STATISTICAL METHODS IN AI

Assignment 1:

K-NEAREST NEIGHBOUR CLASSIFIER

Megha Agarwal

January 20, 2016
M.Tech - CSIS (201506511)

Introduction

K-Nearest Neighbours Algorithm:

In pattern recognition, the ***k*-Nearest Neighbors algorithm** is a non-parametric method used for classification and regression. In both cases, the input consists of the *k* closest training examples in the feature space. The output depends on whether *k*-NN is used for classification or regression. It is a type of instance-based learning, or lazy learning.

- The training examples are vectors in a multidimensional feature space, each with a class label.
- The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.
- In the classification phase, *k* is a user-defined constant.
- The unlabeled vector (test sample) is classified by assigning the label which is most frequent among the *k* training samples nearest to that query point.
- The distance between the unlabeled test samples and all the points in the training sample are calculated by Euclidean distance, which is given by:

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \mathrm{d}(\mathbf{q},\mathbf{p}) = \sqrt{(q_1-p_1)^2 + (q_2-p_2)^2 + \cdots + (q_n-p_n)^2}$$
$$= \sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}.$$

Requirements:

The given assignment requires us to implement K-nearest neighbour (kNN) classifier and test it on three different datasets.

Data Sets Chosen :

1. Iris Data Set
2. Wine Data Set
3. Breast Cancer Data Set

Methods to divide data in training sample and test sample :

1. Random Subsampling
2. Five Fold Cross Validation

References : UCI Machine learning repository (http://archive.ics.uci.edu/ml/).

Programming Language: Python

Experimental Set-Up :

1. Handle data :

We need to load the data from the file. All the lines are read into a lists of lists i.e. dataset. It is further divided into training sample and test sample as per the proper split ratio. (0.5 in case of random subsampling) All the numerical values are also converted to float and then stored in the training and test data respectively.

2. Find the distance between the test instance and training set:

For able to make the prediction of the class name, the data belongs to, we need to calculate the similarity i.e distances between any two given data instances. This is done so that we can locate the k (1 or 3) most nearest data instances, and hence can make the prediction. Given that all four attributes are numeric and have the same units, we can directly use the Euclidean distance measure. Additionally, we want to 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 neighbours and handling ties

Now, we need to use the distances obtained to collect the k most similar instances for a given unseen instance. It is done by sorting the distance list, on the distance value, and hence selecting the first k values irrespective of the ties (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. Now, 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. Calculate Accuracy

We 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.

QUESTIONS TO BE ANSWERED:

Q-1 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.

I) Data Set # 1 : IRIS DATA SET (http://archive.ics.uci.edu/ml/datasets/Iris)

Data Set Description :

- ➔ Title: Iris Plants Database.
- ➔ Predicted attribute: class of iris plant.
- ➔ Number of Instances: 150 (50 in each of three classes).
- ➔ Number of Attributes: 4 numeric, predictive attributes and the class.
- ➔ Class Distribution: 33.3% for each of 3 classes.
- ➔ Attribute Information:
  - sepal length in cm
  - sepal width in cm
  - petal length in cm
  - petal width in cm
  - class:
    - Iris Setosa
    - Iris Versicolour
    - Iris Virginica
- ➔ Missing Attribute Values: None.

II) Data Set # 2 : WINE DATA SET (http://archive.ics.uci.edu/ml/datasets/Wine)

Data Set Description :

- ➔ Title of Database: Wine recognition data
- ➔ Predicted attribute: Name of wines grown in the same region in Italy but derived from three different cultivars.
- ➔ Number of Instances:
  - class 1: 59
  - class 2 : 71

- class 3 : 48
- Number of Attributes: 13. All attributes are continuous. 1st attribute is class identifier (1-3).
- Attribute Information:
  - Alcohal
  - Malic Acid
  - Ash
  - Alcalinity of ash
  - Magnesium
  - Total Phenols
  - Flavanoids
  - Nonflavanoid phenols
  - Proanthocyanins
  - Color Intensity
  - Hue
  - OD 280/ OD 315 of diluted wines
  - Proline
- Missing Attribute Values: None.

III) Data Set # 3 : BREAST CANCER - WISCONSIN DATA SET

Data Set Description :

- Title of Database: Wisconsin Breast Cancer Database (January 8, 1991)
- 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)

<u>The above data samples are taken considering the following things:</u>

-> The attributes are numerical values.

-> Distance between the two classes can be calculated by using the Euclidean Distance only i.e for continuous variables, it can be calculated here.

-> 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 continuous 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.

Q-2 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.

RANDOM SUBSAMPLING:

- ➢ Divides the data into two partitions of equal size i.e. Training Sample and Test Sample.
- ➢ User is asked the choice for it. i.e. Enter 1...for Random Subsampling.
- ➢ After selecting the type of algorithm, the user enters the name of dataset, he wants to work upon, the value of k (1 or 3) and the index of the class i.e. the index of the attribute in the data set.
- ➢ Sample output :

Data Sets - Observation:

Observation of IRIS data set : (k=1)

Sample output obtained for Random Subsampling over 10 iterations is, class index = 4 :

```
ITERATION NO : 1
Accuracy: 93.33333333333333
                                PREDICTED
+--------+-----------------+----------------+-------------+-----------------+
|        |                 | Iris-virginica | Iris-setosa | Iris-versicolor |
+--------+-----------------+----------------+-------------+-----------------+
|        | Iris-virginica  | 21             | 0           | 3               |
+--------+-----------------+----------------+-------------+-----------------+
| ACTUAL | Iris-setosa     | 0              | 27          | 0               |
+--------+-----------------+----------------+-------------+-----------------+
|        | Iris-versicolor | 2              | 0           | 22              |
+--------+-----------------+----------------+-------------+-----------------+
```

```
ITERATION NO : 2
Accuracy: 97.33333333333334
                                PREDICTED
+--------+-----------------+----------------+-------------+-----------------+
|        |                 | Iris-virginica | Iris-setosa | Iris-versicolor |
+--------+-----------------+----------------+-------------+-----------------+
|        | Iris-virginica  | 21             | 0           | 2               |
+--------+-----------------+----------------+-------------+-----------------+
| ACTUAL | Iris-setosa     | 0              | 24          | 0               |
+--------+-----------------+----------------+-------------+-----------------+
|        | Iris-versicolor | 0              | 0           | 28              |
+--------+-----------------+----------------+-------------+-----------------+
```

```
ITERATION NO : 3
Accuracy: 93.33333333333333
                                PREDICTED
+--------+-----------------+----------------+-------------+-----------------+
|        |                 | Iris-virginica | Iris-setosa | Iris-versicolor |
+--------+-----------------+----------------+-------------+-----------------+
|        | Iris-virginica  | 23             | 0           | 5               |
+--------+-----------------+----------------+-------------+-----------------+
| ACTUAL | Iris-setosa     | 0              | 23          | 0               |
+--------+-----------------+----------------+-------------+-----------------+
|        | Iris-versicolor | 0              | 0           | 24              |
+--------+-----------------+----------------+-------------+-----------------+
```

```
ITERATION NO : 4
Accuracy: 97.33333333333334
                                PREDICTED
+--------+-----------------+----------------+-------------+-----------------+
|        |                 | Iris-virginica | Iris-setosa | Iris-versicolor |
+--------+-----------------+----------------+-------------+-----------------+
|        | Iris-virginica  | 24             | 0           | 0               |
+--------+-----------------+----------------+-------------+-----------------+
| ACTUAL | Iris-setosa     | 0              | 27          | 0               |
+--------+-----------------+----------------+-------------+-----------------+
|        | Iris-versicolor | 2              | 0           | 22              |
+--------+-----------------+----------------+-------------+-----------------+
```

```
ITERATION NO : 5
Accuracy: 98.66666666666667
                                PREDICTED
+---------+-----------------+-----------------+-------------+-----------------+
|         |                 | Iris-virginica  | Iris-setosa | Iris-versicolor |
+---------+-----------------+-----------------+-------------+-----------------+
|         | Iris-virginica  | 24              | 0           | 1               |
+---------+-----------------+-----------------+-------------+-----------------+
| ACTUAL  | Iris-setosa     | 0               | 23          | 0               |
+---------+-----------------+-----------------+-------------+-----------------+
|         | Iris-versicolor | 0               | 0           | 27              |
+---------+-----------------+-----------------+-------------+-----------------+
```

```
ITERATION NO : 6
Accuracy: 96.0
                                PREDICTED
+---------+-----------------+-----------------+-------------+-----------------+
|         |                 | Iris-virginica  | Iris-setosa | Iris-versicolor |
+---------+-----------------+-----------------+-------------+-----------------+
|         | Iris-virginica  | 26              | 0           | 3               |
+---------+-----------------+-----------------+-------------+-----------------+
| ACTUAL  | Iris-setosa     | 0               | 26          | 0               |
+---------+-----------------+-----------------+-------------+-----------------+
|         | Iris-versicolor | 0               | 0           | 20              |
+---------+-----------------+-----------------+-------------+-----------------+
```

```
ITERATION NO : 7
Accuracy: 97.33333333333334
                                PREDICTED
+---------+-----------------+-----------------+-------------+-----------------+
|         |                 | Iris-virginica  | Iris-setosa | Iris-versicolor |
+---------+-----------------+-----------------+-------------+-----------------+
|         | Iris-virginica  | 23              | 0           | 1               |
+---------+-----------------+-----------------+-------------+-----------------+
| ACTUAL  | Iris-setosa     | 0               | 20          | 0               |
+---------+-----------------+-----------------+-------------+-----------------+
|         | Iris-versicolor | 1               | 0           | 30              |
+---------+-----------------+-----------------+-------------+-----------------+
```

```
ITERATION NO : 8
Accuracy: 96.0
                                PREDICTED
+--------+-----------------+-----------------+--------------+-----------------+
|        |                 | Iris-virginica  | Iris-setosa  | Iris-versicolor |
+--------+-----------------+-----------------+--------------+-----------------+
|        | Iris-virginica  | 26              | 0            | 2               |
+--------+-----------------+-----------------+--------------+-----------------+
| ACTUAL | Iris-setosa     | 0               | 27           | 0               |
+--------+-----------------+-----------------+--------------+-----------------+
|        | Iris-versicolor | 1               | 0            | 19              |
+--------+-----------------+-----------------+--------------+-----------------+
```

```
ITERATION NO : 9
Accuracy: 93.33333333333333
                                PREDICTED
+--------+-----------------+-----------------+--------------+-----------------+
|        |                 | Iris-virginica  | Iris-setosa  | Iris-versicolor |
+--------+-----------------+-----------------+--------------+-----------------+
|        | Iris-virginica  | 19              | 0            | 2               |
+--------+-----------------+-----------------+--------------+-----------------+
| ACTUAL | Iris-setosa     | 0               | 28           | 0               |
+--------+-----------------+-----------------+--------------+-----------------+
|        | Iris-versicolor | 3               | 0            | 23              |
+--------+-----------------+-----------------+--------------+-----------------+
```

```
ITERATION NO : 10
Accuracy: 96.0
                                PREDICTED
+--------+-----------------+-----------------+--------------+-----------------+
|        |                 | Iris-virginica  | Iris-setosa  | Iris-versicolor |
+--------+-----------------+-----------------+--------------+-----------------+
|        | Iris-virginica  | 23              | 0            | 1               |
+--------+-----------------+-----------------+--------------+-----------------+
| ACTUAL | Iris-setosa     | 0               | 23           | 0               |
+--------+-----------------+-----------------+--------------+-----------------+
|        | Iris-versicolor | 2               | 0            | 26              |
+--------+-----------------+-----------------+--------------+-----------------+
```

```
Mean: 95.86666666666667
Standard Deviation: 1.8330302779823409
```

