Lab 3: Decision Boundaries

Introduction:

- This lab is an exercise in exploring how decision boundaries are used by machine learning methods to classify data. Specifically, the Iris data set is used to examine how different combination of features effect the efficacy of linear decision boundaries. Also, a training, testing, and validation split is used in order to examine how the accuracy of the model changes depending on what data split the model is evaluated on. Lastly, accuracy metrics are examined in order to gain a greater understanding of what the metrics indicate about the models they are used to evaluate.

Author: Nigel Nelson

Reflection Questions:

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?*
    A. Visually it appears that decision boundary 2 (y = 0.75x – 0.9) does the best job separating the two points.
2. *Which of the two decision boundaries gave the more accurate predictions?*
    A. Decision Boundary 2 gave the more accurate predictions with an accuracy score of 0.9933.
3. *How does the accuracy metric seem to relate to the ability of the decision boundary to separate the classes?*
    A. The accuracy score seems to somewhat relate to the ability of the decision boundary to separate the classes. The reason for this is that while the resulting scores appear to correctly identify which boundaries are better at separating classes compared to other, the magnitudes of these scores don't show as strong as a correlation. This is because decision boundary 3 does not separate any of the data points from each other, and it still manages to have a score of 0.6667.

Problem 2:

1. *For which pair of features are the classes more easily separated?*
    A. Petal width vs. petal length seems to do a better job separating the classes when compared to sepal width vs sepal length. For petal width vs. petal length, each class appeared to only have two data points that appeared on the opposite side of the decision boundary, whereas sepal width vs sepal length had almost 10 each.
2. *Just by looking at your plots, for which pair of features does the decision boundary do a better job of separating the classes?*
    A. Petal width vs. petal length appears to do a better job of seperating the two Iris classes compared to sepal width vs sepal length as its decision boundary does a better job of

isolating more of the correct data points.

3. *Which decision boundary gives the most accurate predictions?*
    A. Decision boundary 2 (y = -0.7x + 5) on the petal feature plot had the highest accuracy score in problem 2 with an accuracy score of 0.95.
4. *How does the choice of features seem to impact the ability to make accurate predictions?*
    A. When dealing with linear decision boundaries, it appears to be very important which features are chosen to make predictions. Obviously, due to their nature, linear decision boundaries are not flexible at all, selecting features that create the clearest linear boundary between classes will result in the most accurate predictions. Otherwise, a decision boundary will imprecisely cut through a mixture of data points, leaving instances of both classes on each side, resulting in inaccuracy and mislabeling.

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?*
    A. The data that was evaluated as the most accurate is the training data, with an accuracy score of 0.991. This does not come as much of a surprise because the model was tuned to have the decision boundary correctly isolate as many data points as possible for this specific set of data. However, I believe the accuracy score of the testing data is a more realistic representation of the performance of the model. The reason for this is similar to what was previously stated, the training set was used to train this model, so the training set does not challenge the model to generalize when evaluating after its initial training. However, the testing set is data the model has never seen before, so if the model is overfit due to training the testing set will expose that, or if the model correctly generalized the problem space, evaluating on the testing set will result in a higher score. It is for these reasons that I believe the testing set's accuracy score of 0.857 is a more realistic representation of the performance of the model.
2. *Look at the scatter plot of the validation data points. Will the model make errors in predicting the labels? Why?*
    A. Yes, this model will make errors in predicting. The reason for this is that the decision boundary cuts through a portion of the Iris Setosa data points, leaving 6 of them on the same side as the rest of the non Iris Setosa data points. This means that when predicting, it will assume that those 6 Iris Setosa data points are actually non Iris Setosa data points.
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?*
    A. The testing set appears to be much more similar to the validation set that the training set does. The biggest thing that the two have in common that the training set does not have, is several Iris setosa data points on the non Iris Setosa side of the decision boundary.
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?*
    A. The accuracy of the training set compared to the validation set are very different, with scores of 0.9907 and 0.8636, respectively. However, the scores of the testing set compared to the validation set are much more similar, 0.857 and 0.8636, respectively. Based upon those accuracy score comparisons, it is clear that the accuracy of the testing set is a better representation of the accuracy for the validation set. Both have a similar

number of entries, and from a visual inspection of the feature values, both seem to contain entries with the same range of feature values. Also, with their accuracy scores being so close in value this further indicates the data sets are a similar challenge for the model to undertake.

