MACHINE LEARNING

ASSIGNMENT-2

SYED ABRAR C.	AIE22147
SACHIN R.	AIE22142
NITHESH NAIR P.S.	AIE22133

Question 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.

Pseudocode:

```
function Euclidean_distance(x, y):
    distance = 0
    for i from 0 to length(x) - 1 do:
        distance += (x[i] - y[i])^2
    return square_root(distance)
function Manhattan_distance(x, y):
    distance = 0
```

```
    for i from 0 to length(x) - 1 do:
        distance += absolute_value(x[i] - y[i])
    return distance
x = [1, 2, 3, 4]
y = [1, 2, 3, 4]
Result_Euclidean = Euclidean_distance(x, y)
print(Result_Euclidean)
Result_Manhattan = Manhattan_distance(x, y)
print(Result_Manhattan)
```

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

Pseudocode:

```
function Euclidean_distance(x, y):
    distance = 0
    for i from 0 to length(x) - 1 do:
        distance += (x[i] - y[i])^2
    return square_root(distance)
function Manhattan_distance(x, y):
    distance = 0
    for i from 0 to length(x) - 1 do:
        distance += absolute_value(x[i] - y[i])
    return distance
```

```
x = [1, 2, 3, 4]
y = [1, 2, 3, 4]
Result_Euclidean = Euclidean_distance(x, y)
print(Result_Euclidean)
Result_Manhattan = Manhattan_distance(x, y)
print(Result_Manhattan)
```

Question 3: Write a function to convert categorical variables to numeric using label encoding. Don't use any existing functionalities

Pseudocode:

```
function label_encoding(inputdata):
    encoding_map_dictionary = empty dictionary
    encoded_data = empty list
    seed_random_number_generator(50)
    for each obj in inputdata do:
        if obj is not in encoding_map_dictionary then:
            encoding_map_dictionary[obj] = generate_random_integer()  # Map random float to integer
        append encoding_map_dictionary[obj] to encoded_data
    return encoded_data
function read_dataset(filename):
    dataset = empty list
    open filename as file
        lines = read_lines(file)
        for each line in lines[1:] do:
            row = split line by ','
```

```
            append row to dataset
    return dataset
function main():
    filename = "C:\\Users\\SACHIN.R\\Downloads\\archive\\cars_ds_final_2021.csv"
    iris_dataset = read_dataset(filename)
    species_column = extract column 15 from iris_dataset
    encoded_species = label_encoding(species_column)
    print "Original species data:", species_column
    print "Encoded species data:", encoded_species
if _name_ == "_main_":
    call main()
```

Question 4: Write a function to convert categorical variables to numeric using One-Hotencoding. Don’t use any existing functionalities.

Pseudocode:

```
function read_csv(file_path):
    open file at file_path for reading as file
        lines = read_lines(file)
        header = split lines[0] by ','
        data = []
        for each line in lines[1:] do:
            row = split line by ','
            append row to data
    return header, data
function print_dataframe(header, data):
    print join(header with ',')
    for each row in data do:
```

```
        print join(row with ',')
function one_hot_encode(column):
    unique_values = list(set(column))
    encoding_dict = empty dictionary
    for each unique_value in unique_values do:
        encoding_dict[unique_value] = length(encoding_dict)
    encoded_data = empty list
    for each value in column do:
        encoding = create list of zeros with length(unique_values)
        encoding[encoding_dict[value]] = 1  # Set corresponding index to 1
        append encoding to encoded_data
    return encoded_data
file_path = 'C:\\Users\\SACHIN.R\\Downloads\\archive\\cars_ds_final_2021.csv'
header, data = read_csv(file_path)
categorical_column_name = 'Drivetrain'
column_index = find index of categorical_column_name in header
categorical_column = create list of values in column_index from data
encoded_data = one_hot_encode(categorical_column)
for each unique_value in set(categorical_column) do:
    new_column_name = concatenate categorical_column_name with '_i'
    append new_column_name to header
    new_column = create list of one-hot encoded values for unique_value
    for each row in encoded_data do:
        append value of new_column to row
print_dataframe(header, data)
```