Name: Trần Đăng Khoa

ID: B2014926

Class: M03

REPORT ASSIGNMENT 1

1) Given a dataset as follows:

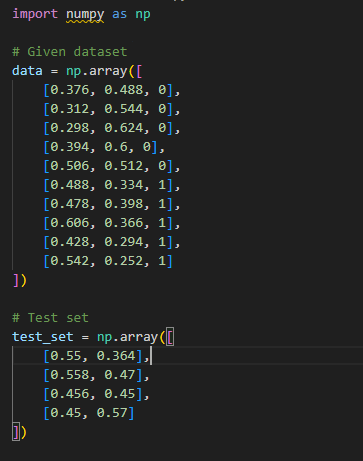
|  |  |  |
| --- | --- | --- |
| **X1** | **X2** | **Class** |
| 0.376000 | 0.488000 | 0 |
| 0.312000 | 0.544000 | 0 |
| 0.298000 | 0.624000 | 0 |
| 0.394000 | 0.600000 | 0 |
| 0.506000 | 0.512000 | 0 |
| 0.488000 | 0.334000 | 1 |
| 0.478000 | 0.398000 | 1 |
| 0.606000 | 0.366000 | 1 |
| 0.428000 | 0.294000 | 1 |
| 0.542000 | 0.252000 | 1 |

- Classifying the testset with *1*NN, *3*NN:

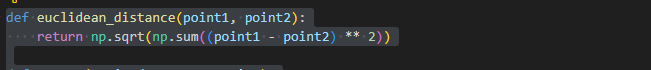
|  |  |  |
| --- | --- | --- |
| **X1** | **X2** | **Class** |
| 0.550000 | 0.364000 | ? |
| 0.558000 | 0.470000 | ? |
| 0.456000 | 0.450000 | ? |
| 0.450000 | 0.570000 | ? |

Answer:

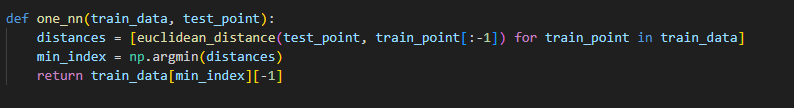
Given a dataset as follow:



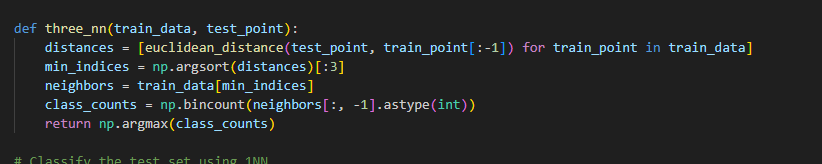
This function calculates the Euclidean distance between two points in a multi-dimensional space. It is used as a measure of similarity between data points.



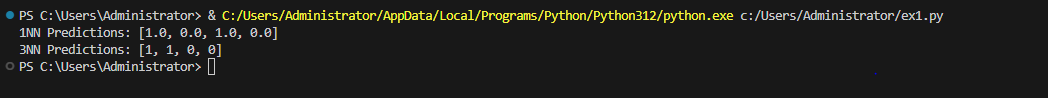
This function implements the 1-nearest neighbor algorithm. It takes training data (train\_data) and a test point (test\_point) as input. For each point in the training data, it calculates the Euclidean distance to the test point. It then finds the index of the point with the minimum distance and returns the class label of that nearest neighbor.



This function is similar to one\_nn but implements the 3-nearest neighbors algorithm. It finds the indices of the three points with the minimum distances, selects those neighbors, and then determines the majority class among these three neighbors using np.bincount.



