SHIVAM KHARE
201505547
ASSIGNMENT-1

1. Iris Data Set

Data Set Information:
This is perhaps the best known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. (See Duda & Hart, for example.) The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

Predicted attribute: class of iris plant.

This is an exceedingly simple domain.

This data differs from the data presented in Fishers article (identified by Steve Chadwick, spchadwick '@' espeedaz.net). The 35th sample should be: 4.9,3.1,1.5,0.2,"Iris-setosa" where the error is in the fourth feature. The 38th sample: 4.9,3.6,1.4,0.1,"Iris-setosa" where the errors are in the second and third features.

Attribute Information:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class:
-- Iris Setosa
-- Iris Versicolour
-- Iris Virginica

Data Set Characteristics:	Multivariate	**Number of Instances:**	150	**Area:**	Life
Attribute Characteristics:	Real	**Number of Attributes:**	4	**Date Donated**	1988-07-01
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	862043

DISTANCE FUNCTION:-(euclidean distance)

```
for i in range(trainigsetlength):
distance=0
for j in range(length):
    distance=distance+pow((testset[x][j]-trainigset[i][j]),2)
distance=math.sqrt(distance)
```

2.Balance Scale Data Set

Data Set Information:

This data set was generated to model psychological experimental results. Each example is classified as having the balance scale tip to the right, tip to the left, or be balanced. The attributes are the left weight, the left distance, the right weight, and the right distance. The correct way to find the class is the greater of (left-distance * left-weight) and (right-distance * right-weight). If they are equal, it is balanced.

Attribute Information:

1. Class Name: 3 (L, B, R)
2. Left-Weight: 5 (1, 2, 3, 4, 5)
3. Left-Distance: 5 (1, 2, 3, 4, 5)
4. Right-Weight: 5 (1, 2, 3, 4, 5)
5. Right-Distance: 5 (1, 2, 3, 4, 5)

Data Set Characteristics:	Multivariate	**Number of Instances:**	625	**Area:**	Social
Attribute Characteristics:	Categorical	**Number of Attributes:**	4	**Date Donated**	1994-04-22
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	103064

DISTANCE FUNCTION:-(euclidean distance)

```
for i in range(trainigsetlength):
distance=0
for j in range(length):
```

```
        distance=distance+pow((testset[x][j]-trainigset[i][j]),2)
    distance=math.sqrt(distance)
```

3.Haberman's Survival Data Set

Data Set Information:

The dataset contains cases from a study that was conducted between 1958 and 1970 at the University of Chicago's Billings Hospital on the survival of patients who had undergone surgery for breast cancer.

Attribute Information:

1. Age of patient at time of operation (numerical)
2. Patient's year of operation (year - 1900, numerical)
3. Number of positive axillary nodes detected (numerical)
4. Survival status (class attribute)
-- 1 = the patient survived 5 years or longer
-- 2 = the patient died within 5 year

Data Set Characteristics:	Multivariate	**Number of Instances:**	306	**Area:**	Life
Attribute Characteristics:	Integer	**Number of Attributes:**	3	**Date Donated**	1999-03-04
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	84419

DISTANCE FUNCTION:-(euclidean distance)

```
    for i in range(trainigsetlength):
    distance=0
    for j in range(length):
        distance=distance+pow((testset[x][j]-trainigset[i][j]),2)
    distance=math.sqrt(distance)
```

1. Iris Data Set

Random subsampling approach

confusion matrix for k=1

	class1	class2	class3
class1	29	0	0
class2	0	21	2
class3	0	0	19

Accuracy for k=1: **97.183099 %**

confusion matrix for k=3

	class1	class2	class3
class1	29	0	0
class2	0	21	2
class3	0	0	19

Accuracy for k=3: **97.183099 %**

5-fold cross validation

confusion matrix for k=1

	class1	class2	class3
class1	42	0	0
class2	0	38	0
class3	0	7	33

confusion matrix for k=3

	class1	class2	class3
class1	42	0	0
class2	0	37	1
class3	0	11	29

iteration 1:
mean Accuracy for k=1: 95.0
mean Accuracy for k=3: 95.1666666667
Standard deviation for k=1: 1.05409255339
Standard deviation for k=3: 1.91485421551

iteration 2:
mean Accuracy for k=1: 95.0
mean Accuracy for k=3: 93.3333333333
Standard deviation for k=1: 0.7453559925
Standard deviation for k=3: 0.986013297183

iteration 3:
mean Accuracy for k=1: 93.5
mean Accuracy for k=3: 95.1666666667
Standard deviation for k=1: 2.84800124844
Standard deviation for k=3: 1.14260910007

