Solution 1

Step 1

We are given the prediction rule is defined as:

$$(2x_1 - x_2 - 6)$$

To find the decision boundary we set the prediction rule equal to zero:

$$2x_1 - x_2 - 6 = 0 \qquad \text{or} \qquad x_2 = 2x_1 - 6$$

We can rearrange this to express x_2 in terms of x_1:

$$x_2 = 2x_1 - 6$$

Step 2

Find the point (x_1, x_2) where the decision boundary intersects the x_1 axis (i.e. $x_2 = 0$):

$$x_2 = 2x_1 - 6 \;\rightarrow\; 0 = 2x_1 - 6 \;\rightarrow\; 2x_1 = 6 \;\rightarrow\; x_1 = 3$$

Hence, the decision boundary intersects the x_1 axis at $(3, 0)$

Step 3

Find the point (x_1, x_2) where the decision boundary intersects the x_2 axis (i.e. $x_1 = 0$):

$$x_2 = 2x_1 - 6 \;\rightarrow\; x_2 = 2(0) - 6 \;\rightarrow\; 2x_2 = -6$$

Hence, the decision boundary intersects the x_1 axis at $(0, 6)$

Step 4

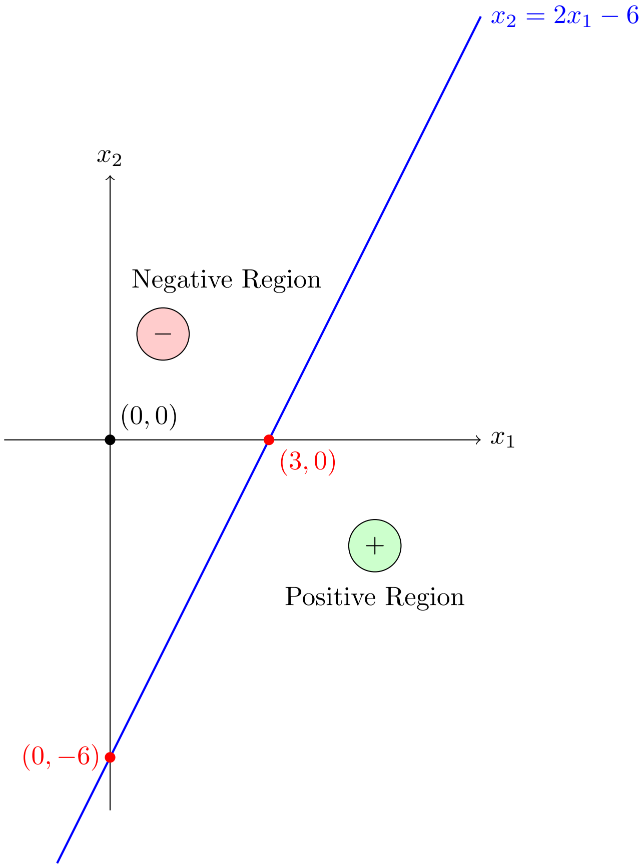
Test prediction rule at the point $(0, 0)$ to determine the classification above the decision boundary.

$$2x_1 - x_2 - 6 \;\rightarrow\; 2(0) - 0 - 6 \;\rightarrow\; -6$$

It follows that, $2x_1 - x_2 - 6 < 0$ at $(0, 0)$ and this area above the decision boundary will be classified as negative.

Step 5

Visualize the decision boundary:



$\therefore$ The decision boundary is the line $x_2 = 2x_1 - 6$, which intersects the x_1 axis at $(3, 0)$ and the x_2 axis at $(0, -6)$. The region below this line is classified as positive, and the region above it is classified as negative.

Solution 2 (a)

∴ The statement: *The data set is linearly separable.* is **definitely true**. The Perceptron algorithm will convergethe data is linearly separable and we are given: *it converges after making k updates.*

Solution 2 (b)

∴ The statement: *If the process were repeated with a different random permutation, it would again converge* is **definitely true**. Since the Perceptron algorithm will converge regardless of order.

Solution 2 (c)

∴ The statement: *If the process were repeated with a different random permutation, it would again converge after making k updates* is **possibly false**. The number of updates required for convergence can change depending on the order of data.

Solution 2 (d)

∴ The statement: *k is at most n* is **possibly false**. The number of updates k can exceed the number of data points n.

Solution 3

Step 1

A point (x, y) is misclassified when:

$$y(w \cdot x + b) \leq 0$$

The Perceptron algorithm tells us to update w and b when a point is misclassified as the following:

$$w = w + yx \text{ and } b = b + y$$

Step 2

We are given the following:

- Perceptron algorithm performs $p + q$ updates before converging
- p updates on data points with label $y_i = -1$
- q updates on data points with label $y_i = +1$

Step 3

Let the initial bias be $b = 0$. Each time a misclassified point is encountered, the bias is updated by adding the label y_i.

- For each of the p negative examples ($y_i = -1$), the bias decreases by 1: total change is $-p$
- For each of the q positive examples ($y_i = 1$), the bias increases by 1: total change is q

$\therefore$ The final value of the parameter b is $q - p$.

Solution 4 (a)

Step 1

Given:

- SVM classifier in $\mathbb{R}^2$
- Weight vector $w = (3, 4)$
- Bias term $b = -12$

The prediction rule would then be defined as:

$$(3x_1 + 4x_2 - 12)$$

Step 2

Find the point (x_1, x_2) where the decision boundary intersects the x_1 axis (i.e. $x_2 = 0$):

$$3x_1 + 4(0) - 12 = 0 \quad \rightarrow \quad 3x_1 = 12 \quad \rightarrow \quad x_1 = 4$$

Hence, the decision boundary intersects the x_1 axis at $(4, 0)$

Step 3

Find the point (x_1, x_2) where the decision boundary intersects the x_2 axis (i.e. $x_1 = 0$):

$$3(0) + 4x_2 - 12 = 0 \quad \rightarrow \quad 4x_2 = 12 \quad \rightarrow \quad x_2 = 3$$

Hence, the decision boundary intersects the x_2 axis at $(0, 3)$

Step 4

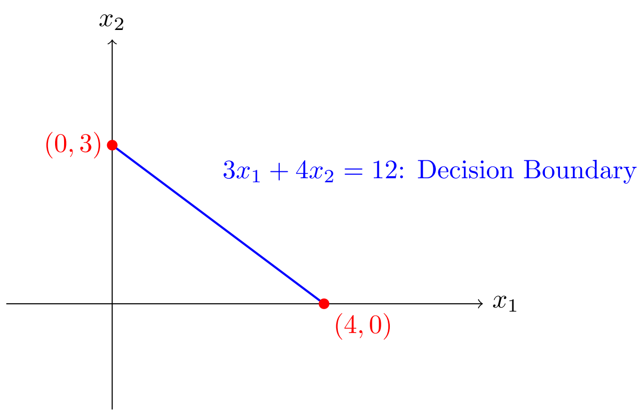
Test prediction rule at the point $(0, 0)$ to determine the classification below the decision boundary.

$$3(0) + 4(0) - 12 = -12$$

It follows that $3x_1 + 4x_2 - 12 < 0$ at $(0, 0)$, and so this region below the decision boundary will be classified as negative.

Step 5

Visualize the decision boundary:



Solution 4 (b)

Step 1

The margin boundaries are defined as:

$$w \cdot x + b = 1 \quad \text{(positive margin boundary)} \tag{1}$$
$$w \cdot x + b = -1 \quad \text{(negative margin boundary)} \tag{2}$$

Remember the prediction rule would then is defined as:

$$(3x_1 + 4x_2 - 12)$$

Step 2

Solve for right hand boundary $((3x_1 + 4x_2 - 13))$:
Find the point (x_1, x_2) where the right hand boundary intersects the x_1 axis (i.e. $x_2 = 0$):

$$3x_1 + 4(0) - 13 = 0 \;\;\rightarrow\;\; 3x_1 = 13 \;\;\rightarrow\;\; x_1 = \frac{13}{3}$$

Hence, the right hand boundary intersects the x_1 axis at $\left(\frac{13}{3}, 0\right)$.

Now, find the point (x_1, x_2) where the right hand boundary intersects the x_2 axis (i.e. $x_1 = 0$):

$$3(0) + 4x_2 - 13 = 0 \;\;\rightarrow\;\; 4x_2 = 13 \;\;\rightarrow\;\; x_2 = \frac{13}{4}$$

Hence, the right hand boundary intersects the x_2 axis at $\left(0, \frac{13}{4}\right)$.

Step 3

Solve for left hand boundary $((3x_1 + 4x_2 - 11))$:
Find the point (x_1, x_2) where the left hand boundary intersects the x_1 axis (i.e. $x_2 = 0$):

$$3x_1 + 4(0) - 11 = 0 \;\;\rightarrow\;\; 3x_1 = 11 \;\;\rightarrow\;\; x_1 = \frac{11}{3}$$

Hence, the left hand boundary intersects the x_1 axis at $\left(\frac{11}{3}, 0\right)$.

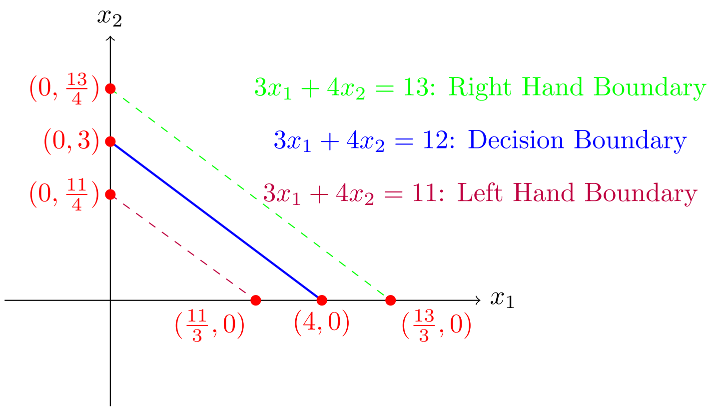
Now, find the point (x_1, x_2) where the left hand boundary intersects the x_2 axis (i.e. $x_1 = 0$):

$$3(0) + 4x_2 - 11 = 0 \;\;\rightarrow\;\; 4x_2 = 11 \;\;\rightarrow\;\; x_2 = \frac{11}{4}$$

Hence, the left hand boundary intersects the x_2 axis at $\left(0, \frac{11}{4}\right)$.

Step 4

Visualize the left hand and right hand boundaries:


x_2
x_1
$3x_1 + 4x_2 = 13$: Right Hand Boundary
$3x_1 + 4x_2 = 12$: Decision Boundary
$3x_1 + 4x_2 = 11$: Left Hand Boundary
$(0, \frac{13}{4})$
$(0, 3)$
$(0, \frac{11}{4})$
$(\frac{11}{3}, 0)$
$(4, 0)$
$(\frac{13}{3}, 0)$

Solution 4 (c)

Step 1

The margin of an SVM classifier is defined below where $||w||$, the Euclidean norm of the weight vector:

$$\text{Margin} = \frac{2}{||w||} = \frac{2}{\sqrt{w_1^2 + w_2^2}}$$

Step 2

Calculate the margin:

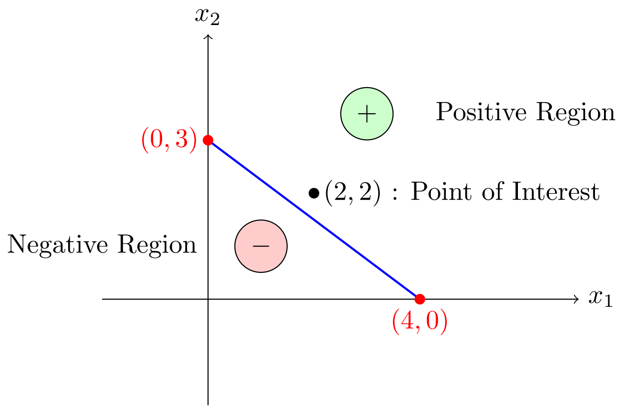
$$\begin{aligned}\text{Margin} &= \frac{2}{||w||} \\ &= \frac{2}{\sqrt{3^2 + 4^2}} \\ &= \frac{2}{\sqrt{25}} \\ &= \frac{2}{5} \\ &= 0.4\end{aligned}$$

$\therefore$ The margin of this SVM classifier is $\frac{2}{5}$ or 0.4 units.

