

## **A PYTHON PROGRAM TO IMPLEMENT SINGLE LAYER PERCEPTRON**

EXPNO. 4

Name: MOHAMED FUZAIL A

Roll no: 241801162

Code:

```
import numpy as np
```

```
import pandas as pd
```

```
input_value = np.array([[0, 0], [0, 1], [1, 1], [1, 0]])
```

```
print(input_value.shape)
```

```
output = np.array([0, 0, 1, 0])
```

```
output = output.reshape(4, 1)
```

```
print(output.shape)
```

```
weights = np.array([[0.1], [0.3]])
```

```
print(weights)
```

```
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)
```

```
pred = np.array([1, 0])  
result = np.dot(pred, weights) + bias  
res = sigmoid_func(result)  
print(res)
```

```
pred = np.array([1, 1])  
result = np.dot(pred, weights) + bias  
res = sigmoid_func(result)  
print(res)
```

```
pred = np.array([0, 0])  
result = np.dot(pred, weights) + bias  
res = sigmoid_func(result)  
print(res)
```

```
pred = np.array([0, 1])  
result = np.dot(pred, weights) + bias  
res = sigmoid_func(result)  
print(res)
```

output:

```
(4, 2)
(4, 1)
[[0.1]
 [0.3]]
[[6.62916366]
 [6.62916441]]
[-10.23197316]
[0.02652435]
[0.95375065]
[3.59993686e-05]
[0.02652437]
```