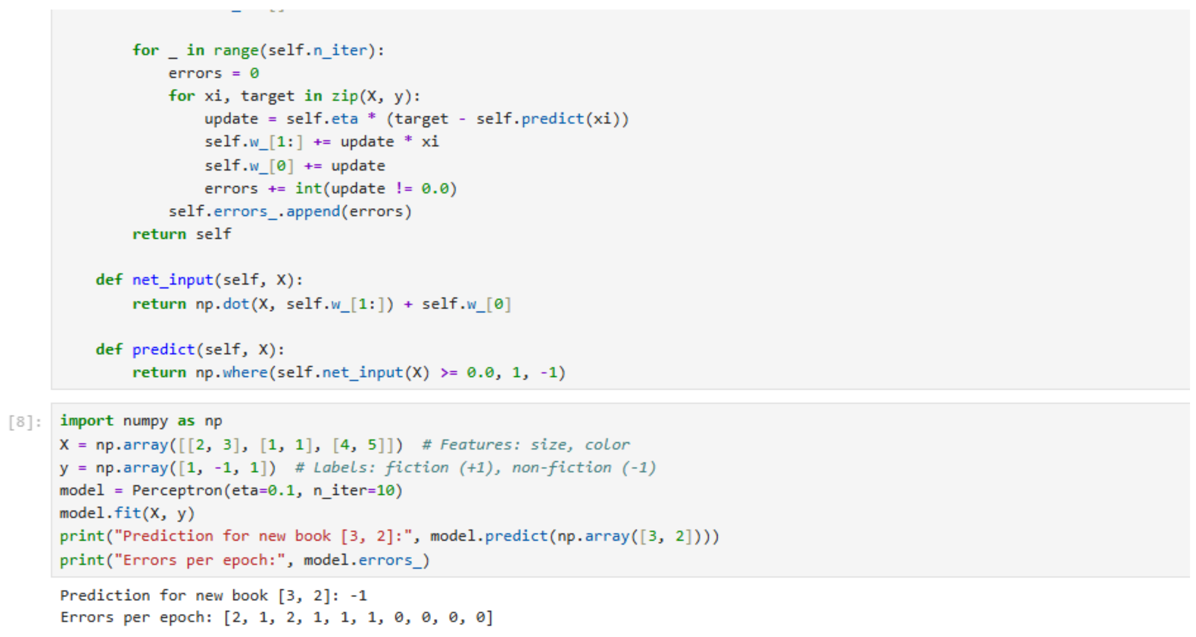
Lab-2-assignment- Perceptron_Lab_Exercise (1)

EXERCISE 1



```
        for _ in range(self.n_iter):
            errors = 0
            for xi, target in zip(X, y):
                update = self.eta * (target - self.predict(xi))
                self.w_[1:] += update * xi
                self.w_[0] += update
                errors += int(update != 0.0)
            self.errors_.append(errors)
        return self

    def net_input(self, X):
        return np.dot(X, self.w_[1:]) + self.w_[0]

    def predict(self, X):
        return np.where(self.net_input(X) >= 0.0, 1, -1)
```

```
[8]: import numpy as np
X = np.array([[2, 3], [1, 1], [4, 5]])  # Features: size, color
y = np.array([1, -1, 1])  # Labels: fiction (+1), non-fiction (-1)
model = Perceptron(eta=0.1, n_iter=10)
model.fit(X, y)
print("Prediction for new book [3, 2]:", model.predict(np.array([3, 2])))
print("Errors per epoch:", model.errors_)
```

```
Prediction for new book [3, 2]: -1
Errors per epoch: [2, 1, 2, 1, 1, 1, 0, 0, 0, 0]
```

