Name: Tania Rajabally

Roll No: 43

UID: 2017130047

Batch: C

Course Code: CEL71 (AI and Soft Computing Lab)

Date:12/08/2020

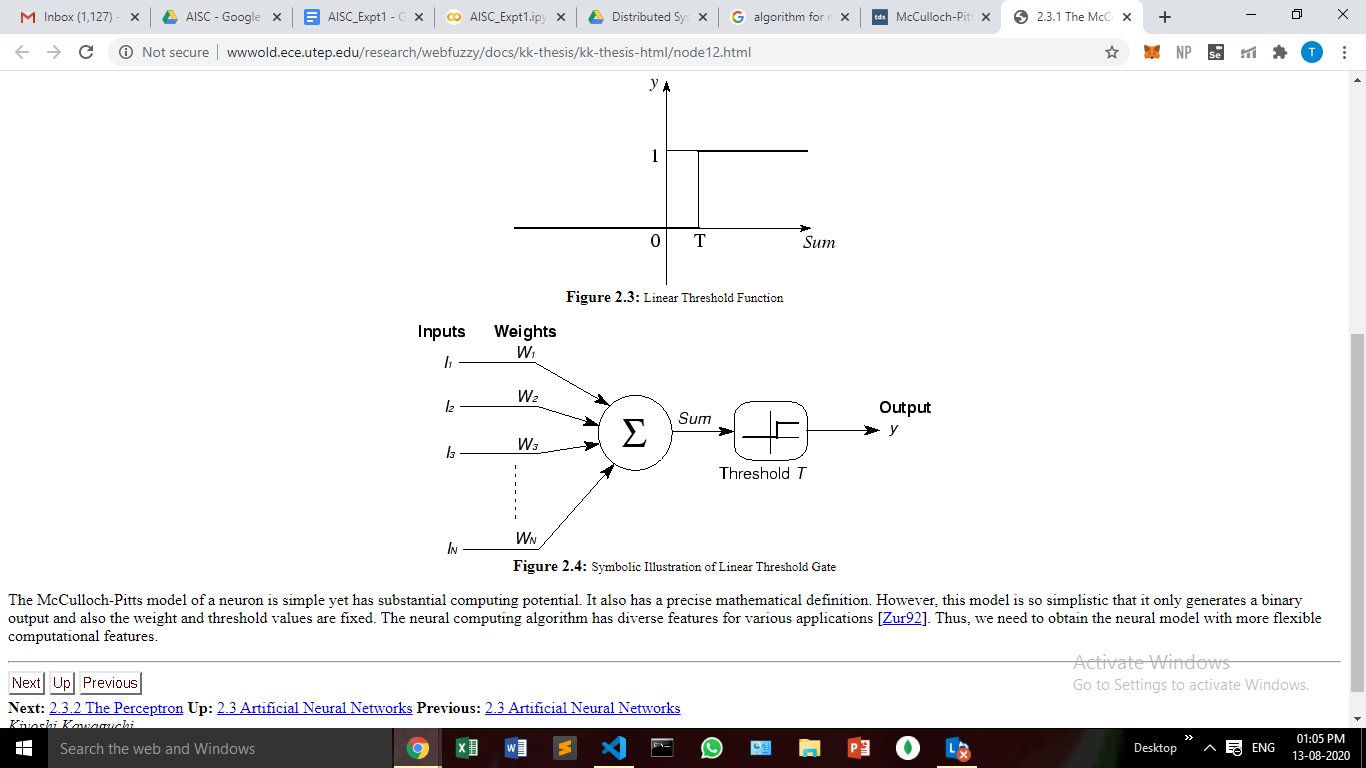
**Experiment No 1**

**Mc-Culloch Pitts Model**

**Aim:** To implement Mc-Culloch Pitts Model

**Theory:**

The very first step towards the perceptron we use today was taken in 1943 by McCulloch and Pitts, by mimicking the functionality of a biological neuron. The McCulloch-Pitts neural model is also known as linear threshold gate. It is a neuron of a set of inputs I1, I2, I3, ..., Im and one output y.The linear threshold gate simply classifies the set of inputs into two different classes. Thus the output y is binary.



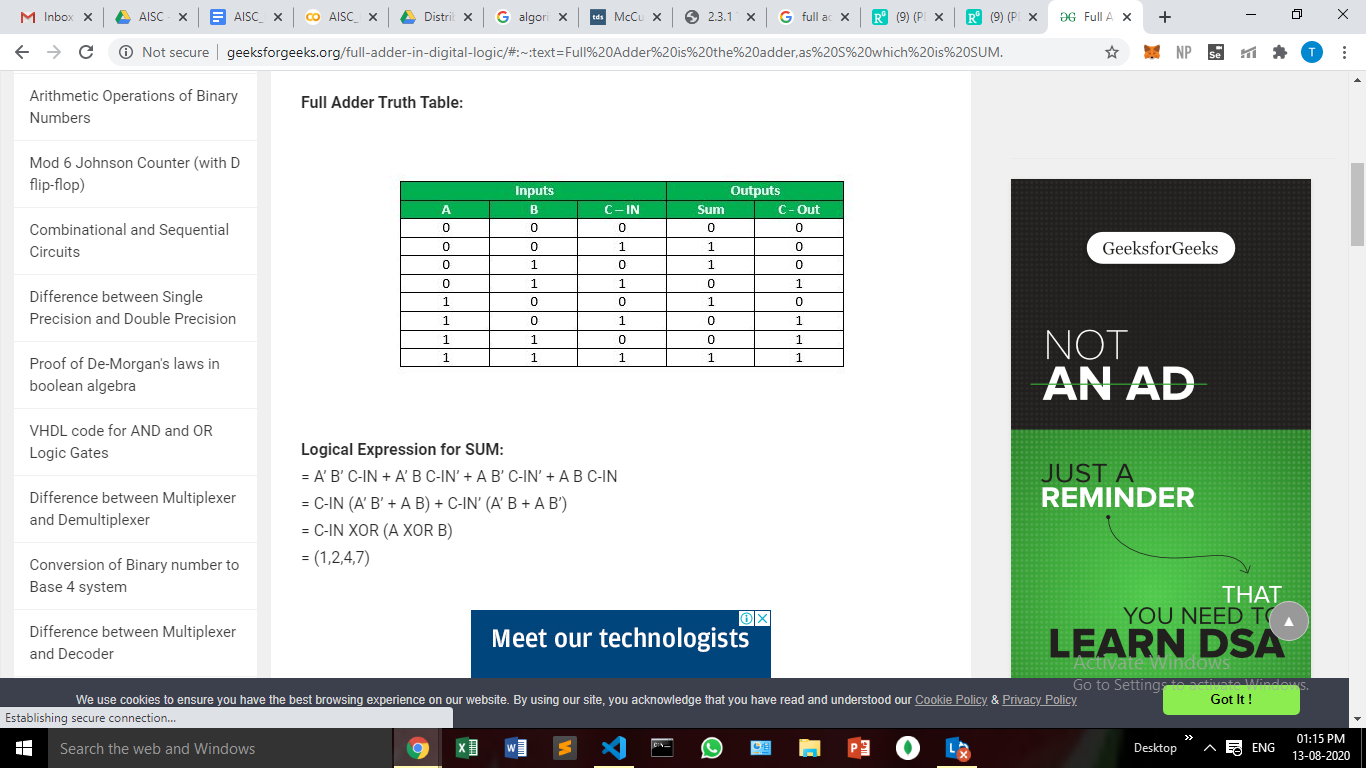
These inputs can either be excitatory or inhibitory. Inhibitory inputs are those that have maximum effect on the decision making irrespective of other inputs i.e., if x\_3 is 1 then my output will always be 0 i.e., the neuron will never fire, so x\_3 is an inhibitory input. Excitatory inputs are NOT the ones that will make the neuron fire on their own but they might fire it when combined together.

