ÖZYEĞİN UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE

CS 454

2021 Fall

Homework2

Nearest-Mean and Nearest-Neighbor Classification

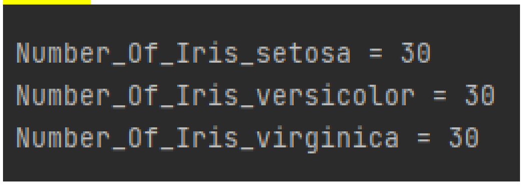
By
Tuğcan Hoşer
S015561

Supervised By
Ethem Alpaydın

PART A

1) First of all, the data in the training.csv file in the project was separated according to their classes. There are 3 classes and each class has 30 data.

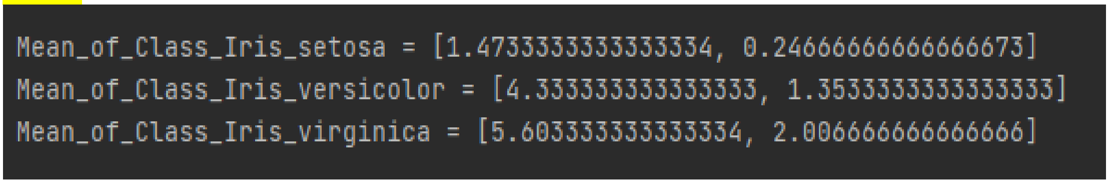
Result:



```
Number_Of_Iris_setosa = 30
Number_Of_Iris_versicolor = 30
Number_Of_Iris_virginica = 30
```

2) Mean of each class were found, first column PetalLengthCm, second column PetalWidthCm.

Result:



```
Mean_of_Class_Iris_setosa = [1.4733333333333334, 0.24666666666666673]
Mean_of_Class_Iris_versicolor = [4.333333333333333, 1.3533333333333333]
Mean_of_Class_Iris_virginica = [5.603333333333334, 2.0066666666666666]
```

Mean Formula:

$$\frac{\Sigma_{i=1}^{n} x_i}{n} = \mu$$

3) The euclidean algorithm was used to calculate their distance to make an estimation with respect to the meaners.

Having the smallest value indicates that the prediction will belong to that class.

Traning set and testing set were tested separately. A column named Prediction has been created.

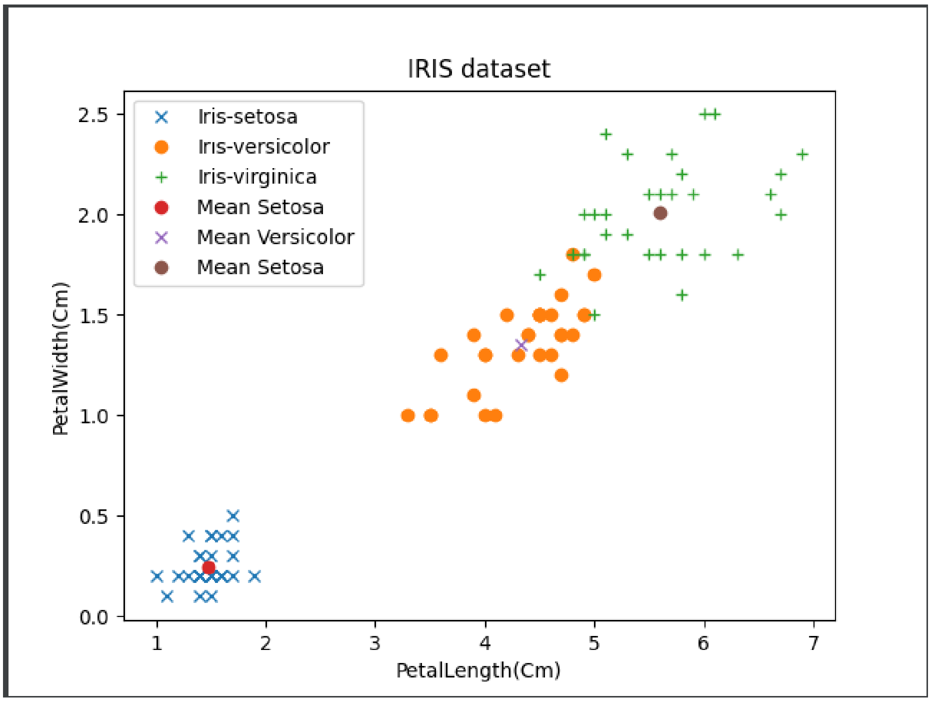
Euclidean Formula:

$$(x_1 - x_2)^2 + (y_1 - y_2)^2$$

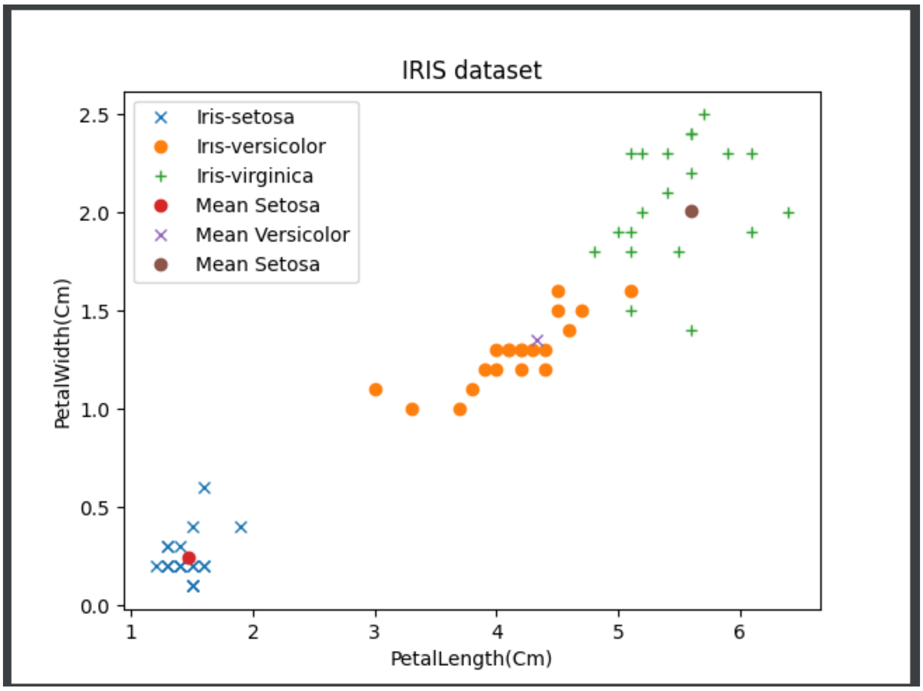
4)

a) Both means were determined using the matplotlib. In addition, traninig set and test set predictions were shown.

For Training Set:

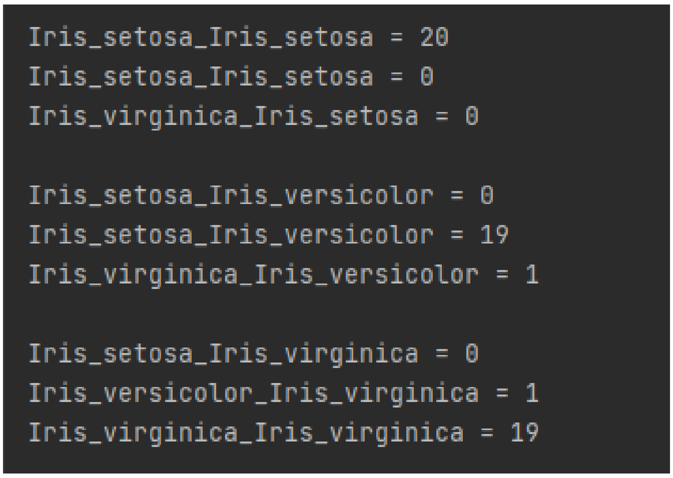


For Testing Set:

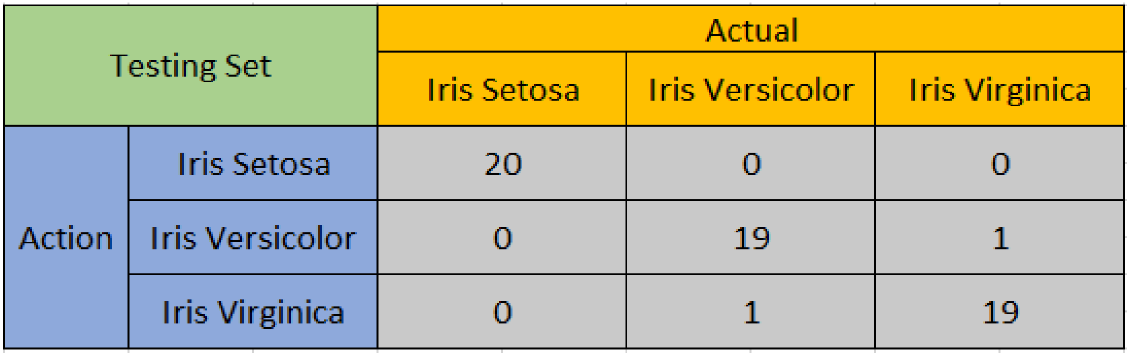


b) Shown in the 3 x 3 confusion matrix chart. Both the training set and the testing set were used.

