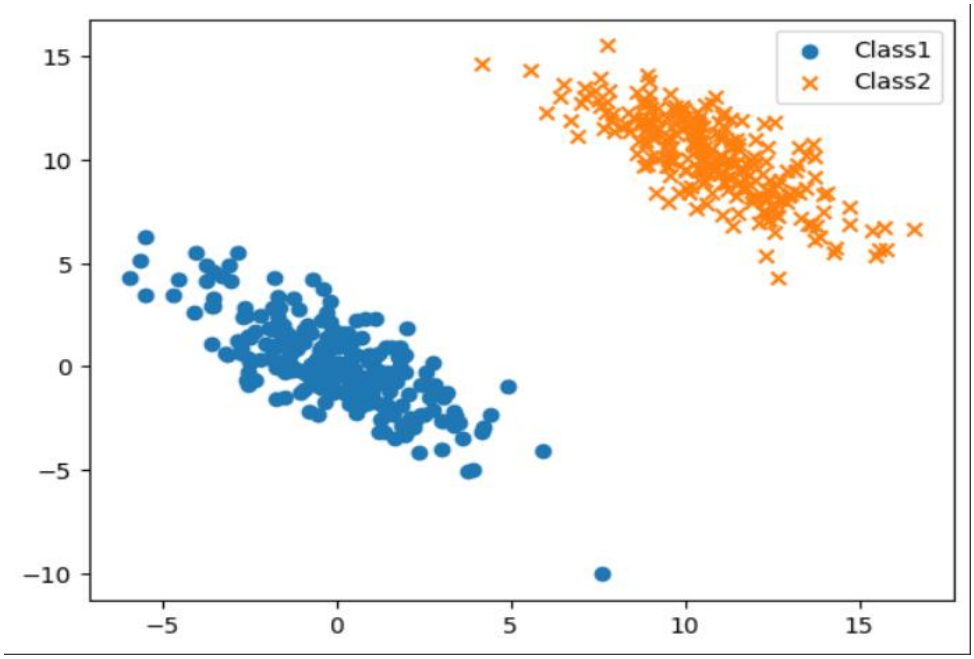
```
import math
import numpy as np
import matplotlib . pyplot as plt
import operator
import csv
import random
from pandas import *
loaddata_train1 = np.loadtxt ('class1_train.txt')
loaddata_train2 = np.loadtxt ( 'class2_train.txt')
loaddata_test1 = np.loadtxt ( 'class1_test.txt')
loaddata_test2 = np.loadtxt ( 'class2_test.txt')
loaddata_train = np.concatenate (( loaddata_train1 , loaddata_train2 )
, axis =0)
loaddata_test = np.concatenate (( loaddata_test1 , loaddata_test2 ) ,
axis =0)
label1 = np . ones (( loaddata_train1 . shape [0] , 1) )
label2 = np . ones (( loaddata_train2 . shape [0] , 1) ) * -1
r1 = np.append(label1 , loaddata_train1 , axis =1)
r2 = np.append( label2 , loaddata_train2 , axis =1)
train_data = np.concatenate ((r1 ,r2))
label3 = np.ones (( loaddata_test1 . shape [0] , 1) )
label4 = np.ones (( loaddata_test2 . shape [0] , 1) ) * -1
r3 = np.append ( label3 , loaddata_test1 , axis =1)
r4 = np.append ( label4 , loaddata_test2 , axis =1)
test_data = np.concatenate (( r3 , r4 ))
plt.scatter ( loaddata_train1 [: ,0] , loaddata_train1 [: ,1] , marker
= 'o' , label = 'Class1')
plt.scatter ( loaddata_train2 [: ,0] , loaddata_train2 [: ,1] , marker
= 'x' , label = 'Class2')
plt.legend ()
plt.show ()
def euclideanDistance ( instance1 , instance2 , length ) :
    distance = 0
    for x in range ( length ) :
        distance += pow (( float ( instance1 [ x ]) - float ( instance2
[ x ]) ) , 2)
    return math . sqrt ( distance )
def getKNeighbors ( train_data , testInstance , k ) :
    distances = []
    length = len ( testInstance ) -1
    for x in range ( len ( train_data ) ) :
        dist = euclideanDistance ( testInstance , train_data [ x ] ,
length )
        distances . append (( train_data [ x ] , dist ) )
    distances . sort ( key = operator . itemgetter (1) )
```

```
    neighbors = []
    for x in range(k) :
        neighbors . append ( distances [ x ][0])
    return neighbors
def getResponse ( neighbors ) :
    classVotes = {}
    for x in range ( len ( neighbors ) ) :
        response = neighbors [ x ][0]
        if response in classVotes :
            classVotes [ response ] += 1
        else :
            classVotes [ response ] = 1
    sortedVotes = sorted ( classVotes . items () , key = operator .
itemgetter (1) , reverse = True )
    return sortedVotes [0][0]
def main () :
    l00 = 0
    l01 = 0
    l10 = 0
    l11 = 0
    split = 0.70
    print ( ' Train : ' + repr (len(train_data)))
    print ( 'Test : ' + repr (len(test_data)))
    predictions = []
    k = int(input(" Enter value of k : "))
    list_pred = []
    list_act = []
    for x in range (len(test_data)) :
        neighbors = getKNeighbors (train_data , test_data [x] , k)
        result = getResponse (neighbors)
        predictions.append (result)
        list_pred.append (result)
        list_act.append (test_data [x][0])
    for i in range (0 ,len ( test_data)) :
        x = list_pred [0]
        y = list_act [0]
        if int ( list_pred [i]) == -1 and int ( list_act [i]) == -1:
            l00 += 1
        elif int ( list_pred [i]) == -1 and int ( list_act [i]) == 1:
             l01 += 1
        elif int ( list_pred [i]) == 1 and int ( list_act [i]) == -1:
            l10 += 1
        elif int ( list_pred [i]) == 1 and int ( list_act [i]) == 1:
             l11 += 1
        i +=1
    a = np . array ([[100,101] ,[110,111]])
    print ( 'Confusion Matrix : ')
```

```
    print ( DataFrame (a , columns = [ ' class_0 ' , ' class_1 '] ,
index = [ ' class_0 ' , ' class_1 ' ]) )
    prec_0 = (100/ float ( 100+101))
    prec_1 = (111/float(110+111))
    acc = (100+111) *100/( 100 + 101 + 110 + 111)
    print ( 'Accuracy : ' + repr ( acc ))
    print ( 'Precision : ')
    print ( 'Precision for class 0: '+ repr (prec_0))
    print ( 'Precision for class 1: ' + repr (prec_1))
    print ( 'Average Precision : ' + repr ((prec_0 + prec_1 ) /2) )
    rec_0 = ( 100/ float ( 100+101))
    rec_1 = ( 110/float( 110+111) )
    print ( ' Recall : ')
    print ( ' Recall for class 0: ' + repr ( rec_0 ) )
    print ( 'Recall for class 1: ' + repr ( rec_1 ) )
    print ( 'Average Recall : ' + repr (( rec_0 + rec_1 ) /2) )
    f0 = (2*( prec_0 * rec_0 ) /( prec_0 + rec_0 ) )
    f1 = (2*( prec_1 * rec_1 ) /( prec_1 + rec_1 ) )
    print ( ' F1 Score : ')
    print ( 'F1 Score for class 0: '+ repr ( f0 ) )
    print ( ' F1 Score for class 1: ' + repr ( f1 ) )
    print ( 'Average F1 Score : ' + repr (( f0 + f1 ) /2) )
```

