Experiment No. 6

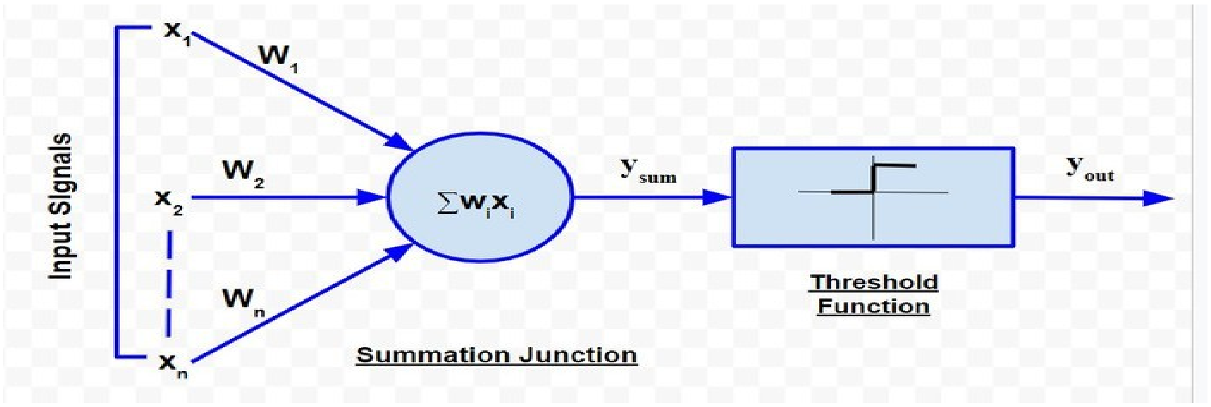
Aim: To implement McCulloch Pitts Model for ANN.

Objective: Able to design a neural network and use activation function as learning rule defined by McCulloch Piits Model.

Theory:

McCulloch-Pitts Model of Neuron

The McCulloch-Pitts neural model, which was the earliest ANN model, has only two types of inputs — **Excitatory and Inhibitory.** The excitatory inputs have weights of positive magnitude and the inhibitory weights have weights of negative magnitude. The inputs of the McCulloch-Pitts neuron could be either 0 or 1. It has a threshold function as an activation function. So, the output signal y_{out} is 1 if the input y_{sum} is greater than or equal to a given threshold value, else 0. The diagrammatic representation of the model is as follows:



Simple McCulloch-Pitts neurons can be used to design logical operations. For that purpose, the connection weights need to be correctly decided along with the threshold function (rather than the threshold value of the activation function).

Example:

John carries an umbrella if it is sunny or if it is raining. There are four given situations. I need to decide when John will carry the umbrella. The situations are as follows:

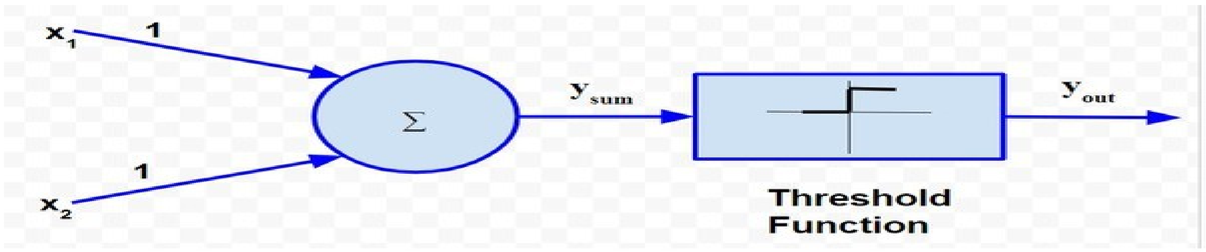
- First scenario: It is not raining, nor it is sunny

- Second scenario: It is not raining, but it is sunny
- Third scenario: It is raining, and it is not sunny
- Fourth scenario: It is raining as well as it is sunny