5. *Explain why training and evaluating a model on the same data set can be deceptive.*
   A. Training and evaluating a model on the same data set can be deceptive in that it can give a false sense of real world accuracy. The reason for this is that when training, a model can become overfit, and memorize the training data instead of generalizing trends. However, by also testing on the same data, this over fitting will not become apparent, as the model will be tested on the same data it has overfit itself to. This can lead to high accuracy scores that can give a false sense that the model is generalizing the problem space. Whereas if separate data is used for testing it will likely result in a low testing accuracy score, exposing the overfit model.
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.*
   A. Dividing the data into separate training and testing sets resolves the issue of training and testing on the same data as it shows whether the model is generalizing or memorizing. It does this because if the model memorizes the training set and is overfit, testing on the testing set will result in a low accuracy score and uncover the fact the model was memorizing. Whereas if the testing set also returns a high score, then that is an indication that the model is actually generalizing the problem space correctly.
7. *Name 3 potential ramifications for publishing a model that was trained and characterized using the same data set.*
   A. The first possible ramification comes from lending readers a false sense of how accurate the model is in terms of real world data. This could result in readers attempting to use the model in the real world, with real world consequences, and cause harm to businesses or individuals. Another ramification would be a losing credibility with peers. After publishing a model, peers would likely try to replicate the results published, and likely have the good practice of using a training and testing split, resulting in the discovery that you didn't use this practice, and thus showed that you're willing to cut corners and publish incomplete findings. Lastly, this could result in damages to yourself and career from the result of lawsuits. If faith was put in your published model and it was implemented in a medical setting, patients could be harmed from the likely overfit model, and the medical professionals that used your model may sue for negligence.

Imports:

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from decision_boundaries import linear_decision_boundary_classifier
```

1. Decision Boundaries and Evaluation of Model Predictions with Metrics

- **Loading Iris Setosa Data:**

In [2]:
```
setosa_df = pd.read_csv("setosa_data.csv")
setosa_df.head()
```

Out[2]:

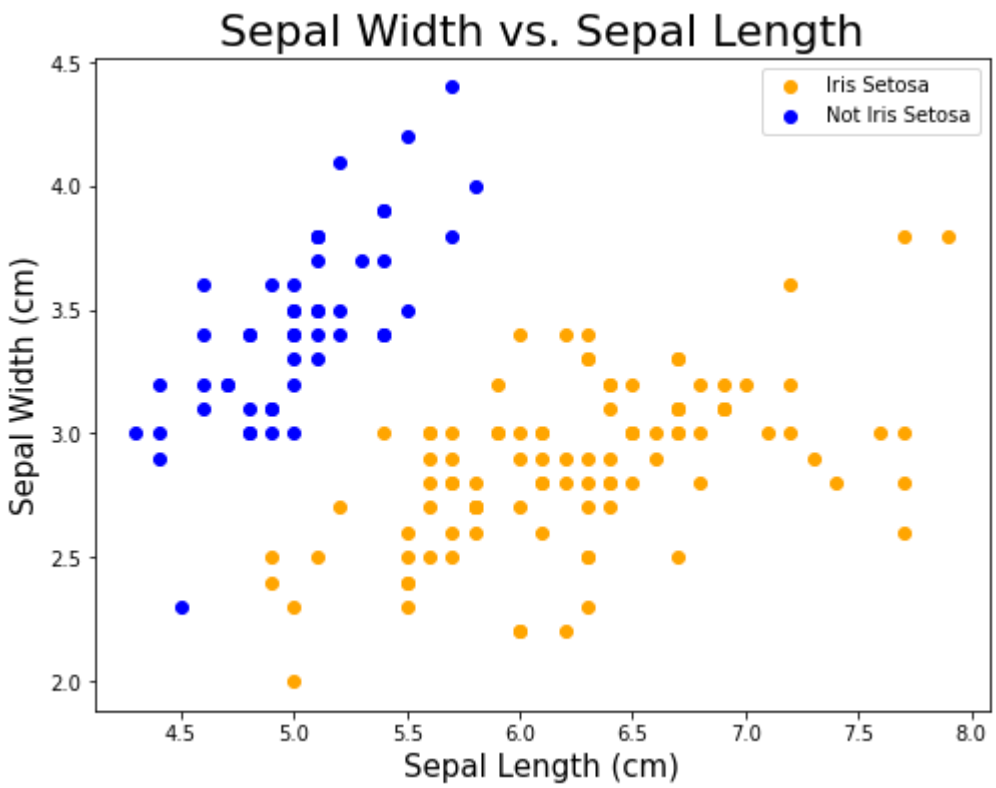
	label	sepal_length (cm)	sepal_width (cm)
0	not setosa	5.1	3.5
1	not setosa	4.9	3.0
2	not setosa	4.7	3.2
3	not setosa	4.6	3.1
4	not setosa	5.0	3.6

- **Displaying Sepal Width vs. Sepal Length plot:**

In [3]:
```
sepal_lengths = setosa_df[["sepal_length (cm)"]]
sepal_widths = setosa_df[["sepal_width (cm)"]]
labels = setosa_df[["label"]]
setosa_mask = (labels == 'setosa').values
setosa_sepal_widths = sepal_widths.values[setosa_mask]
setosa_sepal_lengths = sepal_lengths.values[setosa_mask]
non_setosa_sepal_widths = sepal_widths.values[setosa_mask == False]
non_setosa_sepal_lengths = sepal_lengths.values[setosa_mask == False]
```

In [4]:
```
figure, axis = plt.subplots(figsize=(8,6))
axis.scatter(setosa_sepal_lengths, setosa_sepal_widths, label="Iris Setosa",
axis.scatter(non_setosa_sepal_lengths, non_setosa_sepal_widths, label="Not Ir
axis.set_xlabel('Sepal Length (cm)', fontsize=15)
axis.set_ylabel('Sepal Width (cm)', fontsize=15)
axis.set_title("Sepal Width vs. Sepal Length", fontsize=22)
plt.legend()
```

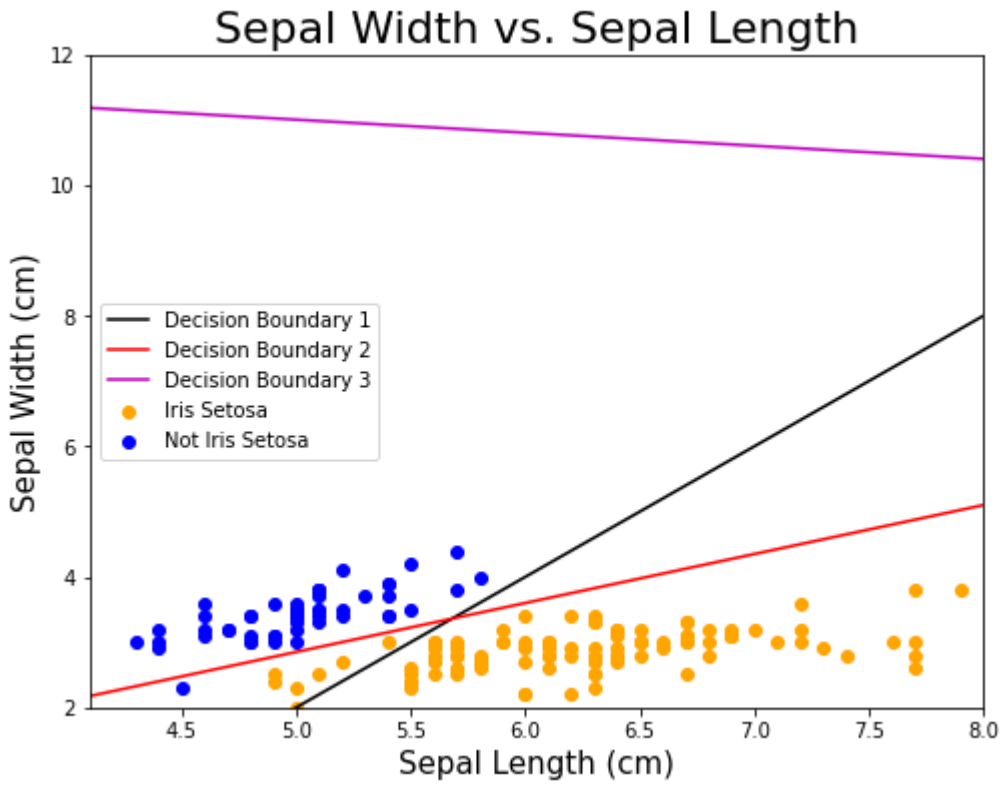
Out[4]: <matplotlib.legend.Legend at 0x1bdd949a550>



- **Displaying Sepal Width vs. Sepal Length plot with Decision Boundary:**

In [5]:
```
figure, axis = plt.subplots(figsize=(8,6))
axis.scatter(setosa_sepal_lengths, setosa_sepal_widths, label="Iris Setosa",
axis.scatter(non_setosa_sepal_lengths, non_setosa_sepal_widths, label="Not Ir
axis.set_xlabel('Sepal Length (cm)', fontsize=15)
axis.set_ylabel('Sepal Width (cm)', fontsize=15)
axis.set_title("Sepal Width vs. Sepal Length", fontsize=22)

