# K NEAREST NEIGHBORS

Name: Nguyen Trung Tam

Students ID: B1910697

Class: M02

## Given a dataset as follows:

|  |  |  |
| --- | --- | --- |
| X1 | X2 | Class |
| 0.376 | 0.488 | 0 |
| 0.312 | 0.544 | 0 |
| 0.298 | 0.624 | 0 |
| 0.394 | 0.6 | 0 |
| 0.506 | 0.512 | 0 |
| 0.488 | 0.334 | 1 |
| 0.478 | 0.398 | 1 |
| 0.606 | 0.366 | 1 |
| 0.428 | 0.294 | 1 |
| 0.542 | 0.252 | 1 |

Classifying the testset with 1NN, 3NN:

|  |  |  |  |
| --- | --- | --- | --- |
|  | X1 | X2 | Class |
| P1 | 0.55 | 0.364 | ? |
| P2 | 0.558 | 0.47 | ? |
| P3 | 0.456 | 0.45 | ? |
| P4 | 0.45 | 0.57 | ? |

We will use Euclidean distance to classify the test set as below:

Euclidean distance

with is Euclidean distance

First, we will need to calculate the distance between the test points and every points in the training set

### Calculate the distance between training points and P1

|  |  |  |  |
| --- | --- | --- | --- |
| X1 | X2 | Class | Distance |
| 0.376 | 0.488 | 0 | 0.213663287 |
| 0.312 | 0.544 | 0 | 0.298402413 |
| 0.298 | 0.624 | 0 | 0.362082863 |
| 0.394 | 0.6 | 0 | 0.282899275 |
| 0.506 | 0.512 | 0 | 0.154402073 |
| 0.488 | 0.334 | 1 | 0.068876701 |
| 0.478 | 0.398 | 1 | 0.079624117 |
| 0.606 | 0.366 | 1 | 0.056035703 |
| 0.428 | 0.294 | 1 | 0.140655608 |
| 0.542 | 0.252 | 1 | 0.112285351 |

From the results above, we see that the nearest point is

|  |  |  |  |
| --- | --- | --- | --- |
| 0.606 | 0.366 | 1 | 0.056035703 |

So 1NN = 1 for P1

From the results above, we see that the 3 nearest point is

|  |  |  |  |
| --- | --- | --- | --- |
| 0.606 | 0.366 | 1 | 0.056035703 |
| 0.488 | 0.334 | 1 | 0.068876701 |
| 0.478 | 0.398 | 1 | 0.079624117 |

So 3NN = 1 for P1

### Calculate the distance between training points and P1

|  |  |  |  |
| --- | --- | --- | --- |
| X1 | X2 | Class | Distance |
| 0.376 | 0.488 | 0 | 0.182887944 |
| 0.312 | 0.544 | 0 | 0.256889081 |
| 0.298 | 0.624 | 0 | 0.302185374 |
| 0.394 | 0.6 | 0 | 0.209274939 |
| 0.506 | 0.512 | 0 | 0.0668431 |
| 0.488 | 0.334 | 1 | 0.15295751 |
| 0.478 | 0.398 | 1 | 0.107628992 |
| 0.606 | 0.366 | 1 | 0.114542569 |
| 0.428 | 0.294 | 1 | 0.21880585 |
| 0.542 | 0.252 | 1 | 0.218586367 |

From the results above, we see that the nearest point is

|  |  |  |  |
| --- | --- | --- | --- |
| 0.506 | 0.512 | 0 | 0.0668431 |

So 1NN = 0 for P2

From the results above, we see that the 3 nearest point is

|  |  |  |  |
| --- | --- | --- | --- |
| 0.506 | 0.512 | 0 | 0.0668431 |
| 0.478 | 0.398 | 1 | 0.107628992 |
| 0.606 | 0.366 | 1 | 0.114542569 |

So 3NN = 1 for P2

### Calculate the distance between training points and P3

|  |  |  |  |
| --- | --- | --- | --- |
| X1 | X2 | Class | Distance |
| 0.376 | 0.488 | 0 | 0.088566359 |
| 0.312 | 0.544 | 0 | 0.171965113 |
| 0.298 | 0.624 | 0 | 0.235031913 |
| 0.394 | 0.6 | 0 | 0.162308349 |
| 0.506 | 0.512 | 0 | 0.079649231 |
| 0.488 | 0.334 | 1 | 0.120332872 |
| 0.478 | 0.398 | 1 | 0.056462377 |
| 0.606 | 0.366 | 1 | 0.171918585 |
| 0.428 | 0.294 | 1 | 0.158492902 |
| 0.542 | 0.252 | 1 | 0.215870331 |

From the results above, we see that the nearest point is

|  |  |  |  |
| --- | --- | --- | --- |
| 0.478 | 0.398 | 1 | 0.056462377 |

So 1NN = 1 for P3

From the results above, we see that the 3 nearest point is

|  |  |  |  |
| --- | --- | --- | --- |
| 0.478 | 0.398 | 1 | 0.056462377 |
| 0.506 | 0.512 | 0 | 0.079649231 |
| 0.376 | 0.488 | 0 | 0.088566359 |

So 3NN = 0 for P3

### Calculate the distance between training points and P4

|  |  |  |  |
| --- | --- | --- | --- |
| X1 | X2 | Class | Distance |
| 0.376 | 0.488 | 0 | 0.11045361 |
| 0.312 | 0.544 | 0 | 0.140427917 |
| 0.298 | 0.624 | 0 | 0.16130716 |
| 0.394 | 0.6 | 0 | 0.063529521 |
| 0.506 | 0.512 | 0 | 0.080622577 |
| 0.488 | 0.334 | 1 | 0.239039746 |
| 0.478 | 0.398 | 1 | 0.174264167 |
| 0.606 | 0.366 | 1 | 0.256811215 |
| 0.428 | 0.294 | 1 | 0.276875423 |
| 0.542 | 0.252 | 1 | 0.331040783 |

From the results above, we see that the nearest point is

|  |  |  |  |
| --- | --- | --- | --- |
| 0.394 | 0.6 | 0 | 0.063529521 |

So 1NN = 0 for P4

From the results above, we see that the 3 nearest point is

|  |  |  |  |
| --- | --- | --- | --- |
| 0.394 | 0.6 | 0 | 0.063529521 |
| 0.506 | 0.512 | 0 | 0.080622577 |
| 0.376 | 0.488 | 0 | 0.11045361 |

So 3NN = 0 for P4

To summary, we have the result as below

#### 1NN

|  |  |  |  |
| --- | --- | --- | --- |
|  | X1 | X2 | Class |
| P1 | 0.55 | 0.364 | 1 |
| P2 | 0.558 | 0.47 | 0 |
| P3 | 0.456 | 0.45 | 1 |
| P4 | 0.45 | 0.57 | 0 |

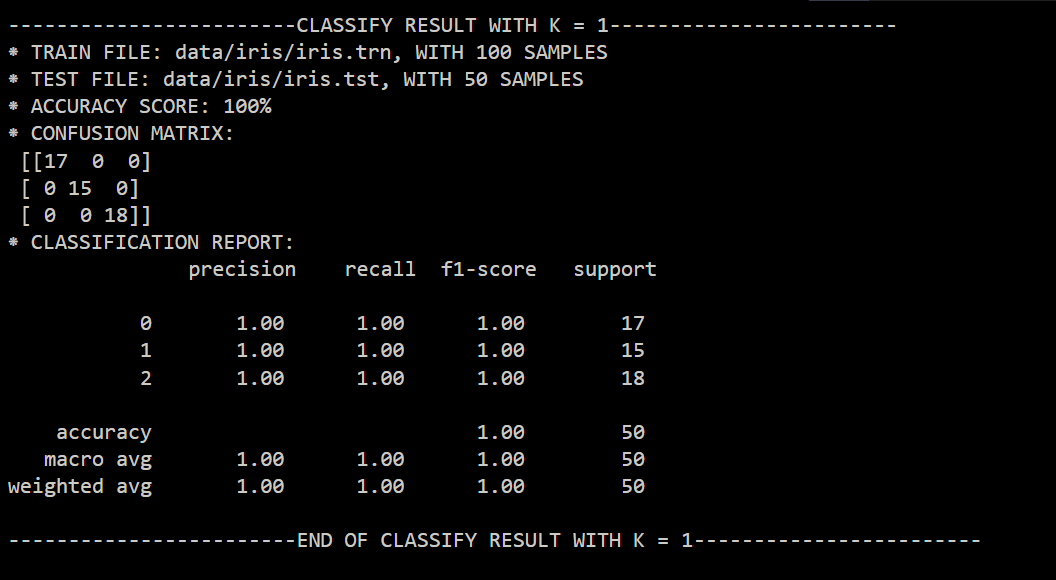
#### 3NN

|  |  |  |  |
| --- | --- | --- | --- |
|  | X1 | X2 | Class |
| P1 | 0.55 | 0.364 | 1 |
| P2 | 0.558 | 0.47 | 1 |
| P3 | 0.456 | 0.45 | 0 |
| P4 | 0.45 | 0.57 | 0 |

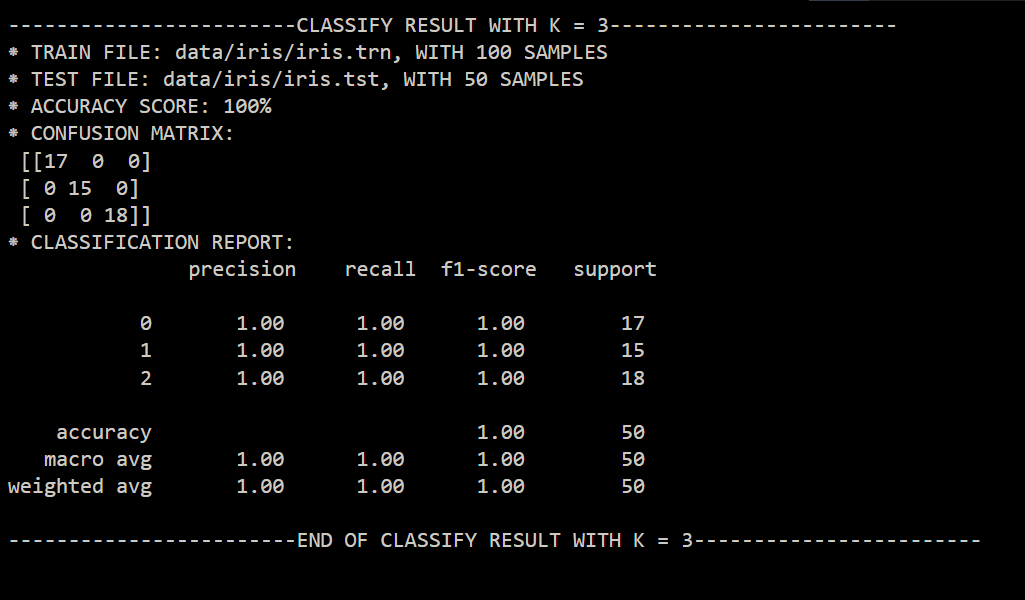
## Implement kNN from scratch in Python

import numpy as np  
import pandas as pd  
from scipy.stats import mode  
from sklearn import metrics  
  
  
def euclidean\_distance(point1, point2):  
 # calculating Euclidean distance  
 # using linalg.norm()  
 dist = np.linalg.norm(point1 - point2)  
 return dist  
  
  
# Locate the most similar neighbors  
def get\_neighbors(train, test\_row, num\_neighbors):  
 distances = list()  
 for train\_row in train:  
 dist = euclidean\_distance(test\_row, train\_row)  
 distances.append(dist)  
 distances = np.array(distances)  
 neighbors = train[np.argpartition(distances, num\_neighbors)[:num\_neighbors]]  
 return neighbors  
  
  
def predict(train, test, k):  
 labels = []  
 for item in test:  
 neighbors = get\_neighbors(train, item, k)  
 pred\_labels = neighbors[:, -1]  
 # noinspection PyUnresolvedReferences  
 labels.append(mode(pred\_labels).mode[0])  
 return labels  
  
  
def report(train\_f, test\_f, k):  
 train\_set = pd.read\_csv(train\_f, sep='[,,\s]', header=None, engine='python')  
 test\_set = pd.read\_csv(test\_f, sep='[,,\s]', header=None, engine='python')  
  
 x\_train = train\_set.values # Get training data points (exclude class value)  
  
 num\_row, num\_col = train\_set.shape  
 test\_num\_row, test\_num\_col = test\_set.shape  
  
 x\_test = test\_set.values # Get training data points (exclude class value)  
 y\_test = test\_set.iloc[:, test\_set.shape[1] - 1].values # Get training class data points (the last column)  
  
 pred = predict(x\_train, x\_test, k)  
  
 print("-" \* 24 + "CLASSIFY RESULT WITH K = %s" % k + "-" \* 24)  
 print("⁕ TRAIN FILE: %s, WITH %d SAMPLES" % (train\_f, num\_row))  
 print("⁕ TEST FILE: %s, WITH %d SAMPLES" % (test\_f, test\_num\_row))  
 print("⁕ ACCURACY SCORE: %d%%" % (metrics.accuracy\_score(y\_test, pred) \* 100))  
 print("⁕ CONFUSION MATRIX:\n", metrics.confusion\_matrix(y\_test, pred))  
 print("⁕ CLASSIFICATION REPORT:\n", metrics.classification\_report(y\_test, pred))  
 print("-" \* 24 + "END OF CLASSIFY RESULT WITH K = %s" % k + "-" \* 24)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 report('data/faces/data.trn', 'data/faces/data.tst', 1)

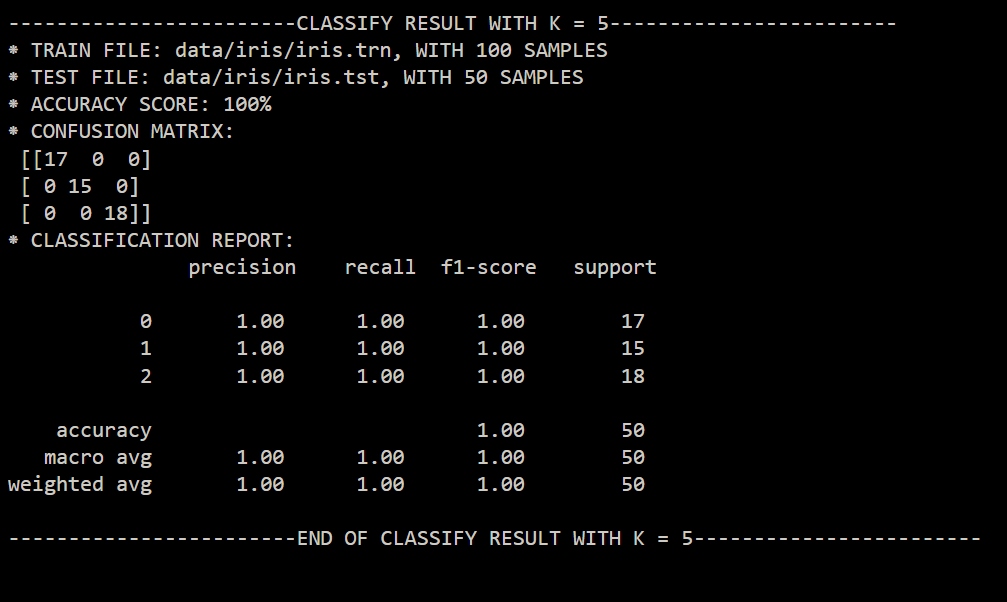
### Iris (K = 1)



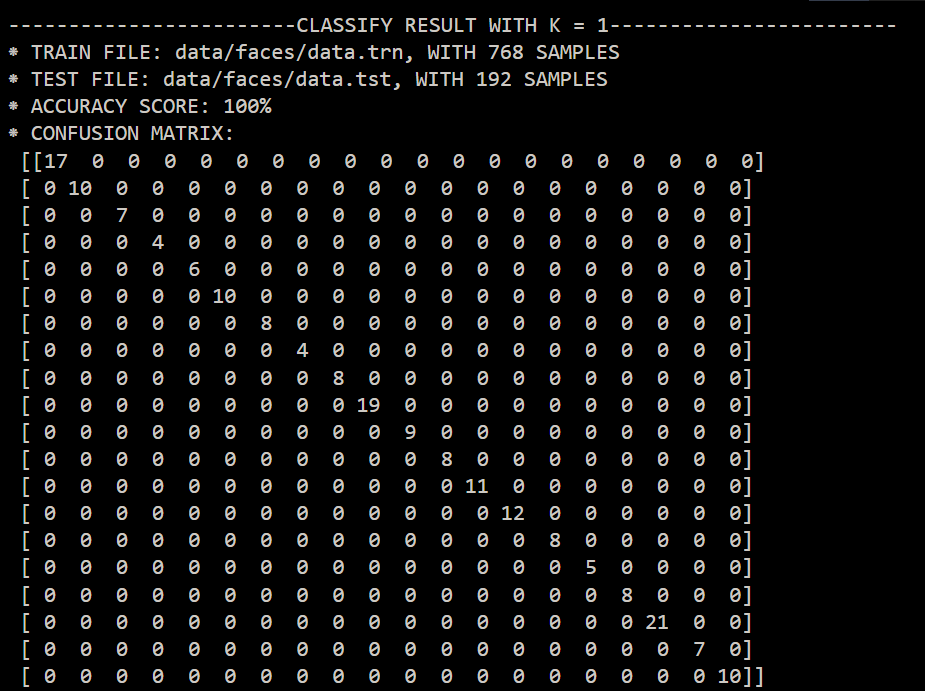
### Iris (K = 3)

