Lab 3 - Decision Boundaries

Name: Paige Rosynek

Date: 9.28.2022

Introduction

The purpose of this lab is to gain familiarity with decision boundaries and how they help classify data, as well as how to effectively evaluate a model. The data set used throughout this lab is the Iris dataset. Each part of this lab followed a similar procedure: plot two features (and color according to class), plot a decision boundary, classify points based on the decision boundary, and the calculate the accuracy of the predicted labels. In Part 1 of the lab, only two features are plotted and 3 different decision boundaries are given. The accuracy of the model for each of the decision boundaries was calculated and compared in order to observe the relationship between how well a decision boundary separates data and its accuracy. In Part 2 of the lab, two different sets of two features is plotted and compared to show the difference in how a model performs on data that is easily separated and data that is not. Lastly, Part 3 of the lab performs an experimental setup with test-train splitting of the data and how it effects the accuracy of a model. We will be observing how evaluating data on the training set, testing set, and validation set influences how well the model can predict labels given data, as well as how to interpret accuracy in each of these situations. In general, the experiments performed in this lab further exemplify the importance of an experimental setup when training model as well as the general psteps required for training a classification model when given a data set.

Problem 1:

1. Just by looking at your plot, which of the three decision boundaries does the best job of separating the two classes of points?

```
It can be observed from the plot in 1D that boundary line 2 (red
line) does the best job separating the two classes. The decision
boundary has the equation: $x_2 = 0.75x_1 - 0.9$ and does a good
job separating the classes such that the majority of points for
each class are on opposite sides of the line.
```

2. Which of the three decision boundaries gave the more accurate predictions?

```
The decision boundary that had the highest prediction accuracy was
decision boundary 2 (red line), which is also the line that best
separates the classes. It had an accuracy of 99.33%.
```

3. How does the accuracy metric seem to relate to the ability of the decision boundary to separate the classes?

```
The decision boundary's ability to separate the classes is directly
related to the accuracy of the predictions that the decision
boundary produces. Therefore, the better a decision boundary can
separate the classes, the more accurate the model predictions will
be for that boundary line.
```

Problem 2:

1. For which pair of features are the classes more easily separated?

For the veriscolor and virgnica dataset, it can be observed from the plots in 2c, that petal width and petal length are more easily separated than sepal width/length. The scatter plot for sepal width versus sepal height shows the different categories of flowers mixed together; and it has no evident pattern in which the features can be easily separated. However, the petal width versus height plot shows points in which points from the same class are near each other making it easily separable.

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 decision boundary best separates the different classes for the features petal width and petal height. For the most part, all the points on each side of the line are the same class.

3. Which decision boundary gives the most accurate predictions?

The decision boundary that produced the most accurate predictions was the decision boundary for the petal width v. petal length plot, with the equation: $x_2 = -0.7x_1 + 5$. It had an accuracy of about 95%.

4. How does the choice of features seem to impact the ability to make accurate predictions?

The accuracy of predicitions is better if the features are easily separated by a decision boundary line.

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 accuracy calculated for the training set was 100% while the
accuracy calculated for the testing set was about 85.71%, thus the
predictions evaluated on the training set were calculated to be
"more accurate". However, it is misleading to believe that the
accuracy for the model evaluated on the training set will be 100%
accurate. The reason that the accuracy for the model evaluated on
the training set was 100% is because the model was both trained and
evaluated on the same exact set of data, thus the model wasn't
really evaluated at all since it had seen all of the data before
and used that same data to build the model. Therefore, it is more
realistic to represent the accuracy of the model with the accuracy
of the model that was evaluated on the testing set. The model
evaluated on the testing set actually shows a representation of the
ability of the model to predict unseen data.
```

2. Look at the scatter plot of the validation data points. Will the model make errors in predicting the labels? Why?

```
The model will make errors predicting the labels because the
decision boundary for the validation data separates only most of
the classes. Since the decision boundary doesn't separate the
validation points perfectly, it is to be expected that the model
will mislabel some of the data.
```

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 set is more representative of the validation data
because the graphs are similarly separated and because the
validation and testing data both contain data unseen by the model.
The testing set will have error similar to the validation set
because the data within these sets has not been seen by the model
during training, which better tests the model's ability to predict
a label for a new data point not within the dataset.
```

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 calculated accuracy for the testing set was 85.71% and the
training set had an accuracy of 100%. The testing accuracy is more
representative of the accuracy for the validation set because the
data in the testing and validation sets is unseen data to the model
that was trained on the training set. The training set had an
accuracy of 100% because the model was trained on the data in the
training set, so by evaluating the model with the training set we
are giving the model everything it already knows so it doesn't make
any mistakes. By evaluating the model with data it hasn't seen, it
better simulates how accurate the model would be when used in a
real-world setting.
```

5. Explain why training and evaluating a model on the same data set can be deceptive.

```
As mentioned above, training and evaluating a model on the same
data set can be deceptive because although you might find the
```

```
accuracy of your model to be 100%, it will not perform this well
when given an observation not in the training set. The model forms
its predictions because it found patterns in the training set. Then
if you test the model on how well it can predict the same data it
was trained on, the model will have an accuracy of 100% since it
"knows" the data already. Thus, training and testing a model on the
same data gives a false sense of confidence in the model because
the accuracy appears to be really good when in reality the model's
accuracy is much less for data it has never seen before.
```

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.

```
Dividing a dataset into training and testing sets without
replacement solves some of the similar issues of testing a model on
the same data it was trained on. By not training a model on
redundant data the possibility of overfitting the model is reduced
because the model won't be able to recognize or "memorize" certain
data. Having redundant data in the testing and training sets will
also make your model accuracy deceiving because the model will
correctly predict any data it has seen previously, like the data
from the training set.
```

7. Name 3 potential ramifications for publishing a model that was trained and characterized using the same data set.

