# Perceptron withou any Library @ Logical Gates

```
import numpy as np

class Perceptron:
  def __init__(self,input_size, learning_rate=0.0001):
    self.weight = np.zeros(input_size)
    self.bias = 0
    self.learning_rate = learning_rate

    #Function For Activation Function
    #we are using Heaviside Activation Function
  def activation_function(self, x):
      return 1 if x>= 0 else 0

    #function For Summation
    #x1.w1+x2.w2+....+xn.wn
  def predict(self, inputs):
      linear_output = np.dot(inputs, self.weight)+ self.bias
      #this output will be passed into the activation function
      return self.activation_function(linear_output)


    #defineing function for Train
  def train(self, X, y, epochs=100):
      for epoch in range (epochs):
        print(f"\nEpoch {epoch+1}:")
        for inputs, label in zip(X, y):


          #predicting the output
          prediction = self.predict(inputs)

          #calculating error
          error=label-prediction


          #updating weights to minimize the error
          self.weight += self.learning_rate*error*inputs
          #updating bias
          self.bias += self.learning_rate*error



          #final predictions for outputs
          print(f"Input: {inputs}, Prediction: {prediction}, True:
{label}, Error: {error}")
```

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


perceptron = Perceptron(input_size=2)
perceptron.train(X, y, epochs=100)

print("\n Testing AND Gate:")

for inputs in X:
  print(f"Input: {inputs}, Predicted Output:
{perceptron.predict(inputs)}")


Epoch 1:
Input: [0 0], Prediction: 1, True: 0, Error: -1
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 0, True: 1, Error: 1

Epoch 2:
Input: [0 0], Prediction: 1, True: 0, Error: -1
Input: [0 1], Prediction: 1, True: 0, Error: -1
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 0, True: 1, Error: 1

Epoch 3:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 1, True: 0, Error: -1
Input: [1 0], Prediction: 1, True: 0, Error: -1
Input: [1 1], Prediction: 0, True: 1, Error: 1

Epoch 4:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 5:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 6:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 7:
```

```
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 8:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 9:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 10:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 11:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 12:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 13:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 14:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 15:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
```

```
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 16:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 17:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 18:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 19:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 20:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 21:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 22:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 23:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0
```

```
Epoch 24:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 25:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 26:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 27:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 28:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 29:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 30:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 31:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 32:
```

```
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 33:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 34:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 35:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 36:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 37:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 38:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 39:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 40:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
```

```
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 41:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 42:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 43:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 44:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 45:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 46:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 47:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 48:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0
```

```
Epoch 49:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 50:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 51:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 52:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 53:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 54:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 55:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 56:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 57:
```

```
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 58:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 59:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 60:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 61:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 62:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 63:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 64:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 65:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
```

```
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 66:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 67:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 68:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 69:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 70:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 71:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 72:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 73:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0
```

```
Epoch 74:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 75:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 76:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 77:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 78:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 79:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 80:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 81:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 82:
```

```
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 83:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 84:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 85:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 86:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 87:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 88:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 89:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 90:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
```

```
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 91:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 92:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 93:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 94:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 95:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 96:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 97:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 98:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0
```

```
Epoch 99:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 100:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 0, Error: 0
Input: [1 0], Prediction: 0, True: 0, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

 Testing AND Gate:
Input: [0 0], Predicted Output: 0
Input: [0 1], Predicted Output: 0
Input: [1 0], Predicted Output: 0
Input: [1 1], Predicted Output: 1
```

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


perceptron.train(X, y, epochs=10)



print("\nTesting OR Gate:")
for inputs in X:
    print(f"Input: {inputs}, Predicted Output:
{perceptron.predict(inputs)}")
```

```
Epoch 1:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 0, True: 1, Error: 1
Input: [1 0], Prediction: 1, True: 1, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 2:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 1, True: 1, Error: 0
Input: [1 0], Prediction: 1, True: 1, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 3:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 1, True: 1, Error: 0
Input: [1 0], Prediction: 1, True: 1, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0
```

```
Epoch 4:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 1, True: 1, Error: 0
Input: [1 0], Prediction: 1, True: 1, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 5:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 1, True: 1, Error: 0
Input: [1 0], Prediction: 1, True: 1, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 6:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 1, True: 1, Error: 0
Input: [1 0], Prediction: 1, True: 1, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 7:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 1, True: 1, Error: 0
Input: [1 0], Prediction: 1, True: 1, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 8:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 1, True: 1, Error: 0
Input: [1 0], Prediction: 1, True: 1, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 9:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 1, True: 1, Error: 0
Input: [1 0], Prediction: 1, True: 1, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Epoch 10:
Input: [0 0], Prediction: 0, True: 0, Error: 0
Input: [0 1], Prediction: 1, True: 1, Error: 0
Input: [1 0], Prediction: 1, True: 1, Error: 0
Input: [1 1], Prediction: 1, True: 1, Error: 0

Testing OR Gate:
Input: [0 0], Predicted Output: 0
Input: [0 1], Predicted Output: 1
Input: [1 0], Predicted Output: 1
Input: [1 1], Predicted Output: 1
```

# Perceptron Using an inbuilt library @ Logical Gates

```
from sklearn.linear_model import Perceptron
import numpy as np

X=np.array([[0,0],[0,1],[1,0],[1,1]])
y=np.array([0,0,0,1])

model = Perceptron(max_iter=100, eta0=0.1, random_state=42)

model.fit(X,y)
```

```
Perceptron(eta0=0.1, max_iter=100, random_state=42)
```

```
print("\nTesting AND Gate:")
for inputs in X:
    predicted_output = model.predict([inputs])[0]
    print(f"Input: {inputs}, Predicted Output: {predicted_output}")
```

```
Testing AND Gate:
Input: [0 0], Predicted Output: 0
Input: [0 1], Predicted Output: 0
Input: [1 0], Predicted Output: 0
Input: [1 1], Predicted Output: 1
```

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

model = Perceptron(max_iter=100, eta0=0.1, random_state=42)

model.fit(X,y)
```

```
Perceptron(eta0=0.1, max_iter=100, random_state=42)
```

```
print("\nTesting OR Gate:")
for inputs in X:
    predicted_output = model.predict([inputs])[0]
    print(f"Input: {inputs}, Predicted Output: {predicted_output}")
```

```
Testing OR Gate:
Input: [0 0], Predicted Output: 0
Input: [0 1], Predicted Output: 1
Input: [1 0], Predicted Output: 1
Input: [1 1], Predicted Output: 1
```

# Implementing Perceptron for Binary Classification by taking a dataset from Scratch

```
import numpy as np

#loading my data
def load_data(file_path):
    data = []
    labels = []

    with open(file_path, 'r') as file:
        for line in file:
            values = line.strip().split(',')

            # Extract features (all columns except the last one) and
label (last column)
            features = [float(value) for value in values[:-1]]
            label = int(values[-1])  # Last value is the label (0 or
1)

            data.append(features)
            labels.append(label)

    return np.array(data), np.array(labels)



# Path to your text file containing the dataset
file_path = '/content/data_banknote_authentication.txt'  # Replace
with your actual path

# Load the dataset
X, y = load_data(file_path)

# Split the data into training (80%) and test (20%) sets
train_size = int(0.8 * len(X))
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

# Normalize the features (important for training the perceptron)
X_train = (X_train - X_train.mean(axis=0)) / X_train.std(axis=0)
X_test = (X_test - X_test.mean(axis=0)) / X_test.std(axis=0)

print(f"Training data size: {X_train.shape[0]}, Test data size:
{X_test.shape[0]}")
```

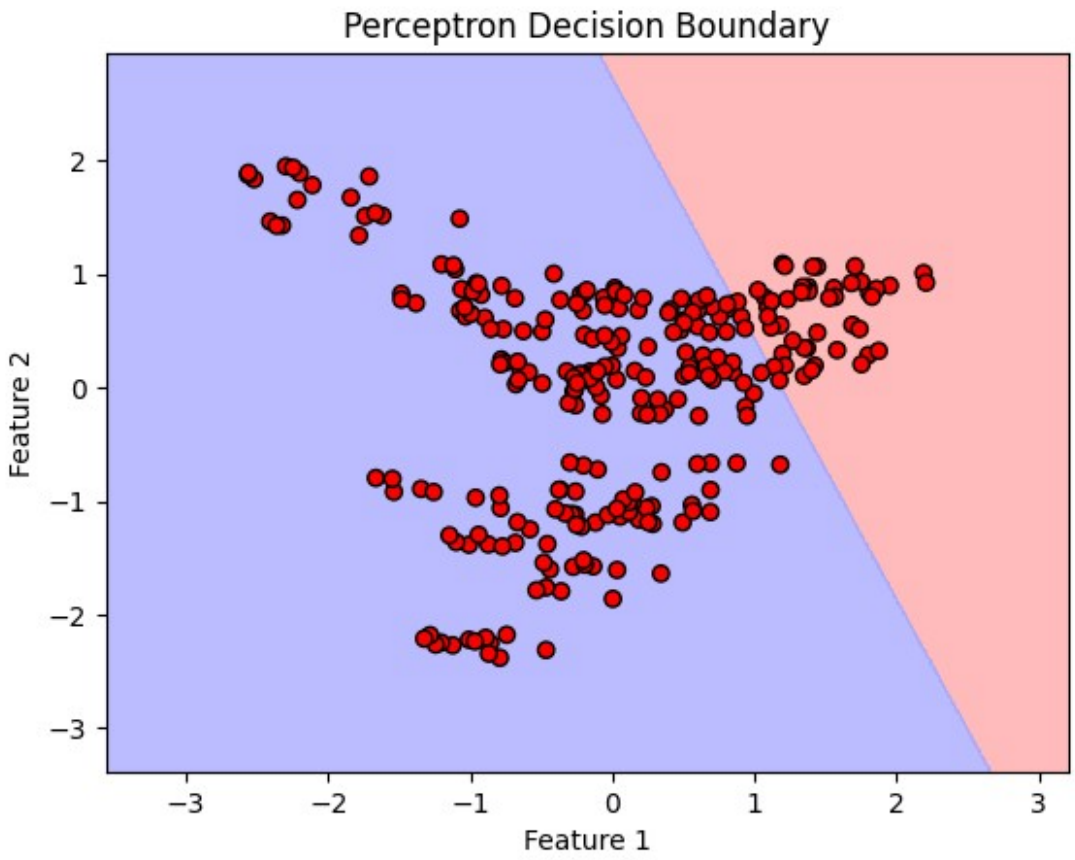
```
Training data size: 1097, Test data size: 275
```

Appying perceptron for classification

```python
class Perceptron:
  def __init__(self, input_size, learning_rate=0.001):
    self.weights = np.zeros(input_size)
    self.bias = 0
    self.learning_rate = learning_rate

  def activation_function(self, x):
    return 1 if x>=0 else 0

  def predict(self, X):
    linear_output = np.dot(X, self.weights) + self.bias
    return self.activation_function(linear_output)

  def train(self, X, y, epochs=100):
    for epochs in range(epochs):
      for i in range(len(X)):
        prediction = self.predict(X[i])

        error = y[i]-prediction


        self.weights +=self.learning_rate*error*X[i]
        self.bias += self.learning_rate*error

      if epochs % 10 ==0:
        accuracy = self.evaluate(X,y)
        print(f"Epoch {epochs}, Accuracy: {accuracy}% ")
  def evaluate(self, X, y):
    correct_prediction = 0
    for i in range(len(X)):
      prediction = self.predict(X[i])
      if prediction ==y[i]:
        correct_prediction +=1
    return (correct_prediction/ len(X)) * 100



input_size = X_train.shape[1]
perceptron = Perceptron(input_size=input_size, learning_rate=0.001)

perceptron.train(X_train, y_train, epochs=100)

test_accuracy = perceptron.evaluate(X_test, y_test)
print(f"Test Accuracy:{test_accuracy}%")
```

```
Epoch 0, Accuracy: 81.22151321786691%
Epoch 10, Accuracy: 95.80674567000912%
Epoch 20, Accuracy: 96.53600729261622%
Epoch 30, Accuracy: 97.62989972652689%
Epoch 40, Accuracy: 97.35642661804923%
Epoch 50, Accuracy: 97.35642661804923%
```

```
Epoch 60, Accuracy: 97.35642661804923%
Epoch 70, Accuracy: 97.44758432087511%
Epoch 80, Accuracy: 97.72105742935278%
Epoch 90, Accuracy: 98.08568824065634%
Test Accuracy:22.181818181818183%

import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

def plot_decision_boundary(X, y, model):
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01),
                         np.arange(y_min, y_max, 0.01))

