Kalpish Singhal
M.Tech - CSE
Roll No - 201505513

STATISTICAL METHODS IN AI

Assignment 1:

K-NEAREST NEIGHBOUR CLASSIFIER

20th January 2016

Q1.

Please list the names and salient characteristics (Number of features, Number of instances, Number of Classes, etc) of the datasets you chose from the UCI ML repository for your experiments. Mention any criteria you used in deciding on the datasets and the distance function used for each dataset.

Sol:

Date set names :-

1. **IRIS data set**
2. **Wine data set**
3. **WISCONSIN DATA SET**

- **Iris Dataset Characteristics:-**
    - Number of Attributes: 4 numeric, predictive attributes and the class.
    - 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
    - Missing Attribute Values: None
    - Number of Instances: 150 (50 in each of three classes)

- Class Distribution: 33.3% for each of 3 classes.
- LINK:-http://archive.ics.uci.edu/ml/datasets/Iris

- **Wine Dataset Characteristics:-**
  - Wine recognition data
  - Number of Attributes :13
  - For Each Attribute: All attributes are continuous No statistics available, but suggest to standardize variables for certain uses (e.g. for us with classifiers which are NOT scale invariant)
  - Missing Attribute Values:None
  - Number of class :3
  - Class Distribution: number of instances per class c
    - class 1 -> 59
    - class 2 -> 71
    - class 3 -> 48
  - Link:-http://archive.ics.uci.edu/ml/datasets/Wine

- **Breast Cancer Wiscons Dataset Characteristics:-**
  - Number of Instances: (Number of Instances: 699 (as of 15 July 1992))
    - Benign: 458 (65.5%)
    - Malignant: 241 (34.5%)
  - Number of Attributes: Number of Attributes: 10 plus the class attribute
  - Attribute Information:
    - Sample code number: id number
    - Clump Thickness

        - Uniformity of Cell Size
        - Uniformity of Cell Shape
        - Marginal Adhesion
        - Single Epithelial Cell Size
        - Bare Nuclei
        - Bland Chromatin
        - Normal Nucleoli
        - Mitoses
        - Class: (2 for benign, 4 for malignant)
    - Missing Attribute Values: 16 (The '?' in the missing attributes are replaced by the average value)

The above data samples are taken with respect to considering following general idea:

-> The attributes are numerical values.

-> Distance between the two classes can be calculated by using the Eulidean Distance only.

-> A commonly used distance metric for continuous variables is Euclidean distance.

-> The first two data sets i.e iris.data and wine.data don't contain any missing values and hence euclidean distance is applied directly on the continous numerical values.

-> The third class consist of missing values i.e. breast cancer data set. The data set with missing values is taken so as to analyse the behaviour of data set with missing value classified with knn classifier.

Q2.

For each of the datasets, give the results of classification using the (1- and 3-) nearest neighbor classifiers (mean and variance of accuracy and the confusion matrix). Give your observations related to the results.

Sol:

Knn Algorithm:-

Lazy Learning Algorithm Defer the decision to generalize beyond the training examples till a new query is encountered . Whenever we have a new point to classify, classify, we find its K nearest neighbors from the training data.

- For each training example, add the example to the list of training_examples
- Given a query instance xq ¨ Given a query instance x to be classified
    - Let x1 ,x2....xk denote the k instances from training_examples that are nearest to xq.
    - Return the class that represents the maximum of the k instances.

Experimental Set-Up :

1. data Handling:-

load the data from the file. A list of list reads the dataset and divides it into training sample and test sample as per the proper split ratio. (split=0.5 for random subsampling and 0.2 for 5 fold) numerical values in data are converted to float and stored in the training and test list respectively.

2. Distance b/w the test instance and training set :-

To make the prediction of the class name, the data belongs to, we calculate the similarity i.e distances between any two given data instances to locate the k most nearest data instances I.e(k=1 or 3 here), and hence can make the prediction. All attributes are numeric and have the same units, we can directly use the Euclidean distance measure to compute the distance. Additionally, we use only numerical fields to include to calculate the distance. We limit the calculation of Euclidean distance by not using the attribute containing class label.

3. Finding k-nearest neighbors and handling ties

We need to use the distances obtained to collect the k most similar instances for a given test instance. It is done by sorting the distance list, on the distance value, and hence selecting the first k values irrespective of the ties.for handling ties FCFS is used I.e Select the first k classes with minimum value after shuffling and comparing.

4. Prediction:

Once we have located the k most similar neighbors for a test instance Prediction of the class name has to be done. For this we make a dictionary of the classes, and hence find the majority vote class, with the max key value in the dictionary.

5. Accuracy

calculate accuracy by comparing the predictions list of the test sample and the actual class list of the test sample. Accuracy will be the ratio of the total correct predictions out of all the predictions made.

6. Confusion Matrix

The confusion matrix of each dataset depicts the classes that are most confused. It is made by comparing the predicted outputs and actual classes. It is made dynamic, by first identifying which classes are present i.e. by forming a dictionary.

Now, the matrix is formed and data is populated in the matrix by comparing the two lists. Finally the data is printed in form of confusion matrix using tabulate library in grid form.

7. Mean & Standard Deviation

The accuracies of the 10 iterations are recorded and hence used to compute the final mean and standard deviation of the accuracies so obtained.

CODE:-

```
import csv
import random
from operator import itemgetter
from math import *

def euclidean_dist(var2, var1,index_class,dimension):
    for x in range(0,dimension):
        if x==0:
            distance=0
        if x==index_class:
            continue
        if var1[x]=='?':
            var1[x]='5'
```

```
        if var2[x]=='?':
            var2[x]='5'
        var1[x]=float(var1[x])
        var2[x]=float(var2[x])
        if var1[x]>100000:
            continue
        temp_power=    pow((var1[x]) - (var2[x]), 2)
        distance = distance + temp_power
    result=sqrt(distance)
    return result

def find_knn(training, test_inst, k,index_class,length):
    distances=[]
    neighbours = []
    for i in range(length):
        result_d=euclidean_dist(training[i],test_inst,index_class,len(test_inst))
        distances.append((training[i], result_d))
    distances.sort(key=itemgetter(1))
    for i in range(0,k):
        neighbours.append(distances[i][0])
    return neighbours
    pass


def predict(index_class,neighbours,len_neighbour):
    class_dict = {}
    for i in range(0,len_neighbour):
        pred_clss = neighbours[i][index_class]
        if pred_clss not in class_dict:
            class_dict[pred_clss]=1
        else:
            class_dict[pred_clss]=class_dict[pred_clss]+1
    temp_sort=sorted(class_dict.iteritems(), key=itemgetter(1), reverse=True)
    class_dict = {}
    return temp_sort[0][0]
```

```
def get_accuracy(test, predictions, index_class):
    correct=0
    for i in range(len(test)):
        temp = predictions[i]
        if test[i][index_class] != temp:
            continue
        elif test[i][index_class] == predictions[i]:
            correct+=1
    accuracy_percentage = (correct/(float(len(test)))) * 100
    return accuracy_percentage

def mean_standard_deviation(accur_per_list):
    sum=0
    diff_sq = 0
    mean_sd=[]
    length = len(accur_per_list)
    for i in range(length):
        sum+=float(accur_per_list[i])

    mean = sum/length
    mean_sd.append(mean)
    for i in range(length):
        diff_sq+= pow((accur_per_list[i] - mean_sd[0]), 2)

    diff_sq = diff_sq/len(accur_per_list)
    sd= sqrt(diff_sq)
    mean_sd.append(sd)
    return mean_sd

def print_confusion_matrix(predictions, actual_classes):
    #Fetching the name of the classes to dictionary and then to the list
    classes={}
    class_list =[]
    for i in range(len(actual_classes)):
        if actual_classes[i] in classes:
            classes[actual_classes[i]]= 1
        else:
```

```
        classes[actual_classes[i]]= 1

for i in classes.keys():
    class_list.append(i)

#Creating confusion matrix as list -> empty list and hence comparing and increasing the count
confusion_matrix=[]
count = 0
for i in range(len(class_list)):
    for j in range(len(class_list)):
        confusion_matrix.append(0)
temp = count
for i in range(len(actual_classes)):
    for j in range(len(class_list)):
        for k in range(len(class_list)):
            temp1 = class_list[j]
            temp2 = class_list[k]
            if actual_classes[i] == temp1 and predictions[i] == temp2:
                count+=1
                index1 = j*len(class_list)+k
                confusion_matrix[index1]+=1

#Printing confusion matrix
index = 0
clas_indx=0
for i in range(len(class_list)+1):
    for j in range(len(class_list)+1):
        if i==j==0:
            print '{0:15}'.format(' ') ,
        elif i == 0:
            print '{0:15}'.format(class_list[j-1]) ,
        elif j==0:
            print '{0:15}'.format(class_list[i-1]) ,
        else:
            print '{0:15}'.format(confusion_matrix[index]),
            index+=1
    print '\n'
```

```
def main():
    datasetname = raw_input('Enter the dataset filename eg. iris.data:')
    index_class = input('Index of the class in dataset: ')
    k = input('Enter the value of k for knn: ')
    print 'Split Ratio : 0.50 - Random SubSampling'
    print 'Split Ratio : 0.20 - Five Fold Cross Validation'
    split=input('Enter the split ratio : ')
    mean_record_of_iterations=[]

    if split !=.5:
        if split !=.2:
            print "invalid input taking random split .5"

    if split==0.5:
        accur_per_list=[]
        for no_of_iterations in range(0,10):
            test_class = []
            test_ver=[]
            with open(datasetname,'rb') as file_dataset:
                lines=csv.reader(file_dataset)
                listing=list(lines)
                training=[]
                test=[]
                random.shuffle(listing)
                for x in range(0, len(listing), 2):
                    training.append(listing[x])
                for x in range(1,len(listing),2):
                    test.append(listing[x])
                    pass
                length=len(training)
                predictions=[]

                for i in range(0,len(test)):
```

```
                nbrs = find_knn(training, test[i], k,index_class,length)
                len_neighbour = len(nbrs)
                predicted_output = predict(index_class,nbrs,len_neighbour)
                predictions.append(predicted_output)

            accuracy = get_accuracy(test, predictions, index_class)
            accur_per_list.append(accuracy)

            for i in range(len(test)):
                test_class.append(test[i][index_class])

            print '----------------------------------------------------------------------'
            print 'ITERATION: #',
            print no_of_iterations+1
            print 'Accuracy: ',
            print accuracy
            print '----------------------------------------------------------------------'
            print_confusion_matrix(predictions, test_class)
            print '**********************************************************************'
            print


    mean_sd = mean_standard_deviation(accur_per_list)
    print 'Mean: ',
    print mean_sd[0]
    print 'Standard Deviation: ',
    print mean_sd[1]

elif split==0.2:
    for no_of_iterations in range(0,10):
        print '----------------------------------------------------------------------'
        print 'ITERATION: #',
        print no_of_iterations+1
        accur_per_list=[]
        start = 0.0
        end = 0.2
        for sub_iterations in range(0,5):
```

```
test_class = []
test_ver=[]
with open(datasetname,'rb') as file_dataset:
    lines=csv.reader(file_dataset)
    listing=list(lines)
    training=[]
    test=[]
    random.shuffle(listing)

    start_index = int(len(listing) * start)
    end_index = int(len(listing) * end)
    for x in range(0, len(listing)):
        if(x>=start_index and x<end_index ):
            pass
        else:
            training.append(listing[x])
    for x in range(start_index,end_index):
        test.append(listing[x])
        pass
    length=len(training)
    #print len(training)
    #print len(test)
    #print start_index
    #print end_index
    predictions=[]
    start+=0.2
    end+=0.2
    for i in range(0,len(test)):
        nbrs = find_knn(training, test[i], k,index_class,length)
        len_neighbour = len(nbrs)
        predicted_output = predict(index_class,nbrs,len_neighbour)
        predictions.append(predicted_output)

    accuracy = get_accuracy(test, predictions, index_class)
    accur_per_list.append(accuracy)

    for i in range(len(test)):
```

```
                test_class.append(test[i][index_class])

            print 'Fold #',
            print sub_iterations+1
            print 'Accuracy: ',
            print accuracy


        print '------------------------------------------------------------------------'
        mean_sd = mean_standard_deviation(accur_per_list)
        print 'Mean: ',
        print mean_sd[0]
        print 'Standard Deviation: ',
        print mean_sd[1]
        mean_record_of_iterations.append(mean_sd[0])
        print

    grand_mean_sd=[]
    grand_mean_sd = mean_standard_deviation(mean_record_of_iterations)
    print
    print 'Grand Mean: ',
    print grand_mean_sd[0]
    print 'Grand Standard Deviation: ',
    print grand_mean_sd[1]
    print
    print '------------------------------------------------------------------------'
    print_confusion_matrix(predictions, test_class)
    print '*************************************************************************'


main()
```

Random subsampling:-

- Divide the data set into 2 parts by skipping index by 2, resulting in test list and training list

- User is asked for data set to be given as input
- User is asked for Split Ratio I.e .5 for random sub-sampling.

1) Data Set - IRIS SET. Observation for (k=1)

Index of the class in dataset: 4

Enter the value of k for knn: 1

Split Ratio : 0.50 - Random SubSampling

Split Ratio : 0.20 - Five Fold Cross Validation

Enter the split ratio : 0.5

ITERATION: # 1

Accuracy: 97.3333333333

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	25	0	0
Iris-setosa	0	24	0
Iris-versicolor	2	0	24

****

ITERATION: # 2

Accuracy: 97.3333333333

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	23	0	0
Iris-setosa	0	22	0
Iris-versicolor	2	0	28

****

ITERATION: # 3
Accuracy: 96.0

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	27	0	1
Iris-setosa	0	25	0
Iris-versicolor	2	0	20

****

ITERATION: # 4
Accuracy: 96.0

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	21	0	2
Iris-setosa	0	27	0
Iris-versicolor	1	0	24

****

ITERATION: # 5
Accuracy: 96.0

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	29	0	0

Iris-setosa 0 23 0

Iris-versicolor 3 0 20

****

ITERATION: # 6

Accuracy: 96.0

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	21	0	1
Iris-setosa	0	29	0
Iris-versicolor	2	0	22

****

ITERATION: # 7

Accuracy: 93.3333333333

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	24	0	3
Iris-setosa	0	24	0
Iris-versicolor	2	0	22

****

ITERATION: # 8

Accuracy: 94.6666666667

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	26	0	4
Iris-setosa	0	23	0
Iris-versicolor	0	0	22

****

ITERATION: # 9

Accuracy: 96.0

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	21	0	1
Iris-setosa	0	27	0
Iris-versicolor	2	0	24

****

ITERATION: # 10

Accuracy: 92.0

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	24	0	4
Iris-setosa	0	20	0
Iris-versicolor	2	0	25