Result



1. Implement *k*NN from scratch in Python. The program requires 3 parameters:
   * file name of trainset
   * file name of testset
   * number of nearest neighbors (k)

Dataset with *m* examples, *n* dimensions (attribute), *c* classes (*0, 1, …, c-1*), is in the format:

val\_i1\_a1 val\_i1\_a2 … val\_i1\_an class\_i1

val\_i2\_a1 val\_i2\_a2 … val\_i2\_an class\_i2

…

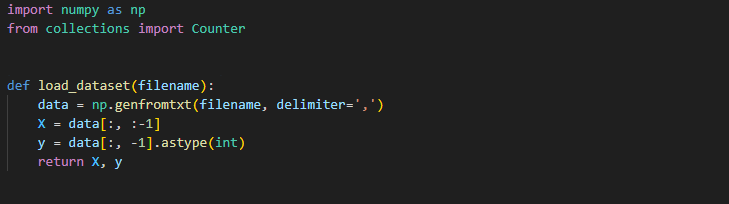
val\_im\_a1 val\_im\_a2 … val\_im\_an class\_im

The program reports the classification results (accuracy, confusion matrix) with different trials k=1, 3, etc for 5 datasets:

* + Iris (**.trn**: trainset, **.tst**: testset)
  + Optics (**.trn**: trainset, **.tst**: testset)
  + Letter (**.trn**: trainset, **.tst**: testset)
  + Face (**.trn**: trainset, **.tst**: testset)
  + Fp (**.trn**: trainset, **.tst**: testset)

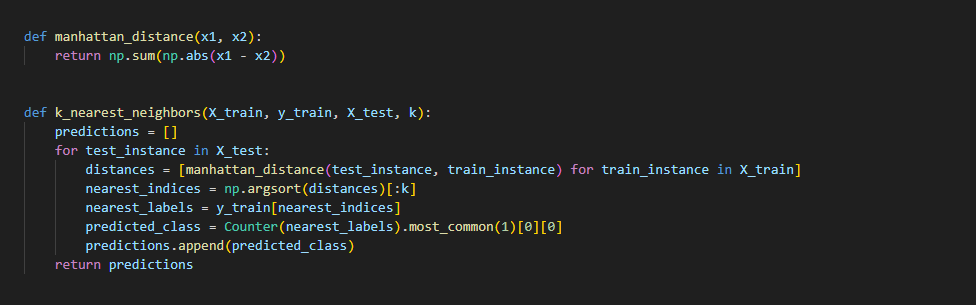
datasets: http://www.cit.ctu.edu.vn/~dtnghi/ml/data.tar.gz

Answer:



Assuming that the last column of the loaded data contains the labels, X = data[:, :-1] extracts all columns except the last one, representing the feature matrix.

y = data[:, -1].astype(int) extracts the last column and converts it to an integer data type, representing the labels.



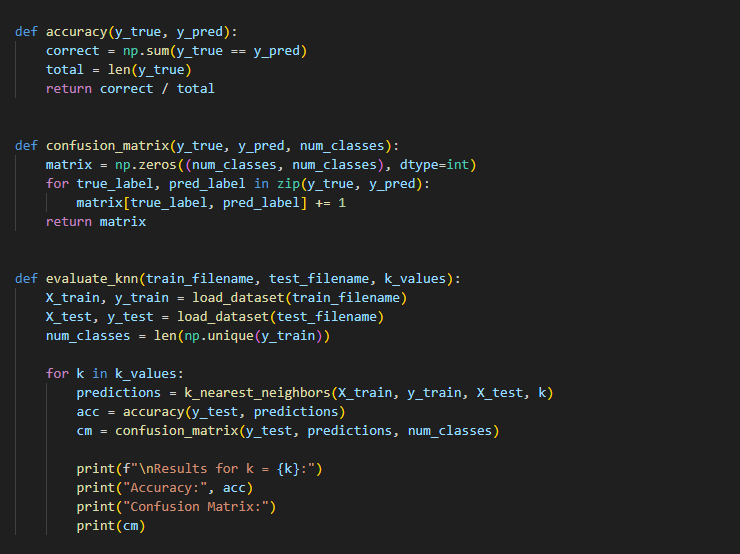
This function calculates the Manhattan distance between two vectors x1 and x2. The Manhattan distance, also known as the L1 distance or taxicab distance, is the sum of the absolute differences between corresponding elements of the vectors.

+ np.abs(x1 - x2): Computes the absolute differences element-wise.

+ np.sum(...): Sum of the absolute differences.

+ nearest\_indices = np.argsort(distances)[:k]: Finds the indices of the k-nearest neighbors based on the sorted distances.

+ nearest\_labels = y\_train[nearest\_indices]: Retrieves the labels of the k-nearest neighbors.



