BUILDERS ENGINEERING COLLEGE
Approved by AICTE, New Delhi | Affiliated to Anna University, Chennai
ISO Certified Institution | Recognized 2(f) Status by UGC
CIVIL, CSE, ECE, EEE & MECH - Accredited by NBA
EBET Knowledge Park, Nathakadaiyur, Kangeyam,
Tirupur - 638 108, Tamil Nadu, India.
AUTONOMOUS

Naan Mudhalvan Project Report

Name of the Student	**:**	R.Suruthi
Student Register Number	**:**	730321104056
Year/Semester:	**:**	III/VI
Department	**:**	Computer Science and Engineering
Naan Mudhalvan Id	**:**	au730321104056
Student Mail I'd	**:**	suruthir344@gmail.com
Course Code & Name	**:**	NM1009 & Generative AI for Engineering
Date of Submission	**:**	03-05-2024
Project Title	**:**	Heart disease analysis using RNN

Student's Sign

HoD **Course Coordinator sign** **SPOC**

Naan Mudhalvan -NM1009 Generative AI for Engineering

PROJECT TITLE : Heart disease analysis using RNN

Presented By :-
Name : Suruthi R
Reg No : 730321104056
Department : Computer science and engineering

AGENDA :

- *Problem Statement*
- *Project Overview*
- *End Users*
- *Our Solution*
- *Special Features*
- *Modelling Approach*
- *Results*
- *Conclusion*

PROBLEM STATEMENT :

- ✓ Heart disease is a leading cause of mortality worldwide, necessitating accurate and timely diagnosis for effective treatment and prevention.
- ✓ Traditional methods of diagnosis often rely on static measurements and do not fully leverage the sequential nature of patient data over time.
- ✓ This project aims to develop a predictive model using Recurrent Neural Networks (RNNs) to analyze longitudinal patient data and accurately predict the likelihood of heart disease occurrence or progression.

PROJECT OVERVIEW :

- ✓ project aims to develop a RNN-based system for Heart disease analysis.

- ✓ Utilize a dataset containing patient demographics, medical history, and physiological measurements to train and evaluate the RNN model.

- ✓ Employ an RNN architecture capable of capturing temporal dependencies within sequential patient data.

- ✓ Identify areas for further research, such as model refinement, integration of additional data sources, and exploration of real-time monitoring capabilities.

WHO ARE THE END USERS?

- ✓ Healthcare Professionals
- ✓ Patients
- ✓ Healthcare Institutions
- ✓ Medical Researchers
- ✓ Public Health Organizations

SOLUTION AND ITS VALUE PROPOSITION

- ✓ A heart disease analysis system offers early detection, personalized risk assessment, treatment planning, and data-driven decision-making for improved patient outcomes. It also fuels research and innovation in cardiovascular health.

- ✓ A recurrent neural network (RNN) for heart disease analysis offers continuous monitoring and prediction capabilities, allowing for early detection of abnormalities and personalized risk assessment. By analyzing longitudinal patient data, it provides timely interventions, personalized treatment plans, and data-driven decision-making for improved patient outcomes and resource optimization.

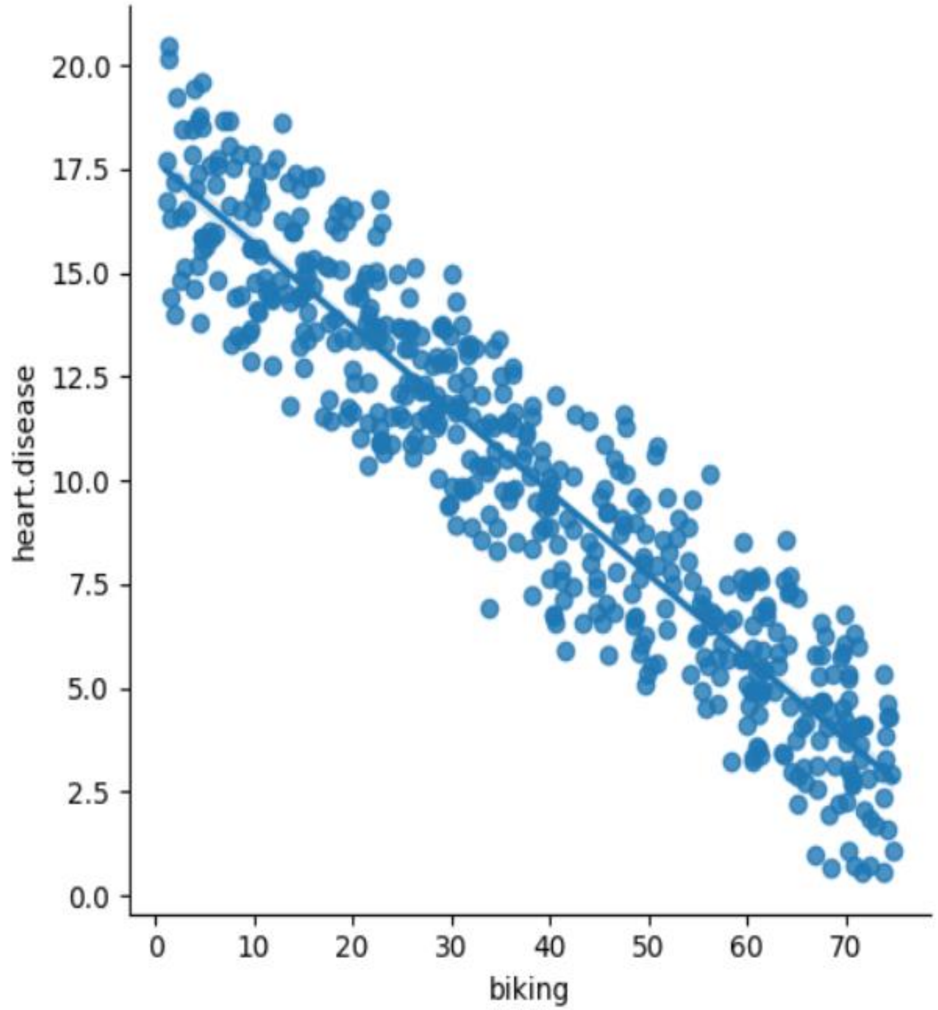
SPECIAL FEATURES :