****

Mean: 95.4666666667

Standard Deviation: 1.6

Enter the dataset filename eg. iris.data:iris.data

Index of the class in dataset: 4

Enter the value of k for knn: 3

Split Ratio : 0.50 - Random SubSampling

Split Ratio : 0.20 - Five Fold Cross Validation

Enter the split ratio : 0.5

ITERATION: # 1

Accuracy: 97.3333333333

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	25	0	1
Iris-setosa	0	24	0
Iris-versicolor	1	0	24

****

ITERATION: # 2

Accuracy: 97.3333333333

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	26	0	1
Iris-setosa	0	25	0
Iris-versicolor	1	0	22

****

ITERATION: # 3

Accuracy: 98.6666666667

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	25	0	0
Iris-setosa	0	26	0
Iris-versicolor	1	0	23

****

ITERATION: # 4

Accuracy: 96.0

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	23	0	1
Iris-setosa	0	25	0
Iris-versicolor	2	0	24

****

ITERATION: # 5

Accuracy: 96.0

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	23	0	1

Iris-setosa	0	26	0
Iris-versicolor	2	0	23

****

ITERATION: # 6
Accuracy: 93.3333333333

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	23	0	3
Iris-setosa	0	25	0
Iris-versicolor	2	0	22

****

ITERATION: # 7
Accuracy: 96.0

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	23	0	1
Iris-setosa	0	25	0
Iris-versicolor	2	0	24

****

ITERATION: # 8
Accuracy: 96.0

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	17	0	2
Iris-setosa	0	30	0
Iris-versicolor	1	0	25

****

ITERATION: # 9

Accuracy: 96.0

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	23	0	0
Iris-setosa	0	22	0
Iris-versicolor	3	0	27

****

ITERATION: # 10

Accuracy: 97.3333333333

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	21	0	2
Iris-setosa	0	30	0
Iris-versicolor	0	0	22

****

Mean: 96.4

Standard Deviation: 1.33998341615

2) Data Set - IRIS SET. Observation for (k=3)

Enter the dataset filename eg. iris.data:iris.data

Index of the class in dataset: 4

Enter the value of k for knn: 3

Split Ratio : 0.50 - Random SubSampling

Split Ratio : 0.20 - Five Fold Cross Validation

Enter the split ratio : 0.5

ITERATION: # 1

Accuracy: 97.3333333333

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	25	0	1
Iris-setosa	0	24	0
Iris-versicolor	1	0	24

****

ITERATION: # 2

Accuracy: 97.3333333333

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	26	0	1
Iris-setosa	0	25	0

```
Iris-versicolor          1          0          22

**********************************************************************

----------------------------------------------------------------------
ITERATION: # 3
Accuracy:  98.6666666667
----------------------------------------------------------------------
               Iris-virginica  Iris-setosa     Iris-versicolor

Iris-virginica          25          0          0

Iris-setosa             0          26          0

Iris-versicolor          1          0          23

**********************************************************************

----------------------------------------------------------------------
ITERATION: # 4
Accuracy:  96.0
----------------------------------------------------------------------
               Iris-virginica  Iris-setosa     Iris-versicolor

Iris-virginica          23          0          1

Iris-setosa             0          25          0

Iris-versicolor          2          0          24

**********************************************************************

----------------------------------------------------------------------
ITERATION: # 5
Accuracy:  96.0
----------------------------------------------------------------------
```

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	23	0	1
Iris-setosa	0	26	0
Iris-versicolor	2	0	23

****

ITERATION: # 6

Accuracy: 93.3333333333

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	23	0	3
Iris-setosa	0	25	0
Iris-versicolor	2	0	22

****

ITERATION: # 7

Accuracy: 96.0

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	23	0	1
Iris-setosa	0	25	0
Iris-versicolor	2	0	24

****

ITERATION: # 8
Accuracy: 96.0

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	17	0	2
Iris-setosa	0	30	0
Iris-versicolor	1	0	25

****

ITERATION: # 9
Accuracy: 96.0

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	23	0	0
Iris-setosa	0	22	0
Iris-versicolor	3	0	27

****

ITERATION: # 10
Accuracy: 97.3333333333

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	21	0	2

Iris-setosa	0	30	0
Iris-versicolor	0	0	22

****

Mean: 96.4

Standard Deviation: 1.33998341615

3) Data Set - WINE SET. Observation for (k=1)

Enter the dataset filename eg. iris.data:wine.data

Index of the class in dataset: 0

Enter the value of k for knn: 1

Split Ratio : 0.50 - Random SubSampling

Split Ratio : 0.20 - Five Fold Cross Validation

Enter the split ratio : 0.5

ITERATION: # 1

Accuracy: 74.1573033708

	1	3	2
1	27	4	1
3	0	14	3
2	2	13	25

****

ITERATION: # 2

Accuracy: 75.2808988764

	1	3	2

```
1       29     2     0

3        3    12     3

2        2    12    26

****************************************************************************

---------------------------------------------------------------------------
ITERATION: # 3
Accuracy:  73.0337078652
----------------------------------------------------------------------------
         1     3     2

1       25     1     1

3        3     9    12

2        1     6    31

****************************************************************************

---------------------------------------------------------------------------
ITERATION: # 4
Accuracy:  71.9101123596
----------------------------------------------------------------------------
         1     3     2

1       22     1     2

3        1    14     8

2        4     9    28

****************************************************************************
```

ITERATION: # 5

Accuracy: 69.6629213483

	1	3	2
1	23	3	0
3	4	14	9
2	1	10	25

****

ITERATION: # 6

Accuracy: 68.5393258427

	1	3	2
1	22	1	1
3	4	18	5
2	5	12	21

****

ITERATION: # 7

Accuracy: 68.5393258427

	1	3	2
1	24	3	3
3	1	10	14

2	2	5	27

****

ITERATION: # 8

Accuracy: 69.6629213483

	1	3	2
1	21	2	5
3	3	15	4
2	3	10	26

****

ITERATION: # 9

Accuracy: 71.9101123596

	1	3	2
1	25	1	0
3	2	17	3
2	5	14	22

****

ITERATION: # 10

Accuracy: 74.1573033708

	1	3	2
1	27	3	1
3	2	14	6
2	2	9	25

****

Mean: 71.6853932584
Standard Deviation: 2.34613629414

<u>4) Data Set - WINE SET. Observation for (k=3)</u>

Enter the dataset filename eg. iris.data:wine.data
Index of the class in dataset: 0
Enter the value of k for knn: 3
Split Ratio : 0.50 - Random SubSampling
Split Ratio : 0.20 - Five Fold Cross Validation
Enter the split ratio : 0.5

ITERATION: # 1
Accuracy: 75.2808988764

	1	3	2
1	28	1	1
3	4	10	5
2	3	8	29

****

ITERATION: # 2

Accuracy: 75.2808988764

	1	3	2
1	25	5	0
3	0	15	6
2	3	8	27

****

ITERATION: # 3

Accuracy: 74.1573033708

	1	3	2
1	24	3	0
3	4	15	5
2	4	7	27

****

ITERATION: # 4

Accuracy: 73.0337078652

	1	3	2
1	29	2	1
3	1	12	9

2 5 6 24

****

ITERATION: # 5

Accuracy: 70.7865168539

	1	3	2
1	24	2	0
3	1	13	9
2	3	11	26

****

ITERATION: # 6

Accuracy: 69.6629213483

	1	3	2
1	26	2	0
3	3	15	2
2	3	17	21

****

ITERATION: # 7

Accuracy: 66.2921348315

	1	3	2
1	22	1	3
3	5	12	10
2	5	6	25

****

ITERATION: # 8
Accuracy: 70.7865168539

	1	3	2
1	25	4	2
3	1	14	8
2	2	9	24

****

ITERATION: # 9
Accuracy: 74.1573033708

	1	3	2
1	25	2	1
3	2	11	7
2	1	10	30

****

ITERATION: # 10

Accuracy: 71.9101123596

	1	3	2
1	27	0	2
3	3	15	6
2	2	12	22

****

Mean: 72.1348314607

Standard Deviation: 2.69662921348

5) Data Set - BREAST CANCER-WISCONSIN SET. Observation for (k=1)

Enter the dataset filename eg. iris.data:wisconsin.data

Index of the class in dataset: 10

Enter the value of k for knn: 1

Split Ratio : 0.50 - Random SubSampling

Split Ratio : 0.20 - Five Fold Cross Validation

Enter the split ratio : 0.5

ITERATION: # 1

Accuracy: 93.9828080229

	2	4
2	226	3
4	18	102

****

ITERATION: # 2
Accuracy: 94.5558739255

	2	4
2	215	5
4	14	115

****

ITERATION: # 3
Accuracy: 95.1289398281

	2	4
2	208	7
4	10	124

****

ITERATION: # 4
Accuracy: 95.4154727794

	2	4
2	232	7
4	9	101

****

ITERATION: # 5

Accuracy: 96.8481375358

	2	4
2	217	8
4	3	121

****

ITERATION: # 6

Accuracy: 95.7020057307

	2	4
2	228	9
4	6	106

****

ITERATION: # 7

Accuracy: 94.5558739255

	2	4
2	218	8
4	11	112

****

ITERATION: # 8

Accuracy: 95.1289398281

	2	4
2	218	6
4	11	114

****

ITERATION: # 9

Accuracy: 95.1289398281

	2	4
2	232	10
4	7	100

****

ITERATION: # 10

Accuracy: 95.4154727794

	2	4
2	234	5
4	11	99

****

Mean: 95.1862464183

Standard Deviation: 0.73388243409

6) Data Set - BREAST CANCER-WISCONSIN SET. Observation for (k=3)

Enter the dataset filename eg. iris.data:wisconsin.data

Index of the class in dataset: 10

Enter the value of k for knn: 3

Split Ratio : 0.50 - Random SubSampling

Split Ratio : 0.20 - Five Fold Cross Validation

Enter the split ratio : 0.5

ITERATION: # 1

Accuracy: 96.5616045845

	2	4
2	221	5
4	7	116

****

ITERATION: # 2

Accuracy: 95.9885386819

	2	4
2	221	9
4	5	114

****

ITERATION: # 3

Accuracy: 97.4212034384

	2	4
2	234	6
4	3	106

****

ITERATION: # 4

Accuracy: 96.8481375358

	2	4
2	219	8
4	3	119

****

ITERATION: # 5

Accuracy: 94.5558739255

	2	4
2	228	8
4	11	102

****

ITERATION: # 6

Accuracy: 95.1289398281

----

2 4

2 218 6

4 11 114

****

ITERATION: # 7

Accuracy: 96.2750716332

----

2 4

2 219 11

4 2 117

****

ITERATION: # 8

Accuracy: 97.1346704871

----

2 4

2 226 6

4 4 113

****

ITERATION: # 9

Accuracy: 95.4154727794

----

2 4

2 223 6

4 10 110

****

ITERATION: # 10
Accuracy: 95.9885386819
----
2 4

2 227 9

4 5 108

****

Mean: 96.1318051576
Standard Deviation: 0.861983321289

K-Fold Cross Validation

- Divide the data set into 2 parts by taking 80% for training set and 20% for test set iteratively for 5 folds.
- User is asked for data set to be given as input
- User is asked for Split Ratio I.e .2 for five fold cross validation.

1) Data Set - IRIS SET. Observation for (k=1)

Enter the dataset filename eg. iris.data:iris.data
Index of the class in dataset: 4
Enter the value of k for knn: 1

Split Ratio : 0.50 - Random SubSampling

Split Ratio : 0.20 - Five Fold Cross Validation

Enter the split ratio : 0.2

ITERATION: # 1

Fold # 1

Accuracy: 96.6666666667

Fold # 2

Accuracy: 96.6666666667

Fold # 3

Accuracy: 96.6666666667

Fold # 4

Accuracy: 93.3333333333

Fold # 5

Accuracy: 90.0

Mean: 94.6666666667

Standard Deviation: 2.66666666667

ITERATION: # 2

Fold # 1

Accuracy: 90.0

Fold # 2

Accuracy: 96.6666666667

Fold # 3

Accuracy: 93.3333333333

Fold # 4

Accuracy: 93.3333333333

Fold # 5

Accuracy: 96.6666666667

Mean: 94.0

Standard Deviation: 2.49443825785

ITERATION: # 3

Fold # 1
Accuracy: 93.3333333333
Fold # 2
Accuracy: 90.0
Fold # 3
Accuracy: 96.6666666667
Fold # 4
Accuracy: 100.0
Fold # 5
Accuracy: 90.0

Mean: 94.0
Standard Deviation: 3.88730126323

ITERATION: # 4
Fold # 1
Accuracy: 93.3333333333
Fold # 2
Accuracy: 100.0
Fold # 3
Accuracy: 96.6666666667
Fold # 4
Accuracy: 96.6666666667
Fold # 5
Accuracy: 96.6666666667

Mean: 96.6666666667
Standard Deviation: 2.10818510678

ITERATION: # 5
Fold # 1
Accuracy: 100.0
Fold # 2
Accuracy: 96.6666666667
Fold # 3

Accuracy: 96.6666666667
Fold # 4
Accuracy: 100.0
Fold # 5
Accuracy: 93.3333333333

Mean: 97.3333333333
Standard Deviation: 2.49443825785

ITERATION: # 6
Fold # 1
Accuracy: 96.6666666667
Fold # 2
Accuracy: 100.0
Fold # 3
Accuracy: 96.6666666667
Fold # 4
Accuracy: 96.6666666667
Fold # 5
Accuracy: 93.3333333333

Mean: 96.6666666667
Standard Deviation: 2.10818510678

ITERATION: # 7
Fold # 1
Accuracy: 96.6666666667
Fold # 2
Accuracy: 93.3333333333
Fold # 3
Accuracy: 96.6666666667
Fold # 4
Accuracy: 100.0
Fold # 5
Accuracy: 100.0

Mean: 97.3333333333

Standard Deviation: 2.49443825785

ITERATION: # 8

Fold # 1

Accuracy: 86.6666666667

Fold # 2

Accuracy: 93.3333333333

Fold # 3

Accuracy: 93.3333333333

Fold # 4

Accuracy: 96.6666666667

Fold # 5

Accuracy: 100.0

Mean: 94.0

Standard Deviation: 4.42216638714

ITERATION: # 9

Fold # 1

Accuracy: 100.0

Fold # 2

Accuracy: 96.6666666667

Fold # 3

Accuracy: 93.3333333333

Fold # 4

Accuracy: 90.0

Fold # 5

Accuracy: 96.6666666667

Mean: 95.3333333333

Standard Deviation: 3.3993463424

ITERATION: # 10
Fold # 1
Accuracy: 96.6666666667
Fold # 2
Accuracy: 93.3333333333
Fold # 3
Accuracy: 93.3333333333
Fold # 4
Accuracy: 96.6666666667
Fold # 5
Accuracy: 96.6666666667

