import random

import math

import operator

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 getNeighbors(trainingSet,testInstance,k):

distances=[]

length=4

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 convert\_str\_to\_float(x,n):

for i in range(n):

x[i] = float(x[i])

return x

def load\_split\_data(split,training\_data = [],test\_data = []):

for line in open("iris.txt",'r'):

temp = convert\_str\_to\_float(line[0:-1].split(','),4)

if random.random()<=split:

training\_data.append(temp)

else:

test\_data.append(temp)

return(training\_data,test\_data)

def euclideanDistance(instance1,instance2,length):

distance = 0

for x in range(length):

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

return math.sqrt(distance)

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():

trainingSet=[]

testSet=[]

split=0.70

trainingSet, testSet = load\_split\_data(split)

print('Training Set:'+str(len(trainingSet)))

print('Test Set:'+str(len(testSet)))

#generate predictions

predictions=[]

k=3

for x in range(len(testSet)):

neighbors=getNeighbors(trainingSet,testSet[x],k)

result = getResponse(neighbors)

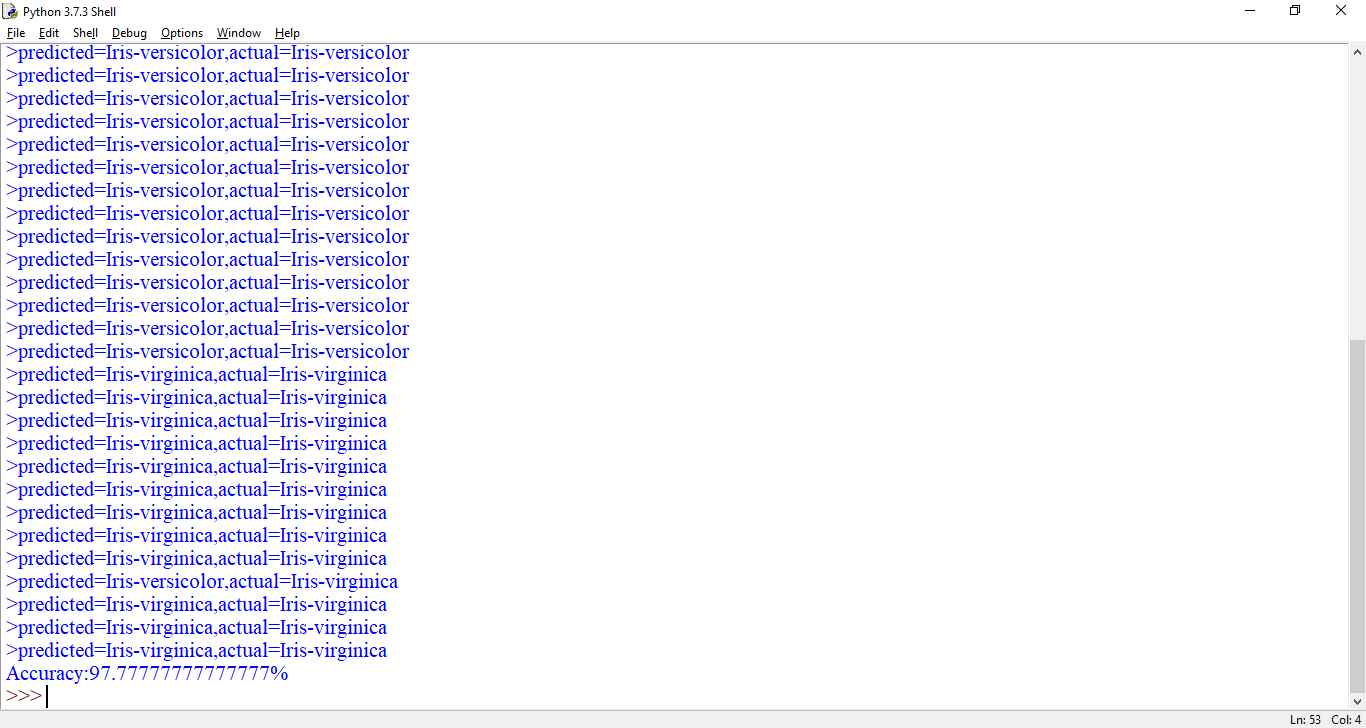
predictions.append(result)

print('>predicted='+str(result)+',actual='+str(testSet[x][-1]))

accuracy=getAccuracy(testSet,predictions)

print('Accuracy:'+str(accuracy)+'%')

main()



import random

import math

import operator

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 getNeighbors(trainingSet,testInstance,k):

distances=[]

length=4

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 convert\_str\_to\_float(x,n):

for i in range(n):

x[i] = float(x[i])

return x

def load\_split\_data(split,training\_data = [],test\_data = []):

for line in open("iris.txt",'r'):

temp = convert\_str\_to\_float(line[0:-1].split(','),4)

if random.random()<=split:

training\_data.append(temp)

else:

test\_data.append(temp)

return(training\_data,test\_data)

def euclideanDistance(instance1,instance2,length):

distance = 0

for x in range(length):

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

return math.sqrt(distance)

def main():

trainingSet=[]

testSet=[]

split=1

trainingSet, testSet = load\_split\_data(split)

sl=float(input("Enter sepal length:"))

sw=float(input("Enter sepal width:"))

pl=float(input("Enter petal length:"))

pw=float(input("Enter petal width:"))

testSet=[sl,sw,pl,pw,'']

k=3

neighbors=getNeighbors(trainingSet,testSet,k)

result = getResponse(neighbors)

print('>predicted='+str(result))

main()