Observation of IRIS data set : (k=3)

Sample output obtained for Random Subsampling over 10 iterations is, class index = 4 :

```
Enter 1..........Random Sub Sampling
Enter 2..........Five Fold Cross Validation
Enter Choice
1
Enter the source dataset filename: iris.data
Enter the index of the class in the given dataset: 4
Enter the value of k: 3


ITERATION NO : 1
Accuracy: 94.66666666666667
                                PREDICTED
+--------+-----------------+----------------+-------------+-----------------+
|        |                 | Iris-virginica | Iris-setosa | Iris-versicolor |
+--------+-----------------+----------------+-------------+-----------------+
|        | Iris-virginica  | 23             | 0           | 4               |
+--------+-----------------+----------------+-------------+-----------------+
| ACTUAL | Iris-setosa     | 0              | 29          | 0               |
+--------+-----------------+----------------+-------------+-----------------+
|        | Iris-versicolor | 0              | 0           | 19              |
+--------+-----------------+----------------+-------------+-----------------+


ITERATION NO : 2
Accuracy: 97.33333333333334
                                PREDICTED
+--------+-----------------+----------------+-------------+-----------------+
|        |                 | Iris-virginica | Iris-setosa | Iris-versicolor |
+--------+-----------------+----------------+-------------+-----------------+
|        | Iris-virginica  | 22             | 0           | 1               |
+--------+-----------------+----------------+-------------+-----------------+
| ACTUAL | Iris-setosa     | 0              | 27          | 0               |
+--------+-----------------+----------------+-------------+-----------------+
|        | Iris-versicolor | 1              | 0           | 24              |
+--------+-----------------+----------------+-------------+-----------------+
```

```
ITERATION NO : 3
Accuracy: 97.33333333333334
                                PREDICTED
+--------+-----------------+----------------+-------------+-----------------+
|        |                 | Iris-virginica | Iris-setosa | Iris-versicolor |
+--------+-----------------+----------------+-------------+-----------------+
|        | Iris-virginica  | 27             | 0           | 1               |
+--------+-----------------+----------------+-------------+-----------------+
| ACTUAL | Iris-setosa     | 0              | 24          | 0               |
+--------+-----------------+----------------+-------------+-----------------+
|        | Iris-versicolor | 1              | 0           | 22              |
+--------+-----------------+----------------+-------------+-----------------+
```

```
ITERATION NO : 4
Accuracy: 96.0
                                       PREDICTED
+--------+-----------------+----------------+-------------+-----------------+
|        |                 | Iris-virginica | Iris-setosa | Iris-versicolor |
+--------+-----------------+----------------+-------------+-----------------+
|        | Iris-virginica  | 21             | 0           | 1               |
+--------+-----------------+----------------+-------------+-----------------+
| ACTUAL | Iris-setosa     | 0              | 29          | 0               |
+--------+-----------------+----------------+-------------+-----------------+
|        | Iris-versicolor | 2              | 0           | 22              |
+--------+-----------------+----------------+-------------+-----------------+
```

```
ITERATION NO : 5
Accuracy: 96.0
                                       PREDICTED
+--------+-----------------+----------------+-------------+-----------------+
|        |                 | Iris-virginica | Iris-setosa | Iris-versicolor |
+--------+-----------------+----------------+-------------+-----------------+
|        | Iris-virginica  | 23             | 0           | 2               |
+--------+-----------------+----------------+-------------+-----------------+
| ACTUAL | Iris-setosa     | 0              | 23          | 0               |
+--------+-----------------+----------------+-------------+-----------------+
|        | Iris-versicolor | 1              | 0           | 26              |
+--------+-----------------+----------------+-------------+-----------------+
```

```
ITERATION NO : 6
Accuracy: 97.33333333333334
                                       PREDICTED
+--------+-----------------+----------------+-------------+-----------------+
|        |                 | Iris-virginica | Iris-setosa | Iris-versicolor |
+--------+-----------------+----------------+-------------+-----------------+
|        | Iris-virginica  | 24             | 0           | 1               |
+--------+-----------------+----------------+-------------+-----------------+
| ACTUAL | Iris-setosa     | 0              | 21          | 0               |
+--------+-----------------+----------------+-------------+-----------------+
|        | Iris-versicolor | 1              | 0           | 28              |
+--------+-----------------+----------------+-------------+-----------------+
```

```
ITERATION NO : 7
Accuracy: 93.33333333333333
                                       PREDICTED
+--------+-----------------+----------------+-------------+-----------------+
|        |                 | Iris-virginica | Iris-setosa | Iris-versicolor |
+--------+-----------------+----------------+-------------+-----------------+
|        | Iris-virginica  | 20             | 0           | 4               |
+--------+-----------------+----------------+-------------+-----------------+
| ACTUAL | Iris-setosa     | 0              | 27          | 0               |
+--------+-----------------+----------------+-------------+-----------------+
|        | Iris-versicolor | 1              | 0           | 23              |
+--------+-----------------+----------------+-------------+-----------------+
```

```
ITERATION NO : 8
Accuracy: 97.33333333333334
                                PREDICTED
+--------+-----------------+----------------+--------------+-----------------+
|        |                 | Iris-virginica | Iris-setosa  | Iris-versicolor |
+--------+-----------------+----------------+--------------+-----------------+
|        | Iris-virginica  | 23             | 0            | 1               |
+--------+-----------------+----------------+--------------+-----------------+
| ACTUAL | Iris-setosa     | 0              | 33           | 0               |
+--------+-----------------+----------------+--------------+-----------------+
|        | Iris-versicolor | 1              | 0            | 17              |
+--------+-----------------+----------------+--------------+-----------------+
```

```
ITERATION NO : 9
Accuracy: 97.33333333333334
                                PREDICTED
+--------+-----------------+----------------+--------------+-----------------+
|        |                 | Iris-virginica | Iris-setosa  | Iris-versicolor |
+--------+-----------------+----------------+--------------+-----------------+
|        | Iris-virginica  | 25             | 0            | 0               |
+--------+-----------------+----------------+--------------+-----------------+
| ACTUAL | Iris-setosa     | 0              | 26           | 0               |
+--------+-----------------+----------------+--------------+-----------------+
|        | Iris-versicolor | 2              | 0            | 22              |
+--------+-----------------+----------------+--------------+-----------------+
```

```
ITERATION NO : 10
Accuracy: 96.0
                                PREDICTED
+--------+-----------------+----------------+--------------+-----------------+
|        |                 | Iris-virginica | Iris-setosa  | Iris-versicolor |
+--------+-----------------+----------------+--------------+-----------------+
|        | Iris-virginica  | 25             | 0            | 0               |
+--------+-----------------+----------------+--------------+-----------------+
| ACTUAL | Iris-setosa     | 0              | 26           | 0               |
+--------+-----------------+----------------+--------------+-----------------+
|        | Iris-versicolor | 3              | 0            | 21              |
+--------+-----------------+----------------+--------------+-----------------+
```

```
Mean: 96.26666666666668
Standard Deviation: 1.306394529484366
```

Observation of WINE data set : (k=1)

