**Contents**

Question 1 -------------------------------------------------------------------------------------------- 2

1. Image of code --------------------------------------------------------------------------------- 2

2. Code ---------------------------------------------------------------------------------------------- 2

3. Test ----------------------------------------------------------------------------------------------- 2

Question 2 -------------------------------------------------------------------------------------------- 3

1. Image of code --------------------------------------------------------------------------------- 3

2. Code ----------------------------------------------------------------------------------------------4

3. Test ----------------------------------------------------------------------------------------------- 5

Question 3 -------------------------------------------------------------------------------------------- 5

1. Image of code --------------------------------------------------------------------------------- 5

2. Code ----------------------------------------------------------------------------------------------6

3. Test ----------------------------------------------------------------------------------------------- 7

Question 3 -------------------------------------------------------------------------------------------- 8

1. Image of code --------------------------------------------------------------------------------- 8

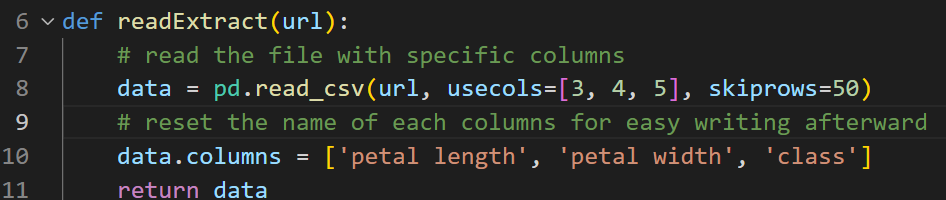
2. Code ----------------------------------------------------------------------------------------------8

3. Test ----------------------------------------------------------------------------------------------- 9

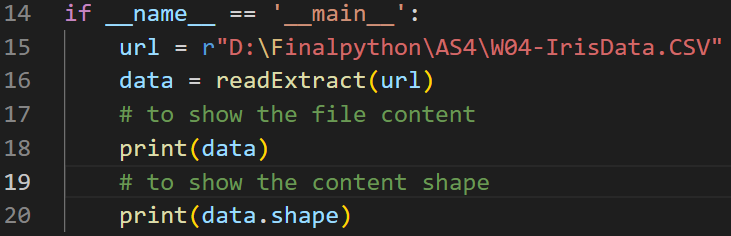
Question 3 -------------------------------------------------------------------------------------------- 10

**Question 1**

1. **Image of code for the better reading experience**
2. Function to read the file and extract the features



1. Main function to test the function



1. **Code details**

def readExtract(url):

# read the file with specific columns

data = pd.read\_csv(url, usecols=[3, 4, 5], skiprows=50)

# reset the name of each columns for easy writing afterward

data.columns = ['petal length', 'petal width', 'class']

return data

if \_\_name\_\_ == '\_\_main\_\_':

url = r"D:\Finalpython\AS4\W04-IrisData.CSV"

data = readExtract(url)

# to show the file content

print(data)

# to show the content shape

print(data.shape)

1. **Test**

The figures below is the result when using main function to test

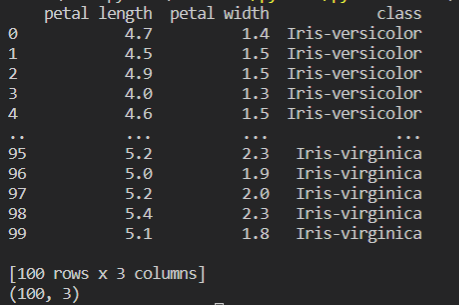
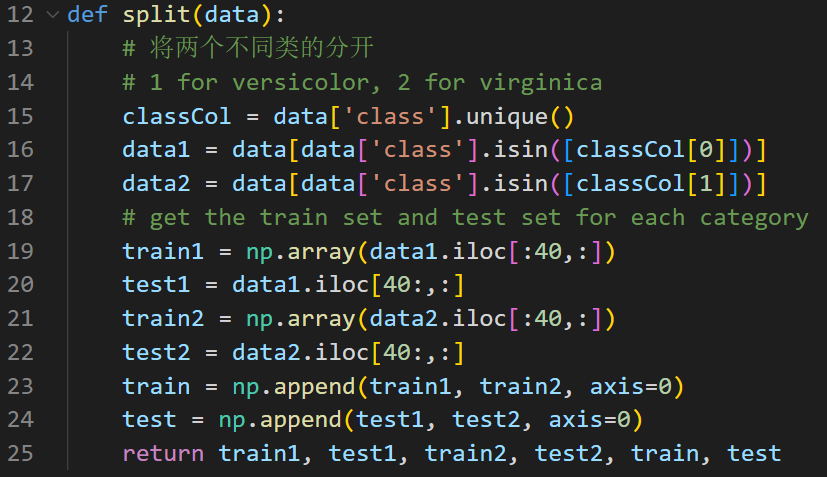


