

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
from sklearn.datasets import load_iris
iris = load_iris()
print(iris)
```

```
{'data': 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, 0.2])
```

In [3]:

```
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 [4]:

```
X = iris.data
y = iris.target
print("x",X[:5])
print("y",y[:5])
```

```
x [[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]]
y [0 0 0 0 0]
```

In [5]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train,y_test = train_test_split(X,y, test_size=0.20)
print(iris.data.shape)
print(len(X_train))
print(len(y_train))
print(len(X_test))
print(len(y_test))
```

```
(150, 4)
120
120
30
30
```

In [6]:

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, y_train)
```

Out[6]:

```
KNeighborsClassifier(n_neighbors=3)
```

In [7]:

```
y_pred = knn.predict(X_test)
print(y_pred)
```

```
[1 1 1 2 2 0 2 2 2 0 1 1 0 0 0 0 0 1 2 2 0 1 1 1 2 1 0 0 2 1]
```

In [8]:

```

from sklearn import metrics
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
print("Accuracy = ", metrics.accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

```

Accuracy = 1.0

```

[[10  0  0]
 [ 0 11  0]
 [ 0  0  9]]

```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	1.00	1.00	11
2	1.00	1.00	1.00	9
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

In [9]:

```

pre_target = [iris.target_names[i] for i in y_pred]
print("Predicted Target = ", pre_target, "\n\n")
actual_target = [iris.target_names[i] for i in y_test]
print("Actual Target = ", actual_target, "\n\n")
print("\t Predicted", "\t\t Actual", "\t\t\t Answer")
for i in range(0, len(pre_target)):
    print(i, ":\t", pre_target[i], "\t\t", actual_target[i], "\t\t", end="\t")
    if(pre_target[i] == actual_target[i]):
        print("Yes")
    else:
        print("No")

```

```

Predicted Target = ['versicolor', 'versicolor', 'versicolor', 'virginica',
'virginica', 'setosa', 'virginica', 'virginica', 'virginica', 'setosa', 'ver
sicolor', 'versicolor', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'v
ersicolor', 'virginica', 'virginica', 'setosa', 'versicolor', 'versicolor',
'versicolor', 'virginica', 'versicolor', 'setosa', 'setosa', 'virginica', 'v
ersicolor']

```

```

Actual Target = ['versicolor', 'versicolor', 'versicolor', 'virginica', 'vi
rginica', 'setosa', 'virginica', 'virginica', 'virginica', 'setosa', 'versic
olor', 'versicolor', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'vers
icolor', 'virginica', 'virginica', 'setosa', 'versicolor', 'versicolor', 've
rsicolor', 'virginica', 'versicolor', 'setosa', 'setosa', 'virginica', 'vers
icolor']

```

	Predicted	Actual	Answer
0 :	versicolor	versicolor	Yes
1 :	versicolor	versicolor	Yes
2 :	versicolor	versicolor	Yes
3 :	virginica	virginica	Yes
4 :	virginica	virginica	Yes
5 :	setosa	setosa	Yes
6 :	virginica	virginica	Yes
7 :	virginica	virginica	Yes
8 :	virginica	virginica	Yes
9 :	setosa	setosa	Yes
10 :	versicolor	versicolor	Yes
11 :	versicolor	versicolor	Yes
12 :	setosa	setosa	Yes
13 :	setosa	setosa	Yes
14 :	setosa	setosa	Yes
15 :	setosa	setosa	Yes
16 :	setosa	setosa	Yes
17 :	versicolor	versicolor	Yes
18 :	virginica	virginica	Yes
19 :	virginica	virginica	Yes
20 :	setosa	setosa	Yes
21 :	versicolor	versicolor	Yes
22 :	versicolor	versicolor	Yes
23 :	versicolor	versicolor	Yes
24 :	virginica	virginica	Yes
25 :	versicolor	versicolor	Yes
26 :	setosa	setosa	Yes
27 :	setosa	setosa	Yes
28 :	virginica	virginica	Yes
29 :	versicolor	versicolor	Yes

In [10]:

```
sam1 = [X_test[2]]  
a = knn.predict(sam1)  
print(a)  
sam2 = [y_test[2]]  
print(sam2)  
if(a == sam2):  
    print("Correct")  
else:  
    print("Wrong")
```

```
[1]  
[1]  
Correct
```

In [ ]: