KNN - Algorithm

ab Assignment 3 :- Write a program to implement K-Nearest Neighbor algorithm to classify the Iris Data set. Print both correct and wrong predictions. Java/Python ML Library classes can be used for this problem.

Algorithm Steps

Step 1 :- Load/Read/Scan the training Data set, and Testing data set

Step 2 :- Next, Choose/Input the value of K i.e. the nearest data points. K can be any integer

Step 3 :- For each point in the test data do the following

3.1 → Calculate distance b/w test data & each row of training data with the help of ~~any~~ Euclidian Distance ~~criteria~~ method

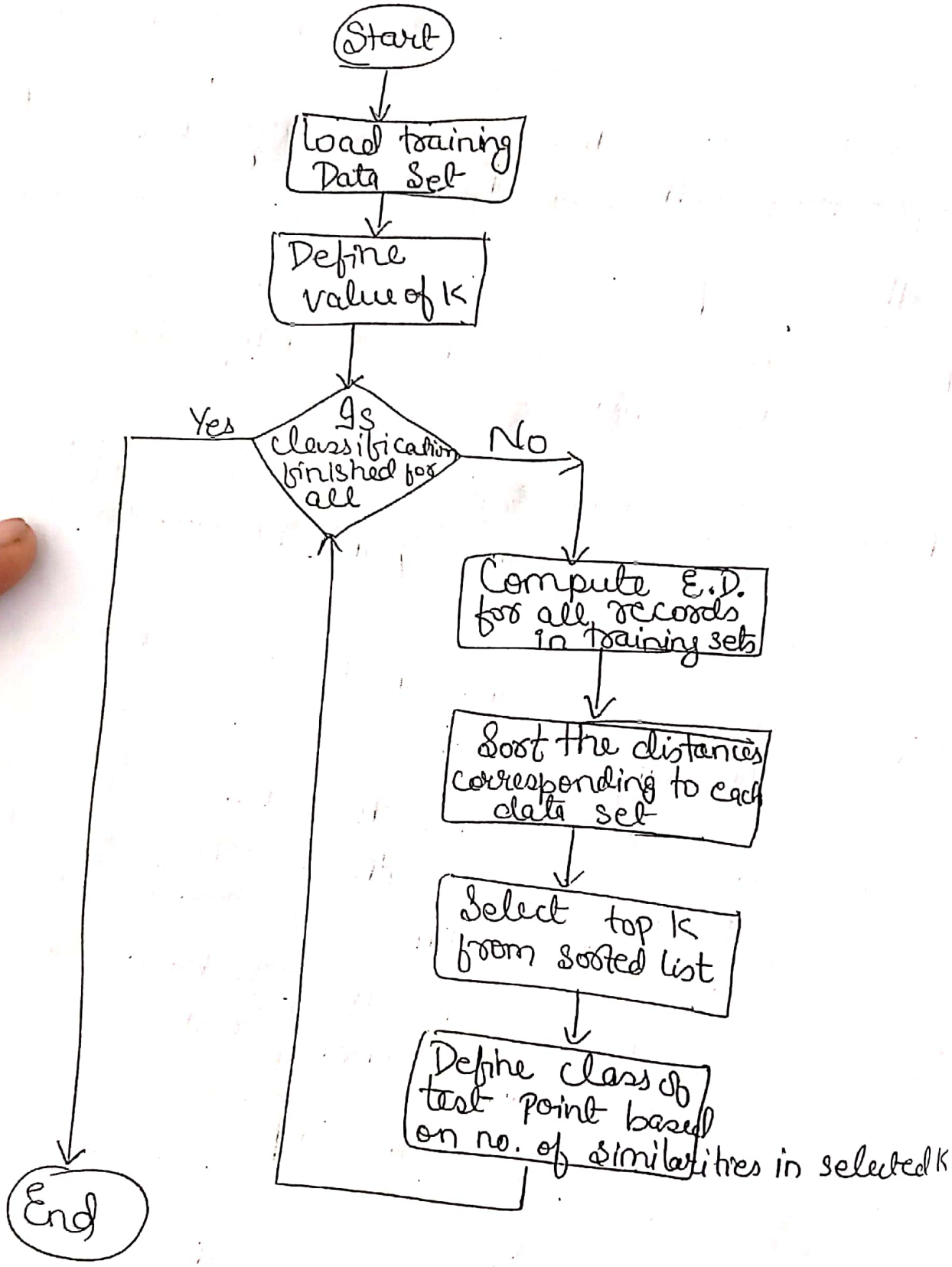
3.2 → Now, based on the distance value, sort them in ascending order.