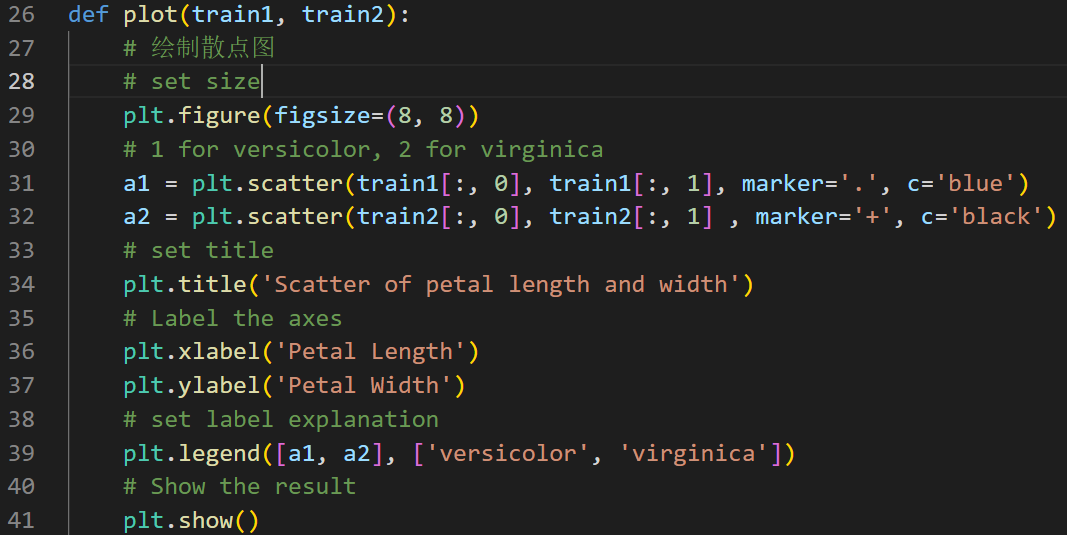
Figure 1: Test result

**Question 2**

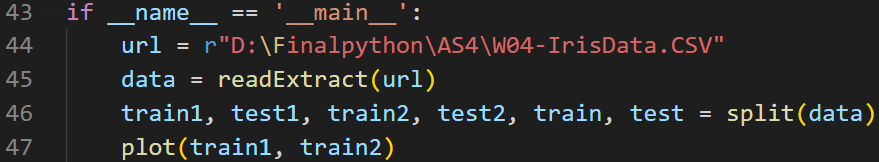
1. **Image of code for the better reading experience**
2. Function to split the data into training set and test set of two different categories.



1. Function to plot distribution of training sets



1. Main function



1. **Code details**

def split(data):

# 1 for versicolor, 2 for virginica

classCol = data['class'].unique()

data1 = data[data['class'].isin([classCol[0]])]

data2 = data[data['class'].isin([classCol[1]])]

# get the train set and test set for each category

train1 = np.array(data1.iloc[:40,:])

test1 = data1.iloc[40:,:]

train2 = np.array(data2.iloc[:40,:])

test2 = data2.iloc[40:,:]

train = np.append(train1, train2, axis=0)

test = np.append(test1, test2, axis=0)

return train1, test1, train2, test2, train, test

def plot(train1, train2):

# set size

plt.figure(figsize=(8, 8))

# 1 for versicolor, 2 for virginica

a1 = plt.scatter(train1[:, 0], train1[:, 1], marker='.', c='blue')

a2 = plt.scatter(train2[:, 0], train2[:, 1] , marker='+', c='black')

# set title

plt.title('Scatter of petal length and width')

# Label the axes

plt.xlabel('Petal Length')

plt.ylabel('Petal Width')