```
One problem would be the model's poor ability to generalize. A
model trained and evaluated on the same data set is overfit (has a
high variance) and specific to the dataset, thus it cannot
accurately predict new, never seen data making it a bad model to
publish. Another possible consequence is that if the data used to
train and test the model were not scaled it is possible that one
feature will have a higher importance than another feature due to
differences in the value ranges for the features. Overall, the
biggest issue with training and evaluating a model on the same data
is overfitting. By training and testing on the same data, the model
learns patterns that are very specific to that data and is
therefore overfit. An overfit model does not generalize well to new
data points and will thus perform poorly in the real-world.
```

1. Decision Boundaries & Evaluation of Model Predictions

Part A

In [ ]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

setosa_df = pd.read_csv('../Data/setosa_data.csv')
```

Part B

In [ ]:
```python
setosa_df.head()
```

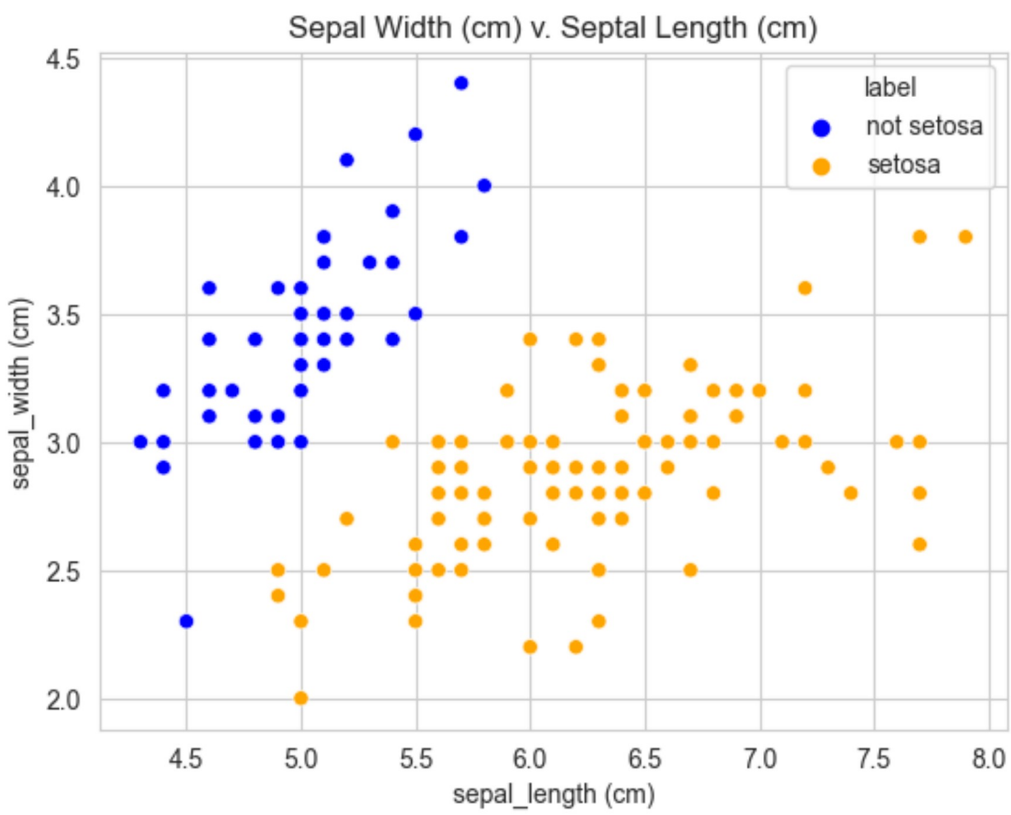
Out[ ]:

	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

Part C

In [ ]:

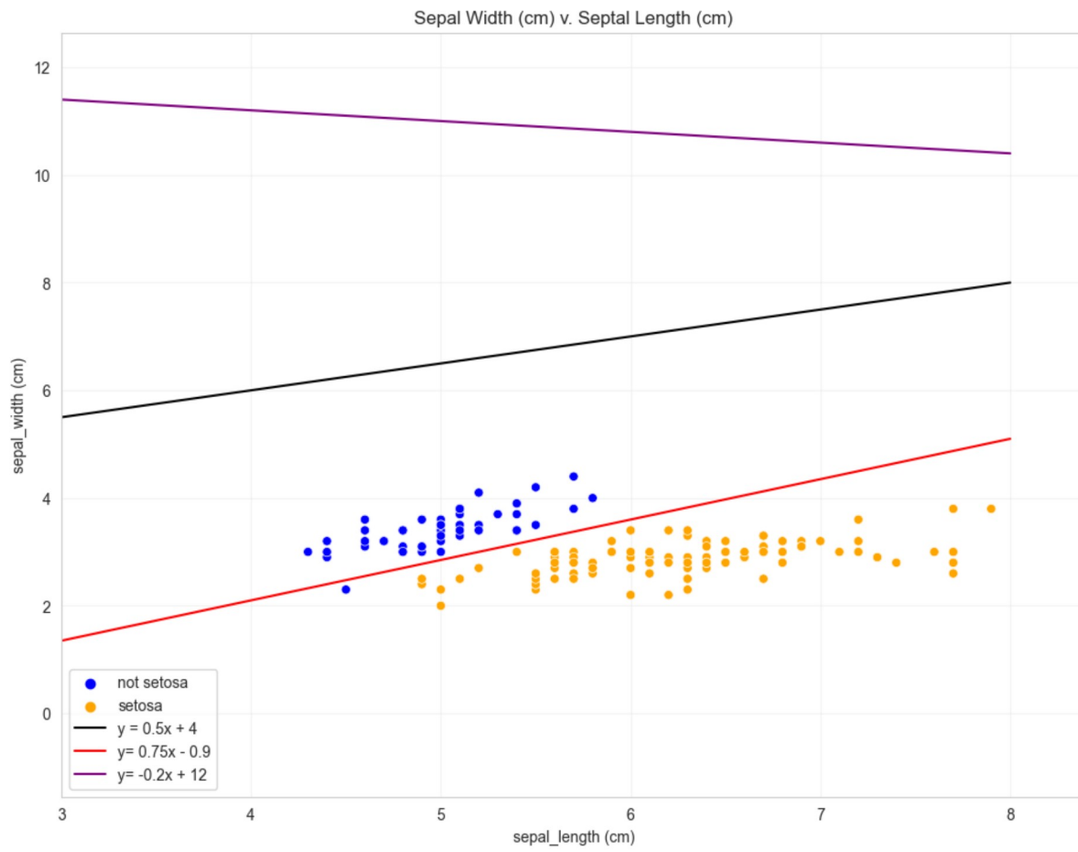
```
colors = {'setosa' : 'orange', 'not setosa' : 'blue'}

sns.scatterplot(data=setosa_df, x='sepal_length (cm)', y='sepal_width (cm)', hue='l
plt.title('Sepal Width (cm) v. Septal Length (cm)')
plt.show()
```