Mean: 95.3333333333
Standard Deviation: 1.63299316186

Grand Mean: 95.5333333333
Grand Standard Deviation: 1.30128141973

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	9	0	1
Iris-setosa	0	12	0
Iris-versicolor	0	0	8

****

<u>2) Data Set - IRIS SET. Observation for (k=3)</u>

Enter the dataset filename eg. iris.data:iris.data
Index of the class in dataset: 4
Enter the value of k for knn: 3
Split Ratio : 0.50 - Random SubSampling
Split Ratio : 0.20 - Five Fold Cross Validation

```
Enter the split ratio : 0.2
-----------------------------------------------------------------------
ITERATION: # 1
Fold # 1
Accuracy:  100.0
Fold # 2
Accuracy:  100.0
Fold # 3
Accuracy:  90.0
Fold # 4
Accuracy:  100.0
Fold # 5
Accuracy:  100.0
------------------------------------------------------------------------
Mean:  98.0
Standard Deviation:  4.0

-----------------------------------------------------------------------
ITERATION: # 2
Fold # 1
Accuracy:  100.0
Fold # 2
Accuracy:  100.0
Fold # 3
Accuracy:  93.3333333333
Fold # 4
Accuracy:  96.6666666667
Fold # 5
Accuracy:  100.0
------------------------------------------------------------------------
Mean:  98.0
Standard Deviation:  2.66666666667

-----------------------------------------------------------------------
ITERATION: # 3
Fold # 1
Accuracy:  96.6666666667
```

Fold # 2

Accuracy: 93.3333333333

Fold # 3

Accuracy: 96.6666666667

Fold # 4

Accuracy: 96.6666666667

Fold # 5

Accuracy: 100.0

Mean: 96.6666666667

Standard Deviation: 2.10818510678

ITERATION: # 4

Fold # 1

Accuracy: 96.6666666667

Fold # 2

Accuracy: 96.6666666667

Fold # 3

Accuracy: 100.0

Fold # 4

Accuracy: 90.0

Fold # 5

Accuracy: 96.6666666667

Mean: 96.0

Standard Deviation: 3.26598632371

ITERATION: # 5

Fold # 1

Accuracy: 90.0

Fold # 2

Accuracy: 96.6666666667

Fold # 3

Accuracy: 100.0

Fold # 4

Accuracy: 100.0
Fold # 5
Accuracy: 96.6666666667

Mean: 96.6666666667
Standard Deviation: 3.6514837167

ITERATION: # 6
Fold # 1
Accuracy: 96.6666666667
Fold # 2
Accuracy: 96.6666666667
Fold # 3
Accuracy: 100.0
Fold # 4
Accuracy: 100.0
Fold # 5
Accuracy: 96.6666666667

Mean: 98.0
Standard Deviation: 1.63299316186

ITERATION: # 7
Fold # 1
Accuracy: 93.3333333333
Fold # 2
Accuracy: 96.6666666667
Fold # 3
Accuracy: 96.6666666667
Fold # 4
Accuracy: 93.3333333333
Fold # 5
Accuracy: 100.0

Mean: 96.0

Standard Deviation: 2.49443825785

ITERATION: # 8

Fold # 1

Accuracy: 96.6666666667

Fold # 2

Accuracy: 93.3333333333

Fold # 3

Accuracy: 90.0

Fold # 4

Accuracy: 93.3333333333

Fold # 5

Accuracy: 96.6666666667

Mean: 94.0

Standard Deviation: 2.49443825785

ITERATION: # 9

Fold # 1

Accuracy: 96.6666666667

Fold # 2

Accuracy: 96.6666666667

Fold # 3

Accuracy: 93.3333333333

Fold # 4

Accuracy: 96.6666666667

Fold # 5

Accuracy: 96.6666666667

Mean: 96.0

Standard Deviation: 1.33333333333

ITERATION: # 10

Fold # 1

Accuracy: 96.6666666667
Fold # 2
Accuracy: 93.3333333333
Fold # 3
Accuracy: 96.6666666667
Fold # 4
Accuracy: 100.0
Fold # 5
Accuracy: 100.0

Mean: 97.3333333333
Standard Deviation: 2.49443825785

Grand Mean: 96.6666666667
Grand Standard Deviation: 1.192569588

	Iris-virginica	Iris-setosa	Iris-versicolor
Iris-virginica	7	0	0
Iris-setosa	0	14	0
Iris-versicolor	0	0	9

****

3) Data Set - WINE SET. Observation for (k=1)

Enter the dataset filename eg. iris.data:wine.data
Index of the class in dataset: 0
Enter the value of k for knn: 1
Split Ratio : 0.50 - Random SubSampling
Split Ratio : 0.20 - Five Fold Cross Validation
Enter the split ratio : 0.2

ITERATION: # 1

Fold # 1

Accuracy: 60.0

Fold # 2

Accuracy: 83.3333333333

Fold # 3

Accuracy: 68.5714285714

Fold # 4

Accuracy: 72.2222222222

Fold # 5

Accuracy: 83.3333333333

Mean: 73.4920634921

Standard Deviation: 8.96171977161

ITERATION: # 2

Fold # 1

Accuracy: 60.0

Fold # 2

Accuracy: 75.0

Fold # 3

Accuracy: 80.0

Fold # 4

Accuracy: 75.0

Fold # 5

Accuracy: 77.7777777778

Mean: 73.5555555556

Standard Deviation: 7.03255217709

ITERATION: # 3

Fold # 1

Accuracy: 80.0

Fold # 2

Accuracy: 77.7777777778

Fold # 3

Accuracy: 62.8571428571
Fold # 4
Accuracy: 72.2222222222
Fold # 5
Accuracy: 75.0

Mean: 73.5714285714
Standard Deviation: 5.96115755075

ITERATION: # 4
Fold # 1
Accuracy: 74.2857142857
Fold # 2
Accuracy: 58.3333333333
Fold # 3
Accuracy: 82.8571428571
Fold # 4
Accuracy: 75.0
Fold # 5
Accuracy: 80.5555555556

Mean: 74.2063492063
Standard Deviation: 8.57598350461

ITERATION: # 5
Fold # 1
Accuracy: 77.1428571429
Fold # 2
Accuracy: 72.2222222222
Fold # 3
Accuracy: 80.0
Fold # 4
Accuracy: 77.7777777778
Fold # 5
Accuracy: 69.4444444444

Mean: 75.3174603175

Standard Deviation: 3.88289139084

ITERATION: # 6

Fold # 1

Accuracy: 71.4285714286

Fold # 2

Accuracy: 88.8888888889

Fold # 3

Accuracy: 82.8571428571

Fold # 4

Accuracy: 75.0

Fold # 5

Accuracy: 61.1111111111

Mean: 75.8571428571

Standard Deviation: 9.5563465369

ITERATION: # 7

Fold # 1

Accuracy: 68.5714285714

Fold # 2

Accuracy: 66.6666666667

Fold # 3

Accuracy: 80.0

Fold # 4

Accuracy: 69.4444444444

Fold # 5

Accuracy: 77.7777777778

Mean: 72.4920634921

Standard Deviation: 5.34607321931

ITERATION: # 8

Fold # 1

Accuracy: 82.8571428571

Fold # 2

Accuracy: 83.3333333333

Fold # 3

Accuracy: 77.1428571429

Fold # 4

Accuracy: 72.2222222222

Fold # 5

Accuracy: 83.3333333333

Mean: 79.7777777778

Standard Deviation: 4.4451246645

ITERATION: # 9

Fold # 1

Accuracy: 77.1428571429

Fold # 2

Accuracy: 80.5555555556

Fold # 3

Accuracy: 74.2857142857

Fold # 4

Accuracy: 69.4444444444

Fold # 5

Accuracy: 77.7777777778

Mean: 75.8412698413

Standard Deviation: 3.76849600057

ITERATION: # 10

Fold # 1

Accuracy: 80.0

Fold # 2

Accuracy: 69.4444444444

Fold # 3

Accuracy: 77.1428571429

Fold # 4

Accuracy: 83.3333333333

Fold # 5

Accuracy: 83.3333333333

Mean: 78.6507936508

Standard Deviation: 5.15176575071

Grand Mean: 75.2761904762

Grand Standard Deviation: 2.2417981301

	1	3	2
1	14	0	1
3	0	5	2
2	0	3	11

****

4) Data Set - WINE SET. Observation for (k=3)