    Z = np.array([model.predict(point) for point in np.c_[xx.ravel(),
yy.ravel()]])
    Z = Z.reshape(xx.shape)

    cmap_background = ListedColormap(["#FFAAAA", "#AAAAFF"])
    cmap_points = ListedColormap(["#FF0000", "#0000FF"])

    plt.contourf(xx, yy, Z, alpha=0.8, cmap=cmap_background)
    plt.scatter(X[:, 0], X[:, 1], c=y, edgecolor='k',
cmap=cmap_points)
    plt.title("Perceptron Decision Boundary")
    plt.xlabel("Feature 1")
    plt.ylabel("Feature 2")
    plt.show()

X_train_2D = X_train[:, :2]
X_test_2D = X_test[:, :2]


perceptron_2D = Perceptron(input_size=2, learning_rate=0.01)
perceptron_2D.train(X_train_2D, y_train, epochs=100)


plot_decision_boundary(X_test_2D, y_test, perceptron_2D)

Epoch 0, Accuracy: 64.9954421148587%
Epoch 10, Accuracy: 46.58158614402917%
Epoch 20, Accuracy: 56.42661804922516%
Epoch 30, Accuracy: 60.71103008204193%
Epoch 40, Accuracy: 54.14767547857794%
Epoch 50, Accuracy: 47.21969006381039%
Epoch 60, Accuracy: 46.672743846855056%
Epoch 70, Accuracy: 46.672743846855056%
Epoch 80, Accuracy: 46.763901549680945%
Epoch 90, Accuracy: 46.94621695533272%
```



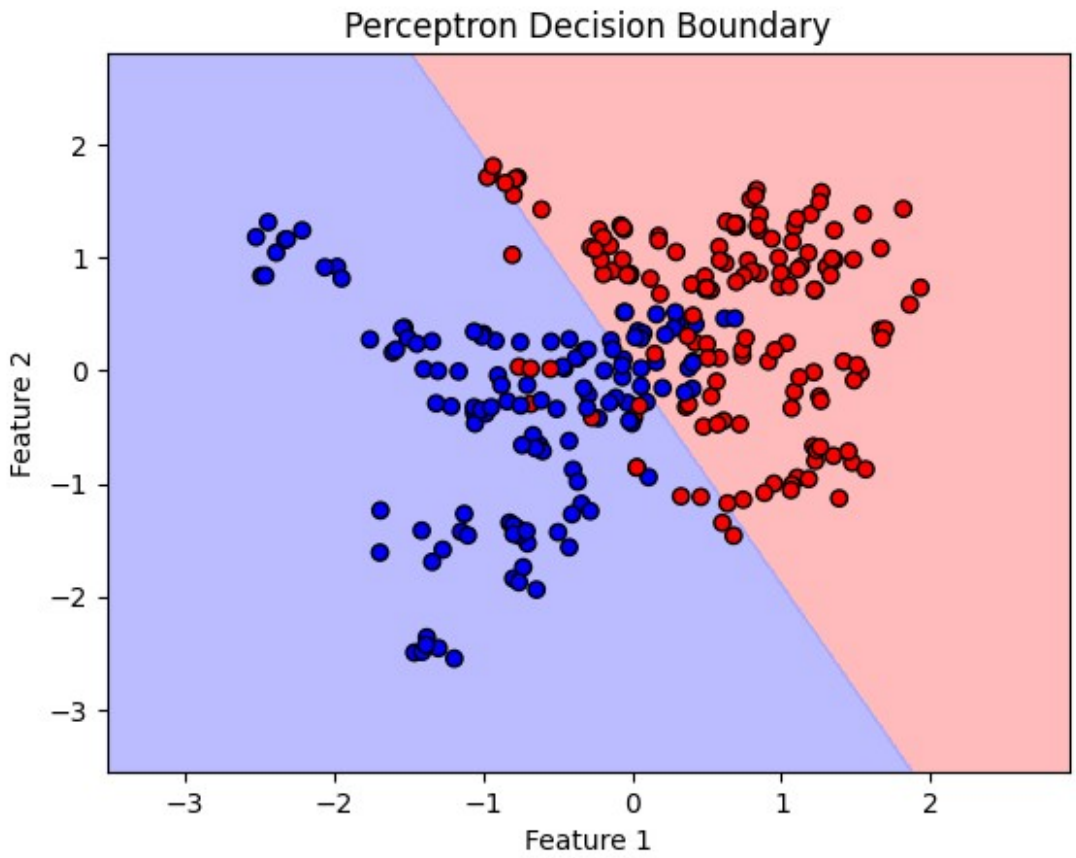

# Implementing Perceptron for Binary Classification by taking a dataset by importing libraries

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import Perceptron
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_classification
from matplotlib.colors import ListedColormap

def load_data(file_path):
    data = []
    labels = []
    with open(file_path, 'r') as file:
        for line in file:
            values = line.strip().split(',')
            data.append([float(value) for value in values[:-1]])
```

```
            labels.append(int(values[-1]))
    return np.array(data), np.array(labels)

file_path ='/content/data_banknote_authentication.txt'
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

scaler =StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

perceptron = Perceptron(max_iter=100, eta0=0.001, random_state=42)
perceptron.fit(X_train[:, :2], y_train)
```

```
Perceptron(eta0=0.001, max_iter=100, random_state=42)
```

```
def plot_decision_boundary(X, y, model):
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01),
                         np.arange(y_min, y_max, 0.01))

    # Predict for each point in the grid
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    # Plot decision boundary
    cmap_background = ListedColormap(["#FFAAAA", "#AAAAFF"])
    cmap_points = ListedColormap(["#FF0000", "#0000FF"])
    plt.contourf(xx, yy, Z, alpha=0.8, cmap=cmap_background)
    plt.scatter(X[:, 0], X[:, 1], c=y, edgecolor='k',
cmap=cmap_points)
    plt.title("Perceptron Decision Boundary")
    plt.xlabel("Feature 1")
    plt.ylabel("Feature 2")
    plt.show()

plot_decision_boundary(X_test[:, :2], y_test, perceptron)
```




