# Implementing K-Nearest Neighbors (KNN)

Tahira Salwa Rabbi Nishat
*Computer Science and Engineering*
*Ahsanullah University of Science and Technology*
Dhaka, Bangladesh
Dhaka, Bangladesh

***Abstract*—We were given some data that includes train data and test data. We measured the distance of the test data from the train data. Then, We took input from the user on how many nearest neighbour they want. Then we sorted them by their distances and took top k data with the lowest distances. As k was an odd number, we could easily predicted the class of the test data from the results of the majority class of the nearest neighbour's.**

***Index Terms*—KNN, knn, K-nearest neighbours, knn with Euclidean, supervised knn.**

## I. Introduction

K-Nearest Neighbour is a Supervised Machine Learning technique. It is one of the simplest machine learning algorithm too.The KNN algorithm assumes that similar things exist near from each other. In other way we can predict that, data points that are close to each other may belongs to the same classes. So, when a new point comes, its distances from all the other points are calculated using:

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

Here, $(x_1, y_1)$ is a point of the train data and $(x_2, y_2)$ is a point from the test data. Then the distance is sorted and ranked based on the lowest distances.

K is the number of the nearest neighbours. It is taken from the user. The value of k should be an Odd number. For any Point, the k number of neighbours whose distances are the lowest from that point is taken. Then, the class in which the majority of the neighbours belong, is predicted to be the class of that taken point. As k is an odd number, there is no chance of a tie to occur.

## II. Experimental Design / Methodology

We were given two data sets, one as a training set and the other was for testing purpose. The main task was to plot the training points according to theirs classes, take the testing data points and to plot them using the KNN algorithm.

The steps I have followed for this experience are:

1) First, I have loaded the train file and test file using read_csv() and converted them into Dataframe using Panda library.
2) Then, I calculated the row number of the testing dataset and extended the column number of the training dataset to that number. This is as an empty place where I am going to keep my calculated distances.
3) I created a method euclidDistance(a,b,c,d) that takes the x and y coordinates of two points and returns their distance.
4) Next, I calculated distance of every point on the testing dataset from each point of the training dataset using the euclidDistance(a,b,c,d) method, and put them on the extended column of the training dataframe.
5) I took input from user on how many nearest neighbours they want and kept in a variable named K.
6) I opened a text file named "prediction.txt".
7) I sorted the distances for each point on the training set and wrote the top K neighbours with the lowest distance the text file. I did this for every point on the test dataset.
8) As K was an odd number, we could easily determine the class of the test data points from the majority of their k neighbour's classes. And, I wrote them on the text file.
9) Finally, I plotted the test and train data using matplotlib.pyplot.

## III. Result Analysis

The KNN algorithm assumes that the data points that stays closer to each other will belong to the same class. So, after we plotted the train data, we can see that data from each class are so nearer to each other that they kind of formed a cluster and we can easily distinguish them by looking.

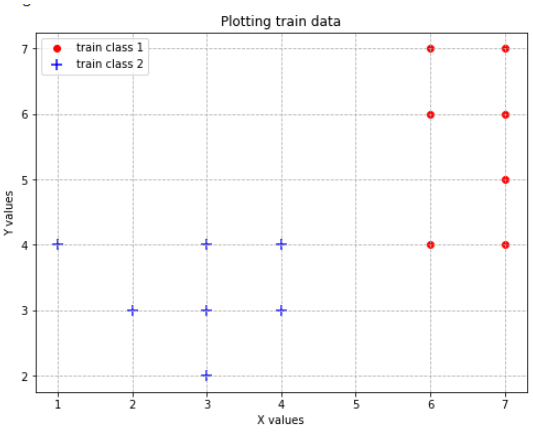


Fig. 1. Plotting train data

Here, we assumed the number of the neighbours, K=3. After we calculated the distances of the test data points, we assigned

them into our predicted classes which is the least distance from the neighbours.The result it printed in a text file named prediction.txt . For K=3 it contains,

```
Test Point : 3,7
Distance 0 : 9     Class: 1
Distance 1 : 9     Class: 2
Distance 2 : 10     Class: 1
Predicted class: 1

Test Point : 7,7
Distance 0 : 0     Class: 1
Distance 1 : 1     Class: 1
Distance 2 : 1     Class: 1
Predicted class: 1

Test Point : 4,3
Distance 0 : 0     Class: 2
Distance 1 : 1     Class: 2
Distance 2 : 1     Class: 2
Predicted class: 2

Test Point : 2,8
Distance 0 : 17     Class: 1
Distance 1 : 17     Class: 2
Distance 2 : 17     Class: 2
Predicted class: 2

Test Point : 3,5
Distance 0 : 1     Class: 2
Distance 1 : 2     Class: 2
Distance 2 : 4     Class: 2
Predicted class: 2
```

Fig. 2. Result of the test data(1)

```
Test Point : 1,2
Distance 0 : 2     Class: 2
Distance 1 : 4     Class: 2
Distance 2 : 4     Class: 2
Predicted class: 2

Test Point : 4,8
Distance 0 : 5     Class: 1
Distance 1 : 8     Class: 1
Distance 2 : 10     Class: 1
Predicted class: 1

Test Point : 8,3
Distance 0 : 2     Class: 1
Distance 1 : 5     Class: 1
Distance 2 : 5     Class: 1
Predicted class: 1

Test Point : 8,4
Distance 0 : 1     Class: 1
Distance 1 : 2     Class: 1
Distance 2 : 4     Class: 1
Predicted class: 1
```

