Lab Program 9: Write a program to implement k-Nearest Neighbor algorithm to classify the iris data set. Print both correct and wrong predictions. Java/Python ML library classes can be used for this problem

```
import csv
import random
import math
import operator
def loadDataset(filename, split, trainingSet=[], testSet=[]):
        with open(filename) as csvfile:
           lines = csv.reader(csvfile)
           dataset = list(lines)
           for x in range(len(dataset)-1):
              for y in range(4):
                 dataset[x][y] = float(dataset[x][y])
              if random.random() < split:
                 trainingSet.append(dataset[x])
              else:
                 testSet.append(dataset[x])
def euclideanDistance(instance1, instance2, length):
        distance = 0
        for x in range(length):
                distance += pow((instance1[x] - instance2[x]), 2)
        return math.sqrt(distance)

def getNeighbors(trainingSet, testInstance, k):
        distances = []
        length = len(testInstance)-1
        for x in range(len(trainingSet)):
                dist = euclideanDistance(testInstance, trainingSet[x], length)
```

```
                distances.append((trainingSet[x], dist))
        distances.sort(key=operator.itemgetter(1))
        neighbors = []
        for x in range(k):
                neighbors.append(distances[x][0])
        return neighbors

def getResponse(neighbors):
        classVotes = {}
        for x in range(len(neighbors)):
                response = neighbors[x][-1]
                if response in classVotes:
                        classVotes[response] += 1
                else:
                        classVotes[response] = 1
        sortedVotes = sorted(classVotes.items(), key=operator.itemgetter(1), reverse=True)
        return sortedVotes[0][0]

def getAccuracy(testSet, predictions):
        correct = 0
        for x in range(len(testSet)):
                if testSet[x][-1] == predictions[x]:
                        correct += 1
        return (correct/float(len(testSet))) * 100.0

def main():
        # prepare data
        trainingSet=[]
        testSet=[]
```

```
    split = 0.67
    loadDataset('KNN-input.csv', split, trainingSet, testSet)
    print ('\n Number of Training data: ' + (repr(len(trainingSet))))
    print (' Number of Test Data: ' + (repr(len(testSet))))
    # generate predictions
    predictions=[]
    k = 3
    print('\n The predictions are: ')
    for x in range(len(testSet)):
        neighbors = getNeighbors(trainingSet, testSet[x], k)
        result = getResponse(neighbors)
        predictions.append(result)
        print(' predicted=' + repr(result) + ', actual=' + repr(testSet[x][-1]))
    accuracy = getAccuracy(testSet, predictions)
    print('\n The Accuracy is: ' + repr(accuracy) + '%')

main()
```