schulzd_lab02

September 21, 2020

1 Lab 02 - Decision Boundaries

David Schulz

1.1 Introduction

In this lab, we learn how decision boundaries, choice of features, and a training-testing split of the data affect the accuracy of a model's predictions. Problem 1 displays how the placement of a decision boundary will change how well it can separate the classes. Problem 2 shows how the features you choose to train with can affect its ability to separate the classes. Problem 3 proves that splitting the data into training/validation/testing subsets will allow the model's predictions to be more generalized and prevent overfitting.

1.2 Reflection Questions

1.2.1 Problem 1

1. Just by looking at your plot, which of the two decision boundaries does a better job of separating the two classes of points?
   - Decision boundaries 1 and 2
2. Which of the two decision boundaries gave the more accurate predictions?
   - Decision boundary 2
3. How does the accuracy metric seem to relate to the ability of the decision boundary to separate the classes?
   - The better the classes are correctly separated, the better the accuracy is.

1.2.2 Problem 2

1. For which pair of features are the classes more easily separated?
   - The petal lengths and widths
2. Just by looking at your plots, for which pair of features does the decision boundary do a better job of separating the classes?
   - The petal lengths and widths
3. Which decision boundary gives the most accurate predictions?
   - Decision boundary 2

4. How does the choice of features seem to impact the ability to make accurate predictions?
    - Some features will expose the differences between the classes better than others.

1.2.3 Problem 3

1. Compare the accuracies you calculated from the training and testing data sets. Predictions for which data set were evaluated as more accurate? Which accuracy score do you think is a more realistic representation of the performance of the model?
    - The training data set was evaluated as more accurate, but the testing data set is a more realistic representation because it isn't overfit.
2. Look at the scatter plot of the validation data points. Will the model make errors in predicting the labels? Why?
    - Yes because some setosa points are on the wrong side of the decision boundary.
3. Look at the scatter plots of the training and testing data points. Which data set is more representative of the validation data set? Which data set will demonstrate errors similar to the validation set?
    - The testing data set is more representative and will demonstrate similar errors.
4. Compare the accuracies you calculated from the training and testing data sets to the validation data set. Which accuracy calculation is more representative of the accuracy for the validation data set?
    - The accuracy of the testing data set is more representative.
5. Explain why training and evaluating a model on the same data set can be deceptive.
    - The model can "overfit" or memorize the exact points rather than make more generalized predictions.
6. Explain how dividing data into training and testing sets with no repeated points resolves some of the problems associated with training and evaluating a model on the same data set.
    - It's done to make sure that the model is accurately predicting with new points that it wasn't trained with.
7. Name 3 potential ramifications for publishing a model that was trained and characterized using the same data set.
    - It won't be able to make accurate predictions with new data, it will only memorize the data it was trained with, and its predictions won't be generalized enough.