Enter the dataset filename eg. iris.data:wine.data

Index of the class in dataset: 0

Enter the value of k for knn: 3

Split Ratio : 0.50 - Random SubSampling

Split Ratio : 0.20 - Five Fold Cross Validation

Enter the split ratio : 0.2

ITERATION: # 1

Fold # 1

Accuracy: 65.7142857143

Fold # 2

Accuracy: 75.0

Fold # 3

Accuracy: 82.8571428571

Fold # 4

Accuracy: 83.3333333333

Fold # 5

Accuracy: 72.2222222222

Mean: 75.8253968254

Standard Deviation: 6.65902812584

ITERATION: # 2

Fold # 1

Accuracy: 54.2857142857

Fold # 2

Accuracy: 69.4444444444

Fold # 3

Accuracy: 60.0

Fold # 4

Accuracy: 72.2222222222

Fold # 5

Accuracy: 80.5555555556

Mean: 67.3015873016

Standard Deviation: 9.24335747142

ITERATION: # 3

Fold # 1

Accuracy: 74.2857142857

Fold # 2

Accuracy: 63.8888888889

Fold # 3

Accuracy: 80.0

Fold # 4

Accuracy: 61.1111111111

Fold # 5

Accuracy: 58.3333333333

Mean: 67.5238095238

Standard Deviation: 8.24838028184

ITERATION: # 4

Fold # 1

Accuracy: 71.4285714286

Fold # 2

Accuracy: 75.0

Fold # 3

Accuracy: 71.4285714286

Fold # 4

Accuracy: 66.6666666667

Fold # 5

Accuracy: 91.6666666667

Mean: 75.2380952381

Standard Deviation: 8.630747124

ITERATION: # 5

Fold # 1

Accuracy: 74.2857142857

Fold # 2

Accuracy: 75.0

Fold # 3

Accuracy: 60.0

Fold # 4

Accuracy: 77.7777777778

Fold # 5

Accuracy: 66.6666666667

Mean: 70.746031746

Standard Deviation: 6.51443732612

ITERATION: # 6

Fold # 1

Accuracy: 71.4285714286

Fold # 2

Accuracy: 83.3333333333

Fold # 3

Accuracy: 74.2857142857

Fold # 4

Accuracy: 66.6666666667

Fold # 5

Accuracy: 72.2222222222

Mean: 73.5873015873

Standard Deviation: 5.47524721575

ITERATION: # 7

Fold # 1

Accuracy: 74.2857142857

Fold # 2

Accuracy: 63.8888888889

Fold # 3

Accuracy: 68.5714285714

Fold # 4

Accuracy: 77.7777777778

Fold # 5

Accuracy: 63.8888888889

Mean: 69.6825396825

Standard Deviation: 5.56937057817

ITERATION: # 8

Fold # 1

Accuracy: 60.0

Fold # 2

Accuracy: 69.4444444444

Fold # 3

Accuracy: 80.0

Fold # 4

Accuracy: 63.8888888889

Fold # 5

Accuracy: 66.6666666667

Mean: 68.0

Standard Deviation: 6.7641027801

ITERATION: # 9

Fold # 1

Accuracy: 71.4285714286

Fold # 2

Accuracy: 77.7777777778

Fold # 3

Accuracy: 65.7142857143

Fold # 4

Accuracy: 66.6666666667

Fold # 5

Accuracy: 83.3333333333

Mean: 72.9841269841

Standard Deviation: 6.71216522484

ITERATION: # 10

Fold # 1

Accuracy: 68.5714285714

Fold # 2

Accuracy: 77.7777777778

Fold # 3

Accuracy: 65.7142857143

Fold # 4

Accuracy: 86.1111111111

Fold # 5

Accuracy: 75.0

Mean: 74.6349206349

Standard Deviation: 7.18411365723

Grand Mean: 71.5523809524

Grand Standard Deviation: 3.1400725198

	1	3	2
1	11	0	0
3	2	3	5
2	0	2	13

****

<u>5) Data Set - BREAST CANCER DATA SET. Observation for (k=1)</u>

Enter the dataset filename eg. iris.data:wisconsin.data

Index of the class in dataset: 10

Enter the value of k for knn: 1

Split Ratio : 0.50 - Random SubSampling

Split Ratio : 0.20 - Five Fold Cross Validation

Enter the split ratio : 0.2

ITERATION: # 1

Fold # 1

Accuracy: 94.964028777

Fold # 2

Accuracy: 95.0

Fold # 3

Accuracy: 93.5714285714
Fold # 4
Accuracy: 93.5714285714
Fold # 5
Accuracy: 93.5714285714

Mean: 94.1356628983
Standard Deviation: 0.691136713132

ITERATION: # 2
Fold # 1
Accuracy: 93.5251798561
Fold # 2
Accuracy: 96.4285714286
Fold # 3
Accuracy: 95.0
Fold # 4
Accuracy: 97.1428571429
Fold # 5
Accuracy: 95.0

Mean: 95.4193216855
Standard Deviation: 1.25925328178

ITERATION: # 3
Fold # 1
Accuracy: 97.1223021583
Fold # 2
Accuracy: 95.7142857143
Fold # 3
Accuracy: 94.2857142857
Fold # 4
Accuracy: 97.1428571429
Fold # 5
Accuracy: 92.8571428571

