Name – Akshat Bhargava

Roll no 22563

ANN – SL 2

Prac 6

Code –

import numpy as np

# Step 1: Define the sigmoid function and its derivative

def sigmoid(x):

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

def sigmoid\_derivative(x):

return x \* (1 - x)

# Step 2: Define the training function for the neural network

def train\_neural\_network(X, y, learning\_rate, epochs):

# Step 3: Initialize the weights and biases with random values

input\_neurons = X.shape[1]

hidden\_neurons = 4

output\_neurons = y.shape[1]

hidden\_weights = np.random.uniform(size=(input\_neurons, hidden\_neurons))

hidden\_bias = np.random.uniform(size=(1, hidden\_neurons))

output\_weights = np.random.uniform(size=(hidden\_neurons, output\_neurons))

output\_bias = np.random.uniform(size=(1, output\_neurons))

# Step 4: Perform the training iterations

for i in range(epochs):

# Step 4.1: Forward propagation

hidden\_layer\_activation = np.dot(X, hidden\_weights) + hidden\_bias

hidden\_layer\_output = sigmoid(hidden\_layer\_activation)

output\_layer\_activation = np.dot(hidden\_layer\_output, output\_weights) + output\_bias

predicted\_output = sigmoid(output\_layer\_activation)

# Step 4.2: Backward propagation

error = y - predicted\_output

d\_predicted\_output = error \* sigmoid\_derivative(predicted\_output)

error\_hidden\_layer = d\_predicted\_output.dot(output\_weights.T)

d\_hidden\_layer = error\_hidden\_layer \* sigmoid\_derivative(hidden\_layer\_output)

# Step 4.3: Update the weights and biases

output\_weights += hidden\_layer\_output.T.dot(d\_predicted\_output) \* learning\_rate

output\_bias += np.sum(d\_predicted\_output, axis=0, keepdims=True) \* learning\_rate

hidden\_weights += X.T.dot(d\_hidden\_layer) \* learning\_rate

hidden\_bias += np.sum(d\_hidden\_layer, axis=0, keepdims=True) \* learning\_rate

# Step 5: Return the predicted output

return predicted\_output

# Example usage

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

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

# Train the Neural Network

predicted\_output = train\_neural\_network(X, y, learning\_rate=0.1, epochs=10000)

print(predicted\_output)

Output –