Sample output obtained for Random Subsampling over 10 iterations is, class index = 0 :

```
Enter 1..........Random Sub Sampling
Enter 2..........Five Fold Cross Validation
Enter Choice
1
Enter the source dataset filename: wine.data
Enter the index of the class in the given dataset: 0
Enter the value of k: 1
```

```
ITERATION NO : 1
Accuracy: 65.1685393258427
                  PREDICTED
+--------+---+----+----+----+
|        |   |  1 |  3 |  2 |
+--------+---+----+----+----+
|        | 1 | 27 |  1 |  1 |
+--------+---+----+----+----+
| ACTUAL | 3 |  3 |  9 | 12 |
+--------+---+----+----+----+
|        | 2 |  2 | 12 | 22 |
+--------+---+----+----+----+
```

```
ITERATION NO : 2
Accuracy: 74.15730337078652
                  PREDICTED
+--------+---+----+----+----+
|        |   |  1 |  3 |  2 |
+--------+---+----+----+----+
|        | 1 | 24 |  2 |  2 |
+--------+---+----+----+----+
| ACTUAL | 3 |  3 | 11 |  8 |
+--------+---+----+----+----+
|        | 2 |  3 |  5 | 31 |
+--------+---+----+----+----+
```

```
ITERATION NO : 3
Accuracy: 75.28089887640449
                  PREDICTED
+--------+---+----+----+----+
|        |   |  1 |  3 |  2 |
+--------+---+----+----+----+
|        | 1 | 20 |  2 |  2 |
+--------+---+----+----+----+
| ACTUAL | 3 |  1 | 15 | 11 |
+--------+---+----+----+----+
|        | 2 |  2 |  4 | 32 |
+--------+---+----+----+----+
```

```
ITERATION NO : 4
Accuracy: 70.78651685393258
                  PREDICTED
+--------+---+----+----+----+
|        |   |  1 |  3 |  2 |
+--------+---+----+----+----+
|        | 1 | 28 |  3 |  2 |
+--------+---+----+----+----+
| ACTUAL | 3 |  0 | 11 | 12 |
+--------+---+----+----+----+
|        | 2 |  2 |  7 | 24 |
+--------+---+----+----+----+
```

```
ITERATION NO : 5
Accuracy: 67.41573033707866
                  PREDICTED
+--------+---+----+----+----+
|        |   |  1 |  3 |  2 |
+--------+---+----+----+----+
|        | 1 | 22 |  0 |  3 |
+--------+---+----+----+----+
| ACTUAL | 3 |  3 | 12 | 16 |
+--------+---+----+----+----+
|        | 2 |  1 |  6 | 26 |
+--------+---+----+----+----+
```

```
ITERATION NO : 6
Accuracy: 75.28089887640449
                  PREDICTED
+--------+---+----+----+----+
|        |   |  1 |  3 |  2 |
+--------+---+----+----+----+
|        | 1 | 33 |  3 |  1 |
+--------+---+----+----+----+
| ACTUAL | 3 |  1 | 12 |  8 |
+--------+---+----+----+----+
|        | 2 |  2 |  7 | 22 |
+--------+---+----+----+----+
```

```
ITERATION NO : 7
Accuracy: 62.92134831460674
                  PREDICTED
+--------+---+----+----+----+
|        |   |  1 |  3 |  2 |
+--------+---+----+----+----+
|        | 1 | 21 |  1 |  0 |
+--------+---+----+----+----+
| ACTUAL | 3 |  7 | 13 |  7 |
+--------+---+----+----+----+
|        | 2 |  9 |  9 | 22 |
+--------+---+----+----+----+
```

```
ITERATION NO : 8
Accuracy: 79.7752808988764
                  PREDICTED
+--------+---+----+----+----+
|        |   |  1 |  3 |  2 |
+--------+---+----+----+----+
|        | 1 | 28 |  2 |  0 |
+--------+---+----+----+----+
| ACTUAL | 3 |  0 | 13 |  9 |
+--------+---+----+----+----+
|        | 2 |  2 |  5 | 30 |
+--------+---+----+----+----+
```

```
ITERATION NO : 9
Accuracy: 75.28089887640449
                  PREDICTED
+--------+---+----+----+----+
|        |   |  1 |  3 |  2 |
+--------+---+----+----+----+
|        | 1 | 31 |  3 |  1 |
+--------+---+----+----+----+
| ACTUAL | 3 |  1 | 17 |  2 |
+--------+---+----+----+----+
|        | 2 |  1 | 14 | 19 |
+--------+---+----+----+----+
```

```
ITERATION NO : 10
Accuracy: 73.03370786516854
                PREDICTED
+--------+---+----+----+----+
|        |   |  1 |  3 |  2 |
+--------+---+----+----+----+
|        | 1 | 31 |  3 |  4 |
+--------+---+----+----+----+
| ACTUAL | 3 |  2 | 14 |  8 |
+--------+---+----+----+----+
|        | 2 |  0 |  7 | 20 |
+--------+---+----+----+----+
```

```
Mean: 71.91011235955057
Standard Deviation: 4.999684373322918
```

<u>Observation of WINE data set : (k=3)</u>

Sample output obtained for Random Subsampling over 10 iterations is, class index = 0 :

```
Enter 1...........Random Sub Sampling
Enter 2...........Five Fold Cross Validation
Enter Choice
1
Enter the source dataset filename: wine.data
Enter the index of the class in the given dataset: 0
Enter the value of k: 3
```

```
ITERATION NO : 1
Accuracy: 75.28089887640449
                PREDICTED
+--------+---+----+----+----+
|        |   |  1 |  3 |  2 |
+--------+---+----+----+----+
|        | 1 | 28 |  7 |  0 |
+--------+---+----+----+----+
| ACTUAL | 3 |  2 | 11 |  4 |
+--------+---+----+----+----+
|        | 2 |  4 |  5 | 28 |
+--------+---+----+----+----+
```

```
ITERATION NO : 2
Accuracy: 62.92134831460674
                PREDICTED
+--------+---+----+----+----+
|        |   |  1 |  3 |  2 |
+--------+---+----+----+----+
|        | 1 | 24 |  4 |  3 |
+--------+---+----+----+----+
| ACTUAL | 3 |  2 | 14 |  9 |
+--------+---+----+----+----+
|        | 2 |  2 | 13 | 18 |
+--------+---+----+----+----+
```

```
ITERATION NO : 3
Accuracy: 71.91011235955057
                PREDICTED
+--------+---+----+----+----+
|        |   |  1 |  3 |  2 |
+--------+---+----+----+----+
|        | 1 | 27 |  3 |  1 |
+--------+---+----+----+----+
| ACTUAL | 3 |  1 | 10 |  9 |
+--------+---+----+----+----+
|        | 2 |  4 |  7 | 27 |
+--------+---+----+----+----+
```

```
ITERATION NO : 4
Accuracy: 68.53932584269663
                         PREDICTED
+--------+---+----+---+----+
|        |   |  1 | 3 |  2 |
+--------+---+----+---+----+
|        | 1 | 22 | 0 |  0 |
+--------+---+----+---+----+
| ACTUAL | 3 |  9 | 9 | 11 |
+--------+---+----+---+----+
|        | 2 |  4 | 4 | 30 |
+--------+---+----+---+----+
```

```
ITERATION NO : 5
Accuracy: 67.41573033707866
                         PREDICTED
+--------+---+----+---+----+
|        |   |  1 | 3 |  2 |
+--------+---+----+---+----+
|        | 1 | 26 | 3 |  0 |
+--------+---+----+---+----+
| ACTUAL | 3 |  3 | 8 | 11 |
+--------+---+----+---+----+
|        | 2 |  6 | 6 | 26 |
+--------+---+----+---+----+
```

```
ITERATION NO : 6
Accuracy: 65.1685393258427
                         PREDICTED
+--------+---+----+----+----+
|        |   |  1 |  3 |  2 |
+--------+---+----+----+----+
|        | 1 | 28 |  1 |  2 |
+--------+---+----+----+----+
| ACTUAL | 3 |  6 | 13 |  8 |
+--------+---+----+----+----+
|        | 2 |  2 | 12 | 17 |
+--------+---+----+----+----+
```

```
ITERATION NO : 7
Accuracy: 73.03370786516854
                         PREDICTED
+--------+---+----+----+----+
|        |   |  1 |  3 |  2 |
+--------+---+----+----+----+
|        | 1 | 32 |  1 |  0 |
+--------+---+----+----+----+
| ACTUAL | 3 |  5 | 11 |  3 |
+--------+---+----+----+----+
|        | 2 |  4 | 11 | 22 |
+--------+---+----+----+----+
```

```
ITERATION NO : 8
Accuracy: 73.03370786516854
                         PREDICTED
+--------+---+----+----+----+
|        |   |  1 |  3 |  2 |
+--------+---+----+----+----+
|        | 1 | 29 |  0 |  1 |
+--------+---+----+----+----+
| ACTUAL | 3 |  2 | 13 | 10 |
+--------+---+----+----+----+
|        | 2 |  2 |  9 | 23 |
+--------+---+----+----+----+
```

```
ITERATION NO : 9
Accuracy: 74.15730337078652
                         PREDICTED
+--------+---+----+----+----+
|        |   |  1 |  3 |  2 |
+--------+---+----+----+----+
|        | 1 | 25 |  0 |  1 |
+--------+---+----+----+----+
| ACTUAL | 3 |  4 | 14 | 10 |
+--------+---+----+----+----+
|        | 2 |  2 |  6 | 27 |
+--------+---+----+----+----+
```

```
ITERATION NO : 10
Accuracy: 71.91011235955057
                         PREDICTED
+--------+---+----+----+----+
|        |   |  1 |  3 |  2 |
+--------+---+----+----+----+
|        | 1 | 23 |  2 |  0 |
+--------+---+----+----+----+
| ACTUAL | 3 |  4 | 14 | 12 |
+--------+---+----+----+----+
|        | 2 |  3 |  4 | 27 |
+--------+---+----+----+----+
```

```
Mean: 70.33707865168539
Standard Deviation: 3.8987306905387578
```

Observation of WISCONSIN data set : (k=1)

Sample output obtained for Random Subsampling over 10 iterations is, class index = 10:

ITERATION NO : 1
Accuracy: 96.28571428571429
PREDICTED

		Benign	Malignant
	Benign	233	7
ACTUAL	Malignant	6	104

ITERATION NO : 2
Accuracy: 94.85714285714286
PREDICTED

		Benign	Malignant
	Benign	227	7
ACTUAL	Malignant	11	105

ITERATION NO : 3
Accuracy: 95.14285714285714
PREDICTED

		Benign	Malignant
	Benign	226	4
ACTUAL	Malignant	13	107

ITERATION NO : 4
Accuracy: 94.57142857142857
PREDICTED

		Benign	Malignant
	Benign	216	10
ACTUAL	Malignant	9	115

ITERATION NO : 5
Accuracy: 94.57142857142857
PREDICTED

		Benign	Malignant
	Benign	217	7
ACTUAL	Malignant	12	114

ITERATION NO : 6
Accuracy: 94.85714285714286
PREDICTED

		Benign	Malignant
	Benign	221	6
ACTUAL	Malignant	12	111

ITERATION NO : 7
Accuracy: 96.28571428571429
PREDICTED

		Benign	Malignant
	Benign	228	5
ACTUAL	Malignant	8	109

ITERATION NO : 8
Accuracy: 94.85714285714286
PREDICTED

		Benign	Malignant
	Benign	230	8
ACTUAL	Malignant	10	102

ITERATION NO : 9
Accuracy: 96.28571428571429
PREDICTED

		Benign	Malignant
	Benign	222	4
ACTUAL	Malignant	9	115

ITERATION NO : 10
Accuracy: 94.57142857142857
PREDICTED

		Benign	Malignant
	Benign	219	6
ACTUAL	Malignant	13	112

Mean: 95.22857142857144
Standard Deviation: 0.7119963311072662

Observation of WISCONSIN data set : (k=3)