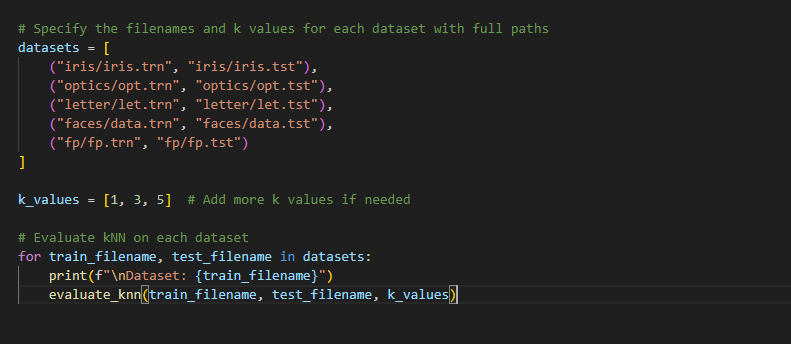
The function calculates the accuracy of a classification model given the true labels (y\_true) and predicted labels (y\_pred).

+ y\_true == y\_pred: Creates a boolean array indicating where the predicted labels match the true labels.

+ np.sum(...): Counts the number of correct predictions.

+ len(y\_true): Represents the total number of instances.

+ The accuracy is calculated as the ratio of correct predictions to the total number of instances.



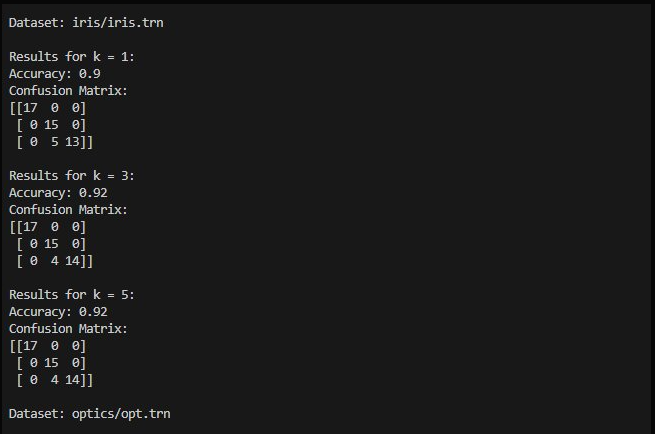
datasets List:

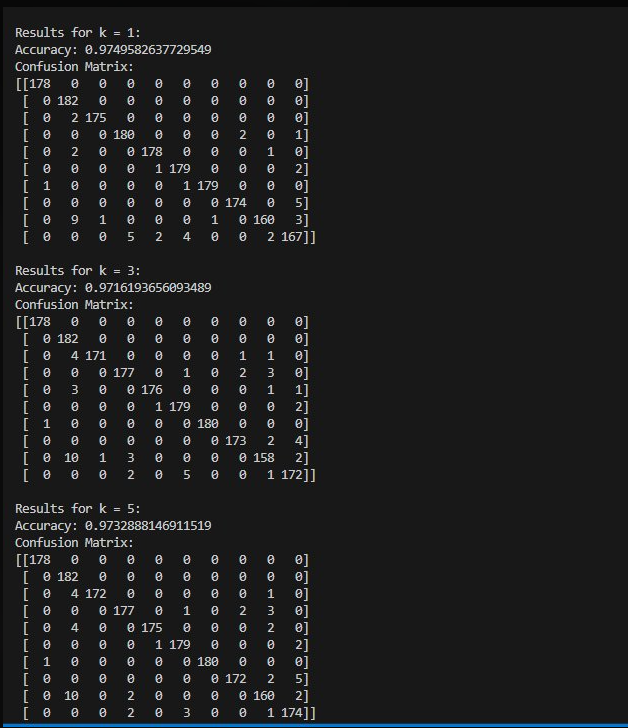
This list contains pairs of training and test dataset filenames for various datasets. Each tuple consists of a training filename and a corresponding test filename.

k\_values List:

This list specifies the values of k for which the kNN algorithm will be evaluated. In this case, it includes 1, 3, and 5. You can add more values as needed.

The result





1. **Proof of Cover-Hart’s theorem:**

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