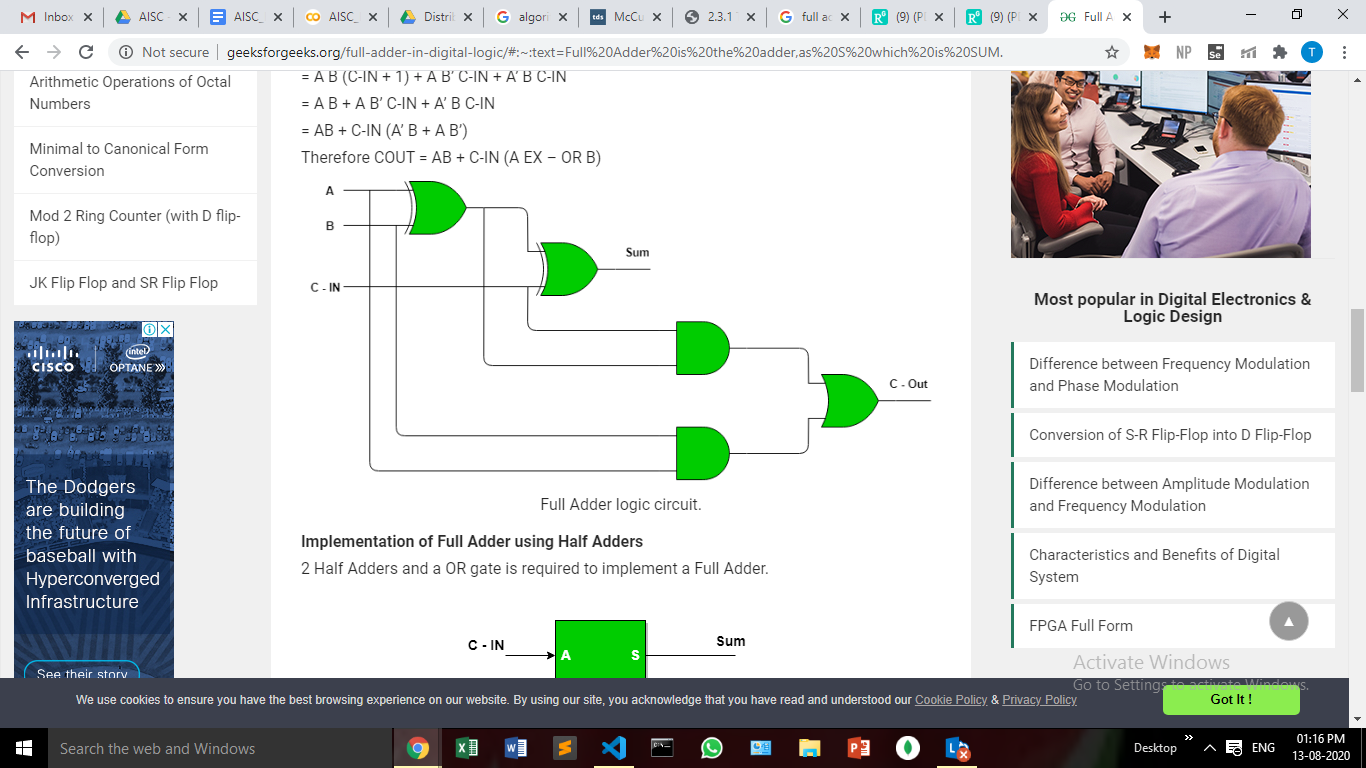
The McCulloch-Pitts model of a neuron is simple yet has substantial computing potential. It also has a precise mathematical definition. However, this model is so simplistic that it only generates a binary output and also the weight and threshold values are fixed. The neural computing algorithm has diverse features for various applications. Thus, we need to obtain the neural model with more flexible computational features.

**Procedure:**

Full Adder is the adder which adds three inputs and produces two outputs. The first two inputs are A and B and the third input is an input carry as C-IN. The output carry is designated as C-OUT and the normal output is designated as S which is SUM.

A full adder logic is designed in such a manner that it can take eight inputs together to create a byte-wide adder and cascade the carry bit from one adder to the other.