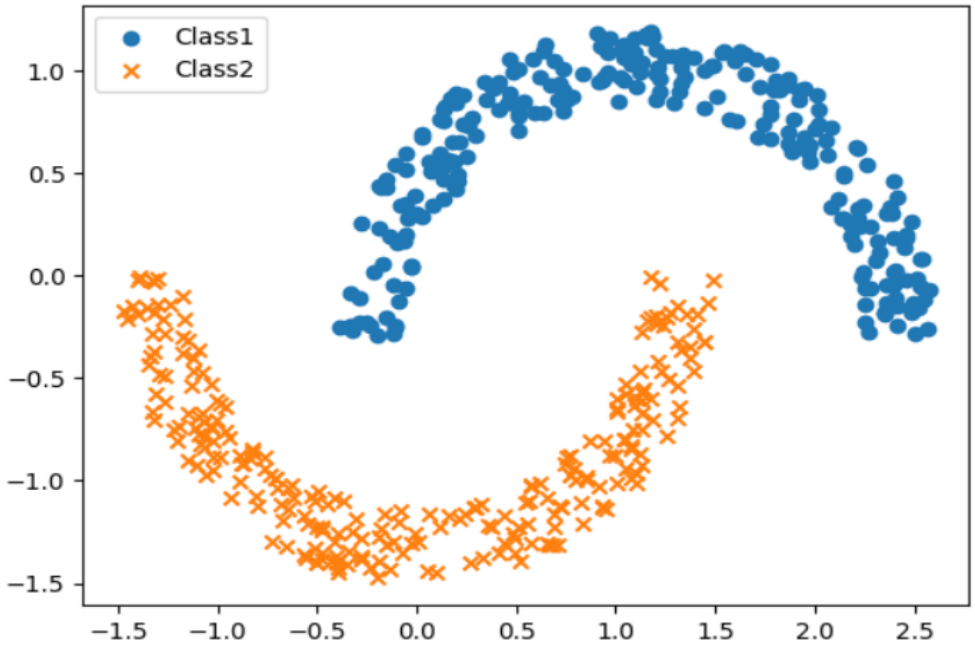
Output :