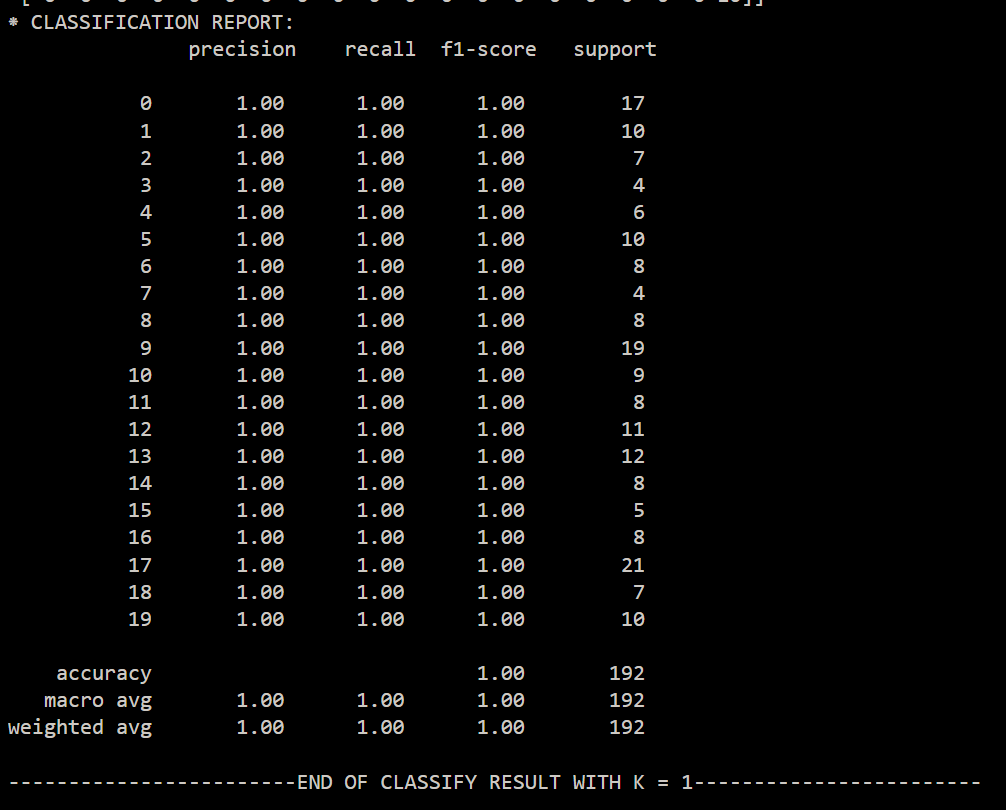


### Iris (K = 5)

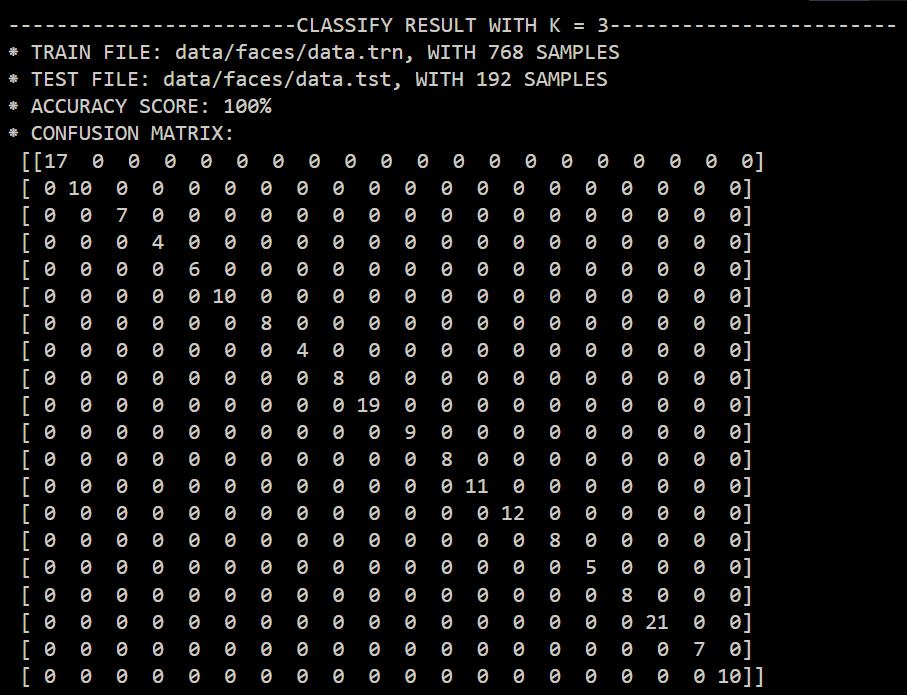


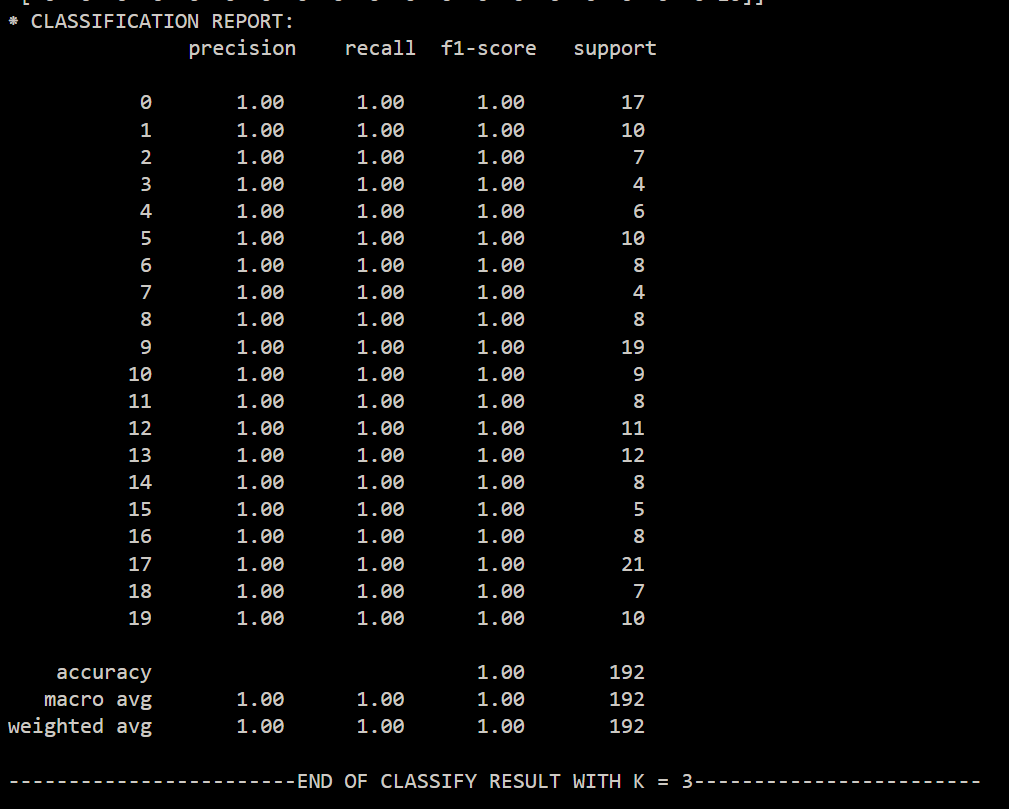
### Faces (K = 1)



