Dalia Ibrahim

Carlos Dasaed Salcedo 201892008

Assignment 1: Implementation of KNN

To execute the program, you must run the following command:

*$python3 KNN.py [TrainingData.File] [TestData.File] [k]*

The command will execute the following pseudocode:

1. Copy training\_file and the testing\_file into as pandas dataframes
2. Display the following Menu:
   1. Select an Option from the menu
      1. Run Classical KNN
      2. Normalize KNN
      3. Run Weighted KNN
      4. Validate Classical KNN
3. If option “i.” is selected:
   1. The function classicalKNN ( *[TrainingData] [TestData] [k]*  ) is executed with the initial input already converted into frames
      * 1. Create a DataFrame called “FinalOutput”, where the final output will be store while the function is executed
        2. For *i* in the rows of *TestData:*
           1. Create variable AllDistancePerTest
           2. For *j* in the rows in the *TrainingData:*

Append to AllDistancePerTest , the ouput of EuclideanDistance ( *i* , *j*)

EuclieanDistance finds the distance between row *i* and row *j, and*

Returns the distance, and the class of row j (example: [ 1.23, 1])

* + - * 1. When the for cycle in b. is done, we have the distance between row *i* in TestData and each of the training data rows stored in AllDistancePerTest. At this point we sort AllDistancePerTest from the closest element to the farthest, and select only the top k results.
        2. Calculate\_Nearest\_neighbour is executed with the top k results obtained in the previous step:

Calculate\_Nearest\_neighbour adds the number of times each class is found in the top results and selects the class with the most votes.

Calculate\_Nearest\_neighbour then takes the number of times the top voted class was selected, and divides it by K to determine the MaxConditionalProbability

The output of Calculate\_Nearest\_neighbour which is the most voted class, and the MaxConditionalProbability, is stored in the data frame “finalOutput”

* + - 1. Once the for cycles are done, finalOutput is stored as a csv file for later retrieval.

\begin{algorithm}[H]

\caption{KNN Function}

\begin{algorithmic}

\IF{$leftIntensity <leftIntensityOld \OR rightIntensity <rightIntensityOld$}

\STATE $rightSpeed \gets 0$

\STATE $ leftSpeed \gets k \times rightIntensity$

\ELSIF {$ (rightIntensity \geq 252 ) \OR (leftIntensity \geq 254)$}

\STATE $leftSpeed \gets 0$

\STATE $ rightSpeed \gets 0$

\ELSIF {$rightIntensity > leftIntensity$}

\STATE $rightSpeed \gets 0$

\STATE $ leftSpeed \gets k \times rightIntensity;$

\ELSIF {$leftIntensity > rightIntensity$}

\STATE $leftSpeed \gets 0 $

\STATE $rightSpeed \gets \dfrac{K}{2} \times leftIntensity$

\ELSE

\STATE $leftSpeed \gets k \times rightIntensityOld $

\STATE $ rightSpeed \gets k \times leftIntensityOld $

\ENDIF

\STATE $rightIntensityOld \gets rightIntensity;$

\STATE $ leftIntensityOld \gets leftIntensity; $

\end{algorithmic}

\end{algorithm}