For Testing Set:

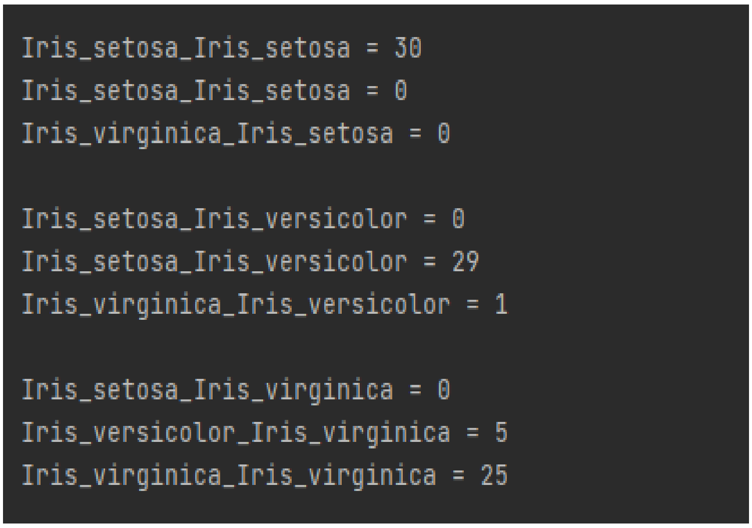


```
Iris_setosa_Iris_setosa = 20
Iris_setosa_Iris_setosa = 0
Iris_virginica_Iris_setosa = 0

Iris_setosa_Iris_versicolor = 0
Iris_setosa_Iris_versicolor = 19
Iris_virginica_Iris_versicolor = 1

Iris_setosa_Iris_virginica = 0
Iris_versicolor_Iris_virginica = 1
Iris_virginica_Iris_virginica = 19
```

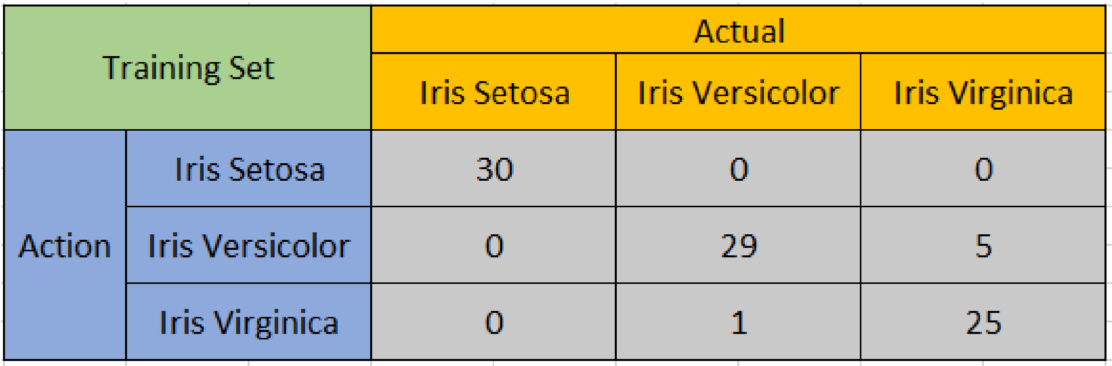


Testing Set		Actual		
		Iris Setosa	Iris Versicolor	Iris Virginica
Action	Iris Setosa	20	0	0
	Iris Versicolor	0	19	1
	Iris Virginica	0	1	19

For Training Set:



```
Iris_setosa_Iris_setosa = 30
Iris_setosa_Iris_setosa = 0
Iris_virginica_Iris_setosa = 0

Iris_setosa_Iris_versicolor = 0
Iris_setosa_Iris_versicolor = 29
Iris_virginica_Iris_versicolor = 1

Iris_setosa_Iris_virginica = 0
Iris_versicolor_Iris_virginica = 5
Iris_virginica_Iris_virginica = 25
```



Training Set		Actual		
		Iris Setosa	Iris Versicolor	Iris Virginica
Action	Iris Setosa	30	0	0
	Iris Versicolor	0	29	5
	Iris Virginica	0	1	25

We see that the model makes misclassifications in both the test set and the training set.

PART B

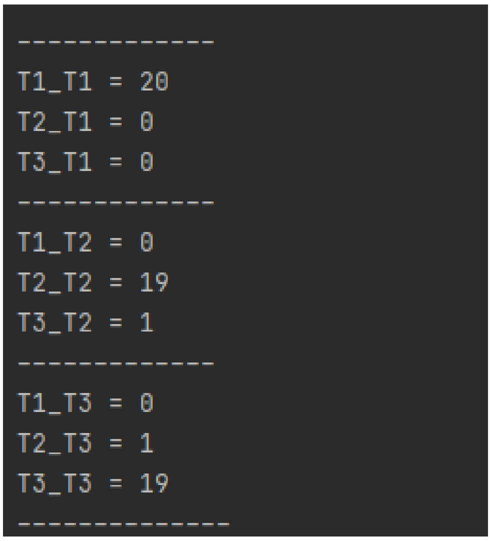
A)

k-Nearest Neighbor classifier

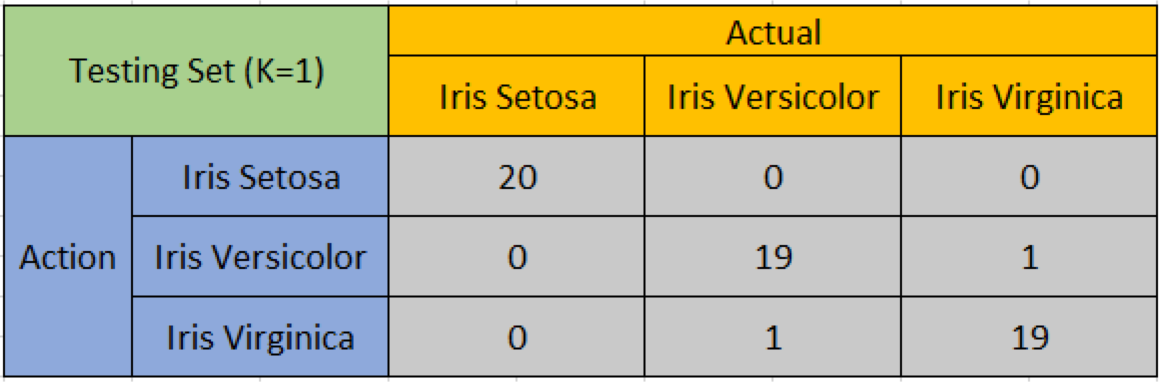
I wrote my sort algorithm myself. I also found the distance using euclid. I found the test set according to k=1, k=3 and k=5.

For Testing Set:

K=1

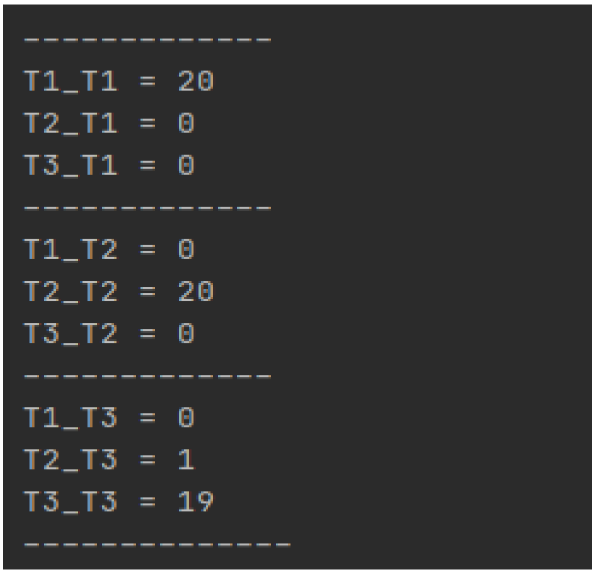


```
-------------
T1_T1 = 20
T2_T1 = 0
T3_T1 = 0
-------------
T1_T2 = 0
T2_T2 = 19
T3_T2 = 1
-------------
T1_T3 = 0
T2_T3 = 1
T3_T3 = 19
--------------
```

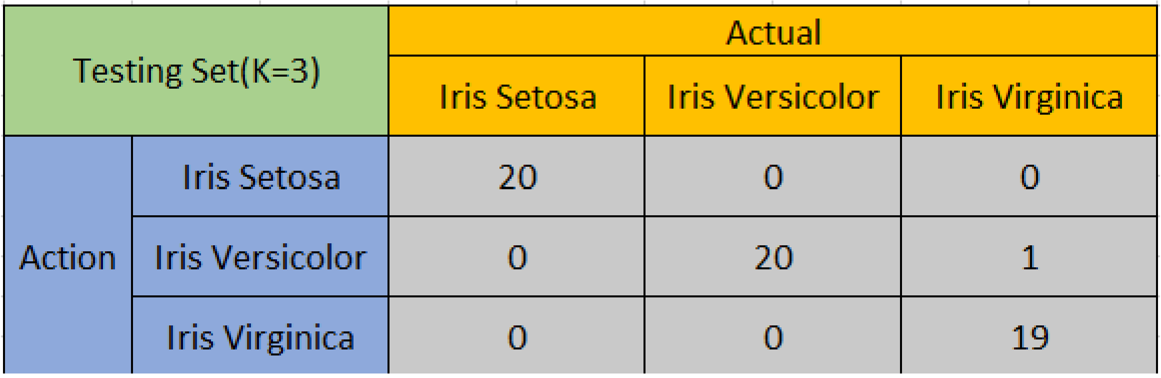


Testing Set (K=1)		Actual		
		Iris Setosa	Iris Versicolor	Iris Virginica
Action	Iris Setosa	20	0	0
	Iris Versicolor	0	19	1
	Iris Virginica	0	1	19

K=3



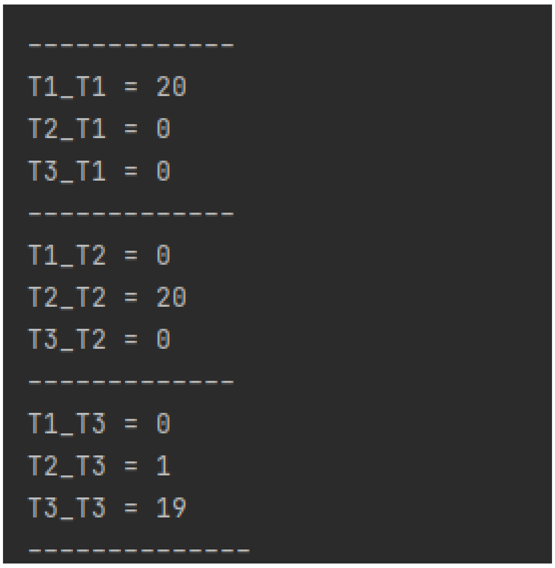
```
-------------
T1_T1 = 20
T2_T1 = 0
T3_T1 = 0
-------------
T1_T2 = 0
T2_T2 = 20
T3_T2 = 0
-------------
T1_T3 = 0
T2_T3 = 1
T3_T3 = 19
--------------
```



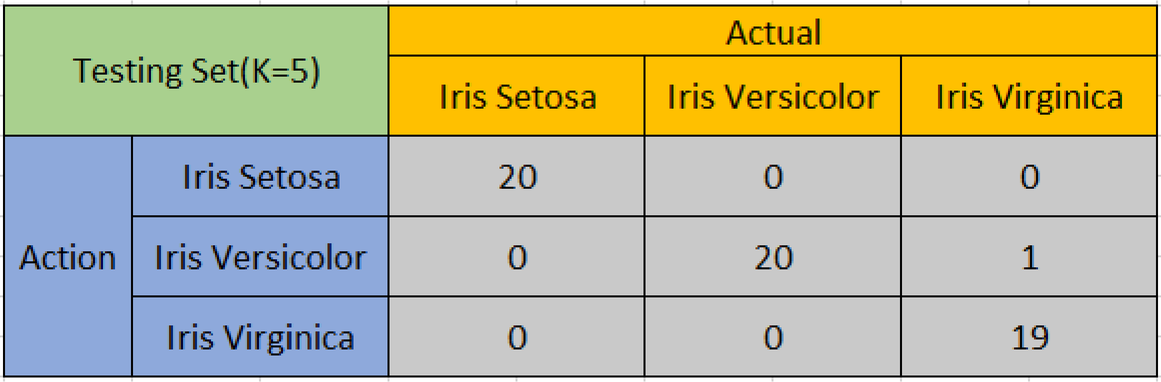
Testing Set(K=3)		Actual		
		Iris Setosa	Iris Versicolor	Iris Virginica
Action	Iris Setosa	20	0	0
	Iris Versicolor	0	20	1
	Iris Virginica	0	0	19

As you can see in the tables, you see some estimation errors. But we got these results because we measured the distance with euclid. More precise results can be achieved with different algorithms.

K=5



```
-------------
T1_T1 = 20
T2_T1 = 0
T3_T1 = 0
-------------
T1_T2 = 0
T2_T2 = 20
T3_T2 = 0
-------------
T1_T3 = 0
T2_T3 = 1
T3_T3 = 19
--------------
```

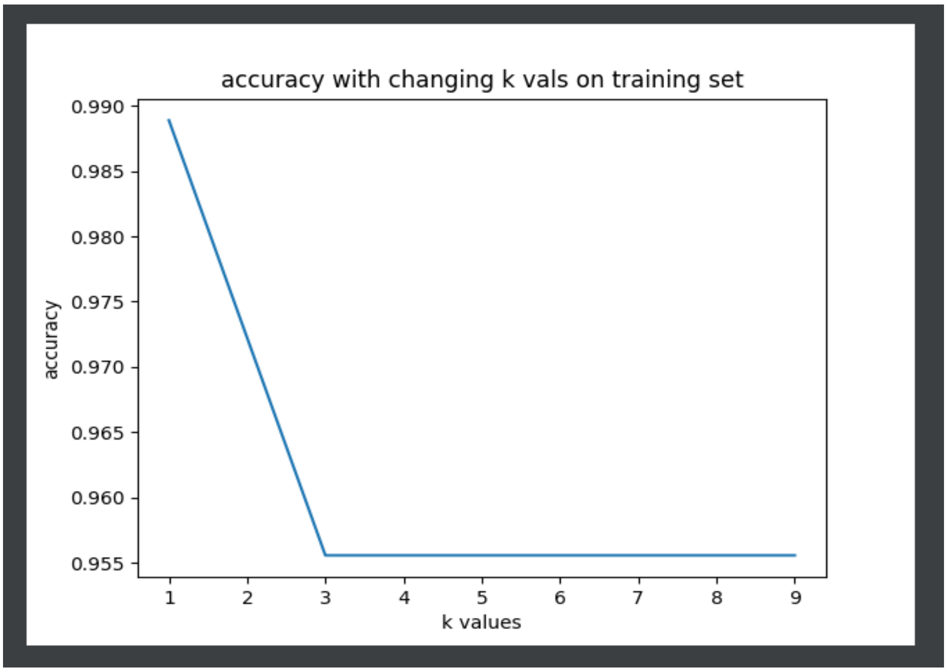


Testing Set(K=5)		Actual		
		Iris Setosa	Iris Versicolor	Iris Virginica
Action	Iris Setosa	20	0	0
	Iris Versicolor	0	20	1
	Iris Virginica	0	0	19

B)

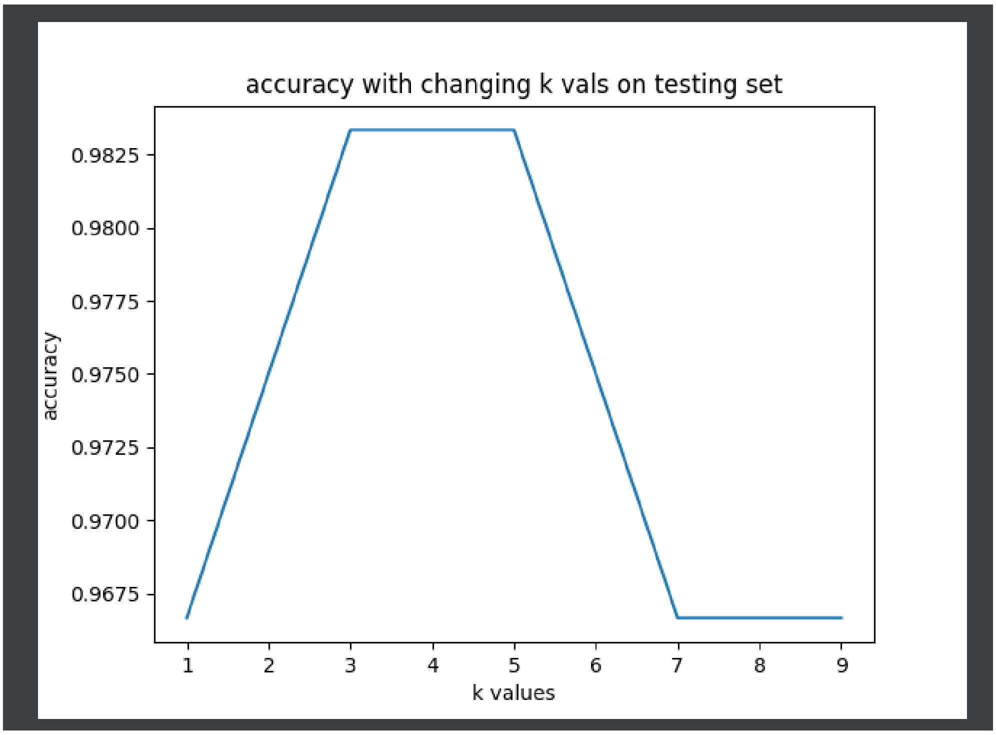
Training Set Accuracies

When k=1, it is at its maximum level and when k=3,5,7,9, the accuracy decreased in estimation.