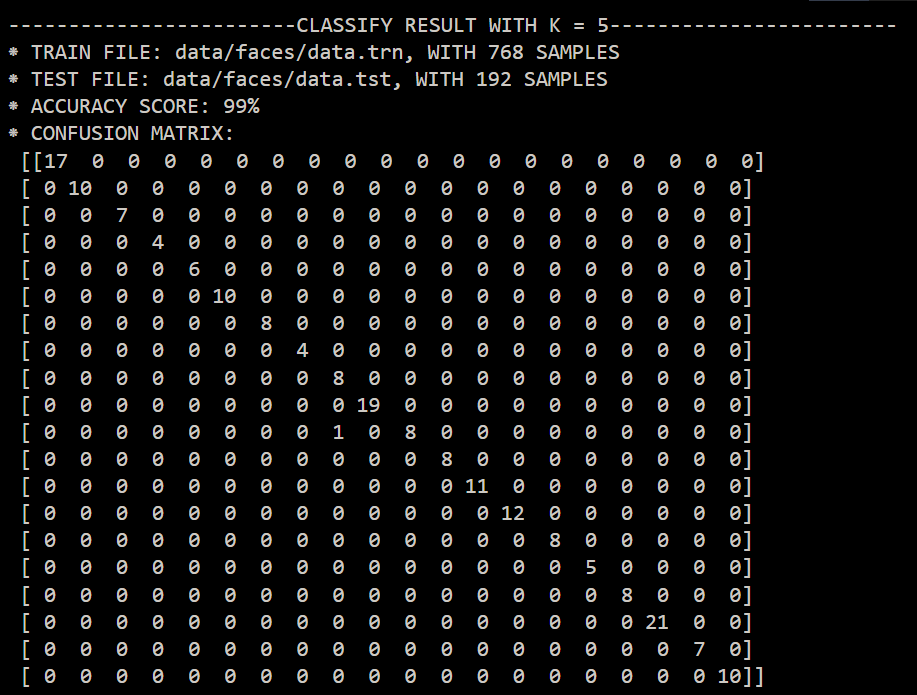


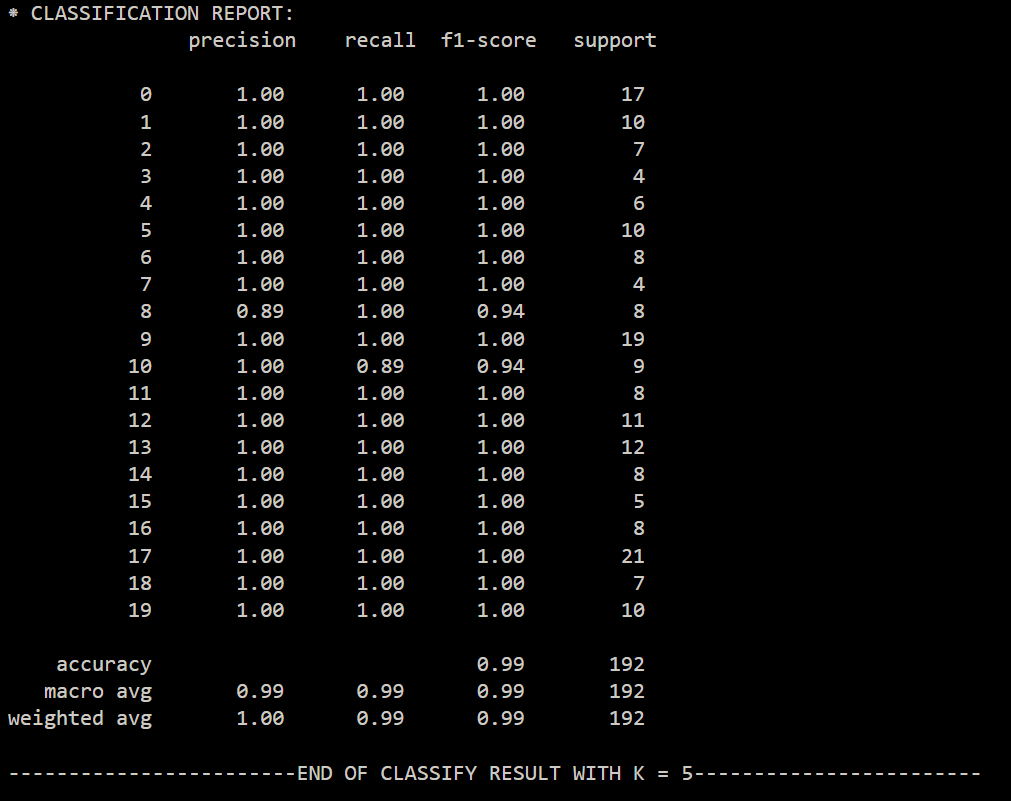
### Faces (K = 3)



