LAB ASSIGNMENT 2

MACHINE LEARNING

TEAM MEMBERS

NITHIN SAGAR - AIE22136

ADISH VAIBHAV – AIE22102

SUSHANTH – AIE22127

Question 01
Pseudocode and Explanation

1) "euclidean_dist" function:
a) Verifies equality of lengths for vectors v1 and v2, raising a ValueError if they differ.
b) Computes squared differences for corresponding elements in v1 and v2.
c) Calculates the sum of squared differences.
d) Takes the square root of the sum to derive the Euclidean distance.
e) Returns the computed Euclidean distance.

2) "manhattan_dist" function:
a) Ensures equality of lengths for vectors v1 and v2, raising a ValueError if they differ.
b) Computes absolute differences for corresponding elements in v1 and v2.
c) Determines the sum of absolute differences to yield the Manhattan distance.
d) Returns the calculated Manhattan distance.

3) Defines two example vectors, vector_a and vector_b.
4) Invokes the euclidean_dist function with these vectors and prints the result.
5) Invokes the manhattan_dist function with these vectors and prints the result.

Question 02
Pseudocode and Explanation

1) "euclidean_dist" function:
a) Accepts two numpy arrays, v1 and v2.
b) Calculates the Euclidean distance between v1 and v2 using the formula: "square root of (v1-v2)^2".
c) Returns the calculated Euclidean distance.

2) "kNN" function:
a) Takes four parameters: X_train (training data features), Y_train (training data labels), X_test (test data features), and k (number of neighbors to consider).
b) Initializes an empty list, predictions, to store predicted labels for the test data.
c) Iterates through each test_case in X_test.
d) Computes Euclidean distances between the test_case and all points in X_train.
e) Selects the k-nearest neighbors based on sorted distances.
f) Uses Counter to determine the most common label among the k neighbors.
g) Appends the most common label to the predictions list.
h) Returns an array of predictions.

3) Defines example training and test data.
4) Sets the value of k (value_of_k).
5) Calls the kNN function with the provided data and prints the predicted labels for the test data.

Question 03
Pseudocode and Explanation

1) "label_encoding" function:

a) Takes a list or numpy array data as input, containing categorical variables.
b) Creates a list of unique components (unique_components) in the input data.
c) Generates a dictionary (category_to_label_mapping) mapping each unique category to its corresponding numeric label, utilizing the enumerate function.
d) Creates a numpy array (encoded_data) where each element is the numeric label corresponding to its original category in the input data.
e) Returns the tuple (encoded_data, category_to_label_mapping).

2) Defines example categorical data (categ_data).
3) Calls the label_encoding function with the provided data.
4) Prints the original categorical data, the encoded data, and the label mapping.
5) This process illustrates how categorical variables can be converted into numeric labels using label encoding.

Question 04
Pseudocode and Explanation

1) "one_hot_encoding" function:
a) Takes a list or numpy array data as input, containing categorical variables.
b) Creates a list of unique components (unique_components) in the input data.
c) Generates a dictionary (component_to_index_mapping) mapping each category to its corresponding index.
d) Initializes a numpy array (one_hot_encode) with zeros, where the number of rows is the length of the input data and the number of columns is the count of unique categories.
e) Iterates through each row of the one-hot encoded array, setting the element at the corresponding column index to 1 based on the mapping of categories to indices.
f) Returns the tuple (one_hot_encode, unique_components).

2) Defines example categorical data (categorical_data).
3) Calls the one_hot_encoding function with the provided data.
4) Prints the original categorical data, the one-hot encoded data, and the list of unique categories.
5) This process demonstrates how categorical variables can be converted into a one-hot encoded representation.