**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=2.217642335.893016210.1598387608-409916526.1598387608

#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)