Classification of different species of Iris flowers using KNN Classifier.

```
The Dataset: We are going to use the famous iris data set for our KNN
example. The dataset consists of four attributes: sepal-width, sepal-length,
petal-width and petal-length. These are the attributes of specific types of
iris plant. The task is to predict the class to which these plants belong.
There are three classes in the dataset: Iris-setosa, Iris-versicolor and
Iris-virginica. Just for your information, the photo’s of above flowers are
as follows:
```

Setosa

Versicolor

Virginica

In [3]:
```
#importing the dataset
from sklearn import datasets
# import some data to play with
iris = datasets.load_iris()
```

In [4]:
```
iris.keys()
```

Out[4]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names
', 'filename', 'data_module'])
```

In [5]:
```
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 Analysi
```

```
s.
     (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 Transactio
ns
     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 [6]:
```
iris.target_names
```

Out[6]:
```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

In [7]:
```
iris.feature_names
```

Out[7]:
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

In [8]:
```
iris.data.shape
```

Out[8]:
```
(150, 4)
```

In [9]:
```
iris.data
```

Out[9]:
```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
```

In [10]:
```
iris.target
```

Out[10]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

In [11]:
```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, te
```

In [12]:
```
# Fitting classifier to the Training set
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
#classifier = KNeighborsClassifier(n_neighbors=5, metric='euclidean')
classifier.fit(X_train, y_train)
```

Out[12]:
```
KNeighborsClassifier()
```

In [13]:
```
y_pred = classifier.predict(X_test)
```

```
C:\Users\student\OneDrive\aNACONDA\lib\site-packages\sklearn\neighbors\_class
ification.py:228: FutureWarning: Unlike other reduction functions (e.g. `skew
`, `kurtosis`), the default behavior of `mode` typically preserves the axis i
t acts along. In SciPy 1.11.0, this behavior will change: the default value o
f `keepdims` will become False, the `axis` over which the statistic is taken
will be eliminated, and the value None will no longer be accepted. Set `keepd
ims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

In [14]:
```
y_pred
```

Out[14]:
```
array([1, 1, 2, 1, 0, 0, 1, 2, 2, 0, 2, 1, 2, 0, 2, 1, 2, 0, 1, 0, 2, 0,
       1, 1, 2, 2, 1, 2, 2, 2])
```

In [15]:
```
y_test
```

Out[15]:
```
array([1, 1, 2, 1, 0, 0, 1, 2, 2, 0, 2, 1, 2, 0, 2, 1, 2, 0, 1, 0, 2, 0,
       1, 1, 2, 2, 1, 2, 2, 2])
```

In [16]:
```
for i in range(len(y_test)):
    if (y_test[i] != y_pred[i]):
        print(i, y_test[i], y_pred[i])
```

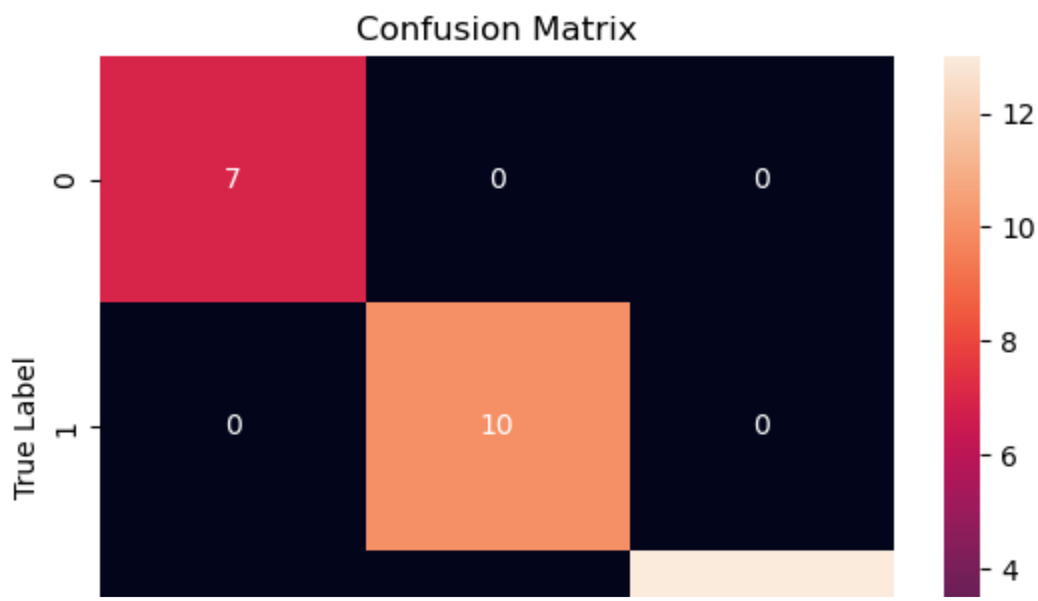
In [17]:
```
#Let's test the performance of our model.
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

In [18]:
```
print ("Confusion Matrix : \n", cm)
```

```
Confusion Matrix :
 [[ 7  0  0]
 [ 0 10  0]
 [ 0  0 13]]
```

In [19]:
```
#It’s often useful to visualize the confusion matrix using Heatmap.
import matplotlib.pyplot as plt
import seaborn as sn
sn.heatmap(cm, annot=True)
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.title('Confusion Matrix')
```

Out[19]:
```
Text(0.5, 1.0, 'Confusion Matrix')
```



In [20]:
```
#Performance measure - Accuracy
from sklearn.metrics import accuracy_score
print ("Accuracy : ", accuracy_score(y_test, y_pred))
```

```
Accuracy :  1.0
```

In [21]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         7
           1       1.00      1.00      1.00        10
           2       1.00      1.00      1.00        13

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

In [ ]: