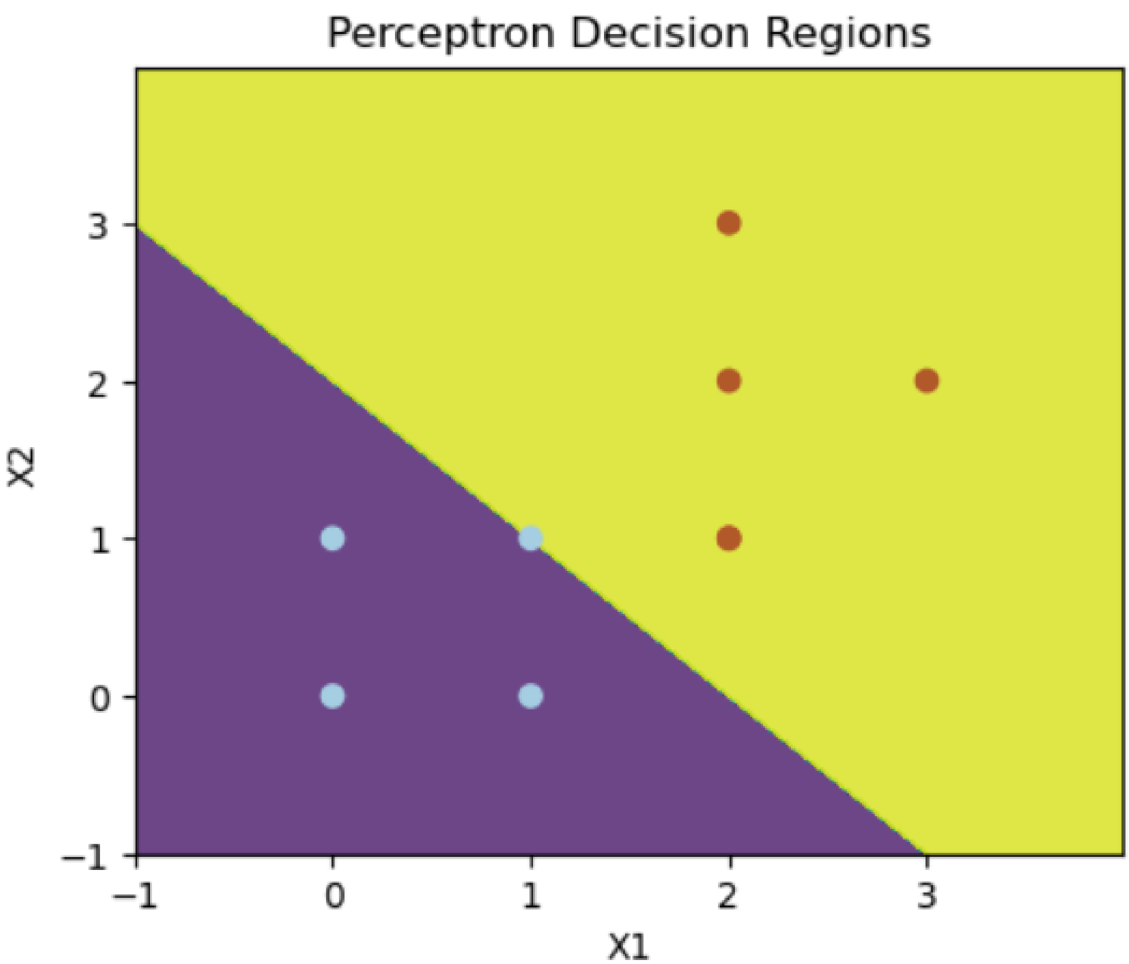
Name : Shreya Kakade

Class:TE-A

Rollno:TEAD22541

Practical NO:4

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt

# New input data
X = np.array([[0, 0], [1, 0], [0, 1], [1, 1],[2,1], [2, 2], [2, 3],[3, 2], [2, 1

# Labels: +1 if above the line x1 + x2 > 1, else -1
Y = np.array([-1, -1, -1, -1, 1, 1, 1, 1,1])

# Initialize weights and bias
w = np.zeros(X.shape[1])
b = 0

# Training process
learning_rate = 0.3
for _ in range(6):
    for i in range(X.shape[0]):
        # Predict using the sign function
        y_pred = np.sign(np.dot(X[i], w) + b)

        # If the prediction is wrong, update weights and bias
        if y_pred != Y[i]:
            w += learning_rate * Y[i] * X[i]
            b += learning_rate * Y[i]

# Create a grid for plotting decision regions
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))

# Calculate the decision boundary over the grid
Z = np.sign(np.dot(np.c_[xx.ravel(), yy.ravel()], w) + b)
Z = Z.reshape(xx.shape)

plt.figure(figsize=(5, 4))  # Adjust width and height as needed
plt.contourf(xx, yy, Z, alpha=0.8)
plt.scatter(X[:, 0], X[:, 1], c=Y, cmap=plt.cm.Paired)
plt.xlabel('X1')
plt.ylabel('X2')
plt.title('Perceptron Decision Regions')
plt.show()
```



Perceptron Decision Regions
X2
X1
3
2
1
0
−1
−1
0
1
2
3