x = np.linspace(2,8)
y = 2 * x - 8;
axis.plot(x, y, c='k', label='Decision Boundary 1')

y = 0.75 * x - 0.9;
axis.plot(x, y, c='r', label='Decision Boundary 2')

y = -0.2 * x + 12;
axis.plot(x, y, c='m', label='Decision Boundary 3')

plt.legend()
plt.xlim(4.1, 8)
plt.ylim(2, 12)
```

Out[5]: (2.0, 12.0)



- **Rewriting Decision Boundaries:**
  - Decision Boundary 1: **2x - y - 8 = 0**
  - Decision Boundary 2: **0.75x - y - 0.9 = 0**

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

- **Evaluating the accuracy of the Decision Boundaries:**

In [6]:
```
print('\033[1m' +"Accuracy score using sepal width and sepal length features:
print()

decision_boundaries = np.array([[2, -1, -8], [0.75, -1, -0.9], [0.2, 1, -12]]
features = setosa_df[["sepal_length (cm)", "sepal_width (cm)"]].values
true_labels = setosa_df["label"].values

i = 1
for decision_boundary in decision_boundaries:
    pred_labels = linear_decision_boundary_classifier(decision_boundary,
    features, true_labels, features)
    score = accuracy_score(true_labels, pred_labels)
    print("Decision Boundary " + str(i) + "'s accuracy score: "
          + '\033[1m' + str(round(score, 4)) + '\033[0m')
    i += 1
```

```
Accuracy score using sepal width and sepal length features:

