EX NO :4
DATE:

## A PYTHON PROGRAM TO IMPLEMENT SINGLE LAYER PERCEPTRON

**AIM:**

To implement python program for the single layer perceptron.

**PROGRAM:**

```
import numpy as np
import pandas as pd
input_value=np.array ([[0,0] ,[0,1], [1,1], [1,0]])
input_value.shape
#(4,2)
output = np.array([0,0,1,0])
output = output.reshape(4,1)
output.shape
#(4,1)
weights=np.array([[0.1],[0.3]])
weights
#array ([[0.1], [0.3]])
bias = 0.2
def sigmoid_func(x):
    return 1/(1+np.exp(-x))
def der(x):
    return sigmoid_func(x)*(1 - sigmoid_func(x))
for epochs in range(15000):
    input_arr = input_value
    weighted_sum=np.dot(input_arr,weights)+bias
    first_output=sigmoid_func(weighted_sum)
    error=first_output - output
    total_error=np.square(np.subtract(first_output,output)).mean()
    first_der=error
    second_der=der(first_output)
    derivative=first_der*second_der
    t_input = input_value.T
    final_derivative=np.dot(t_input,derivative)
    weights=weights - (0.05 * final_derivative)
    for i in derivative:
        bias=bias-(0.05*i)
print(weights)
print(bias)
#[16.57299223]
#[16.57299223]]
#[-25.14783487]
pred=np.array([1,0])
result = np.dot(pred,weights)+bias
res = sigmoid_func(result)
print(res)
#[0.00018876]
pred=np.array([1,1])
result = np.dot(pred,weights)+bias
res = sigmoid_func(result)
```

```
print(res)
#[0.99966403]
pred=np.array([0,0])
result = np.dot(pred,weights)+bias
res = sigmoid_func(result)
print(res)
#[1.19793729e-11]
pred=np.array([0,1])
result = np.dot(pred,weights)+bias
res = sigmoid_func(result)
print(res)
#[0.00063036]\
```

**OUTPUT:**

```
[[6.62916366]
 [6.62916441]]
[-10.23197316]
[0.02652435]
[0.95375065]
[3.59993686e-05]
[0.02652437]
```

**RESULT:**

Thus, the Python program to implement a single-layer perceptron has been executed successfully.