Testing Set Accuracies

While it has the highest success rate at k=3 and k=5 levels, it has lower values at k=1 , k=7 and k=9 levels.

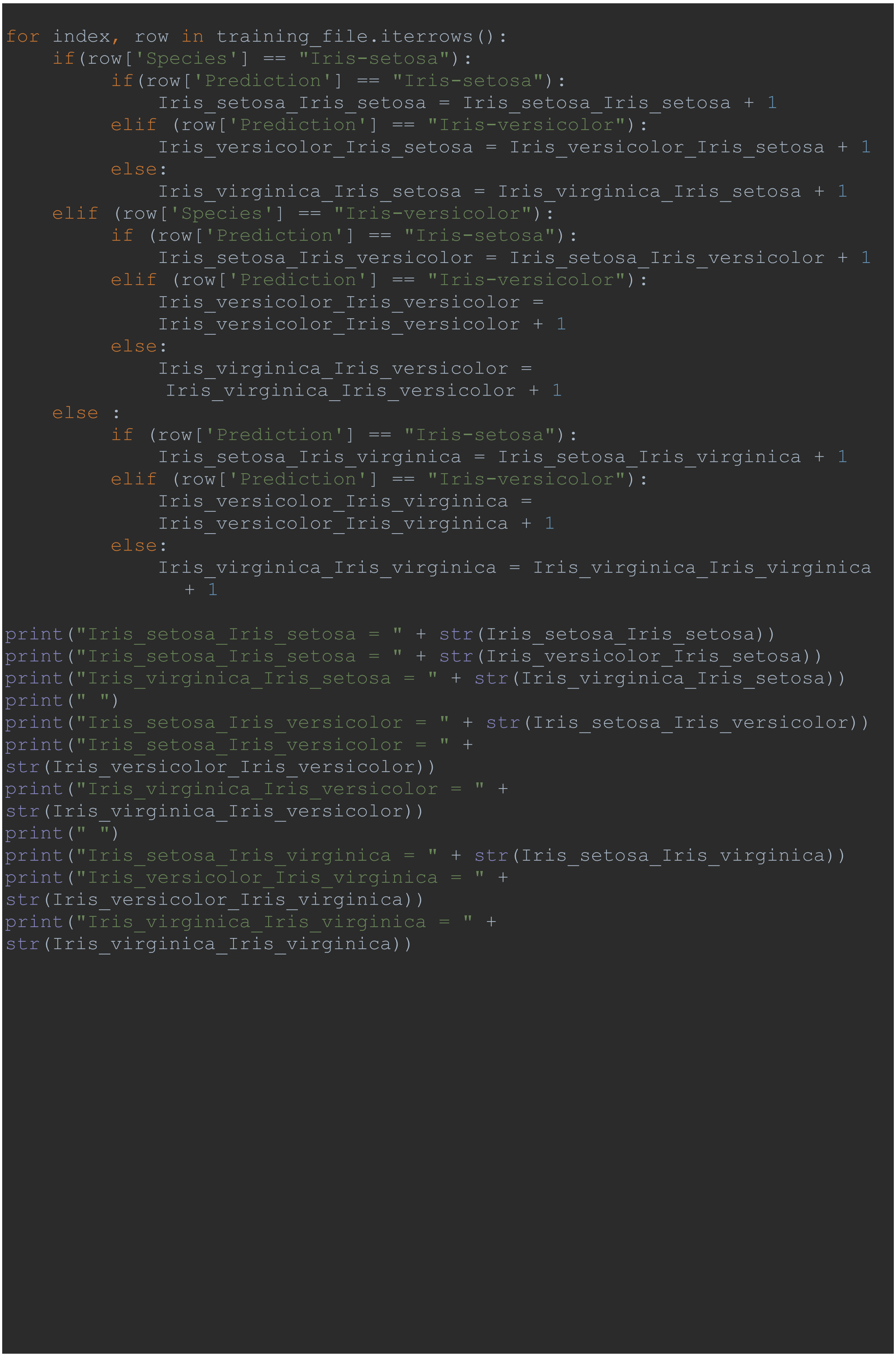


CODING PART

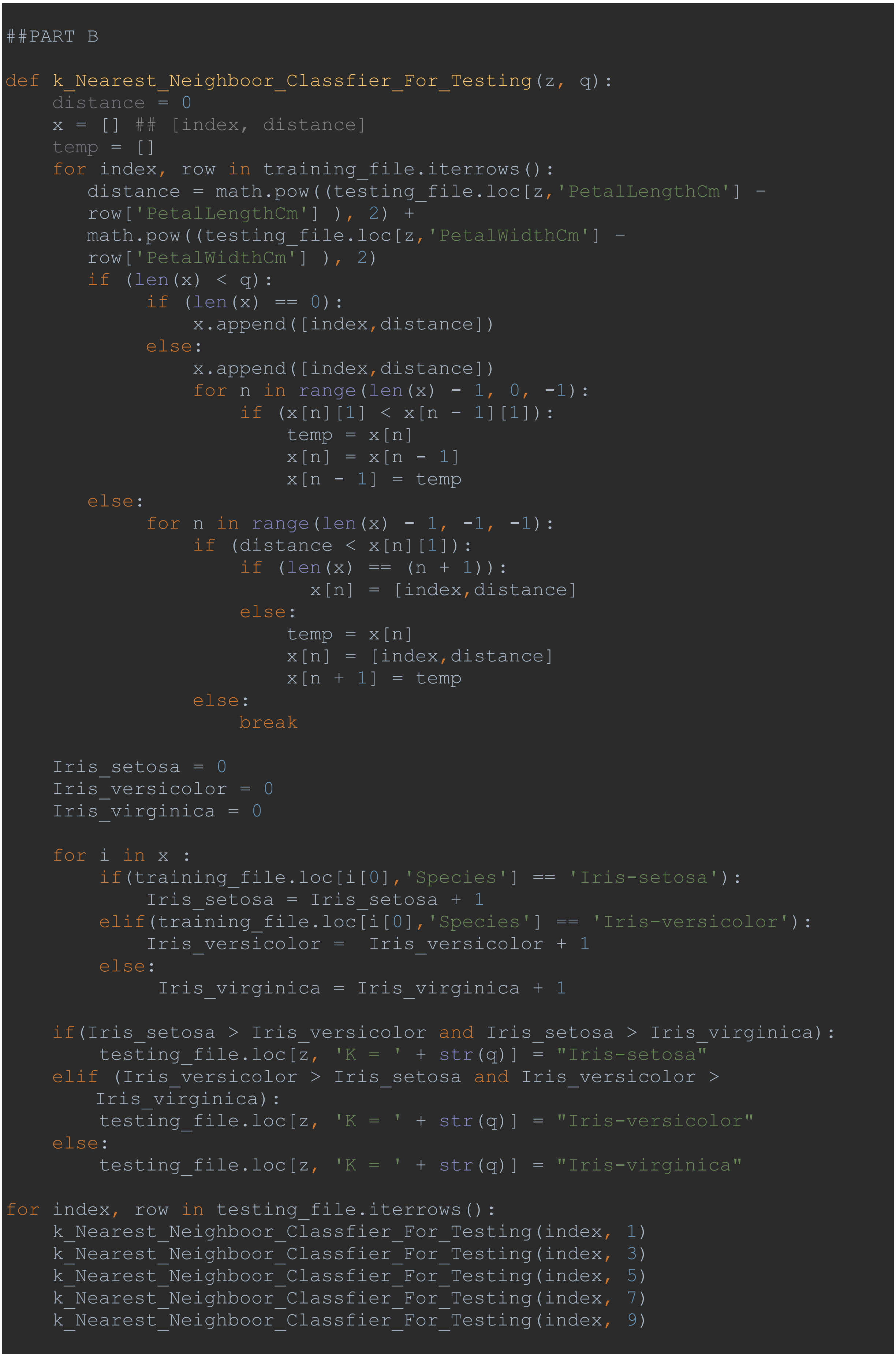
```
import pandas as pd
import math
from matplotlib import pyplot as plt

training_file = pd.read_csv("training.csv")
testing_file = pd.read_csv("testing.csv")

## PART A
## we have 3 classes these are Iris-setosa, Iris-versicolor, Iris-virginica


Number_Of_Iris_setosa = 0
Number_Of_Iris_versicolor = 0
Number_Of_Iris_virginica = 0


for classType in training_file['Species']:
    if classType == 'Iris-setosa':
        Number_Of_Iris_setosa += 1
    elif classType == 'Iris-versicolor':
        Number_Of_Iris_versicolor += 1
    else:
        Number_Of_Iris_virginica += 1

print("")
print("Number_Of_Iris_setosa = " + str(Number_Of_Iris_setosa) )
print("Number_Of_Iris_versicolor = " + str(Number_Of_Iris_versicolor) )
print("Number_Of_Iris_virginica = " + str(Number_Of_Iris_virginica) )
print("")



## Find Mean of Each Classes

def Find_Total_Number_Of_Petal_Length_Width(n):
    p1 = []
    count_x = 0 ;
    count_y = 0 ;
    for index, row in training_file.iterrows():
        if row['Species'] == n:
            count_x += row['PetalLengthCm']
            count_y += row['PetalWidthCm']
    p1.append(count_x);
    p1.append(count_y);
    return p1;


Find_Total_Number_Of_Petal_Length_Width("Iris-setosa")
##[44.2,7.400000000000002]
Find_Total_Number_Of_Petal_Length_Width("Iris-versicolor") ## [130.0, 40.6]
Find_Total_Number_Of_Petal_Length_Width("Iris-virginica") ##
[168.10000000000002, 60.19999999999998]

Mean_of_Class_Iris_setosa = []
Mean_of_Class_Iris_versicolor = []
Mean_of_Class_Iris_virginica = []
```

```
Mean_of_Class_Iris_setosa.append(Find_Total_Number_Of_Petal_Length_Width("I
ris-setosa")[0]/Number_Of_Iris_setosa)
Mean_of_Class_Iris_setosa.append(Find_Total_Number_Of_Petal_Length_Width("I
ris-setosa")[1]/Number_Of_Iris_setosa)

Mean_of_Class_Iris_versicolor.append(Find_Total_Number_Of_Petal_Length_Widt
h("Iris-versicolor")[0]/Number_Of_Iris_versicolor)
Mean_of_Class_Iris_versicolor.append(Find_Total_Number_Of_Petal_Length_Widt
h("Iris-versicolor")[1]/Number_Of_Iris_versicolor)

Mean_of_Class_Iris_virginica.append(Find_Total_Number_Of_Petal_Length_Width
("Iris-virginica")[0]/Number_Of_Iris_virginica)
Mean_of_Class_Iris_virginica.append(Find_Total_Number_Of_Petal_Length_Width
("Iris-virginica")[1]/Number_Of_Iris_virginica)

print("Mean_of_Class_Iris_setosa = " + str(Mean_of_Class_Iris_setosa))
print("Mean_of_Class_Iris_versicolor = " +
str(Mean_of_Class_Iris_versicolor))
print("Mean_of_Class_Iris_virginica = " +
str(Mean_of_Class_Iris_virginica))
print("")



def Euclid_by_means_of_class (x,y):
    Euclid_Iris_setosa = math.pow((x - Mean_of_Class_Iris_setosa[0]), 2) +
    math.pow((y - Mean_of_Class_Iris_setosa[1]), 2)
    Euclid_Iris_versicolor = math.pow((x -
    Mean_of_Class_Iris_versicolor[0]), 2) + math.pow((y -
    Mean_of_Class_Iris_versicolor[1]), 2)
