Assignment 1: K-Nearest Neighbors (KNN) From Scratch

Sourav Sarker (201892985) & Isaac Adeuwon (200925352)

To implement for KNN from scratch, we have used libraries named: sys, pandas, itemgetter. In this assignment, we used euclidean distance function and to determine the class we counted the majority vote. The command to execute the program from the command prompt:

```
$ python3 KNN.py  TrainingData_A1.data TestingData_A1.data k
```

We have broken down into the following steps for this assignment:

- **Data Handling**
- **Find the Similarity**
- **Nearest Neighbors**
- **Predict Response**
- **Conditional Probability of the Response**

Data Handling: In this section, we loaded the provided data which is TrainingData_A1.tsv TestingData_A1.tsv We used '**Pandas'** for handling the data and converted those into lists. We separated the TrainingData_A1 from the 'Class' column.

```
1. function loadData
2. requires train_Data, test_Data, k, row
3. full_training_set ← trainData.toList
4. rd_train_data ← traindata.toList
5. test_data ← testData.tolist
6. for classData in rd_train_data
7. delete ClassData[LastColumn]
```

Find the Similarity: To make predictions we need to measure the similarity between TrainingData_A1 and TestData_A1. For that, we need to measure Euclidean Distance, and we calculated the distance with each row of the TestData_A1 and each row of TrainingData_A1. Put those distances into a list. After that, we merged that with the TrainingData_A1 and sorted in ascending order.

```
1. function euclidean_distance
2. requires train_data, test_data
3. for each element of each row of the test_data - for each element of each row
   of the train_data
      a. square of the result
      b. root mean square
4. new list euc_distance [] ← included all the distance
5. for the length of the train_data
      a. merged the euc_distance for each row of rd_test_data
      b. sort the list into → sorted_full_set
```

Nearest Neighbors: So, now we have the similarity measurement. We can find the neighbours based on the value of K and the smallest value of the distances. We made a list of those closest class to the given unseen instances.

```
1. For each value of the range k
       a. selected_class ← take 'Class' column
       b. append a list with the 'Class', which is new_list[ ]
```

Predict Response: When we located the nearest neighbours for the test instances, now we can predict the response which is 'Class'.

```
1. count the maximum iterated 'Class' from the new_list
2. Print that 'Class' as the predicted response
```

Conditional Probability of the Response: We calculated a ratio of the maximum iterated class and the value of the K to find the Conditional Probability of the Response.

```
1. A new list → count_number[ ]
2. Count of the iterated 'Class' in the new_list
3. put each of the count into → count_number
4. calculate the accu_knn by (maximum number of count_number /length of the
   new_list)* 100
5. print →  accu_knn
```

In the main function, we took three arguments, one input for the row number of the testing data for the response and call the **load_data(trainingSet, testingSet, K, R).**

```
1. define a function → main()
2. requires R
3. set the default value of K=3, if no argument pass
4. call the function load_data with trainingSet, testingSet, K, R

→ for item in range 9
→ Call main(item)
```

Validation of my KNN Algorithm:

For validation, I opened the test and training files in excel, and by using Excel's formulas and sorting algorithms, I manually compared and counted the top results for each of the samples in the test data. Afterwards, I manually compared the results in the excel to those generated by my algorithm and confirmed that for this instance, my algorithm was correct.