# set label explanation

plt.legend([a1, a2], ['versicolor', 'virginica'])

# Show the result

plt.show()

# Main function

if \_\_name\_\_ == '\_\_main\_\_':

url = r"D:\Finalpython\AS4\W04-IrisData.CSV"

data = readExtract(url)

train1, test1, train2, test2, train, test = split(data)

plot(train1, train2)

1. **Test**

After running the code,

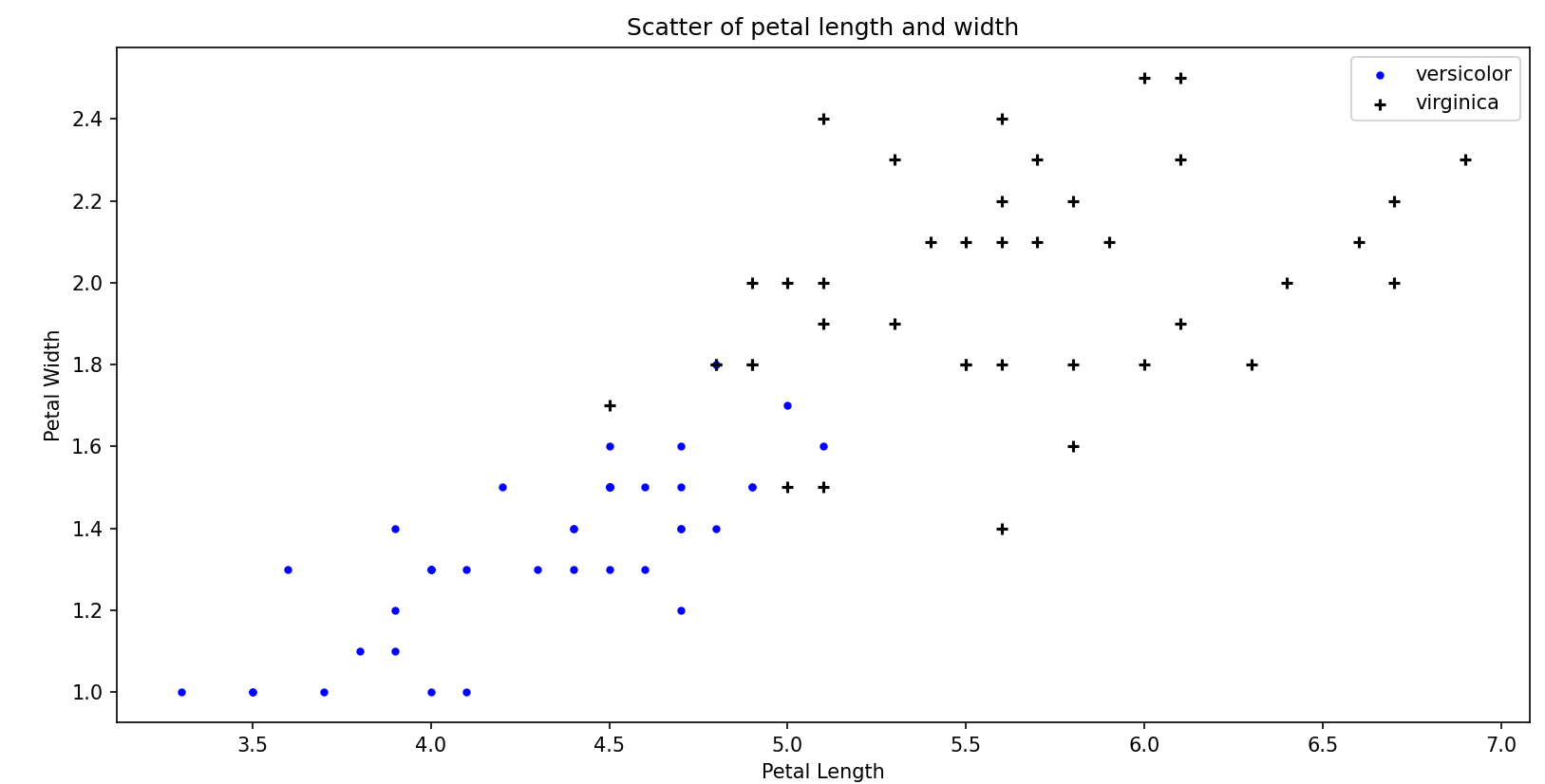
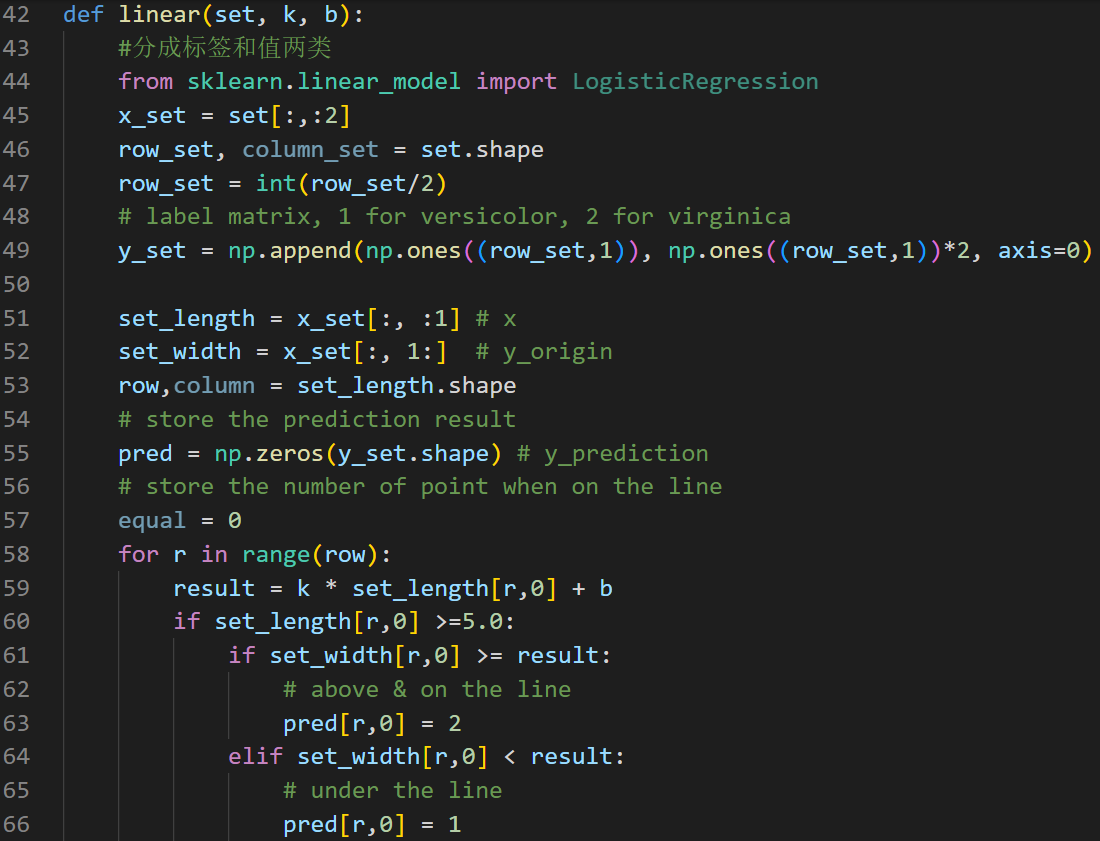
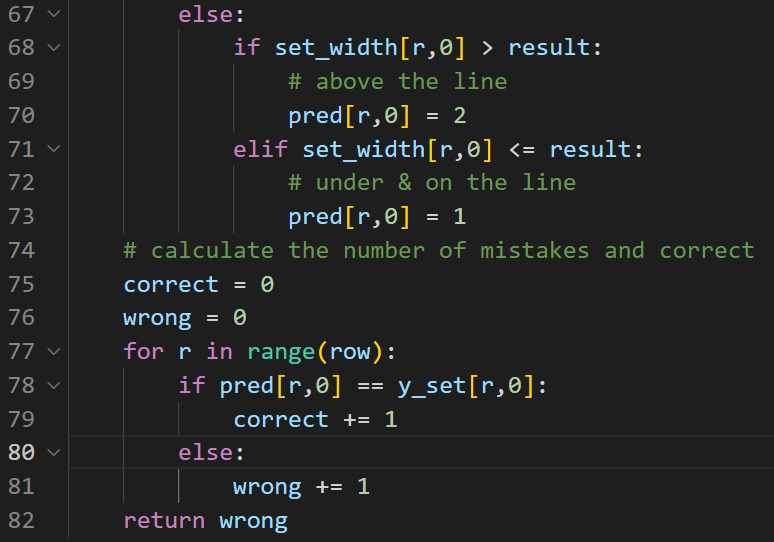