Part D

In [ ]:

```
x = np.linspace(0,8,1000)
colors = {'setosa' : 'orange', 'not setosa' : 'blue'}

y1 = 0.5*x + 4
y2 = 0.75*x - 0.9
y3 = -0.2*x + 12

plt.figure().set_size_inches(12,9)

sns.scatterplot(data=setosa_df, x='sepal_length (cm)', y='sepal_width (cm)', hue='l
plt.title('Sepal Width (cm) v. Septal Length (cm)')

plt.plot(x, y1, c='black', label='y = 0.5x + 4')
plt.plot(x, y2, c='red', label='y= 0.75x - 0.9')
plt.plot(x, y3, c='purple', label='y= -0.2x + 12')

plt.xlim(3)
plt.legend()
plt.grid(alpha=0.2)
plt.show()
```



e.

- Decision Boundary 1 (black) : $x_1 - 2x_2 + 8 = 0$
- Decision Boundary 2 (red) : $-0.75x_1 + x_2 + 0.9 = 0$
- Decision Boundary 3 (purple) : $0.2x_1 + x_2 - 12 = 0$

f.

In [ ]:
```
from decision_boundaries import linear_decision_boundary_classifier

dec_bound_vec1 = np.array([1.0, -2.0, 8.0])
dec_bound_vec2 = np.array([-0.75, 1.0, 0.9])
dec_bound_vec3 = np.array([0.2, 1.0, -12])

features = setosa_df[['sepal_length (cm)', 'sepal_width (cm)']].values
true_labels = setosa_df['label'].values

pred_labels1 = linear_decision_boundary_classifier(dec_bound_vec1, features, true_l
pred_labels2 = linear_decision_boundary_classifier(dec_bound_vec2, features, true_l
pred_labels3 = linear_decision_boundary_classifier(dec_bound_vec3, features, true_l

print('boundary 1 : x1 - 2x2 + 8 = 0\n', pred_labels1)
print('boundary 2 : -0.75x1 + x2 + 0.9 = 0\n', pred_labels2)
print('boundary 3 : 0.2x1 + x2 - 12 = 0\n', pred_labels3)
```

```
boundary 1 : x1 - 2x2 + 8 = 0
 ['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' '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']
boundary 2 : -0.75x1 + x2 + 0.9 = 0
 ['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']
boundary 3 : 0.2x1 + x2 - 12 = 0
 ['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' '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']
```

```
c:\Users\rosynekp\OneDrive - Milwaukee School of Engineering\Desktop\cs3400\labs\la
b03\lab03-rosynekp\Code\decision_boundaries.py:27: FutureWarning: Unlike other redu
ction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically
preserves the axis it acts along. In SciPy 1.11.0, this behavior will change: the d
efault value of `keepdims` will become False, the `axis` over which the statistic i
s taken will be eliminated, and the value None will no longer be accepted. Set `kee
pdims` to True or False to avoid this warning.
  neg_label = mode(training_labels[neg_mask])[0]
c:\Users\rosynekp\OneDrive - Milwaukee School of Engineering\Desktop\cs3400\labs\la
b03\lab03-rosynekp\Code\decision_boundaries.py:31: FutureWarning: Unlike other redu
ction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically
preserves the axis it acts along. In SciPy 1.11.0, this behavior will change: the d
efault value of `keepdims` will become False, the `axis` over which the statistic i
s taken will be eliminated, and the value None will no longer be accepted. Set `kee
pdims` to True or False to avoid this warning.
  pos_label = mode(training_labels[pos_mask])[0]
c:\Python310\lib\site-packages\scipy\stats\_stats_py.py:110: RuntimeWarning: The in
put array could not be properly checked for nan values. nan values will be ignored.
  warnings.warn("The input array could not be properly "
```

g.

In [ ]:
```
from sklearn.metrics import accuracy_score

acc1 = accuracy_score(true_labels, pred_labels1)
acc2 = accuracy_score(true_labels, pred_labels2)
acc3 = accuracy_score(true_labels, pred_labels3)

print('accuracy for boundary 1 = {}'.format(acc1))
print('accuracy for boundary 2 = {}'.format(acc2))
print('accuracy for boundary 3 = {}'.format(acc3))
```

```
accuracy for boundary 1 = 0.6666666666666666
accuracy for boundary 2 = 0.9933333333333333
accuracy for boundary 3 = 0.6666666666666666
```

2. Predictive Power of Features

a.