Solution 4 (d)

Step 1

Use visulization from **Solution 4 (a)** and plot the point $(2, 2)$:



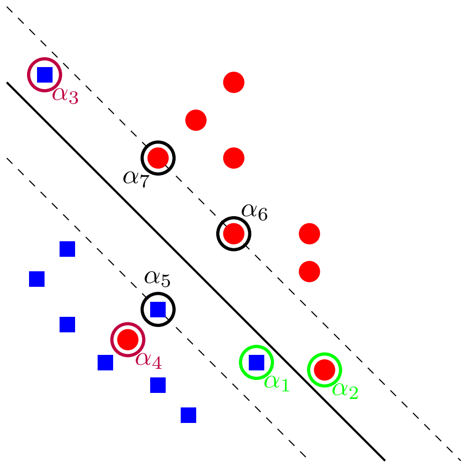
$\therefore$ The point $(2, 2)$ would be classified as positive.

Solution 5 (a)

Step 1

Support vectors(α_i)are the training points that lie exactly on the margin. However, we are told the decision boundary was obtained upon running *soft-margin SVM*. The soft-margin support vectors include the following points:

- points inside margin
- points that are misclassified
- points on the margin



Step 2

The slack variables for the points circled are as follows:

- points inside margin: α_1 and α_2 with slack variable values of approximately 0.5
- points that are misclassified: α_3 and α_4 with slack variable values of $\alpha_3 \approx 1.5$ and $\alpha_4 \approx 2.5$
- points on the margin: α_5, α_6, and α_7 with slack variable values of approximately 0

$\therefore$ the support vectors have been marked in **Step 1** and corresponding slack variable values have been indicated in **Step 2**.

Solution 5 (b)

Step 1

We know that increasing or decreasing the factor C or penalty weight has the following effects:

- decreasing C the margin will grow in size and will allow for more misclassified points
- increasing C the margin will shrink in size and fewer mistakes will be allowed

$\therefore$ increasing the factor C will shrink the margin and fewer mistakes will be allowed.

Solution 6 (a)

∴ The statement: *Each α_i is either 0 or 1.* is **possibly false**. A training point may be misclassified more than once, in which case its corresponding α_i would be incremented multiple times and exceed 1.

Solution 6 (b)

∴ The statement: $\sum_i \alpha_i = k$ is **definitely true**. The algorithm performs k total updates, and each update increases one of the α_i by exactly 1, so the sum of all α_i must equal k.

Solution 6 (c)

∴ The statement: *α has at most k nonzero coordinates.* is **definitely true**. Since only the points that were misclassified at least once are updated, and there are k updates in total, at most k different α_i can be nonzero.

Solution 6 (d)

∴ The statement: *The training data must be linearly separable.* is **definitely true**. The Perceptron algorithm is guaranteed to converge only when the data is linearly separable. Since it converged in k updates, the data must be separable.

Solution 7

Python Code

```
## import libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.utils import shuffle

## import iris dataset
iris = datasets.load_iris()
x = iris.data
y = iris.target

## select data for q7
x_q7 = x[:,[1,3]][np.where(y!=2)]
y_q7 = y[np.where(y!=2)]

## set 0 labels equal to -1
y_q7[y_q7 == 0] = -1

## create binary_perceptron function
def binary_peceptron(w, b, x):
    if np.dot(w, x) + b >= 0:
        label = 1
    else:
        label = -1
    return(label)

## create fit_binary_perceptron function
def fit_binary_perceptron(x, y, track_updates, set_seed):
    if set_seed:
        x, y = shuffle(x, y,random_state=42)
    else:
        x, y = shuffle(x, y)
    w = np.zeros(x.shape[1])
    b = 0
    updates = 0
    max_updates = 1000

    def make_prediction(w, b, x, y, updates):
        error = False
        for xi, yi in zip(x, y):
            if binary_perceptron(w, b, xi) != yi:
                w += yi * xi
                b += yi
                updates += 1
                error = True
        if updates <= max_updates:
            if error:
                return make_prediction(w, b, x, y, updates)
            else:
                return (w, b, updates)
        else:
            print(f"Did not converge after {max_iterations} iterations")
    if track_updates:
        w, b, updates = make_prediction(w, b, x, y, updates)
        return (w, b, updates)
    else:
        w, b, updates = make_prediction(w, b, x, y, updates)
        return (w, b)
```