Figure 2: Scatter plot of training set

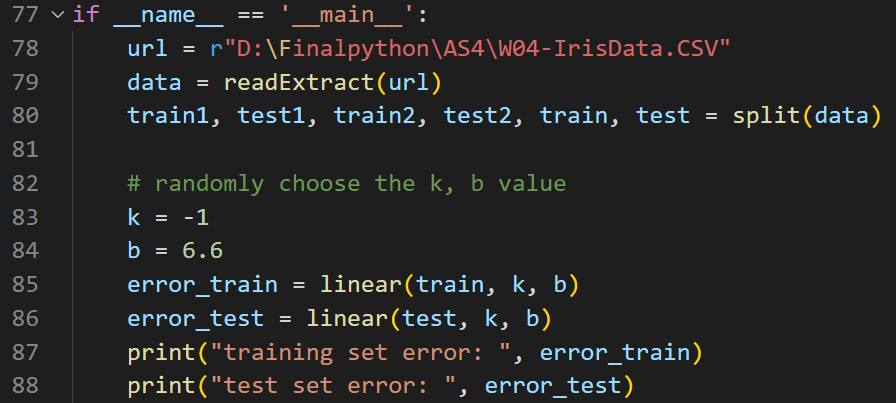
**Question 3**

1. **Image of code for the better reading experience**
2. Function to get the number of error based on given k and b value of a line





1. Main function: randomly given the k and b value as -1 and 6.6



1. **Code details**

def linear(set, k, b):

#分成标签和值两类

from sklearn.linear\_model import LogisticRegression

x\_set = set[:,:2]

row\_set, column\_set = set.shape

row\_set = int(row\_set/2)

# label matrix, 1 for versicolor, 2 for virginica

y\_set = np.append(np.ones((row\_set,1)), np.ones((row\_set,1))\*2, axis=0)

set\_length = x\_set[:, :1] # x

set\_width = x\_set[:, 1:] # y\_origin

row,column = set\_length.shape

# store the prediction result

pred = np.zeros(y\_set.shape) # y\_prediction

# store the number of point when on the line

for r in range(row):

result = k \* set\_length[r,0] + b

if set\_length[r,0] >=5.0:

if set\_width[r,0] >= result:

# above & on the line

pred[r,0] = 2

elif set\_width[r,0] < result:

# under the line

pred[r,0] = 1

else:

if set\_width[r,0] > result:

# above the line

pred[r,0] = 2

elif set\_width[r,0] <= result:

# under & on the line

pred[r,0] = 1

# calculate the number of mistakes and correct

correct = 0

wrong = 0

for r in range(row):

if pred[r,0] == y\_set[r,0]:

correct += 1

else:

wrong += 1

return wrong

if \_\_name\_\_ == '\_\_main\_\_':

url = r"D:\Finalpython\AS4\W04-IrisData.CSV"

data = readExtract(url)

train1, test1, train2, test2, train, test = split(data)

# randomly choose the k, b value

k = -1

b = 6.6

error\_train = linear(train, k, b)

error\_test = linear(test, k, b)

print("training set error: ", error\_train)

print("test set error: ", error\_test)

1. **Test**
2. Decision rule:

Suppose the equation of line is y=kx+b,

Regarding x as petal length value, use the equation to get the prediction of petal width value.

For x >5,

If the original width >= prediction of width, classify it as virginica.

If the original width < prediction of width, classify it as versicolor.

For x<5,

If the original width > prediction of width, classify it as virginica.

If the original width <= prediction of width, classify it as versicolor.