iteration 4:
mean Accuracy for k=1: 92.8333333333
mean Accuracy for k=3: 95.0
Standard deviation for k=1: 1.77169096879
Standard deviation for k=3: 1.78730088246

iteration 5:
mean Accuracy for k=1: 95.8333333333
mean Accuracy for k=3: 95.8333333333
Standard deviation for k=1: 0.986013297183
Standard deviation for k=3: 0.7453559925

iteration 6:
mean Accuracy for k=1: 95.5
mean Accuracy for k=3: 95.8333333333
Standard deviation for k=1: 0.707106781187
Standard deviation for k=3: 0.7453559925

iteration 7:
mean Accuracy for k=1: 94.6666666667
mean Accuracy for k=3: 93.8333333333
Standard deviation for k=1: 1.42400062422
Standard deviation for k=3: 2.0275875101

iteration 8:
mean Accuracy for k=1: 94.8333333333
mean Accuracy for k=3: 93.3333333333
Standard deviation for k=1: 0.235702260396
Standard deviation for k=3: 2.0069324298

iteration 9:
mean Accuracy for k=1: 94.1666666667
mean Accuracy for k=3: 94.3333333333
Standard deviation for k=1: 1.97202659437
Standard deviation for k=3: 2.18581284143

iteration 10:
mean Accuracy for k=1: 95.8333333333
mean Accuracy for k=3: 95.0
Standard deviation for k=1: 1.23603308118
Standard deviation for k=3: 1.53659074288

Grand Mean for k=1: 94.7166666667
Grand Mean for k=3: 94.6833333333

===
==

2. Balance Scale Data Set

<u>Random subsampling approach</u>

confusion matrix for k=1

	class1	class2
class1	85	23
class2	33	10

Accuracy for k=1: **62.913907**

confusion matrix for k=3

	class1	class2
class1	99	9
class2	30	13

Accuracy for k=3: **74.172185**

5-fold cross validation

confusion matrix for k=1

146	37
45	17

confusion matrix for k=3

	class1	class2
class1	157	26
class2	46	16

Iteration: 1
mean Accuracy for k=1: 65.6871863499
mean Accuracy for k=3: 70.0180662429
Standard deviation for k=1: 1.15515510447
Standard deviation for k=3: 1.00189157016

Iteration: 2
mean Accuracy for k=1: 66.5005018401
mean Accuracy for k=3: 69.3629976581
Standard deviation for k=1: 3.55830938769
Standard deviation for k=3: 2.91905807437

Iteration: 3
mean Accuracy for k=1: 66.0946804951
mean Accuracy for k=3: 68.2154566745
Standard deviation for k=1: 2.4696156724
Standard deviation for k=3: 2.23657617157

Iteration: 4
mean Accuracy for k=1: 65.6885245902
mean Accuracy for k=3: 71.8116426899
Standard deviation for k=1: 2.04758728473

Standard deviation for k=3: 1.68173225934

Iteration: 5
mean Accuracy for k=1: 64.6239544998
mean Accuracy for k=3: 68.5453328873
Standard deviation for k=1: 1.8416575861
Standard deviation for k=3: 0.691726515565

Iteration: 6
mean Accuracy for k=1: 66.8287052526
mean Accuracy for k=3: 72.218802275
Standard deviation for k=1: 1.01257915071
Standard deviation for k=3: 1.85467517075

Iteration: 7
mean Accuracy for k=1: 66.0103713617
mean Accuracy for k=3: 71.7296754767
Standard deviation for k=1: 1.62615700847
Standard deviation for k=3: 1.67850749394

Iteration: 8
mean Accuracy for k=1: 67.237537638
mean Accuracy for k=3: 72.3007694881
Standard deviation for k=1: 2.69757781501
Standard deviation for k=3: 3.01236166695

Iteration: 9
mean Accuracy for k=1: 67.4827701572
mean Accuracy for k=3: 72.469053195
Standard deviation for k=1: 1.43092207079
Standard deviation for k=3: 0.880227437256

Iteration: 10
mean Accuracy for k=1: 66.8323854132
mean Accuracy for k=3: 71.2415523586
Standard deviation for k=1: 1.51903852433
Standard deviation for k=3: 1.38785664392