Sample output obtained for Random Subsampling over 10 iterations is, class index = 10 :

```
ITERATION NO : 1
Accuracy: 95.14285714285714
                 PREDICTED
+--------+-----------+--------+-----------+
|        |           | Benign | Malignant |
+--------+-----------+--------+-----------+
|        | Benign    | 221    | 10        |
+--------+-----------+--------+-----------+
| ACTUAL | Malignant | 7      | 112       |
+--------+-----------+--------+-----------+


ITERATION NO : 2
Accuracy: 96.28571428571429
                 PREDICTED
+--------+-----------+--------+-----------+
|        |           | Benign | Malignant |
+--------+-----------+--------+-----------+
|        | Benign    | 217    | 3         |
+--------+-----------+--------+-----------+
| ACTUAL | Malignant | 10     | 120       |
+--------+-----------+--------+-----------+


ITERATION NO : 3
Accuracy: 96.28571428571429
                 PREDICTED
+--------+-----------+--------+-----------+
|        |           | Benign | Malignant |
+--------+-----------+--------+-----------+
|        | Benign    | 209    | 8         |
+--------+-----------+--------+-----------+
| ACTUAL | Malignant | 5      | 128       |
+--------+-----------+--------+-----------+
```

```
ITERATION NO : 4
Accuracy: 95.42857142857143
                 PREDICTED
+--------+-----------+--------+-----------+
|        |           | Benign | Malignant |
+--------+-----------+--------+-----------+
|        | Benign    | 223    | 11        |
+--------+-----------+--------+-----------+
| ACTUAL | Malignant | 5      | 111       |
+--------+-----------+--------+-----------+


ITERATION NO : 5
Accuracy: 94.85714285714286
                 PREDICTED
+--------+-----------+--------+-----------+
|        |           | Benign | Malignant |
+--------+-----------+--------+-----------+
|        | Benign    | 209    | 8         |
+--------+-----------+--------+-----------+
| ACTUAL | Malignant | 10     | 123       |
+--------+-----------+--------+-----------+


ITERATION NO : 6
Accuracy: 95.71428571428572
                 PREDICTED
+--------+-----------+--------+-----------+
|        |           | Benign | Malignant |
+--------+-----------+--------+-----------+
|        | Benign    | 216    | 7         |
+--------+-----------+--------+-----------+
| ACTUAL | Malignant | 8      | 119       |
+--------+-----------+--------+-----------+
```

```
ITERATION NO : 7
Accuracy: 96.0
                 PREDICTED
+--------+-----------+--------+-----------+
|        |           | Benign | Malignant |
+--------+-----------+--------+-----------+
|        | Benign    | 219    | 3         |
+--------+-----------+--------+-----------+
| ACTUAL | Malignant | 11     | 117       |
+--------+-----------+--------+-----------+


ITERATION NO : 8
Accuracy: 95.71428571428572
                 PREDICTED
+--------+-----------+--------+-----------+
|        |           | Benign | Malignant |
+--------+-----------+--------+-----------+
|        | Benign    | 220    | 7         |
+--------+-----------+--------+-----------+
| ACTUAL | Malignant | 8      | 115       |
+--------+-----------+--------+-----------+


ITERATION NO : 9
Accuracy: 95.14285714285714
                 PREDICTED
+--------+-----------+--------+-----------+
|        |           | Benign | Malignant |
+--------+-----------+--------+-----------+
|        | Benign    | 223    | 8         |
+--------+-----------+--------+-----------+
| ACTUAL | Malignant | 9      | 110       |
+--------+-----------+--------+-----------+
```

```
ITERATION NO : 10
Accuracy: 95.14285714285714
                 PREDICTED
+--------+-----------+--------+-----------+
|        |           | Benign | Malignant |
+--------+-----------+--------+-----------+
|        | Benign    | 230    | 3         |
+--------+-----------+--------+-----------+
| ACTUAL | Malignant | 14     | 103       |
+--------+-----------+--------+-----------+
```

```
Mean: 95.57142857142858
Standard Deviation: 0.4823412290324065
```

CODE:

There is a general code for both “Random Subsampling” and “Five Fold Cross Validation”. : User can select the two types of algorithm as per given choice :

```
import random
from matplotlib.pyplot import *
import csv
import math
import operator
from tabulate import tabulate


# Function to load the dataset and create training sample and test sample
# Depending on the split ratio given : Random Subsampling
def read_file_load_dataset_randomsubsampling(filename, index_of_class,
training_sample=[], test_sample=[]):
    f = open(filename, 'rb')
    dataset = f.readlines()
    length = len(dataset)
    random.shuffle(dataset)

    test=[]
    #Spliting the dataset into training sample : list of lists
    for i in range(0, length/2):
        test = dataset[i].split(',')
        for i in range(0, len(test)):
            if filename=='wisconsin.data':
                if i==0:
                    test[i]=0
            if i!=index_of_class:
                if test[i]=='?':
                    test[i]=5
                test[i] = float(test[i])

        training_sample.append(test)
        test=[]

    #Spliting the dataset into test sample - with classname(for simplicity): list of lists
```

```
    for i in range((length/2), length):
        test = dataset[i].split(',')
        for i in range(0, len(test)):
            if filename=='wisconsin.data':
                if i==0:
                    test[i]=0
            if i!=index_of_class:
                if test[i]=='?':
                    test[i]=5
                test[i] = float(test[i])
        test_sample.append(test)
        test=[]

    return dataset

# Function to load the dataset and create training sample and test sample
# Depending on the split ratio given : Randomsubsampling
def read_file_load_dataset_fivefoldcrossvalid(begin, end, filename, index_of_class,
training_sample=[], test_sample=[]):
    f = open(filename, 'rb')
    dataset = f.readlines()
    length = len(dataset)
    random.shuffle(dataset)
    test=[]
    begin_iter_test = int(begin * length)
    end_iter_test = int(end * length)
    begin_iter_training = int(begin * length)
    end_iter_training = int(end * length)

    #Spliting the dataset into training sample : list of lists
    if begin_iter_test==0:
        for i in range(end_iter_test, length):
            test = dataset[i].split(',')
            for i in range(0, len(test)):
                if filename=='wisconsin.data':
                    if i==0:
                        test[i]=0
                if i!=index_of_class:
                    if test[i]=='?':
                        test[i]=5
                    test[i] = float(test[i])

            training_sample.append(test)
            test=[]
```

```
else:
    for i in range(0, begin_iter_test):
        test = dataset[i].split(',')
        for i in range(0, len(test)):
            if filename=='wisconsin.data':
                if i==0:
                    test[i]=0
            if i!=index_of_class:
                if test[i]=='?':
                    test[i]=5
                test[i] = float(test[i])

        training_sample.append(test)
        test=[]
    for i in range(end_iter_test, length):
        test = dataset[i].split(',')
        for i in range(0, len(test)):
            if filename=='wisconsin.data':
                if i==0:
                    test[i]=0
            if i!=index_of_class:
                if test[i]=='?':
                    test[i]=5
                test[i] = float(test[i])

        training_sample.append(test)
        test=[]


#Spliting the dataset into test sample - with classname(for simplicity): list of lists
for i in range(begin_iter_test, end_iter_test):
    test = dataset[i].split(',')
    for i in range(0, len(test)):
        if filename=='wisconsin.data':
            if i==0:
                test[i]=0
        if i!=index_of_class:
            if test[i]=='?':
                test[i]=5
            test[i] = float(test[i])
    test_sample.append(test)
    test=[]

return dataset
```

```
#Fuction to calculate euldiean distance between test instance and each training instance
def calculate_euclidean_distance(variable1, variable2, dimension, index_of_class):
    distance = 0
    for x in range(dimension):
        if x!=index_of_class:
            distance = distance + pow((variable1[x] - variable2[x]), 2)
    return math.sqrt(distance)


def find_k_nearest_neighbours(training_sample, index_of_class, test_instance, k):

    length_training_sample = len(training_sample)
    distances=[]
    for i in range(length_training_sample):
        d = calculate_euclidean_distance(test_instance, training_sample[i],
len(test_instance), index_of_class)
        temp = training_sample[i]
        distances.append((temp, d))

    neighbours = []
    distances.sort(key=operator.itemgetter(1))
    for i in range(k):
        x = distances[i][0]
        neighbours.append(x)
    return neighbours


#Devise prediction class on the basis of the neighbours obtained
def getprediction(neighbours, index_of_class):
    length_of_neighbours = len(neighbours)
    class_votes_dictionary = {}
    for i in range(length_of_neighbours):
        predicted_class = neighbours[i][index_of_class]
        if predicted_class in class_votes_dictionary:
            x = predicted_class
            class_votes_dictionary[x]+=1
        else:
            x = predicted_class
            class_votes_dictionary[x]=1

    sorted_class_votes=[]
    sorted_class_votes = sorted(class_votes_dictionary.iteritems(),
key=operator.itemgetter(1), reverse=True)
```

```
    return sorted_class_votes[0][0]


#Calculate the ratio of the total correct predictions out of all predictions made :
classification accuracy
def calculate_accuracy(test_verify_sample, predictions, index_of_class):
    correct=0
    length_test_sample = len(test_verify_sample)
    for i in range(length_test_sample):
        if test_verify_sample[i][index_of_class] == predictions[i]:
            correct = correct+1
    accuracy_percentage = (correct/(float(length_test_sample))) * 100
    return accuracy_percentage


#Calculating mean of the accuracy percentage
def calculate_mean(accuracy_percentage_list):
    sum=0
    for i in range(len(accuracy_percentage_list)):
        sum=sum+float(accuracy_percentage_list[i])
    mean = sum/len(accuracy_percentage_list)
    return mean


#Calculating standard deviation of the accuracy percentage
def calculate_standard_deviation(accuracy_percentage_list, mean):
    diff_sq = 0
    for i in range(len(accuracy_percentage_list)):
        diff_sq = diff_sq + pow((accuracy_percentage_list[i] - mean), 2)

    diff_sq = diff_sq/len(accuracy_percentage_list)
    return math.sqrt(diff_sq)


def confusion_matrix(filename, predictions, actual_classes):
    #Fetching the name of the classes to dictionary and then to the list
    classes={}
    for i in range(len(actual_classes)):
        if actual_classes[i][(len(actual_classes[i]))-1] == '\n':
                actual_classes[i] = actual_classes[i][0:len(actual_classes[i])-1]
                predictions[i] = predictions[i][0:len(predictions[i])-1]
        if actual_classes[i] in classes:
            classes[actual_classes[i]]= 1
        else:
```

```
                        classes[actual_classes[i]]= 1
    c =[]
    for i in classes.keys():
            c.append(i)
    length = len(c)


    #Creating confusion matrix as list -> empty list and hence comparing and increasing
the count
    confusion_matrix=[]
    for i in range(length):
            for j in range(length):
                        confusion_matrix.append(0)

    count = 0
    for i in range(len(actual_classes)):
            for j in range(length):
                        for k in range(length):
                                    if actual_classes[i] == c[j] and predictions[i] == c[k]:
                                                count = count +1
                                                confusion_matrix[j*length+k] =
confusion_matrix[j*length+k]+1

    #Printing confusion matrix
    if filename == 'wisconsin.data':
            for i in range(length):
                        if c[i] == '2':
                                    c[i] = 'Benign'
                        if c[i]=='4':
                                    c[i] ='Malignant'
    print "\t\t"+'PREDICTED'
    table = []

    #Append Classes name
    L=[]
    L.append('\t')
    L.append('\t')
    for i in range(length):
            L.append(c[i])
    table.append(L)

    #Create Empty Table
    L=[]
    for i in range(length):
```

```
        for j in range(length+2):
            if i==length/2:
                if j==0:
                    L.append('ACTUAL')
                elif j==1:
                    L.append(c[i])
                else:
                    L.append('\t')
            else:
                if j==1:
                    L.append(c[i])
                else:
                    L.append('\t')
        table.append(L)
        L=[]

    #Populate value to the confusion matrix/empty table
    value_index=0
    for i in range(1, length+1):
        for j in range(2, length+2):
            table[i][j] = confusion_matrix[value_index]
            value_index+=1

    print tabulate(table, tablefmt="grid")



if __name__ == '__main__':

    #Taking inputs
    filename = raw_input('Enter the source dataset filename: ')
    index_of_class = input('Enter the index of the class in the given dataset: ')
    k = input('Enter the value of k: ')
    print "Enter 1...........Random Sub Sampling"
    print "Enter 2...........Five Fold Cross Validation"
    choice = input("Enter Choice\n")

    if choice == 1:
        accuracy_percentage_list=[]
        for x in range(10):
            #Initiliasing Variables
            training_sample = []
            test_sample=[]
```

```
                dataset=[]
                dataset = read_file_load_dataset_randomsubsampling(filename,
index_of_class, training_sample, test_sample)
                predictions = []
                for i in range(len(test_sample)):
                        neighbours = find_k_nearest_neighbours(training_sample,
index_of_class, test_sample[i], k)
                        predicted_output = getprediction(neighbours, index_of_class)
                        predictions.append(predicted_output)


                #print predictions
                actual_classes = []
                for i in range(len(test_sample)):
                        actual_classes.append(test_sample[i][index_of_class])

                accuracy_percentage = calculate_accuracy(test_sample, predictions,
index_of_class)
                print '\n'
                print 'ITERATION NO : ' + repr(x+1)
                print 'Accuracy: ' + repr(accuracy_percentage)
                accuracy_percentage_list.append(accuracy_percentage)
                confusion_matrix(filename, predictions, actual_classes)


        print accuracy_percentage_list
        mean = calculate_mean(accuracy_percentage_list)
        sd = calculate_standard_deviation(accuracy_percentage_list, mean)
        print 'Mean: ' + repr(mean)
        print 'Standard Deviation: ' + repr(sd)

  elif choice == 2:
        mean_list = []
        for x in range(10):
                accuracy_percentage_list=[]
                print '\n'
                print 'ITERATION NO : ' + repr(x+1)
                split_ratio = 0.0
                for y in range(5):
                        #Initiliasing Variables
                        training_sample = []
                        test_sample=[]
                        dataset=[]
```

```
                dataset = read_file_load_dataset_fivefoldcrossvalid(split_ratio,
split_ratio+0.2, filename, index_of_class, training_sample, test_sample)
                split_ratio+=0.2
                predictions = []
                for i in range(len(test_sample)):
                    neighbours = find_k_nearest_neighbours(training_sample,
index_of_class, test_sample[i], k)
                    predicted_output = getprediction(neighbours, index_of_class)
                    predictions.append(predicted_output)


                #print predictions
                actual_classes = []
                for i in range(len(test_sample)):
                    actual_classes.append(test_sample[i][index_of_class])

                accuracy_percentage = calculate_accuracy(test_sample, predictions,
index_of_class)
                print 'FOLD #' + repr(y+1)
                print 'Accuracy: ' + repr(accuracy_percentage)
                accuracy_percentage_list.append(accuracy_percentage)
                #confusion_matrix(filename, predictions, actual_classes)


            mean = calculate_mean(accuracy_percentage_list)
            sd = calculate_standard_deviation(accuracy_percentage_list, mean)
            mean_list.append(mean)
            print '---------------------------------------------'
            print 'Mean: ' + repr(mean)
            print 'Standard Deviation: ' + repr(sd)
            print '---------------------------------------------'


        print
        print '=================================================='
        grand_mean = calculate_mean(mean_list)
        grand_sd = calculate_standard_deviation(mean_list, grand_mean)
        print 'Grand Mean: ' + repr(grand_mean)
        print 'Grand Standard Deviation: ' + repr(grand_sd)
        print '=================================================='

    else:
        print "Wrong Choice Entered!"
```

