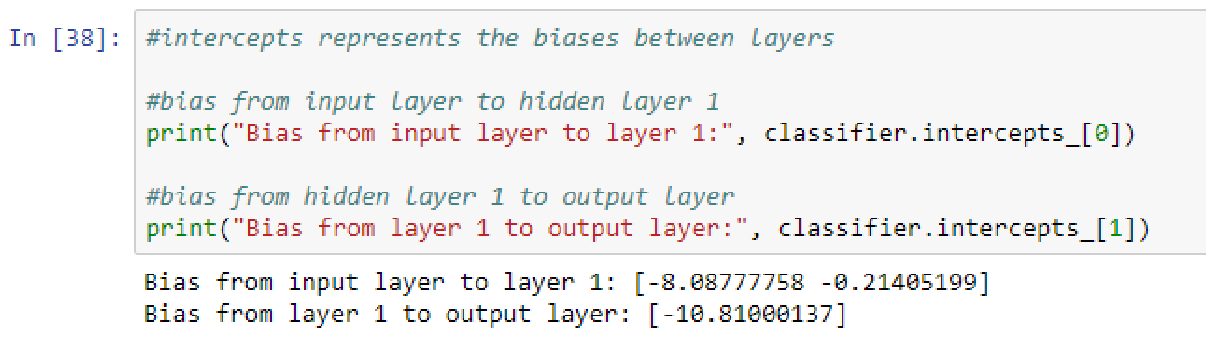
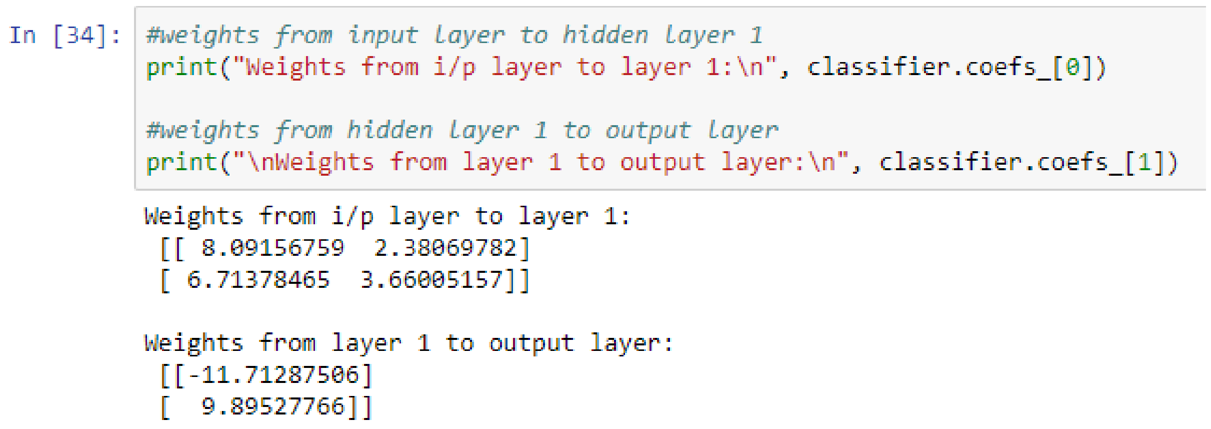
OUTPUT 1

1. XOR using a Neural Network.



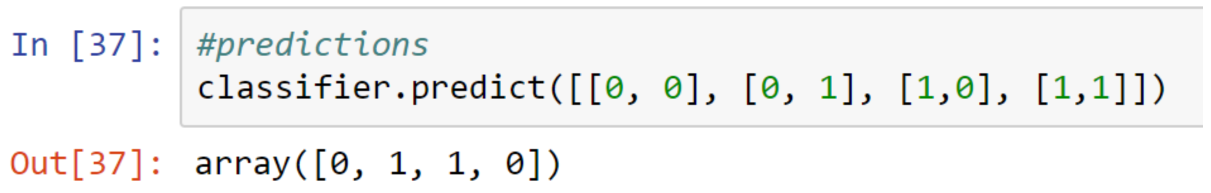
```
In [38]: #intercepts represents the biases between layers

         #bias from input layer to hidden layer 1
         print("Bias from input layer to layer 1:", classifier.intercepts_[0])

         #bias from hidden layer 1 to output layer
         print("Bias from layer 1 to output layer:", classifier.intercepts_[1])

         Bias from input layer to layer 1: [-8.08777758 -0.21405199]
         Bias from layer 1 to output layer: [-10.81000137]
```



```
In [34]: #weights from input layer to hidden layer 1
         print("Weights from i/p layer to layer 1:\n", classifier.coefs_[0])

         #weights from hidden layer 1 to output layer
         print("\nWeights from layer 1 to output layer:\n", classifier.coefs_[1])

         Weights from i/p layer to layer 1:
          [[ 8.09156759  2.38069782]
          [ 6.71378465  3.66005157]]

         Weights from layer 1 to output layer:
          [[-11.71287506]
          [  9.89527766]]
```