In [ ]:
```
vv_data = pd.read_csv('../Data/versicolor_virginica_data.csv')
```

b.

In [ ]:
```
vv_data.head()
```

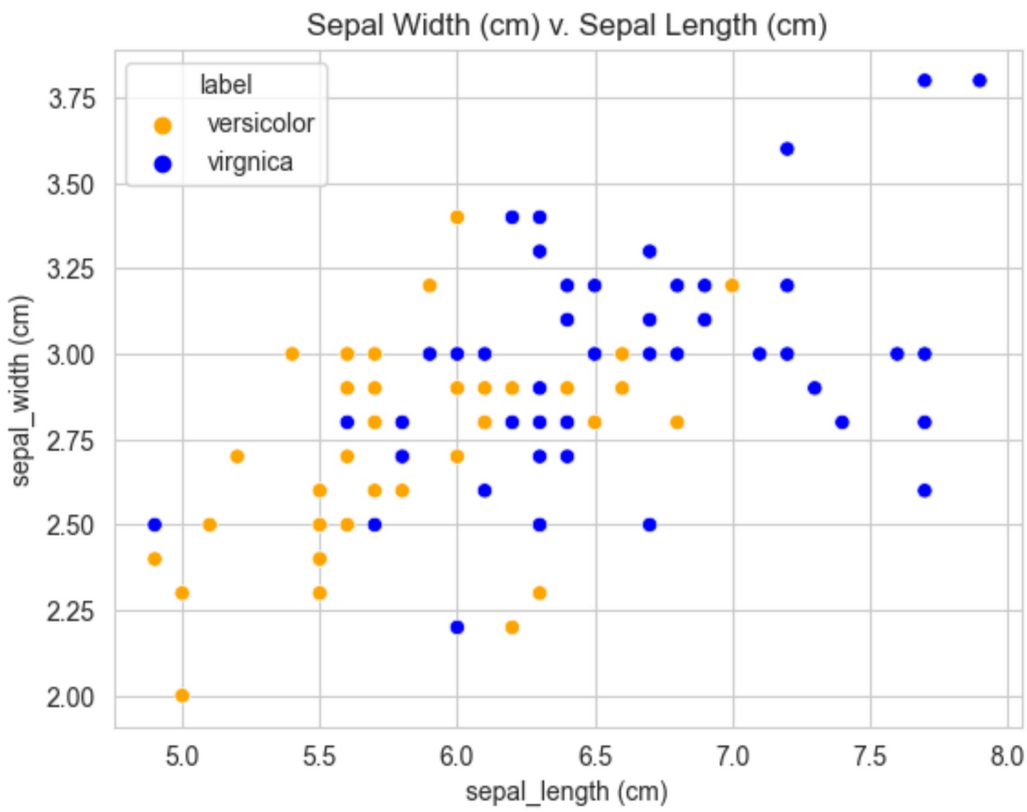
Out[ ]:

	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

c.

In [ ]:
```
colors = {'versicolor' : 'orange', 'virgnica' : 'blue'}
sns.scatterplot(data=vv_data, x='sepal_length (cm)', y='sepal_width (cm)', hue='lab
plt.title('Sepal Width (cm) v. Sepal Length (cm)')
```

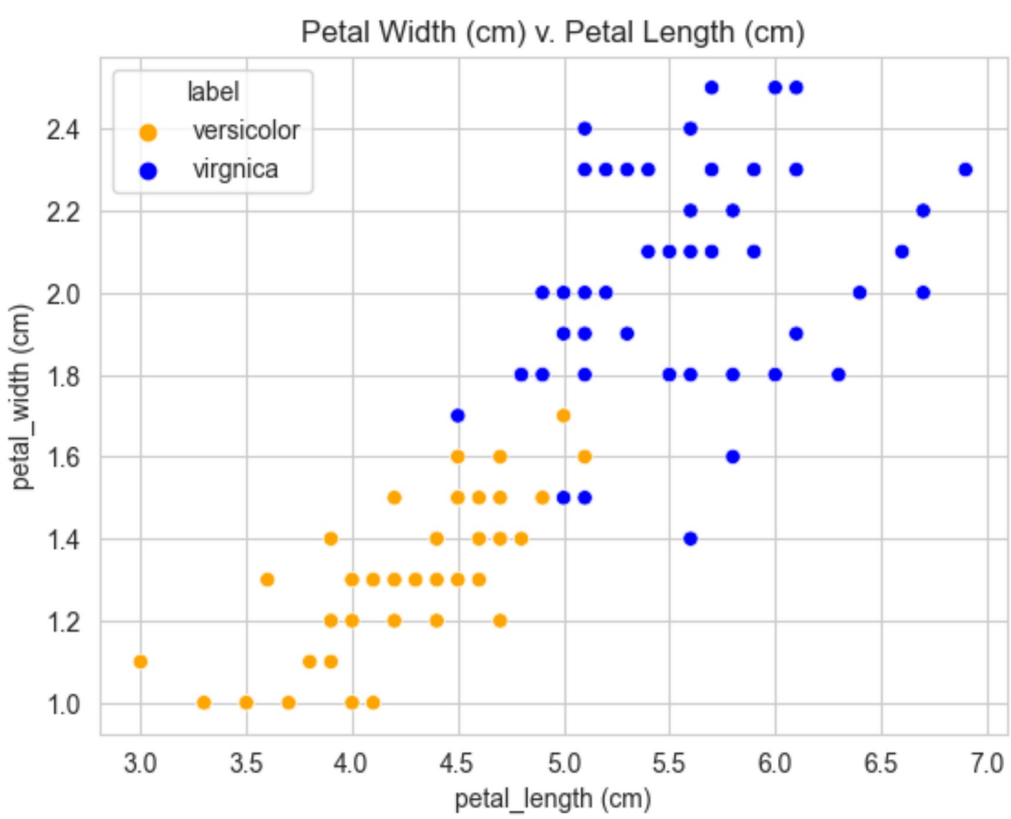
Out[ ]:
```
Text(0.5, 1.0, 'Sepal Width (cm) v. Sepal Length (cm)')
```



d.

