**Practical 2**

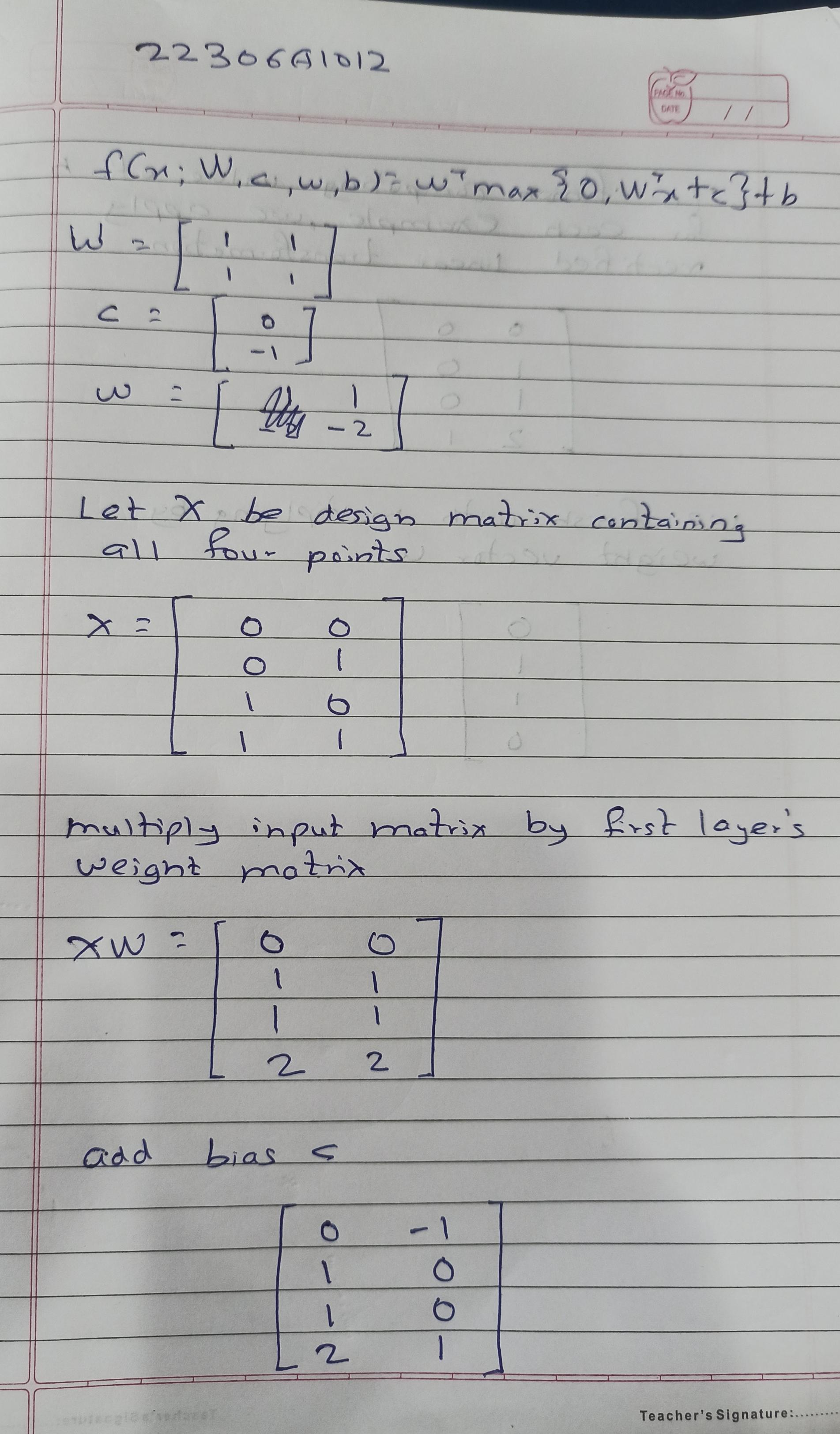
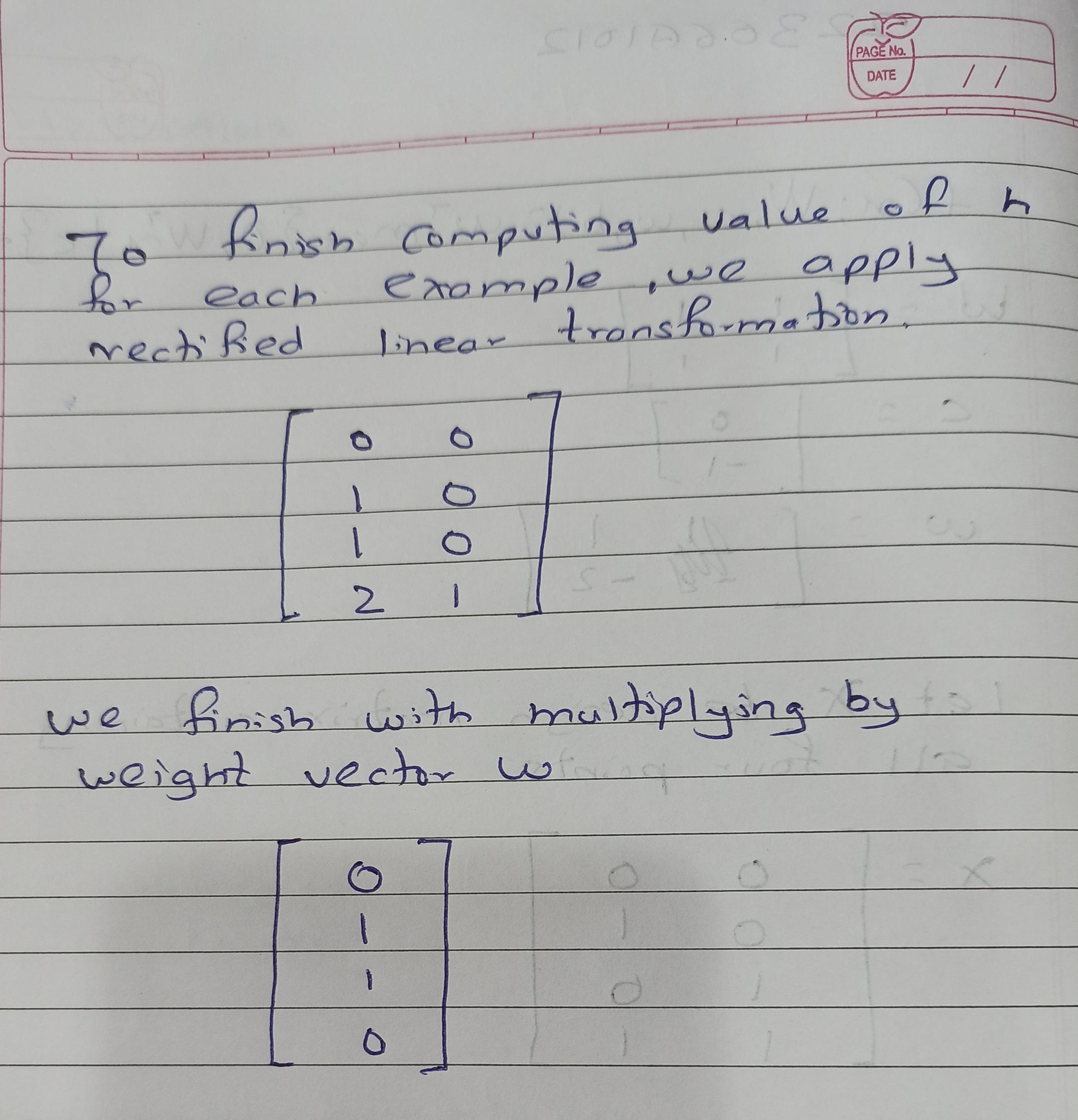
**Solving XOR problem using deep feed forward network**

