Emin Sadikhov **S017026**

Introduction to Machine Learning and Artificial Neural Networks

Nearest-Mean and Nearest-Neighbor Classification

7 November 2021

Part 1

a)

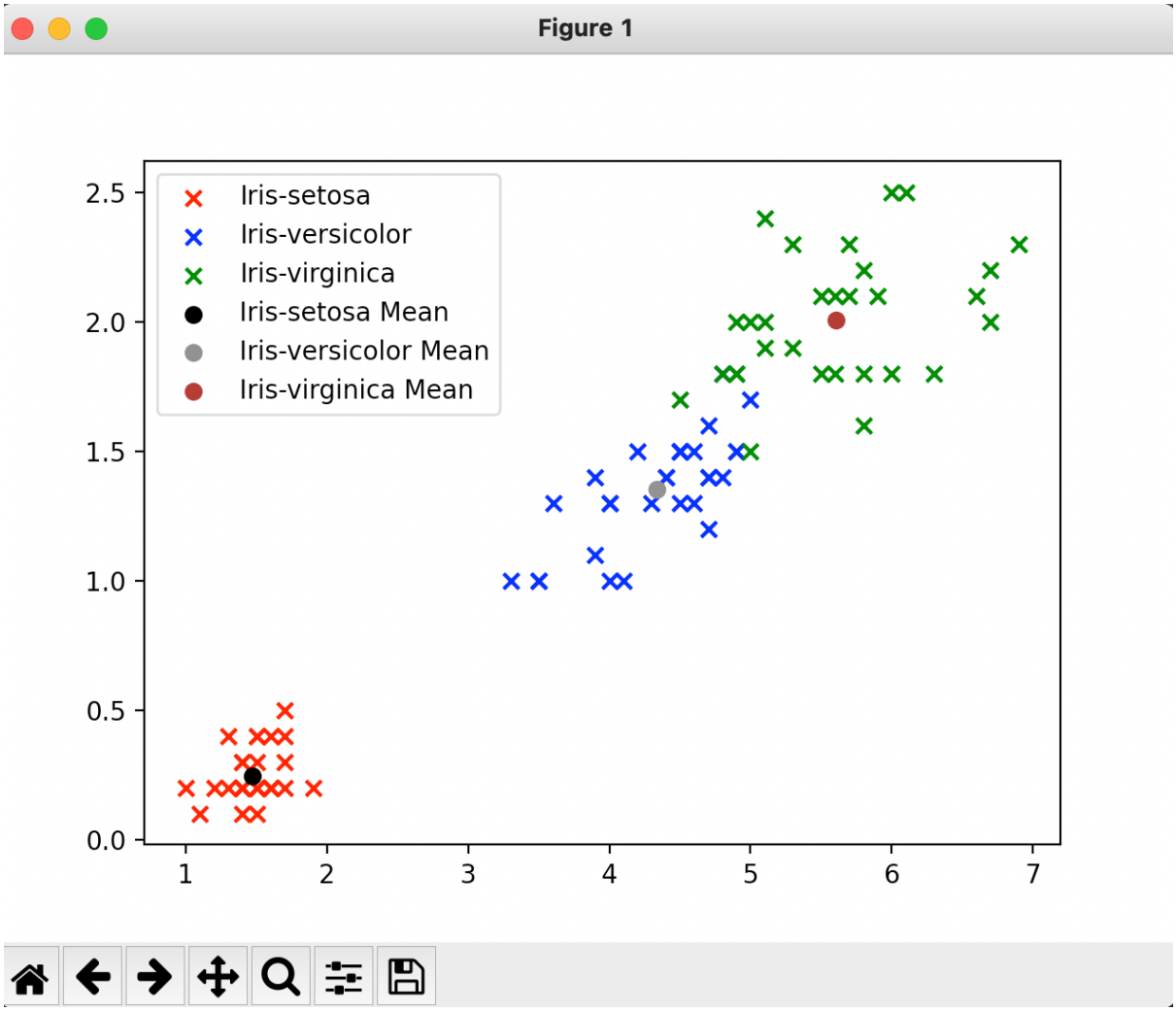
Confusion Matrix for the Part 1 for both Training and Testing data set.

Training	C1	C2	C3
α1 choose C1	30	0	0
α2 choose C2	0	29	5
α3 choose C3	0	1	25

Testing	C1	C2	C3
α1 choose C1	20	0	0
α2 choose C2	0	19	1
α3 choose C3	0	1	19

b)

Plot that represents each species together with their means.



