```
import numpy as np

# Adaline neural network
def Adaline(Input, Target, lr=0.2, stop=0.001):
    weight = np.random.random(Input.shape[1])  # Initialize weights randomly
    bias = np.random.random(1)  # Initialize bias randomly

    Error = [stop + 1]  # Initialize error list to track convergence
    # Check the stop condition for the network
    while Error[-1] > stop or (Error[-1] - Error[-2]) > 0.0001:
        error = []
        for i in range(Input.shape[0]):
            Y_input = np.dot(Input[i], weight) + bias  # Calculate output

            # Update the weight
            for j in range(Input.shape[1]):
                weight[j] = weight[j] + lr * (Target[i] - Y_input) * Input[i][j]

            # Update the bias
            bias = bias + lr * (Target[i] - Y_input)

            # Store squared error value
            error.append((Target[i] - Y_input)**2)
        # Store sum of square errors
        Error.append(sum(error))
        print('Error:', Error[-1])

    return weight, bias

# Input dataset for AND gate
x = np.array([[0.0, 0.0],
              [0.0, 1.0],
              [1.0, 0.0],
              [1.0, 1.0]])

# Target values for AND gate
t = np.array([0, 0, 0, 1])

# Train the Adaline network
w, b = Adaline(x, t, lr=0.1, stop=0.001)
```

```
# Display the learned weights and bias
print('Learned weight:', w)
print('Learned bias:', b)

# Test the Adaline network
def test_adaline(inputs, weight, bias):
    return np.dot(inputs, weight) + bias

# Test the network on the inputs for AND gate
print("\nTesting the trained Adaline network on AND gate inputs:")
for i in range(x.shape[0]):
    result = test_adaline(x[i], w, b)
    print(f"Input: {x[i]} -> Output: {result} (Target: {t[i]})")
```