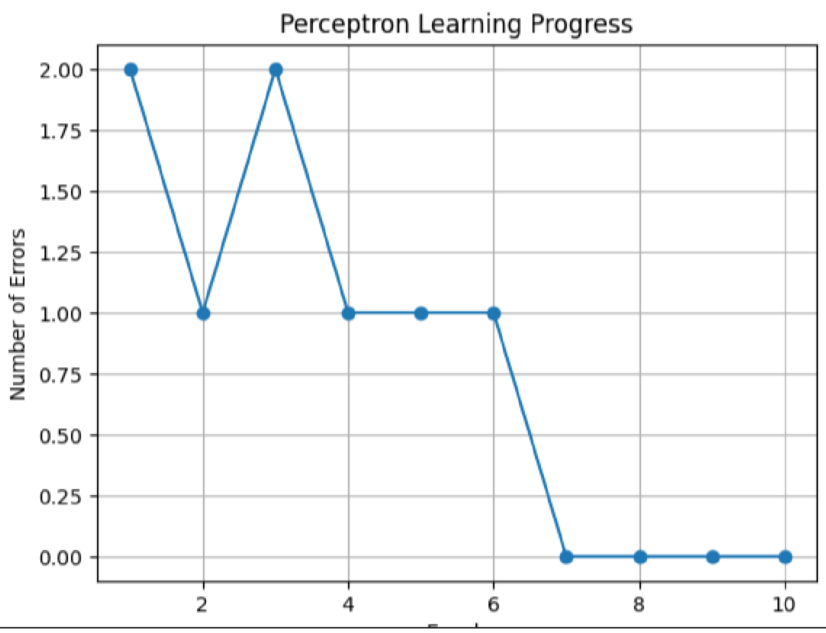
1. What does the prediction -1 mean for the book [3, 2]? (Hint: Think about the labels in y.)
   - **It means Perceptron classified the book as non-fiction.**

2. Look at the errors list. How many total errors did the Perceptron make across all 10 epochs? (Sum the numbers.)
   - **2 + 1 + 2 + 1 + 1 + 1 + 0 + 0 + 0 + 0 = 8 errors total**

3. Why do the errors drop to 0 by epoch 7? What does this tell you about the data?
   - **Perceptron has found a perfect decision boundary that separates both the fiction and non-fiction books correctly, so no more updates are needed.**

EXERCISE 2

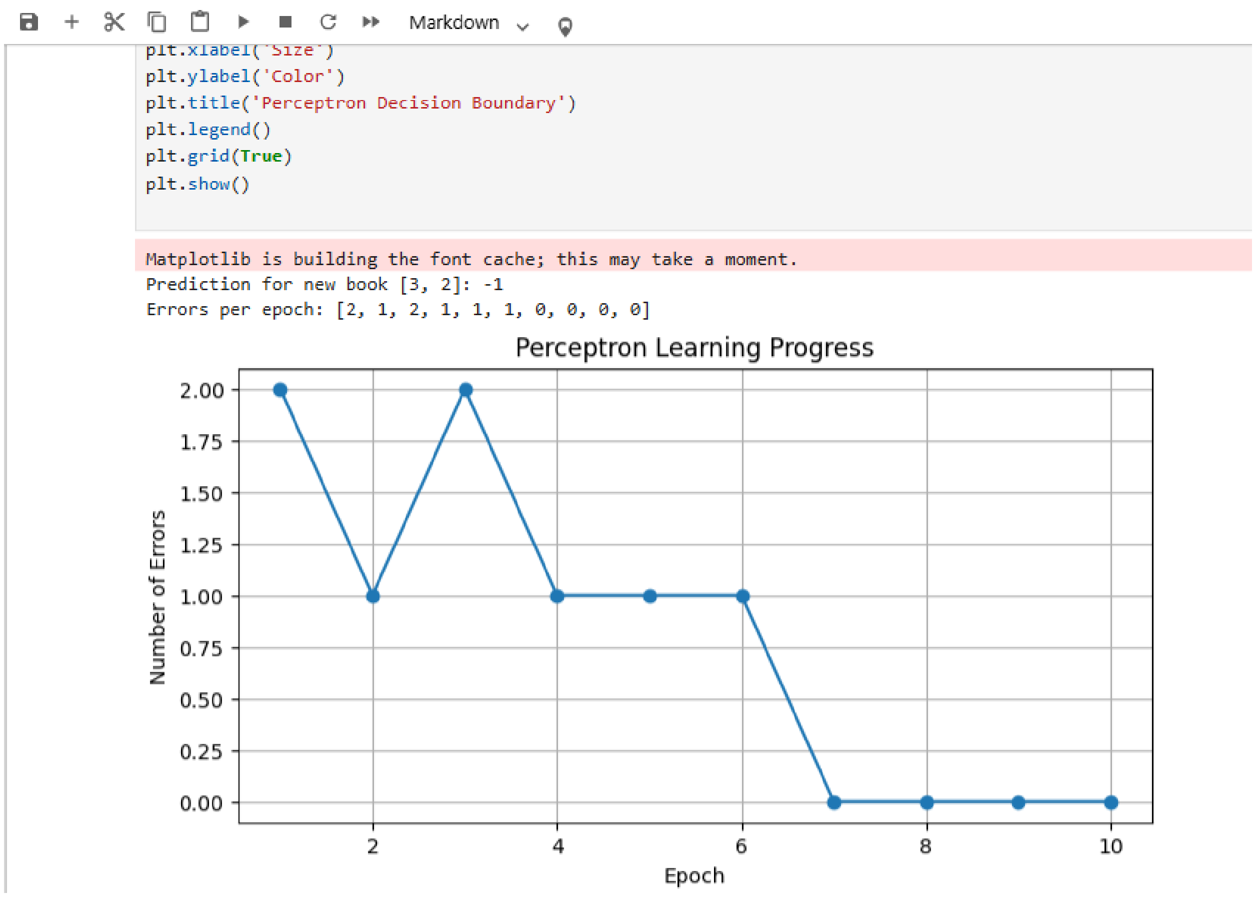
```
[9]: import matplotlib.pyplot as plt
     plt.plot(range(1, len(model.errors_) + 1), model.errors_, marker='o')
     plt.xlabel('Epoch')
     plt.ylabel('Number of Errors')
     plt.title('Perceptron Learning Progress')
     plt.grid(True)
     plt.show()
```



1. Why do the errors fluctuate (e.g., 2, 1, 2, 1) before reaching 0?
   - **The model is still modifying the weights and because updates are made for each individual sample, the errors may rise temporarily if previous updates alter the weights in an unfavorable way.**

2. What does it mean when errors reach 0? (Hint: Think about the librarian's sorting rule.)
   - **Perceptron has effectively learned a rule that accurately classifies all training samples.**

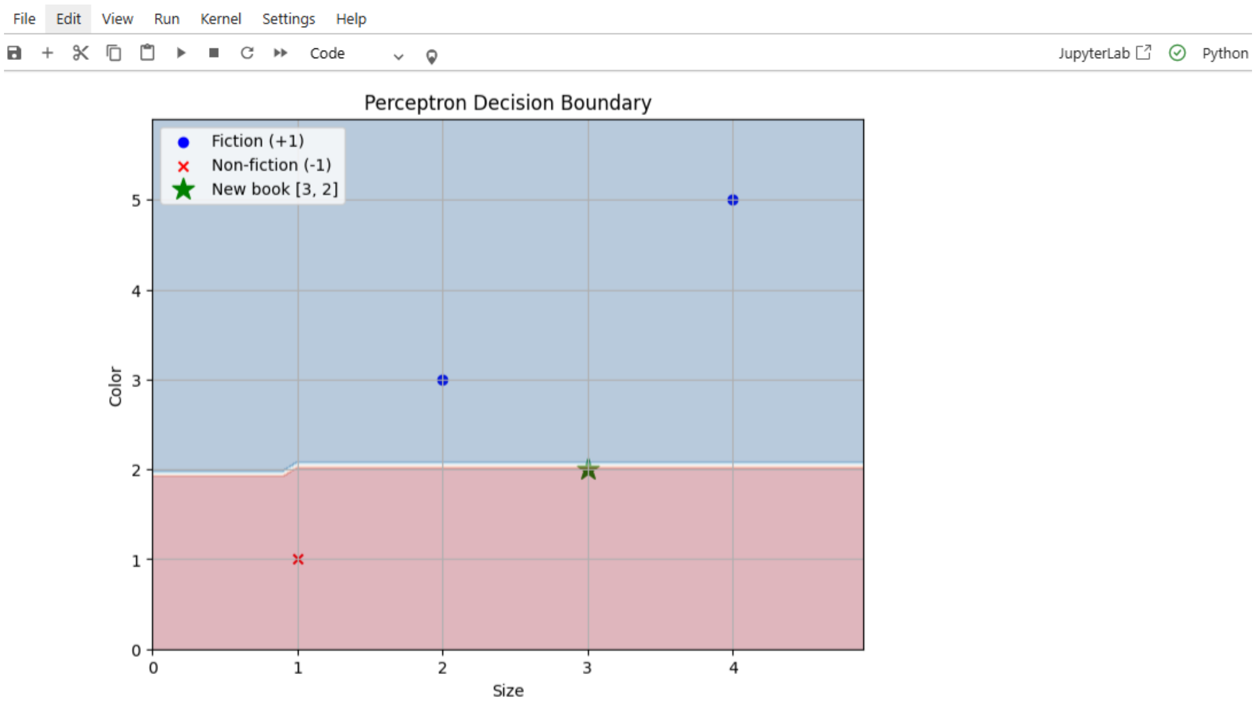
EXERCISE 3



```
plt.xlabel('Size')
plt.ylabel('Color')
plt.title('Perceptron Decision Boundary')
plt.legend()
plt.grid(True)
plt.show()
```

```
Matplotlib is building the font cache; this may take a moment.
Prediction for new book [3, 2]: -1
Errors per epoch: [2, 1, 2, 1, 1, 1, 0, 0, 0, 0]
```

1. Where is the new book [3, 2] located relative to the decision boundary? Does this explain the -1 prediction?



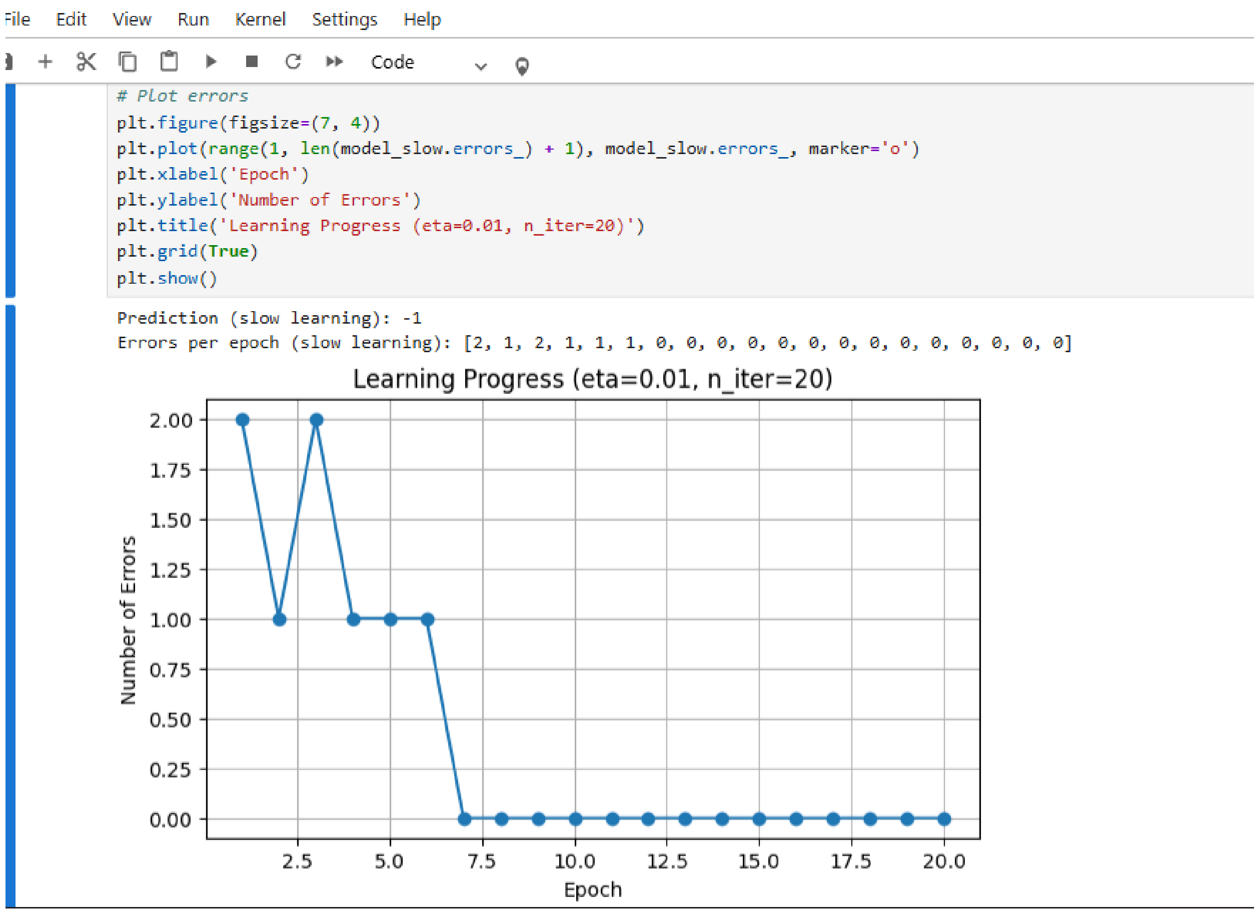
- ➢ **It lies in the non-fiction region (red), so yes it explains the prediction.**

2. How does the decision boundary separate the fiction and non-fiction books?
   - ➢ **Separates them vertically based on the weighted combination of size and color.**

3. If you move the new book to [4, 4], what prediction would you expect? Why?
   - ➢ **Fiction, because it falls in the blue region above the decision boundary**

EXERCISE 4

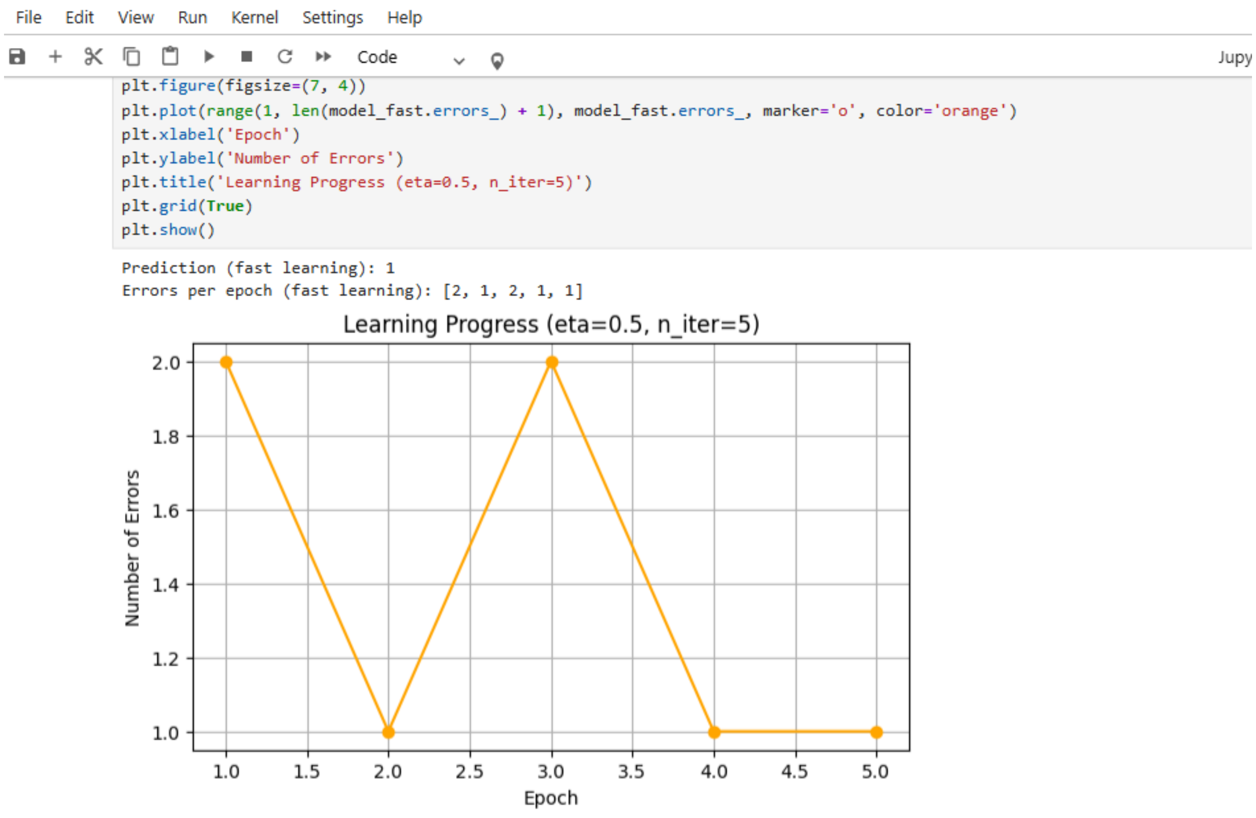
1. How does changing eta affect the errors list? (Compare the speed of learning.)



```
# Plot errors
plt.figure(figsize=(7, 4))
plt.plot(range(1, len(model_slow.errors_) + 1), model_slow.errors_, marker='o')
plt.xlabel('Epoch')
plt.ylabel('Number of Errors')
plt.title('Learning Progress (eta=0.01, n_iter=20)')
plt.grid(True)
plt.show()
```

```
Prediction (slow learning): -1
Errors per epoch (slow learning): [2, 1, 2, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

- **With eta = 0.01, the model takes longer to converge (more epochs with errors)**


File Edit View Run Kernel Settings Help
Code
Jupy

```
plt.figure(figsize=(7, 4))
plt.plot(range(1, len(model_fast.errors_) + 1), model_fast.errors_, marker='o', color='orange')
plt.xlabel('Epoch')
plt.ylabel('Number of Errors')
plt.title('Learning Progress (eta=0.5, n_iter=5)')
plt.grid(True)
plt.show()
```

```
Prediction (fast learning): 1
Errors per epoch (fast learning): [2, 1, 2, 1, 1]
```

Learning Progress (eta=0.5, n_iter=5)
Number of Errors
Epoch
2.0
1.8
1.6
1.4
1.2
1.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

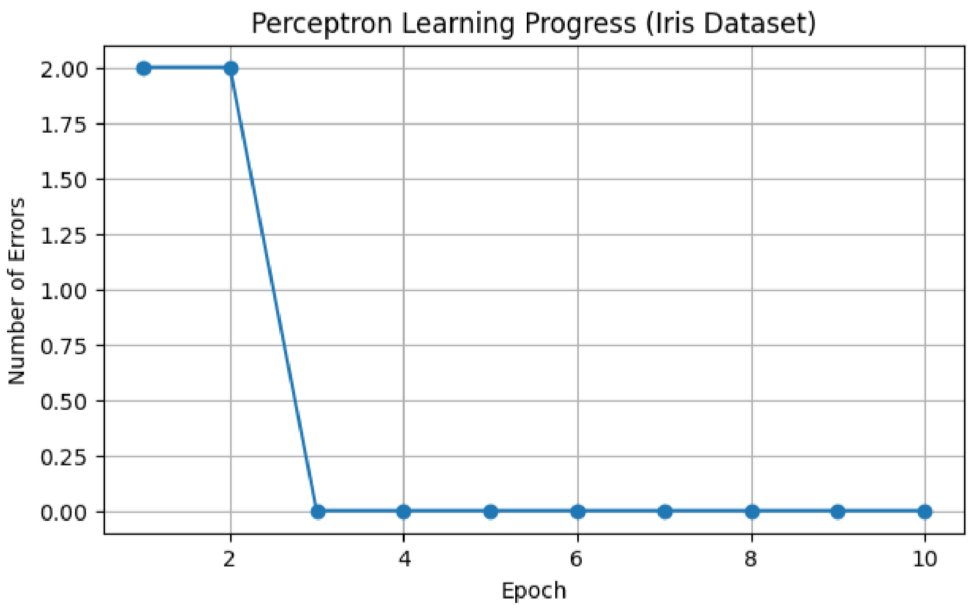
- **With eta = 0.5, the model converges faster—errors drop to 0 after just a few epochs**

2. How does changing n_iter affect the results? Did fewer epochs still reach 0 errors?
   - **Yes, fewer epochs reached zero, n_iter = 5 was enough to reach 0 errors when learning rate was high (eta = 0.5).**

3. Did the prediction for [3, 2] change with different settings? Why or why not?
   - **Stayed the same due to the model converging to a similar decision boundary.**

EXERCISE 5

[11]:
```python
# Plot errors per epoch
plt.figure(figsize=(7, 4))
plt.plot(range(1, len(model.errors_) + 1), model.errors_, marker='o')
plt.xlabel('Epoch')
plt.ylabel('Number of Errors')
plt.title('Perceptron Learning Progress (Iris Dataset)')
plt.grid(True)
plt.show()
```

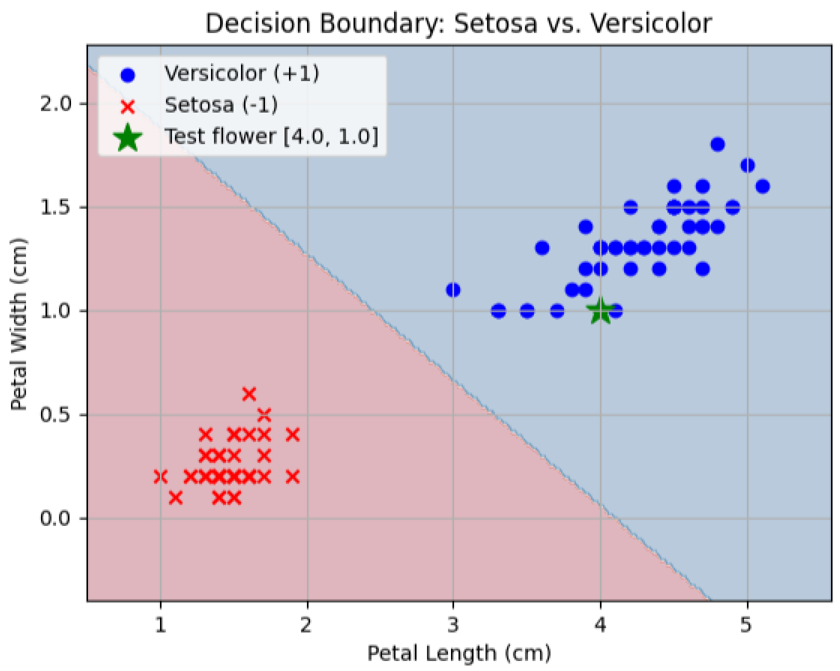


```python
# Plot decision boundary
x1_min, x1_max = X[:, 0].min() - 0.5, X[:, 0].max() + 0.5
x2_min, x2_max = X[:, 1].min() - 0.5, X[:, 1].max() + 0.5
xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, 0.02),
                       np.arange(x2_min, x2_max, 0.02))
Z = model.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
Z = Z.reshape(xx1.shape)

plt.contourf(xx1, xx2, Z, alpha=0.3, cmap='RdBu')

# Plot the data points
plt.scatter(X[y == 1][:, 0], X[y == 1][:, 1], color='blue', marker='o', label='Versicolor (+1)')
plt.scatter(X[y == -1][:, 0], X[y == -1][:, 1], color='red', marker='x', label='Setosa (-1)')
plt.scatter(test_flower[0][0], test_flower[0][1], color='green', marker='*', s=200, label='Test flower [4.0, 1.0]')

plt.xlabel('Petal Length (cm)')
plt.ylabel('Petal Width (cm)')
plt.title('Decision Boundary: Setosa vs. Versicolor')
plt.legend()
plt.grid(True)
plt.show()
```



1. What does the prediction mean (Setosa or Versicolor)? (Check the Iris dataset labels.)
    - **Versicolor**

2. Does the errors list reach 0? Why or why not? (Hint: Is the Iris data linearly separable for these classes?)
    - **Yes, because Setosa and Versicolor are linearly separable based on petal dimensions.**

3. How does the decision boundary look on the Iris data compared to the book dataset?
    - **Looks similar, it too separating two classes in straight line, but in a different feature space (petal features instead of book size/color).**

BONUS CHALLENGE

1. Does the prediction for [3, 2] change? Why?

- **No, even after adding [3, 4] fiction, the new decision boundary did not shift enough to include [3, 2] in the fiction region.**

CONCLUSION

- **Through the robot librarian analogy and the Perceptron lab, I learnt how a simple machine learning model can classify data by learning from mistakes. Every time Perceptron mistakenly classifies a book, it adjusts its decision-making criteria (weights), much like a librarian does while organizing books. It increasingly improves at clearly differentiating between fiction and non-fiction with epochs and eta. I also discovered how starting variables like random_state might affect learning, especially when there is ambiguity. By visualizing the learning process, I was able to see how the model develops, gradually increasing its accuracy.**