Implement and visualize k-NN classifier. Evaluate the algorithm using any dataset of your choice from UCI repository. Output should include accuracy, error rate, sensitivity, specificity, precision, recall.

Link <https://archive.ics.uci.edu/ml/datasets/glass+identification>

Attribute Information:

1. Id number: 1 to 214
2. RI: refractive index
3. Na: Sodium (unit measurement: weight percent in corresponding oxide, as are attributes 4-10)
4. Mg: Magnesium
5. Al: Aluminum
6. Si: Silicon
7. K: Potassium
8. Ca: Calcium
9. Ba: Barium
10. Fe: Iron
11. Type of glass: (class attribute)
    1. building\_windows\_float\_processed
    2. building\_windows\_non\_float\_processed
    3. vehicle\_windows\_float\_processed
    4. vehicle\_windows\_non\_float\_processed (none in this database) 5 containers
12. tableware
13. headlamps

In [99]:

**def** calc(arr,test): arr1**=**[]

**for** i **in** arr: calc**=**0

**for** k **in** range (1,10): calc**+=**pow((i[k]**-**test[k]),2)

arr1**+=**[[calc,i[10]]]

**return** arr1

In [100]

**def** KNNcalc(arr,n):

arr1**=**[9999 **for** i **in** range(0,n)] arr2**=**[9999 **for** i **in** range(0,n)] **for** i **in** arr:

**if** (i[0]**<**max(arr1)):

**for** j **in** range(0,n):

**if**(max(arr1)**==**arr1[j]): arr1[j]**=**i[0]

arr2[j]**=**i[1]

**break**

**return** arr1,arr2

In [101]:

**def** accuraccy(arr,n): count**=**0

**for** i **in** arr:

**if** i**==**n:

count**+=**1

**return** count**\***100**/**len(arr)

In [102]:

**def** split(data): train**=**[] test**=**[]

**for** i **in** range(0,len(data)):

**if**(i**%**15**==**0):

test**+=**[data[i]]

**else**:

train**+=**[data[i]]

**return** train,test

In [103]:

**def** pred(a):

arr**=**[0 **for** i **in** range(0,8)]

**for** i **in** a:

arr[int(i)]**+=**1 flag**=**max(arr)

**for** i **in** range(0,8):

**if**(arr[i]**==**flag): **return**(i)

In [104]:

**def** acc(test,train,n): arr**=**[]

error**=**0

**for** i **in** test: print(i) z**=**calc(train,i)

arr1,arr2**=**KNNcalc(z,n) acc**=**accuraccy(arr2,i[10]) arr**+=**[acc]

print("actual value: "**+**str(i[10])**+**" predicted values: "**+**str(arr2)) print("accuraccy :"**+**str(acc)**+**"\n\n")

**if**(pred(arr2)**!=**i[10]): error**+=**1

print("total accuraccy"**+**str(sum(arr)**/**15))

print("total error"**+**str(error**\***100**/**15))

**return**([sum(arr)**/**15,error**\***100**/**15])

In [105]:

arr**=**[[1,2],[3,5],[8,4],[7,3],[2,5]]

arr1,arr2**=**KNNcalc(arr,4) arr2

Out[105]: [2, 5, 5, 3]

In [106]:

r

**import** pandas **as** pd

**import** numpy **as** np

dataset **=** pd.read\_csv("C:\\Users\\Sid\\Desktop\\python files\\glass prediction KNN from sc

\\data.csv”)

data**=**dataset.values.tolist()

train,test**=**split(data)

In [107]:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| dataset | | | | | | | | | | | |
| **196** | 197 | 1.51556 | 13.87 | 0.00 | 2.54 | 73.23 | 0.14 | 9.41 | 0.81 | 0.01 | 7 |
| **197** | 198 | 1.51727 | 14.70 | 0.00 | 2.34 | 73.28 | 0.00 | 8.95 | 0.66 | 0.00 | 7 |
| **198** | 199 | 1.51531 | 14.38 | 0.00 | 2.66 | 73.10 | 0.04 | 9.08 | 0.64 | 0.00 | 7 |
| **199** | 200 | 1.51609 | 15.01 | 0.00 | 2.51 | 73.05 | 0.05 | 8.83 | 0.53 | 0.00 | 7 |
| **200** | 201 | 1.51508 | 15.15 | 0.00 | 2.25 | 73.50 | 0.00 | 8.34 | 0.63 | 0.00 | 7 |
| **201** | 202 | 1.51653 | 11.95 | 0.00 | 1.19 | 75.18 | 2.70 | 8.93 | 0.00 | 0.00 | 7 |
| **202** | 203 | 1.51514 | 14.85 | 0.00 | 2.42 | 73.72 | 0.00 | 8.39 | 0.56 | 0.00 | 7 |
| **203** | 204 | 1.51658 | 14.80 | 0.00 | 1.99 | 73.11 | 0.00 | 8.28 | 1.71 | 0.00 | 7 |
| **204** | 205 | 1.51617 | 14.95 | 0.00 | 2.27 | 73.30 | 0.00 | 8.71 | 0.67 | 0.00 | 7 |
| **205** | 206 | 1.51732 | 14.95 | 0.00 | 1.80 | 72.99 | 0.00 | 8.61 | 1.55 | 0.00 | 7 |
| **206** | 207 | 1.51645 | 14.94 | 0.00 | 1.87 | 73.11 | 0.00 | 8.67 | 1.38 | 0.00 | 7 |
| **207** | 208 | 1.51831 | 14.39 | 0.00 | 1.82 | 72.86 | 1.41 | 6.47 | 2.88 | 0.00 | 7 |
| **208** | 209 | 1.51640 | 14.37 | 0.00 | 2.74 | 72.85 | 0.00 | 9.45 | 0.54 | 0.00 | 7 |