Mean: 95.4244604317

Standard Deviation: 1.66177103077

ITERATION: # 4

Fold # 1

Accuracy: 94.964028777

Fold # 2

Accuracy: 94.2857142857

Fold # 3

Accuracy: 95.7142857143

Fold # 4

Accuracy: 92.8571428571

Fold # 5

Accuracy: 95.0

Mean: 94.5642343268

Standard Deviation: 0.965824023024

ITERATION: # 5

Fold # 1

Accuracy: 94.964028777

Fold # 2

Accuracy: 93.5714285714

Fold # 3

Accuracy: 92.1428571429

Fold # 4

Accuracy: 94.2857142857

Fold # 5

Accuracy: 95.7142857143

Mean: 94.1356628983

Standard Deviation: 1.22396001548

ITERATION: # 6

Fold # 1

Accuracy: 93.5251798561

Fold # 2

Accuracy: 92.1428571429

Fold # 3

Accuracy: 97.1428571429

Fold # 4

Accuracy: 98.5714285714

Fold # 5

Accuracy: 96.4285714286

Mean: 95.5621788284

Standard Deviation: 2.37258886829

ITERATION: # 7

Fold # 1

Accuracy: 92.8057553957

Fold # 2

Accuracy: 93.5714285714

Fold # 3

Accuracy: 95.0

Fold # 4

Accuracy: 95.0

Fold # 5

Accuracy: 95.7142857143

Mean: 94.4182939363

Standard Deviation: 1.06525302805

ITERATION: # 8

Fold # 1

Accuracy: 97.1223021583

Fold # 2

Accuracy: 96.4285714286

Fold # 3

Accuracy: 96.4285714286

Fold # 4

Accuracy: 96.4285714286

Fold # 5

Accuracy: 97.8571428571

Mean: 96.8530318602

Standard Deviation: 0.569428724541

ITERATION: # 9

Fold # 1

Accuracy: 95.6834532374

Fold # 2

Accuracy: 92.8571428571

Fold # 3

Accuracy: 95.0

Fold # 4

Accuracy: 95.7142857143

Fold # 5

Accuracy: 95.0

Mean: 94.8509763618

Standard Deviation: 1.0448070559

ITERATION: # 10

Fold # 1

Accuracy: 96.4028776978

Fold # 2

Accuracy: 95.0

Fold # 3

Accuracy: 97.1428571429

Fold # 4

Accuracy: 97.1428571429

Fold # 5

Accuracy: 96.4285714286

Mean: 96.4234326824

Standard Deviation: 0.7825282901

Grand Mean: 95.178725591

Grand Standard Deviation: 0.886405135307

	2	4
2	92	3
4	2	43

****

6) Data Set - BREAST CANCER DATA SET. Observation for (k=3)

Enter the dataset filename eg. iris.data:wisconsin.data

Index of the class in dataset: 10

Enter the value of k for knn: 3

Split Ratio : 0.50 - Random SubSampling

Split Ratio : 0.20 - Five Fold Cross Validation

Enter the split ratio : 0.2

ITERATION: # 1

Fold # 1

Accuracy: 99.2805755396

Fold # 2

Accuracy: 95.7142857143

Fold # 3

Accuracy: 96.4285714286

Fold # 4

Accuracy: 96.4285714286

Fold # 5

Accuracy: 95.7142857143

Mean: 96.7132579651

Standard Deviation: 1.32280788417

ITERATION: # 2

Fold # 1

Accuracy: 97.1223021583

Fold # 2

Accuracy: 94.2857142857

Fold # 3

Accuracy: 97.8571428571

Fold # 4

Accuracy: 95.0

Fold # 5

Accuracy: 95.7142857143

Mean: 95.9958890031

Standard Deviation: 1.32127706315

ITERATION: # 3

Fold # 1

Accuracy: 94.964028777

Fold # 2

Accuracy: 96.4285714286

Fold # 3

Accuracy: 95.7142857143

Fold # 4

Accuracy: 95.7142857143

Fold # 5

Accuracy: 96.4285714286

Mean: 95.8499486125

Standard Deviation: 0.546126638323

ITERATION: # 4

Fold # 1

Accuracy: 95.6834532374

Fold # 2

Accuracy: 95.7142857143

Fold # 3

Accuracy: 97.1428571429

Fold # 4

Accuracy: 98.5714285714

Fold # 5

Accuracy: 96.4285714286

Mean: 96.7081192189

Standard Deviation: 1.07486847394

ITERATION: # 5

Fold # 1

Accuracy: 97.8417266187

Fold # 2

Accuracy: 97.1428571429

Fold # 3

Accuracy: 97.1428571429

Fold # 4

Accuracy: 97.8571428571

Fold # 5

Accuracy: 96.4285714286

Mean: 97.282631038

Standard Deviation: 0.531251917396

ITERATION: # 6

Fold # 1

Accuracy: 93.5251798561

Fold # 2

Accuracy: 95.7142857143

Fold # 3

Accuracy: 98.5714285714

Fold # 4

Accuracy: 99.2857142857

Fold # 5

Accuracy: 95.7142857143

Mean: 96.5621788284

Standard Deviation: 2.10313756757

ITERATION: # 7

Fold # 1

Accuracy: 99.2805755396

Fold # 2

Accuracy: 95.7142857143

Fold # 3

Accuracy: 96.4285714286

Fold # 4

Accuracy: 96.4285714286

Fold # 5

Accuracy: 95.0

Mean: 96.5704008222

Standard Deviation: 1.45494810717

ITERATION: # 8

Fold # 1

Accuracy: 95.6834532374

Fold # 2

Accuracy: 98.5714285714

Fold # 3

Accuracy: 95.7142857143

Fold # 4

Accuracy: 93.5714285714

Fold # 5

Accuracy: 97.8571428571

Mean: 96.2795477903

Standard Deviation: 1.77483989948

ITERATION: # 9

Fold # 1

Accuracy: 96.4028776978

Fold # 2

Accuracy: 94.2857142857

Fold # 3

Accuracy: 97.8571428571

Fold # 4

Accuracy: 97.1428571429

Fold # 5

Accuracy: 95.7142857143

Mean: 96.2805755396

Standard Deviation: 1.22834909342

ITERATION: # 10

Fold # 1

Accuracy: 94.964028777

Fold # 2

Accuracy: 96.4285714286

Fold # 3

Accuracy: 95.7142857143

Fold # 4

Accuracy: 95.7142857143

Fold # 5

Accuracy: 97.1428571429

Mean: 95.9928057554

Standard Deviation: 0.738381854279

Grand Mean: 96.4235354573

Grand Standard Deviation: 0.410164004534

	2	4
2	89	2
4	2	47

****

Q3.

Plot the Iris dataset using only two features, namely, Petal width and Sepal width features (2D Plot). Plot the decision boundaries of the 1-nearest neighbor classifier in this plot. You should generate the plot automatically (using code). A simple but crude method is to classify each point in a 2D grid and find the transition points

Sol:

