**Name: Varsha JJ**

**Roll No: 48**

**Batch: B**

**Date: 15/09/2022**

**DATA SCIENCE LAB**

**Experiment No.: 3**

**Aim**

Implement KNN algorithm using python.

**Procedure**

# Example of making predictions

from math import sqrt

# calculate the Euclidean distance between two vectors

def euclidean\_distance(row1, row2):

  distance = 0.0

  for i in range(len(row1)-1):

    distance += (row1[i] - row2[i])\*\*2

  return sqrt(distance)

# Locate the most similar neighbors

def get\_neighbors(train, test\_row, num\_neighbors):

  distances = list()

  for train\_row in train:

    dist = euclidean\_distance(test\_row, train\_row)

    distances.append((train\_row, dist))

  distances.sort(key=lambda tup: tup[1])

  neighbors = list()

  for i in range(num\_neighbors):

    neighbors.append(distances[i][0])

  return neighbors

# Make a classification prediction with neighbors

def predict\_classification(train, test\_row, num\_neighbors):

  neighbors = get\_neighbors(train, test\_row, num\_neighbors)

  output\_values = [row[-1] for row in neighbors]

  prediction = max(set(output\_values), key=output\_values.count)

  return prediction

# Test distance function

dataset = [[2.7810836,2.550537003,0],

  [1.465489372,2.362125076,0],

  [3.396561688,4.400293529,0],

  [1.38807019,1.850220317,0],

  [3.06407232,3.005305973,0],

  [7.627531214,2.759262235,1],

  [5.332441248,2.088626775,1],

  [6.922596716,1.77106367,1],

  [8.675418651,-0.242068655,1],

  [7.673756466,3.508563011,1]]

prediction = predict\_classification(dataset, dataset[0], 3)

print('Expected %d, Got %d.' % (dataset[0][-1], prediction))

# Example of making predictions

from math import sqrt

# calculate the Euclidean distance between two vectors

def euclidean\_distance(row1, row2):

  distance = 0.0

  for i in range(len(row1)-1):

    distance += (row1[i] - row2[i])\*\*2

  return sqrt(distance)

# Locate the most similar neighbors

def get\_neighbors(train, test\_row, num\_neighbors):

  distances = list()

  for train\_row in train:

    dist = euclidean\_distance(test\_row, train\_row)

    distances.append((train\_row, dist))

  distances.sort(key=lambda tup: tup[1])

  neighbors = list()

  for i in range(num\_neighbors):

    neighbors.append(distances[i][0])

  return neighbors

# Make a classification prediction with neighbors

def predict\_classification(train, test\_row, num\_neighbors):

  neighbors = get\_neighbors(train, test\_row, num\_neighbors)

  output\_values = [row[-1] for row in neighbors]

  prediction = max(set(output\_values), key=output\_values.count)

  return prediction

# Test distance function

dataset = [[2.7810836,2.550537003,0],

  [1.465489372,2.362125076,0],

  [3.396561688,4.400293529,0],

  [1.38807019,1.850220317,0],

  [3.06407232,3.005305973,0],

  [7.627531214,2.759262235,1],

  [5.332441248,2.088626775,1],

  [6.922596716,1.77106367,1],

  [8.675418651,-0.242068655,1],

  [7.673756466,3.508563011,1]]

prediction = predict\_classification(dataset, dataset[0], 3)

print('Expected %d, Got %d.' % (dataset[0][-1], prediction))

**Output**

****