SINGLE LAYER PERCEPTRON

AND PROBLEM

CODE :-

```
import numpy as np
```

```
class Perceptron(object):
    """Implements a perceptron network"""
    def __init__(self, input_size, lr=1, epochs=100):
        #initializing with random number
        self.W = np.random.rand(input_size+1)*0.1-0.05
        # add one for bias

        self.epochs = epochs
        self.lr = lr

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

    def predict(self, x):
        z = self.W.dot(x)
        a = self.activation_fn(z)
        return a

    def fit(self, X, d):
        for _ in range(self.epochs):
            for i in range(d.shape[0]):
                x = np.insert(X[i], 0, 1)
                y = self.predict(x)
                e = d[i] - y
                self.W = self.W + self.lr * e * x
```

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

```
d = np.array([0, 0, 0, 1])
perceptron = Perceptron(input_size=2, lr=0.3, epochs=100)
perceptron.fit(X, d)
print(perceptron.W)
```

```
[-0.91971726  0.62887288  0.32210451]
```

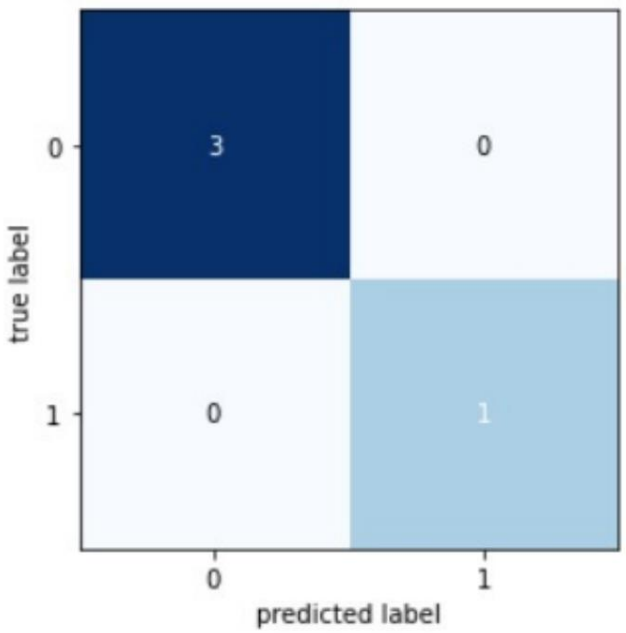
```
for i in X:
    val = perceptron.predict(np.insert(i, 0, 1))
    print("X=%d, Y=%d, output=" %(i[0], i[1]), val)
```

```
X=0, Y=0, output= 0
X=0, Y=1, output= 0
X=1, Y=0, output= 0
X=1, Y=1, output= 1
```

```
import matplotlib.pyplot as plt
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

y_true = d
y_pred = [0,0,0,1]

output = np.array(confusion_matrix(y_true, y_pred))
print(output)

# outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(y_true,y_pred,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)

# classification report for precision, recall f1-score and accuracy
matrix = classification_report(y_true,y_pred,labels=[1,0])
print('Classification report : \n',matrix)
```

```
[[3 0]
 [0 1]]
Outcome values :
 1 0 0 3
Classification report :
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         1
           0       1.00      1.00      1.00         3

    accuracy                           1.00         4
   macro avg       1.00      1.00      1.00         4
weighted avg       1.00      1.00      1.00         4
```