    Euclid_Iris_virginica = math.pow((x - Mean_of_Class_Iris_virginica[0]),
    2) + math.pow((y - Mean_of_Class_Iris_virginica[1]), 2)
    if ((Euclid_Iris_setosa < Euclid_Iris_versicolor) and
        (Euclid_Iris_setosa < Euclid_Iris_versicolor)):
        return 1
    elif ((Euclid_Iris_versicolor < Euclid_Iris_setosa) and
          (Euclid_Iris_versicolor < Euclid_Iris_virginica)):
        return 2
    else :
        return 3

for index, row in testing_file.iterrows():
    if(Euclid_by_means_of_class(row['PetalLengthCm'],
       row['PetalWidthCm'])== 1):
        testing_file.loc[index, 'Prediction'] = "Iris-setosa"
    elif (Euclid_by_means_of_class(row['PetalLengthCm'],
          row['PetalWidthCm']) == 2):
          testing_file.loc[index, 'Prediction']= "Iris-versicolor"
    else:
        testing_file.loc[index, 'Prediction'] = "Iris-virginica"

for index, row in training_file.iterrows():
    if(Euclid_by_means_of_class(row['PetalLengthCm'], row['PetalWidthCm'])
         == 1):
        training_file.loc[index, 'Prediction'] = "Iris-setosa"
    elif (Euclid_by_means_of_class(row['PetalLengthCm'],
        row['PetalWidthCm']) == 2):
        training_file.loc[index, 'Prediction']= "Iris-versicolor"
    else:
        training_file.loc[index, 'Prediction'] = "Iris-virginica"
```

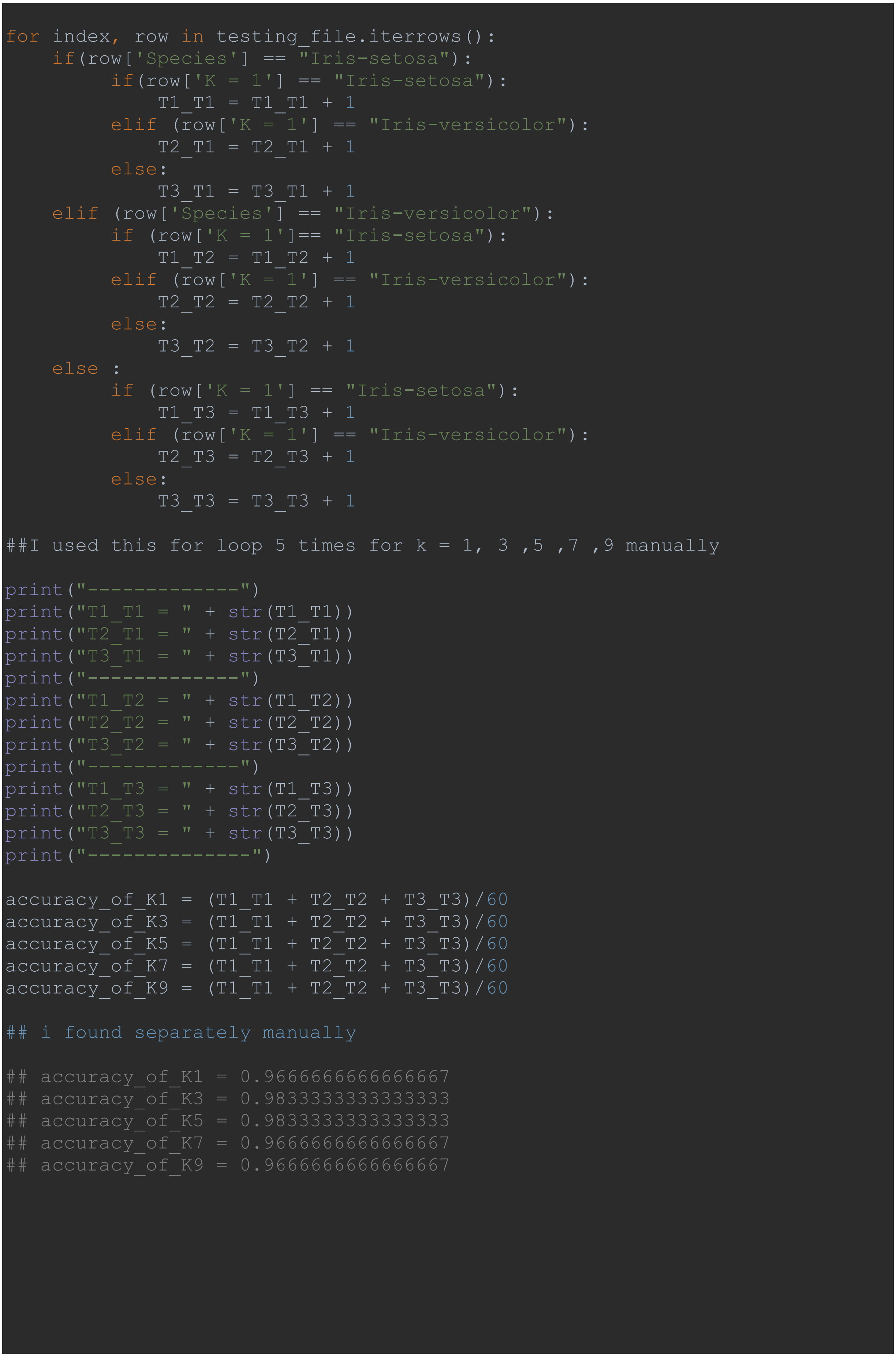
```
plt.plot(testing_file[testing_file['Species'] == "Iris-
setosa"]['PetalLengthCm'] ,testing_file[testing_file['Species'] == "Iris-
setosa"]["PetalWidthCm"]  , 'x' )
plt.plot(testing_file[testing_file['Species'] == "Iris-
versicolor"]['PetalLengthCm'] ,testing_file[testing_file['Species'] ==
"Iris-versicolor"]["PetalWidthCm"]  , 'o' )

plt.plot(testing_file[testing_file['Species'] == "Iris-
virginica"]['PetalLengthCm'] ,testing_file[testing_file['Species'] ==
"Iris-virginica"]["PetalWidthCm"]  , '+' )

plt.plot(training_file[training_file['Species'] == "Iris-
setosa"]['PetalLengthCm'] ,training_file[training_file['Species'] == "Iris-
setosa"]["PetalWidthCm"]  , 'x' )

plt.plot(training_file[training_file['Species'] == "Iris-
versicolor"]['PetalLengthCm'] ,training_file[training_file['Species'] ==
"Iris-versicolor"]["PetalWidthCm"]  , 'o' )

plt.plot(training_file[training_file['Species'] == "Iris-
virginica"]['PetalLengthCm'] ,training_file[training_file['Species'] ==
"Iris-virginica"]["PetalWidthCm"]  , '+' )



plt.plot(Mean_of_Class_Iris_setosa[0],Mean_of_Class_Iris_setosa[1], 'o')
plt.plot(Mean_of_Class_Iris_versicolor[0],Mean_of_Class_Iris_versicolor[1]
, 'x')
plt.plot(Mean_of_Class_Iris_virginica[0],Mean_of_Class_Iris_virginica[1] ,
'o')
plt.legend(['Iris-setosa','Irıs-versicolor','Iris-virginica','Mean
Setosa','Mean Versicolor','Mean Setosa'])
plt.title('IRIS dataset')
plt.xlabel('PetalLength(Cm)')
plt.ylabel('PetalWidth(Cm)')
plt.show()


Iris_setosa_Iris_setosa = 0
Iris_versicolor_Iris_setosa = 0
Iris_virginica_Iris_setosa = 0

Iris_setosa_Iris_versicolor = 0
Iris_versicolor_Iris_versicolor = 0
Iris_virginica_Iris_versicolor = 0

Iris_setosa_Iris_virginica = 0
Iris_versicolor_Iris_virginica = 0
Iris_virginica_Iris_virginica = 0
```