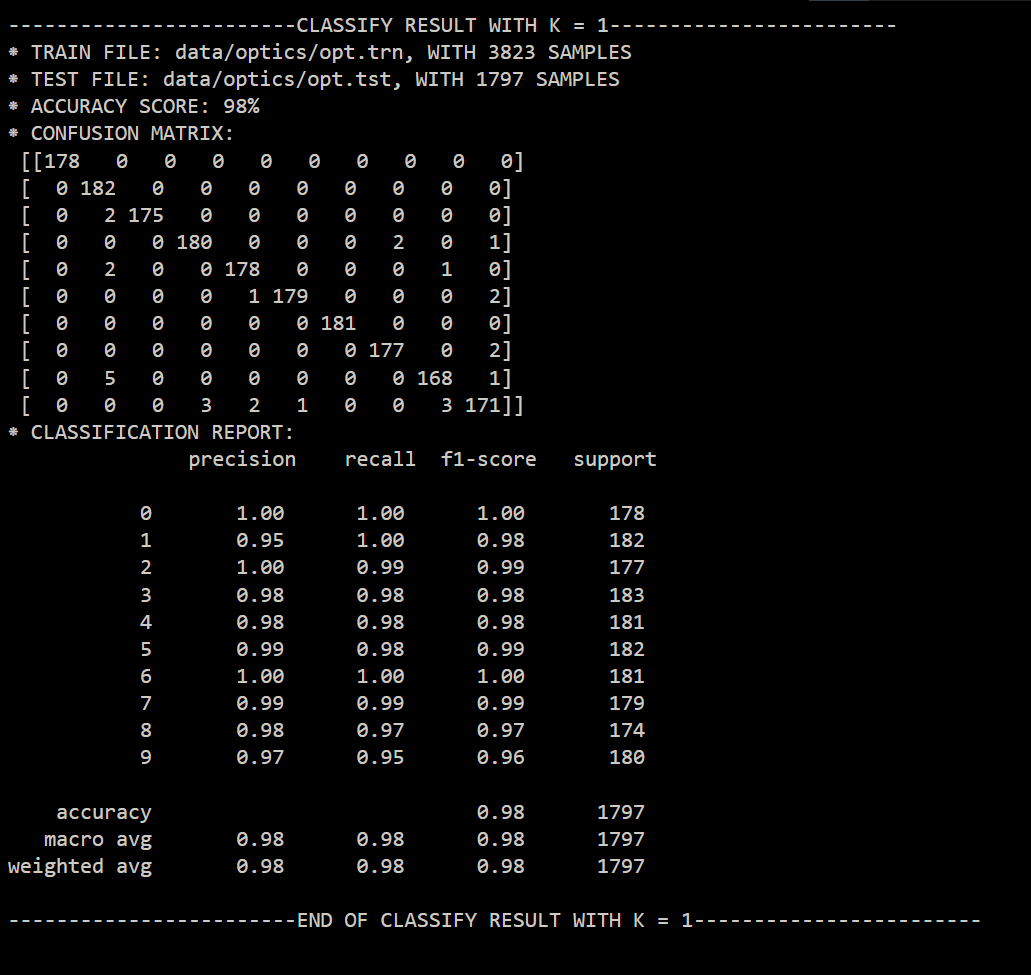


### Faces (K = 5)

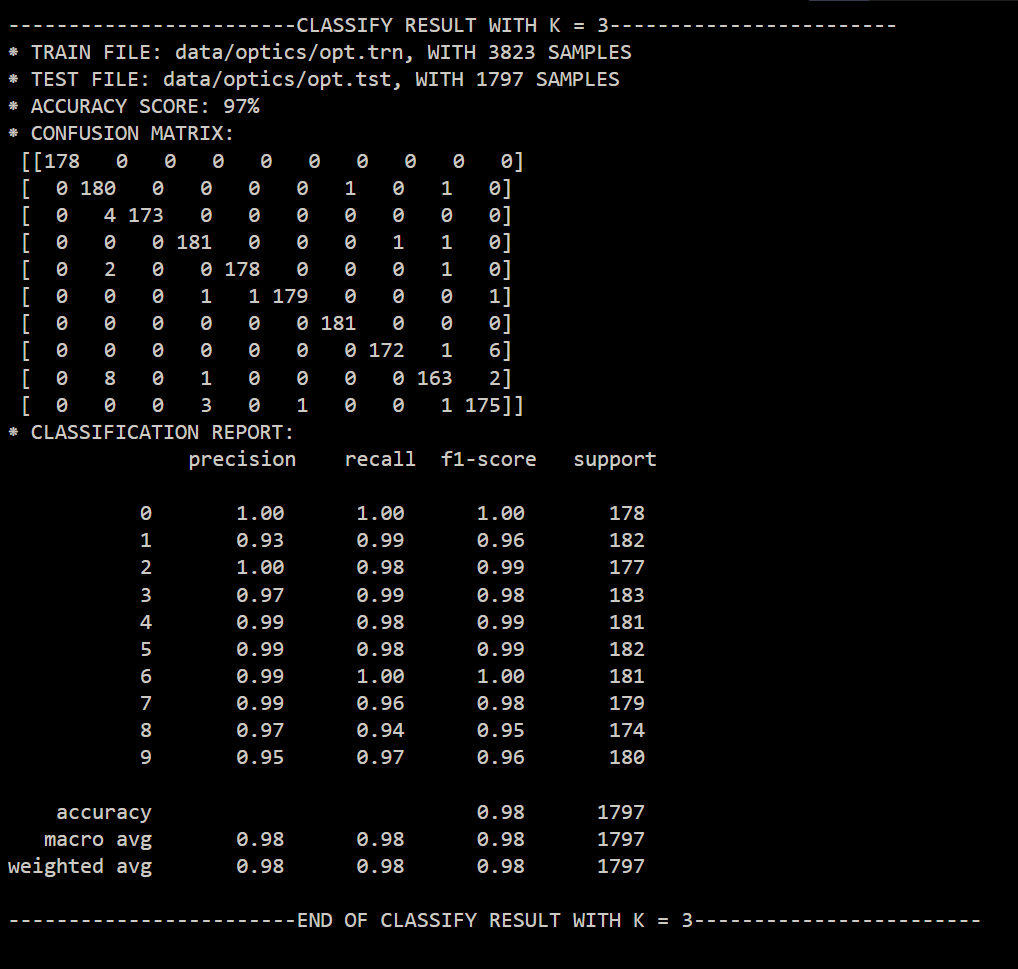




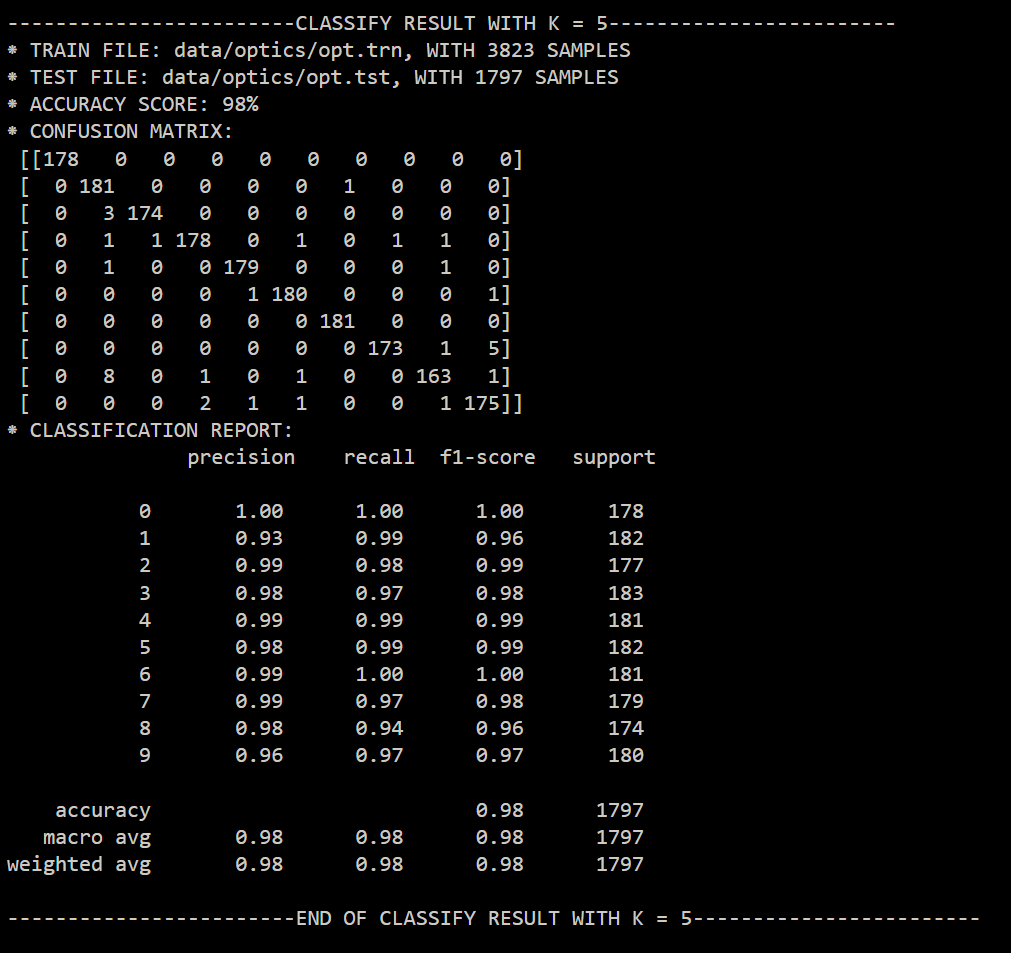
### Optics (K = 1)



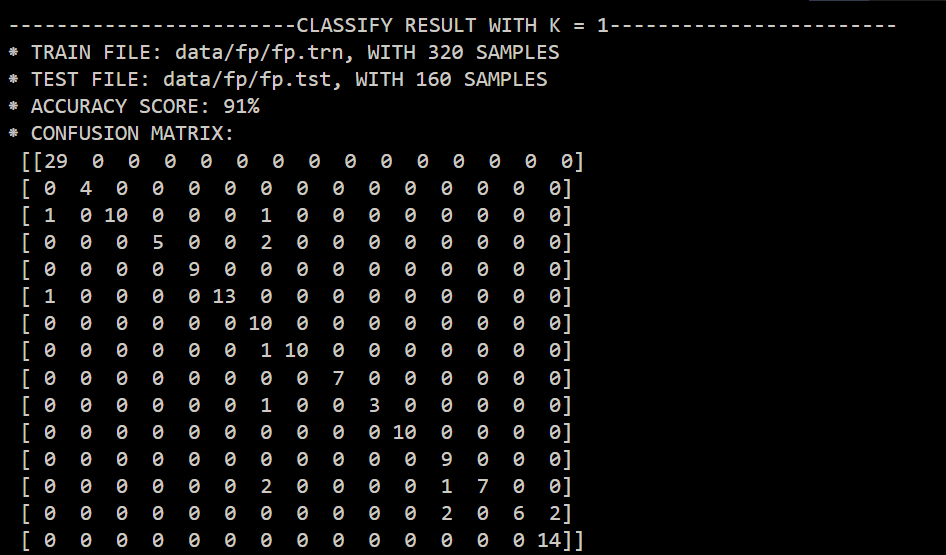
### Optics (K = 3)