3.3 → Next, choose the top K rows from the sorted array.

3.4 → Now, Assign a class to the test point based on most frequent class of these rows.

Step 4 :- End

Flow Chart



K-NN Program

blem:-

X	Y	Class
2.78	2.55	0
1.46	2.36	0
3.39	4.04	0
1.38	1.85	0
3.06	3	0
→ 7.62	2.75	1
5.33	2.08	1
6.92	1.77	1
8.67	-0.24	1
7.67	3.5	1

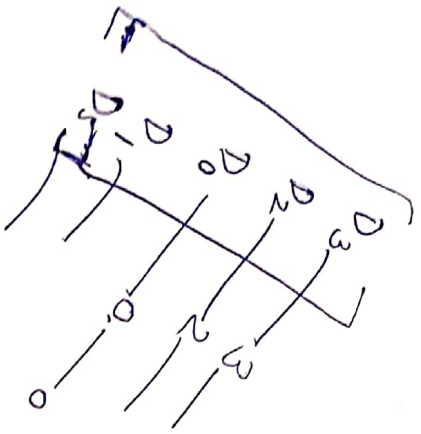
① KNN Theory
② KNN Algo steps
③ KNN flowchart
④ Problem solving
⑤ Code Explanation along with Python basics
⑥

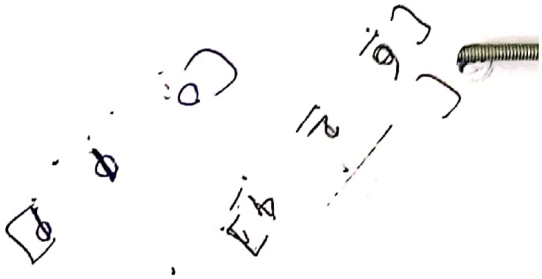
ode→ Using Library:-

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, Y_train)
pred = knn.predict(X_test)
pred.
```

Code

```
import numpy as np
import pandas as pd
from math import sqrt
dt = pd.read_csv("KNN1.csv")
dataset = np.array(dt)
dt.head()
X = dataset  -> training
Y = dataset[ ]  -> testing
#print("X=", X)
#print("Y=", Y)
k = 3
ds = [ ]
for i in range(len(X)-1):
    d = np.subtract(X[i], Y)
    t_s = np.square(d)
    rr = np.sum(t_s)
    ds = np.append(ds, [rr])
#print(ds)
asd = np.argsort(ds)
print(asd)
sd = [ ]
for i in range(k):
    sd.append([X[asd[i]], -1])
#print(sd)
wx = sd.count(2.0)
wy = sd.count(1.0)
wz = sd.count(0.0)
```





$$x = (sum(X1 - X2)^{**}2)^{**}0.5$$

```
if (wx > wy and wx > wz):
    print('class label is 2.0')
elif (wy > wz):
    print('class label is 1.0')
else:
    print('class label is 0.0')
```

Assignment-3

Objective: Write a program to implement K-Nearest Neighbor Algorithm to classify the Iris Data Set. Print both correct and wrong predictions. Java/python Library Classes can be used for this problem.

Step-1: Importing necessory libraries.

- numpy for numerical operations.
- pandas for reading csv.

In [1]:
```
import numpy as np
import pandas as pd
```

Step-2: Reading csv file.

In [6]:
```
df = pd.read_csv("kNN1.csv")
```

Step-3: Inspection of data and understanding it.

In [7]:
```
df.head()
```

Out[7]:

	S1	S2	S3	S4	Flower type	Flower Name
0	5.1	3.5	1.4	0.2	0	Iris-setosa
1	4.9	3.0	1.4	0.2	0	Iris-setosa
2	4.7	3.2	1.3	0.2	0	Iris-setosa
3	4.6	3.1	1.5	0.2	0	Iris-setosa
4	5.0	3.6	1.4	0.2	0	Iris-setosa

About Code (Just for knowledge):

- This df.head() code prints first 5 lines of your dataframe.

About Data:

- Total 5 columns are there.
- Sepal length, Sepal width, Petal length, Petal width are features.
- Flower type classification will be done for Iris Setosa as 0, Iris versicolor as 1 and Iris-viginica as 2,in our target column.
- Output of the program would be the flower type classification

In [8]:
```
df.shape
```

Out[8]:
```
(30, 6)
```

This shows that in our dataframe total 30 rows and 6 columns are available.

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 6 columns):
```

```
(30, 6)
```