In [108]:

print(len(train)) print(len(test))

199

15

In [109]:

z**=**calc(data,[1.0, 1.52101, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1.0])

r1,r2**=**KNNcalc(z,7) accuraccy(r2,1)

Out[109]:

57.142857142857146

In [110]:

1.0, 2.0, 3.0, 2.0]

bestKNN**=**[]

**for** i **in** range(2,15): bestKNN**+=**[[i]**+**acc(test,train,i)]

accuraccy :18.181818181818183

[166.0, 1.5217100000000001, 11.56, 1.88, 1.56, 72.86, 0.47, 11.41, 0.0,

0.0, 5.0]

actual value: 5.0 predicted values: [2.0, 1.0, 5.0, 5.0, 2.0, 5.0, 2.0,

5.0, 2.0, 5.0, 5.0]

accuraccy :54.54545454545455

[181.0, 1.51299, 14.4, 1.74, 1.54, 74.55, 0.0, 7.59, 0.0, 0.0, 6.0]

actual value: 6.0 predicted values: [7.0, 1.0, 2.0, 7.0, 2.0, 6.0, 2.0,

7.0, 2.0, 2.0, 7.0]

accuraccy :9.090909090909092

[196.0, 1.51545, 14.14, 0.0, 2.68, 73.39, 0.08, 9.07, 0.61, 0.05, 7.0]

actual value: 7.0 predicted values: [7.0, 7.0, 7.0, 7.0, 7.0, 7.0, 7.0,

In [111]:

acc(test,train,4)

[1.0, 1.52101, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1.0]

actual value: 1.0 predicted values: [1.0, 2.0, 2.0, 1.0]

accuraccy :50.0

[16.0, 1.5176100000000001, 12.81, 3.54, 1.23, 73.24, 0.58, 8.39, 0.0, 0.

0, 1.0]

actual value: 1.0 predicted values: [1.0, 1.0, 1.0, 1.0]

accuraccy :100.0

[31.0, 1.51768, 12.65, 3.56, 1.3, 73.08, 0.61, 8.69, 0.0, 0.14, 1.0]

actual value: 1.0 predicted values: [1.0, 1.0, 1.0, 1.0]

accuraccy :100.0

[46.0, 1.5190000000000001, 13.49, 3.48, 1.35, 71.95, 0.55, 9.0, 0.0, 0.0,

1.0]

actual value: 1.0 predicted values: [3.0, 3.0, 3.0, 1.0]

In [114]:

*#the best result is found using 8 neighbours having 64% acc match and 20% error*

bestKNN

Out[114]:

[[2, 70.0, 20.0],

[3, 66.66666666666667, 33.333333333333336],

[4, 65.0, 26.666666666666668],

[5, 66.66666666666667, 33.333333333333336],

[6, 65.55555555555556, 26.666666666666668],

[7, 63.80952380952381, 26.666666666666668],

[8, 64.16666666666667, 20.0],

[9, 62.96296296296295, 26.666666666666668],

[10, 62.666666666666664, 20.0],

[11, 63.03030303030302, 26.666666666666668],

[12, 62.777777777777786, 20.0],

[13, 61.02564102564102, 20.0],

[14, 60.476190476190474, 26.666666666666668]]

In [119]:

z**=**calc(data,[176.0, 1.52119, 12.97, 0.33, 1.51, 73.39, 0.13, 11.27, 0.0, 0.28, 5.0])

arr1,arr2**=**KNNcalc(z,8)

In [120]:

arr2

Out[120]:

[5.0, 2.0, 6.0, 5.0, 5.0, 2.0, 5.0, 5.0]

In [121]:

pred(arr2)

Out[121]:

5