From the above graph we can see that Iris-setosa are located far from the Iris-versicolor and Iris-virginica. Also, some of the instances of Iris-versicolor and Iris-verginica are very close to each other. We can see the result of this in our confusion matrixes. We have 20 for the Iris-setosa and 0, 0 which means 100% accuracy while in the rest two 1 instances have been misclassified.

Part 2

a)

Confusion matrix for testing data set for k=1, k=3, k=5 is given below.

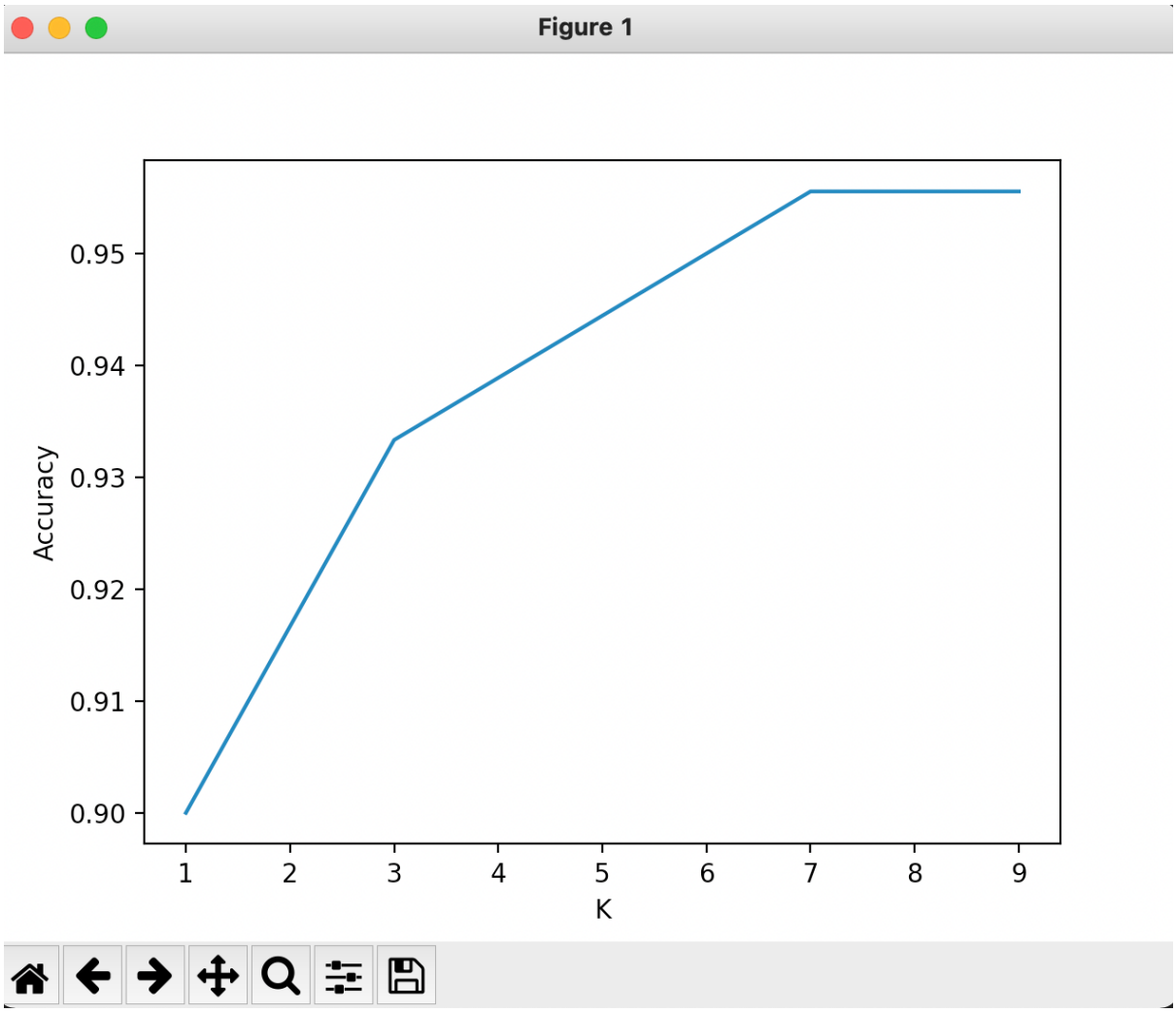
Testing k=1	C1	C2	C3
α1 choose C1	20	0	0
α2 choose C2	0	19	1
α3 choose C3	0	1	19