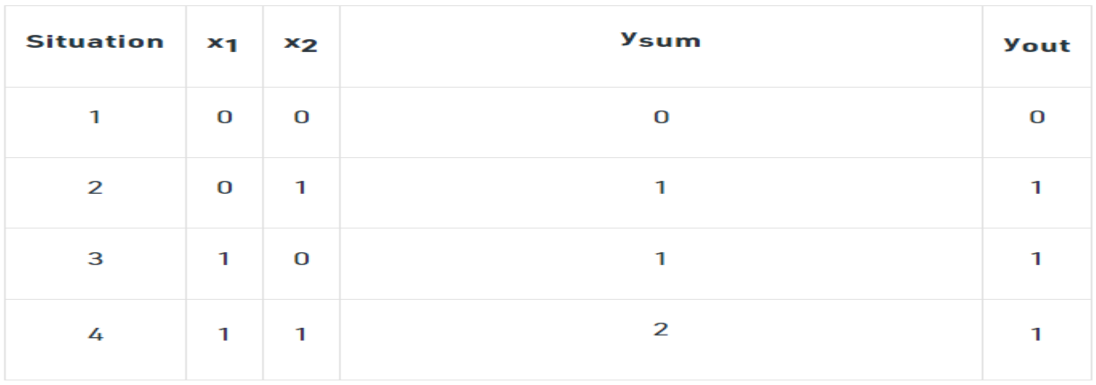
To analyse the situations using the McCulloch-Pitts neural model, I can consider the input signals as follows:

- X_1: Is it raining?
- X_2 : Is it sunny?

So, the value of both scenarios can be either 0 or 1. We can use the value of both weights X_1 and X_2 as 1 and a threshold function as 1. So, the neural network model will look like:



Truth Table for this case will be:



Situation	x1	x2	Ysum	Yout
1	0	0	0	0
2	0	1	1	1
3	1	0	1	1
4	1	1	2	1

So,

$$y_{sum} = \sum_{i=1}^{2} w_i x_i$$

$$y_{out} = f(y_{sum}) = \begin{cases} 1, x \geq 1 \\ 0, x < 1 \end{cases}$$

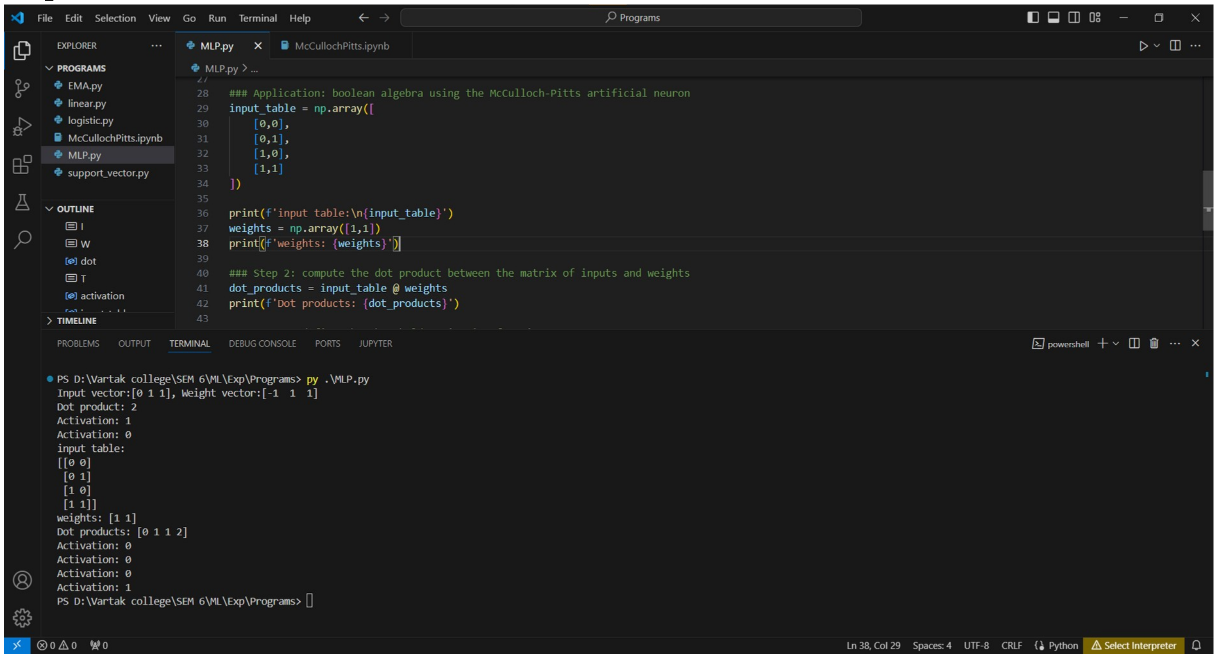
Implementation:

