**5. Build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets.**

import numpy as np

import matplotlib as m

X=np.array(([2,9],[1,5],[3,6]),dtype=float)

y=np.array(([92],[86],[89]),dtype=float)

X=X/np.amax(X,axis=0)

y=y/100

def sigmoid(x):

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

def derivatives\_sigmoid(x):

return x\*(1-x)

epoch=7000

lr=0.1

inputlayer\_neurons=2

hiddenlayer\_neurons=3

output\_neurons=1

wh=np.random.uniform(size=(inputlayer\_neurons,hiddenlayer\_neurons))

bh=np.random.uniform(size=(1,hiddenlayer\_neurons))

wout=np.random.uniform(size=(hiddenlayer\_neurons,output\_neurons))

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

for i in range(epoch):

hinp1=np.dot(X,wh)

hinp=hinp1+bh

hlayer\_act=sigmoid(hinp)

outinp1=np.dot(hlayer\_act,wout)

outinp=outinp1+bout

output=sigmoid(outinp)

EO=y-output

outgrad=derivatives\_sigmoid(output)

d\_output=EO\*outgrad

EH=d\_output.dot(wout.T)

hiddengrad=derivatives\_sigmoid(hlayer\_act)

d\_hiddenlayer=EH\*hiddengrad

wout+=hlayer\_act.T.dot(d\_output)\*lr

wh+=X.T.dot(d\_hiddenlayer)\*lr

print("Input:\n"+str(X))

print("Actual Output:\n"+str(y))

print("Predicted Output:\n",output)

**OUTPUT:**

Input:

[[0.66666667 1. ]

[0.33333333 0.55555556]

[1. 0.66666667]]

Actual Output:

[[0.92]

[0.86]

[0.89]]

Predicted Output:

[[0.8942265 ]

[0.87914546]

[0.89735678]]