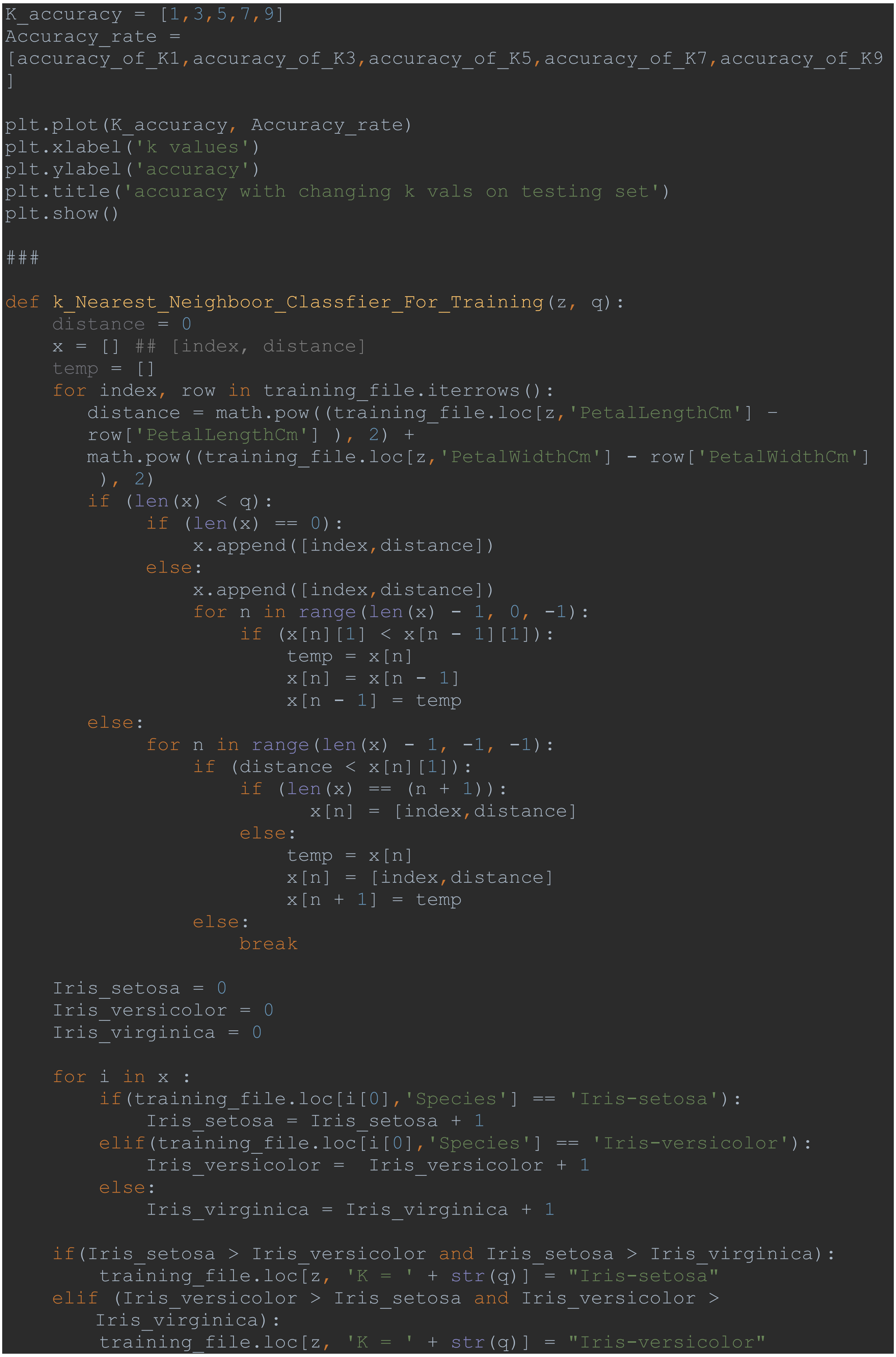
```
for index, row in training_file.iterrows():
    if(row['Species'] == "Iris-setosa"):
         if(row['Prediction'] == "Iris-setosa"):
             Iris_setosa_Iris_setosa = Iris_setosa_Iris_setosa + 1
         elif (row['Prediction'] == "Iris-versicolor"):
             Iris_versicolor_Iris_setosa = Iris_versicolor_Iris_setosa + 1
         else:
             Iris_virginica_Iris_setosa = Iris_virginica_Iris_setosa + 1
    elif (row['Species'] == "Iris-versicolor"):
         if (row['Prediction'] == "Iris-setosa"):
             Iris_setosa_Iris_versicolor = Iris_setosa_Iris_versicolor + 1
         elif (row['Prediction'] == "Iris-versicolor"):
             Iris_versicolor_Iris_versicolor =
             Iris_versicolor_Iris_versicolor + 1
         else:
             Iris_virginica_Iris_versicolor =
              Iris_virginica_Iris_versicolor + 1
    else :
         if (row['Prediction'] == "Iris-setosa"):
             Iris_setosa_Iris_virginica = Iris_setosa_Iris_virginica + 1
         elif (row['Prediction'] == "Iris-versicolor"):
             Iris_versicolor_Iris_virginica =
             Iris_versicolor_Iris_virginica + 1
         else:
             Iris_virginica_Iris_virginica = Iris_virginica_Iris_virginica
               + 1

print("Iris_setosa_Iris_setosa = " + str(Iris_setosa_Iris_setosa))
print("Iris_setosa_Iris_setosa = " + str(Iris_versicolor_Iris_setosa))
print("Iris_virginica_Iris_setosa = " + str(Iris_virginica_Iris_setosa))
print(" ")
print("Iris_setosa_Iris_versicolor = " + str(Iris_setosa_Iris_versicolor))
print("Iris_setosa_Iris_versicolor = " +
str(Iris_versicolor_Iris_versicolor))
print("Iris_virginica_Iris_versicolor = " +
str(Iris_virginica_Iris_versicolor))
print(" ")
print("Iris_setosa_Iris_virginica = " + str(Iris_setosa_Iris_virginica))
print("Iris_versicolor_Iris_virginica = " +
str(Iris_versicolor_Iris_virginica))
print("Iris_virginica_Iris_virginica = " +
str(Iris_virginica_Iris_virginica))
```



