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


#  import the Iris Flower dataset
dataset = pd.read_csv('https://raw.githubusercontent.com/mk-gurucharan/Classification/master/IrisDataset.csv')
# assign the 4 independent variables to X
X = dataset.iloc[:,:4].values
# assign dependent variable ‘species’ to Y
y = dataset['species'].values
# print first five value of dataset
dataset.head(5)
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
# split the data into the training set and the test set
from sklearn.model_selection import train_test_split
# test_size=0.2, which means that 20% of the dataset will be used
# for testing purpose as the test set
# remaining 80% will be used as the training set for training
# the Naive Bayes classification model
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)


# feature scaling
from sklearn.preprocessing import StandardScaler
# dataset is scaled down to a smaller range using the Feature Scaling
sc = StandardScaler()
# X_train and X_test values are scaled down to smaller values
# to improve the speed of the program.
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)


# introduce the class GaussianNB that is used from the sklearn.naive_bayes
from sklearn.naive_bayes import GaussianNB
```

```
# assign the GaussianNB class to the variable classifier
classifier = GaussianNB()
# fit the X_train and y_train values to it for training purpose.
classifier.fit(X_train, y_train)
```

```
GaussianNB()
```

```
# Predicting the Test set results
# use the the classifier.predict() to predict the values for the Test set
y_pred = classifier.predict(X_test)
# values predicted are stored to the variable y_pred
print(y_pred)
```

```
['setosa' 'versicolor' 'versicolor' 'versicolor' 'virginica' 'versicolor'
 'setosa' 'versicolor' 'setosa' 'setosa' 'virginica' 'versicolor'
 'versicolor' 'virginica' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'setosa' 'virginica' 'setosa'
 'virginica' 'virginica' 'virginica' 'virginica' 'versicolor' 'setosa'
 'versicolor']
```

```
# see the Accuracy of the trained model and plot the confusion matrix
# confusion matrix is a table that is used to show the number of
# correct and incorrect predictions on a classification problem
# when the real values of the Test Set are known
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

from sklearn.metrics import accuracy_score
# show accuracy of correctly clasified test data => 96.67%
print("Accuracy : ", accuracy_score(y_test, y_pred))
print(cm)
```

```
Accuracy :  0.9
[[ 7  0  0]
 [ 0 14  2]
 [ 0  1  6]]
```

```
# out of 30 test set data, 29 were correctly classified and
# only 1 was incorrectly classified.

# a Pandas DataFrame is created to compare the classified values
# of both the original Test set (y_test) and the predicted results (y_pred)
df = pd.DataFrame({'Real Values':y_test, 'Predicted Values':y_pred})
# one incorrect prediction that has predicted versicolor instead of virginica
print(df)
```

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

```
from sklearn.metrics import precision_score, recall_score, accuracy_score
m = accuracy_score(y_test, y_pred)
# Error rate (ERR) is calculated as the number of all incorrect
# predictions divided by the total number of the dataset
print("error rate:-", 1 - m)
```

```
error rate:- 0.09999999999999998
```

```
# precision is the ratio where TP is the number of true positives FP False postivies
print(f"Precision:{precision_score(y_test,y_pred,average='micro'):.2f}")
```

```
Precision:0.90
```

```
# recall is the ratio where TP is the number of true positives FN false negatives.
print(f"Recall Score: {recall_score(y_test,y_pred,average='micro'):.2f}")
```

```
Recall Score: 0.90
```

✓ 0s completed at 16:03