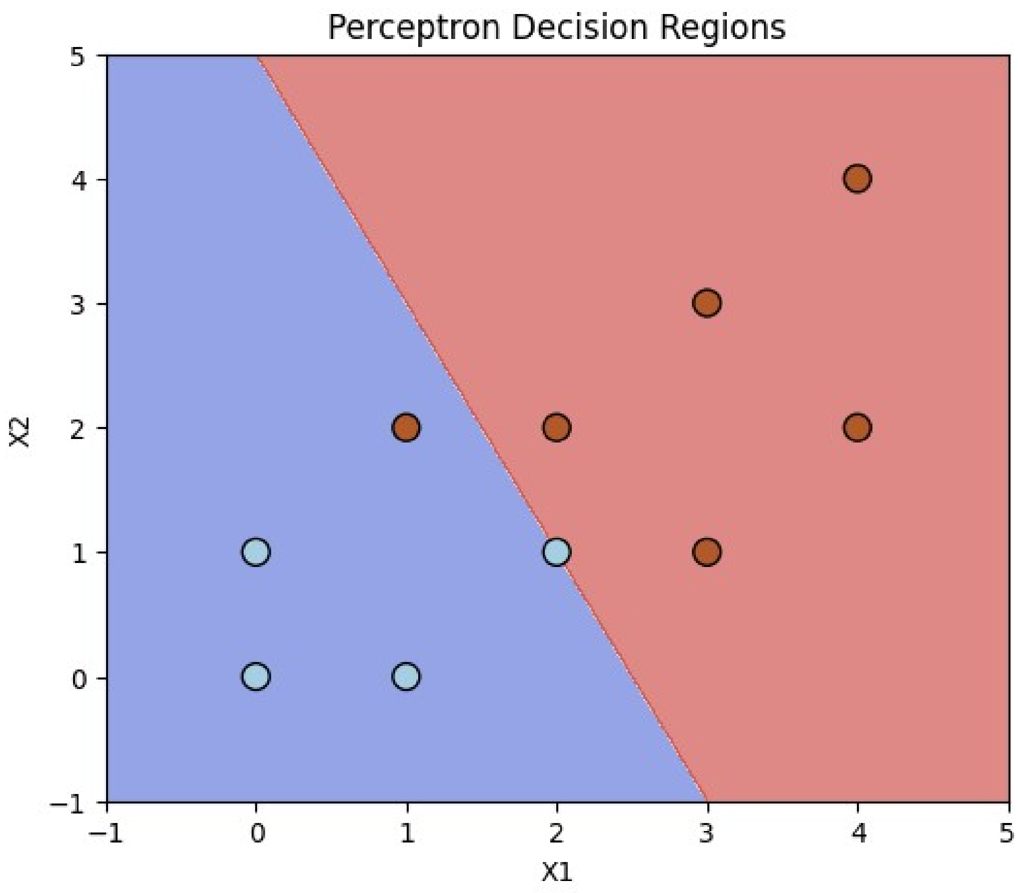
```
Name:Aarya Admane
RollNo:22630
B-B2
```

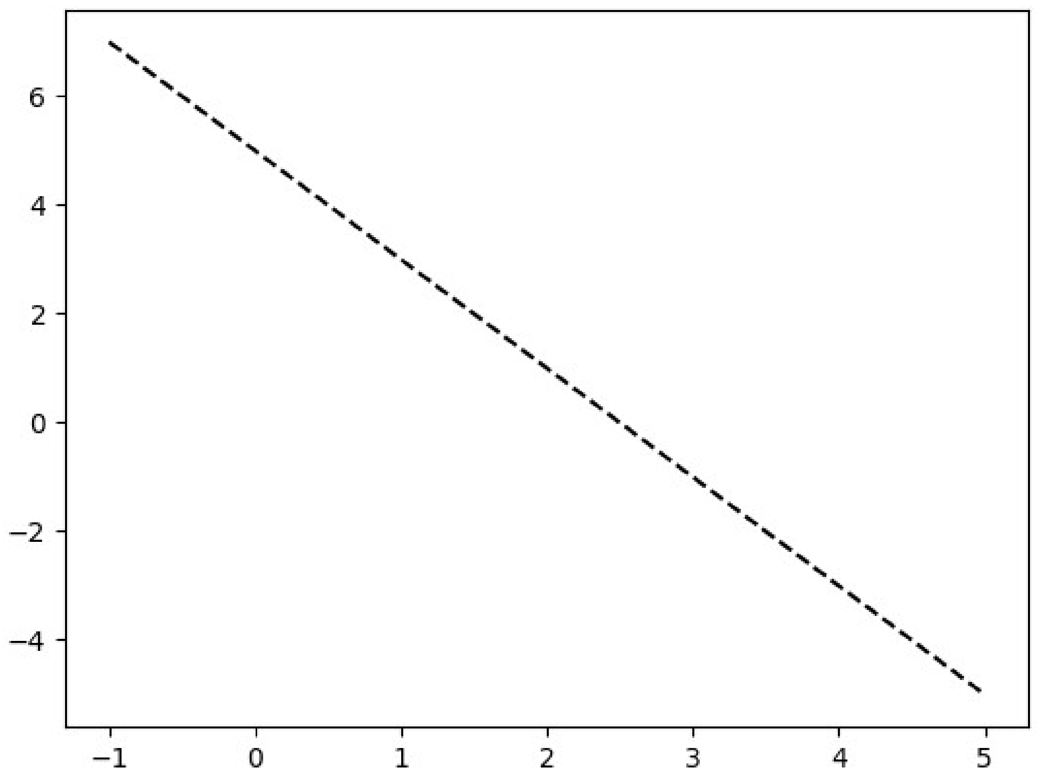
import numpy as np import matplotlib.pyplot as plt

```
# New input data (modified points)
X = np.array([[0, 0], [1, 0], [0, 1], [2, 2], [3, 3], [4, 4], [1, 2],
[2, 1], [3, 1], [4, 2]])

# Labels: +1 if above the line x1 + x2 > 2, else -1
Y = np.array([-1, -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]):
        y_pred = np.sign(np.dot(X[i], w) + b)
        if y_pred != Y[i]:  # Update rule
            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.linspace(x_min, x_max, 500),
                     np.linspace(y_min, y_max, 500))

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

# Plot decision regions
plt.figure(figsize=(6, 5))
plt.contourf(xx, yy, Z, alpha=0.6, cmap=plt.cm.coolwarm)
plt.scatter(X[:, 0], X[:, 1], c=Y, cmap=plt.cm.Paired, edgecolors='k',
s=100)
plt.xlabel('X1')
plt.ylabel('X2')
plt.title('Perceptron Decision Regions')

Text(0.5, 1.0, 'Perceptron Decision Regions')
```



```
# Plot decision boundary line
x_line = np.linspace(x_min, x_max, 100)
y_line = -(w[0] * x_line + b) / w[1] if w[1] != 0 else
np.full_like(x_line, -b/w[0])
plt.plot(x_line, y_line, 'k--', label='Decision Boundary')
```

```
[<matplotlib.lines.Line2D at 0x286ac1d1400>]
```



```
plt.show()
```