**Name :- Shivam Indrabhan Borse**

**Roll No :- 19**

**Subject: Artificial Neural Network (SL - II)**

**Class : TE**

**Branch: AI & DS**

**Practical – 5**

**Problem statement :** Implement an Artificial Neural Network training process in Python by using Forward Propagation, and Back Propagation.

**Code :**

import numpy as np

def sigmoid(x):

    return 1 / (1 + np.exp(-x))

def sigmoid\_derivative(x):

    return x \* (1 - x)

class NeuralNetwork:

def \_\_init\_\_(self, x, y):

self.input = x

self.output = y

self.hidden\_size = 4

self.weights1 = np.random.randn(self.input.shape[1], self.hidden\_size)

self.weights2 = np.random.randn(self.hidden\_size, 1)

def feedforward(self):

self.hidden = sigmoid(np.dot(self.input, self.weights1))

self.predicted\_output = sigmoid(np.dot(self.hidden, self.weights2))

def backpropagate(self):

output\_error = self.output - self.predicted\_output

d\_predicted\_output = output\_error \* sigmoid\_derivative(self.predicted\_output)

hidden\_error = d\_predicted\_output.dot(self.weights2.T)

d\_hidden = hidden\_error \* sigmoid\_derivative(self.hidden)

self.weights1 += self.input.T.dot(d\_hidden)

self.weights2 += self.hidden.T.dot(d\_predicted\_output)

def train(self, epochs):

for epoch in range(epochs):

self.feedforward()

self.backpropagate()

def predict(self, x):

hidden = sigmoid(np.dot(x, self.weights1))

predicted\_output = sigmoid(np.dot(hidden, self.weights2))

return predicted\_output

X = np.array([[0, 0, 1], [0, 1, 1], [1, 0, 1], [1, 1, 1]])

y = np.array([[0], [1], [1], [0]])

nn = NeuralNetwork(X, y)

nn.train(10000)

x\_test = np.array([[0, 0, 0], [1, 0, 0]])

for x in x\_test:

print("Input:", x)

print("Output:", nn.predict(x))

**OUTPUT :**