K-FOLD CROSS VALIDATION:

- ➢ Divides the data into five folds (20% test data and 80% training data) each in consecutive five folds respectively.
- ➢ User is asked the choice for it. i.e. Enter 2...Five Fold Cross Validation
- ➢ After selecting the type of algorithm, the user enters the name of dataset, he wants to work upon, the value of k (1 or 3) and the index of the class i.e. the index of the attribute in the data set.
- ➢ Sample output :

1) Observation of IRIS data set : (k=1)

Sample output obtained for K-FOLD CROSS VALIDATION over 10 iterations:

```
ITERATION NO : 1
FOLD #1
Accuracy: 93.33333333333333
FOLD #2
Accuracy: 93.33333333333333
FOLD #3
Accuracy: 86.66666666666667
FOLD #4
Accuracy: 93.33333333333333
FOLD #5
Accuracy: 93.33333333333333
---------------------------------------------
Mean: 91.99999999999999
Standard Deviation: 2.666666666666663
---------------------------------------------


ITERATION NO : 2
FOLD #1
Accuracy: 100.0
FOLD #2
Accuracy: 96.66666666666667
FOLD #3
Accuracy: 100.0
FOLD #4
```

```
Accuracy: 100.0
FOLD #5
Accuracy: 93.33333333333333
--------------------------------------------
Mean: 98.0
Standard Deviation: 2.666666666666668
--------------------------------------------


ITERATION NO : 3
FOLD #1
Accuracy: 96.66666666666667
FOLD #2
Accuracy: 100.0
FOLD #3
Accuracy: 96.66666666666667
FOLD #4
Accuracy: 96.66666666666667
FOLD #5
Accuracy: 100.0
--------------------------------------------
Mean: 98.00000000000001
Standard Deviation: 1.6329931618554498
--------------------------------------------


ITERATION NO : 4
FOLD #1
Accuracy: 96.66666666666667
FOLD #2
Accuracy: 96.66666666666667
FOLD #3
Accuracy: 96.66666666666667
FOLD #4
Accuracy: 100.0
FOLD #5
Accuracy: 96.66666666666667
--------------------------------------------
Mean: 97.33333333333334
Standard Deviation: 1.3333333333333315
--------------------------------------------


ITERATION NO : 5
```

FOLD #1
Accuracy: 96.66666666666667
FOLD #2
Accuracy: 93.33333333333333
FOLD #3
Accuracy: 93.33333333333333
FOLD #4
Accuracy: 93.33333333333333
FOLD #5
Accuracy: 90.0

Mean: 93.33333333333333
Standard Deviation: 2.108185106778921

ITERATION NO : 6
FOLD #1
Accuracy: 100.0
FOLD #2
Accuracy: 93.33333333333333
FOLD #3
Accuracy: 100.0
FOLD #4
Accuracy: 93.33333333333333
FOLD #5
Accuracy: 96.66666666666667

Mean: 96.66666666666666
Standard Deviation: 2.981423969999722

ITERATION NO : 7
FOLD #1
Accuracy: 96.66666666666667
FOLD #2
Accuracy: 100.0
FOLD #3
Accuracy: 96.66666666666667
FOLD #4
Accuracy: 96.66666666666667
FOLD #5
Accuracy: 100.0

Mean: 98.00000000000001
Standard Deviation: 1.6329931618554498

ITERATION NO : 8
FOLD #1
Accuracy: 100.0
FOLD #2
Accuracy: 96.66666666666667
FOLD #3
Accuracy: 93.33333333333333
FOLD #4
Accuracy: 100.0
FOLD #5
Accuracy: 90.0

Mean: 96.0
Standard Deviation: 3.887301263230201

ITERATION NO : 9
FOLD #1
Accuracy: 100.0
FOLD #2
Accuracy: 96.66666666666667
FOLD #3
Accuracy: 96.66666666666667
FOLD #4
Accuracy: 93.33333333333333
FOLD #5
Accuracy: 90.0

Mean: 95.33333333333334
Standard Deviation: 3.3993463423951913

ITERATION NO : 10
FOLD #1
Accuracy: 96.66666666666667
FOLD #2

Accuracy: 93.33333333333333
FOLD #3
Accuracy: 96.66666666666667
FOLD #4
Accuracy: 100.0
FOLD #5
Accuracy: 96.66666666666667

Mean: 96.66666666666667
Standard Deviation: 2.108185106778921

==
Grand Mean: 96.13333333333334
Grand Standard Deviation: 1.9504985117770455
==

PREDICTED

		Iris-virginica	Iris-setosa	Iris-versicolor
	Iris-virginica	7	0	0
ACTUAL	Iris-setosa	0	10	0
	Iris-versicolor	1	0	12

2) Observation of IRIS data set : (k=3)

Sample output obtained for K-FOLD CROSS VALIDATION over 10 iterations:

ITERATION NO : 1
FOLD #1
Accuracy: 96.66666666666667
FOLD #2
Accuracy: 93.33333333333333
FOLD #3
Accuracy: 96.66666666666667
FOLD #4
Accuracy: 93.33333333333333
FOLD #5

Accuracy: 96.66666666666667

Mean: 95.33333333333334
Standard Deviation: 1.6329931618554567

ITERATION NO : 2
FOLD #1
Accuracy: 93.33333333333333
FOLD #2
Accuracy: 96.66666666666667
FOLD #3
Accuracy: 96.66666666666667
FOLD #4
Accuracy: 90.0
FOLD #5
Accuracy: 96.66666666666667

Mean: 94.66666666666667
Standard Deviation: 2.6666666666666696

ITERATION NO : 3
FOLD #1
Accuracy: 96.66666666666667
FOLD #2
Accuracy: 100.0
FOLD #3
Accuracy: 96.66666666666667
FOLD #4
Accuracy: 96.66666666666667
FOLD #5
Accuracy: 96.66666666666667

Mean: 97.33333333333334
Standard Deviation: 1.3333333333333315

ITERATION NO : 4
FOLD #1
Accuracy: 96.66666666666667

FOLD #2
Accuracy: 86.66666666666667
FOLD #3
Accuracy: 96.66666666666667
FOLD #4
Accuracy: 93.33333333333333
FOLD #5
Accuracy: 96.66666666666667

