Step 1: Load Dataset and split it

sklearn.datasets.load_iris()

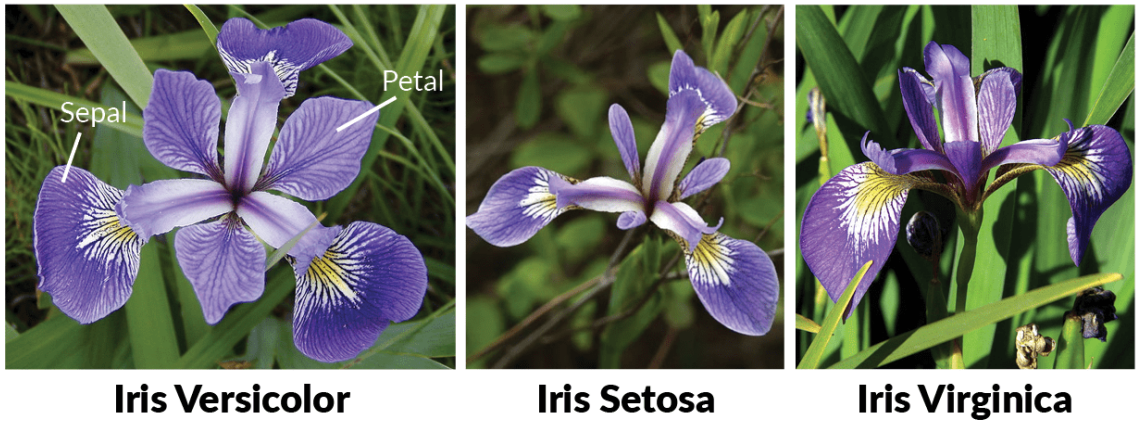
Loads and returns iris dataset as a Bunch object with the following attributes.

--> data: ndarray object of input samples of size (150, 4)

--> target: ndarray object of taget values of size (150,)

This data sets consists of 3 different types of irises' -> Setosa, Versicolour, and Virginica.

The rows being the samples and the columns being: Sepal Length, Sepal Width, Petal Length and Petal Width.



Other attributes:

--> feature_names: List of all input feature names

--> target_names: List of all target classes

- We can use **as_frame = True** to get data as pandas DataFrame and target as pandas Series/DataFrame.

In [1]:
```
from sklearn.datasets import load_iris

iris = load_iris()
print(iris.feature_names)
print(iris.target_names)
x = iris.data
y = iris.target
print(x[0:6])
print(y)
print(x.shape)
print(y.shape)
```

```
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal wid
th (cm)']
['setosa' 'versicolor' 'virginica']
[[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]]
[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 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]
(150, 4)
(150,)
```

sklearn.model_selection.train_test_split(x, y, test_size)

Returns the train and test sets from given data set

In [2]:
```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.33
print(x_train.shape)
print(x_test.shape)
```

```
(100, 4)
(50, 4)
```

Step 2: Build the model

sklearn.naive_bayes.GaussianNB()

--> Creates a Gaussian Naive Bayes classifier object.

--> fit(x_train, y_train) method train the model with train data.

--> predict(x_test) method predicts the output for given test input.

In [7]:
```
from sklearn.naive_bayes import GaussianNB

model1 = GaussianNB()
model1.fit(x_train, y_train)
```

Out[7]:
```
GaussianNB()
```

Step 3: Test the model

In [8]:
```
import numpy as np
y_pred = model1.predict(np.array([6.3, 3.3, 6.0, 0.5]).reshape(1, -1))
print(y_pred)
y_pred = model1.predict(x_test)
print(f"Number of mislabeled points out of a total of {x_test.shape[0]}: {
```

```
[2]
Number of mislabeled points out of a total of 50: 0
```

sklearn.metrics.accuracy_score(y_pred, y_test)

--> Returns the accuracy from predicted and actual test label values

In [9]:
```
from sklearn.metrics import accuracy_score

print("Accuracy:", accuracy_score(y_pred, y_test))
```

```
Accuracy: 1.0
```

Perform the classification using Multinomial Naive Bayes Classifier

In [10]:
```
from sklearn.naive_bayes import MultinomialNB

model2 = MultinomialNB()
model2.fit(x_train, y_train)
y_pred = model2.predict(x_test)
print("Accuracy:", accuracy_score(y_pred, y_test))
```

```
Accuracy: 0.64
```

Deployment

In [15]:
```
sl = float(input("Enter Sepal length:"))
sw = float(input("Enter Sepal width:"))
pl = float(input("Enter Petal length:"))
pw = float(input("Enter Petal width:"))
pred = model1.predict(np.array([[sl, sw, pl, pw]]))
print(iris.target_names[pred])
```

```
Enter Sepal length:5.1
Enter Sepal width:3.5
Enter Petal length:1.4
Enter Petal width:0.2
['setosa']
```