1. Set k as -1 and b as 6.6, the classification errors for training set and test set is,

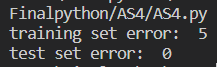


Figure 3: Classification errors results

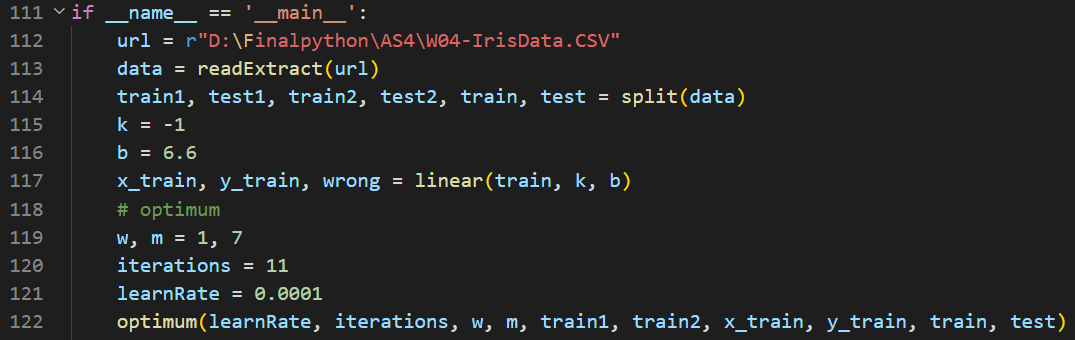
*Mention: The code should be run together with the* ***split*** *function shown in Question 2.*

**Question 4**

1. **Image of code for the better reading experience**
2. Function for optimum



1. Main function: randomly given the start value of **w** and **b** are **1** and 7**,** the **learn rate** is **0.0001** and **iteration times** is **10**.



1. **Code details**

def optimum(learnRate, iterations, w, b, train1, train2, x\_set, y\_set, train, test):

# Plot the figure

# sample

plt.figure(figsize=(8, 8))

a1 = plt.scatter(train1[:, 0], train1[:, 1], marker='.', c='blue')

a2 = plt.scatter(train2[:, 0], train2[:, 1] , marker='+', c='black')

color = ['b','b','b','b','g','r','c','m','y','k']

for i in range(iterations):

w = w - learnRate \* np.sum((y\_set - (b + w \* x\_set)) \* (-x\_set))

b = b - learnRate \* np.sum((y\_set - (b + w \* x\_set)) \* (-1))

temp1, temp2, error\_train = linear(train, w, b)

temp1, temp2, error\_test = linear(test, w, b)

print(i, "time: (", k, ",", b, ")", "train error: ", error\_train, "test error: ", error\_test)

# line

x = np.linspace(2,7, 100)

y = w\*x+b

plt.plot(x, y, c=color[i])

# title

plt.title('Scatter of petal length and width')

# axes

plt.xlabel('Petal Length')

plt.ylabel('Petal Width')

# label

plt.legend([a1, a2], ['versicolor', 'virginica'])

# Show

plt.show()

if \_\_name\_\_ == '\_\_main\_\_':

url = r"D:\Finalpython\AS4\W04-IrisData.CSV"

data = readExtract(url)

train1, test1, train2, test2, train, test = split(data)

k = -1

b = 6.6

x\_train, y\_train, wrong = linear(train, k, b)

# optimum

w, m = 1, 7

iterations = 10

learnRate = 0.0001

optimum(learnRate, iterations, w, m, train1, train2, x\_train, y\_train, train, test)

1. **Test**

From the above blue line to the bottom black line, there are 11 lines and corresponding to the 11 iterations respectively, in Figure 4.