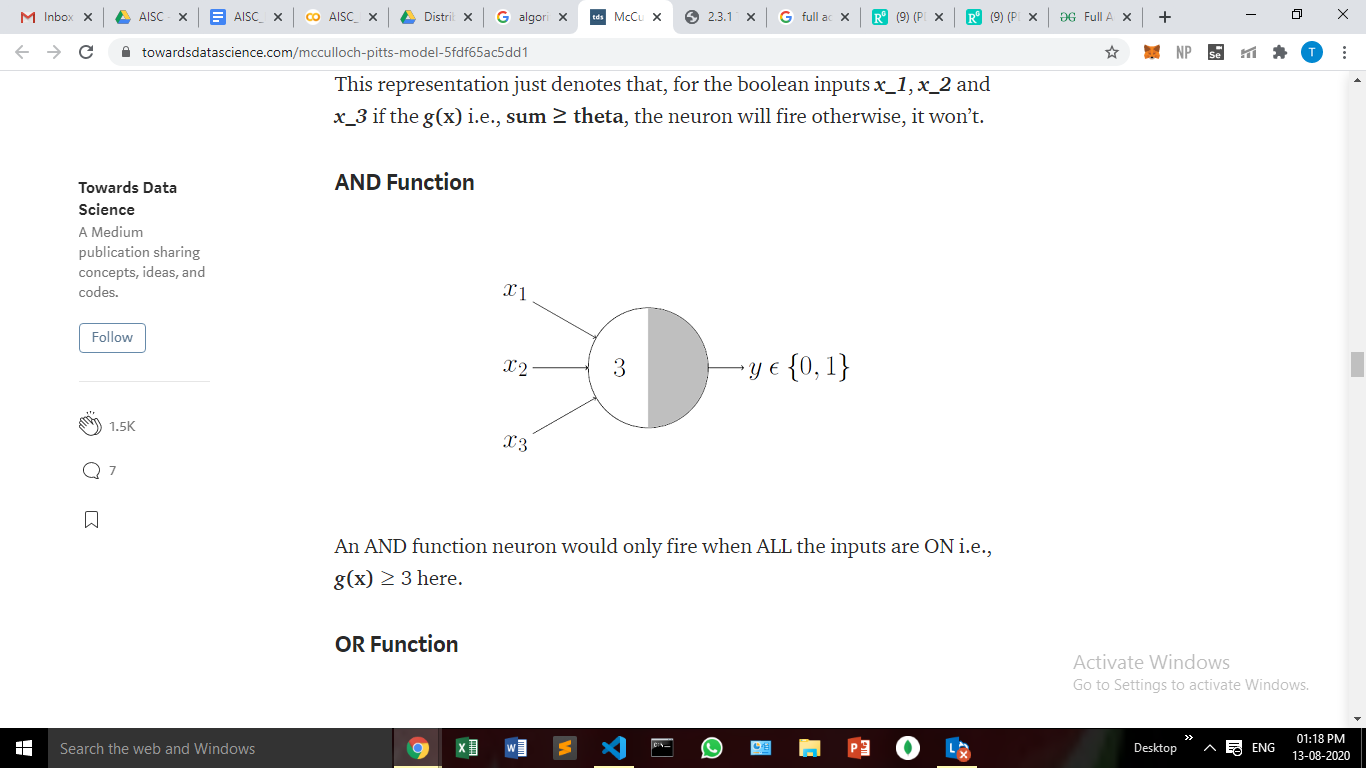




Boolean Functions Using M-P Neuron:

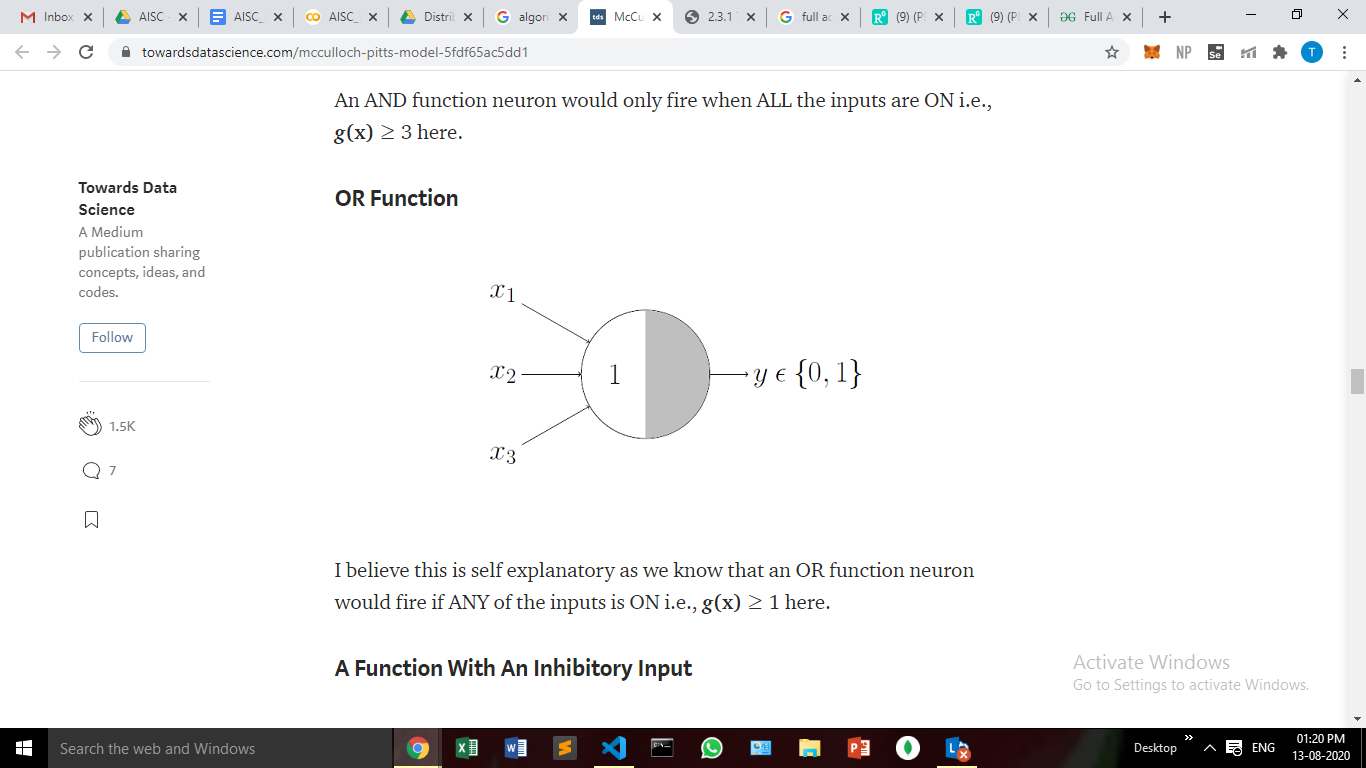
Our inputs are all boolean and the output is also boolean so essentially, the neuron is just trying to learn a boolean function. A lot of boolean decision problems can be cast into this, based on appropriate input variables.

AND Function:



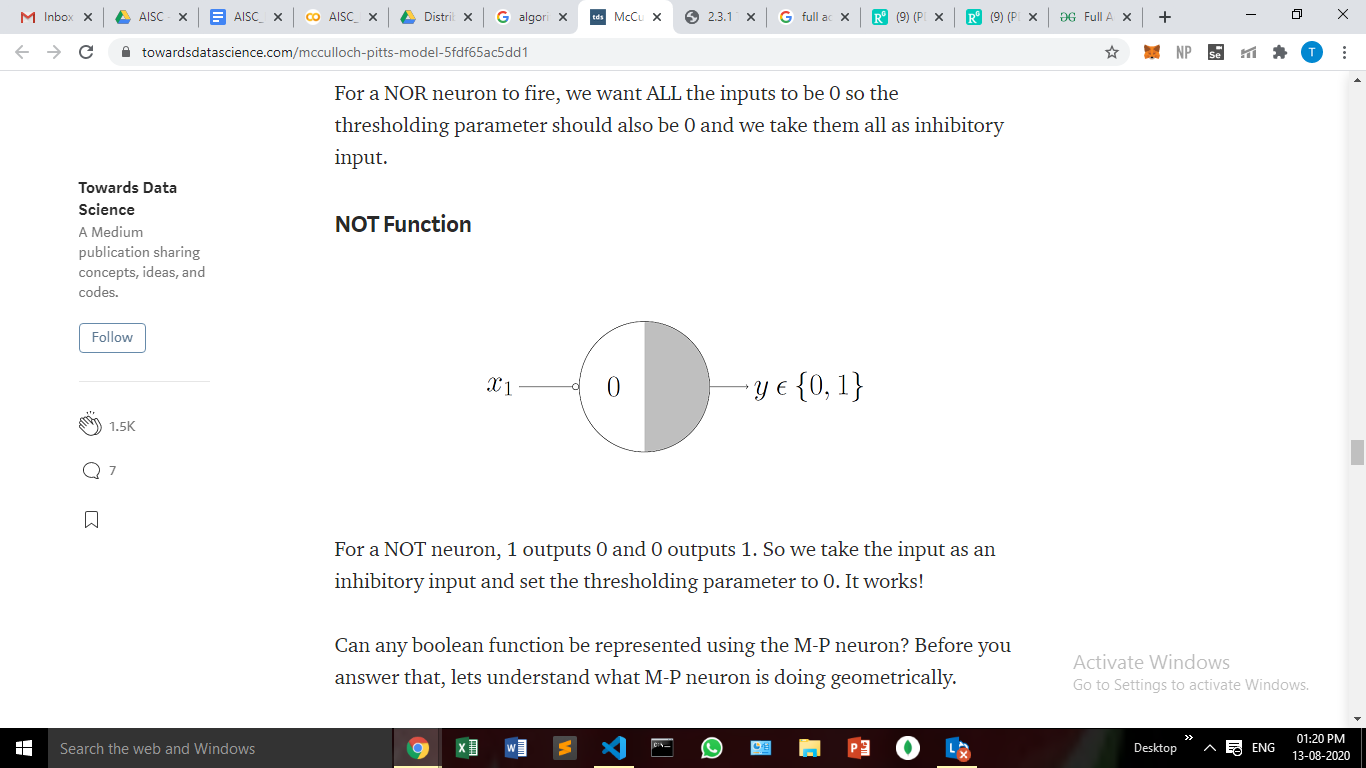
An AND function neuron would only fire when ALL the inputs are ON i.e., g(x) ≥ 3 here.

OR Function:



An OR function neuron would fire if ANY of the inputs is ON i.e., g(x) ≥ 1 here.

NOT Function:



For a NOT neuron, 1 outputs 0 and 0 outputs 1. So we take the input as an inhibitory input and set the thresholding parameter to 0.

**Code:**

#AND OR NOT

def AND(x1,x2):

w1 = 1

w2 = 1

b = -1

if w1\*x1 + w2\*x2 + b > 0:

return 1

else:

return 0

def OR(x1,x2):

w1 = 2

w2 = 2

b = -1

if w1\*x1 + w2\*x2 + b > 0:

return 1

else:

return 0

def NOT(x):

w = -1

b = 1

if w\*x + b > 0:

return 1

else:

return 0

def SUM\_FN(A,B,C):

y1 = AND(AND(NOT(A),NOT(B)),C)

y2 = AND(AND(NOT(A),B),NOT(C))

y3 = AND(AND(A,NOT(B)),NOT(C))

y4 = AND(AND(A,B),C)

return OR(OR(OR(y1,y2),y3),y4)

def COUT\_FN(A,B,C):

y1 = AND(A,B)

y2 = AND(B,C)

y3 = AND(A,C)

return OR(OR(y1,y2),y3)

print("A", "B", "C", "\t", "Sum","\t", "Cout")

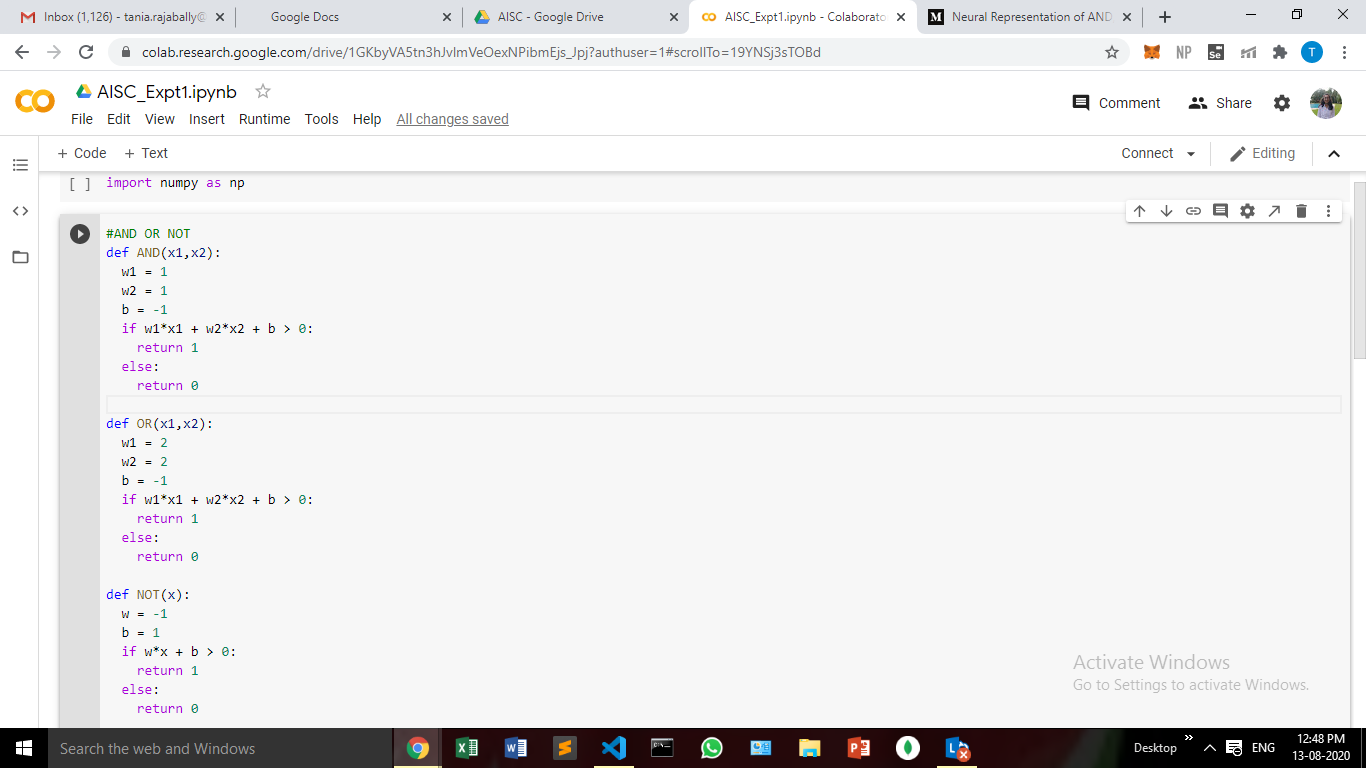
for i in range(2):