Decision Boundary 1's accuracy score: 0.9533
Decision Boundary 2's accuracy score: 0.9933
Decision Boundary 3's accuracy score: 0.6667
```

2. Predictive Power of Features

- **Importing Iris Versicolor and Iris Virginica data:**

In [7]:
```
versicolor_virginica_df = pd.read_csv("versicolor_virginica_data.csv")
versicolor_virginica_df.head()
```

Out[7]:

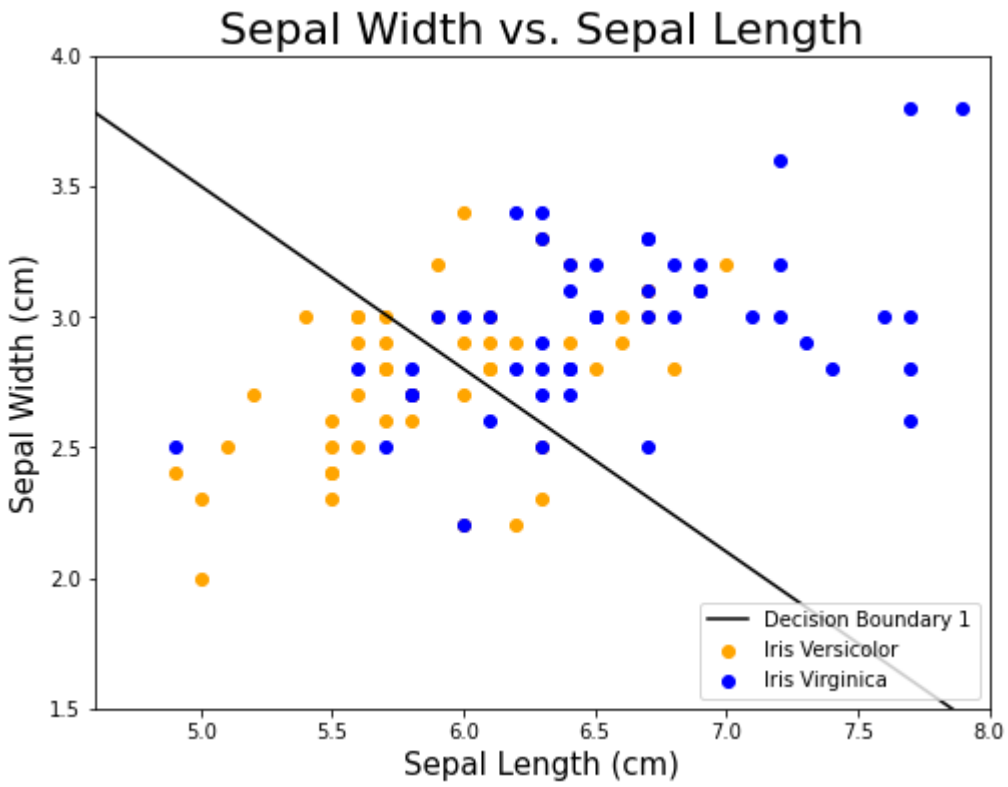
	label	sepal_length (cm)	sepal_width (cm)	petal_length (cm)	petal_width (cm)
0	versicolor	7.0	3.2	4.7	1.4
1	versicolor	6.4	3.2	4.5	1.5
2	versicolor	6.9	3.1	4.9	1.5
3	versicolor	5.5	2.3	4.0	1.3
4	versicolor	6.5	2.8	4.6	1.5

- **Displaying Sepal Width vs. Sepal Length plot with Decision Boundary:**

In [8]:
```
vv_sepal_lengths = versicolor_virginica_df[["sepal_length (cm)"]]
vv_sepal_widths = versicolor_virginica_df[["sepal_width (cm)"]]
labels = versicolor_virginica_df[["label"]]
versicolor_mask = (labels == 'versicolor').values
versicolor_sepal_widths = vv_sepal_widths.values[versicolor_mask]
versicolor_sepal_lengths = vv_sepal_lengths.values[versicolor_mask]
virginica_sepal_widths = vv_sepal_widths.values[versicolor_mask == False]
virginica_sepal_lengths = vv_sepal_lengths.values[versicolor_mask == False]
```

In [9]:
```
figure, axis = plt.subplots(figsize=(8,6))
axis.scatter(versicolor_sepal_lengths, versicolor_sepal_widths, label="Iris V
axis.scatter(virginica_sepal_lengths, virginica_sepal_widths, label="Iris Vir
axis.set_xlabel('Sepal Length (cm)', fontsize=15)
axis.set_ylabel('Sepal Width (cm)', fontsize=15)
axis.set_title("Sepal Width vs. Sepal Length", fontsize=22)

x = np.linspace(2,8)
y = -0.7 * x + 7;
axis.plot(x, y, c='k', label='Decision Boundary 1')

