate the presence of heart risk, the user can only predict whether the risk of a heart attack is great or low, with nothing in between. Next, in order for Neural Networks to learn and produce correct predictions, a substantial volume of training data is required. There is a greater likelihood of improved accuracy in the estimated cardiac risk the more training data we use.

CONCLUSION

In conclusion, this study paper examined the use of neural network models for heart attack risk prediction analysis; however, the findings indicate that the predictions are not perfect. Although we were able to use a neural network to successfully achieve the results, the accuracy is not perfect. These predictions, on the other hand, can be regularly utilised by healthcare workers, doctors can use it to anticipate the heart risk of each patient. They are capable of paying closer attention to their health by taking the risk of a heart attack into consideration. The need for risk management and approaches to decision-making that allow for uncertainty and unforeseen developments is also emphasized by this study. Additionally, this technology can assist professionals by allowing them to prescribe medication to their patients depending on the prevailing underlying conditions.

We should use input data from a wider range of elements to achieve more accurate forecasts in order to increase the accuracy of our heart attack risk prediction. In order to make an accurate prediction of heart attack risk in the future, we should also take into account and test alternative soft computing models. We should carefully consider the learning rate and network architecture to lower prediction errors and enhance system performance. If we heed the advice and stay away from any errors that can affect the result, we can create a prediction system that is more accurate. Prediction analysis is dependent on numerous aspects and can never be 100% accurate, thus it is important to consider our health carefully.

CONTRIBUTIONS

Contributions	Alya	Maisara	Azliyana	Adlin	Sofea
Project Execution	Transferring and Collecting Data Planning soft computing model Collecting data Compiling the collected data Slide Presentation	Prepare python user interface code for testing purposes Troubleshoot coding Planning soft computing modelling Compile the collected data into a table Slide Presentation	Prepare Neural network python coding Prepare python code for GUI Troubleshoot coding Planning soft computing modelling Slide Presentation	Collecting data Compiling the collected data Planning soft computing model Slide Presentation	Collecting data Compiling the collected data Transferring and Collecting Data Planning soft computing modelling Slide Presentation
Report	Introduction Methodology Proofreading Conclusion	Simulation (Findings/Result) References Proofreading	Implementation details Simulation (Findings/Result) Proofreading	Validation Conclusion Implementation details Proofreading	Introduction Objectives Methodology Proofreading Evaluation Simulation

APPENDIX

PYTHON CODE FOR NEURAL NETWORK AND USER INTERFACE FOR TESTING PURPOSES

```
from google.colab import files
import numpy as np
import pandas as pd

data = files.upload()
df=pd.read_csv("heart_attack_prediction_dataset.csv")
df

x = pd.DataFrame(df.iloc[: ,1:6].values) #df.iloc[rowstart:rowend ,
columnstart:columnend]
y = df.iloc[:, -1:].values #last column (output)
print(x)
print(y)
print('\n')

from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import LabelEncoder, normalize
labelencoder_X = LabelEncoder()
x.loc[:, 1] = labelencoder_X.fit_transform(x.iloc[:, 1]) #gender
#x.iloc[:, 1:3] = normalize(x.iloc[:, 1:3], axis=0).round(4) #cholesterol and
heart rate

print(x)

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(x) #standardize inputs (x)

# Split the data into training (%80) and testing sets (%20)
from sklearn.model_selection import train_test_split
x_train_scaled, x_test_scaled, y_train, y_test = train_test_split(X_scaled, y,
```

```
test_size=0.2, random_state=0)

class NeuralNetwork:
    def __init__(self, input_size, hidden_size, output_size):
        self.input_size = input_size
        self.hidden_size = hidden_size
        self.output_size = output_size

        # Initialize weights and biases
        self.W1 = np.random.randn(input_size, hidden_size)
        self.b1 = np.zeros((1, hidden_size))
        self.W2 = np.random.randn(hidden_size, output_size)
        self.b2 = np.zeros((1, output_size))

    def get_weights(self):
        return self.W1, self.W2

    def get_biases(self):
        return self.b1, self.b2

    def relu(self, x):
        return np.maximum(0, x)

    def sigmoid(self, x):
        return 1 / (1 + np.exp(-x))

    def forward_propagation(self, X):
        # Input layer to hidden layer
        self.z1 = np.dot(X, self.W1) + self.b1
        self.a1 = self.relu(self.z1)
        # Hidden layer to output layer
        self.z2 = np.dot(self.a1, self.W2) + self.b2
        self.a2 = self.sigmoid(self.z2)
        return self.a2

    def backward_propagation(self, X, y, learning_rate):
        m = len(X) #number of sample data
        # Compute gradients
```

```
        dZ2 = self.a2 - y
        dW2 = (1 / m) * np.dot(self.a1.T, dZ2)
        db2 = (1 / m) * np.sum(dZ2, axis=0, keepdims=True) #keepdims keep the
dimension same
        dZ1 = np.dot(dZ2, self.W2.T) * (self.a1 > 0)
        dW1 = (1 / m) * np.dot(X.T, dZ1)
        db1 = (1 / m) * np.sum(dZ1, axis=0, keepdims=True)
        # Update weights and biases
        self.W1 -= learning_rate * dW1
        self.b1 -= learning_rate * db1
        self.W2 -= learning_rate * dW2
        self.b2 -= learning_rate * db2

    def train(self, X, y, learning_rate, num_epochs):
        for epoch in range(num_epochs):
            # Forward propagation
            self.forward_propagation(X)
            # Backward propagation
            self.backward_propagation(X, y, learning_rate)

            loss = np.mean(-y * np.log(self.a2) - (1 - y) * np.log(1 - self.a2))
            if epoch % 10 == 0:
                print(f"Epoch {epoch}, Loss: {loss:.2f}")
            elif epoch == num_epochs -1:
                print(f"Epoch {epoch}, Loss: {loss:.2f}")

    def predict(self, X):
        return np.round(self.forward_propagation(X))

input_size = 5
hidden_size = 5
output_size = 1
learning_rate = 0.01
num_epochs = 100

model = NeuralNetwork(input_size, hidden_size, output_size)

# Train the model
```

```
model.train(x_train_scaled, y_train, learning_rate, num_epochs)

# Evaluate the model
predictions = model.predict(x_test_scaled)
accuracy = np.mean(predictions.flatten() == y_test) #accuracy = total of correct
predictions / total of predictions
weights = model.get_weights()
biases = model.get_biases()

# Print weights and biases
print("Weights:")
print("Hidden Layer :", weights[0])
print("Output Layer :", weights[1])
print("Biases:")
print("Hidden Layer:", biases[0])
print("Output Layer :", biases[1])

print(f'Test Accuracy: {accuracy:.2f}')

print('HEART RISK PREDICTION')
print("For this prediction please input numbers\nREFERENCE\nMale= 1
Female=0\nDiabetes\nNo=0 Yes=1\n")
predict_input = input("Please input age Sex Cholesterol Heart_Rate Diabetes in
order separated by spaces: ")
predict_list = list(map(float, predict_input.split()))

# Standardize the input data
predict_scaled = scaler.transform([predict_list])

# Predict the risk using the trained model
predicted_risk = model.predict(predict_scaled)
print(f"Heart Risk Detection: {predicted_risk[0][0]:.2f}")
if predicted_risk[0][0] > 0.5:
    print('HIGH RISK')
else:
    print('LOW RISK')
```

PYTHON CODE FOR NEURAL NETWORK AND USER INTERFACE FOR GUI

```
import tkinter as tk
from tkinter import ttk
from tkinter.ttk import Combobox
from tkinter import messagebox
import pathlib
import numpy as np
import pandas as pd
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, LabelEncoder, normalize, StandardScaler

class HeartAttackPredictionGUI:

    def __init__(self):
        self.root = tk.Tk()
        self.root.title("Heart Risk Prediction")
        self.root.geometry("700x400")
        self.root.configure(bg="#856ff8")

        self.menubar = tk.Menu(self.root)

        self.filemenu = tk.Menu(self.menubar, tearoff=0)
        self.filemenu.add_command(label="Save", command=self.exit)
        self.filemenu.add_command(label="Close", command=self.exit)

        self.menubar.add_cascade(menu=self.filemenu, label="Home")
        self.root.config(menu=self.menubar)
        self.menubar.add_cascade(menu=self.filemenu, label="Login")
        self.root.config(menu=self.menubar)

        self.label = tk.Label(self.root, text=" ", font=('Futura',18), bg="#856ff8",
fg="#fff").pack(padx=20, pady=20)
        self.label = tk.Label(self.root, text="Heart Risk Prediction", font=('Futura',18),
bg="#856ff8", fg="#fff")
        self.label.pack(padx=20, pady=5)

        self.label = tk.Label(self.root, text="'Making sure your heart stays young and healthy!'",
font=('Futura',12), bg="#856ff8", fg="#fff")
        self.label.pack(padx=20, pady=5)

        self.button = tk.Button(self.root, text="Let's get started!", font=('Futura',14),
bg="#faf9f6", fg="#6330ff", command=self.start)
        self.button.pack(padx=5,pady=30)
```

```
        self.root.mainloop()

    def start(self):
        start_window = tk.Toplevel()
        start_window.title('Heart Risk Prediction')
        start_window.geometry('700x400')
        start_window.configure(bg='#856ff8')

        start_window.label = tk.Label(start_window, text="", font=('Futura',18), bg="#856ff8",
fg="#fff")
        start_window.label = tk.Label(start_window, text="Please fill out the following:",
font=('Futura',18), bg="#856ff8", fg="#fff")
        start_window.label.pack(padx=20, pady=20)

        #Label
        start_window.label = tk.Label(start_window, text="Name", font=('Futura',16),
bg="#856ff8", fg="#fff")
        start_window.label.place(x=50, y=100)
        start_window.label = tk.Label(start_window, text="Age(y/o)", font=('Futura',16),
bg="#856ff8", fg="#fff")
        start_window.label.place(x=50, y=150)
        start_window.label = tk.Label(start_window, text="Gender", font=('Futura',16),
bg="#856ff8", fg="#fff")
        start_window.label.place(x=420, y=150)
        start_window.label = tk.Label(start_window, text="Cholesterol (mmol/L)",
font=('Futura',16), bg="#856ff8", fg="#fff")
        start_window.label.place(x=50, y=200)
        start_window.label = tk.Label(start_window, text="Heart Rate (bpm)", font=('Futura',16),
bg="#856ff8", fg="#fff")
        start_window.label.place(x=50, y=250)
        start_window.label = tk.Label(start_window, text="Diabetes?", font=('Futura',16),
bg="#856ff8", fg="#fff")
        start_window.label.place(x=50, y=300)


        #Entry
        self.nameValue = tk.StringVar()
        self.ageValue = tk.IntVar()
        self.cholesterolValue = tk.DoubleVar()
        self.heart_rateValue = tk.DoubleVar()

        def clear():
            nameEntry.delete(0, tk.END)
            ageEntry.delete(0, tk.END)
            cholesterolEntry.delete(0, tk.END)
```

```
            heart_rateEntry.delete(0, tk.END)
            self.gender_dropdown.set('')