Grand Mean for k=1: 66.2986617598
Grand Mean for k=3: 70.7913348946

===
===

3.Haberman's Survival Data Set

<u>Random subsampling approach</u>

confusion matrix for k=1

	class1	class2	class3
class1	107	27	16
class2	3	2	16

class3	6	6	128

Accuracy for k=1: **76.205788** %

confusion matrix for k=3

	class1	class2	class3
class1	115	16	19
class2	4	2	15
class3	4	7	129

Accuracy for k=3: **79.099678 %**

<u>5-fold cross validation</u>

confusion matrix for k=1

	class1	class2	class3
class1	193	18	28
class2	14	2	19
class3	8	23	195

confusion matrix for k=3

	class1	class2	class3
class1	195	17	27
class2	6	5	24
class3	2	18	206

Iteration: 1
mean Accuracy for k=1: 79.2
mean Accuracy for k=3: 82.24
Standard deviation for k=1: 0.779743547585
Standard deviation for k=3: 0.960832971957

Iteration: 2
mean Accuracy for k=1: 76.84
mean Accuracy for k=3: 81.64
Standard deviation for k=1: 0.712179752591
Standard deviation for k=3: 1.36937942149

Iteration: 3
mean Accuracy for k=1: 77.88
mean Accuracy for k=3: 81.44
Standard deviation for k=1: 1.32544332206
Standard deviation for k=3: 0.739729680356

Iteration: 4
mean Accuracy for k=1: 78.64
mean Accuracy for k=3: 81.56
Standard deviation for k=1: 1.14332847424
Standard deviation for k=3: 0.99357938787

Iteration: 5

mean Accuracy for k=1: 79.16
mean Accuracy for k=3: 81.04
Standard deviation for k=1: 0.393954312072
Standard deviation for k=3: 1.20797350964

Iteration: 6
mean Accuracy for k=1: 76.2
mean Accuracy for k=3: 80.16
Standard deviation for k=1: 1.54660919433
Standard deviation for k=3: 1.03305372561

Iteration: 7
mean Accuracy for k=1: 79.16
mean Accuracy for k=3: 82.44
Standard deviation for k=1: 0.712179752591
Standard deviation for k=3: 0.913892772704

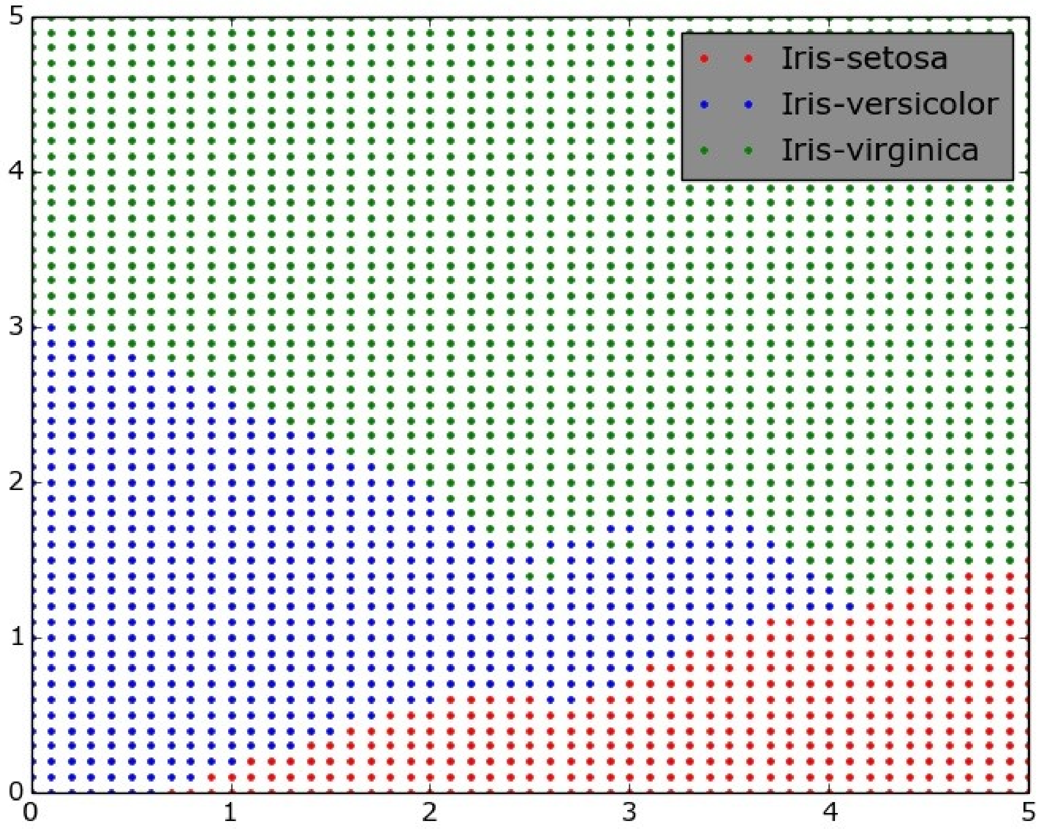
Iteration: 8
mean Accuracy for k=1: 77.68
mean Accuracy for k=3: 81.96
Standard deviation for k=1: 0.678822509939
Standard deviation for k=3: 1.15723809132