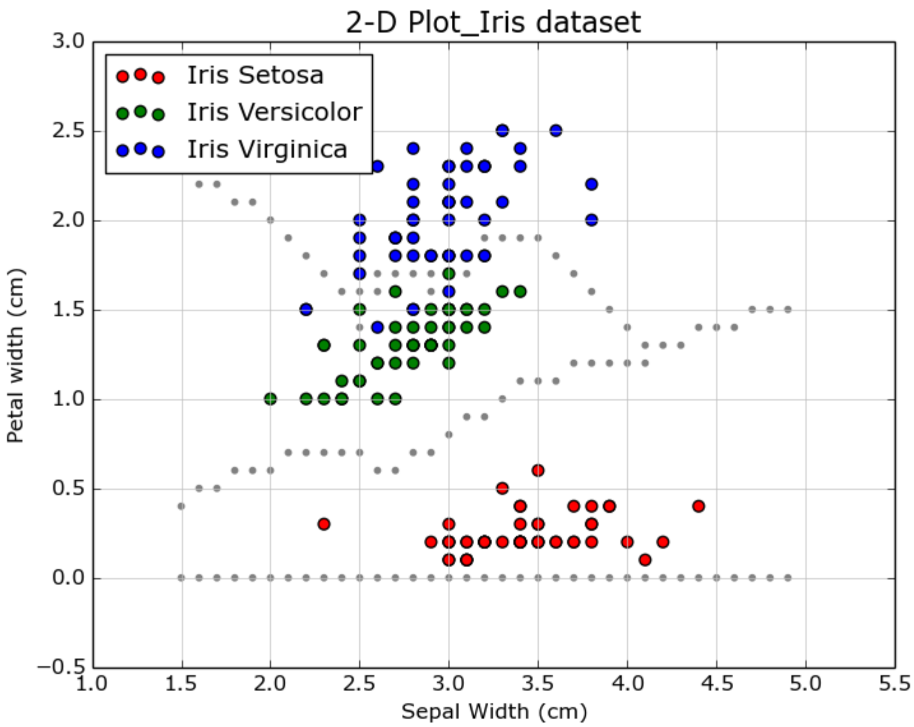
<u>Code for Plotting</u>:-

```
import matplotlib.pyplot as py

index=0
preindex=0

j=0.5
lists=[]
decision_line_x=[]
decision_line_y=[]
f = open('iris.data', 'rb')
dataset = f.readlines()
 # petal width
```

```
xcord3 = []
ycord3 = []
for i in range(len(dataset)):
    test=[]
     test = dataset[i].split(',')
     length=len(lists)
     lists.insert(length,test)
y = 3
x = 1

xcord1 = []
ycord1 = []
xcord2 = []
ycord2 = []
matrix, labels=[map(float, l[:4]) for l in lists], [l[-1] for l in lists]
#Locating x and y coordinates of the points given ( sepal width and petal width)
for n, elem in enumerate(matrix):
    if labels[n] == 'Iris-setosa\n' and labels[n]!='':
        xcord1.insert(len(xcord1),matrix[n][x])
        ycord1.insert(len(ycord1),matrix[n][y])
    if labels[n] == 'Iris-versicolor\n' and labels[n]!='':
        xcord2.insert(len(xcord2),matrix[n][x])
        ycord2.insert(len(ycord2),matrix[n][y])
    if labels[n] == 'Iris-virginica\n' and labels[n]!='':
        xcord3.insert(len(xcord3),matrix[n][x])
        ycord3.insert(len(ycord3),matrix[n][y])
i=1.5
j=0
ax = py.figure().add_subplot(111)
while i<=4.9:
    while j<3:
        preindex=index
        max_value=10000000
        for k in range(0,150):
            if max_value>pow(matrix[k][1]-i,2)+pow(matrix[k][3]-j,2):
                max_value=pow(matrix[k][1]-i,2)+pow(matrix[k][3]-j,2)
                if labels[k]=='Iris-setosa\n':
```

```
                index=1
            if labels[k]=='Iris-versicolor\n':
                index=2
            if labels[k]=='Iris-virginica\n':
                index=3
            else:
                pass
        if preindex ==index:
            pass
        else:
            length_line=len(decision_line_x)
            decision_line_x.insert(length_line,i)
            decision_line_y.insert(length_line,j)
        j+=0.1
    j=0
    i+=0.1
ax.scatter(decision_line_x, decision_line_y, 10, color ='grey')
ax.set_title('2-D Plot_Iris dataset', fontsize=16)
ax.set_xlabel('Sepal Width (cm)')
ax.set_ylabel('Petal width (cm)')
ax.legend([ax.scatter(xcord1, ycord1, s=40, c='red'), ax.scatter(xcord2, ycord2, s=40, c='green'),
ax.scatter(xcord3, ycord3, s=40, c='blue')], ["Iris Setosa", "Iris Versicolor", "Iris Virginica"], loc=2)
ax.grid(True,linestyle='-',color='0.75')
py.show()
```

Graph:-



Q4.

Will the decision boundary of a 3-NN (nearest neighbor) classifier be piecewise linear? Argue the correctness of your answer.

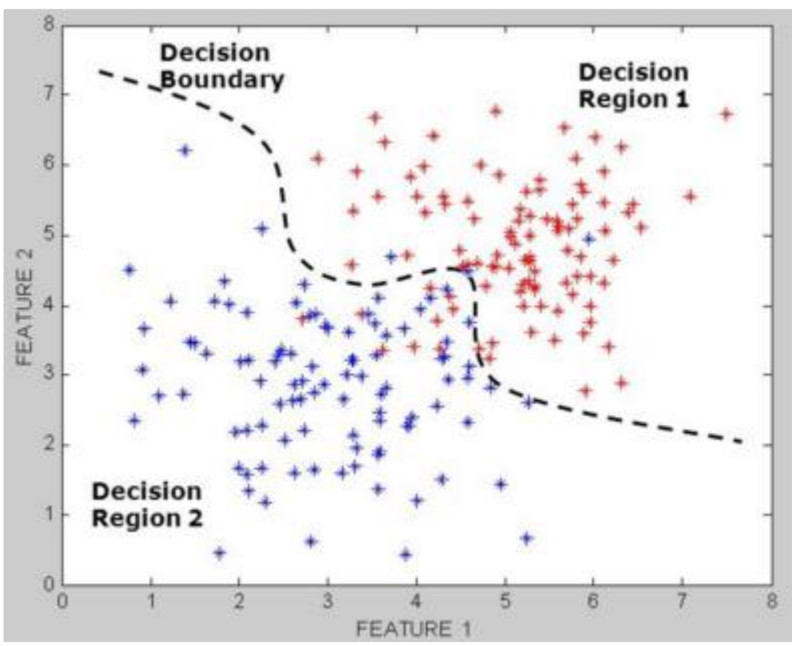
Sol:

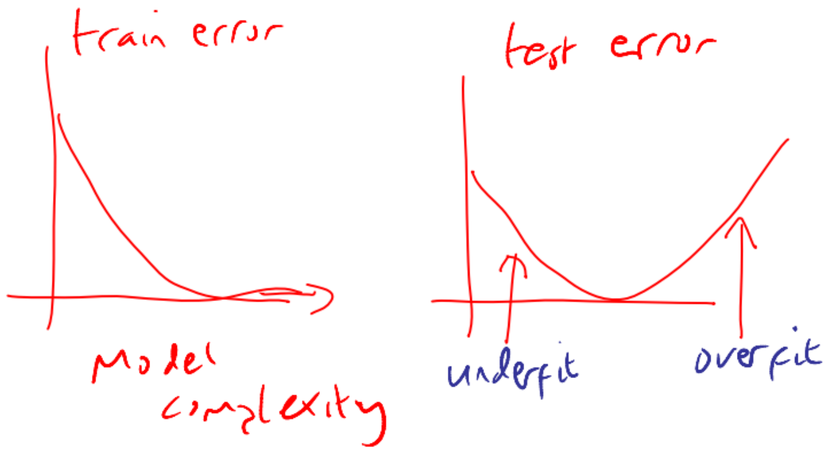
- Classification decision based on majority of k nearest neighbors.
- Here for k=3, The decision boundaries between classes are piecewise linear .
- But they are in general not linear classifiers that can be described as $\sum_{i=1}^{M} w_i d_i = \theta.$
- In terms of actual computation, there are two types of learning algorithms.
    - Simple learning algorithms that estimate the parameters of the classifier directly from the training data, often in one linear pass.
        - Naive Bayes, Rocchio, kNN are all examples of this.

- Iterative algorithms
    - Support vector machines
    - Perceptron
- The best performing learning algorithms usually require iterative learning.

- Euclidean distance is used as the distance measure (the most common choice), the nearest neighbor classifier. It results in piecewise linear decision boundaries. (Here it is piecewise linear as k=3).

- Increasing the value of k for KNN “Simplifies” / “Smoothens” decision boundary.
    - Majority voting means less emphasis on individual points.
    - Here K variation is small, thus we cannot figure out easily, but effectively, with





- We can see that the training error rate tends to grow when k grows. The test error rate or cross-validation results indicate there is a balance between k and the error rate. When k first increases, the error rate decreases, and it increases again when k becomes too big. Hence, there is a preference for k in a certain range.

- K=1 yields y=piecewise constant labeling. K = N predicts y=globally constant (majority) label.

- Knn is not a linear classifier