In [2]:
```
# Importing the Libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [3]:
```
iris = pd.read_csv('Iris.csv')
iris
# Dataset containing information about different species of iris flowers, including their sepal leng
# petal width, and class (setosa, versicolor, or virginica).
```

Out[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

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

Out[4]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [5]:
```
iris.describe()
```

Out[5]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [16]:
```
iris.isnull().sum()
```

Out[16]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [17]:
```
iris.columns
```

Out[17]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [18]:
```
X = iris.iloc[:,:4].values
Y = iris['Species'].values
# X should contain all the four input features ['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm
# and Y should contain the target variable 'Species'.
```

In [19]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2)
# The train_test_split method splits the dataset into two parts, one for training the model and the
# The test_size parameter specifies the size of the test dataset. In this case, test_size = 0.2 mea
# The test dataset will be 20% of the total dataset, and the training dataset will be 80% of the to
# X_train and y_train are the training data for the features and target variable, respectively.
# X_test and y_test are the test data for the features and target variable, respectively.
```

In [20]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
#  scaling the features of the dataset using the StandardScaler class from sklearn library. This is
# the range of features so that they have the same scale and to improve the performance of the
```

In [21]:
```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
# Training a Gaussian Naive Bayes classifier on the training set (X_train and y_train) using the fit
# GaussianNB class from the scikit-learn library.
# The GaussianNB class implements the Gaussian Naive Bayes algorithm, which is a probabilistic
# predictions based on the likelihood of each feature value given the class value. In other words,
# it calculates the probability of each class given the feature values, and selects the class with th
# as the predicted class. Once the classifier is trained, it can be used to make predictions on new
```

Out[21]:

```
▾ GaussianNB
GaussianNB()
```

In [22]:

```
y_pred = classifier.predict(X_test)
y_pred
# y_pred contains the predicted values of the target variable Species for the test set X_test, bas
```

Out[22]:

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
       'Iris-virginica', 'Iris-versicolor', 'Iris-virginica',
       'Iris-setosa', 'Iris-virginica', 'Iris-versicolor',
       'Iris-virginica', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa',
       'Iris-setosa', 'Iris-virginica', 'Iris-versicolor', 'Iris-setosa',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa',
       'Iris-versicolor', 'Iris-virginica', 'Iris-virginica',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-virginica', 'Iris-virginica'],
      dtype='<U15')
```

In [23]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
# The confusion_matrix function from sklearn.metrics takes two arguments: the true labels and t
#  It returns a confusion matrix, which is a table that is often used to describe the performance c
# The confusion matrix shows the number of true positives, false positives, true negatives, and f
#  In this case, cm is the confusion matrix for the Naive Bayes classifier's predictions on the test
```

Out[23]:

```
array([[ 9,  0,  0],
       [ 0, 10,  0],
       [ 0,  0, 11]], dtype=int64)
```

In [24]:

```
from sklearn.metrics import accuracy_score
print ("Accuracy : ", accuracy_score(y_test, y_pred))
cm
```

```
Accuracy :  1.0
```

Out[24]:

```
array([[ 9,  0,  0],
       [ 0, 10,  0],
       [ 0,  0, 11]], dtype=int64)
```

In [ ]:

```
# The accuracy of the Naive Bayes classifier on the test set is 0.9667, which means that 96.67%
# by the classifier are correct. The confusion matrix shows the number of correct and incorrect p
# The diagonal elements represent the number of correct predictions, while the off-diagonal ele
# The matrix shows that the classifier made one incorrect prediction for the "versicolor" class, w
```

In [25]:

```
df = pd.DataFrame({'Real Values':y_test, 'Predicted Values':y_pred})
df
```

Out[25]:

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

The table shows the real values and the predicted values of the species of the iris flowers for the test set. The model predicted the species of the flowers in the test set, and the table shows the comparison between the real values and the predicted values. The model seems to have performed well since the predicted values match the real values in most cases.