In [ ]:
```
colors = {'versicolor' : 'orange', 'virgnica' : 'blue'}
sns.scatterplot(data=vv_data, x='petal_length (cm)', y='petal_width (cm)', hue='lab
plt.title('Petal Width (cm) v. Petal Length (cm)')
```

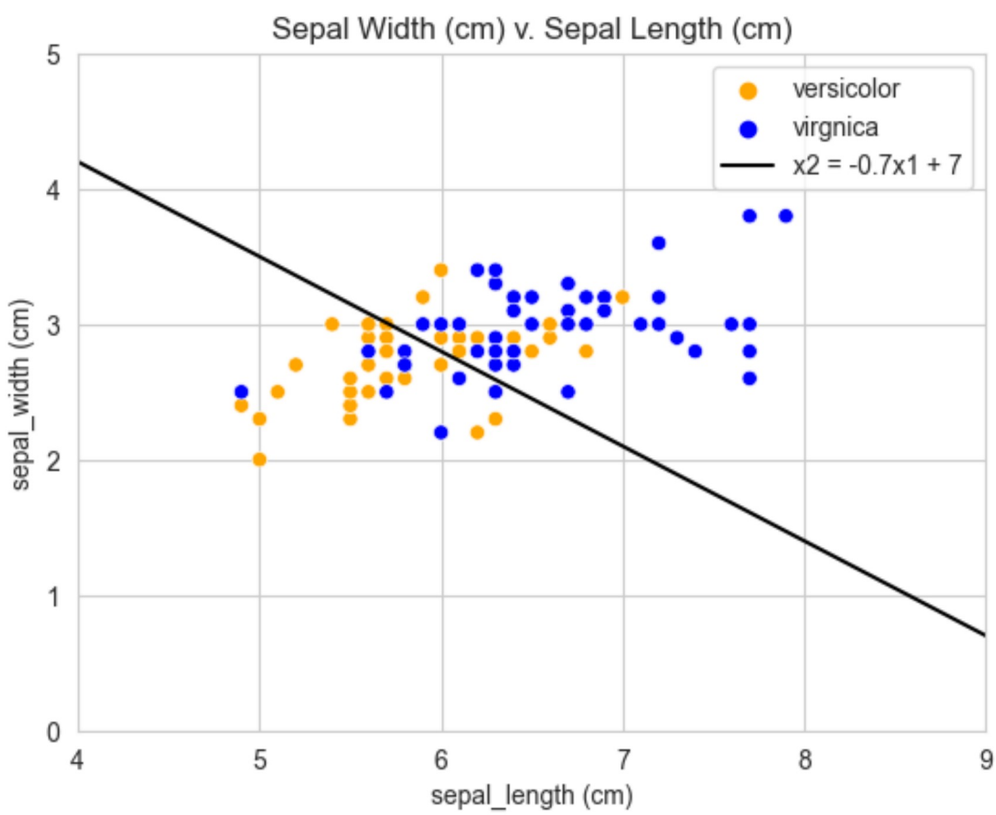
Out[ ]:
```
Text(0.5, 1.0, 'Petal Width (cm) v. Petal Length (cm)')
```

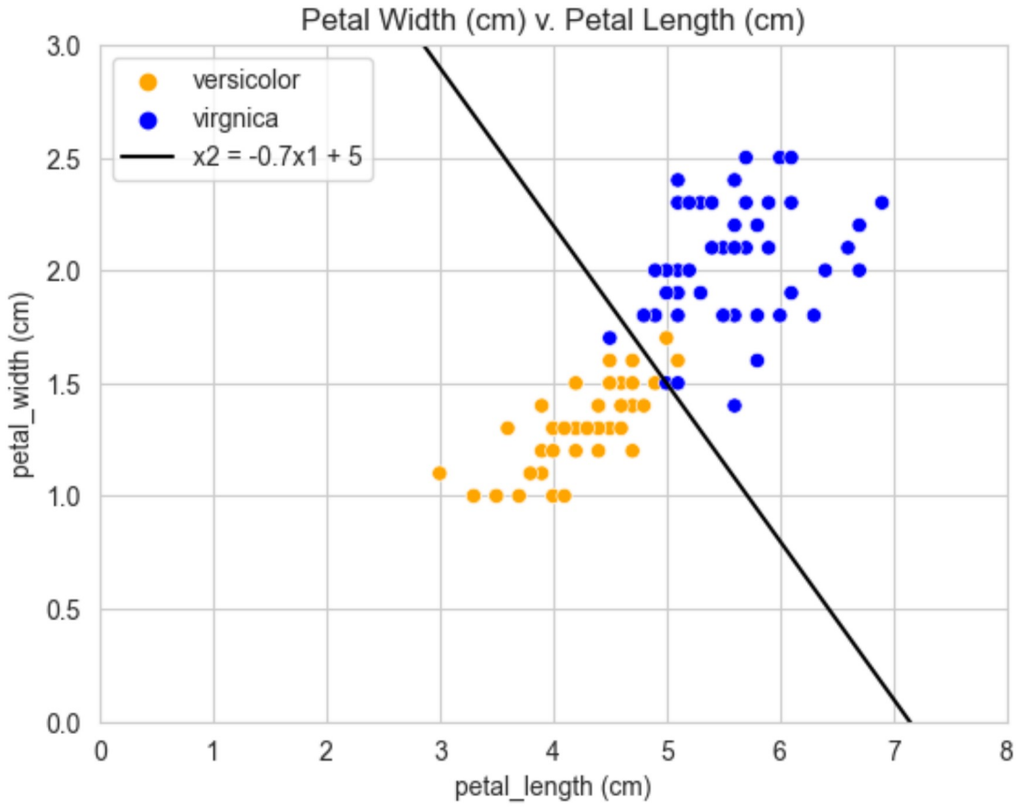


e.