This shows that in our dataframe total 30 rows and 6 columns are available.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   S1            30 non-null     float64
 1   S2            30 non-null     float64
 2   S3            30 non-null     float64
 3   S4            30 non-null     float64
 4   Flower type   30 non-null     int64  
 5   Flower Name   30 non-null     object 
dtypes: float64(4), int64(1), object(1)
memory usage: 1.5+ KB
```

This shows details of your dataframe:

- Data type and non null value of each feature and target.
- memory usage of this dataframe.

Setp-4: Implementing K-Nearest Neighbor Algorithm

- Input the training data and testing data.
- Choose/Input the value of 'k'.
- Compute eucleidean distance between teating data sets and training data sets.
- Sort these distances and select/choose k-distance from the top.
- Assign the flower type to the test data based on most frequent class of these rows.

```
dg = pd.read_csv("C:\\Users\\Bhuvnesh\\Desktop\\kNN3.csv")
dataset=np.array(dg)

X = dataset
Y = dataset[19]
print("X=",X)
print("Y=",Y)
k=3
#ed
ds= []
for i in range (len(X)-1):
    d = np.subtract(X[i],Y)
    t_s = np.square(d)
    rr = np.sum(t_s)
    ds = np.append(ds,[rr])

print('distances=',ds)
asd = np.argsort(ds)
print(asd)
sd=[]
for i in range(k):
    sd.append(X[asd[i],-1])
print(sd)
```

```
wx=sd.count(2.0)
wy=sd.count(1.0)
wz=sd.count(0.0)

#w=max(wx,wy,wz)

if (wx>wy and wx>wz):
    print('class label is 2.0')
elif (wy>wz):
    print('class label is 1.0')
else:
    print('class label is 0.0')
```

```
X= [[5.1 3.5 1.4 0.2 0. ]
 [4.9 3.  1.4 0.2 0. ]
 [4.7 3.2 1.3 0.2 0. ]
 [4.6 3.1 1.5 0.2 0. ]
 [5.  3.6 1.4 0.2 0. ]
 [5.4 3.9 1.7 0.4 0. ]
 [4.6 3.4 1.4 0.3 0. ]
 [5.  3.4 1.5 0.2 0. ]
 [4.4 2.9 1.4 0.2 0. ]
 [4.9 3.1 1.5 0.1 0. ]
 [7.  3.2 4.7 1.4 1. ]
 [6.4 3.2 4.5 1.5 1. ]
 [6.9 3.1 4.9 1.5 1. ]
 [5.5 2.3 4.  1.3 1. ]
 [6.5 2.8 4.6 1.5 1. ]
 [5.7 2.8 4.5 1.3 1. ]
 [6.3 3.3 4.7 1.6 1. ]
 [4.9 2.4 3.3 1.  1. ]
 [6.6 2.9 4.6 1.3 1. ]
 [5.2 2.7 3.9 1.4 1. ]
 [6.3 3.3 6.  2.5 2. ]
 [5.8 2.7 5.1 1.9 2. ]
 [7.1 3.  5.9 2.1 2. ]
 [6.3 2.9 5.6 1.8 2. ]
 [6.5 3.  5.8 2.2 2. ]
 [7.6 3.  6.6 2.1 2. ]
 [4.9 2.5 4.5 1.7 2. ]
 [7.3 2.9 6.3 1.8 2. ]
 [6.7 2.5 5.8 1.8 2. ]
 [7.2 3.6 6.1 2.5 2. ]]
Y= [5.2 2.7 3.9 1.4 1. ]
distances= [ 9.34  8.87  9.7   8.72  9.54  8.32  9.31  8.73  9.37  8.7   4.13  2.06
  4.06  0.27  2.2   0.63  2.25  0.7   2.5   0.    8.19  3.05  9.19  5.3
  7.03 14.63  1.58 11.37  7.06]
[19 13 15 17 26 11 14 16 18 21 12 10 23 24 28 20  5  9  3  7  1 22  6  0
  8  4  2 27 25]
[1.0, 1.0, 1.0]
class label is 1.0
```

[ ]: