

**Department of Computer Science and Engineering**

**21st Batch**

**Lab Report 2**

|  |  |
| --- | --- |
| Course title | : Digital Signal Processing Lab |
| Course Code | : CSE-414 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Submitted By** | | **Submitted To** | |
|  |  |  |  |
| Name | : Md. Mahfujur Rahman | Name | : Salma Akter Lima |
| ID | : 192311014 | Designation | : Lecturer(Provisional),  Varendra University,  Rajshahi |
| Section | : A |  |
| Semester | : 10th |  |
| Batch | : 21st | Name | : Sumaiya Tasnim |
|  |  | Designation | : Lecturer(Provisional),  Varendra University,  Rajshahi |
|  |  |  |

------------

|  |  |
| --- | --- |
| Signature | Submission date: 04-08-2022 |
|  |  |

**Problem Statement:** Determining the sports background of a person depending on their height and weight .

**Theory:** In [statistics](https://en.wikipedia.org/wiki/Statistics), the *k*-nearest neighbors algorithm (*k*-NN) is a [non-parametric](https://en.wikipedia.org/wiki/Non-parametric_statistics) [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning) method first developed by [Evelyn Fix](https://en.wikipedia.org/wiki/Evelyn_Fix) and [Joseph Hodges](https://en.wikipedia.org/wiki/Joseph_Lawson_Hodges_Jr.) in 1951, and later expanded by [Thomas Cover](https://en.wikipedia.org/wiki/Thomas_M._Cover). It is used for [classification](https://en.wikipedia.org/wiki/Statistical_classification) and [regression](https://en.wikipedia.org/wiki/Regression_analysis). In both cases, the input consists of the *k* closest training examples in a [data set](https://en.wikipedia.org/wiki/Data_set). The output depends on whether *k*-NN is used for classification or regression:

* In *k-NN classification*, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive [integer](https://en.wikipedia.org/wiki/Integer), typically small). If *k* = 1, then the object is simply assigned to the class of that single nearest neighbor.
* In *k-NN regression*, the output is the property value for the object. This value is the average of the values of *k* nearest neighbors.

*k*-NN is a type of [classification](https://en.wikipedia.org/wiki/Classification) where the function is only approximated locally and all computation is deferred until function evaluation. Since this algorithm relies on distance for classification, if the features represent different physical units or come in vastly different scales then [normalizing](https://en.wikipedia.org/wiki/Normalization_(statistics)) the training data can improve its accuracy dramatically.

Both for classification and regression, a useful technique can be to assign weights to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of 1/*d*, where *d* is the distance to the neighbor.

