```python
"""Supervised Regression Problem"""
import numpy as np

"""Training Data"""
#Input Data = hours slept and hours studied
X = np.array(([3,5],[5,1], [10,2]), dtype=float)

#Output data = score on test
y = np.array(([75],[82], [93]), dtype=float)

"""Testing Data"""
#(8,3) = ?

"""
Using Artificial Neural Networks
Other ways are as follows:
    1. Support Vector Machines
    2. Gaussion Process
    3. Classification + Regression Trees
    4. Random Forests
    5. Etc.
"""

"""Now need to account for units of data. X is in hours y is in 1 of 100%"""
#Scale data (All data is positive)

y = y/100 # Max test score is 100
#output layer will be called y-hat because it is an estimate of y. Not y.
class Neural_Network(object):
    def __init__(self):
        #Define HyperParameters
        self.inputLayerSize = 2
        self.outputLayerSize = 1
        self.hiddenLayerSize = 3

        #Weights (Parameters)
        self.W1 = np.random.rand(self.inputLayerSize, \
                                    self.hiddenLayerSize)
        self.W2 = np.random.rand(self.hiddenLayerSize, \
                                    self.outputLayerSize)
    def forward(self, X):
        #Propagate inputs through network
        self.z2 = np.dot(X, self.W1)
        self.a2 = self.sigmoid(self.z2)
        self.z3 = np.dot(self.a2, self.W2)
        yHat = self.sigmoid(self.z3)
        return yHat

    def sigmoid(self, z):
        #Apply sigmoid activation function
        return 1/(1+np.exp(-z))

#s
NN = Neural_Network()
yHat = NN.forward(X)
print("\nHours Slept and Studied:\n\n %s" % X)
print("\nPredicted:\n\n %s" % yHat)
print("\nActual Score:\n\n %s" % y)
```