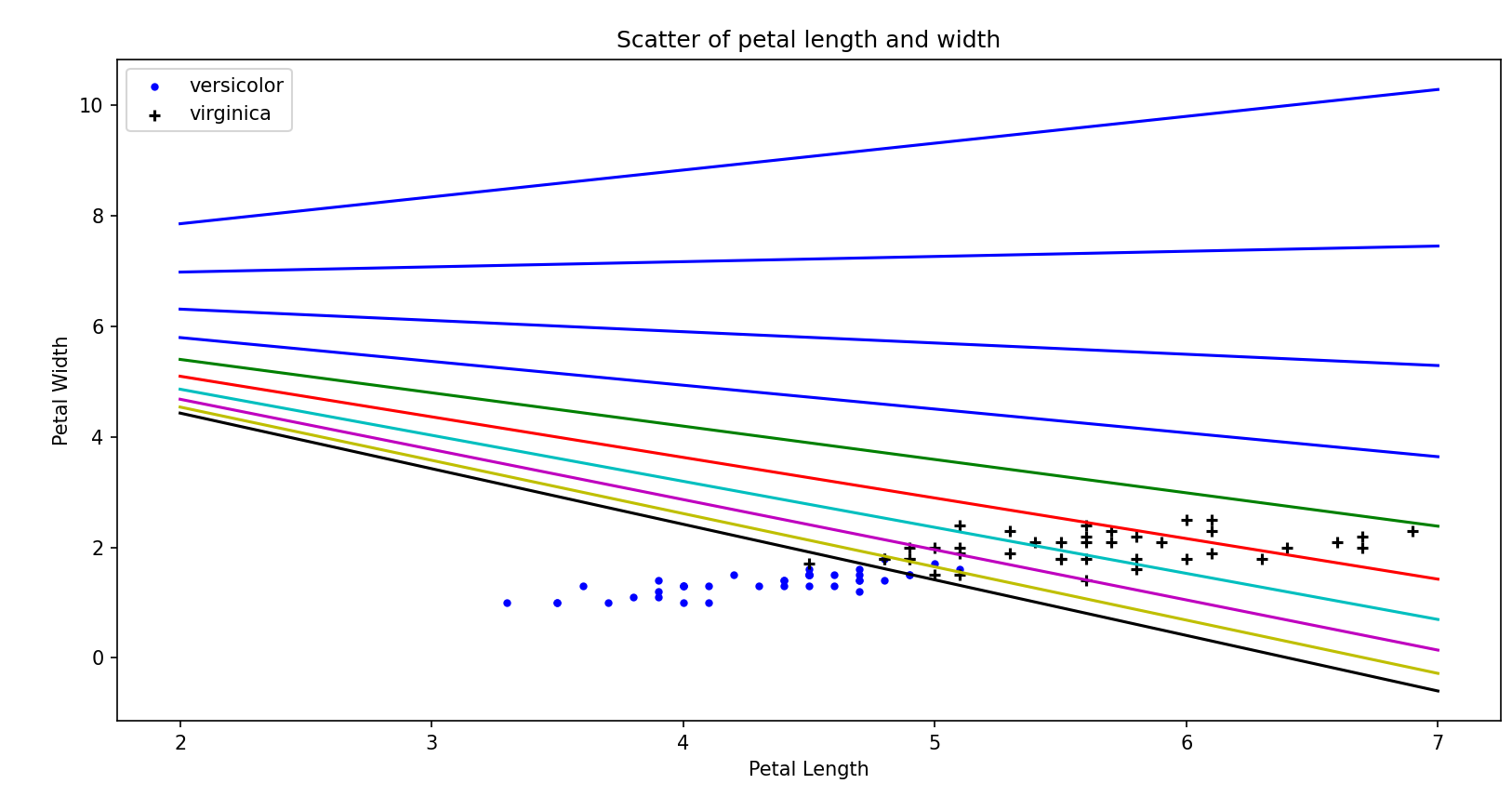


Figure 4: classification lines and sample points

Figure 5 shows the iteration time, collection of k and b value, training error and test error respectively.

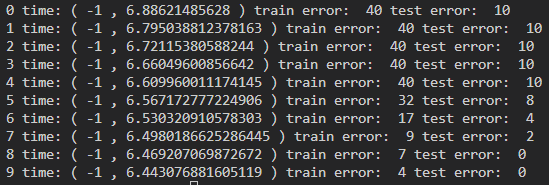


Figure 5: Result

*Mention: The code should be run together with the* ***split*** *function shown in Question 2 and* ***linear*** *function shown in the Question 3.*

**Question 5**

From Figure 5, we can find out that the training error and test error become fewer as the b value becomes smaller. The generalization performance, I think, can be regarded as the relationship of the prediction accuracies of training set and test set. Using the question 4 result as an example, the train error and test error began to decrease when the iteration time is 4 (starting from 0). As the iteration time increasing, these errors decreased which shows the model was underfitting. Generally, the test error will increase while the training error keeps decreasing after a collection of k and b values, which is called overfitting and not appeared in question 4 result. However, in the model of question 4, the test has fewer prediction error than the training set, which is not common. And I think it might because the test set is purposely chosen as the last 10 for each category, which lead to some special value of petal width and petal length.