TRIBHUVAN UNIVERSITY



**Sagarmatha College of Science &**

**Technology**

Lab Report On: Neural Network

Lab Report No.: 05

Date: 2077-11-21

**SUBMITTED BY SUBMITTED TO**

***Name: Ramesh Neupane CSIT Department***

***Roll no.: 37***

**Qeustion 01**

Write the Python program to predict the diabetes using MLP. Use the given dataset.

**Source Code**

# import necessary libraries/modules

import numpy as np

import pandas as pd

from sklearn.preprocessing import StandardScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from sklearn import metrics

# import dataset

diabetes = pd.read\_csv("Diabetes.csv")

# split dataset into input and target

inputs = diabetes.iloc[0:, 0:8]

target = diabetes.iloc[0:, 8:9]

# construct the NN model

model = Sequential()

model.add(Dense(32, input\_dim = 8, activation = 'relu')) # first hidden layer

model.add(Dense(16, activation = 'relu')) # second hidden layer

model.add(Dense(8, activation = 'relu')) # third hidden layer

model.add(Dense(1, activation = 'sigmoid')) # output layer

model.compile(loss = 'binary\_crossentropy', optimizer = 'adam', metrics = ['accuracy'])

# train the model

model.fit(inputs, target, epochs = 1000, batch\_size = 10, verbose = 0)

# make the prediction with the trained model

predictions = model.predict(inputs)

pred\_round = []

for e in predictions:

pred\_round.append(np.round(e))

# display actual and predicted value

print("Actual output: ", \*np.array(target))

print("\nRounded predicted output: ", \*pred\_round)

# accuracy, recall, precision and f1-score

print(f"Accuracy: {metrics.accuracy\_score(target, pred\_round)}")

print(f"Recall: {metrics.recall\_score(target, pred\_round)}")

print(f"Precision: {metrics.precision\_score(target, pred\_round)}")

print(f"F1-score: {metrics.f1\_score(target, pred\_round)}")

**Output**

Actual output: [0] [1] [0] [1] [0] [1] [0] [1] [1] [0] [1] [0] [1] [1] [1] [1] [1] [0] [1] [0] [0] [1] [1] [1] [1] [1] [0] [0] [0] [0] [1] [0] [0] [0] [0] [0] [1] [1] [1] [0] [0] [0] [1] [0] [1] [0] [0] [1] [0] [0] [0] [0] [1] [0] [0] [1] [0] [0] [0] [0] [1] [0] [0] [1] [0] [1] [0] [0] [0] [1] [0] [1] [0] [0] [0] [0] [0] [1] [0] [0] [0] [0] [0] [1] [0] [0] [0] [1] [0] [0] [0] [0] [1] [0] [0] [0] [0] [0] [1] [1] [0] [0] [0] [0] [0] [0] [0] [0] [1] [1] [1] ……………

Rounded predicted output: [0.] [1.] [0.] [1.] [0.] [1.] [0.] [1.] [1.] [0.] [1.] [0.] [1.] [1.] [1.] [1.] [1.] [0.] [1.] [0.] [0.] [1.] [0.] [1.] [1.] [1.] [0.] [0.] [0.] [0.] [1.] [0.] [0.] [0.] [0.] [1.] [1.] [1.] [1.] [0.] [1.] [0.] [1.] [0.] [1.] [0.] [0.] [1.] [0.] [0.] [0.] [0.] [1.] [0.] [0.] [1.] [0.] [0.] [0.] [0.] [1.] [0.] [0.] [0.] [0.] [1.] [0.] [0.] [0.] [0.] [0.] [1.] [0.] [0.] [0.] [0.] [0.] [1.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] ……………………………

Accuracy: 0.9269882659713168

Recall: 0.8202247191011236

Precision: 0.9647577092511013

F1-score: 0.8866396761133603

**Question 02**

Write the Python program to predict housing price using MLP. Use the given dataset.

**Source Code**

# import necessary libraries/modules

import numpy as np

import pandas as pd

from sklearn.preprocessing import StandardScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from sklearn import metrics

# import dataset

bostonHouse = pd.read\_csv("Housing.csv", delim\_whitespace=True)

# split dataset into input and target

inputs = bostonHouse.iloc[:, 0:13]

target = bostonHouse.iloc[:, 13:14]

# training dataset

x\_train = inputs.iloc[0:400]

y\_train = target.iloc[0:400]

# testing dataset

x\_test = inputs.iloc[400:]

y\_test = target.iloc[400:]

# construct the NN model

model = Sequential()

model.add(Dense(13, input\_dim = 13, kernel\_initializer = 'normal', activation = 'relu'))

model.add(Dense(6, kernel\_initializer = 'normal', activation = 'relu'))

model.add(Dense(1, kernel\_initializer = 'normal'))

model.compile(loss = 'mean\_squared\_error', optimizer = 'adam',

metrics = ['mean\_absolute\_percentage\_error'])

# train the model

model.fit(x\_train, y\_train, epochs = 30, batch\_size = 32, verbose = 0)

# predict using trained model

predictions = model.predict(x\_test)

y\_test = np.array(y\_test)

print(f"MSE: {metrics.mean\_squared\_error(y\_test, predictions)}")

# visualization of actual value and predicted value

y\_test = np.array(y\_test)

plt.figure(figsize = (10, 5))

plt.plot(y\_test, color = "green", linewidth = 2.0)

plt.plot(predictions, color = "blue", linewidth = 2.0)

plt.title("Boston Housing Pricing")

plt.xlabel("house")

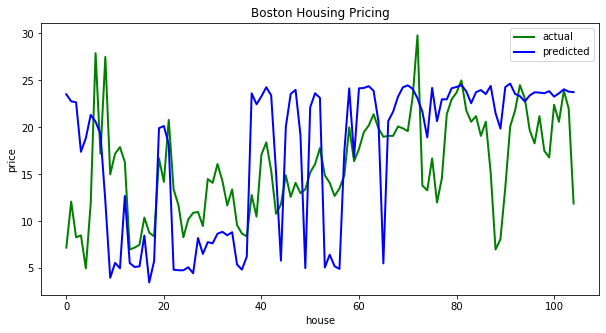
plt.ylabel("price")

plt.legend(["actual", "predicted"], loc = "upper left")

plt.show()

**Ouptut**

MSE: 48.30543111407888



**Conclusion**

**Hence, we are able to implement MLP (Multi-Layer Perceptron) for prediction.**