Mean: 94.0
Standard Deviation: 3.887301263230201

ITERATION NO : 5
FOLD #1
Accuracy: 100.0
FOLD #2
Accuracy: 93.33333333333333
FOLD #3
Accuracy: 100.0
FOLD #4
Accuracy: 100.0
FOLD #5
Accuracy: 96.66666666666667

Mean: 98.0
Standard Deviation: 2.666666666666668

ITERATION NO : 6
FOLD #1
Accuracy: 100.0
FOLD #2
Accuracy: 96.66666666666667
FOLD #3
Accuracy: 96.66666666666667
FOLD #4
Accuracy: 100.0
FOLD #5
Accuracy: 90.0

Mean: 96.66666666666667

Standard Deviation: 3.651483716701107

ITERATION NO : 7
FOLD #1
Accuracy: 96.66666666666667
FOLD #2
Accuracy: 96.66666666666667
FOLD #3
Accuracy: 96.66666666666667
FOLD #4
Accuracy: 93.33333333333333
FOLD #5
Accuracy: 93.33333333333333

Mean: 95.33333333333333
Standard Deviation: 1.6329931618554567

ITERATION NO : 8
FOLD #1
Accuracy: 96.66666666666667
FOLD #2
Accuracy: 93.33333333333333
FOLD #3
Accuracy: 100.0
FOLD #4
Accuracy: 96.66666666666667
FOLD #5
Accuracy: 100.0

Mean: 97.33333333333334
Standard Deviation: 2.494438257849295

ITERATION NO : 9
FOLD #1
Accuracy: 96.66666666666667
FOLD #2
Accuracy: 100.0
FOLD #3

Accuracy: 96.66666666666667
FOLD #4
Accuracy: 90.0
FOLD #5
Accuracy: 96.66666666666667

Mean: 96.00000000000001
Standard Deviation: 3.2659863237109046

ITERATION NO : 10
FOLD #1
Accuracy: 93.33333333333333
FOLD #2
Accuracy: 100.0
FOLD #3
Accuracy: 96.66666666666667
FOLD #4
Accuracy: 100.0
FOLD #5
Accuracy: 90.0

Mean: 96.0
Standard Deviation: 3.887301263230201

==
Grand Mean: 96.06666666666668
Grand Standard Deviation: 1.2092238098144714
==

PREDICTED

		Iris-virginica	Iris-setosa	Iris-versicolor
	Iris-virginica	7	0	2
ACTUAL	Iris-setosa	0	6	0
	Iris-versicolor	1	0	14

3) Observation of WINE data set : (k=1)

Sample output obtained for K-FOLD CROSS VALIDATION over 10 iterations:

ITERATION NO : 1
FOLD #1
Accuracy: 60.0
FOLD #2
Accuracy: 69.44444444444444
FOLD #3
Accuracy: 71.42857142857143
FOLD #4
Accuracy: 75.0
FOLD #5
Accuracy: 69.44444444444444

Mean: 69.06349206349208
Standard Deviation: 4.965337142660831

ITERATION NO : 2
FOLD #1
Accuracy: 80.0
FOLD #2
Accuracy: 77.77777777777779
FOLD #3
Accuracy: 68.57142857142857
FOLD #4
Accuracy: 72.22222222222221
FOLD #5
Accuracy: 75.0

Mean: 74.71428571428571
Standard Deviation: 4.033620361719009

ITERATION NO : 3
FOLD #1
Accuracy: 80.0
FOLD #2

Accuracy: 77.77777777777779
FOLD #3
Accuracy: 77.14285714285715
FOLD #4
Accuracy: 66.66666666666666
FOLD #5
Accuracy: 72.22222222222221

Mean: 74.76190476190476
Standard Deviation: 4.778805985443489

ITERATION NO : 4
FOLD #1
Accuracy: 65.71428571428571
FOLD #2
Accuracy: 69.44444444444444
FOLD #3
Accuracy: 62.857142857142854
FOLD #4
Accuracy: 77.77777777777779
FOLD #5
Accuracy: 63.888888888888886

Mean: 67.93650793650792
Standard Deviation: 5.40825121032142

ITERATION NO : 5
FOLD #1
Accuracy: 80.0
FOLD #2
Accuracy: 75.0
FOLD #3
Accuracy: 74.28571428571429
FOLD #4
Accuracy: 72.22222222222221
FOLD #5
Accuracy: 72.22222222222221

Mean: 74.74603174603175
Standard Deviation: 2.8502562508289113

ITERATION NO : 6
FOLD #1
Accuracy: 80.0
FOLD #2
Accuracy: 75.0
FOLD #3
Accuracy: 65.71428571428571
FOLD #4
Accuracy: 86.11111111111111
FOLD #5
Accuracy: 80.55555555555556

Mean: 77.47619047619048
Standard Deviation: 6.854018982733525

ITERATION NO : 7
FOLD #1
Accuracy: 77.14285714285715
FOLD #2
Accuracy: 77.77777777777779
FOLD #3
Accuracy: 68.57142857142857
FOLD #4
Accuracy: 80.55555555555556
FOLD #5
Accuracy: 69.44444444444444

Mean: 74.6984126984127
Standard Deviation: 4.793913584723651

ITERATION NO : 8
FOLD #1
Accuracy: 80.0
FOLD #2
Accuracy: 75.0
FOLD #3
Accuracy: 74.28571428571429

FOLD #4
Accuracy: 52.77777777777778
FOLD #5
Accuracy: 80.55555555555556

Mean: 72.52380952380952
Standard Deviation: 10.193566262446543

ITERATION NO : 9
FOLD #1
Accuracy: 80.0
FOLD #2
Accuracy: 77.77777777777779
FOLD #3
Accuracy: 85.71428571428571
FOLD #4
Accuracy: 69.44444444444444
FOLD #5
Accuracy: 80.55555555555556

Mean: 78.6984126984127
Standard Deviation: 5.307524477058769

ITERATION NO : 10
FOLD #1
Accuracy: 77.14285714285715
FOLD #2
Accuracy: 69.44444444444444
FOLD #3
Accuracy: 77.14285714285715
FOLD #4
Accuracy: 72.22222222222221
FOLD #5
Accuracy: 75.0

Mean: 74.19047619047619
Standard Deviation: 2.98286025123667

==

Grand Mean: 73.88095238095238
Grand Standard Deviation: 3.154405989706493
==

```
                PREDICTED
+--------+---+---+---+----+
|        |   | 1 | 3 |  2 |
+--------+---+---+---+----+
|        | 1 | 8 | 0 |  0 |
+--------+---+---+---+----+
| ACTUAL | 3 | 0 | 4 |  4 |
+--------+---+---+---+----+
|        | 2 | 2 | 3 | 15 |
+--------+---+---+---+----+
```

4) Observation of WINE data set : (k=3)

Sample output obtained for K-FOLD CROSS VALIDATION over 10 iterations:

ITERATION NO : 1
FOLD #1
Accuracy: 71.42857142857143
FOLD #2
Accuracy: 66.66666666666666
FOLD #3
Accuracy: 80.0
FOLD #4
Accuracy: 63.888888888888886
FOLD #5
Accuracy: 61.111111111111114

Mean: 68.61904761904762
Standard Deviation: 6.632869736949603

ITERATION NO : 2
FOLD #1
Accuracy: 68.57142857142857
FOLD #2

```
Accuracy: 72.22222222222221
FOLD #3
Accuracy: 57.14285714285714
FOLD #4
Accuracy: 69.44444444444444
FOLD #5
Accuracy: 61.111111111111114
--------------------------------------------
Mean: 65.6984126984127
Standard Deviation: 5.640707960334124
--------------------------------------------


ITERATION NO : 3
FOLD #1
Accuracy: 71.42857142857143
FOLD #2
Accuracy: 61.111111111111114
FOLD #3
Accuracy: 62.857142857142854
FOLD #4
Accuracy: 69.44444444444444
FOLD #5
Accuracy: 66.66666666666666
--------------------------------------------
Mean: 66.3015873015873
Standard Deviation: 3.8756171332144387
--------------------------------------------


ITERATION NO : 4
FOLD #1
Accuracy: 74.28571428571429
FOLD #2
Accuracy: 61.111111111111114
FOLD #3
Accuracy: 74.28571428571429
FOLD #4
Accuracy: 75.0
FOLD #5
Accuracy: 72.22222222222221
--------------------------------------------
Mean: 71.38095238095238
Standard Deviation: 5.218240388659981
```

```
------------------------------------------


ITERATION NO : 5
FOLD #1
Accuracy: 65.71428571428571
FOLD #2
Accuracy: 69.44444444444444
FOLD #3
Accuracy: 60.0
FOLD #4
Accuracy: 80.55555555555556
FOLD #5
Accuracy: 80.55555555555556
------------------------------------------
Mean: 71.25396825396825
Standard Deviation: 8.168883805969122
------------------------------------------


ITERATION NO : 6
FOLD #1
Accuracy: 65.71428571428571
FOLD #2
Accuracy: 63.888888888888886
FOLD #3
Accuracy: 60.0
FOLD #4
Accuracy: 69.44444444444444
FOLD #5
Accuracy: 69.44444444444444
------------------------------------------
Mean: 65.6984126984127
Standard Deviation: 3.572451352666761
------------------------------------------


ITERATION NO : 7
FOLD #1
Accuracy: 74.28571428571429
FOLD #2
Accuracy: 69.44444444444444
FOLD #3
Accuracy: 82.85714285714286
```

FOLD #4
Accuracy: 61.111111111111114
FOLD #5
Accuracy: 72.22222222222221

Mean: 71.98412698412699
Standard Deviation: 7.046260415302719

ITERATION NO : 8
FOLD #1
Accuracy: 85.71428571428571
FOLD #2
Accuracy: 63.888888888888886
FOLD #3
Accuracy: 77.14285714285715
FOLD #4
Accuracy: 77.77777777777779
FOLD #5
Accuracy: 77.77777777777779

Mean: 76.46031746031746
Standard Deviation: 7.037243900368803

ITERATION NO : 9
FOLD #1
Accuracy: 74.28571428571429
FOLD #2
Accuracy: 69.44444444444444
FOLD #3
Accuracy: 71.42857142857143
FOLD #4
Accuracy: 69.44444444444444
FOLD #5
Accuracy: 58.333333333333336

Mean: 68.5873015873016
Standard Deviation: 5.425089387297346

ITERATION NO : 10
FOLD #1
Accuracy: 68.57142857142857
FOLD #2
Accuracy: 72.22222222222221
FOLD #3
Accuracy: 77.14285714285715
FOLD #4
Accuracy: 58.333333333333336
FOLD #5
Accuracy: 75.0