1 . Linearly Seperable Data :



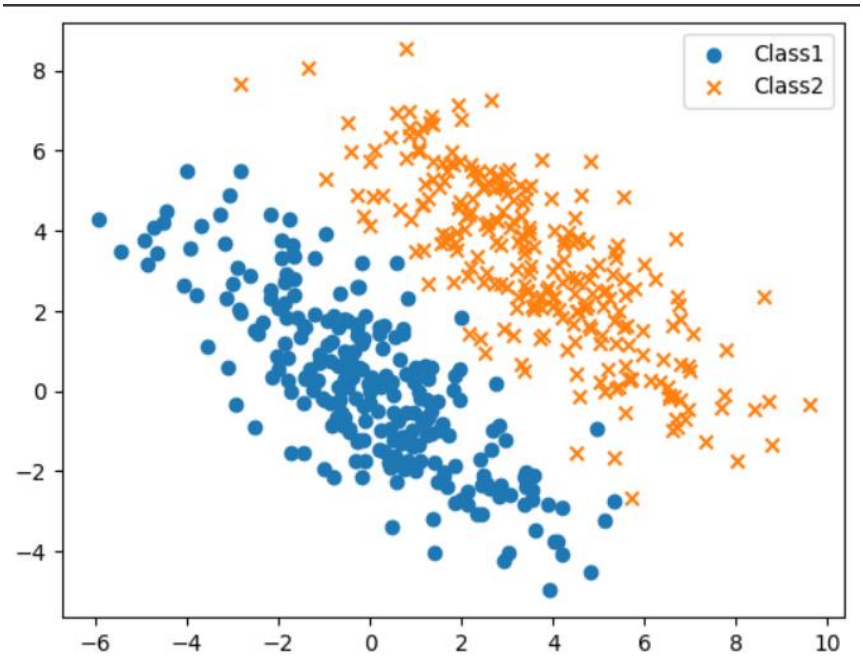

```
 Train : 500
Test : 200
 Enter value of k : 10
Confusion Matrix :
             class_0    class_1
 class_0          100         101
 class_1          110         111
Accuracy : 50.0
Precision :
Precision for class 0: 0.49751243781094527
Precision for class 1: 0.5022624434389140
Average Precision : 0.4998874406249296
 Recall :
 Recall for class 0: 0.49751243781094527
Recall for class 1: 0.497737556561086
Average Recall : 0.4976249971860156
 F1 Score :
F1 Score for class 0: 0.49751243781094527
 F1 Score for class 1: 0.499989762699371 45
Average F1 Score : 0.498751100255158 34
```

2. Non – Linearly Seperable Data




```
 Train : 500
Test : 200
 Enter value of k : 10
Confusion Matrix :
            class_0    class_1
 class_0         100        101
 class_1         110        111
Accuracy : 50.0
Precision :
Precision for class 0: 0.4975124378109453
Precision for class 1: 0.502262443438914
Average Precision : 0.4998874406249296
 Recall :
 Recall for class 0: 0.4975124378109453
Recall for class 1: 0.497737556561086
Average Recall : 0.4976249971860156
 F1 Score :
F1 Score for class 0: 0.4975124378109453
 F1 Score for class 1: 0.49998976269937145
Average F1 Score : 0.49875110025515834
```

3. Overlapping Data




```
 Train : 500
Test : 200
 Enter value of k : 10
Confusion Matrix :
            class_0    class_1
 class_0         100        101
 class_1         110        111
Accuracy : 50.0
Precision :
Precision for class 0: 0.4975124378109453
Precision for class 1: 0.502262443438914
Average Precision : 0.4998874406249296
 Recall :
 Recall for class 0: 0.4975124378109453
Recall for class 1: 0.497737556561086
Average Recall : 0.4976249971860156
 F1 Score :
F1 Score for class 0: 0.4975124378109453
 F1 Score for class 1: 0.49998976269937145
Average F1 Score : 0.49875110025515834
```