Name: Kunal Nand Khaware

Roll No: 201450828

CSE471: Statistical Methods in AI -- Spring 2015

Assignment 1: ONE-NEAREST NEIGHBOUR CLASSIFIER

Answer 1

1. Iris Plants Database:

a. Number of instances: 150

b. Number of Attributes: 4 numeric, predictive attributes and class

c. Attribute Information:

1. sepal length in cm

2. sepal width in cm

3. petal length in cm

4. petal width in cm

5. class:

6. Iris Setosa, Iris Versicolour, Iris Virginica

Distance Function Used : Euclidean distance

Answer 2

DataSet :Iris Plants Database

1.Confusion Matrix:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Iris Setosa | Iris Versicolour | Iris Virginica |
| Iris Setosa | 22 | 0 | 0 |
| Iris Versicolour | 0 | 15 | 1 |
| Iris Virginica | 0 | 0 | 17 |

Accuracy 98.1818181818

2.Confusion Matrix:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Iris Setosa | Iris Versicolour | Iris Virginica |
| Iris Setosa | 14 | 0 | 0 |
| Iris Versicolour | 0 | 19 | 4 |
| Iris Virginica | 0 | 2 | 15 |

Accuracy 88.8888888889

3.Confusion Matrix:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Iris Setosa | Iris Versicolour | Iris Virginica |
| Iris Setosa | 15 | 0 | 0 |
| Iris Versicolour | 0 | 21 | 3 |
| Iris Virginica | 0 | 0 | 16 |

Accuracy 94.5454545455

4.Confusion Matrix:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Iris Setosa | Iris Versicolour | Iris Virginica |
| Iris Setosa | 14 | 0 | 0 |
| Iris Versicolour | 0 | 15 | 1 |
| Iris Virginica | 0 | 0 | 14 |

Accuracy 97.7272727273

5.Confusion Matrix:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Iris Setosa | Iris Versicolour | Iris Virginica |
| Iris Setosa | 14 | 0 | 0 |
| Iris Versicolour | 0 | 14 | 1 |
| Iris Virginica | 0 | 0 | 16 |

Accuracy 95.652173913

6.Confusion Matrix:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Iris Setosa | Iris Versicolour | Iris Virginica |
| Iris Setosa | 24 | 0 | 0 |
| Iris Versicolour | 0 | 17 | 2 |
| Iris Virginica | 0 | 2 | 14 |

Accuracy 93.2203389831

7. Confusion Matrix:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Iris Setosa | Iris Versicolour | Iris Virginica |
| Iris Setosa | 15 | 0 | 0 |
| Iris Versicolour | 0 | 16 | 2 |
| Iris Virginica | 0 | 1 | 15 |

Accuracy 93.877551020

8. Confusion Matrix:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Iris Setosa | Iris Versicolour | Iris Virginica |
| Iris Setosa | 18 | 0 | 0 |
| Iris Versicolour | 0 | 14 | 3 |
| Iris Virginica | 0 | 0 | 16 |

Accuracy 92.3076923077

9. Confusion Matrix:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Iris Setosa | Iris Versicolour | Iris Virginica |
| Iris Setosa | 13 | 0 | 0 |
| Iris Versicolour | 0 | 15 | 3 |
| Iris Virginica | 0 | 2 | 18 |

Accuracy 90.196078431

10. Confusion Matrix:

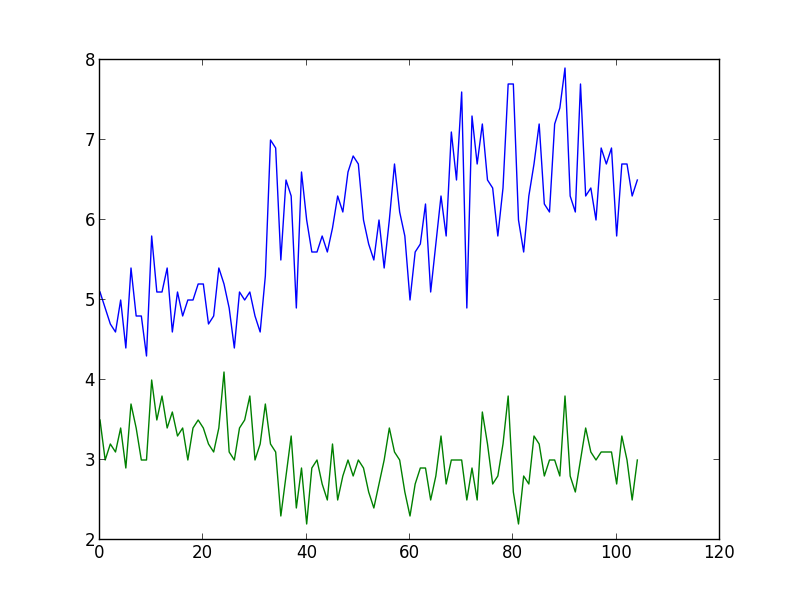
|  |  |  |  |
| --- | --- | --- | --- |
|  | Iris Setosa | Iris Versicolour | Iris Virginica |
| Iris Setosa | 17 | 0 | 0 |
| Iris Versicolour | 0 | 22 | 3 |
| Iris Virginica | 0 | 0 | 17 |

Accuracy 94.9152542373

Mean = 93.946

Standard Deviation=2.81

Answer 3



Source Code

import random as rnd

import math

import operator

import matplotlib.pyplot as plt

def loadDataFile(filename,training,test):

f = open(filename,'r');

for line in f:

if not line.isspace():

data=list(line.strip().split(","))

if rnd.random()<=0.66:

training.append(data)

else:

test.append(data)

def euclideanDistance(data1, data2, length):

distance = 0

for x in range(length):

distance += pow(float(data1[x]) - float(data2[x]), 2)

return math.sqrt(distance)

def getNeighbors(trainingSet, testInstance, k):

distances = []

length = len(testInstance)-2

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 getAccurancy(trainingsataset,testdataset):

count=0

for data in testdataset:

neighbour=getNeighbors(trainingsataset,data,1)

if neighbour[0][-1] == data[-1]:#Since 1-NN so only one element will be in neighbour list

count=count+1

accuracy=count/float(len(testdataset))

return accuracy\*100

def calculateMean(data):

sum=0

for num in data:

sum+=num

average=sum/float(len(data))

return average

def calculateStandarDeviation(data,mean):

sum=0

for num in data:

sum+=math.pow((num-mean),2)

varience=sum/float(len(data))

return math.sqrt(varience)

def populateConfusionMatrix(training,testdata,classifiers):

prediction=list()

actualdict=dict()

predictiondict=dict()

falsepredlist=list();

for clss in classifiers:

actualdict.update({clss:0})

for data in testdata:

neighbour=getNeighbors(training,data,1)

actualdict.update({data[-1]:actualdict.get(data[-1])+1})

prediction.append((data[-1],neighbour[0][-1]))

# print actualdict

#print prediction

for labels in classifiers:

predictiondict.update({labels:0})

for val in prediction:

if(val[0]==val[1]):

predictiondict.update({val[0]:predictiondict.get(val[0])+1})

else:

falsepredlist.append(val)

print "Prediction ",predictiondict

drawMatrix(classifiers,predictiondict,actualdict,falsepredlist)

def getMatrixIndexForFalsePrediction(confusionmatrix,falselist):

#print " FalseList",falselist[0],falselist[1]

index=list()

xindex=0

yindex=1

for x in confusionmatrix[0]:

if x==falselist[0]:

index.append(xindex)

xindex+=1

for y in range(1,len(confusionmatrix[0])):

if confusionmatrix[y][0]==falselist[1]:

index.append(yindex)

yindex+=1

return index

def drawMatrix(classifiers,predict,actualdict,falsepredlist):

classifiers.insert(0,'')

confusion\_matrix=[classifiers]

for cls in classifiers:

if cls:

confusion\_matrix.append([cls])

for i in range(1,len(classifiers)):

for j in range(1,len(classifiers)):

confusion\_matrix[i].append(0)

#print confusion\_matrix

for i in range(1,len(classifiers)):

confusion\_matrix[i][i]=predict.get(confusion\_matrix[i][0])

print "Flase PredList ",falsepredlist

for values in falsepredlist:

falseindex=getMatrixIndexForFalsePrediction(confusion\_matrix,values)

#print "False Index",falseindex

confusion\_matrix[falseindex[0]][falseindex[1]]=confusion\_matrix[falseindex[0]][falseindex[1]]+1

print "Confusion Matrix "

for rows in confusion\_matrix:

print rows

def plotGraph(datalist):

plt.plot(datalist)

plt.show()

accuracyList=list()

for i in range(9):

filename="iris.data"

classNamesList=['Iris-setosa','Iris-versicolor','Iris-virginica']

training=list()

test=list()

loadDataFile(filename,training,test)

accuracy=(getAccurancy(training,test))

accuracyList.append(accuracy)

print "Accuracy ",accuracy

populateConfusionMatrix(training,test,classNamesList)

mean=calculateMean(accuracyList)

print "Mean ",mean

standard\_deviation=calculateStandarDeviation(accuracyList,mean)

print "Standard Deviation ",standard\_deviation

coordinatelist=list()

for elem in training:

coordinatelist.append((elem[0],elem[1]))

plotGraph(coordinatelist)

Answer 4:

Decision boundary of 3NN classification will be piecewise linear. The reason behind it is the more strict classification of the data and thus for better accuracy a single equation of line will not serve the purpose. A single line will not be able to draw proper classification and thus if multiple single lines wil be connected the whole boundary will look like piecewise linear.