In [ ]:

```
x = np.linspace(0,10,100)
colors = {'versicolor' : 'orange', 'virgnica' : 'blue'}

#---boundary1---
sns.scatterplot(data=vv_data, x='sepal_length (cm)', y='sepal_width (cm)', hue='lab
plt.title('Sepal Width (cm) v. Sepal Length (cm)')

y1 = -0.7*x + 7
plt.plot(x, y1, c='black', label='x2 = -0.7x1 + 7')
plt.legend()
plt.xlim(4,9)
plt.ylim(0,5)
plt.show()

#---boundary2---
sns.scatterplot(data=vv_data, x='petal_length (cm)', y='petal_width (cm)', hue='lab
plt.title('Petal Width (cm) v. Petal Length (cm)')

y2 = -0.7*x + 5
plt.plot(x, y2, c='black', label='x2 = -0.7x1 + 5')
plt.legend()
plt.xlim(0,8)
plt.ylim(0,3)
plt.show()
```





f.

- Decision boundary $x_2 = -0.7x_1 + 7$

  standard form : $0.7x_1 + x_2 - 7 = 0$

- Decision boundary $x_2 = -0.7x_1 + 5$

  standard form : $0.7x_1 + x_2 - 5 = 0$

g.

In [ ]:
```
db_vec1 = np.array([0.7, 1.0, -7.0])
db_vec2 = np.array([0.7, 1.0, -5.0])

features1 = vv_data[['sepal_length (cm)', 'sepal_width (cm)']].values
features2 = vv_data[['petal_length (cm)', 'petal_width (cm)']].values
true_labels = vv_data['label'].values

pred1 = linear_decision_boundary_classifier(db_vec1, features1, true_labels, featur
pred2 = linear_decision_boundary_classifier(db_vec2, features2, true_labels, featur

print('boundary 1 : x2 = -0.7x1 + 7\n', pred1)
print('boundary 2 : -x2 = -0.7x + 5\n', pred2)
```

```
boundary 1 : x2 = -0.7x1 + 7
 ['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']
boundary 2 : -x2 = -0.7x + 5
 ['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' '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']
```

```
c:\Users\rosynekp\OneDrive - Milwaukee School of Engineering\Desktop\cs3400\labs\la
b03\lab03-rosynekp\Code\decision_boundaries.py:27: FutureWarning: Unlike other redu
ction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically
preserves the axis it acts along. In SciPy 1.11.0, this behavior will change: the d
efault value of `keepdims` will become False, the `axis` over which the statistic i
s taken will be eliminated, and the value None will no longer be accepted. Set `kee
pdims` to True or False to avoid this warning.
  neg_label = mode(training_labels[neg_mask])[0]
c:\Python310\lib\site-packages\scipy\stats\_stats_py.py:110: RuntimeWarning: The in
put array could not be properly checked for nan values. nan values will be ignored.
  warnings.warn("The input array could not be properly "
c:\Users\rosynekp\OneDrive - Milwaukee School of Engineering\Desktop\cs3400\labs\la
b03\lab03-rosynekp\Code\decision_boundaries.py:31: FutureWarning: Unlike other redu
ction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically
preserves the axis it acts along. In SciPy 1.11.0, this behavior will change: the d
efault value of `keepdims` will become False, the `axis` over which the statistic i
s taken will be eliminated, and the value None will no longer be accepted. Set `kee
pdims` to True or False to avoid this warning.
  pos_label = mode(training_labels[pos_mask])[0]
```

h.

In [ ]:
```
acc1 = accuracy_score(true_labels, pred1)
acc2 = accuracy_score(true_labels, pred2)

print('accuracy for boundary 1 = {}'.format(acc1))
print('accuracy for boundary 2 = {}'.format(acc2))
```

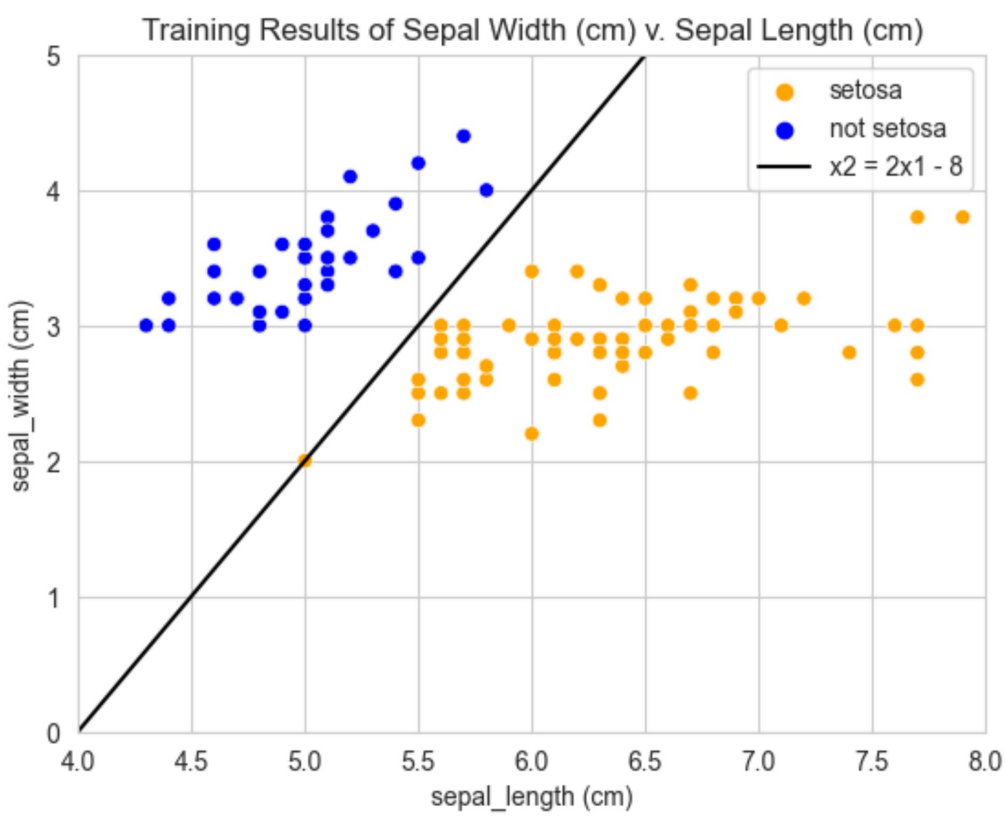
```
accuracy for boundary 1 = 0.7
accuracy for boundary 2 = 0.95
```