Iteration: 9
mean Accuracy for k=1: 77.92
mean Accuracy for k=3: 83.0
Standard deviation for k=1: 0.932094415818
Standard deviation for k=3: 0.6

Iteration: 10
mean Accuracy for k=1: 77.72
mean Accuracy for k=3: 82.48
Standard deviation for k=1: 1.39025177576
Standard deviation for k=3: 0.822678552048

Grand Mean for k=1: 78.04
Grand Mean for k=3: 81.796

Answer: 3



Answer4
yes
The decision boundary of 3-NN will be piece-wise linear. Reason: In 3-NN the decision boundary is formed by considering the perpendicular bisectors of the imaginary line formed by connecting the transition-points. Eah of these perpendicular bisectors are linear. Hene we can conlude that the deision boundary of 3-NN is piece-wise linear

Graph plot code:

```
#!/usr/bin/python
import csv
import matplotlib.pyplot as plt
import math
import operator
from numpy import arange

l1=[]
i=0
while(i<4):
	j=0
	while(j<4):
		l1.append((i,j))
		j+=.1
	i=i+0.1
print len(l1)

f= open('iris.data','rb')
xcord=[]
ycord=[]
lines=csv.reader(f)
data=list(lines)
for i in range(len(data)):
	for j in range(4):
		data[i][j]=float(data[i][j])
		if(j==1):
			xcord.append(data[i][j])
		if(j==3):
			ycord.append(data[i][j])


neighbour=[]
for i in range(len(l1)):
	distancelist=[]
	dist1=0
	dist2=0
	for j in range(len(data)):
		dist1=pow((xcord[j]-l1[i][0]),2)
		dist2=pow((ycord[j]-l1[i][1]),2)
		dist3=math.sqrt(dist1+dist2)
		distancelist.append((data[j][-1],dist3))
	distancelist.sort(key=operator.itemgetter(1))
	#print distancelist
	neighbour.append(distancelist[0][0])
l2=[]
l3=[]
l4=[]
l5=[]
l6=[]
l7=[]
```

```
for i in range(len(l1)):
	if(neighbour[i]=='Iris-setosa'):
		l2.append(l1[i][0])
		l3.append(l1[i][1])
	elif(neighbour[i]=='Iris-versicolor'):
		l4.append(l1[i][0])
		l5.append(l1[i][1])
	else:
		l6.append(l1[i][0])
		l7.append(l1[i][1])
count=0
lx=[]
ly=[]
for i in arange(0,4,.1):
	for j in arange(0,4,.1):
		count+=1
		if(neighbour[count-1]!=neighbour[count]):
			lx.append(i)
			ly.append(j)
		break
lx1=[]
ly1=[]
count1=0
for i in arange(0,4,.1):
	count1+=400
	for j in arange(4,0,.1):
		count-=1;
		if(neighbour[count1+1]!=neighbour[count1]):
			lx1.append(i)
			ly1.append(j)
		break

plt.plot(l2,l3,'.r',label='Iris-setosa')
plt.plot(l4,l5,'.b',label='Iris-versicolor')
plt.plot(l6,l7,'.g',label='Iris-virginica')
legend=plt.legend(loc='upper right')
frame=legend.get_frame()
frame.set_facecolor('.55')
plt.show()

l8=[]
l9=[]
l10=[]
l11=[]
l12=[]
l13=[]
for i in range(len(data)):
	if(data[i][-1]=='Iris-setosa'):
		l8.append(data[i][1])
		l9.append(data[i][3])
	elif(data[i][-1]=='Iris-versicolor'):
		l10.append(data[i][1])
```

```
            l11.append(data[i][3])
        else:
            l12.append(data[i][1])
            l13.append(data[i][3])

plt.plot(l8,l9,'.r',label='Iris-setosa')
plt.plot(l10,l11,'.b',label='Iris-versicolor')
plt.plot(l12,l13,'.g',label='Iris-virginica')
plt.plot(lx,ly,'.k')
plt.plot(lx1,ly1,'.k')
plt.show()
```