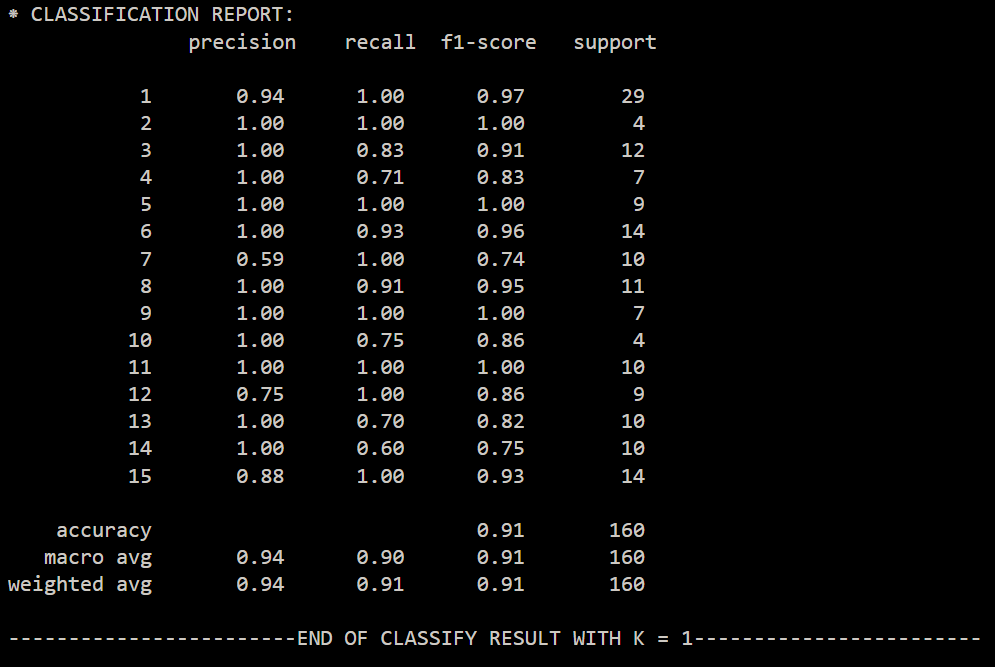


### Optics (K = 5)

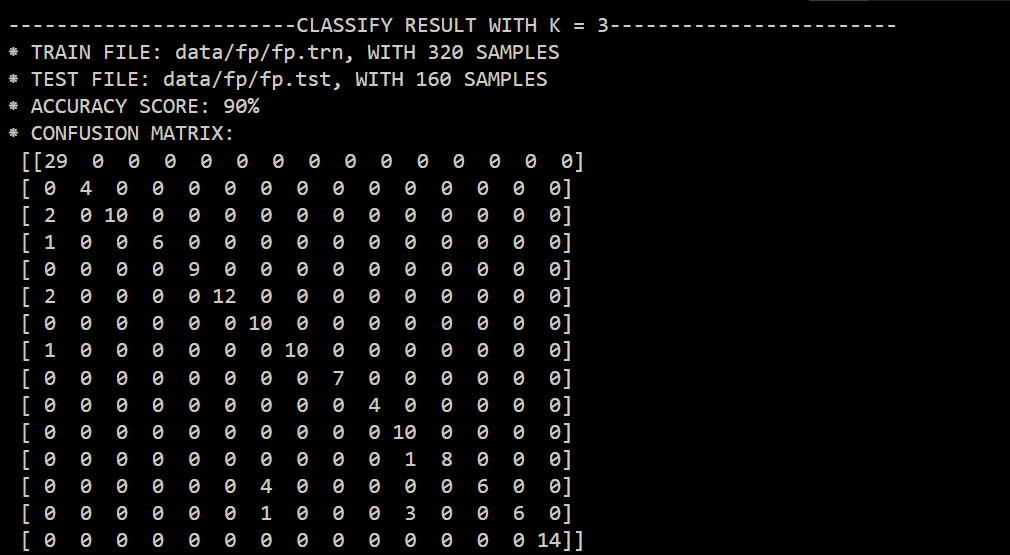


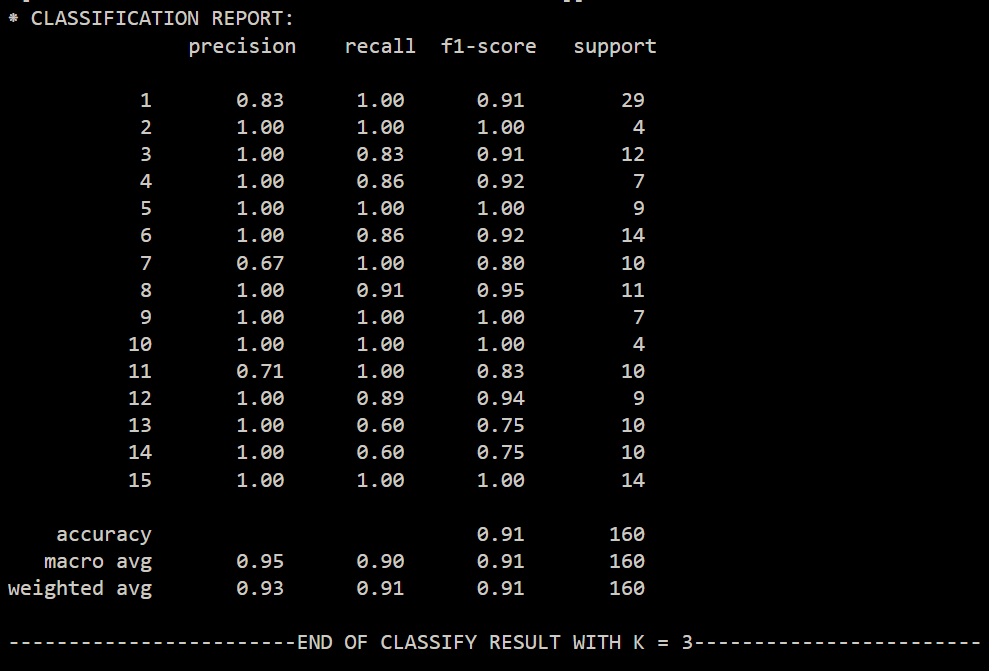
### Fp (K = 1)



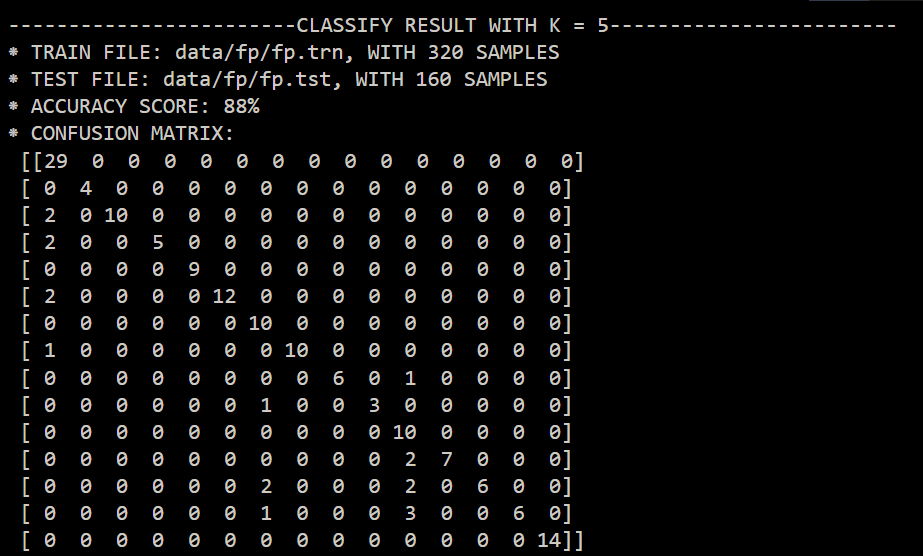


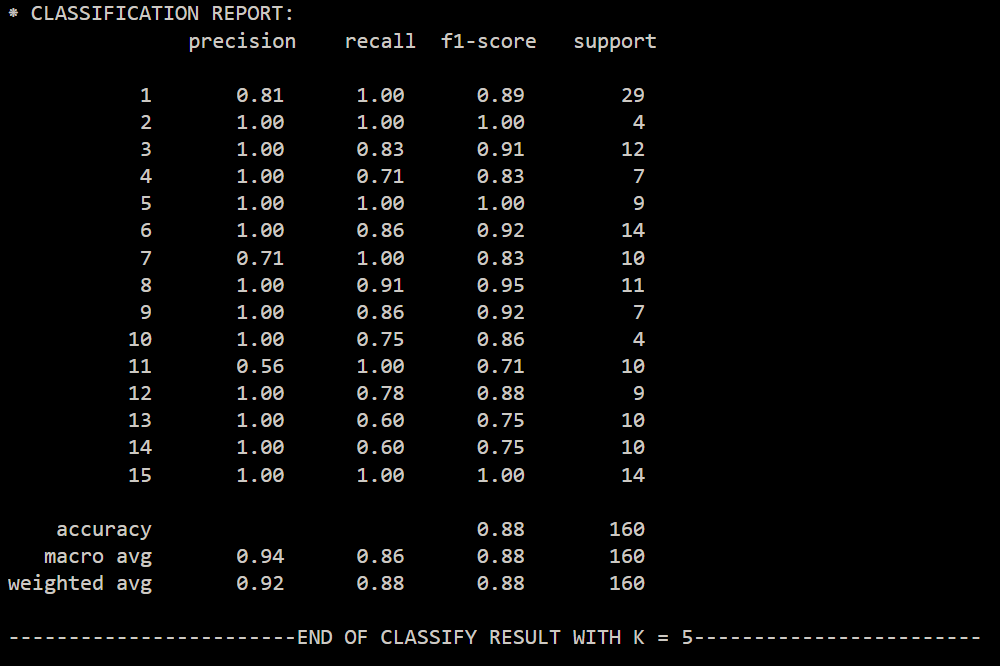
### Fp (K = 3)



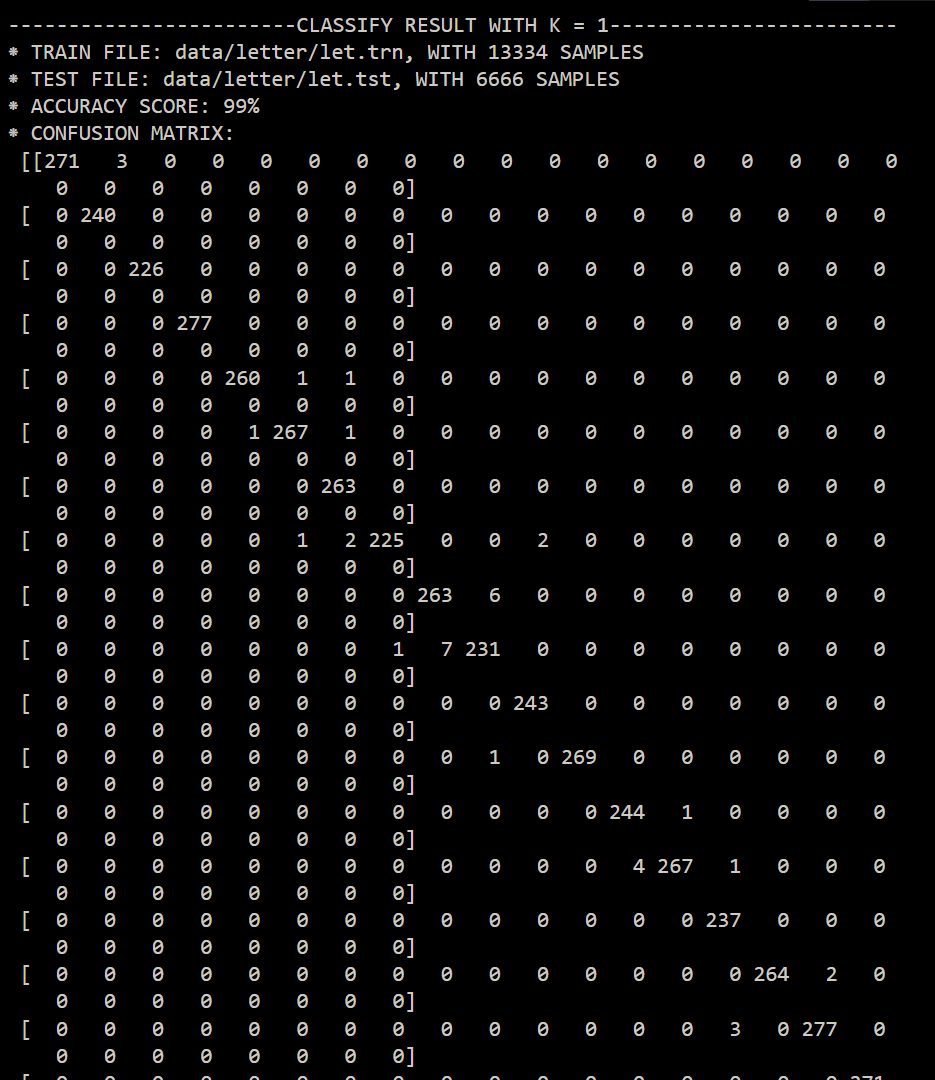


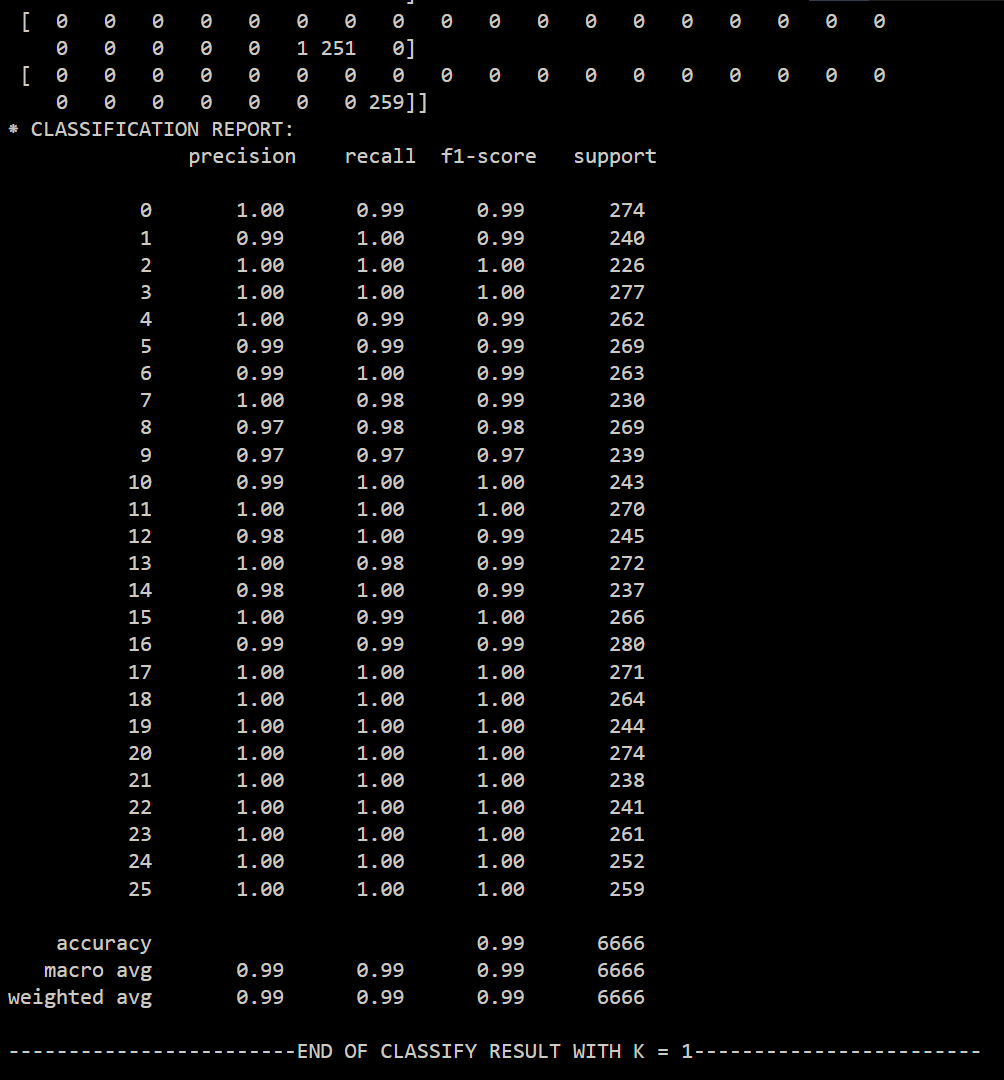
### Fp (K = 5)



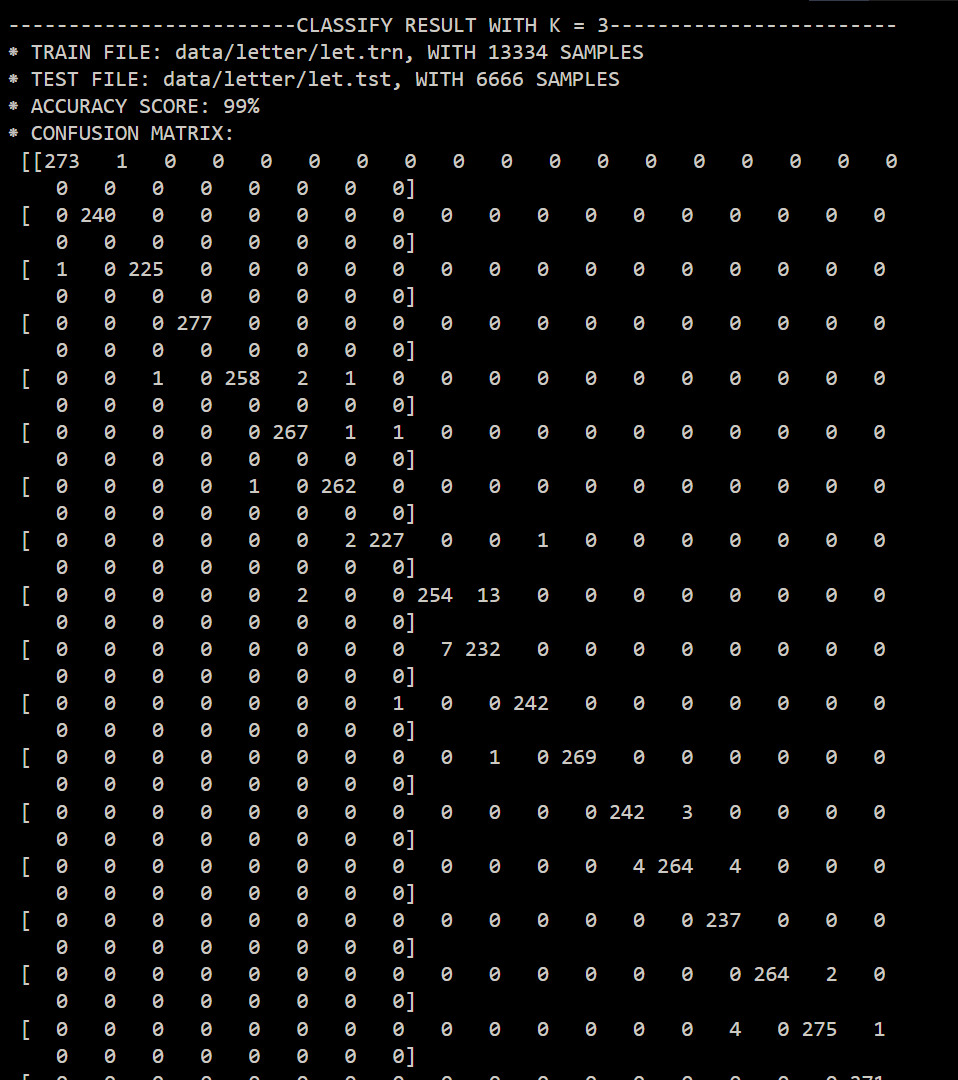


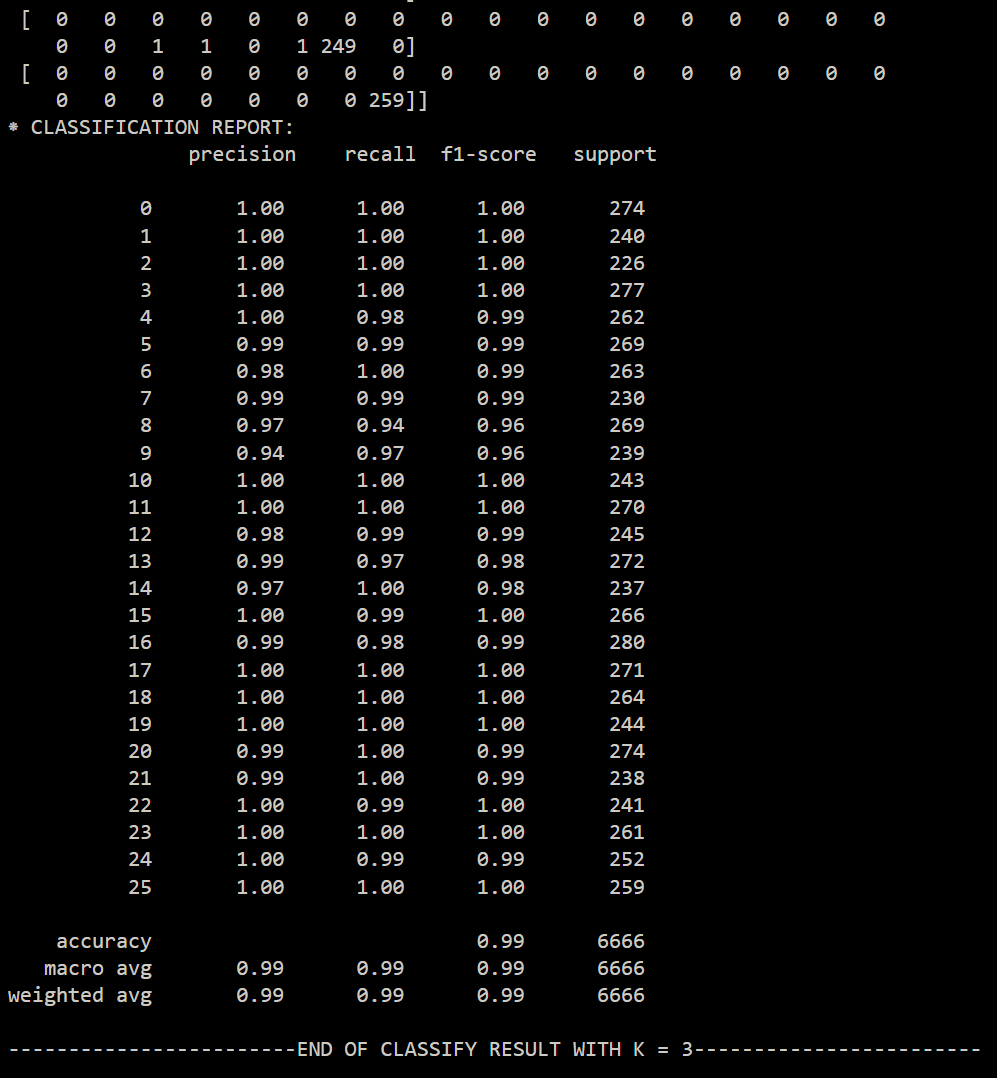
### Letter (K = 1)



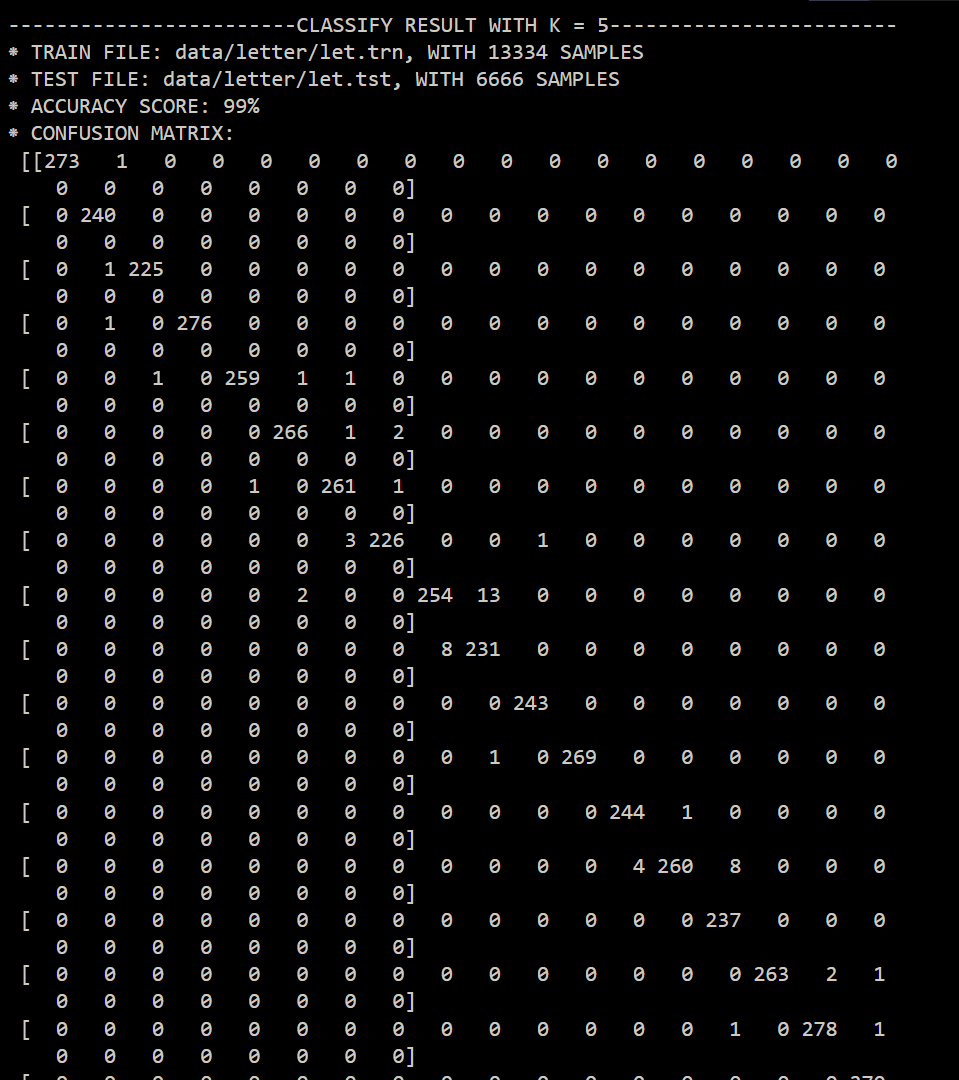


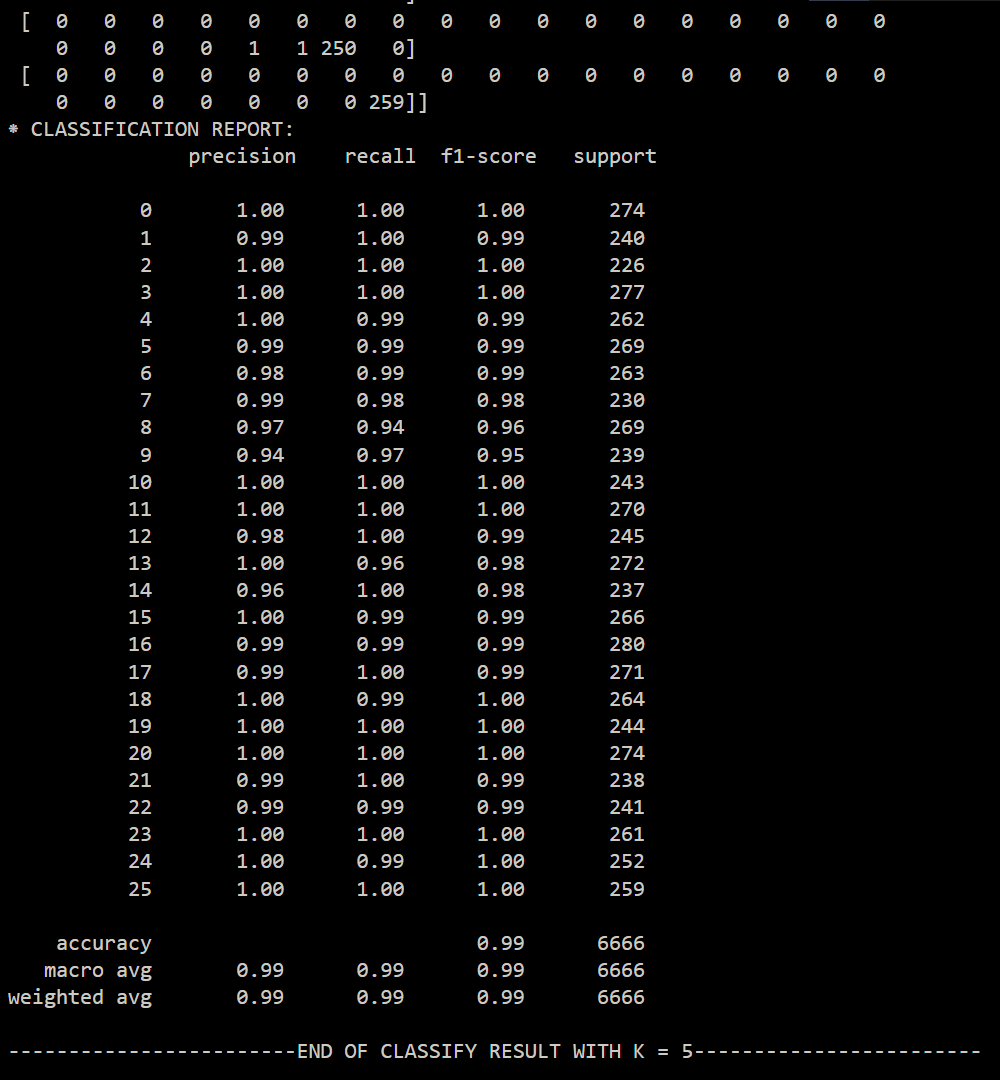
### Letter (K = 3)





### Letter (K = 5)





## Proof of Cover-Hart’s theorem:

Theorem: For sufficiently large training set size n, the error rate of the 1NN classifier is less than twice the Bayes error rate.

Proof: Let be a query point and let be its closest neighbor. The expected error rate of the 1NN classifier is

where is the probability that has label and is the probability that has a different label. The critical fact is that if the number n of training examples is large enough, then the label probability distributions for all and will be essentially the same. In this case, the expected error rate of the 1NN classifier is

To prove the theorem we need to show that

Let and let this maximum be attained with . Then the lefthand side is

and the righthand side is . The summation above is maximized when all values are equal for . The value of the lefthand side is then

Now and so which is what we wanted to prove