1.3 1. Decision Boundaries and Evaluation of Model Predictions with Metrics

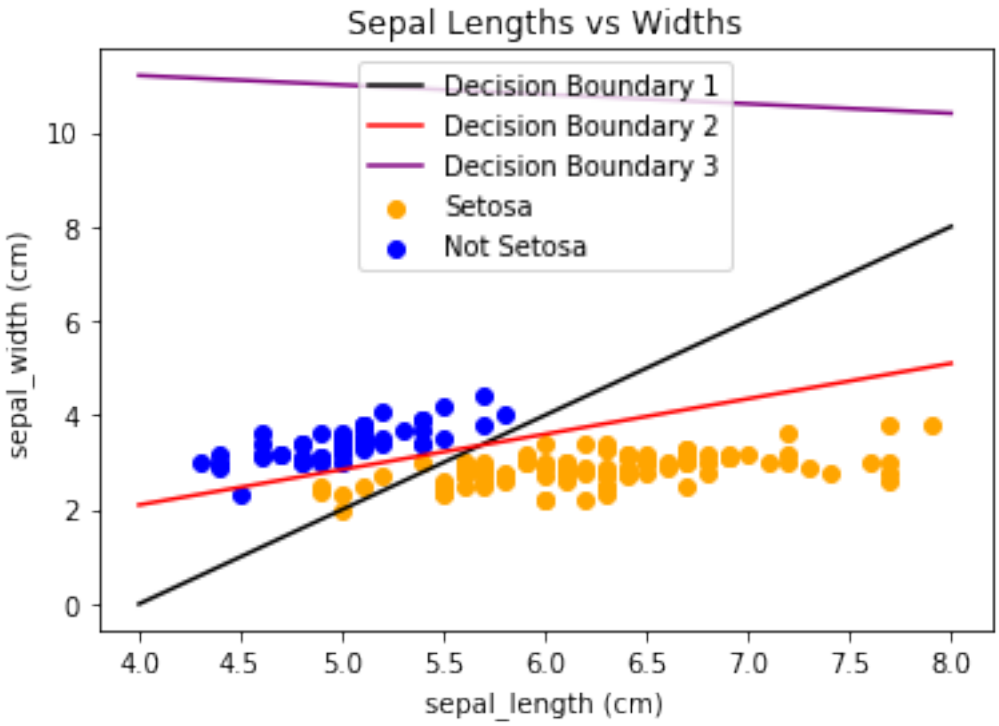
[1]:
```
import decision_boundaries as db
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn.metrics as sklm

setosa = pd.read_csv('setosa_data.csv')
print(setosa.head())
```

```
        label  sepal_length (cm)  sepal_width (cm)
0  not setosa                5.1               3.5
```

```
1  not setosa              4.9               3.0
2  not setosa              4.7               3.2
3  not setosa              4.6               3.1
4  not setosa              5.0               3.6
```

[2]:
```
setosas = setosa.loc[setosa['label'] == 'setosa']
non_setosas = setosa.drop(setosas.index)

set_lengths = setosas['sepal_length (cm)']
set_widths = setosas['sepal_width (cm)']
non_lengths = non_setosas['sepal_length (cm)']
non_widths = non_setosas['sepal_width (cm)']

plt.title('Sepal Lengths vs Widths')
plt.xlabel('sepal_length (cm)')
plt.ylabel('sepal_width (cm)')
plt.scatter(set_lengths, set_widths, c='orange', label='Setosa')
plt.scatter(non_lengths, non_widths, c='blue', label='Not Setosa')

x = np.array(range(4, 9))
plt.plot(x, 2*x-8, c='black', label='Decision Boundary 1')
plt.plot(x, 0.75*x-0.9, c='red', label='Decision Boundary 2')
plt.plot(x, -0.2*x+12, c='purple', label='Decision Boundary 3')
plt.legend()
```

[2]:
```
<matplotlib.legend.Legend at 0x7fc7369f0cd0>
```



Decision Boundary 1: 2x - y - 8 = 0

Decision Boundary 2: 0.75x - y - 0.9 = 0

Decision Boundary 3: -0.2x - y + 12 = 0

[3]:
```
dec_bound_vec = np.array([0.75, -1.00, -0.90])
features = setosa[["sepal_length (cm)", "sepal_width (cm)"]].values
true_labels = setosa["label"].values
pred_labels = db.linear_decision_boundary_classifier(dec_bound_vec, features,␣
 ↪true_labels, features)
print(pred_labels)
```

```
['not setosa' 'not setosa' 'not setosa' 'not setosa' 'not setosa'
 'not setosa' 'not setosa' 'not setosa' 'not setosa' 'not setosa'
 'not setosa' 'not setosa' 'not setosa' 'not setosa' 'not setosa'
 'not setosa' 'not setosa' 'not setosa' 'not setosa' 'not setosa'
 'not setosa' 'not setosa' 'not setosa' 'not setosa' 'not setosa'
 'not setosa' 'not setosa' 'not setosa' 'not setosa' 'not setosa'
 'not setosa' 'not setosa' 'not setosa' 'not setosa' 'not setosa'
 'not setosa' 'not setosa' 'not setosa' 'not setosa' 'not setosa'
 'not setosa' 'setosa' 'not setosa' 'not setosa' 'not setosa' 'not setosa'
 'not setosa' 'not setosa' 'not setosa' 'not setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa']
```

[4]:
```
sklm.accuracy_score(true_labels, pred_labels)
```

[4]: 0.9933333333333333

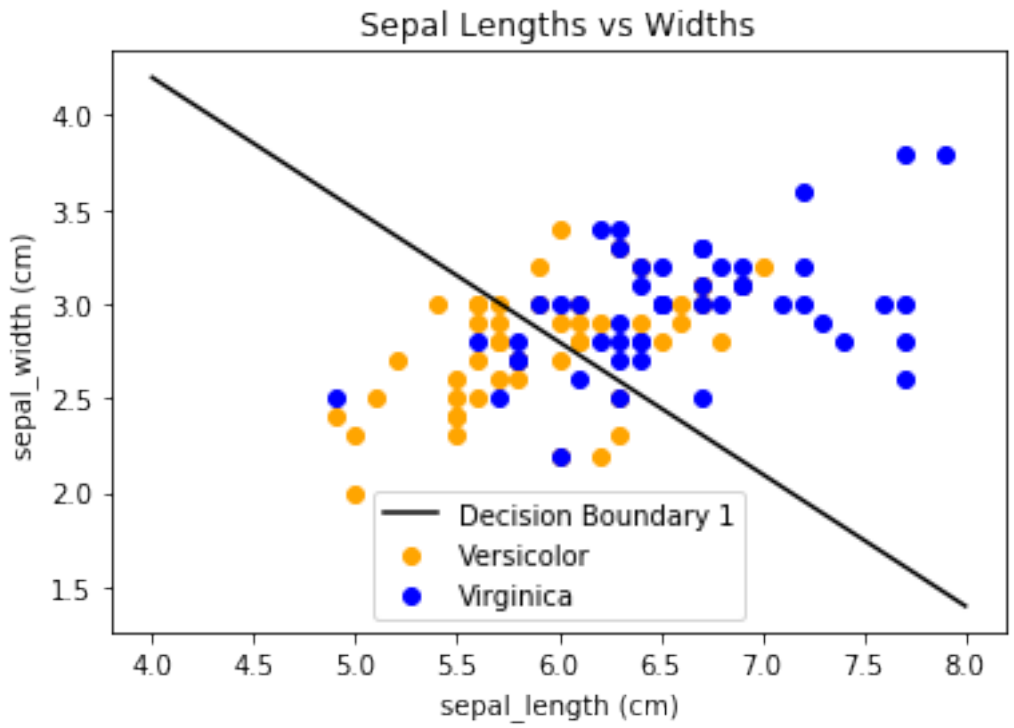
1.4 2. Predictive Power of Features

[5]:
```
vs_vg = pd.read_csv('versicolor_virginica_data.csv')
print(vs_vg.head())
```

```
        label  sepal_length (cm)  sepal_width (cm)  petal_length (cm)  \
0  versicolor                7.0               3.2                4.7
1  versicolor                6.4               3.2                4.5
2  versicolor                6.9               3.1                4.9
3  versicolor                5.5               2.3                4.0
4  versicolor                6.5               2.8                4.6

   petal_width (cm)
0               1.4
1               1.5
2               1.5
3               1.3
4               1.5
```

[6]:
```
vs = vs_vg.loc[vs_vg['label'] == 'versicolor']
vg = vs_vg.drop(vs.index)

vs_sepal_lengths = vs['sepal_length (cm)']
vs_sepal_widths = vs['sepal_width (cm)']
vg_sepal_lengths = vg['sepal_length (cm)']
vg_sepal_widths = vg['sepal_width (cm)']

plt.title('Sepal Lengths vs Widths')
plt.xlabel('sepal_length (cm)')
plt.ylabel('sepal_width (cm)')
plt.scatter(vs_sepal_lengths, vs_sepal_widths, c='orange', label='Versicolor')
plt.scatter(vg_sepal_lengths, vg_sepal_widths, c='blue', label='Virginica')

x = np.array(range(4, 9))
plt.plot(x, -0.7*x+7, c='black', label='Decision Boundary 1')
plt.legend()
```