Plots

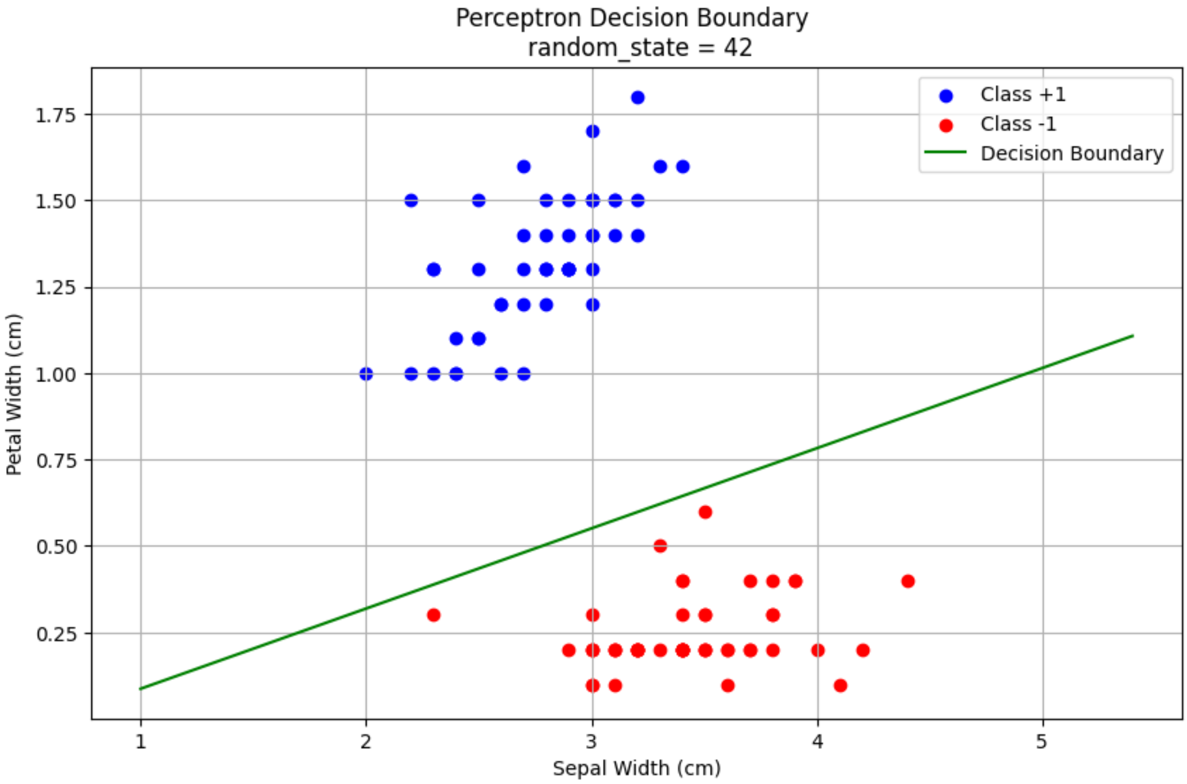


Figure 1: Perceptron Decision Boundary

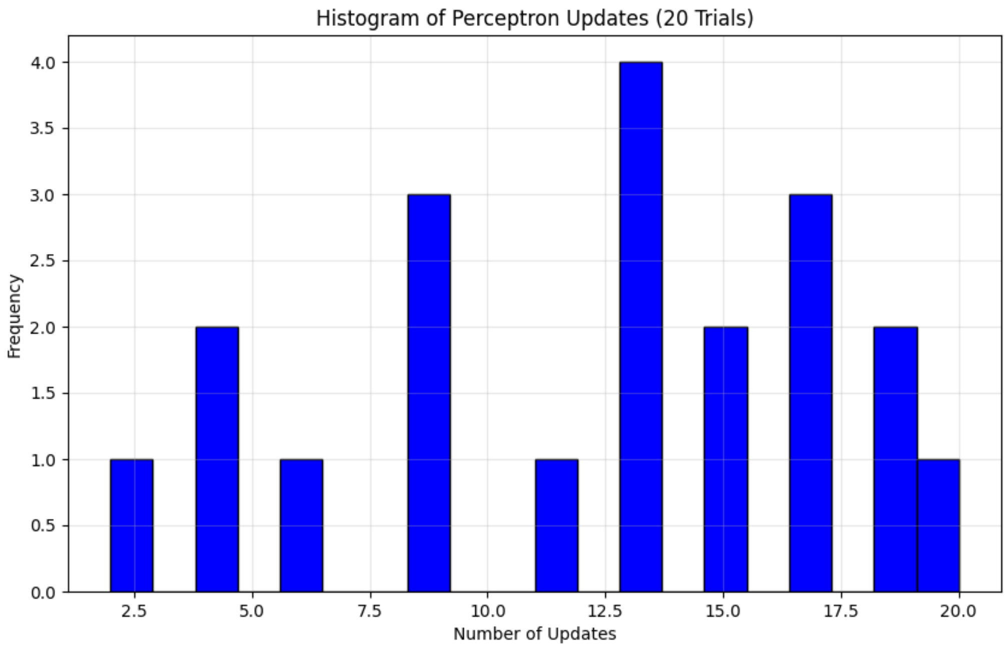


Figure 2: Convergence Plot for 20 trials

Solution 8

Linear Separation

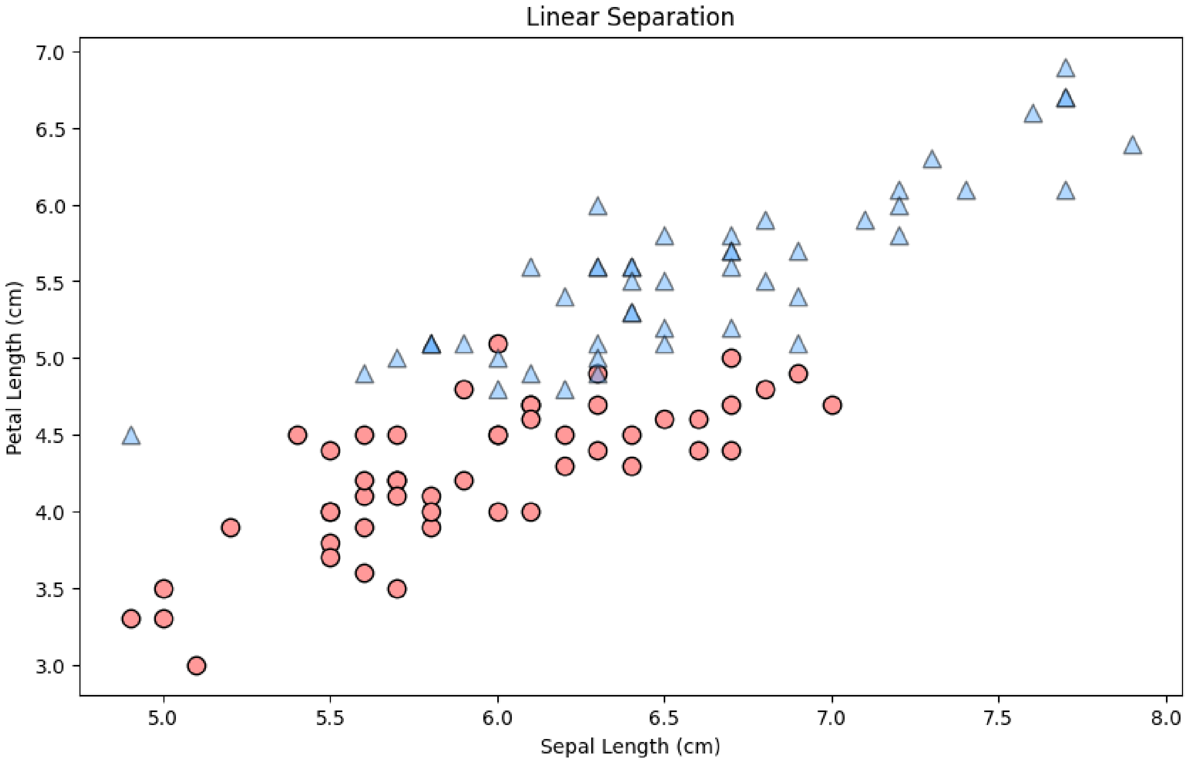


