**Department of Computing**

**CS370: Artificial Intelligence**

**Class: BSCS-10AB**

**Lab 09: Nearest Neighbors**

**Date: 13-04-2023**

**Instructor: Dr. Hashir Kiani**

**Lab Engineer: Ms Shakeela Bibi**

# Lab 09: Nearest Neighbors

**Introduction:**

The nearest neighbor algorithm is most suitable for classification tasks. In nearest neighbor approach to learning, the whole training dataset is used for the purposes of prediction for a new data point. For each new data point, its distance from all the data points in the training set is calculated. The prediction is then the label of the data point in the training examples that is closest to the new data point. In the k-nearest neighbor approach, we find k training examples that are closest to the new data point. The most common label/output of those k training examples is the prediction for the new data point.

**Lab Task:**

In this lab, you are supposed to implement the k-Nearest Neighbor Classifier. Data in excel files (both the training and test sets) are uploaded on LMS (same data that was used for previous lab). In the said training and test data files, each row contains data about one instance of a plant category where f

our predictors/attributes are recorded for each plant (namely, leaf length, leaf width, flower length, and flower width), while “plant” is the target class which could be any one of the following at a time: “Arctica” or “Harlequin” or “Caroliniana”.

You are supposed to perform classification with the following values of k.

1. k = 3
2. k = 5
3. k = 7

Figure out how the results vary on the test data (How results vary by increasing the number of k). Please submit your results for the complete test.

**Note:** You are expected to code the kNN algorithm from scratch in Python. However, you are allowed (in fact encouraged) to compare your results with standard implementations available in off-the-shelf libraries and note if (the how/why) your results vary in terms of classification and execution time from that of the standard implementation available in the library you used for comparison.