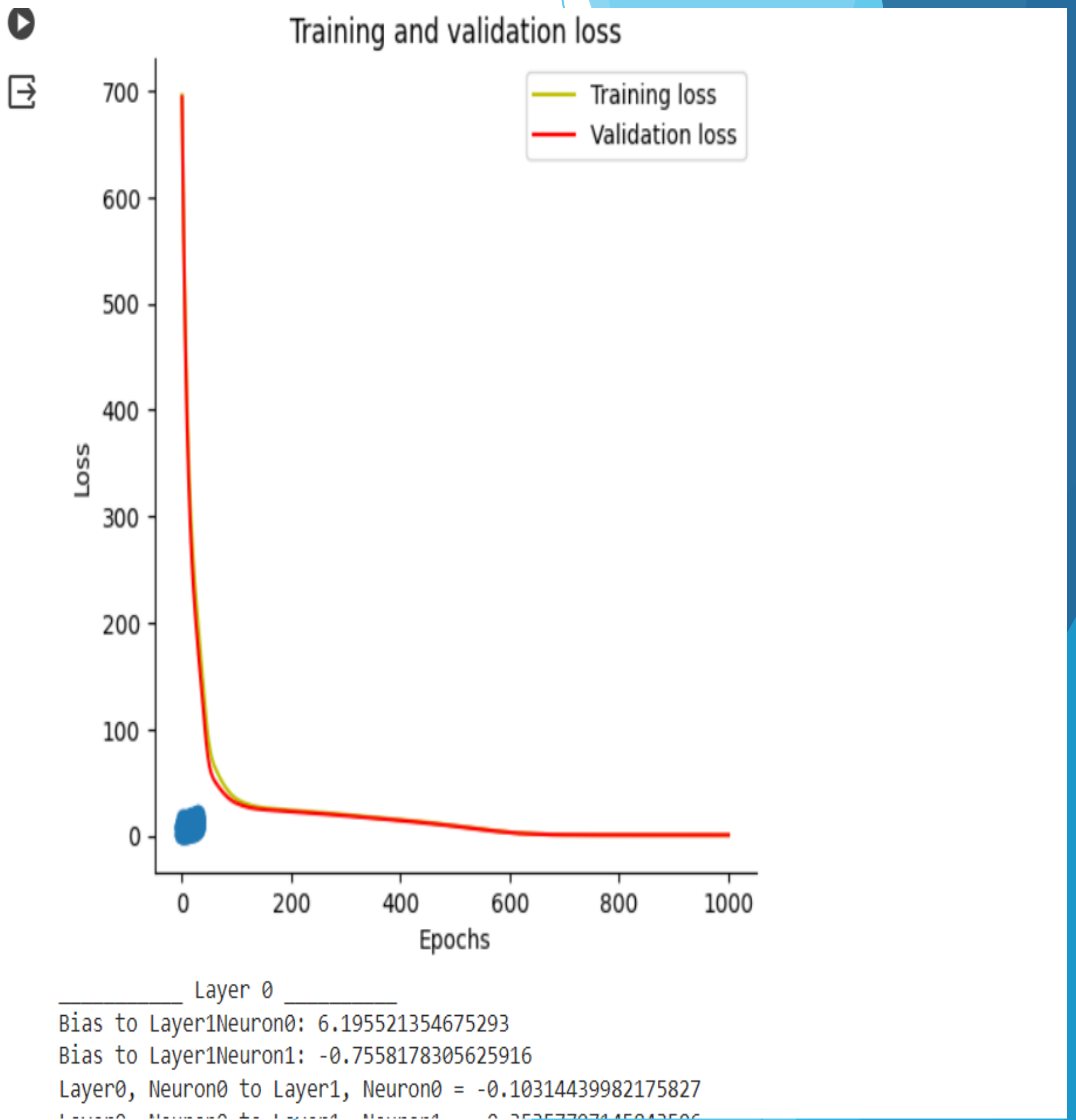
- ✓ heart disease analysis system could be real-time monitoring of vital signs and symptoms, using wearable devices or sensors, coupled with AI algorithms to detect early signs of heart issues and provide timely alerts or recommendations for medical intervention.