**Aim:** Solving XOR problem using deep feed forward network

**Description:**

A deep feedforward network, also known as a feedforward neural network or multilayer perceptron (MLP), is a foundational architecture in deep learning. It consists of multiple layers of interconnected nodes, with each layer feeding its output forward as input to the next layer. These networks are characterized by their ability to learn complex representations of data through hierarchical feature extraction. They are widely used for supervised learning tasks such as classification and regression, where the input-output mapping is learned from labeled data. Training is typically done using techniques like backpropagation and stochastic gradient descent to minimize a specified loss function.

**Handwritten:**

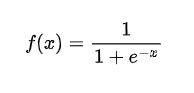


**Sigmoid Function:**

The sigmoid function is a popular activation function used in neural networks, especially in the output layer for binary classification tasks. It squashes the output of a neuron to a value between 0 and 1, which can be interpreted as a probability.

Mathematically, the sigmoid function is defined as:

\[ f(x) = \frac{1}{1 + e^{-x}} \]



Here's what the sigmoid function does:

- For large positive values of \( x \), \( f(x) \) approaches 1.

- For large negative values of \( x \), \( f(x) \) approaches 0.

- For \( x = 0 \), \( f(x) \) is exactly 0.5.

The sigmoid function introduces non-linearity to the network, letting it learn from the error and adjust the weights during training. However, it's worth noting that sigmoid can suffer from the vanishing gradient problem, especially in deeper networks, which is why other activation functions like ReLU are often preferred for hidden layers.

**Code:**

import numpy as np

from keras.models import Sequential

from keras.layers import Dense

model = Sequential()

model.add(Dense(units = 2, activation = 'relu', input\_dim = 2))

model.add(Dense(units=1, activation = 'sigmoid'))

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

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

print("Input data:")

print(X)

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

print("\nTarget labels:")

print(y)

model.get\_weights()

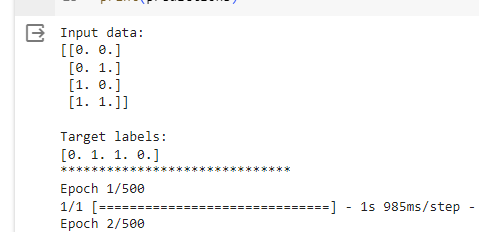
model.fit(X,y,epochs = 500)

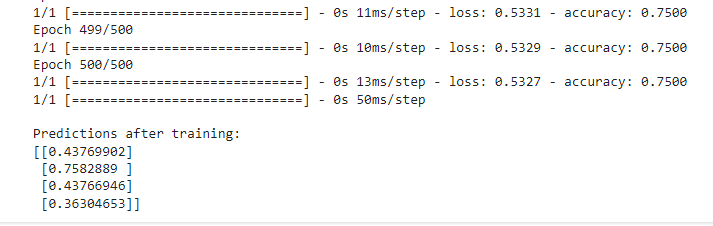
predictions = model.predict(X)

print("\nPredictions after training:")

print(predictions)

**Output:**





**Learning:**

This code defines a neural network model using Keras to perform binary classification on a dataset with four samples, each containing two features. The model is trained for 500 epochs using binary crossentropy loss and the Adam optimizer. After training, it makes predictions on the input data, yielding the output probabilities of belonging to the positive class.