Mean: 70.25396825396825
Standard Deviation: 6.6149543888938025

===
Grand Mean: 69.62380952380951
Grand Standard Deviation: 3.1998366202938295
===

```
                 PREDICTED
+--------+---+----+---+---+
|        |   |  1 | 3 | 2 |
+--------+---+----+---+---+
|        | 1 | 15 | 1 | 0 |
+--------+---+----+---+---+
| ACTUAL | 3 |  1 | 4 | 4 |
+--------+---+----+---+---+
|        | 2 |  2 | 1 | 8 |
+--------+---+----+---+---+
```

<u>5) Observation of WISCONSIN data set : (k=1)</u>

<u>Sample output obtained for K-FOLD CROSS VALIDATION over 10 iterations:</u>

ITERATION NO : 1
FOLD #1
Accuracy: 96.40287769784173
FOLD #2

Accuracy: 95.0
FOLD #3
Accuracy: 97.14285714285714
FOLD #4
Accuracy: 93.57142857142857
FOLD #5
Accuracy: 95.0

Mean: 95.42343268242549
Standard Deviation: 1.2413092051214294

ITERATION NO : 2
FOLD #1
Accuracy: 97.12230215827337
FOLD #2
Accuracy: 94.28571428571428
FOLD #3
Accuracy: 97.85714285714285
FOLD #4
Accuracy: 95.0
FOLD #5
Accuracy: 97.14285714285714

Mean: 96.28160328879753
Standard Deviation: 1.382529393841414

ITERATION NO : 3
FOLD #1
Accuracy: 96.40287769784173
FOLD #2
Accuracy: 97.14285714285714
FOLD #3
Accuracy: 95.0
FOLD #4
Accuracy: 92.85714285714286
FOLD #5
Accuracy: 92.14285714285714

Mean: 94.70914696813978
Standard Deviation: 1.9438120886495422

ITERATION NO : 4
FOLD #1
Accuracy: 94.24460431654677
FOLD #2
Accuracy: 95.0
FOLD #3
Accuracy: 95.71428571428572
FOLD #4
Accuracy: 95.71428571428572
FOLD #5
Accuracy: 92.85714285714286

Mean: 94.70606372045222
Standard Deviation: 1.0724621069638771

ITERATION NO : 5
FOLD #1
Accuracy: 93.5251798561151
FOLD #2
Accuracy: 95.0
FOLD #3
Accuracy: 95.71428571428572
FOLD #4
Accuracy: 93.57142857142857
FOLD #5
Accuracy: 95.71428571428572

Mean: 94.70503597122303
Standard Deviation: 0.9799285744627404

ITERATION NO : 6
FOLD #1
Accuracy: 96.40287769784173
FOLD #2
Accuracy: 95.71428571428572
FOLD #3
Accuracy: 96.42857142857143

FOLD #4
Accuracy: 95.71428571428572
FOLD #5
Accuracy: 97.85714285714285

Mean: 96.4234326824255
Standard Deviation: 0.7825282901000978

ITERATION NO : 7
FOLD #1
Accuracy: 94.96402877697841
FOLD #2
Accuracy: 97.14285714285714
FOLD #3
Accuracy: 97.85714285714285
FOLD #4
Accuracy: 95.71428571428572
FOLD #5
Accuracy: 96.42857142857143

Mean: 96.42137718396711
Standard Deviation: 1.0203774676416417

ITERATION NO : 8
FOLD #1
Accuracy: 94.96402877697841
FOLD #2
Accuracy: 95.0
FOLD #3
Accuracy: 97.14285714285714
FOLD #4
Accuracy: 97.85714285714285
FOLD #5
Accuracy: 93.57142857142857

Mean: 95.70709146968139
Standard Deviation: 1.5682678702155175

ITERATION NO : 9
FOLD #1
Accuracy: 96.40287769784173
FOLD #2
Accuracy: 97.14285714285714
FOLD #3
Accuracy: 95.71428571428572
FOLD #4
Accuracy: 96.42857142857143
FOLD #5
Accuracy: 95.71428571428572

Mean: 96.28057553956835
Standard Deviation: 0.5332463777444121

ITERATION NO : 10
FOLD #1
Accuracy: 95.68345323741008
FOLD #2
Accuracy: 97.14285714285714
FOLD #3
Accuracy: 95.71428571428572
FOLD #4
Accuracy: 95.71428571428572
FOLD #5
Accuracy: 93.57142857142857

Mean: 95.56526207605344
Standard Deviation: 1.1421526587190403

===
Grand Mean: 95.62230215827337
Grand Standard Deviation: 0.6865163683041445
===

PREDICTED

		Benign	Malignant
	Benign	84	4
ACTUAL	Malignant	5	47

6) Observation of WISCONSIN data set : (k=3)

Sample output obtained for K-FOLD CROSS VALIDATION over 10 iterations:

ITERATION NO : 1
FOLD #1
Accuracy: 97.84172661870504
FOLD #2
Accuracy: 97.85714285714285
FOLD #3
Accuracy: 95.71428571428572
FOLD #4
Accuracy: 95.0
FOLD #5
Accuracy: 97.14285714285714

Mean: 96.71120246659814
Standard Deviation: 1.157553185771413

ITERATION NO : 2
FOLD #1
Accuracy: 97.84172661870504
FOLD #2
Accuracy: 98.57142857142858
FOLD #3
Accuracy: 97.14285714285714
FOLD #4
Accuracy: 95.71428571428572
FOLD #5
Accuracy: 99.28571428571429

Mean: 97.71120246659817
Standard Deviation: 1.2285606128258935

ITERATION NO : 3
FOLD #1
Accuracy: 97.12230215827337
FOLD #2

```
Accuracy: 96.42857142857143
FOLD #3
Accuracy: 96.42857142857143
FOLD #4
Accuracy: 95.0
FOLD #5
Accuracy: 96.42857142857143
-------------------------------------------
Mean: 96.28160328879753
Standard Deviation: 0.6948496932891896
-------------------------------------------


ITERATION NO : 4
FOLD #1
Accuracy: 95.68345323741008
FOLD #2
Accuracy: 97.85714285714285
FOLD #3
Accuracy: 97.14285714285714
FOLD #4
Accuracy: 97.14285714285714
FOLD #5
Accuracy: 95.0
-------------------------------------------
Mean: 96.56526207605343
Standard Deviation: 1.0548763216207666
-------------------------------------------


ITERATION NO : 5
FOLD #1
Accuracy: 97.12230215827337
FOLD #2
Accuracy: 95.71428571428572
FOLD #3
Accuracy: 95.0
FOLD #4
Accuracy: 96.42857142857143
FOLD #5
Accuracy: 97.85714285714285
-------------------------------------------
Mean: 96.42446043165468
Standard Deviation: 1.007274993092589
```

ITERATION NO : 6
FOLD #1
Accuracy: 94.96402877697841
FOLD #2
Accuracy: 95.0
FOLD #3
Accuracy: 96.42857142857143
FOLD #4
Accuracy: 96.42857142857143
FOLD #5
Accuracy: 97.14285714285714

Mean: 95.9928057553957
Standard Deviation: 0.865615038791126

ITERATION NO : 7
FOLD #1
Accuracy: 96.40287769784173
FOLD #2
Accuracy: 94.28571428571428
FOLD #3
Accuracy: 98.57142857142858
FOLD #4
Accuracy: 97.85714285714285
FOLD #5
Accuracy: 92.14285714285714

Mean: 95.85200411099693
Standard Deviation: 2.3634867270207987

ITERATION NO : 8
FOLD #1
Accuracy: 95.68345323741008
FOLD #2
Accuracy: 96.42857142857143
FOLD #3
Accuracy: 97.14285714285714

FOLD #4
Accuracy: 95.71428571428572
FOLD #5
Accuracy: 96.42857142857143

Mean: 96.27954779033915
Standard Deviation: 0.5412151251221536

ITERATION NO : 9
FOLD #1
Accuracy: 95.68345323741008
FOLD #2
Accuracy: 96.42857142857143
FOLD #3
Accuracy: 97.85714285714285
FOLD #4
Accuracy: 97.14285714285714
FOLD #5
Accuracy: 97.14285714285714

Mean: 96.85097636176772
Standard Deviation: 0.7381457810829236

ITERATION NO : 10
FOLD #1
Accuracy: 97.84172661870504
FOLD #2
Accuracy: 97.85714285714285
FOLD #3
Accuracy: 97.14285714285714
FOLD #4
Accuracy: 97.14285714285714
FOLD #5
Accuracy: 96.42857142857143