MODELLING :

- ✓ Utilizing a recurrent neural network (RNN) for heart disease analysis could involve capturing temporal dependencies in patient data, such as heart rate variability over time.
- ✓ The RNN could process sequences of medical data to detect patterns or anomalies indicative of heart disease progression or risk factors.
- ✓ Additionally, incorporating natural language processing (NLP) capabilities could enable the model to analyze textual patient records or medical literature for further insights.

RESULTS :-


heart.disease
biking
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
10
20
30
40
50
60
70


Training and validation loss
Training loss
Validation loss
Loss
Epochs
700
600
500
400
300
200
100
0
0
200
400
600
800
1000
__________ Layer 0 __________
Bias to Layer1Neuron0: 6.195521354675293
Bias to Layer1Neuron1: -0.7558178305625916
Layer0, Neuron0 to Layer1, Neuron0 = -0.10314439982175827

CONCLUSION :

- ✓ In conclusion, employing a recurrent natural network (RNN) for heart disease analysis offers promising avenues for capturing temporal dependencies and linguistic nuances in medical data. By integrating recurrent neural network architecture with natural language processing capabilities, this approach enables comprehensive analysis of both structured and unstructured patient data, facilitating early detection, risk assessment, and personalized intervention strategies for heart disease management.

THANK YOU!!!

Source code:-

```
"""@author: Sreenivas Bhattiprolu

This code makes sense when you watch the accompanying video:

 https://youtu.be/j2kfzYR_abI

#Dataset link:

https://cdn.scribbr.com/wp-
content/uploads//2020/02/heart.data_.zip?_ga=2.217642335.893016210.159
8387608-409916526.1598387608


#Heart disease

The effect that the independent variables biking and smoking

have on the dependent variable heart disease “""


from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Activation

import numpy as np

import pandas as pd

import seaborn as sns

import numpy as np

from matplotlib import pyplot as plt

df = pd.read_csv('data/heart_data.csv')

print(df.head())

df = df.drop("Unnamed: 0", axis=1)

#A few plots in Seaborn to understand the data

sns.lmplot(x='biking', y='heart.disease', data=df)

sns.lmplot(x='smoking', y='heart.disease', data=df)
```

```
x_df = df.drop('heart.disease', axis=1)

y_df = df['heart.disease']

x = x_df.to_numpy()

y = y_df.to_numpy()


from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state=42)


# Build the network

# sgd = optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)


model = Sequential()

model.add(Dense(2, input_dim=2, activation='relu'))

model.add(Dense(1))

model.compile(loss='mean_squared_error', optimizer='adam')

print(model.summary())

history = model.fit(X_train, y_train ,verbose=1, epochs=1000,

          validation_data=(X_test, y_test))


# Predict


prediction_test = model.predict(X_test)

print(y_test, prediction_test)

print("Mean sq. errror between y_test and predicted =",
np.mean(prediction_test-y_test)**2)
```

```
#plot the training and validation accuracy and loss at each epoch
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'y', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()


# Print weights
for layer_depth, layer in enumerate(model.layers):
    weights = layer.get_weights()[0]
    biases = layer.get_weights()[1]
    print('____________ Layer', layer_depth, '___________')
    for toNeuronNum, bias in enumerate(biases):
        print(f'Bias to Layer{layer_depth+1}Neuron{toNeuronNum}: {bias}')


    for fromNeuronNum, wgt in enumerate(weights):
        for toNeuronNum, wgt2 in enumerate(wgt):
            print(f'Layer{layer_depth}, Neuron{fromNeuronNum} to
Layer{layer_depth+1}, Neuron{toNeuronNum} = {wgt2}')
```

```
“””As the weights change for each run let us use the following weights for our calculations

___________ Layer 0 __________

Bias to Layer1Neuron0: 4.4128947257995605

Bias to Layer1Neuron1: 4.5146260261535645

Layer0, Neuron0 to Layer1, Neuron0 = -0.08574138581752777

Layer0, Neuron0 to Layer1, Neuron1 = -0.059531815350055695

Layer0, Neuron1 to Layer1, Neuron0 = 0.1630137860774994

Layer0, Neuron1 to Layer1, Neuron1 = -0.015843335539102554

___________ Layer 1 __________

Bias to Layer2Neuron0: 2.504946708679199

Layer1, Neuron0 to Layer2, Neuron0 = 1.4296010732650757

Layer1, Neuron1 to Layer2, Neuron0 = 1.1467727422714233
"""


x0 = 65.1292

x1 = 2.21956

hidden0 = max(0, ((x0*-0.08574138)+(x1*0.163013786)+(4.4128947)))

hidden1 = max(0, ((x0*-0.0595318)+(x1*-0.0158433)+(4.514626)))

output = max(0, ((hidden0*1.4296)+(hidden1*1.14677)+(2.504947)))

print(output)
```