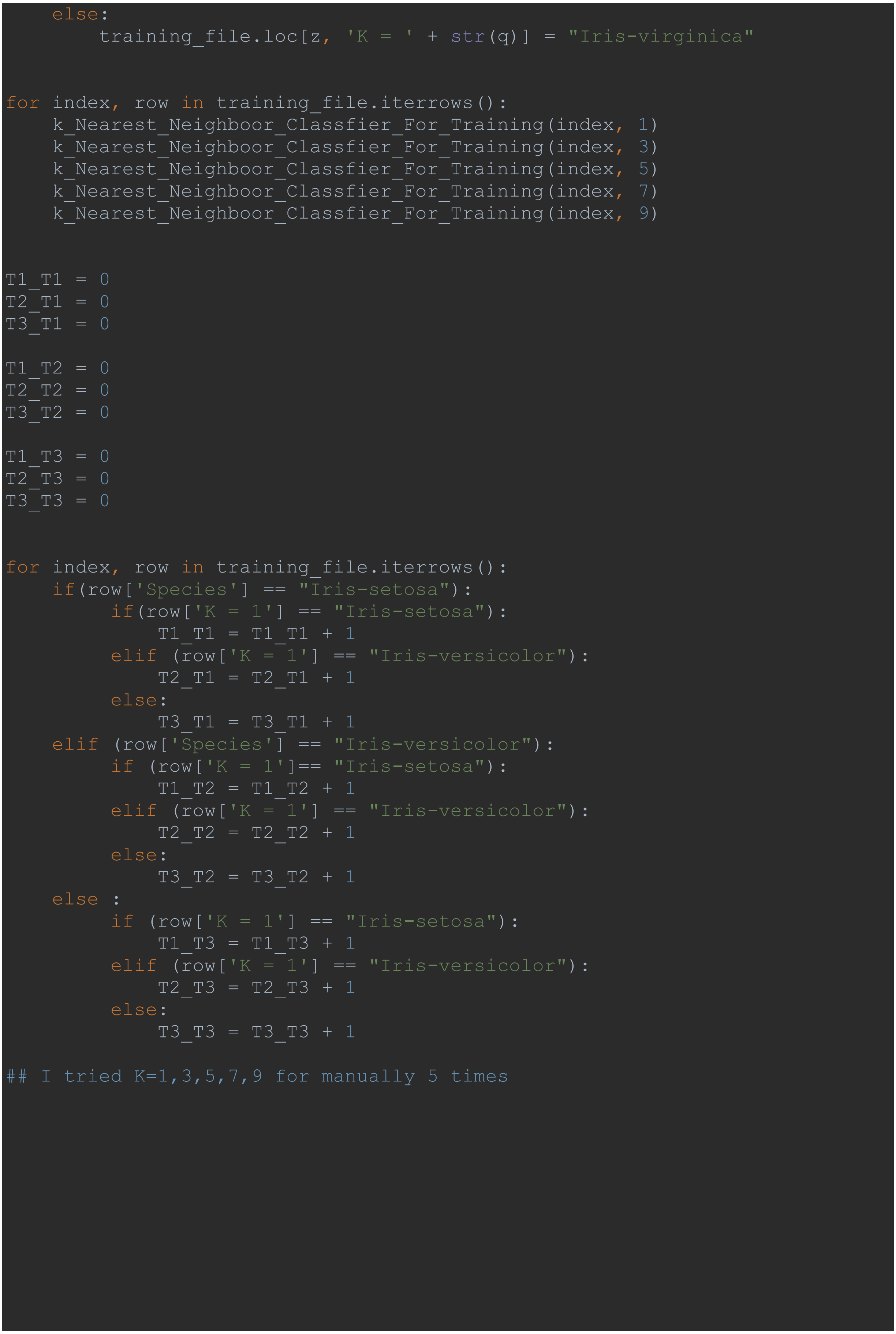
```
##PART B

def k_Nearest_Neighboor_Classfier_For_Testing(z, q):
    distance = 0
    x = [] ## [index, distance]
    temp = []
    for index, row in training_file.iterrows():
       distance = math.pow((testing_file.loc[z,'PetalLengthCm'] -
       row['PetalLengthCm'] ), 2) +
       math.pow((testing_file.loc[z,'PetalWidthCm'] -
       row['PetalWidthCm'] ), 2)
       if (len(x) < q):
            if (len(x) == 0):
                x.append([index,distance])
            else:
                x.append([index,distance])
                for n in range(len(x) - 1, 0, -1):
                    if (x[n][1] < x[n - 1][1]):
                        temp = x[n]
                        x[n] = x[n - 1]
                        x[n - 1] = temp
       else:
            for n in range(len(x) - 1, -1, -1):
                if (distance < x[n][1]):
                    if (len(x) == (n + 1)):
                          x[n] = [index,distance]
                    else:
                        temp = x[n]
                        x[n] = [index,distance]
                        x[n + 1] = temp
                else:
                    break

    Iris_setosa = 0
    Iris_versicolor = 0
    Iris_virginica = 0

    for i in x :
        if(training_file.loc[i[0],'Species'] == 'Iris-setosa'):
            Iris_setosa = Iris_setosa + 1
        elif(training_file.loc[i[0],'Species'] == 'Iris-versicolor'):
            Iris_versicolor =  Iris_versicolor + 1
        else:
             Iris_virginica = Iris_virginica + 1

    if(Iris_setosa > Iris_versicolor and Iris_setosa > Iris_virginica):
        testing_file.loc[z, 'K = ' + str(q)] = "Iris-setosa"
    elif (Iris_versicolor > Iris_setosa and Iris_versicolor >
        Iris_virginica):
        testing_file.loc[z, 'K = ' + str(q)] = "Iris-versicolor"
    else:
        testing_file.loc[z, 'K = ' + str(q)] = "Iris-virginica"

for index, row in testing_file.iterrows():
    k_Nearest_Neighboor_Classfier_For_Testing(index, 1)
    k_Nearest_Neighboor_Classfier_For_Testing(index, 3)
    k_Nearest_Neighboor_Classfier_For_Testing(index, 5)
    k_Nearest_Neighboor_Classfier_For_Testing(index, 7)
    k_Nearest_Neighboor_Classfier_For_Testing(index, 9)
```



