Name: Tarun Tanmay

Class: MBATech CE

Sem: 6

Roll No: N049

Experiment 2

Importing important libraries

#importing all the required libraries:

import numpy as np #numpy for performing mathematical calculations

import tensorflow as tf #numerical computations of machine learning and deep learning algorithms

import pandas as pd #pandas for reading and exploring the dataset

Exploring the Dataset

df=pd.read\_csv('diabetes.csv') #reading the diabetes dataset using pandas

df.head() #displaying the content of the dataset

|  | **Glucose** | **BMI** | **Outcome** |
| --- | --- | --- | --- |
| **0** | 148 | 33.6 | 1 |
| **1** | 85 | 26.6 | 0 |
| **2** | 183 | 23.3 | 1 |
| **3** | 89 | 28.1 | 0 |
| **4** | 137 | 43.1 | 1 |

#Feature Scaling

from sklearn.preprocessing import StandardScaler

scaler=StandardScaler()#initialising the StandardScaler function with a variable named scaler

feature=df.iloc[:,0:2] #taking all rows; and columns from 0 to 2

scaler.fit(feature) #fitting the model

feature=scaler.transform(feature) #transforming the model

df.iloc[:,0:2]=feature #storing transformed values back into the dataset

df.head() #displaying the transformed dataset

| **Glucose** | **BMI** | **Outcome** | |
| --- | --- | --- | --- |
| **0** | 0.848324 | 0.204013 | 1 |
| **1** | -1.123396 | -0.684422 | 0 |
| **2** | 1.943724 | -1.103255 | 1 |
| **3** | -0.998208 | -0.494043 | 0 |
| **4** | 0.504055 | 1.409746 | 1 |

#Splitting the dataset into train and test

from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test = train\_test\_split(df.iloc[:,0:2], df.Outcome, test\_size=0.2, random\_state=4)

#print(x\_train)

#614 rows and 2 columns are there after splitting the model for training and testing

#Building a Neural Network Model for Keras

from keras import models as mdls #for grouping input and output layers into object

from keras import layers as lyrs #building block of keras model

model=mdls.Sequential([lyrs.Dense(1, input\_shape=(2,), activation='sigmoid', kernel\_initializer='ones', bias\_initializer='zeros')])

model=mdls.Sequential([lyrs.Dense(1,activation='sigmoid', kernel\_initializer='ones', bias\_initializer='zeros' )])

#Compiling the Model

model.compile(optimizer='SGD', loss='MeanSquaredError', metrics=['accuracy']) #using SGD algorithm as optimizer

#fitting the model

model.fit(x\_train,y\_train, epochs=10)

Epoch 1/10 WARNING:tensorflow:Layer dense\_2 is casting an input tensor from dtype float64 to the layer's dtype of float32, which is new behavior in TensorFlow 2. The layer has dtype float32 because its dtype defaults to floatx. If you intended to run this layer in float32, you can safely ignore this warning. If in doubt, this warning is likely only an issue if you are porting a TensorFlow 1.X model to TensorFlow 2. To change all layers to have dtype float64 by default, call `tf.keras.backend.set\_floatx('float64')`. To change just this layer, pass dtype='float64' to the layer constructor. If you are the author of this layer, you can disable autocasting by passing autocast=False to the base Layer constructor. 20/20 [==============================] - 0s 997us/step - loss: 0.1957 - accuracy: 0.6954 Epoch 2/10 20/20 [==============================] - 0s 910us/step - loss: 0.1951 - accuracy: 0.6938 Epoch 3/10 20/20 [==============================] - 0s 837us/step - loss: 0.1944 - accuracy: 0.6938 Epoch 4/10 20/20 [==============================] - 0s 1ms/step - loss: 0.1938 - accuracy: 0.6954 Epoch 5/10 20/20 [==============================] - 0s 847us/step - loss: 0.1932 - accuracy: 0.6954 Epoch 6/10 20/20 [==============================] - 0s 908us/step - loss: 0.1926 - accuracy: 0.6987 Epoch 7/10 20/20 [==============================] - 0s 993us/step - loss: 0.1921 - accuracy: 0.7003 Epoch 8/10 20/20 [==============================] - 0s 967us/step - loss: 0.1916 - accuracy: 0.7036 Epoch 9/10 20/20 [==============================] - 0s 913us/step - loss: 0.1911 - accuracy: 0.7036 Epoch 10/10 20/20 [==============================] - 0s 1ms/step - loss: 0.1905 - accuracy: 0.7052

<tensorflow.python.keras.callbacks.History at 0x7f54b5010f98>

#Evaluate the Model

model.evaluate(x\_test, y\_test)

5/5 [==============================] - 0s 2ms/step - loss: 0.1692 - accuracy: 0.7532

[0.16924573481082916, 0.7532467246055603]

model.summary()

Model: "sequential\_2" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Layer (type) Output Shape Param # ================================================================= dense\_2 (Dense) (None, 1) 3 ================================================================= Total params: 3 Trainable params: 3 Non-trainable params: 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

yhat= model.predict(x\_test)

y\_test\_float=y\_test.astype('float32')

yhat=np.insert(yhat, -1, y\_test\_float, axis=1)

print(yhat)

[[0. 0.3181274 ] [0. 0.1163142 ] [0. 0.28703865] [0. 0.8136753 ] [1. 0.72302276] [1. 0.81193817] [1. 0.8387553 ] [1. 0.1741294 ] [1. 0.8868479 ] [0. 0.18671277] [0. 0.6979434 ] [0. 0.22978547] [1. 0.39758408] [0. 0.19335097] [0. 0.2381855 ] [1. 0.7429305 ] [1. 0.90767264] [1. 0.9762547 ] [0. 0.07567739] [0. 0.17731097] [0. 0.2685849 ] [0. 0.78329766] [0. 0.23477411] [1. 0.9568111 ] [1. 0.22308457] [0. 0.80283356] [0. 0.21792084] [1. 0.5725291 ] [1. 0.5427946 ] [1. 0.3025397 ] [1. 0.88069034] [0. 0.16214323] [0. 0.4321878 ] [0. 0.10215646] [0. 0.87814164] [0. 0.32353935] [0. 0.7222866 ] [0. 0.6725074 ] [0. 0.10231572] [1. 0.80207133] [1. 0.7222866 ] [0. 0.6006443 ] [0. 0.4244719 ] [1. 0.9252434 ] [0. 0.29162377] [0. 0.12910476] [0. 0.37851402] [1. 0.925658 ] [0. 0.44504347] [0. 0.15592006] [1. 0.7539655 ] [0. 0.6609061 ] [0. 0.21622992] [0. 0.7931596 ] [1. 0.6213496 ] [0. 0.11910832] [0. 0.1837002 ] [0. 0.11218619] [0. 0.15256691] [1. 0.8591645 ] [0. 0.31943223] [0. 0.09293497] [1. 0.6129888 ] [0. 0.03234825] [0. 0.49953523] [0. 0.19384089] [0. 0.13081098] [0. 0.6026286 ] [0. 0.5560167 ] [1. 0.94549704] [0. 0.22862941] [1. 0.57589155] [0. 0.36623436] [0. 0.12830362] [0. 0.24775106] [1. 0.9114561 ] [0. 0.49532893] [1. 0.45254162] [0. 0.26019698] [0. 0.81666255] [1. 0.8649124 ] [0. 0.5408057 ] [0. 0.26993796] [1. 0.9816883 ] [0. 0.7518294 ] [0. 0.10419497] [0. 0.08672506] [1. 0.8269937 ] [0. 0.57483155] [1. 0.37036598] [0. 0.702805 ] [0. 0.16249722] [0. 0.4391974 ] [0. 0.3291905 ] [0. 0.1082063 ] [0. 0.45850274] [0. 0.3789223 ] [1. 0.89049387] [1. 0.7639555 ] [0. 0.4993346 ] [0. 0.32724708] [0. 0.7035294 ] [1. 0.39389598] [1. 0.5168211 ] [1. 0.9267124 ] [0. 0.57559836] [0. 0.1566332 ] [1. 0.85354733] [1. 0.6047529 ] [0. 0.59671456] [0. 0.8821835 ] [1. 0.9065418 ] [1. 0.80966437] [0. 0.13202259] [0. 0.28932583] [0. 0.29772964] [0. 0.5541041 ] [0. 0.74733007] [1. 0.6739613 ] [1. 0.75543666] [0. 0.47222283] [0. 0.21719342] [0. 0.5618598 ] [0. 0.21209499] [0. 0.58907574] [0. 0.33451962] [0. 0.31603092] [0. 0.1332671 ] [0. 0.3416163 ] [1. 0.95329314] [1. 0.6232326 ] [0. 0.11940983] [1. 0.34257805] [0. 0.3436765 ] [1. 0.72623307] [0. 0.14254305] [0. 0.49053955] [0. 0.32310116] [1. 0.7365031 ] [0. 0.94741195] [0. 0.20853561] [0. 0.13593444] [0. 0.19456166] [0. 0.59892946] [1. 0.83941334] [1. 0.79666424] [1. 0.96063167] [0. 0.60945106] [0. 0.3837892 ] [0. 0.61094385] [0. 0.09404969] [0. 0.58256346] [1. 0.7309202 ] [1. 0.81682277]]