Assignment-2

Topic : Distance Algorithms, Encoding, K-NN

Ishvarya G

In this program for questions 1,3 and 4, I have made the code modularized specific to each question’s functionality. The main function provides the functionality of taking input from the user’s end for each function and a call is made to the respective function to perform the operation, the result of which is displayed using the main function. However for implementation of the K-NN classifier there is no main function. The code uses custom fit(), predict() functions without using any externel ML libraries (except pandas that is used to load in the train and test csv files) for its implementation.

**Question 1:**

*To calculate the Euclidean and Manhattan distance between 2 vectors whose dimensions are variable :*

***Explanation:***

The code uses 2 For loops to loop through the range of numbers from 0 till the length of the vector. Using the formula of taking the sum of the squares of differences of elements of the 2 given vectors Euclidean distance is calculated. Similarly, for finding the Manhattan distance, the sum of the absolute differences of the vector elements is taken. Finally the calculated values are returned.

***Pseudocode:***

Function findEuclidean( INPUT vec1,vec2):  
 SET euclidean=0

FOR i in the range of length of vec2:

#since the length of both the vectors must be equal : length of vec1 = length of vec2

SET euclidean = euclidean + (vec2[i] – vec1[i]) \*\* 2

REPEAT until i reaches the length of vec2

RETURN sqrt(euclidean)

Function findManhattan( INPUT vec1,vec2):  
 SET manhattan=0

FOR j in the range of length of vec2:

#since the length of both the vectors must be equal : length of vec1 = length of vec2

SET manhattan= manhattan + abs(vec2[j] – vec1[j])

REPEAT until i reaches the length of vec2

RETURN sqrt(manhattan)

**Question 2:**

*To implement K-NN classifier using a function :*

***Explanation:***

The code defines a KNNclassifier class that initializes k (no.of neighbours to check for), fit() that just stores the train and test data to be used during predictions and the predict() function that has the function to calculate the Euclidean distances between each feature of the training and test samples and returns distances\_for\_test\_point list, which is further sorted in ascending order. Then, we extract the indices of the first ‘k’ closest test points and also extract its corresponding labels. Among these, we find the majority of all the labels ( taking the max()) and add it to the predictions\_list for the test set.

***Pseudocode :***

class KNNclassifier:

Function \_\_init\_\_(INPUT k):

Initialize k

Function fit(INPUT X\_train, y\_train):

Store X\_train and y\_train as instance variables

Function predict(INPUT X\_test,X\_train):

Function findEuclidean(vec1, vec2):

#same as in Question 1

Convert train and test set to NumPy arrays

SET predictions\_list to []

FOR test\_point in test\_features:

SET distances\_for\_test\_point to []

FOR train\_point in train\_features:

SET distance to the result of the Euclidean distance between train and test points

Append to the distances\_for\_test\_point list

SET sorted\_indices to indices sorted in test\_features according to the distances

SET k\_nearest\_indices to the first k closest indices

SET unique\_labels = set(k\_nearest\_neighbours)

SET prediction = max(unique\_labels, count of k\_nearest\_labels as the key)

Append predictions to predictions\_list

RETURN predictions\_list

Load the data

Create the KNN classifier object

Call fit() method on X\_train and y\_train

Call the predict() method on X\_test and X\_train

Print the returned predictions

Print the accuracy

**Question 3:**

*To convert categorical variables to numeric using label encoding :*

***Explanation:***

This function aims to encode the categorical inputs to numeric values by converting assigning a unique number to each unique label in the data. It first finds the unique\_labels by using the set() method and uses For loop to iterate through the unique labels and sets the corresponding index as its value.

***Pseudocode :***

Function encodeLabel (INPUT categ):

SET label\_mapping = {}

SET unique\_labels to the unique labels in categ using set()

FOR index ,label in enumerate(unique\_labels):

SET label\_mapping[label] to index

SET encoded\_labels to [label\_mapping[label] for label in categ]

RETURN encoded\_labels

**Question 4:**

*To convert categorical variables to numeric using One-hot encoding :*

***Explanation:***

This function aims to encode the categorical inputs to numeric values by converting assigning a binary value to each unique label in the data. It first finds the unique\_labels by using the set() method then creates a list of lists that is filled with 0s whose row length is equal to the number of categorical variables and the column length is equal to the number of unique variables. We set the label\_mapping to be equal to the corresponding index value. Then using 2 For loops we check if the value of j (iterable) matches the index of the label for current element in categ (categ[i]) , if so it sets the element in position i, j to 1. Finally it returns the encoded\_labels.

***Pseudocode:***

Function encodeOneHot (INPUT categ):

SET unique\_labels to the unique labels in categ using set()

SET encoded\_labels to 0 where the rows is the length of items in categ and columns is the length of items in unique\_labels

SET label\_mapping to {}

FOR index ,label in enumerate(unique\_labels):

SET label\_mapping[label] to index

FOR i in range of length of categ:

FOR j in range of length of unique\_labels:

Check if j matches the index of the label for the current element in categ:

SET encoded\_labels[i][j] = 1

RETURN encoded\_labels