Machine Learning

22AIE213

Assignment – 2

Submited by : Team 4

P.Anju chowdary-Aie22170

T.Mahithi Reddy-Aie22178

Akmal-Aie22150

Q.1)Write a function to calculate the Euclidean distance and Manhattan distance between two vectors. The vectors dimension is variable. Please don't use any distance calculation functions available in Python.

Pseudo code :

FUNCTION Euclidean_Distance(Vector1, Vector2):

For all vectors that are not equal in size to Vector2

THROW ValueError("Vectors should be the same dimensionality")

END IF

Squared_Distances = [(a - b)**2 FOR (a, b) IN zip(Vector1, Vector2)]

Sum_Of_Squares = SUM(Squared_Distances)

Euclidean_Distance = SQRT(Sum_Of_Squares)

RETURN Euclidean_Distance

END FUNCTION

FUNCTION Manhattan_Distance(Vector1, Vector2):

If the length of Vector1 is not equal to the length of Vector2, THEN

RAISE ValueError("Vectors must have same dimension")

END IF

Manhattan_Distance = SUM(F-abs(a - b) FOR a, b IN zip(Vector1, Vector2))

RETURN Manhattan_Distance

END FUNCTION

FUNCTION get_user_input():

Proffer the 1st vector (Use a comma to separate the values)

Please input the 2nd vector (comma-separated values): Vector2 =

Vector1 = [FLOAT(x.strip()) for x in TOKPLIST(Vector1, COMMA-SEP)]

Vector2 = list(map(float, SPLIT(Vector2, ',')))

RETURN Vector1, Vector2

END FUNCTION

Vector_a, Vector_b = get_user_input()

Euclidean_Result = Euclidean_Distance(Vector_a, Vector_b)

Manhattan_Result = Manhattan_Distance(Vector_a, Vector_b)

OUTPUT "Euclidean Distance: Euclidean_Result

OUTPUT "Manhattan Distance: String(result())

Explanation :

Function Euclidean Distance(Vector1, Vector2): If the length of Vector1 is equal to the length of Vector2, then raise ValueError("Vectors should be in the same dimensionality").

The squared distances between Vector1 and Vector2 are computed using the formula (a - b)^2, where a and b are corresponding elements from Vector1 and Vector2, respectively. The sum of these squared distances is then calculated as Sum_Of_Squares. Finally, the Euclidean distance is obtained by taking the square root of Sum_Of_Squares.If we are going to calculate the Euclidean distance, then we are going to use the function RETURN Euclidean_Distance END FUNCTION.

Function Manhattan_Distance(Vector1, Vector2): If the length of Vector1 is not equal to the length of Vector2, then raise the ValueError("Vectors should be in the same dimensionality"). End if The formula to calculate the Manhattan Distance between two vectors is as follows: Manhattan Distance = SUM(ABS(a - b) FOR a, b IN zip(Vector1, Vector2)).

* Using data base these 3 questions are answered :

Q.2)Write a function to implement k-NN classifier. k is a variable and based on that the count of neighbors should be selected.

Pseudo code :

```
FUNCTION calculate_euclidean_distance(vector1, vector2):

IF length(vector1) != length(vector2) THEN

RAISE ValueError "Vectors must have the same dimensionality"

END IF

squared_diff = SUM((x - y)^2 FOR x, y IN zip(vector1, vector2))

RETURN SQRT(squared_diff)

END FUNCTION

FUNCTION knn_classifier(training_data, labels, test_instance, k)

distances = [(index, euclidean_distance(training_instance, test_instance))

FOR index,training_instance IN EVERY(training_data)]

distances.SORT(key=LAMBDA x: x[1])

k_nearest_indices = [index FOR index, _ IN distances[:k

k_nearest_labels = [labels[index] for index in k_nearest_indices if index < length (labels)]

label_counts = COUNTER(k_nearest_labels)

majority_label = label_counts.max(key=label_counts.get) if label_counts else "No Valid Labels"

RETURN majority_label

END FUNCTION

df = READ_CSV(file_path)

numerical_data = EXTRACT_NUMERIC(df["Memory"])
```

labels = ([INPUT_LABEL(i)] for i, in ENUMERATE(numerical_data))

test_instance_values = EXTRACT_NUMERIC(INPUT("Enter test instance (comma-separated numeric values):)

Check if numeric values are valid.

IF NOT test_instance_values THEN

PRINT "Error: The test instance has provided no valid numeric values.

ELSE:

k = INPUT_INT("Enter the value of k:)

predicted_label = nearest_knn_classifier(numerical_data, labels, test_instance_values, k)

PRINT "Predicted Label:", predicted_label

Explanation :

Pseudocode implements a KNN classifier, in which the Euclidean distance is the similarity metric. calculate_euclidean_distance function works out the difference between the two vectors by calculating the square root of the sum of squared difference between the corresponding dimensions. This ensures that the input vectors have the same dimensionality to maintain consistency in the comparison. The knn_classifier function uses this distance metric to find the k-nearest neighbors in the training data for a given test example. The neighbors are found by sorting the distances and choosing the indices of k smallest elements. The function carries out a majority vote on the labels of these neighbors, which serve as the prediction of the label of the test instance.

