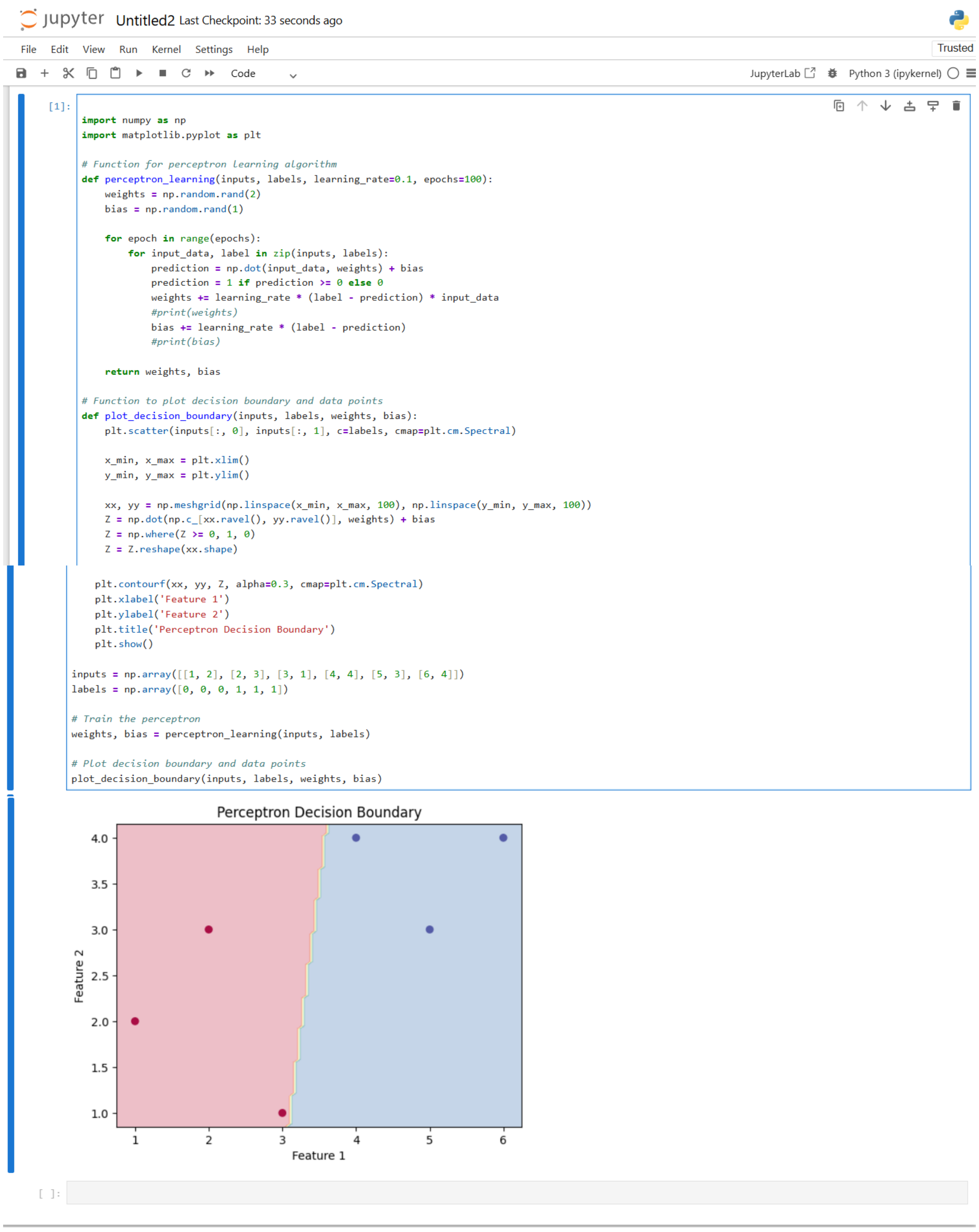
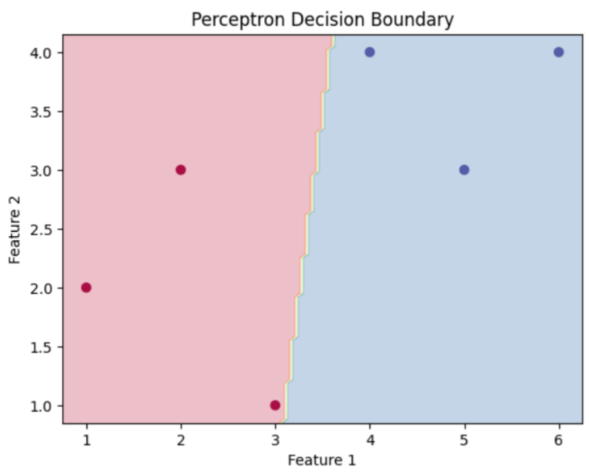


Untitled2 Last Checkpoint: 33 seconds ago

[1]:

```python
import numpy as np
import matplotlib.pyplot as plt

# Function for perceptron learning algorithm
def perceptron_learning(inputs, labels, learning_rate=0.1, epochs=100):
    weights = np.random.rand(2)
    bias = np.random.rand(1)

    for epoch in range(epochs):
        for input_data, label in zip(inputs, labels):
            prediction = np.dot(input_data, weights) + bias
            prediction = 1 if prediction >= 0 else 0
            weights += learning_rate * (label - prediction) * input_data
            #print(weights)
            bias += learning_rate * (label - prediction)
            #print(bias)

    return weights, bias

# Function to plot decision boundary and data points
def plot_decision_boundary(inputs, labels, weights, bias):
    plt.scatter(inputs[:, 0], inputs[:, 1], c=labels, cmap=plt.cm.Spectral)

    x_min, x_max = plt.xlim()
    y_min, y_max = plt.ylim()

    xx, yy = np.meshgrid(np.linspace(x_min, x_max, 100), np.linspace(y_min, y_max, 100))
    Z = np.dot(np.c_[xx.ravel(), yy.ravel()], weights) + bias
    Z = np.where(Z >= 0, 1, 0)
    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, alpha=0.3, cmap=plt.cm.Spectral)
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.title('Perceptron Decision Boundary')
    plt.show()

inputs = np.array([[1, 2], [2, 3], [3, 1], [4, 4], [5, 3], [6, 4]])
labels = np.array([0, 0, 0, 1, 1, 1])

# Train the perceptron
weights, bias = perceptron_learning(inputs, labels)

# Plot decision boundary and data points
plot_decision_boundary(inputs, labels, weights, bias)
```



[ ]: