EXPERIMENT -4

A python program to implement single layer perceptron.

AIM:

To code a python program to implement single layer perceptron.

CODE:

```
import numpy as np
import pandas as pd

# Input and Output
input_value = np.array([[0,0], [0,1], [1,1], [1,0]])
output = np.array([0,0,1,0]).reshape(4,1)

# Initialize weights and bias
weights = np.array([[0.1],[0.3]])
bias = 0.2
```

```
# Activation functions
def sigmoid_func(x):
    return 1/(1+np.exp(-x))

def der(x):
    return sigmoid_func(x)*(1 - sigmoid_func(x))

# Training loop
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)

    # Update weights and bias
```

```
    weights = weights - (0.05 * final_derivative)
    for i in derivative:
        bias = bias - (0.05 * i)

# Final weights and bias
print("Final Weights:\n", weights)
print("Final Bias:", bias)

# Predictions
predictions = [
    np.array([1,0]),
    np.array([1,1]),
    np.array([0,0]),
    np.array([0,1])
]

print("\nPredictions:")
for pred in predictions:
    result = np.dot(pred, weights) + bias
    res = sigmoid_func(result)
    print(f"Input {pred} => Output {res}")
```

OUTPUT:

Final Weights:

[[16.57299223]

[16.57299223]]

Final Bias: -25.14783487087293

Predictions:

Input [1 0] => Output [0.00018876]

Input [1 1] => Output [0.99966403]

Input [0 0] => Output [1.19793729e-11]

Input [0 1] => Output [0.00063036]

RESULT:

Thus a python program to implement single layer perceptron.