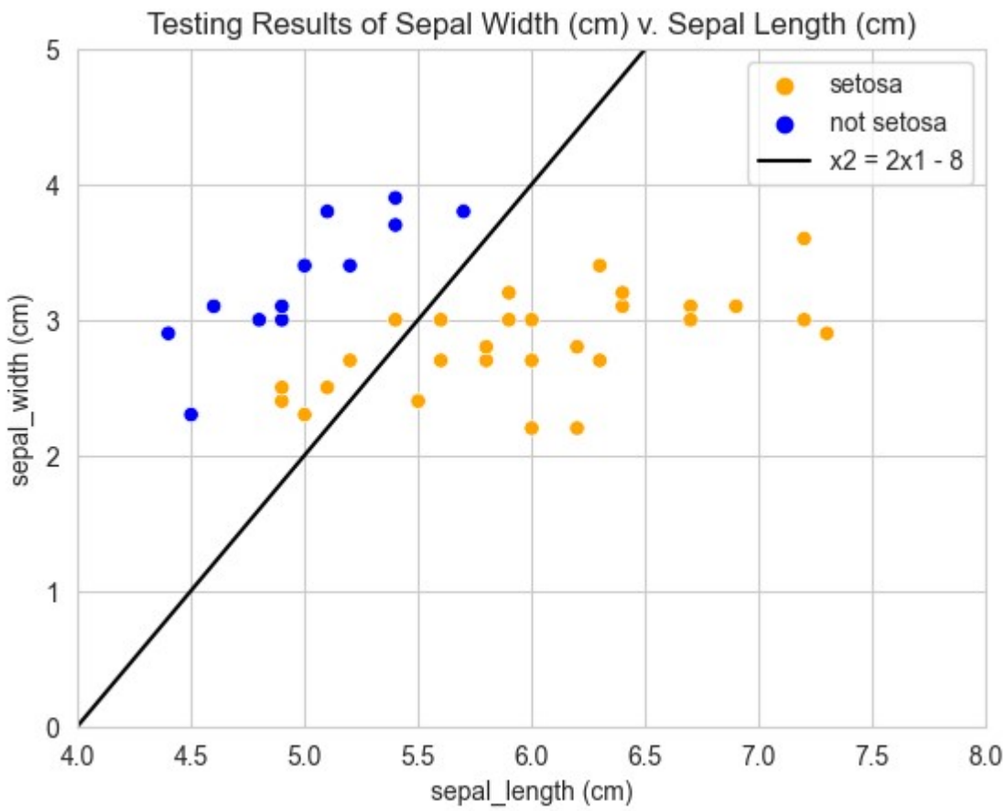
3. Experimental Setup with Test-Train Splitting

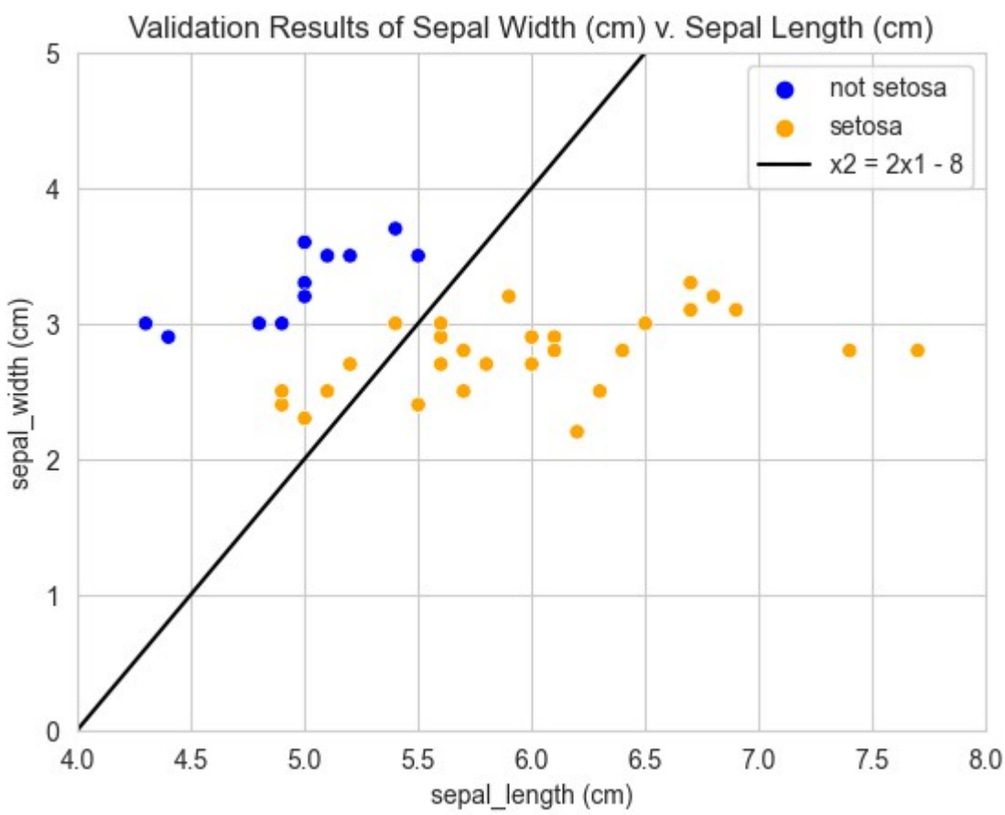
a.