lets try with some input data

```
def make_prediction(input_data):
  input_data= np.array(input_data).reshape(1, -1)
  input_data = scaler.transform(input_data)
  prediction = perceptron.predict(input_data[:, :2])
  return"Class 0(Benign)"if prediction[0]==0 else "Class 1
(Malignant)"

ex=[3.5,8.5,-2.0,-0.5]
print(f"Prediction for:{make_prediction(ex)}")

Prediction for:Class 0(Benign)
```

# Implement AND, OR, and XOR gates using a single-layer perceptron from Scratch

```
import numpy as np

class LogicalGatePerceptron:
  def __init__(self, learning_rate=0.0001, epochs=100):
```

```
    self.learning_rate =learning_rate
    self.epochs=epochs
    self.weights =None
    self.bias=None


  def activation(self,x):
    return 1 if x >= 0 else 0

  def train(self, X, y):
    n_samples, n_features=X.shape
    self.weights = np.zeros(n_features)
    self.bias = 0

    for _ in range(self.epochs):
      for idx, x_i in enumerate(X):
        linear_output = np.dot(x_i, self.weights)+self.bias
        y_pred = self.activation(linear_output)

        update=self.learning_rate*(y[idx]-y_pred)
        self.weights += update*x_i
        self.bias += update


  def predict(self, X):
    linear_output = np.dot(X, self.weights) + self.bias
    return np.array([self.activation(x) for x in linear_output])
```

defining data for logical gates

```
def logical_gate_dataset(gate_type):
  if gate_type=="AND":
    X=np.array([[0, 0],[0,1],[1,0],[1,1]])
    y=np.array([0,0,0,1])

  elif gate_type=="OR":
    X=np.array([[0,0],[0,1],[1,0],[1,1]])
    y=np.array([0,1,1,1])

  elif gate_type == "XOR":
    X=np.array([[0,0],[0,1],[1,0],[1,1]])
    y=np.array([0,1,1,0])

  else:
    raise ValueError("Unknown gate Type ")
  return X,y
```

```
for gate in ["AND", "OR", "XOR"]:
  X, y =logical_gate_dataset(gate)
  perceptron = LogicalGatePerceptron(learning_rate=0.001, epochs=100)
  perceptron.train(X,y)
  predictions = perceptron.predict(X)
  print(f"GAte:{gate}")
  print(f"Predictions:{predictions}")
  print(f"Actual:{y}")
  print("-"*30)
```

```
GAte:AND
Predictions:[0 0 0 1]
Actual:[0 0 0 1]
------------------------------
GAte:OR
Predictions:[0 1 1 1]
Actual:[0 1 1 1]
------------------------------
GAte:XOR
Predictions:[1 1 0 0]
Actual:[0 1 1 0]
------------------------------
```

# implement AND, OR, and XOR gates using a single-layer perceptron by importing libraries

```
from sklearn.linear_model import Perceptron
from sklearn.metrics import accuracy_score

def logical_gate_dataset(gate_type):
    if gate_type == "AND":
        X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
        y = np.array([0, 0, 0, 1])
    elif gate_type == "OR":
        X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
        y = np.array([0, 1, 1, 1])
    elif gate_type == "XOR":
        X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
        y = np.array([0, 1, 1, 0])
    else:
        raise ValueError("Unknown gate type. Choose 'AND', 'OR', or
'XOR'.")
    return X, y

for gate in ["AND", "OR", "XOR"]:
    X, y = logical_gate_dataset(gate)
    perceptron = Perceptron(max_iter=1000, eta0=0.1, random_state=42)
```

```
    perceptron.fit(X, y)
    predictions = perceptron.predict(X)
    accuracy = accuracy_score(y, predictions)
    print(f"Gate: {gate}")
    print(f"Predictions: {predictions}")
    print(f"Actual: {y}")
    print(f"Accuracy: {accuracy * 100:.2f}%")
    print("-" * 30)
```

```
Gate: AND
Predictions: [0 0 0 1]
Actual: [0 0 0 1]
Accuracy: 100.00%
------------------------------
Gate: OR
Predictions: [0 1 1 1]
Actual: [0 1 1 1]
Accuracy: 100.00%
------------------------------
Gate: XOR
Predictions: [0 0 0 0]
Actual: [0 1 1 0]
Accuracy: 50.00%
------------------------------
```