The method involves reading a dataset that is represented by the pandas DataFrame (df), extracting numerical data from a particular column such as "Memory," and receiving user input of labels for the training instances. The test instance is taken as a comma-separated input of numeric values. The code guarantees the validity of the provided number values for the trial case. The user is asked to input the value of k, which determines the number of neighbors taken into account in the classification. Finally, the knn_classifier function is run with the training data, labels, test instance, and k, thus making the prediction of the test instance’s label, which is then displayed.

The KNN classifier is a popular, versatile algorithm that is widely used in machine learning for tasks like classification and regression. Its strength is its use of the local patterns in the data, which makes it very useful for dealing with non-linear decision boundaries and uneven density in the feature space. The pseudocode demonstrates a concise and modular approach that outlines the main steps of the KNN algorithm for classification using Euclidean distance.

Q.3)Write a function to convert categorical variables to numeric using label encoding. Don’t use any existing functionalities.

Pseudo code :

```
FUNCTION Label_encode_categorical(Data):

Unique_Categories = LIST(SET(Data))

Category_Mapping = {Category: uniqueCategories.indexOf(i, Category)

Numeric_Representation = [Category_Mapping[Category] FOR Category in Data]

RETURN Numeric_Representation

END FUNCTION

file_path = 'C:C:\Users\pinni\OneDrive - Amrita vishwa vidyapeetham\jupyter projects\ML\Amazon EC2 Instance Comparison.csv

df = READ_CSV(file_path)

Categorical_Column = 'Arch'

Use Label Encoder to the column of your choice

df[Categorical_Column + '_encoded'] = Label_encode_categorical(df[Categorical_Column]).rename(columns={'index': Categorical_Column + '_encoded'})

PRINT "Original Data:"

PRINT df[[Categorical_Column]]

PRINT "Numeric Representation:"

PRINT df[[Categorical_Column + '_encoded']]
```

Explanation :

The Label_encode_categorical function was meant to apply label encoding on a given categorical data column. This begins with retrieving the unique categories in the data and creating a dictionary (Category_Mapping) to map each category to a numeric index. The function then creates a list of the numeric representation corresponding to each category in the original data defined in the mapping dictionary.

In the main program, the dataset is loaded from a CSV file, and 'Arch' is chosen as the categorical column to be encoded using label encoding. The Label_encode_categorical function is applied to the current column to create a new column with the suffix '_encoded' to store the numerical representations. In the end, the original and coded data are printed to show the results of the transformation. One of the prevalent pre-processing methods in machine learning is label encoding, which is used to transform categorical data into a suitable format for algorithms that need numerical input, such as decision trees or neural networks.

Q.4)Write a function to convert categorical variables to numeric using One-Hotencoding. Don't use any existing functionalities.

Pseudo code :

FUNCTION One_Hot_Encode_Categorical(Data, Categorical_column1):

Unique_Categories = Data[Categorical_column1].UNIQUE()

FOR category IN Unique_Categories:

Data[Categorical_column1 + '_' + STRING(category)] = "0"

FOR i, category IN ENUMERATE(Unique_Categories):The government financed the first reforestation programme in the country.

DATA[DATA.catagorical_column1 == 'category'].at[catagorical_column1 + '_' + STRING(category), 'catagorical_column1'] = 1

Data = Data.DROP(Categorical_column1, AXIS=1)

RETURN Data

END FUNCTION

file_path = 'C:\Users\pinni\OneDrive - Amrita vishwa vidyapeetham\jupyter projects\ML\Amazon EC2 Instance Comparison.csv'

df = READ_CSV(file_path)

Select a column to perform one-hot encoding (replace 'Arch' with your column name)

Categorical_column1 = 'Linux Virtualization'

Perform one-hot encoding for the selected column

```
df = One_Hot_Encode_Categorical(df, Categorical_column1)

PRINT "One-Hot Encoded Data:"

PRINT df
```

Explanation :

One_Hot_Encode_Categorical function is used for one-hot encoding a particular categorical column in a DataFrame. It begins with getting the unique categories in the specified column and then creates new columns for each unique category, which is initially set to zero. The function sets the 1 and 0 values on the corresponding one-hot encoded column according to the presence of the category in the original column. Finally, after the categorical column is dropped, the modified DataFrame is returned

In the main program, the data from the CSV file is loaded, and a particular categorical column called ‘Linux Virtualization’ is extracted for one-hot encoding. The One_Hot_Encode_Categorical function is applied to this column, which generates a DataFrame with the original categorical column replaced by one-hot encoded columns. This approach has value in machine learning when dealing with categorical data as it enables algorithms to read and learn from the categorical information in a numerical form.