**Code:**

**# Import required libraries**

**import pandas**

**import math as m**

**from sklearn import preprocessing**

**import numpy as np**

**# Set value of k for k-NN algorithm**

**k = 3**

**# Read training and testing datasets from Excel files**

**excel\_data\_df\_Train = pandas.read\_excel('TrainingSet.xlsx', sheet\_name='iris')**

**excel\_data\_df\_Test = pandas.read\_excel('TestingSet.xlsx', sheet\_name='Sheet1')**

**# Extract required features and labels from training dataset and normalize the values**

**leafLength = excel\_data\_df\_Train.loc[:, "leaf.length"].tolist()**

**leafLength = np.transpose(preprocessing.normalize([leafLength]).tolist())**

**leafWidth = excel\_data\_df\_Train.loc[:, "leaf.width"].tolist()**

**leafWidth = np.transpose(preprocessing.normalize([leafWidth]).tolist())**

**flowerLength = excel\_data\_df\_Train.loc[:, "flower.length"].tolist()**

**flowerLength = np.transpose(preprocessing.normalize([flowerLength]).tolist())**

**flowerWidth = excel\_data\_df\_Train.loc[:, "flower.width"].tolist()**

**flowerWidth = np.transpose(preprocessing.normalize([flowerWidth]).tolist())**

**plantType = np.transpose(excel\_data\_df\_Train.loc[:, "plant"].tolist())**

**plantType = plantType.tolist()**

**# Extract required features and labels from testing dataset and normalize the values**

**leafLengthTest = excel\_data\_df\_Test.loc[:, "leaf.length"].tolist()**

**leafLengthTest = np.transpose(**

**preprocessing.normalize([leafLengthTest]).tolist())**

**leafWidthTest = excel\_data\_df\_Test.loc[:, "leaf.width"].tolist()**

**leafWidthTest = np.transpose(preprocessing.normalize([leafWidthTest]).tolist())**

**flowerLengthTest = excel\_data\_df\_Test.loc[:, "flower.length"].tolist()**

**flowerLengthTest = np.transpose(**

**preprocessing.normalize([flowerLengthTest]).tolist())**

**flowerWidthTest = excel\_data\_df\_Test.loc[:, "flower.width"].tolist()**

**flowerWidthTest = np.transpose(**

**preprocessing.normalize([flowerWidthTest]).tolist())**

**plantTypeTest = np.transpose(excel\_data\_df\_Test.loc[:, "plant"].tolist())**

**plantTypeTest = plantTypeTest.tolist()**

**# Create an empty list to store distances**

**distanceList = list()**

**# Loop through each instance in the testing dataset**

**for i in range(len(excel\_data\_df\_Test)):**

**# Loop through each instance in the training dataset**

**for j in range(len(excel\_data\_df\_Train)):**

**# Calculate Euclidean distance between the features of the testing and training instances**

**distance = m.sqrt(pow(leafLengthTest[i]-leafLength[j], 2)+pow(leafWidthTest[i]-leafWidth[j], 2)**

**+ pow(flowerLengthTest[i]-flowerLength[j], 2)+pow(flowerWidthTest[i]-flowerWidth[j], 2))**

**# Add the distance and the corresponding label to a list**

**distanceList.append(distance)**

**# Combine the distances and the corresponding labels into a 2D array and sort the array by distance**

**con = np.vstack((distanceList, plantType))**

**con = np.transpose(con)**

**lis = sorted(con, key=lambda x: x[0])**

**# Reset count variables for each instance in the testing dataset**

**countArctica, countCarolinian, countHarlequin = 0, 0, 0**

**# Loop through the k nearest neighbors and increment the count for each class**

**for y in range(k):**

**if (lis[y])[1] == "Arctica":**

**countArctica = countArctica+1**

**elif (lis[y])[1] == "Carolinian":**

**countCarolinian = countCarolinian+1**

**elif (lis[y])[1] == "Harlequin":**

**countHarlequin = countHarlequin+1**

**if max(countHarlequin, countArctica, countCarolinian) == countArctica:**

**plantTypeTest[i] = ("Arctica")**

**elif max(countHarlequin, countArctica, countCarolinian) == countCarolinian:**

**plantTypeTest[i] = ("Carolinian")**

**elif max(countHarlequin, countArctica, countCarolinian) == countHarlequin:**

**plantTypeTest[i] = ("Harlequin")**

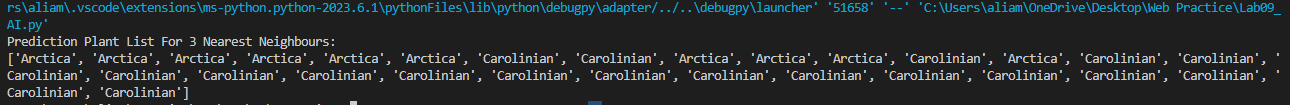
**lis.clear()**

**distanceList.clear()**

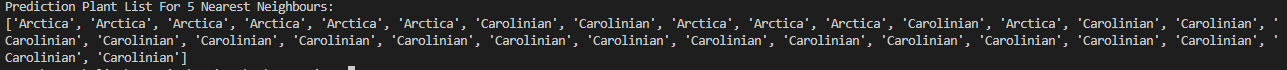
**print("Prediction",i+1,":",plantTypeTest[i])**

**Screenshots:**

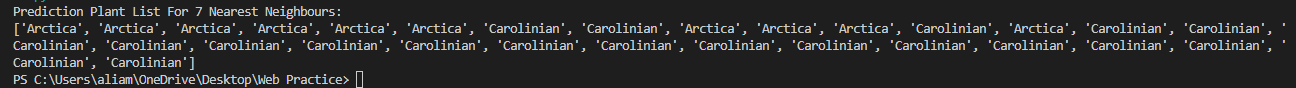
* **For k=3**

****

* **For k=5**

****

* **For k=7**

****

**Findings:**

* **The code iterates over each sample in the test set and calculates the distance between that sample and each sample in training set using Euclidean distance.**
* **It then sorts the training set samples based on their distances to the test sample and selects the k-nearest neighbours.**
* **The code counts the occurrences of each plant type among the k-nearest neighbours and assigns the predicted plant type based on the majority vote.**
* **Finally, the code clears the distance and sorted list for the next iteration and prints the predicted plant type for the i-th test sample.**