            self.diabetes_dropdown.set('')

        nameEntry = tk.Entry(start_window, textvariable=self.nameValue, width=45,bd=2,
font=15)
        ageEntry = tk.Entry(start_window, textvariable=self.ageValue, width=20, bd=2, font=15)
        cholesterolEntry = tk.Entry(start_window, textvariable=self.cholesterolValue, width=20,
bd=2, font=15)
        heart_rateEntry = tk.Entry(start_window, textvariable=self.heart_rateValue, width=20,
bd=2, font=15)

        nameEntry.place(x=170,y=100)
        ageEntry.place(x=170,y=150)
        cholesterolEntry.place(x=270,y=200)
        heart_rateEntry.place(x=230,y=250)

        #gender dropdown
        self.gender_dropdown = Combobox(start_window, values= ['Male','Female'],
font='Futura 11', state='r', width=14)
        self.gender_dropdown.place(x=500, y=150)
        self.gender_dropdown.set('Select')

        #Diabetes dropdown
        self.diabetes_dropdown = Combobox(start_window, values= ['Yes','No'], font='Futura 11',
state='r', width=14)
        self.diabetes_dropdown.place(x=200, y=300)
        self.diabetes_dropdown.set('Select')

        self.button = tk.Button(start_window, text="Get Result", font=('Futura',14),
command=self.submit)
        self.button.place(x=200, y=350)
        self.button = tk.Button(start_window, text="Clear", font=('Futura',14), command=clear)
        self.button.place(x=320, y=350)
        self.button = tk.Button(start_window, text="Exit", font=('Futura',14), command=self.exit)
        self.button.place(x=400, y=350)

    # Layer function
    def relu(self, x):
        return np.maximum(0, x)

    def sigmoid(self,y):
        return 1 / (1 + np.exp(-y))

    def submit(self):
        # Here, you would process the input data and calculate the probability of a heart attack
```

```
result_window = tk.Toplevel()
result_window.title('Heart Attack Probability')
result_window.geometry('700x400')
result_window.configure(bg='#faf9f6')

self.menubar = tk.Menu(self.root)

self.filemenu = tk.Menu(self.menubar, tearoff=0)
self.filemenu.add_command(label="Save", command=self.exit)
self.filemenu.add_command(label="Close", command=self.exit)

self.menubar.add_cascade(menu=self.filemenu, label="Home")
self.root.config(menu=self.menubar)
self.menubar.add_cascade(menu=self.filemenu, label="Login")
self.root.config(menu=self.menubar)

# Fetching input data
name = self.nameValue.get()
age = self.ageValue.get()
cholesterol = self.cholesterolValue.get()
heart_rate = self.heart_rateValue.get()
gender = self.gender_dropdown.get()
diabetes = self.diabetes_dropdown.get()

#assign as 0 and 1
gender_num = 0 if gender == 'Male' else 1
diabetes_num = 0 if diabetes == 'No' else 1

#Calculation using neural network:

# Normalizing cholesterol
cholesterol_normalized = normalize([[cholesterol]], axis=0).round(4)
cholesterol_num = cholesterol_normalized.item()  # Extracting the scalar value

# Normalizing heart rate
heart_rate_normalized = normalize([[heart_rate]], axis=0).round(4)
heart_rate_num = heart_rate_normalized.item()  # Extracting the scalar value


# Store the output data (excluding name) in a list
output_data = [
    [age],
    [gender_num],
    [cholesterol_num],
    [heart_rate_num],
```

```
    [diabetes_num]
]

# Convert the list to a numpy array
x = np.array(output_data)

# Initialize weight transverse matrix values
weight1_values = [
    [0.948,   -0.056,   -1.092,   1.157,   0.491],
    [1.859,   -0.933,   -0.025,   0.242,  -1.03],
    [0.868,   -1.240,   -1.845,  -0.131,  -0.56],
    [2.289,   -0.5,     -0.759,   1.779,   1.048],
    [1.528,   -0.575,   -0.920,  -2.214,   1.336]