# Implemeting Binary classification on dataset by using Single Layered Perceptron from scratch

data preprocessing

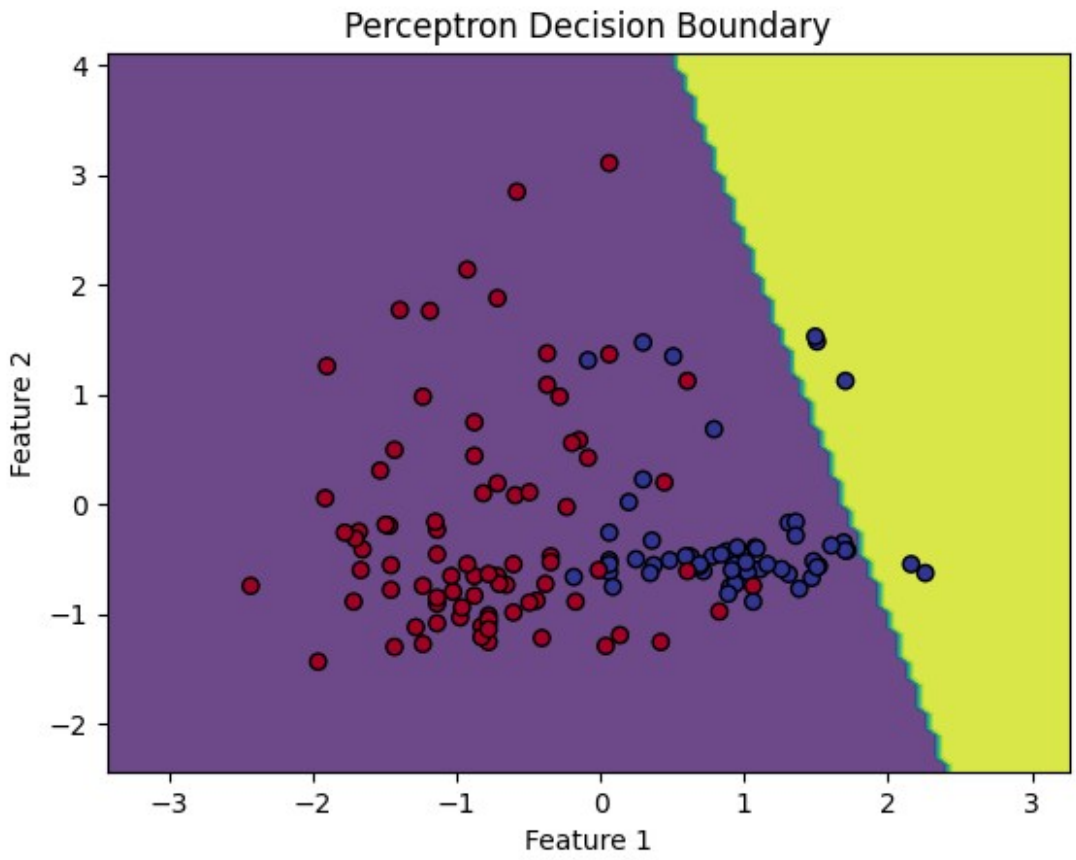
```
def load_data(file_path):
  data=[]
  labels=[]

  with open(file_path, 'r') as file:
    for line in file:
      values = line.strip().split(',')
      data.append([float(value) for value in values[1:]])
      labels.append(1 if int(values[0])== 1 else 0)

  return data, labels

#data Loading
file_path ="wine.data"
X,y = load_data(file_path)
```

```
def normalize_features(X):
  X=np.array(X)
  mean=X.mean(axis=0)
  std=X.std(axis=0)
  X_normalized=(X-mean)/std

  return X_normalized

X_normalized = normalize_features(X)
train_size=int(0.8*len(X_normalized))
X_train, X_test=X_normalized[:train_size],X_normalized[train_size:]
y_train,y_test = y[:train_size],y[train_size:]

print(f"traing size:{len(X_train)},test size:{len(X_test)}")
```

```
traing size:142,test size:36
```

```
import numpy as np

class Perceptron:
    def __init__(self, input_size, learning_rate=0.01, epochs=1000):
        self.weights = np.zeros(input_size)
        self.bias = 0
        self.learning_rate = learning_rate
        self.epochs = epochs
    def predict(self, X):
        linear_output = np.dot(X, self.weights) + self.bias
        return np.where(linear_output >= 0, 1, 0)
    def fit(self, X, y):
        for epoch in range(self.epochs):
            for i in range(len(X)):
                prediction = self.predict(X[i])
                error = y[i] - prediction
                self.weights += self.learning_rate * error * X[i]
                self.bias += self.learning_rate * error
    def evaluate(self, X, y):
        # Evaluate the perceptron
        predictions = self.predict(X)
        accuracy = np.mean(predictions == y)  # accuracy = correct
predictions / total samples
        return accuracy

input_size = len(X_train[0])
perceptron = Perceptron(input_size=input_size)

perceptron.fit(X_train, y_train)

train_accuracy = perceptron.evaluate(X_train, y_train)
test_accuracy = perceptron.evaluate(X_test, y_test)
```

```
print(f"Training Accuracy: {train_accuracy * 100:.2f}%")
print(f"Test Accuracy: {test_accuracy * 100:.2f}%")

Training Accuracy: 100.00%
Test Accuracy: 97.22%

X_train_2D = X_train[:, :2]
X_test_2D = X_test[:, :2]

input_size = X_train_2D.shape[1]
perceptron_2D = Perceptron(input_size=input_size)
perceptron_2D.fit(X_train_2D, y_train)

def plot_decision_boundary(X, y, perceptron):
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.linspace(x_min, x_max, 100),
np.linspace(y_min, y_max, 100))
    Z = perceptron.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    plt.contourf(xx, yy, Z, alpha=0.8)

    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.RdYlBu,
edgecolors='k', marker='o')
    plt.title("Perceptron Decision Boundary")
    plt.xlabel("Feature 1")
    plt.ylabel("Feature 2")
    plt.show()

plot_decision_boundary(X_train_2D, y_train, perceptron_2D)
```