Fig. 3. Result of the test data(2)

After plotting those points from the test data into predicted classes the figure looks like this:

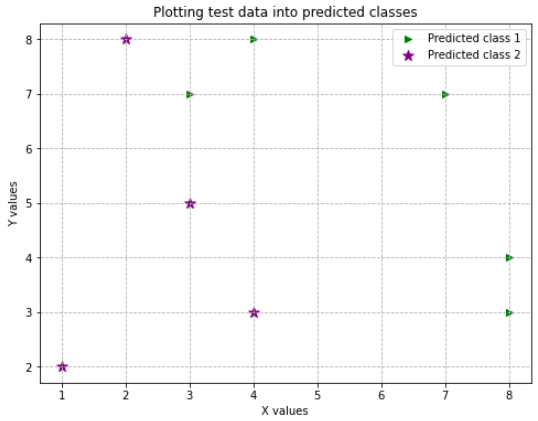


Fig. 4. Plotting test data into predicted classes

## IV. Conclusion

KNN was easy to implement but powerful enough to show how we can categorize unknown points just from their distances from their neighbour's. In real life circumstances, the K-nearest neighbor classification performance can often be significantly improved through feature extraction, in order to perform the desired task using the reduced representation instead of the full size input.

## V. Algorithm Implementation / Code

```
# -*- coding: utf-8 -*-
"""160204070_Assignment4.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1
    It5l4wFxkRStMIIN263aH6X1EzcEB_vk
"""

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

from google.colab import files
uploaded = files.upload()

df_train = pd.read_csv('train_knn.txt', sep=",",
    header = None,dtype='Int64')
df_train = pd.DataFrame(df_train.values, columns = [
    'X', 'Y', 'Class'])

df_test= pd.read_csv('test_knn.txt', sep="," ,
    header = None,dtype='Int64')
df_test = pd.DataFrame(df_test.values, columns = ['X
    ', 'Y'])

for j in range(0, df_test.shape[0]):
    df_train[j]=0

def euclidDistance(a,b,c,d):
    return (np.power(a-b,2)+np.power(c-d,2))

for i in range(0,df_test.shape[0]):
    for j in range(0, df_train.shape[0]):
        a = df_test.iloc[i,0];
        b = df_train.iloc[j,0];
        c = df_test.iloc[i,1];
```

```
        d = df_train.iloc[j,1];

        df_train.iloc[j,i+3] = euclidDistance(a,b,c,
    d)
while(True):
    k = int(input('Enter The value of K :'))
    if (k > df_test.shape[0]-1):
        print("K must be less than " + str(df_test.
    shape[0]-1))
    elif (k%2 == 0):
      print("k must be an odd number.")

    else : break;

f = open("prediction.txt", "w")

for i in range(0, df_test.shape[0]):

    f.write('Test Point : '+str(df_test.iloc[i,0])+'
    ,'+str(df_test.iloc[i,1])+'\n')
    df_sorted = df_train.sort_values(by=[i])
    df_knn = df_sorted.iloc[0:k,[0,1,2,i+3]]

    #print(df_knn)

    for j in range(0,df_knn.shape[0]):
        f.write('Distance ' +str(j)+' : '+str(df_knn
    .iloc[j,3])+'     Class: '+str(df_knn.iloc[j,2])
    +'\n')


    clss = df_knn['Class'].value_counts(sort=True).
    index[0]
    df_test.loc[i,'Class'] = clss;
    f.write('Predicted class: '+str(clss)+'\n'+'\n')

f.close();

df1 = df_train[df_train['Class'] == 1];
df1 = df1.iloc[:,0:2]
w1 = df1.values

df2 = df_train[df_train['Class'] == 2];
df2 = df2.iloc[:,0:2]
w2 = df2.values

df1 = df_test[df_test['Class'] == 1];
df1 = df1.iloc[:,0:2]
w3 = df1.values

df2 = df_test[df_test['Class'] == 2];
df2 = df2.iloc[:,0:2]
w4 = df2.values

plt.figure(0);
fig, ax = plt.subplots(1, figsize = (8, 6))
plt.grid(linestyle="--")

plt.scatter(w1[:,0],w1[:,1],color = 'red', marker =
    'o',label="train class 1")
plt.scatter(w2[:,0],w2[:,1],color = 'blue', marker =
     '+', s=100, label="train class 2")
plt.title('Plotting train data')
plt.xlabel("X values")
plt.ylabel("Y values")
plt.legend(loc="best")
plt.show()

plt.figure(0);
fig, ax = plt.subplots(1, figsize = (8, 6))
plt.grid(linestyle="--")

plt.scatter(w3[:,0],w3[:,1],color = 'green', marker
    = '>',label="Predicted class 1")
plt.scatter(w4[:,0],w4[:,1],color = 'purple', marker
     = '*', s=100, label="Predicted class 2")
plt.title('Plotting test data into predicted classes
    ')
plt.xlabel("X values")
plt.ylabel("Y values")
plt.legend(loc="best")
plt.show()
```