**Assignment No-7**

**Name-** Hule Samiksha Anil

**Roll No-**24

**Title:** Python program to show back propagationnetwork for XOR function with Binary Input and Output.

**Program:**

import numpy as np

class XORNetwork:

def \_\_init\_\_(self):

self.W1 = np.random.randn(2, 2)

self.b1 = np.random.randn(2)

self.W2 = np.random.randn(2, 1)

self.b2 = np.random.randn(1)

def sigmoid(self, x):

return 1 / (1 + np.exp(-x))

def sigmoid\_derivative(self, x):

return x \* (1 - x)

def forward(self, X):

self.z1 = np.dot(X, self.W1) + self.b1

self.a1 = self.sigmoid(self.z1)

self.z2 = np.dot(self.a1, self.W2) + self.b2

self.a2 = self.sigmoid(self.z2)

return self.a2

def backward(self, X, y, output):

self.output\_error = y - output

self.output\_delta = self.output\_error \* self.sigmoid\_derivative(output)

self.z1\_error = self.output\_delta.dot(self.W2.T)

self.z1\_delta = self.z1\_error \* self.sigmoid\_derivative(self.a1)

self.W1 += X.T.dot(self.z1\_delta)

self.b1 += np.sum(self.z1\_delta, axis=0)

self.W2 += self.a1.T.dot(self.output\_delta)

self.b2 += np.sum(self.output\_delta, axis=0)

def train(self, X, y, epochs):

for \_ in range(epochs):

output = self.forward(X)

self.backward(X, y, output)

def predict(self, X):

return self.forward(X)

xor\_nn = XORNetwork()

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

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

xor\_nn.train(X, y, epochs=10000)

predictions = xor\_nn.predict(X)

print(predictions)

**Output:**

[[0.01075848]

[0.98777524]

[0.9877705 ]

[0.01148649]]