plt.legend(loc='lower right')
plt.xlim(4.6, 8)
plt.ylim(1.5, 4)
```

Out[9]: (1.5, 4.0)

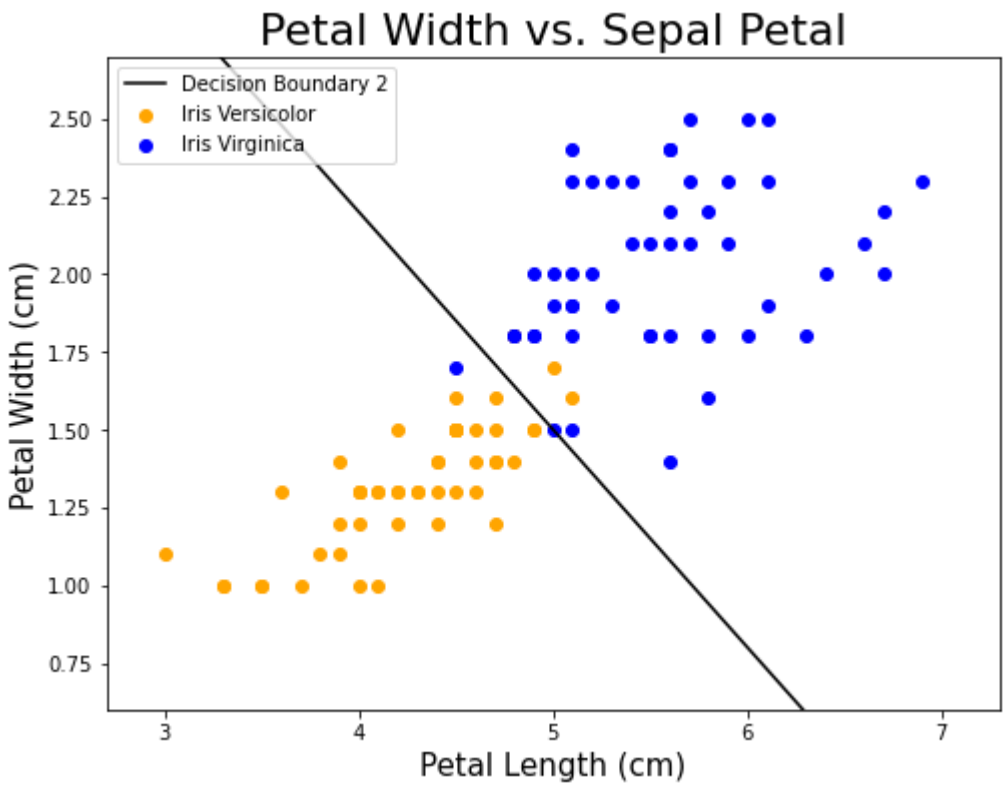


- **Displaying Petal Width vs. Petal Length plot with Decision Boundary:**

In [10]:
```
vv_petal_lengths = versicolor_virginica_df[["petal_length (cm)"]]
vv_petal_widths = versicolor_virginica_df[["petal_width (cm)"]]
labels = versicolor_virginica_df[["label"]]
versicolor_mask = (labels == 'versicolor').values
versicolor_petal_widths = vv_petal_widths.values[versicolor_mask]
versicolor_petal_lengths = vv_petal_lengths.values[versicolor_mask]
virginica_petal_widths = vv_petal_widths.values[versicolor_mask == False]
virginica_petal_lengths = vv_petal_lengths.values[versicolor_mask == False]
```

In [11]:
```
figure, axis = plt.subplots(figsize=(8,6))
axis.scatter(versicolor_petal_lengths, versicolor_petal_widths, label="Iris V
axis.scatter(virginica_petal_lengths, virginica_petal_widths, label="Iris Vir
axis.set_xlabel('Petal Length (cm)', fontsize=15)
axis.set_ylabel('Petal Width (cm)', fontsize=15)
axis.set_title("Petal Width vs. Sepal Petal", fontsize=22)

x = np.linspace(2,8)
y = -0.7 * x + 5;
axis.plot(x, y, c='k', label='Decision Boundary 2')

plt.legend(loc='upper left')
plt.xlim(2.7, 7.3)
plt.ylim(.6, 2.7)
```

Out[11]: (0.6, 2.7)



- **Rewriting Decision Boundaries:**
  - Decision Boundary 1: **0.7x + y - 7 = 0**
  - Decision Boundary 2: **0.7x + y - 5 = 0**

- **Evaluating the accuracy of the Decision Boundaries:**

In [12]:
```
sepal_dec_bound_vec = np.array([0.7, 1.0, -7])
sepal_features = versicolor_virginica_df[["sepal_length (cm)", "sepal_width (
sepal_true_labels = versicolor_virginica_df["label"].values
sepal_pred_labels = linear_decision_boundary_classifier(sepal_dec_bound_vec,
                                                        sepal_true_labels, se
sepal_score = accuracy_score(sepal_true_labels, sepal_pred_labels)
print('\033[1m' +"Accuracy score using sepal width and sepal length features:
print()
print("Decision Boundary 1's accuracy score: " + str(round(sepal_score, 4)))
print()

petal_dec_bound_vec = np.array([0.7, 1, -5])
petal_features = versicolor_virginica_df[["petal_length (cm)", "petal_width (
petal_true_labels = versicolor_virginica_df["label"].values
petal_pred_labels = linear_decision_boundary_classifier(petal_dec_bound_vec,
                                                        petal_true_labels, pe
petal_score = accuracy_score(petal_true_labels, petal_pred_labels)
print('\033[1m' +"Accuracy score using petal width and petal length features:
print()
print("Decision Boundary 2's accuracy score: " + str(round(petal_score, 4)))
```

```
Accuracy score using sepal width and sepal length features:

Decision Boundary 1's accuracy score: 0.7

Accuracy score using petal width and petal length features:

Decision Boundary 2's accuracy score: 0.95
```

3. Experimental Setup with Train-Test Splitting

- **Importing training, testing, and validation Iris Setosa data:**

In [13]:
```
training_setosa_df = pd.read_csv("setosa_training.csv")
testing_setosa_df = pd.read_csv("setosa_testing.csv")
validation_setosa_df = pd.read_csv("setosa_validation.csv")
```

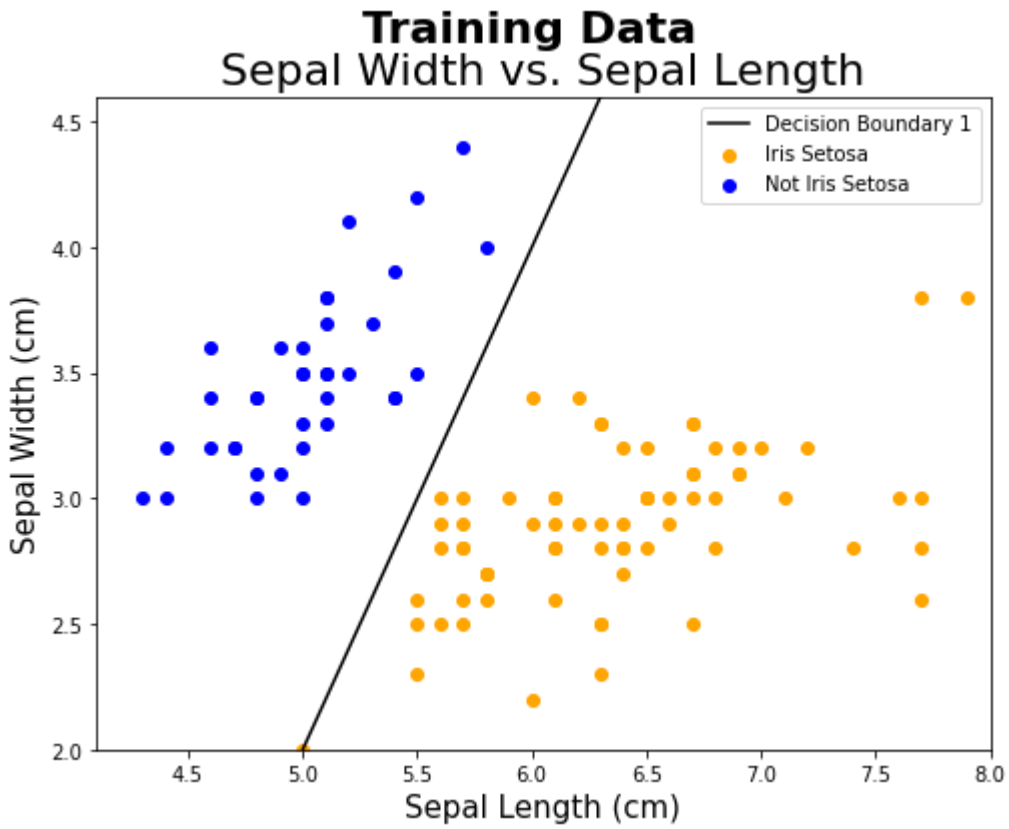
- **Defining Decision Boundary:**
  - Decision Boundary 1: y = 2x -8

- **Plotting Sepal Width vs. Sepal Length for Training Data:**

In [14]:
```
sepal_lengths = training_setosa_df[["sepal_length (cm)"]]
sepal_widths = training_setosa_df[["sepal_width (cm)"]]
labels = training_setosa_df[["label"]]
setosa_mask = (labels == 'setosa').values
setosa_sepal_widths = sepal_widths.values[setosa_mask]
setosa_sepal_lengths = sepal_lengths.values[setosa_mask]
non_setosa_sepal_widths = sepal_widths.values[setosa_mask == False]
non_setosa_sepal_lengths = sepal_lengths.values[setosa_mask == False]
```

In [15]:
```
figure, axis = plt.subplots(figsize=(8,6))
axis.scatter(setosa_sepal_lengths, setosa_sepal_widths, label="Iris Setosa",
axis.scatter(non_setosa_sepal_lengths, non_setosa_sepal_widths, label="Not Ir
axis.set_xlabel('Sepal Length (cm)', fontsize=15)
axis.set_ylabel('Sepal Width (cm)', fontsize=15)
axis.set_title("Sepal Width vs. Sepal Length", fontsize=22)
figure.suptitle("Training Data", fontsize=22, weight='bold')

x = np.linspace(2,8)
y = 2 * x - 8;
axis.plot(x, y, c='k', label='Decision Boundary 1')

plt.legend()
plt.xlim(4.1, 8)
plt.ylim(2, 4.6)
```

Out[15]: (2.0, 4.6)

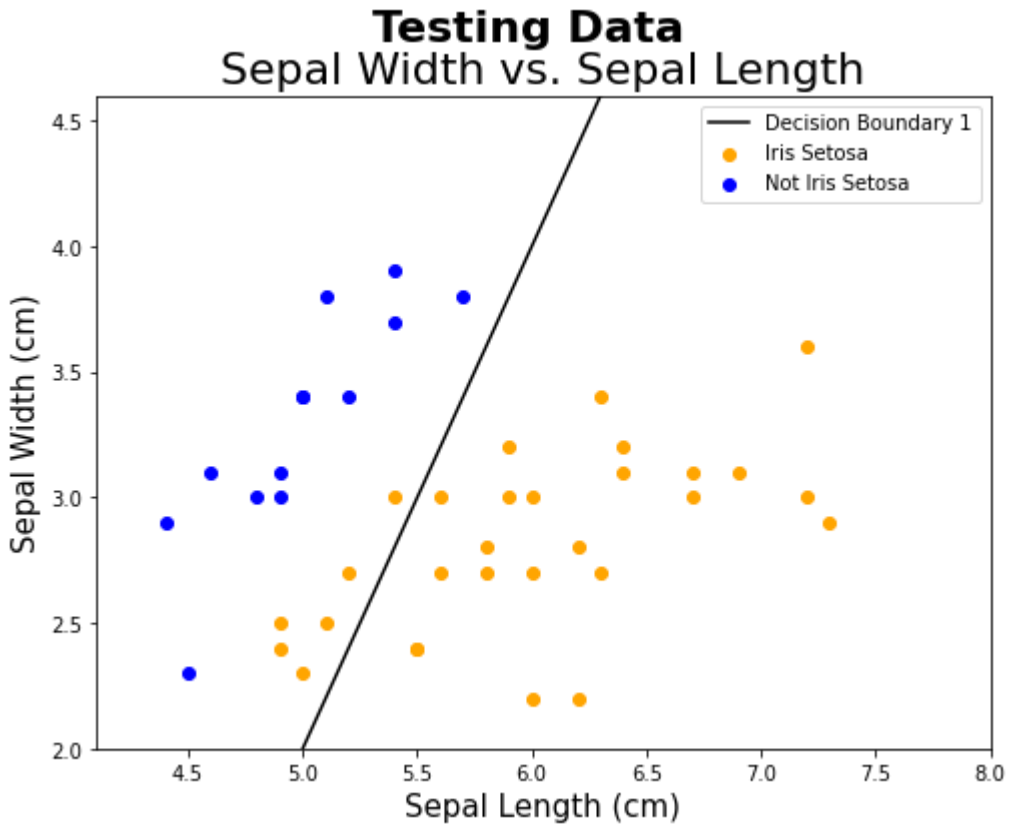


- **Plotting Sepal Width vs. Sepal Length for Testing Data:**

In [16]:
```
sepal_lengths = testing_setosa_df[["sepal_length (cm)"]]
sepal_widths = testing_setosa_df[["sepal_width (cm)"]]
labels = testing_setosa_df[["label"]]
setosa_mask = (labels == 'setosa').values
setosa_sepal_widths = sepal_widths.values[setosa_mask]
setosa_sepal_lengths = sepal_lengths.values[setosa_mask]
non_setosa_sepal_widths = sepal_widths.values[setosa_mask == False]
non_setosa_sepal_lengths = sepal_lengths.values[setosa_mask == False]
```

In [17]:
```
figure, axis = plt.subplots(figsize=(8,6))
axis.scatter(setosa_sepal_lengths, setosa_sepal_widths, label="Iris Setosa",
axis.scatter(non_setosa_sepal_lengths, non_setosa_sepal_widths, label="Not Ir
axis.set_xlabel('Sepal Length (cm)', fontsize=15)
axis.set_ylabel('Sepal Width (cm)', fontsize=15)
axis.set_title("Sepal Width vs. Sepal Length", fontsize=22)
figure.suptitle("Testing Data", fontsize=22, weight='bold')

x = np.linspace(2,8)
y = 2 * x - 8;
axis.plot(x, y, c='k', label='Decision Boundary 1')

plt.legend()
plt.xlim(4.1, 8)
plt.ylim(2, 4.6)
```

Out[17]: (2.0, 4.6)

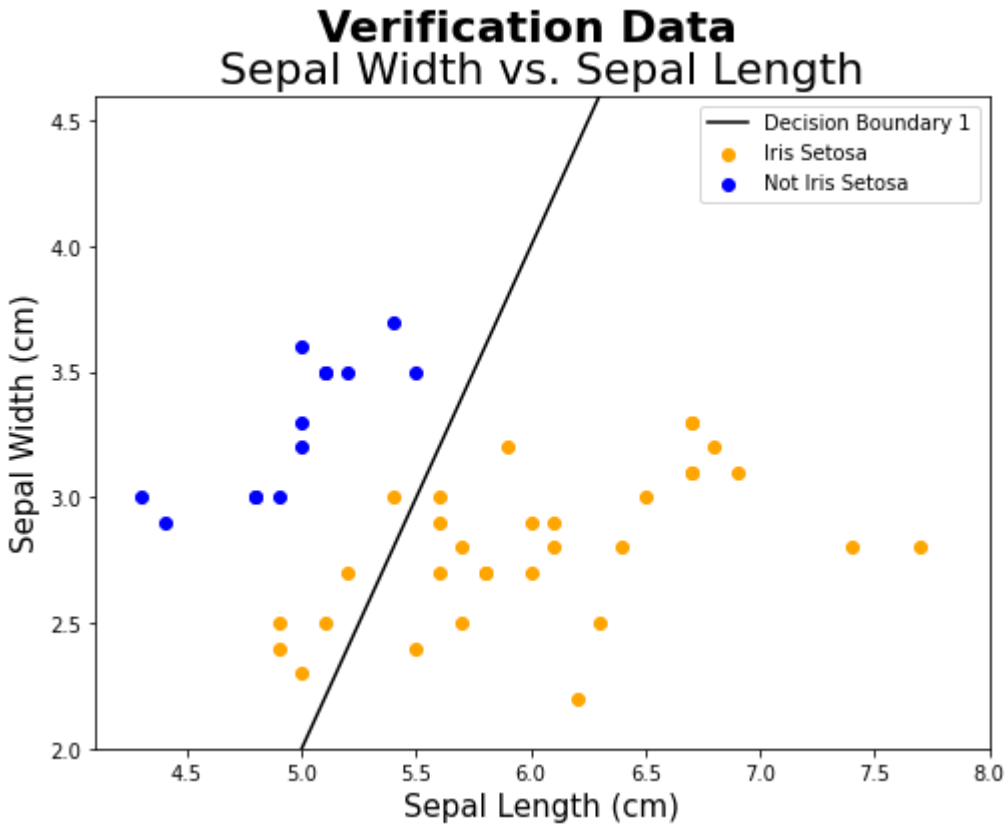


- **Plotting Sepal Width vs. Sepal Length for Validation Data:**

In [18]:
```
sepal_lengths = validation_setosa_df[["sepal_length (cm)"]]
sepal_widths = validation_setosa_df[["sepal_width (cm)"]]
labels = validation_setosa_df[["label"]]
setosa_mask = (labels == 'setosa').values
setosa_sepal_widths = sepal_widths.values[setosa_mask]
setosa_sepal_lengths = sepal_lengths.values[setosa_mask]
non_setosa_sepal_widths = sepal_widths.values[setosa_mask == False]
non_setosa_sepal_lengths = sepal_lengths.values[setosa_mask == False]
```

In [19]:
```
figure, axis = plt.subplots(figsize=(8,6))
axis.scatter(setosa_sepal_lengths, setosa_sepal_widths, label="Iris Setosa",
axis.scatter(non_setosa_sepal_lengths, non_setosa_sepal_widths, label="Not Ir
axis.set_xlabel('Sepal Length (cm)', fontsize=15)
axis.set_ylabel('Sepal Width (cm)', fontsize=15)
axis.set_title("Sepal Width vs. Sepal Length", fontsize=22)
figure.suptitle("Verification Data", fontsize=22, weight='bold')

x = np.linspace(2,8)
y = 2 * x - 8;
axis.plot(x, y, c='k', label='Decision Boundary 1')

plt.legend()
plt.xlim(4.1, 8)
plt.ylim(2, 4.6)
```

Out[19]: (2.0, 4.6)



- **Evaluating the trained model on the training, testing, and validation data sets:**

In [20]:
```
dec_bound_vec = np.array([2, -1, -8])
training_features = training_setosa_df[["sepal_length (cm)", "sepal_width (cm
testing_features = testing_setosa_df[["sepal_length (cm)", "sepal_width (cm)"
validation_features = validation_setosa_df[["sepal_length (cm)", "sepal_width
training_true_labels = training_setosa_df["label"].values
testing_true_labels = testing_setosa_df["label"].values
validation_true_labels = validation_setosa_df["label"].values

training_pred_labels = linear_decision_boundary_classifier(dec_bound_vec, tra
                                                           training_true_labe
testing_pred_labels = linear_decision_boundary_classifier(dec_bound_vec, trai
                                                          training_true_label
validation_pred_labels = linear_decision_boundary_classifier(dec_bound_vec, t
                                                             training_true_la

training_score = accuracy_score(training_true_labels, training_pred_labels)
testing_score = accuracy_score(testing_true_labels, testing_pred_labels)
validation_score = accuracy_score(validation_true_labels, validation_pred_lab
print('\033[1m' + "Accuracy scores for different combinations of training and
print()
print("Model trained on training set, evaluated on " + '\033[1m' + "training
      + "set: " + str(round(training_score, 4)))
print("Model trained on training set, evaluated on " + '\033[1m' + "testing "
      + "set: " + str(round(testing_score, 4)))
print("Model trained on training set, evaluated on " + '\033[1m' + "validatio
      + "set: " + str(round(validation_score, 4)))
```

```
Accuracy scores for different combinations of training and evaluating data
 sets:

Model trained on training set, evaluated on training set: 0.9907
Model trained on training set, evaluated on testing set: 0.8571
Model trained on training set, evaluated on validation set: 0.8636
```

Conclusion:

- This lab was an exercise in exploring how decision boundaries are used by machine learning methods to classify data. The importance of how different combinations of features effect the efficacy of linear decision boundaries was discovered through experimentation using the Iris data set. Also, it was discovered that by using a training, testing, and validation split, undesirable traits such as overfitting could be exposed in a model. Lastly, a greater understanding of what accuracy metrics convey about a model was gained through experimentation with various decision boundaries.