Mean: 97.28263103802672
Standard Deviation: 0.5312519173956596

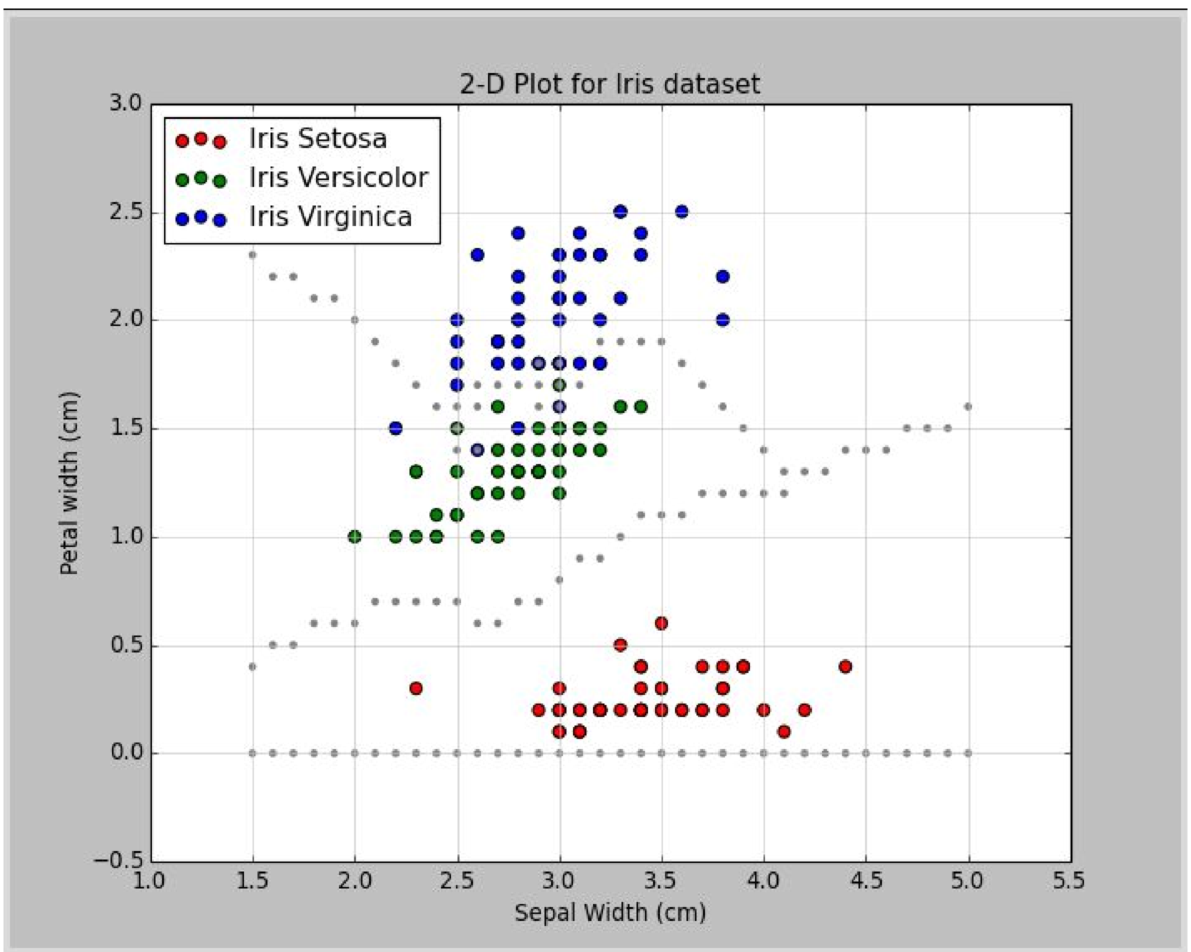
===

Grand Mean: 96.59516957862282
Grand Standard Deviation: 0.5422312696009228
===

PREDICTED

		Benign	Malignant
	Benign	88	0
ACTUAL	Malignant	5	47

Q-3 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 in each row and column where the classification changes from one class to another.



CODE:

There is the code for plotting the graph, with x axis as "Sepal Width" and y axis as "Petal Width" : The decision boundary is automatically plotted by recording transition from one class to other :

```
from matplotlib.pyplot import *

def data():
f = open('iris.data', 'rb')
dataset = f.readlines()
length = len(dataset)
test=[]
lists=[]
for i in range(0, length):
        test = dataset[i].split(',')
        lists.append(test)
        test=[]

return [map(float, l[:4]) for l in lists], [l[-1] for l in lists]

index=0
preindex=0
i=1.5
j=-0.5
#Obtaining the numerical values and the class labels:
matrix, labels = data()
xcord1 = []
ycord1 = []
# sepal width
xcord2 = []
ycord2 = []
 # petal width
xcord3 = []
ycord3 = []
decision_line_x=[]
decision_line_y=[]
y = 3
x = 1
#Locating x and y coordinates of the points given ( sepal width and petal width)
for n, elem in enumerate(matrix):
```

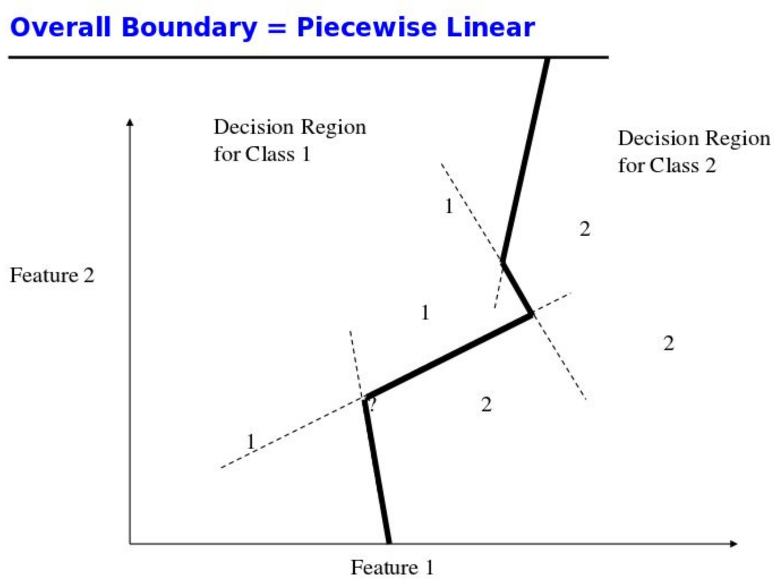
```
if labels[n] == 'Iris-setosa\n':
        length=len(xcord1)
        xcord1.insert(length,matrix[n][x])
        length=len(ycord1)
        ycord1.insert(length,matrix[n][y])
if labels[n] == 'Iris-versicolor\n':
        length=len(xcord2)
        xcord2.insert(length,matrix[n][x])
        length=len(ycord2)
        ycord2.insert(length,matrix[n][y])
if labels[n] == 'Iris-virginica\n':
        length=len(xcord3)
        xcord3.insert(length,matrix[n][x])
        length=len(ycord3)
        ycord3.insert(length,matrix[n][y])


#Plotting the decision bounday
while i<5:
j=0
while j<3:
        max_value=10000000
        temp1='iris'
        preindex=index
        for k in range(150):
        t = matrix[k][1]-i
        t2 = matrix[k][3]-j
        temp = pow(t,2)+pow(t2,2)
        if max_value>temp:
        max_value=temp
        if labels[k]=='Iris-setosa\n':
                index=1
        elif labels[k]=='Iris-versicolor\n':
                index=2
        else :
                index=3
        if preindex==index:
        pass
        if preindex !=index:
        decision_line_x.append(i)
        decision_line_y.append(j)
        j=j+0.1
i=i+0.1
```

```
#Plotting the graph
ax = figure().add_subplot(111)
type4 = ax.scatter(decision_line_x, decision_line_y, 10, color ='grey')

ax.set_title('2-D Plot for Iris dataset', fontsize=14)
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.set_xlabel('Sepal Width (cm)')
ax.set_ylabel('Petal width (cm)')
ax.grid(True,linestyle='-',color='0.75')

show()
```

Q-4 Will the decision boundary of a 3-NN (nearest neighbor) classifier be piecewise linear? Argue the correctness of your answer.

<u>For plotting :</u>

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 is small i.e k=3).

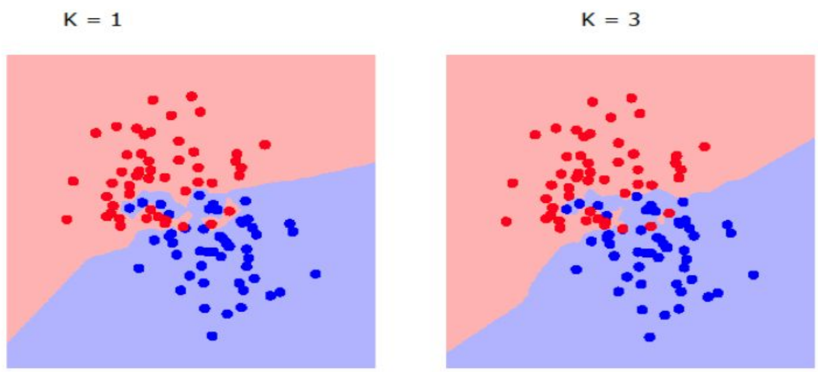


Increasing the value of k for KNN “Simplifies” / “Smoothens” decision boundary.

-> Majority voting means less emphasis on individual points.

-> Here K variation is small, thus we cannot figure out easily, but effectively, with proper K lines almost becomes curves.

-> Averaging over more data will lead to more smoothness. (Increasing k)

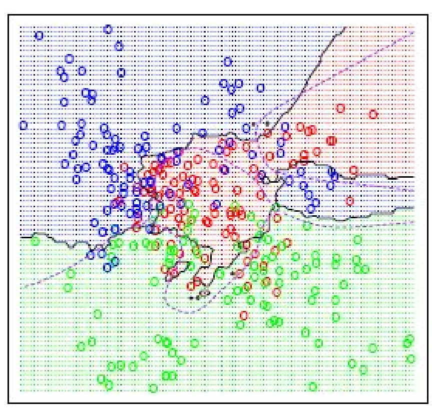


Increasing further, smoothens the decision boundary further.

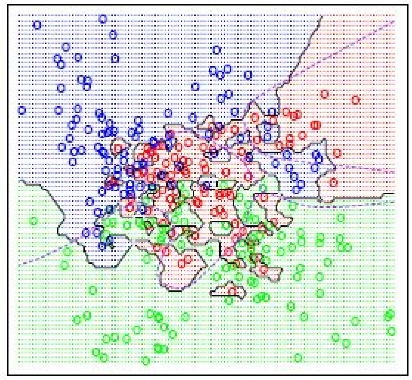
The classification boundaries generated by a given training data set and 15 Nearest Neighbors are shown below. As a comparison, the classification boundaries generated for the same training data but with 1 Nearest Neighbor.

We can see that the classification boundaries induced by 1 NN are much more complicated than 15 NN.

15 Nearest Neighbors (below)



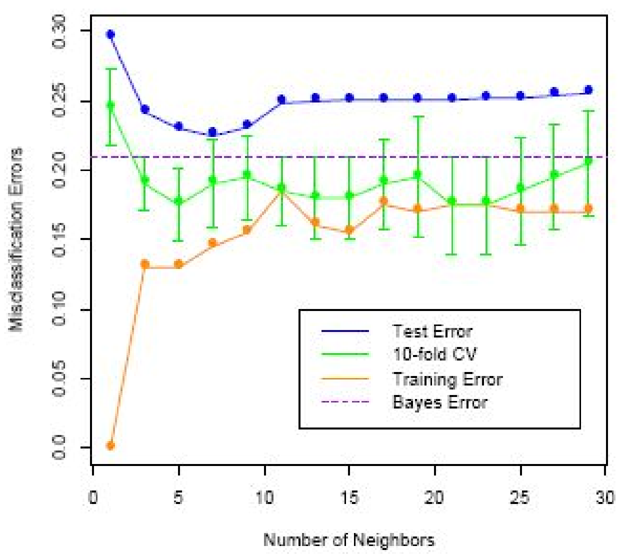
1 Nearest Neighbor (below)



For K-nearest neighbor (kNN) - We can find the K nearest neighbors, and return the majority vote of their labels. K yields smoother predictions, since we average over more data.

K=1 yields y=piecewise constant labeling.

K = N predicts y=globally constant (majority) label.



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.