In [1]:
```
from sklearn.metrics import confusion_matrix
y_true=[2,0,2,2,0,1]
y_pred=[0,0,2,2,0,2]
confusion_matrix(y_true,y_pred)
```

Out[1]:
```
array([[2, 0, 0],
       [0, 0, 1],
       [1, 0, 2]], dtype=int64)
```

Python script for confusion matrix creation.

In [2]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
actual = [1, 1, 0, 1, 0, 0, 1, 0, 0, 0]
predicted = [1, 0, 0, 1, 0, 0, 1, 1, 1, 0]
results = confusion_matrix(actual, predicted)
print('Confusion Matrix :')
print(results)
print('Accuracy Score :',accuracy_score(actual, predicted))
print('Report : ')
print(classification_report(actual, predicted))
```

```
Confusion Matrix :
[[4 2]
 [1 3]]
Accuracy Score : 0.7
Report :
              precision    recall  f1-score   support

           0       0.80      0.67      0.73         6
           1       0.60      0.75      0.67         4

    accuracy                           0.70        10
   macro avg       0.70      0.71      0.70        10
weighted avg       0.72      0.70      0.70        10
```

Apply Multi-Class Classification on the suitable dataset (Using KNN).

In [3]:

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import datasets
```

In [4]:

```
# Load dataset
iris=datasets.load_iris()
print("Iris Data set loaded...")
```

```
Iris Data set loaded...
```

In [5]:

```
# Split the data into train and test samples
x_train, x_test, y_train, y_test = train_test_split(iris.data,iris.target,test_size=0.1)
print("Dataset is split into training and testing...")
print("Size of training data and its label",x_train.shape,y_train.shape)
print("Size of testing data and its label",x_test.shape, y_test.shape)
```

```
Dataset is split into training and testing...
Size of training data and its label (135, 4) (135,)
Size of testing data and its label (15, 4) (15,)
```

In [6]:

```
# Prints Label no. and their names
for i in range(len(iris.target_names)):
        print("Label", i , "-",str(iris.target_names[i]))
```

```
Label 0 - setosa
Label 1 - versicolor
Label 2 - virginica
```

In [7]:
```
print(iris['DESCR'])
```

```
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :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

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken
from Fisher's paper. Note that it's the same as in R, but not as in the UCI
Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the field and
is referenced frequently to this day.  (See Duda & Hart, for example.)  The
```

```
data set contains 3 classes of 50 instances each, where each class refers to a
type of iris plant.  One class is linearly separable from the other 2; the
latter are NOT linearly separable from each other.

.. topic:: References

   - Fisher, R.A. "The use of multiple measurements in taxonomic problems"
     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
     Mathematical Statistics" (John Wiley, NY, 1950).
   - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.
     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
     Structure and Classification Rule for Recognition in Partially Exposed
     Environments".  IEEE Transactions on Pattern Analysis and Machine
     Intelligence, Vol. PAMI-2, No. 1, 67-71.
   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions
     on Information Theory, May 1972, 431-433.
   - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II
     conceptual clustering system finds 3 classes in the data.
   - Many, many more ...
```

In [8]:

```python
# Create object of KNN classifier-Create a KNN(KNeighborsClassifier) object
classifier = KNeighborsClassifier(n_neighbors=1)

# Perform Training-fit it to the data
classifier.fit(x_train, y_train)

# Perform testing
y_pred=classifier.predict(x_test)
```

In [9]:

```
# Display the results
print("Results of Classification using K-nn with K=1 ")
for r in range(0,len(x_test)):
    print(" Sample:", str(x_test[r]), " Actual-label:", str(y_test[r]), " Predicted-label:",
str(y_pred[r]))
print("Classification Accuracy :" , classifier.score(x_test,y_test));
```

```
Results of Classification using K-nn with K=1
 Sample: [7.1 3.  5.9 2.1]  Actual-label: 2  Predicted-label: 2
 Sample: [5.2 4.1 1.5 0.1]  Actual-label: 0  Predicted-label: 0
 Sample: [4.4 2.9 1.4 0.2]  Actual-label: 0  Predicted-label: 0
 Sample: [4.8 3.  1.4 0.1]  Actual-label: 0  Predicted-label: 0
 Sample: [5.5 2.4 3.8 1.1]  Actual-label: 1  Predicted-label: 1
 Sample: [5.  2.3 3.3 1. ]  Actual-label: 1  Predicted-label: 1
 Sample: [6.4 2.9 4.3 1.3]  Actual-label: 1  Predicted-label: 1
 Sample: [5.4 3.9 1.3 0.4]  Actual-label: 0  Predicted-label: 0
 Sample: [7.7 2.8 6.7 2. ]  Actual-label: 2  Predicted-label: 2
 Sample: [4.9 2.5 4.5 1.7]  Actual-label: 2  Predicted-label: 1
 Sample: [7.9 3.8 6.4 2. ]  Actual-label: 2  Predicted-label: 2
 Sample: [6.9 3.2 5.7 2.3]  Actual-label: 2  Predicted-label: 2
 Sample: [6.7 3.1 5.6 2.4]  Actual-label: 2  Predicted-label: 2
 Sample: [4.6 3.4 1.4 0.3]  Actual-label: 0  Predicted-label: 0
 Sample: [5.2 3.5 1.5 0.2]  Actual-label: 0  Predicted-label: 0
Classification Accuracy : 0.9333333333333333
```

In [10]:
```
from sklearn.metrics import classification_report, confusion_matrix
print('Confusion Matrix')
print(confusion_matrix(y_test,y_pred))
print('Accuracy Metrics')
print(classification_report(y_test,y_pred))
```

```
Confusion Matrix
[[6 0 0]
 [0 3 0]
 [0 1 5]]
Accuracy Metrics
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         6
           1       0.75      1.00      0.86         3
           2       1.00      0.83      0.91         6

    accuracy                           0.93        15
   macro avg       0.92      0.94      0.92        15
weighted avg       0.95      0.93      0.94        15
```

Accuracy Metrices : precision recall f1-score support

```
    0       1.00       1.00       1.00        11
    1       1.00       0.92       0.96        13
    2       0.86       1.00       0.92         6

accuracy                          0.97        30
```

macro avg 0.95 0.97 0.96 30 weighted avg 0.97 0.97 0.97 30

The Acccuracy of Metrices is : 0.9666666666666667 Confusion Matrix [[11 0 0] [0 12 1] [0 0 6]]

Precision — The percentage of correctly classified results among that class.

Recall — The number of true positive cases found over the total number of positive cases found (true positives + false negatives).

F1-score — The harmonic mean of precision and recall. The F1-score will always be between 1.00 and 0.00 with 1.00 being the best score.

Support — The number of occurrences of the class in the dataset.

Accuracy—The sum of the true positives and true negatives over the total number of samples.

Macro Average — The mean average of the precision/recall/F1-score of all the classes.(1+1+0.86)/3=0.95

Weighted Average — Calculates the scores for each class independent of one another but when it adds them together it takes into account the number of true classifications of each class. (1(Precision)*11(support))/30(Total support) + (1*13)/30 + (0.86*6)/30 = 0.97