Date of Performance :

Date of Submission :

EXPERIMENT NUMBER: 1

Aim: Implement Mc-Culloch Pitts model for AND and XOR logic gates.

Objective:

To implement basic neural network models for simulating logic gate

Software Used :Python

Theory:

Mc-Culloch Pitt Model :The McCulloch-Pitts (M-P) model, introduced in 1943 by Warren McCulloch and Walter Pitts, is a foundational model of artificial neural networks. The model is based on the concept of binary neurons, which are either on (1) or off (0), and the model defines a set of logical rules that determine the state of a neuron based on the states of its inputs.

In the M-P model, a neuron is represented as a threshold gate, which receives inputs from other neurons and outputs a binary value depending on whether the sum of the inputs exceeds a threshold value. The threshold gate acts as a simple decision maker, allowing a representation of logical statements such as "if A and B are true, then output 1."

One important aspect of the M-P model is that it allows for the creation of a network of neurons that can work together to solve a problem. For example, multiple neurons can be connected to recognize patterns in input data, making it possible to use the M- P model for applications such as image recognition or decision making.

Despite its simplicity, the M-P model was a major contribution to the development of artificial neural networks and provided a basis for further research in the field. The M-P model inspired the development of more sophisticated neural network models with continuous activation functions and multi-layer networks, which expanded the capability of artificial neural networks from simple decision making to more complex pattern recognition and data analysis.

In conclusion, the McCulloch-Pitts model is a simple yet powerful model of artificial neurons that laid the foundation for the development of artificial intelligence and artificial neural networks. It introduced the idea of binary neurons and showed how a network of neurons can work together to solve problems, paving the way for the development of more advanced neural network models.

Algorithm:

Here is a high-level pseudocode for a McCulloch-Pitts (M-P) model:

```
Input: binary inputs x1, x2, ..., xn
Output: binary output y

Step 1: Initialize weights w1, w2, ..., wn and threshold valueθ
Step 2: Calculate the weighted sum of inputs:
    weighted_sum = w1 * x1 + w2 * x2 + ... + wn * xn
Step 3: Compare weighted_sum to thresholdθ :
    if weighted_sum >θ :
        y = 1
    else:
        y = 0
Step 4: Return output y
```

Note that this is just a high-level pseudocode and not a complete implementation of the M-P model. A complete implementation would typically involve training the network to optimize the weights and threshold based on a set of input-output examples, and would involve additional steps and algorithms for weight update and error calculation.

Program:

*********Mc-Culloch Pitts model for AND gate*****************

```
import numpy as np
def ANDThreshold(v):
  if v > 1:
    return 1
  else:
    return 0
def MCPModel(x, w):
  v = np.dot(w, x)
  y = ANDThreshold(v)
  return y
def ANDGate(x):
  w = np.array([1, 1])
return MCPModel(x, w)
test1 = np.array([0, 0])
test2 = np.array([0, 1])
test3 = np.array([1, 0])
test4 = np.array([1, 1])
print(f"AND({0}, {0}) = { ANDGate(test1)}")
print(f"AND({0}, {1}) = { ANDGate(test2)}")
print(f"AND({1}, {0}) = { ANDGate(test3)}")
print(f"AND({1}, {1}) = { ANDGate(test4)}")
```

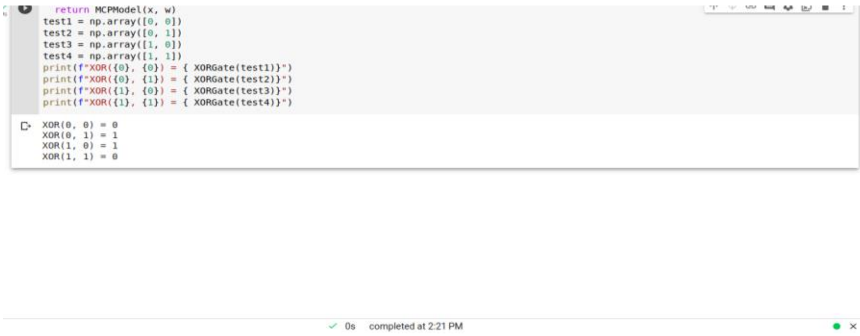
*********Mc-Culloch Pitts model for XOR gate*****************

```
def XORThreshold(v):
  if v == 1:
    return 1
  else:
```

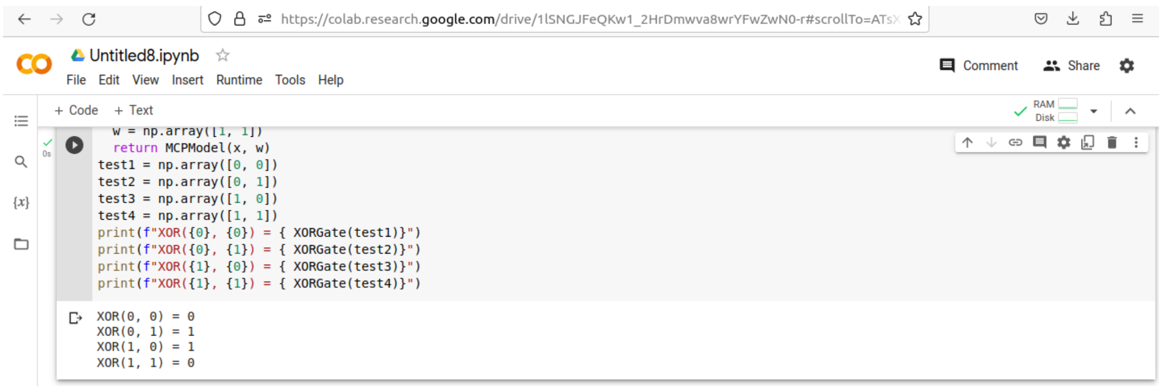
```
   return 0
def MCPModel(x, w):
  v = np.dot(w, x)
  y = XORThreshold(v)
  return y
def XORGate(x):
  w = np.array([1, 1])
return MCPModel(x, w)
test1 = np.array([0, 0])
test2 = np.array([0, 1])
test3 = np.array([1, 0])
test4 = np.array([1, 1])
print(f"XOR({0}, {0}) = { XORGate(test1)}")
print(f"XOR({0}, {1}) = { XORGate(test2)}")
print(f"XOR({1}, {0}) = { XORGate(test3)}")
print(f"XOR({1}, {1}) = { XORGate(test4)}")
```

Output:

Mc-Culloch Pitts model for AND gate



Mc-Culloch Pitts model for OR gate



Conclusion/Outcome:

Thus we have implemented Mc-Culloch Pitts model for AND and XOR logic gates
We also understood that implement basic neural network models for simulating logic gate

Marks & Signature:

R1 (4 Marks)	R2 (4 Marks)	R3 (4 Marks)	R4 (3 Mark)	Total (15 Marks)	Signature