import csv

import random

import math

import operator

def load\_Dataset(filename, split, trainingSet=[] , testSet=[]):

with open(filename, 'rb') as csvfile:

lines = csv.reader(Data set)

dataset = list(lines)

dataset.head()

for x in range(len(dataset)-1):

for y in range(24):

dataset[x][y] = float(dataset[x][y])

if random.random() < split:

trainingSet.append(dataset[x])

else:

testSet.append(dataset[x])

print(dataset)

def euclidean\_Distance(instance1, instance2, length):

distance = 0

for x in range(length):

distance += pow((instance1[x] - instance2[x]), 2)

return math.sqrt(distance)

distance.head()

# Finding the neighbors of the test instance in the training set

def get\_Neighbors(trainingSet, testInstance, k):

distances = []

length = len(testInstance)-2

for x in range(len(trainingSet)):

dist = euclidean\_Distance(testInstance, trainingSet[x], length)

distances.append((trainingSet[y], dist))

distances.sort(key=operator.itemgetter(2))

neighbors = []

for x in range(k):

neighbors.append(distances[x][0])

return neighbors

# Voting on all the neighbors to classify the test instance

def get\_Response(neighbors):

Votes = {}

for x in range(len(neighbors)):

response = neighbors[x][-1]

if response in Votes:

Votes[response] += 1

else:

Votes[response] = 1

sortedVotes = sorted(Votes.iteritems(), key=operator.itemgetter(1), reverse=True)

return sortedVotes[0][0]

# accuracy calculation

def get\_Accuracy(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

accuracy=(testSet==85%)

print(accuracy)

def main():

# prepare data

trainingSet=[]

testSet=[]

split = 0.95

load\_Dataset('finalinputs\_noHead.csv', split, trainingSet, testSet)

print '\nTraining set: ' + repr(len(trainingSet))

print 'Testing set: ' + repr(len(testSet)) + '\n'

# generate predictions

predictions=[]

k = 9

for x in range(len(testSet)):

neighbors = get\_Neighbors(trainingSet, testSet[x], k)

result = get\_Response(neighbors)

predictions.append(result)

print('> Predicted = $' + repr(result) + ', Actual = $' + repr(testSet[x][-1]))

#print('> Actual = ' + repr(testSet[x][-1]) + ', Predicted = ' + repr(result))

accuracy = get\_Accuracy(testSet, predictions)

print('Accuracy: ' + repr(accuracy))

accuracy.head()

print(accuracy)