This Figure shows the values of biases and weights between the input layer and the hidden layer, hidden layer and the output layer respectively.

The Model after learning the weights, is ready to make predictions. The following figure shows the predictions.

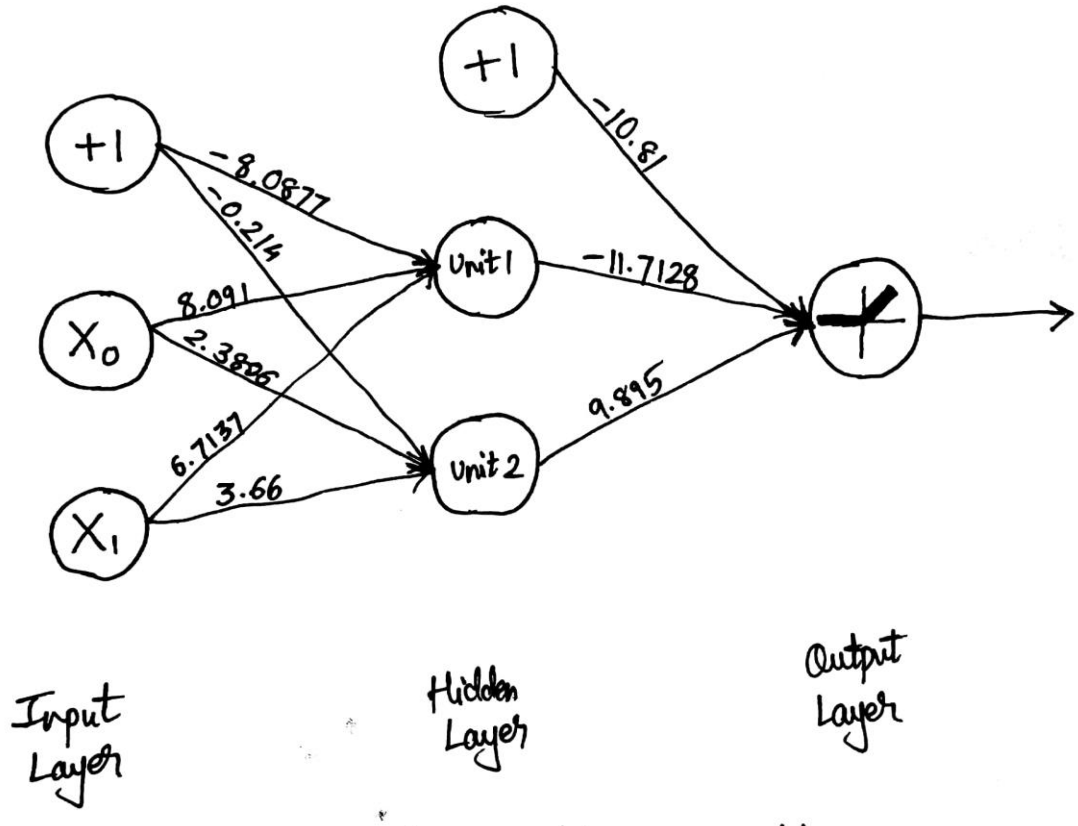


```
In [37]: #predictions
         classifier.predict([[0, 0], [0, 1], [1,0], [1,1]])

Out[37]: array([0, 1, 1, 0])
```

We know that the characteristics of XOR function is as follows:

X0	X1	Y
0	0	0
0	1	1
1	0	1
1	1	0

The predictions by the neural network are correct for XOR function.



Neural Network for computing XOR function

The schematic representation of the Neural Network: Biases and Weights learnt by the MLP Classifier for XOR function.

I have attached the .py file as well as .ipynb file in this folder.