In [ ]:

```
setosa_train = pd.read_csv('../Data/setosa_training.csv')
setosa_test = pd.read_csv('../Data/setosa_testing.csv')
setosa_val = pd.read_csv('../Data/setosa_validation.csv')

x = np.linspace(0,10,100)
y = 2*x - 8
colors = {'setosa' : 'orange', 'not setosa' : 'blue'}

sns.scatterplot(data=setosa_train, x='sepal_length (cm)', y='sepal_width (cm)', hue
plt.title('Training Results of Sepal Width (cm) v. Sepal Length (cm)')
plt.plot(x, y, c='black', label='x2 = 2x1 - 8')
plt.xlim(4,8)
plt.ylim(0,5)
plt.legend()
plt.show()

sns.scatterplot(data=setosa_test, x='sepal_length (cm)', y='sepal_width (cm)', hue=
plt.title('Testing Results of Sepal Width (cm) v. Sepal Length (cm)')
plt.plot(x, y, c='black', label='x2 = 2x1 - 8')
plt.xlim(4,8)
plt.ylim(0,5)
plt.legend()
plt.show()

sns.scatterplot(data=setosa_val, x='sepal_length (cm)', y='sepal_width (cm)', hue='
plt.title('Validation Results of Sepal Width (cm) v. Sepal Length (cm)')
plt.plot(x, y, c='black', label='x2 = 2x1 - 8')
plt.xlim(4,8)
plt.ylim(0,5)
plt.legend()
plt.show()
```




Testing Results of Sepal Width (cm) v. Sepal Length (cm)
setosa
not setosa
x2 = 2x1 - 8
sepal_width (cm)
sepal_length (cm)
0
1
2
3
4
5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


Validation Results of Sepal Width (cm) v. Sepal Length (cm)
not setosa
setosa
x2 = 2x1 - 8
sepal_width (cm)
sepal_length (cm)
0
1
2
3
4
5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

b.

In [ ]:
```
db_vec = np.array([-2.0, 1.0, 8.0])

feat_train = setosa_train[['sepal_length (cm)', 'sepal_width (cm)']]
feat_test = setosa_test[['sepal_length (cm)', 'sepal_width (cm)']]
feat_val = setosa_val[['sepal_length (cm)', 'sepal_width (cm)']]

true_train = setosa_train['label'].values
true_test = setosa_test['label'].values
true_val = setosa_val['label'].values

# train : training set  eval : training set
pred1 = linear_decision_boundary_classifier(db_vec, feat_train, true_train, feat_tr

# train : training set  eval : testing set
pred2 = linear_decision_boundary_classifier(db_vec, feat_train, true_train, feat_te

# train : training set  eval : validation set
pred3 = linear_decision_boundary_classifier(db_vec, feat_train, true_train, feat_va


acc1 = accuracy_score(true_train, pred1)
acc2 = accuracy_score(true_test, pred2)
acc3 = accuracy_score(true_val, pred3)

print('boundary line for all plots : x2 = 2x1 - 8')
print('evaluate on training set :\n\taccuracy = {}'.format(acc1))
print('evaluate on testing set :\n\taccuracy = {}'.format(acc2))
print('evaluate on validation set :\n\taccuracy = {}'.format(acc3))
```

```
boundary line for all plots : x2 = 2x1 - 8
evaluate on training set :
        accuracy = 1.0
evaluate on testing set :
        accuracy = 0.8571428571428571
evaluate on validation set :
        accuracy = 0.8636363636363636
```

```
c:\Users\rosynekp\OneDrive - Milwaukee School of Engineering\Desktop\cs3400\labs\la
b03\lab03-rosynekp\Code\decision_boundaries.py:27: FutureWarning: Unlike other redu
ction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically
preserves the axis it acts along. In SciPy 1.11.0, this behavior will change: the d
efault value of `keepdims` will become False, the `axis` over which the statistic i
s taken will be eliminated, and the value None will no longer be accepted. Set `kee
pdims` to True or False to avoid this warning.
  neg_label = mode(training_labels[neg_mask])[0]
c:\Python310\lib\site-packages\scipy\stats\_stats_py.py:110: RuntimeWarning: The in
put array could not be properly checked for nan values. nan values will be ignored.
  warnings.warn("The input array could not be properly "
c:\Users\rosynekp\OneDrive - Milwaukee School of Engineering\Desktop\cs3400\labs\la
b03\lab03-rosynekp\Code\decision_boundaries.py:31: FutureWarning: Unlike other redu
ction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically
preserves the axis it acts along. In SciPy 1.11.0, this behavior will change: the d
efault value of `keepdims` will become False, the `axis` over which the statistic i
s taken will be eliminated, and the value None will no longer be accepted. Set `kee
pdims` to True or False to avoid this warning.
  pos_label = mode(training_labels[pos_mask])[0]
```