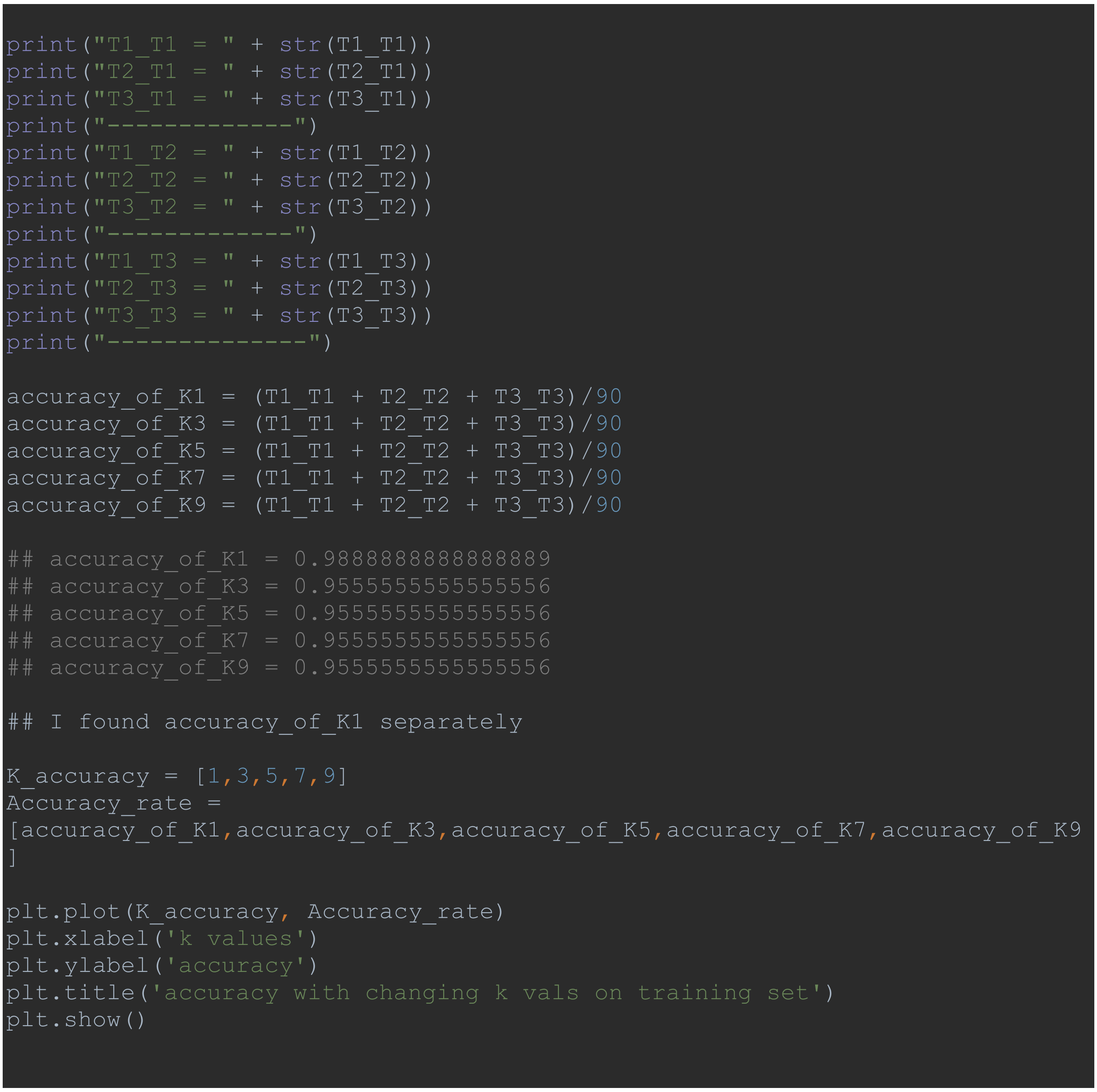
```
for index, row in testing_file.iterrows():
    if(row['Species'] == "Iris-setosa"):
         if(row['K = 1'] == "Iris-setosa"):
             T1_T1 = T1_T1 + 1
         elif (row['K = 1'] == "Iris-versicolor"):
             T2_T1 = T2_T1 + 1
         else:
             T3_T1 = T3_T1 + 1
    elif (row['Species'] == "Iris-versicolor"):
         if (row['K = 1']== "Iris-setosa"):
             T1_T2 = T1_T2 + 1
         elif (row['K = 1'] == "Iris-versicolor"):
             T2_T2 = T2_T2 + 1
         else:
             T3_T2 = T3_T2 + 1
    else :
         if (row['K = 1'] == "Iris-setosa"):
             T1_T3 = T1_T3 + 1
         elif (row['K = 1'] == "Iris-versicolor"):
             T2_T3 = T2_T3 + 1
         else:
             T3_T3 = T3_T3 + 1

##I used this for loop 5 times for k = 1, 3 ,5 ,7 ,9 manually

print("-------------")
print("T1_T1 = " + str(T1_T1))
print("T2_T1 = " + str(T2_T1))
print("T3_T1 = " + str(T3_T1))
print("-------------")
print("T1_T2 = " + str(T1_T2))
print("T2_T2 = " + str(T2_T2))
print("T3_T2 = " + str(T3_T2))
print("-------------")
print("T1_T3 = " + str(T1_T3))
print("T2_T3 = " + str(T2_T3))
print("T3_T3 = " + str(T3_T3))
print("--------------")

accuracy_of_K1 = (T1_T1 + T2_T2 + T3_T3)/60
accuracy_of_K3 = (T1_T1 + T2_T2 + T3_T3)/60
accuracy_of_K5 = (T1_T1 + T2_T2 + T3_T3)/60
accuracy_of_K7 = (T1_T1 + T2_T2 + T3_T3)/60
accuracy_of_K9 = (T1_T1 + T2_T2 + T3_T3)/60

## i found separately manually

## accuracy_of_K1 = 0.9666666666666667
## accuracy_of_K3 = 0.9833333333333333
## accuracy_of_K5 = 0.9833333333333333
## accuracy_of_K7 = 0.9666666666666667
## accuracy_of_K9 = 0.9666666666666667
```



```
K_accuracy = [1,3,5,7,9]
Accuracy_rate =
[accuracy_of_K1,accuracy_of_K3,accuracy_of_K5,accuracy_of_K7,accuracy_of_K9
]

plt.plot(K_accuracy, Accuracy_rate)
plt.xlabel('k values')
plt.ylabel('accuracy')
plt.title('accuracy with changing k vals on testing set')
plt.show()

###

def k_Nearest_Neighboor_Classfier_For_Training(z, q):
    distance = 0
    x = [] ## [index, distance]
    temp = []
    for index, row in training_file.iterrows():
       distance = math.pow((training_file.loc[z,'PetalLengthCm'] -
       row['PetalLengthCm'] ), 2) +
       math.pow((training_file.loc[z,'PetalWidthCm'] - row['PetalWidthCm']
        ), 2)
       if (len(x) < q):
            if (len(x) == 0):
                x.append([index,distance])
            else:
                x.append([index,distance])
                for n in range(len(x) - 1, 0, -1):
                    if (x[n][1] < x[n - 1][1]):
                        temp = x[n]
                        x[n] = x[n - 1]
                        x[n - 1] = temp
       else:
            for n in range(len(x) - 1, -1, -1):
                if (distance < x[n][1]):
                    if (len(x) == (n + 1)):
                          x[n] = [index,distance]
                    else:
                        temp = x[n]
                        x[n] = [index,distance]
                        x[n + 1] = temp
                else:
                    break

    Iris_setosa = 0
    Iris_versicolor = 0
    Iris_virginica = 0

    for i in x :
        if(training_file.loc[i[0],'Species'] == 'Iris-setosa'):
            Iris_setosa = Iris_setosa + 1
        elif(training_file.loc[i[0],'Species'] == 'Iris-versicolor'):
            Iris_versicolor =  Iris_versicolor + 1
        else:
            Iris_virginica = Iris_virginica + 1

    if(Iris_setosa > Iris_versicolor and Iris_setosa > Iris_virginica):
        training_file.loc[z, 'K = ' + str(q)] = "Iris-setosa"
    elif (Iris_versicolor > Iris_setosa and Iris_versicolor >
        Iris_virginica):
        training_file.loc[z, 'K = ' + str(q)] = "Iris-versicolor"
```



```
    else:
        training_file.loc[z, 'K = ' + str(q)] = "Iris-virginica"


for index, row in training_file.iterrows():
    k_Nearest_Neighboor_Classfier_For_Training(index, 1)
    k_Nearest_Neighboor_Classfier_For_Training(index, 3)
    k_Nearest_Neighboor_Classfier_For_Training(index, 5)
    k_Nearest_Neighboor_Classfier_For_Training(index, 7)
    k_Nearest_Neighboor_Classfier_For_Training(index, 9)


T1_T1 = 0
T2_T1 = 0
T3_T1 = 0

T1_T2 = 0
T2_T2 = 0
T3_T2 = 0

T1_T3 = 0
T2_T3 = 0
T3_T3 = 0


for index, row in training_file.iterrows():
    if(row['Species'] == "Iris-setosa"):
         if(row['K = 1'] == "Iris-setosa"):
             T1_T1 = T1_T1 + 1
         elif (row['K = 1'] == "Iris-versicolor"):
             T2_T1 = T2_T1 + 1
         else:
             T3_T1 = T3_T1 + 1
    elif (row['Species'] == "Iris-versicolor"):
         if (row['K = 1']== "Iris-setosa"):
             T1_T2 = T1_T2 + 1
         elif (row['K = 1'] == "Iris-versicolor"):
             T2_T2 = T2_T2 + 1
         else:
             T3_T2 = T3_T2 + 1
    else :
         if (row['K = 1'] == "Iris-setosa"):
             T1_T3 = T1_T3 + 1
         elif (row['K = 1'] == "Iris-versicolor"):
             T2_T3 = T2_T3 + 1
         else:
             T3_T3 = T3_T3 + 1

## I tried K=1,3,5,7,9 for manually 5 times
```



```
print("T1_T1 = " + str(T1_T1))
print("T2_T1 = " + str(T2_T1))
print("T3_T1 = " + str(T3_T1))
print("-------------")
print("T1_T2 = " + str(T1_T2))
print("T2_T2 = " + str(T2_T2))
print("T3_T2 = " + str(T3_T2))
print("-------------")
print("T1_T3 = " + str(T1_T3))
print("T2_T3 = " + str(T2_T3))
print("T3_T3 = " + str(T3_T3))
print("--------------")

accuracy_of_K1 = (T1_T1 + T2_T2 + T3_T3)/90
accuracy_of_K3 = (T1_T1 + T2_T2 + T3_T3)/90
accuracy_of_K5 = (T1_T1 + T2_T2 + T3_T3)/90
accuracy_of_K7 = (T1_T1 + T2_T2 + T3_T3)/90
accuracy_of_K9 = (T1_T1 + T2_T2 + T3_T3)/90

## accuracy_of_K1 = 0.9888888888888889
## accuracy_of_K3 = 0.9555555555555556
## accuracy_of_K5 = 0.9555555555555556
## accuracy_of_K7 = 0.9555555555555556
## accuracy_of_K9 = 0.9555555555555556

## I found accuracy_of_K1 separately

K_accuracy = [1,3,5,7,9]
Accuracy_rate =
[accuracy_of_K1,accuracy_of_K3,accuracy_of_K5,accuracy_of_K7,accuracy_of_K9
]

plt.plot(K_accuracy, Accuracy_rate)
plt.xlabel('k values')
plt.ylabel('accuracy')
plt.title('accuracy with changing k vals on training set')
plt.show()
```