McCulloch Pitts Model:

```
### Step 1: generate a vector of inputs and a vector of weights
import numpy as np
np.random.seed(seed=0)
I = np.random.choice([0,1], 3)
W = np.random.choice([-1,1], 3)
print(f'Input vector:{I}, Weight vector:{W}')
### Step 2: compute the dot product between the vector of inputs and weights
dot = I @ W
print(f'Dot product: {dot}')
### Step 3: define the threshold activation function
def linear_threshold_gate(dot: int, T: float) -> int:
    if dot >= T:
        return 1
    else:
        return 0
## Step 4: compute the output based on the threshold value
T = 1
activation = linear_threshold_gate(dot, T)
print(f'Activation: {activation}')
T = 3
activation = linear_threshold_gate(dot, T)
print(f'Activation: {activation}')
### Application: boolean algebra using the McCulloch-Pitts artificial neuron
input_table = np.array([
    [0,0],
    [0,1],
    [1,0],
    [1,1]
])
```

```
print(f'input table:\n{input_table}')
weights = np.array([1,1])
print(f'weights: {weights}')
### Step 2: compute the dot product between the matrix of inputs and weights
dot_products = input_table @ weights
print(f'Dot products: {dot_products}')
### Step 3: define the threshold activation function
def linear_threshold_gate(dot: int, T: float) -> int:
    if dot >= T:
        return 1
    else:
        return 0
### Step 4: compute the output based on the threshold value
T = 2
for i in range(0,4):
    activation = linear_threshold_gate(dot_products[i], T)
    print(f'Activation: {activation}')
```

Output:



```
### Application: boolean algebra using the McCulloch-Pitts artificial neuron
input_table = np.array([
    [0,0],
    [0,1],
    [1,0],
    [1,1]
])

print(f'input table:\n{input_table}')
weights = np.array([1,1])
print(f'weights: {weights}')

### Step 2: compute the dot product between the matrix of inputs and weights
dot_products = input_table @ weights
print(f'Dot products: {dot_products}')
```

```
PS D:\Vartak college\SEM 6\ML\Exp\Programs> py .\MLP.py
Input vector:[0 1 1], Weight vector:[-1  1  1]
Dot product: 2
Activation: 1
Activation: 0
input table:
[[0 0]
 [0 1]
 [1 0]
 [1 1]]
weights: [1 1]
Dot products: [0 1 1 2]
Activation: 0
Activation: 0
Activation: 0
Activation: 1
PS D:\Vartak college\SEM 6\ML\Exp\Programs>
```

Conclusion:

What is the use of McCulloch Pitts model?

- The McCulloch-Pitts model is used to simulate the behavior of a biological neuron.
- It helps understand the basic principles of neural computation, particularly in terms of binary threshold logic.
- The model can be applied to solve simple logical operations, such as AND, OR, and NOT.

How it will be used to develop ANN?

- McCulloch-Pitts neurons are the basic building blocks of artificial neural networks.
- In ANN, multiple interconnected neurons form layers, and information is propagated through these layers.
- Each neuron receives inputs, computes a weighted sum of these inputs, applies an activation function (often a threshold function), and produces an output.
- By combining multiple neurons in layers and adjusting the weights and thresholds, ANNs can learn to perform complex tasks, such as pattern recognition, classification, and regression.
- ANN architectures, such as feedforward neural networks, recurrent neural networks, and convolutional neural networks, are based on the principles derived from the McCulloch-Pitts model.