Testing k=3	C1	C2	C3
α1 choose C1	20	0	0
α2 choose C2	0	20	1
α3 choose C3	0	0	19

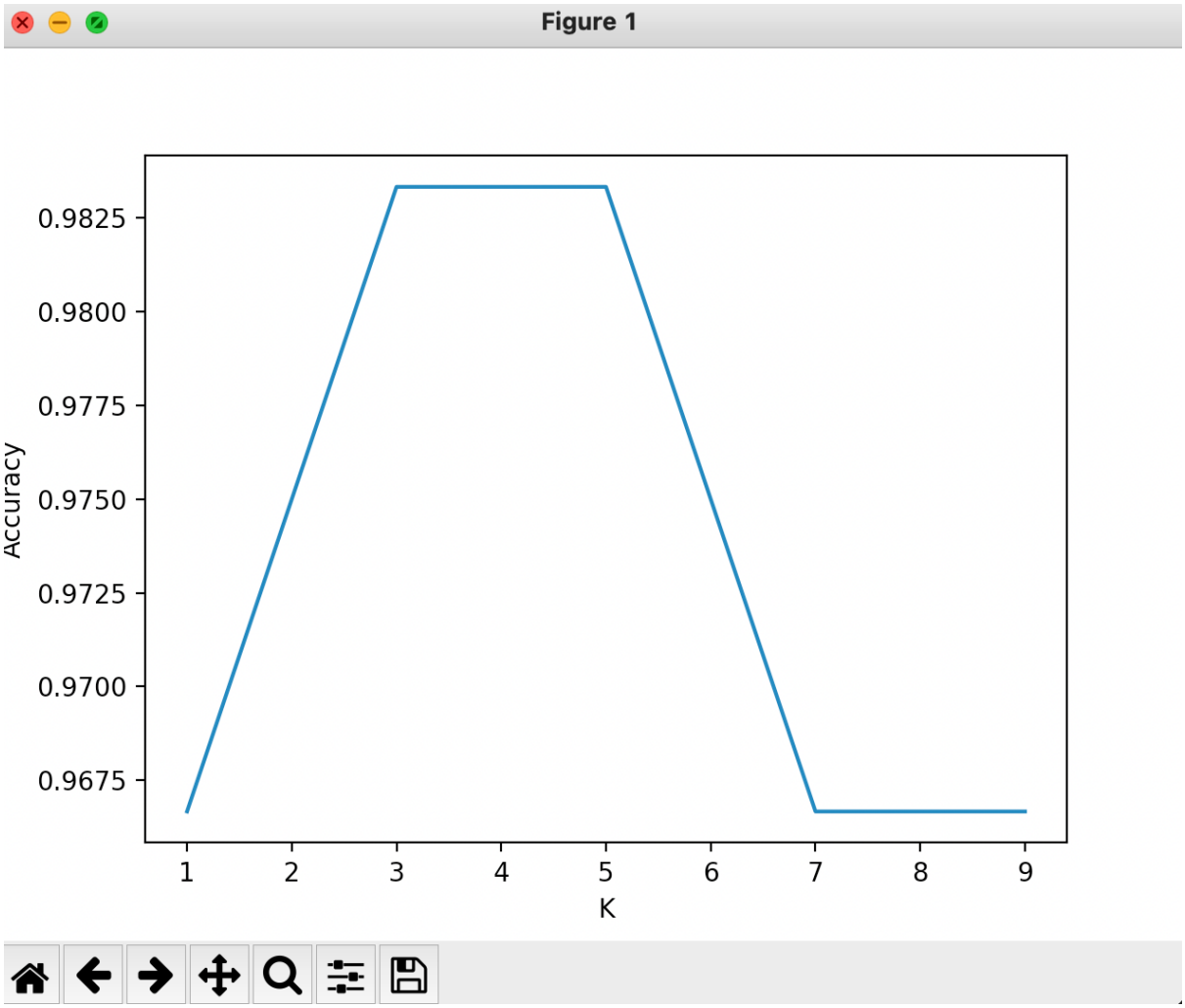
Testing k=5	C1	C2	C3
α1 choose C1	20	0	0
α2 choose C2	0	20	1
α3 choose C3	0	0	19

b)

Training Plot with the accuracies.



Testing Plot with the accuracies.



In this part of the homework, k-Nearest Neighbor classifier was used. The Euclidean distance of each test instance was calculated with respect to all instances of training data. Again, from the plot in the first part of the assignment, we can see that some of the instances (virginica and versicolor) were sparser. As the value of the k increased due to those sparse points the accuracy of the model a bit decreased.

The source code can be found at GitHub after the submission deadline:
https://github.com/SadikhovEmin/CS454

```
from csv import reader
import math
import matplotlib.pyplot as plt

''' VARIABLES '''
training_csv = None
testing_csv = None
petal_length_setosa, petal_width_setosa = [], []
petal_length_versicolor, petal_width_versicolor = [], []
petal_length_virginica, petal_width_virginica = [], []

''' MEANS '''
petal_length_setosa_mean, petal_width_setosa_mean = 0, 0
petal_length_versicolor_mean, petal_width_versicolor_mean = 0, 0
petal_length_virginica_mean, petal_width_virginica_mean = 0, 0


def mean(data):
    values = 0
    count = 0
    for row in data:
        values += float(row)
        count += 1

    return values / count


def part2(data):
    list_euclidean = []
    accuracies = []

    for i in data:
        temp_results = []

        if data == training_csv:
        # if len(data) > 60:
            for j in training_csv:
                if i == j:
                    temp_results.append([math.inf, j[2]])
                else:
                    temp_results.append([math.sqrt(
                        math.pow((float(i[0]) - float(j[0])), 2) + math.pow((float(i
[1]) - float(j[1])),
                                                                           2)), j[2
]])
            list_euclidean.append(sorted(temp_results))  # Sorting all of the euclid
ean

        else:
            for j in training_csv:
                temp_results.append([math.sqrt(
                    math.pow((float(i[0]) - float(j[0])), 2) + math.pow((float(i[1])
 - float(j[1])),
                                                                       2)), j[2]])
            list_euclidean.append(sorted(temp_results))  # Sorting all of the euclid
ean

    for k in range(1, 10, 2):
        confusion_matrix = [
            [0, 0, 0],
            [0, 0, 0],
            [0, 0, 0]
        ]

        for i in list_euclidean:
            setosa, versicolor, virginica = 0, 0, 0

            for j in range(k):
                if i[j][1] == "Iris-setosa":
                    setosa += 1
                elif i[j][1] == "Iris-versicolor":
                    versicolor += 1
                elif i[j][1] == "Iris-virginica":
```

```
                    virginica += 1

            if virginica > setosa and virginica > versicolor:
                if data[list_euclidean.index(i)][2] == "Iris-setosa":
                    confusion_matrix[2][0] += 1
                if data[list_euclidean.index(i)][2] == "Iris-versicolor":
                    confusion_matrix[2][1] += 1
                if data[list_euclidean.index(i)][2] == "Iris-virginica":
                    confusion_matrix[2][2] += 1
            elif setosa > versicolor and setosa > virginica:
                if data[list_euclidean.index(i)][2] == "Iris-setosa":
                    confusion_matrix[0][0] += 1
                if data[list_euclidean.index(i)][2] == "Iris-versicolor":
                    confusion_matrix[0][1] += 1
                if data[list_euclidean.index(i)][2] == "Iris-virginica":
                    confusion_matrix[0][2] += 1
            elif versicolor > setosa and versicolor > virginica:
                if data[list_euclidean.index(i)][2] == "Iris-setosa":
                    confusion_matrix[1][0] += 1
                if data[list_euclidean.index(i)][2] == "Iris-versicolor":
                    confusion_matrix[1][1] += 1
                if data[list_euclidean.index(i)][2] == "Iris-virginica":
                    confusion_matrix[1][2] += 1

        count = 0

        for i in range(len(confusion_matrix)):
            for j in range(len(confusion_matrix[i])):
                if i == j:
                    count += confusion_matrix[i][j]
        accuracies.append((count / len(data)))

        print('Accuracy : ', (count / len(data)))

        print("Confusion matrix k = ", k)
        print(*confusion_matrix, sep='\n')
    plt.plot([1, 3, 5, 7, 9], accuracies)
    plt.xlabel('K')
    plt.ylabel('Accuracy')
    plt.show()


def calculate_confusion_matrix(csv):
    """ Confusion matrix """
    confusion_matrix = [
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]
    ]

    for i in csv:
        euclidean_setosa = math.sqrt(
            math.pow((float(i[0]) - petal_length_setosa_mean), 2) + math.pow((float(
i[1]) - petal_width_setosa_mean),
                                                                             2))
        euclidean_versicolor = math.sqrt(math.pow((float(i[0]) - petal_length_versic
olor_mean), 2) + math.pow(
            (float(i[1]) - petal_width_versicolor_mean), 2))
        euclidean_virginca = math.sqrt(math.pow((float(i[0]) - petal_length_virginic
a_mean), 2) + math.pow(
            (float(i[1]) - petal_width_virginica_mean), 2))

        prediction = min(euclidean_setosa, euclidean_versicolor, euclidean_virginca)
  # Gets the closest point

        if prediction == euclidean_setosa:
            if i[2] == "Iris-setosa":
                confusion_matrix[0][0] += 1
            if i[2] == "Iris-versicolor":
                confusion_matrix[0][1] += 1
            if i[2] == "Iris-virginica":
                confusion_matrix[0][2] += 1
        elif prediction == euclidean_versicolor:
```

```
            if i[2] == "Iris-setosa":
                confusion_matrix[1][0] += 1
            if i[2] == "Iris-versicolor":
                confusion_matrix[1][1] += 1
            if i[2] == "Iris-virginica":
                confusion_matrix[1][2] += 1
        elif prediction == euclidean_virginca:
            if i[2] == "Iris-setosa":
                confusion_matrix[2][0] += 1
            if i[2] == "Iris-versicolor":
                confusion_matrix[2][1] += 1
            if i[2] == "Iris-virginica":
                confusion_matrix[2][2] += 1

    print(*confusion_matrix, sep='\n')


with open('training.csv') as training_file:
    csv_reader = reader(training_file)
    training_csv = list(csv_reader)
    training_csv.pop(0)

    for i in training_csv:
        if i[2] == "Iris-setosa":
            petal_length_setosa.append(float(i[0]))
            petal_width_setosa.append(float(i[1]))
        elif i[2] == "Iris-versicolor":
            petal_length_versicolor.append(float(i[0]))
            petal_width_versicolor.append(float(i[1]))
        elif i[2] == "Iris-virginica":
            petal_length_virginica.append(float(i[0]))
            petal_width_virginica.append(float(i[1]))

    ''' SETOSA '''
    petal_length_setosa_mean = mean(petal_length_setosa)
    petal_width_setosa_mean = mean(petal_width_setosa)

    ''' VERSICOLOR '''
    petal_length_versicolor_mean = mean(petal_length_versicolor)
    petal_width_versicolor_mean = mean(petal_width_versicolor)

    ''' VIRGINICA '''
    petal_length_virginica_mean = mean(petal_length_virginica)
    petal_width_virginica_mean = mean(petal_width_virginica)

    plt.scatter(list(petal_length_setosa), list(petal_width_setosa), color='red', ma
rker='x')
    plt.scatter(list(petal_length_versicolor), list(petal_width_versicolor), color='
blue', marker='x')
    plt.scatter(list(petal_length_virginica), list(petal_width_virginica), color='gr
een', marker='x')
    plt.scatter(petal_length_setosa_mean, petal_width_setosa_mean, color='black')
    plt.scatter(petal_length_versicolor_mean, petal_width_versicolor_mean, color='gr
ay')
    plt.scatter(petal_length_virginica_mean, petal_width_virginica_mean, color='brow
n')
    plt.legend(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Iris-setosa Mea
n', 'Iris-versicolor Mean',
                'Iris-virginica Mean'])

    plt.show()

with open('testing.csv') as testing_file:
    csv_reader = reader(testing_file)
    testing_csv = list(csv_reader)
    testing_csv.pop(0)

    print('Training Confusion Matrix')
    calculate_confusion_matrix(training_csv)
    print('Testing Confusion Matrix')
    calculate_confusion_matrix(testing_csv)
    print('  \n')
```

```
    ''' Part 2 '''
    print('PART 2')

    print('Training part 2')
    part2(training_csv)
    print('Testing part 2')
    part2(testing_csv)
```