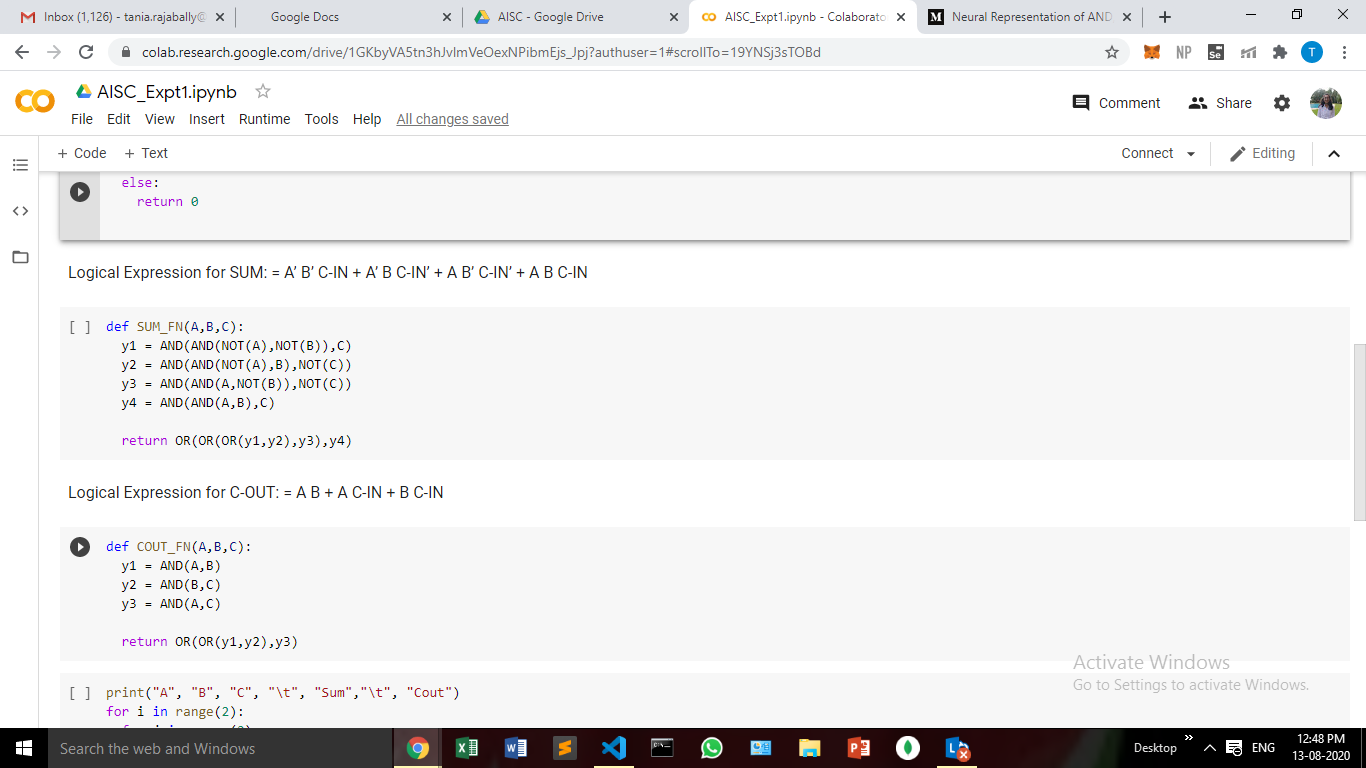
for j in range(2):