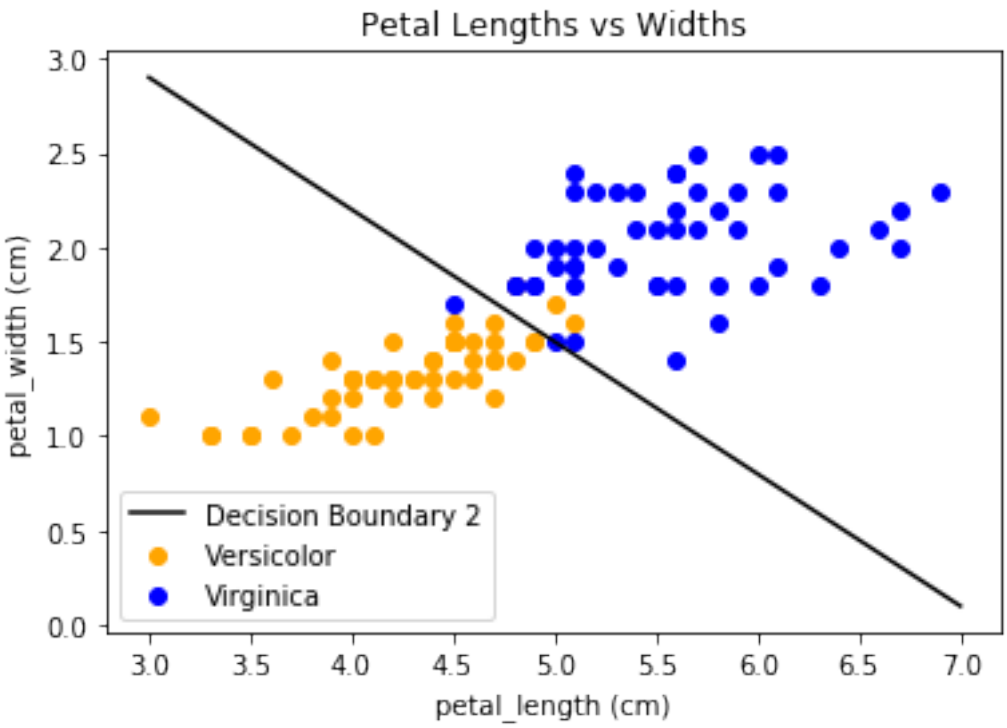
[6]:
```
<matplotlib.legend.Legend at 0x7fc736934e10>
```



Decision Boundary 1: -0.7x - y + 7 = 0

[7]:
```
vs_petal_lengths = vs['petal_length (cm)']
vs_petal_widths = vs['petal_width (cm)']
vg_petal_lengths = vg['petal_length (cm)']
vg_petal_widths = vg['petal_width (cm)']

plt.title('Petal Lengths vs Widths')
plt.xlabel('petal_length (cm)')
plt.ylabel('petal_width (cm)')
plt.scatter(vs_petal_lengths, vs_petal_widths, c='orange', label='Versicolor')
plt.scatter(vg_petal_lengths, vg_petal_widths, c='blue', label='Virginica')

x = np.array(range(3, 8))
plt.plot(x, -0.7*x+5, c='black', label='Decision Boundary 2')
plt.legend()
```

[7]:
```
<matplotlib.legend.Legend at 0x7fc7368bffd0>
```



Decision Boundary 2: -0.7x - y + 5 = 0

[8]:
```
dec_bound_vec = np.array([-0.7, -1.0, 7])
features = vs_vg[["sepal_length (cm)", "sepal_width (cm)"]].values
true_labels = vs_vg["label"].values
pred_labels = db.linear_decision_boundary_classifier(dec_bound_vec, features,␣
 ↪true_labels, features)
print(pred_labels)
sklm.accuracy_score(true_labels, pred_labels)
```

```
['virgnica' 'virgnica' 'virgnica' 'versicolor' 'virgnica' 'versicolor'
 'virgnica' 'versicolor' 'virgnica' 'versicolor' 'versicolor' 'virgnica'
 'versicolor' 'virgnica' 'versicolor' 'virgnica' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'virgnica' 'virgnica' 'versicolor' 'virgnica'
 'virgnica' 'virgnica' 'virgnica' 'virgnica' 'virgnica' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'virgnica' 'virgnica' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'virgnica' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'virgnica' 'versicolor' 'versicolor' 'virgnica' 'versicolor'
 'virgnica' 'virgnica' 'virgnica' 'virgnica' 'versicolor' 'virgnica'
 'virgnica' 'virgnica' 'virgnica' 'virgnica' 'virgnica' 'versicolor'
 'versicolor' 'virgnica' 'virgnica' 'virgnica' 'virgnica' 'versicolor'
 'virgnica' 'versicolor' 'virgnica' 'virgnica' 'virgnica' 'virgnica'
 'virgnica' 'virgnica' 'virgnica' 'virgnica' 'virgnica' 'virgnica'
```

```
 'virgnica' 'virgnica' 'versicolor' 'virgnica' 'virgnica' 'virgnica'
 'virgnica' 'virgnica' 'virgnica' 'virgnica' 'versicolor' 'virgnica'
 'virgnica' 'virgnica' 'versicolor' 'virgnica' 'virgnica' 'virgnica']
```

[8]:
```
0.7
```

[9]:
```
dec_bound_vec = np.array([-0.7, -1.0, 5])
features = vs_vg[["petal_length (cm)", "petal_width (cm)"]].values
true_labels = vs_vg["label"].values
pred_labels = db.linear_decision_boundary_classifier(dec_bound_vec, features,␣
 ↪true_labels, features)
print(pred_labels)
sklm.accuracy_score(true_labels, pred_labels)
```

```
['versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'virgnica' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'virgnica' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'virgnica' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'virgnica' 'virgnica' 'virgnica' 'virgnica' 'virgnica' 'virgnica'
 'versicolor' 'virgnica' 'virgnica' 'virgnica' 'virgnica' 'virgnica'
 'virgnica' 'virgnica' 'virgnica' 'virgnica' 'virgnica' 'virgnica'
 'virgnica' 'virgnica' 'virgnica' 'virgnica' 'virgnica' 'virgnica'
 'virgnica' 'virgnica' 'virgnica' 'virgnica' 'virgnica' 'virgnica'
 'virgnica' 'virgnica' 'virgnica' 'virgnica' 'virgnica' 'virgnica'
 'virgnica' 'virgnica' 'virgnica' 'virgnica' 'virgnica' 'virgnica'
 'virgnica' 'virgnica']
```

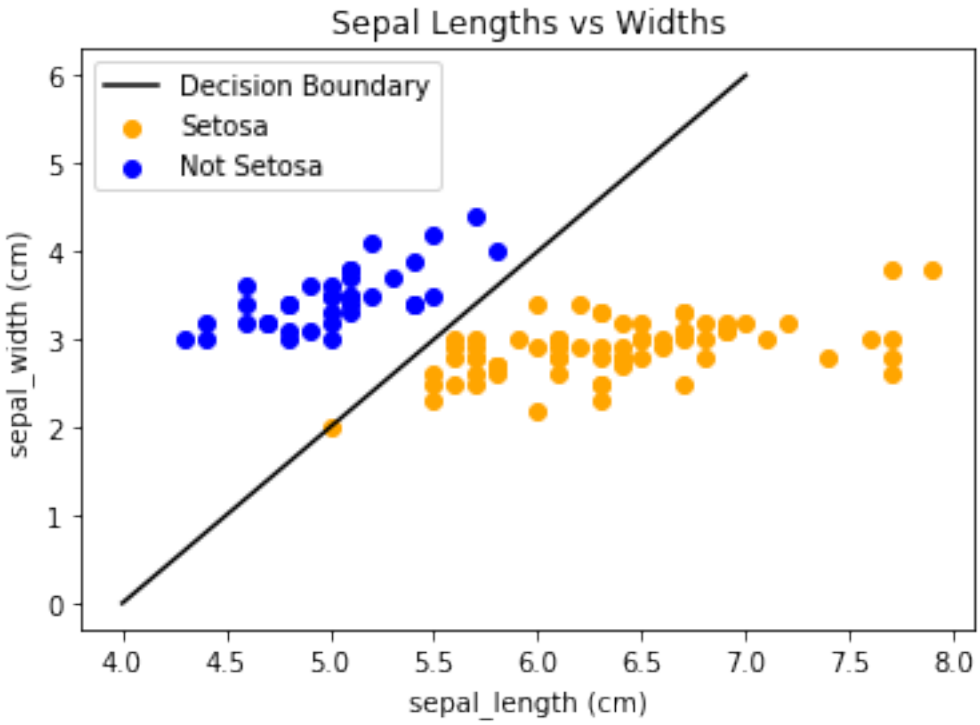
[9]:
```
0.96
```

1.5 3. Experimental Setup with Train-Test Splitting

[10]:
```
training = pd.read_csv('setosa_training.csv')

setosas = training.loc[training['label'] == 'setosa']
non_setosas = training.drop(setosas.index)

set_lengths = setosas['sepal_length (cm)']
set_widths = setosas['sepal_width (cm)']
non_lengths = non_setosas['sepal_length (cm)']
```

```
non_widths = non_setosas['sepal_width (cm)']

plt.title('Sepal Lengths vs Widths')
plt.xlabel('sepal_length (cm)')
plt.ylabel('sepal_width (cm)')
plt.scatter(set_lengths, set_widths, c='orange', label='Setosa')
plt.scatter(non_lengths, non_widths, c='blue', label='Not Setosa')

x = np.array(range(4, 8))
plt.plot(x, 2*x-8, c='black', label='Decision Boundary')
plt.legend()
```

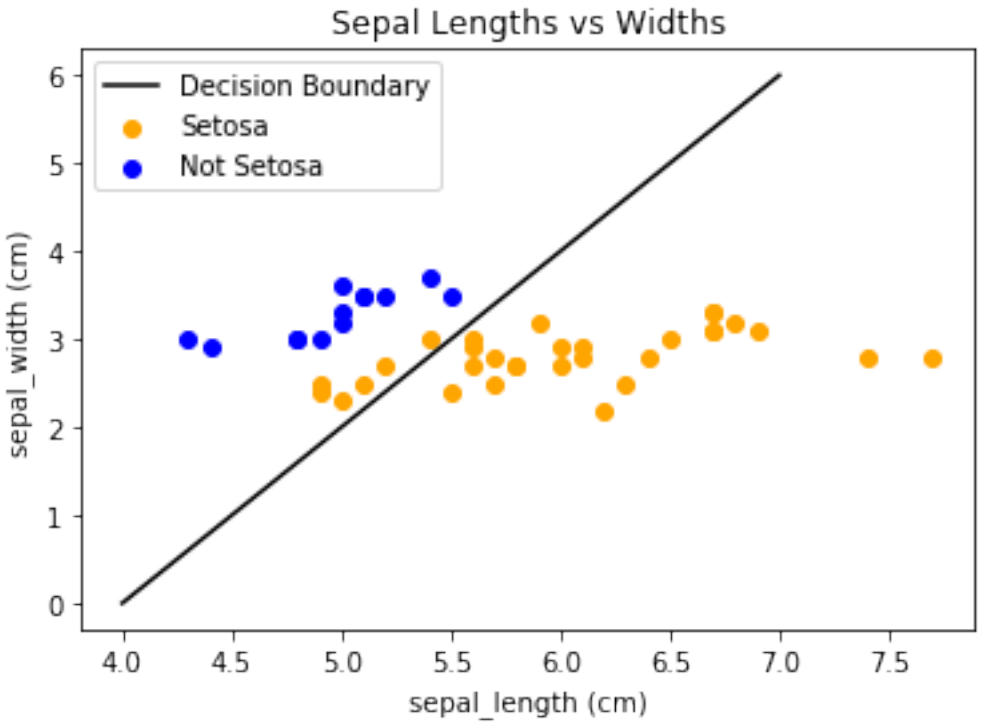
[10]: <matplotlib.legend.Legend at 0x7fc7367d5710>



[11]:
```
val = pd.read_csv('setosa_validation.csv')

setosas = val.loc[val['label'] == 'setosa']
non_setosas = val.drop(setosas.index)

set_lengths = setosas['sepal_length (cm)']
set_widths = setosas['sepal_width (cm)']
non_lengths = non_setosas['sepal_length (cm)']
non_widths = non_setosas['sepal_width (cm)']
```

```
plt.title('Sepal Lengths vs Widths')
plt.xlabel('sepal_length (cm)')
plt.ylabel('sepal_width (cm)')
plt.scatter(set_lengths, set_widths, c='orange', label='Setosa')
plt.scatter(non_lengths, non_widths, c='blue', label='Not Setosa')

x = np.array(range(4, 8))
plt.plot(x, 2*x-8, c='black', label='Decision Boundary')
plt.legend()
```

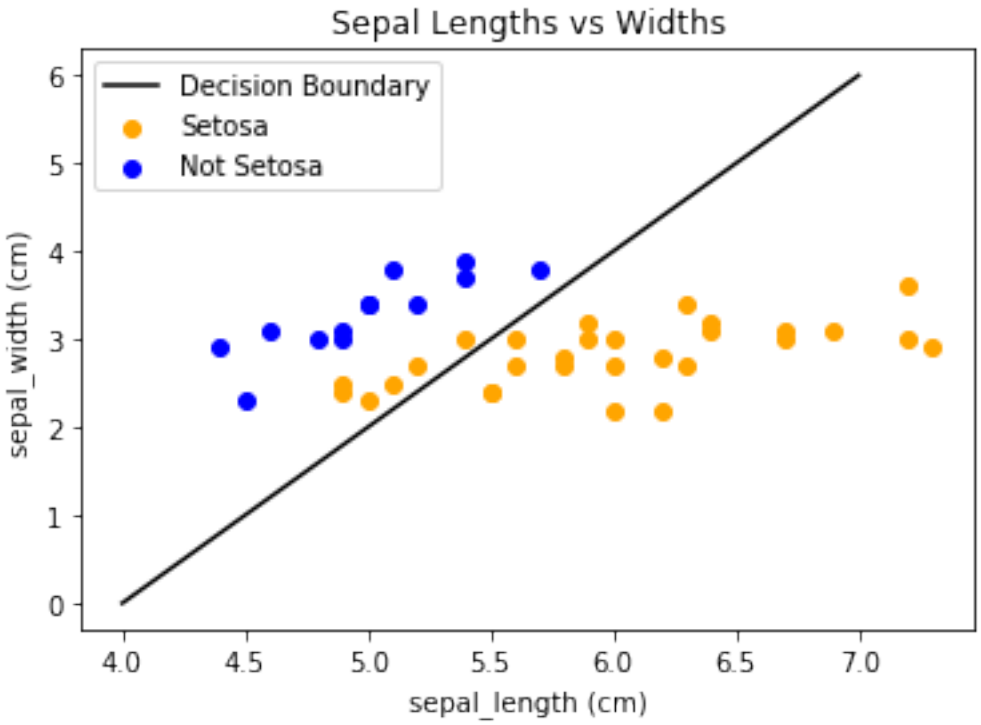
[11]: <matplotlib.legend.Legend at 0x7fc73675fa90>



[12]:
```
testing = pd.read_csv('setosa_testing.csv')

setosas = testing.loc[testing['label'] == 'setosa']
non_setosas = testing.drop(setosas.index)

set_lengths = setosas['sepal_length (cm)']
set_widths = setosas['sepal_width (cm)']
non_lengths = non_setosas['sepal_length (cm)']
non_widths = non_setosas['sepal_width (cm)']
```

```
plt.title('Sepal Lengths vs Widths')
plt.xlabel('sepal_length (cm)')
plt.ylabel('sepal_width (cm)')
plt.scatter(set_lengths, set_widths, c='orange', label='Setosa')
plt.scatter(non_lengths, non_widths, c='blue', label='Not Setosa')

x = np.array(range(4, 8))
plt.plot(x, 2*x-8, c='black', label='Decision Boundary')
plt.legend()
```

[12]: <matplotlib.legend.Legend at 0x7fc7366e3d10>



[13]:
```
dec_bound_vec = np.array([2.0, -1.00, -8.0])
features = training[["sepal_length (cm)", "sepal_width (cm)"]].values
true_labels = training["label"].values
pred_labels = db.linear_decision_boundary_classifier(dec_bound_vec, features,␣
↪true_labels, features)
sklm.accuracy_score(true_labels, pred_labels)
```

[13]: 0.9907407407407407

[14]:
```
dec_bound_vec = np.array([2.0, -1.00, -8.0])
features = val[["sepal_length (cm)", "sepal_width (cm)"]].values
```

```
true_labels = val["label"].values
pred_labels = db.linear_decision_boundary_classifier(dec_bound_vec, features,␣
 ↪true_labels, features)
sklm.accuracy_score(true_labels, pred_labels)
```

[14]: 0.8636363636363636

[15]:
```
dec_bound_vec = np.array([2.0, -1.00, -8.0])
features = testing[["sepal_length (cm)", "sepal_width (cm)"]].values
true_labels = testing["label"].values
pred_labels = db.linear_decision_boundary_classifier(dec_bound_vec, features,␣
 ↪true_labels, features)
sklm.accuracy_score(true_labels, pred_labels)
```

[15]: 0.8571428571428571