**Assignment 2**

**Description of Algorithms**

**KNN Algorithm**:

In the beginning of the algorithm, dataset will be loaded from the sample file given using the CSV library and it will be then divided into a list of training set and test set of 15000 and 5000 points respectively.

Using the numpy library, the training and test sets will be converted to n-dimensional array and will fetch nTrain, nTest, trainX, trainY, testX, testY using the numpy operations on the training and test set formulated initially. These variables will be passed on to the function testKnn.

**testknn** function will internally invoke the function **findNeighbours** to find out the k nearest neighbor of the test set.

findNeighbours will loop for each of the test sets(containing 5000 points) and will find the distance of each point from the trainingSample using the Euclidian distance formula. The distance will be sorted and will be appended in a list of tuples with label of the training Set so we will have a list of tuples like below after the findNeighbours is evaluated. The first element to the tuple is the testData and the next element is a list of tuples with the nearest neighbors.

***('G', [('C', 3.1622776601683795), ('T', 3.605551275463989)])***

Post this we will find out the nearest neighbor of a testData based on the value of k.

**Draw Case**: In case of a draw we handle this by selecting the testData Label appearing first in the list.

We will do a performance evaluation by comparing the value of expected label of G using the N Nearest Neighbor and the actual value of the test Data.

Program is running in approximately 12 minutes for all the samples.

Sample Output:

Prediction Percentage for K= 1 and Sample Size 100 is 42.06

Prediction Percentage for K= 3 and Sample Size 100 is 39.4

Prediction Percentage for K= 5 and Sample Size 100 is 36.94

Prediction Percentage for K= 7 and Sample Size 100 is 35.5

Prediction Percentage for K= 9 and Sample Size 100 is 34.92

**1NN Condensed Algorithm:**

Here in this case we will first try to identify a condensed set for which the training data is consistent. For doing that we will first create a set of 1 training Data and then make it consistent with the training set first. Once we have the consistent training set we will test the test data against the condensed training set. For selecting the first condensed sample we will use a random function.

**Note**: This program is not properly functioning and is taking hours to finish the execution because of the high values of N.

**Challenges**: Using a brute force algorithm to determine the consistent condensed set is causing an issue for the algorithm. Since we are testing each condensed set to the training sample so which is causing a major contribution to the inefficiency of the program.

condenedSet function is currently not invoked from the code because of the issues in algorithm and giving wrong output.