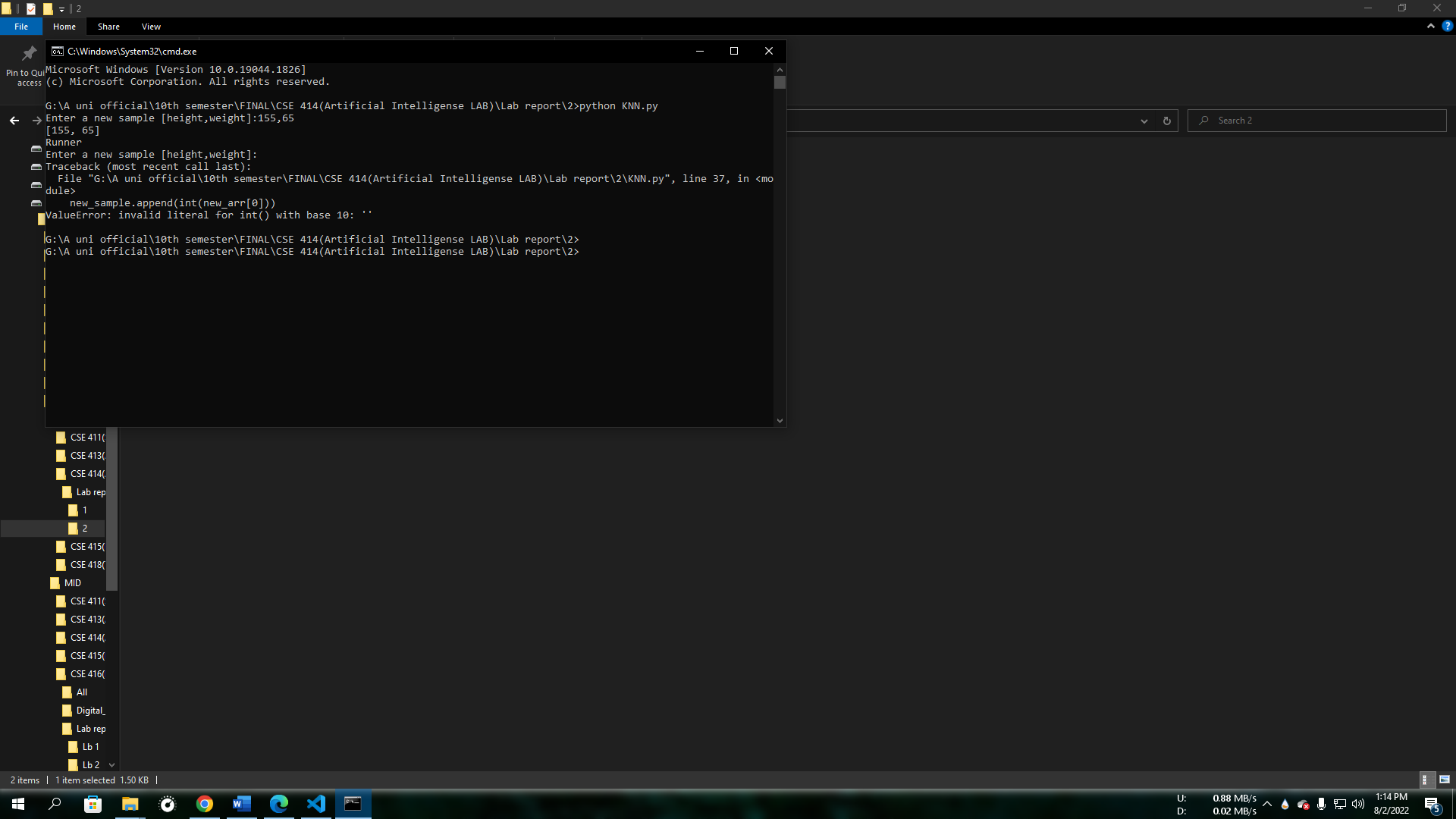
The neighbors are taken from a set of objects for which the class (for *k*-NN classification) or the object property value (for *k*-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

A peculiarity of the *k*-NN algorithm is that it is sensitive to the local structure of the data.

**CODE 1:**

|  |
| --- |
| from turtle import color  import numpy as np  import matplotlib as mtp  from matplotlib import pyplot as plt  from matplotlib import style  from collections import Counter  style.use('fivethirtyeight')  dataset = {      'k':[[210,90],[201,95],[202,97],[203,98],[205,96],[204,99],[207,94]],      'r':[[170,80],[168,82],[175,83],[179,84],[180,79],[173,82],[176,85]],      'c':[[150,60],[158,62],[155,63],[159,64],[140,59],[143,52],[156,55]],      'm':[[130,40],[138,42],[135,33],[139,34],[130,39],[133,42],[136,35]],  }  def scatter\_plot(new\_sample):      for i in dataset:          for j in dataset[i]:              plt.scatter(j[0],j[1],s=100,color=i)      plt.scatter(new\_sample[0],new\_sample[1],color='b',marker = '\*')      plt.show()  def K\_NN(data,predict,k):      distances =[]      for group in data:          for features in data[group]:              euclidian\_distance = np.linalg.norm(np.array(features)-np.array(predict))              distances.append([euclidian\_distance,group])      votes = [i[1]for i in sorted(distances)[:k]]      vote\_result = Counter(votes).most\_common(1)[0][0]      return vote\_result  while True:      new\_arr = input('Enter a new sample [height,weight]:').split(',')      new\_sample = []      new\_sample.append(int(new\_arr[0]))      new\_sample.append(int(new\_arr[1]))      if new\_sample==[0,0]:          break      else:          print(new\_sample)          scatter\_plot(new\_sample)          result = K\_NN(dataset,new\_sample,5)          if result == 'k':              print('Wrestler')          elif result == 'c':              print('Runner')          elif result == 'm':              print('Swimmer')          else:              print('Criketer') |

**Result:**

****

****

**Conclusion :** We have fitted k = 5 in the as per our number of sample was 28 so k = where N is the total number of sample . Here we have fitted k as 5 and as we know from error calculation of knn when k = 5 for the data . The training error rate is **0.0349127** and the test error is **0.0695**. Notice the training error is better than the test error. It’s almost always true that a statistical algorithm will perform better on the data it was trained on that on an independent test set (hence the problem of **overfitting**). The training error goes up as k goes up.The test error goes down then up as k goes up. And so we can say the model **Low Bias** and **High Variance.**