Figure 3: We can not draw a straight line to seperate the two classes, and therefore the data is not linearly separable.

C Value Training Error

C	Training Error	Number of Support Vectors
0.001	0.1700	100
0.01	0.1600	92
0.1	0.0700	56
1	0.0700	31
10	0.0500	18
100	0.0500	14
1000	0.0500	14
10000	0.0500	14
100000	0.0600	14
1000000	0.0700	13

Table 1: SVM results for 10 different C values

Best C Value

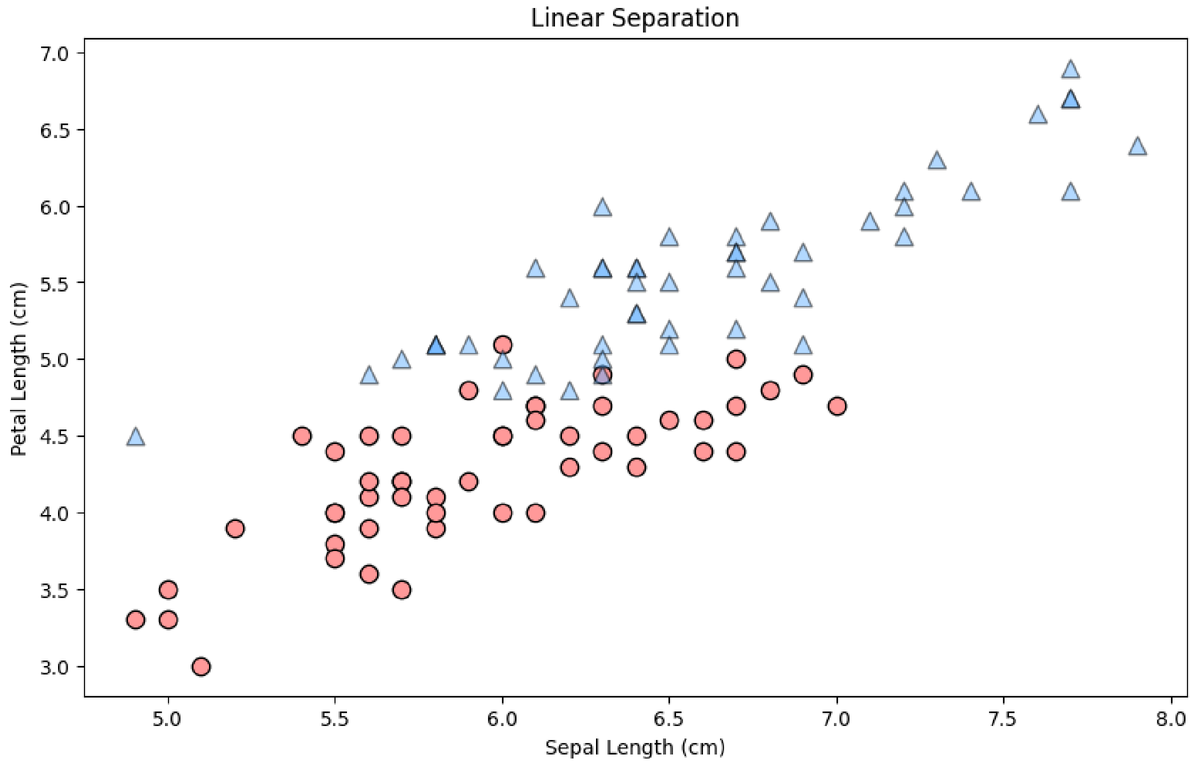


Figure 4: A C value of 10 was selected since it is the lowest C value with the lowest training error of %5.0.