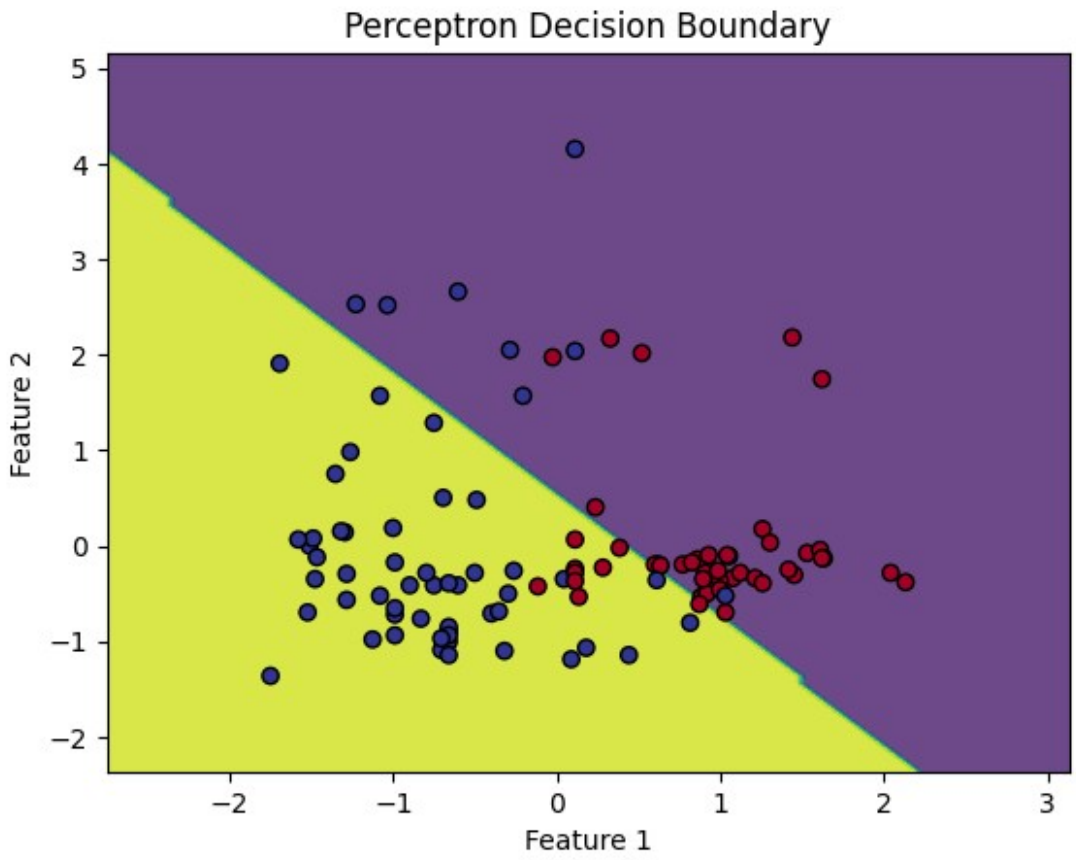

# Implemeting Binary classification on dataset by using Single Layered Perceptron with inbuilt LIbraries

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets
from sklearn.linear_model import Perceptron
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

wine = datasets.load_wine()

X = wine.data[wine.target != 2, :2]
y = wine.target[wine.target != 2]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

perceptron = Perceptron(random_state=42)

perceptron.fit(X_train, y_train)
```

```
Perceptron(random_state=42)
```

```
def plot_decision_boundary(X, y, perceptron):
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.linspace(x_min, x_max, 100),
np.linspace(y_min, y_max, 100))

    Z = perceptron.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    plt.contourf(xx, yy, Z, alpha=0.8)

    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.RdYlBu,
edgecolors='k', marker='o')
    plt.title("Perceptron Decision Boundary")
    plt.xlabel("Feature 1")
    plt.ylabel("Feature 2")
    plt.show()

plot_decision_boundary(X_train, y_train, perceptron)
```




# Implementing multi-dimensional classification by single layered perceptron on dataset

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets
from sklearn.linear_model import Perceptron
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

wine=datasets.load_wine()

X=wine.data
y=wine.target

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

perceptron = Perceptron(random_state=42)

perceptron.fit(X_train, y_train)
```

```
Perceptron(random_state=42)
```

```
accuracy = perceptron.score(X_test, y_test)
print(f"Accuracy of Perceptron on test data: {accuracy * 100:.2f}%")
```

```
Accuracy of Perceptron on test data: 100.00%
```

```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

plt.figure(figsize=(10, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y, cmap=plt.cm.RdYlBu,
edgecolors='k', marker='o')
plt.title("PCA of Wine Dataset (Multidimensional Classification)")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.colorbar(label='Wine Class')
plt.show()
```