]
weight1 = np.array(weight1_values)

weight2_values = [
    [-0.246, -0.078, 0.087, 0.098,-0.022],
]
weight2 = np.array(weight2_values)

# Initialize bias matrix values
bias1_values = [
    [-0.117],
    [-0.510],
    [-0.711],
    [0.281],
    [-0.286]
]
bias1 = np.array(bias1_values)

bias2_values = [
    [-0.287]
]
bias2 = np.array(bias2_values)

# Feed Forward Propagation
hiddenL_i = np.dot(weight1, x) + bias1
hiddenL_o = self.relu(hiddenL_i)
output_i = np.dot(weight2, hiddenL_o) + bias2
output_o = self.sigmoid(output_i)

# Round all variables to three decimal places
hiddenL_i = np.round(hiddenL_i, 3)
hiddenL_o = np.round(hiddenL_o, 3)
output_i = np.round(output_i, 3)
```

```
        output_o = np.round(output_o, 3)

        if output_o[0] > 0.5:
            heart_risk = 'HIGH RISK'
        else:
            heart_risk = 'LOW RISK'


        # Showing input data in a message box
        for widget in result_window.winfo_children():
            widget.destroy()

        # Display input data on result window
        tk.Label(result_window, text = "").pack(padx=10, pady=10)
        tk.Label(result_window, text = "Heart Risk Prediction Result:", font=('Futura', 18),
bg="#faf9f6", fg="#6330ff").pack(padx=10, pady=20)
        tk.Label(result_window, text = f" {heart_risk} ", font=('Futura', 18), bg="#6330ff",
fg="#faf9f6").pack(padx=10, pady=20)
        tk.Label(result_window, text = f"Name: {name}", font=('Futura', 12), bg="#faf9f6",
fg="#6330ff").pack(padx=10, pady=5)
        tk.Label(result_window, text = f"Age: {age} years old", font=('Futura', 12), bg="#faf9f6",
fg="#6330ff").pack(padx=10, pady=5)
        tk.Label(result_window, text = f"Gender: {gender}", font=('Futura', 12), bg="#faf9f6",
fg="#6330ff").pack(padx=10, pady=5)
        tk.Label(result_window, text = f"Cholesterol: {cholesterol} mmol/L", font=('Futura', 12),
bg="#faf9f6", fg="#6330ff").pack(padx=10, pady=5)
        tk.Label(result_window, text = f"Heart Rate: {heart_rate} bpm", font=('Futura', 12),
bg="#faf9f6", fg="#6330ff").pack(padx=10, pady=5)
        tk.Label(result_window, text = f"Diabetes?: {'No' if diabetes_num == 0 else 'Yes'}",
font=('Futura', 12), bg="#faf9f6", fg="#6330ff").pack(padx=10, pady=5)

        print("Input Data", f"Name: {name}\nAge: {age} years old\nGender:
{gender_num}\nCholesterol: {cholesterol} mmol/L\nHeart Rate: {heart_rate} bpm\nDiabetes:
{diabetes_num}")
        print("Output Array:", x)
        print("\nHidden Input:\n\n", hiddenL_i)
        print("\nOutput I:\n\n", output_i)
        print("\nRelu:\n\n", hiddenL_o)
        print("\nSigmoid:\n\n", output_o)
        print("\nHeart Risk:", heart_risk)


    def exit(self):
        self.root.destroy()

HeartAttackPredictionGUI()
```

REFERENCES

Building a neural network FROM SCRATCH (no Tensorflow/Pytorch, just numpy & math). (2017, November 9). YouTube: Home. Retrieved February 15, 2024, from https://youtu.be/w8yWXqWQYmU?si=HhACOYqKHEGfPpvR

Heart Attack Risk Prediction Dataset. (n.d.). Kaggle. Retrieved February 15, 2024, from https://www.kaggle.com/datasets/iamsouravbanerjee/heart-attack-prediction-dataset

Implementing Artificial Neural Network(Classification) in Python From Scratch. (n.d.). Analytics Vidhya. Retrieved February 15, 2024, from https://www.analyticsvidhya.com/blog/2021/10/implementing-artificial-neural-networkclassification-in-python-from-scratch/

Turning a Google Colab Notebook into a Web App - With Nothing But Python. (2017, November 9). YouTube: Home. Retrieved February 15, 2024, from https://youtu.be/ivWp6XTtFjo?si=qoaJFZpVGDrpUqKj