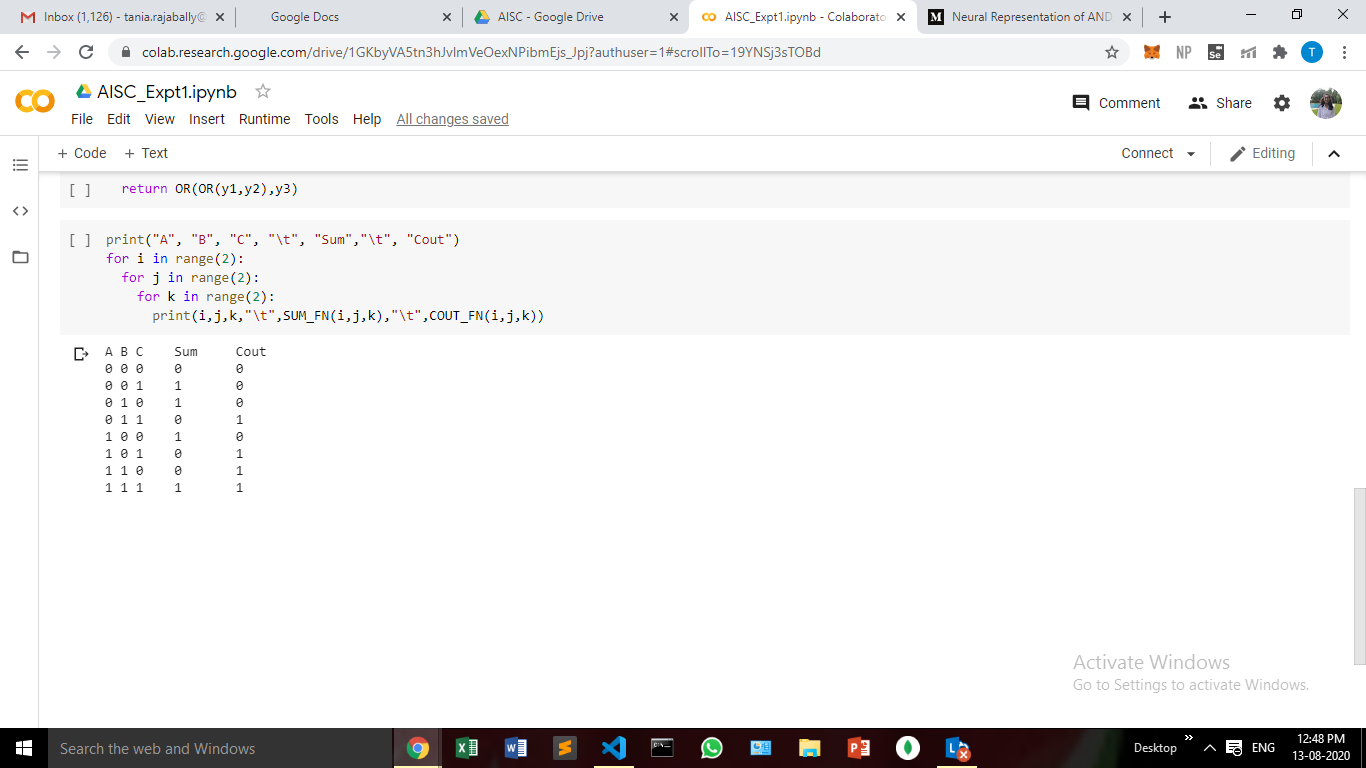
for k in range(2):

print(i,j,k,"\t",SUM\_FN(i,j,k),"\t",COUT\_FN(i,j,k))

**Output:**

****

****

****

**Conclusion:**

In this experiment, we have seen how we can choose weights, bias and a threshold to implement the logic gates and then use that to implement a full adder using the Mc-Culloch Pitts Model.