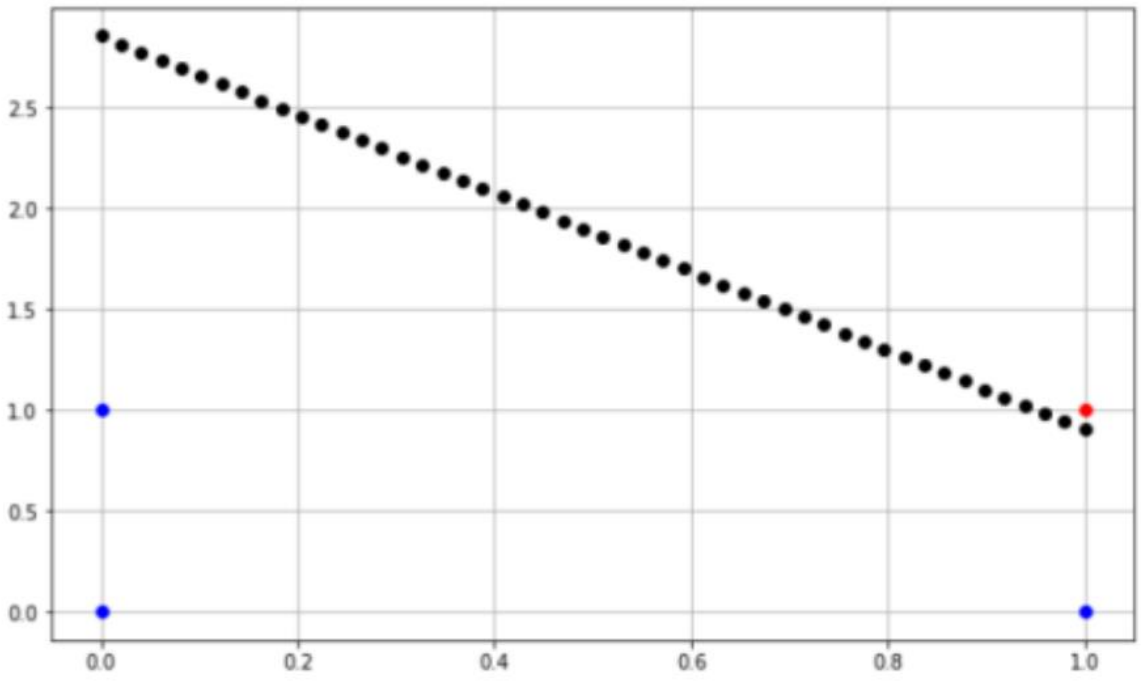
CONFUSION MATRIX:-



DECISION BOUNDARY:-

```
def plot_data(inputs,targets,weights):
    # fig config
    plt.figure(figsize=(10,6))
    plt.grid(True)

    #plot input samples(2D data points) and i have two classes.
    #one is +1 and second one is -1, so it red color for +1 and blue color for -1
    for input,target in zip(inputs,targets):
        plt.plot(input[0],input[1],'ro' if (target == 1.0) else 'bo')

    # Here i am calculating slope and intercept with given three weights
    for i in np.linspace(np.amin(inputs[:,:1]),np.amax(inputs[:,:1])):
        slope = -(weights[0]/weights[2])/(weights[0]/weights[1])
        intercept = -weights[0]/weights[2]

        #y =mx+c, m is slope and c is intercept
        y = (slope*i) + intercept
        plt.plot(i, y,'ko')
```

```
plot_data(X, d, perceptron.W)
```



The performance measure is accuracy and this performs 100%.

OR PROBLEM

CODE :-

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

d = np.array([0, 1, 1, 1])
perceptron = Perceptron(input_size=2, lr=0.3, epochs=50)
perceptron.fit(X, d)
print(perceptron.W)
```

```
[-0.26331235  0.33238228  0.31406196]
```

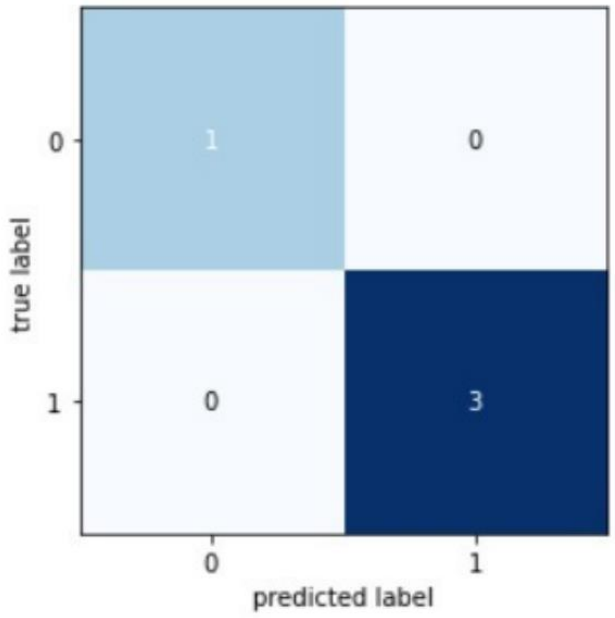
```
for i in X:
    val = perceptron.predict(np.insert(i, 0, 1))
    print("X=%d, Y=%d, output=" %(i[0], i[1]), val)
```

```
X=0, Y=0, output= 0
X=0, Y=1, output= 1
X=1, Y=0, output= 1
X=1, Y=1, output= 1
```

```
import matplotlib.pyplot as plt
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

y_true = d
y_pred = [0,1,1,1]
```

```
output = np.array(confusion_matrix(y_true, y_pred))
print(output)

# outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(y_true,y_pred,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)

# classification report for precision, recall f1-score and accuracy
matrix = classification_report(y_true,y_pred,labels=[1,0])
print('Classification report : \n',matrix)
```

```
[[1 0]
 [0 3]]
Outcome values :
 3 0 0 1
Classification report :
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         3
           0       1.00      1.00      1.00         1

    accuracy                           1.00         4
   macro avg       1.00      1.00      1.00         4
weighted avg       1.00      1.00      1.00         4
```



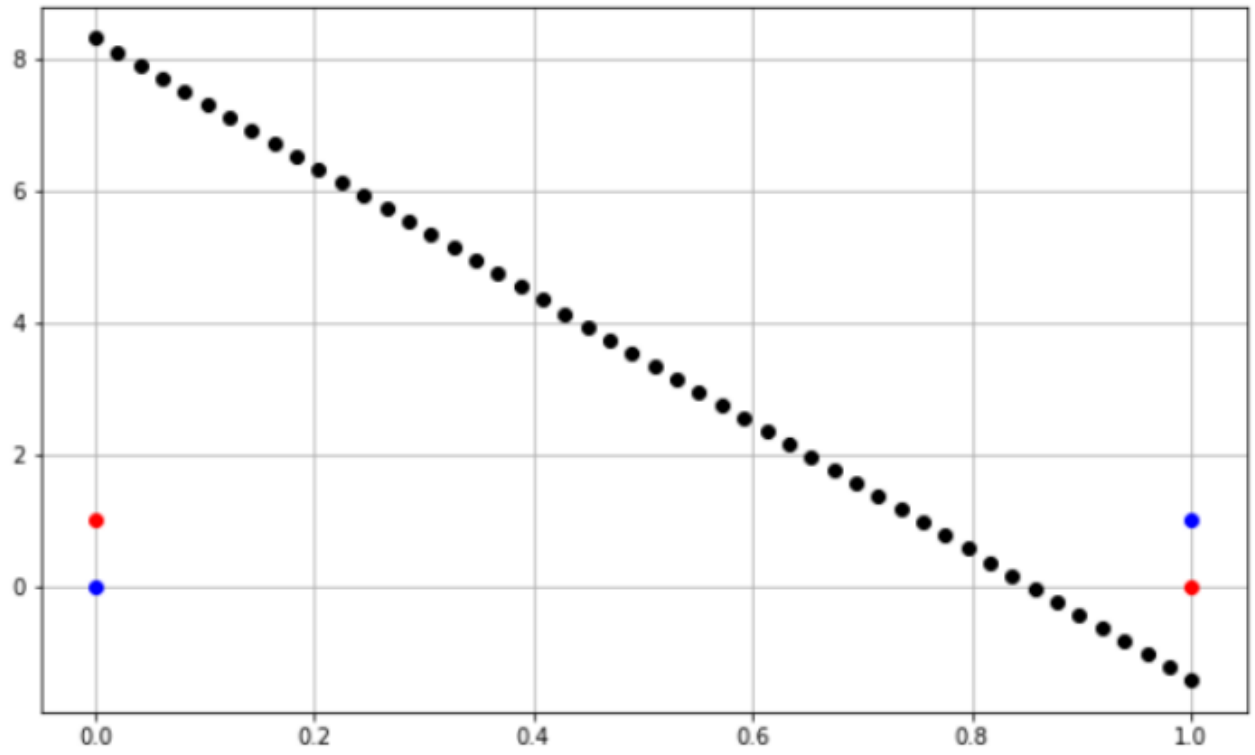
Performance measure is accuracy, and this performs 100%

XOR PROBLEM

DECISION BOUNDARY:-

```
def plot_data(inputs,targets,weights):
    # fig config
    plt.figure(figsize=(10,6))
    plt.grid(True)

    #plot input samples(2D data points) and i have two classes.
    #one is +1 and second one is -1, so it red color for +1 and blue color for -1
    for input,target in zip(inputs,targets):
        plt.plot(input[0],input[1],'ro' if (target == 1.0) else 'bo')

    # Here i am calculating slope and intercept with given three weights
    for i in np.linspace(np.amin(inputs[:,:1]),np.amax(inputs[:,:1])):
        slope = -(weights[0]/weights[2])/(weights[0]/weights[1])
        intercept = -weights[0]/weights[2]

        #y =mx+c, m is slope and c is intercept
        y = (slope*i) + intercept
        plt.plot(i, y,'ko')
```

```
plot_data(X, d, perceptron.W)
```



Performance Measures:- accuracy (50%)

Since perceptrons are limited to solving problems that are linearly separable. Two classes are linearly separable means that we can draw a single line to separate the two classes. We can do this easily for the AND and OR gates, but there is no single line that can separate the classes for the XOR gate. This means that we can't use our single-layer perceptron to model an XOR gate.