In [1]:
```python
import pandas as pd
pf = pd.read_csv("Iris.csv")
```

In [2]:
```python
pf.dtypes
```

Out[2]:
```
Id                 int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

In [3]:
```python
pf
```

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]:
```python
pf.describe
```

Out[4]:
```
<bound method NDFrame.describe of       Id  SepalLengthCm  SepalWi
dthCm  PetalLengthCm  PetalWidthCm  \
0      1            5.1           3.5            1.4           0.2
1      2            4.9           3.0            1.4           0.2
2      3            4.7           3.2            1.3           0.2
3      4            4.6           3.1            1.5           0.2
4      5            5.0           3.6            1.4           0.2
..   ...            ...           ...            ...           ...
145  146            6.7           3.0            5.2           2.3
146  147            6.3           2.5            5.0           1.9
147  148            6.5           3.0            5.2           2.0
148  149            6.2           3.4            5.4           2.3
149  150            5.9           3.0            5.1           1.8

            Species
0       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              ...
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]>
```

In [5]:
```python
pf.isnull().sum()
```

Out[5]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [6]:
```python
pf.drop("Id",axis =1,inplace=True)
```

splitting the data into x and y

In [7]:
```python
x = pf.drop("Species",axis = 1)
```

In [8]:
```python
y = pf["Species"]
```

In [9]:
```python
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(x,y,test_size=0
                                                               )
```

Standardizing the input : fitting to normal distribution and transform

In [10]:

```
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
std = StandardScaler()
X_train = std.fit_transform(X_train)
X_test  = std.fit_transform(X_test)
```

In [11]:

```
X_test
```

```
       [ 0.03028436,  2.2157581 , -1.38669908, -1.25366361],
       [ 0.7475454 , -0.58505972,  1.10437711,  1.3153192 ],
       [ 0.38891488, -0.11825675,  0.53822343,  0.28772607],
       [-0.32834617, -1.75206715,  0.19853122,  0.15927693],
       [ 1.34526294,  0.11514473,  0.82130027,  1.44376834],
       [ 0.26937137, -0.35165824,  0.48160806,  0.41617521],
       [ 0.50845839, -0.35165824,  0.36837733,  0.15927693],
       [ 1.22571943,  0.34854622,  1.27422321,  1.44376834],
       [-0.92606371, -1.75206715, -0.19777635, -0.22607049],
       [-1.16515072,  0.81534919, -1.16023761, -1.25366361],
       [ 0.38891488, -1.05186269,  1.10437711,  0.28772607],
       [-0.68697669,  0.81534919, -1.27346834, -1.25366361],
       [ 1.10617592,  0.11514473,  0.4249927 ,  0.28772607],
       [-0.8065202 ,  1.04875067, -1.27346834, -1.25366361],
       [-0.68697669,  1.04875067, -1.21685297, -1.25366361],
       [-0.32834617,  1.04875067, -1.33008371, -1.25366361],
       [-1.04560721, -1.28526418,  0.48160806,  0.6730735 ],
       [ 0.50845839,  0.81534919,  0.99114637,  1.44376834],
       [ 2.54069802,  1.74895513,  1.55730005,  1.05842092],
       [-0.44788968,  0.81534919, -1.10362224, -1.25366361],
```

Model Building

In [12]:

```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(X_train,Y_train)
```

Out[12]:

```
▾ GaussianNB i ?
GaussianNB()
```

(https://scikit-learn.org/1.4/modules/generated/sklearn.naive_bayes.Gaussian

In [13]:

```
Y_predict = model.predict(X_test)
```

In [14]:

```
print(Y_predict)
```

```
['Iris-setosa' 'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-
setosa'
 'Iris-virginica' 'Iris-setosa' 'Iris-virginica' 'Iris-virginica'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-set
osa'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa' 'Iris-versicolo
r'
 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica' 'Iris-versico
lor'
 'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor' 'Iris-setos
a'
 'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-virginica'
 'Iris-virginica' 'Iris-setosa' 'Iris-versicolor' 'Iris-virginica'
 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolo
r'
 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor' 'Iris-versic
olor'
 'Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica'
 'Iris-setosa' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa' 'Iris
-setosa'
 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica' 'Iris-setosa'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-versicol
or'
 'Iris-setosa' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica' 'Iris-versico
lor'
 'Iris-setosa' 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa' 'Iris-versicolo
r']
```

In [19]:

```
#getting accuracy and other matrices
```

In [ ]:

```
from sklearn.metrics import accuracy_score, precision_score, recall
```

In [ ]:

```
a_score = accuracy_score(Y_test,Y_predict)
```

In [ ]:

```
a_score
```

In [ ]:

```
p_score = precision_score(Y_test,Y_predict,average = 'micro')
p_score
```

In [ ]:

```
r_score = recall_score(Y_test, Y_predict,average = 'micro')
r_score
```

In [ ]:

```
cm = confusion_matrix(Y_test,Y_predict)
cm
```

printing Y predict

In [ ]:

```
Y_predict
```

In [ ]:
```
print(classification_report(Y_predict,Y_test))
```

In [ ]:
```
X_new=[[5.5,3.5,1.3,0.2]]
```

In [ ]:
```
X_new = std.fit_transform(X_new)
```

In [ ]:
```
Y_new = model.predict(X_new)
```

In [ ]:
```
print(Y_new)
```

In [18]:
```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix, class

# Splitting the dataset
X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size

# Train the Naïve Bayes model
model = GaussianNB()
model.fit(X_train, Y_train)

# Make predictions
Y_predict = model.predict(X_test)

# Compute confusion matrix
cm = confusion_matrix(Y_test, Y_predict)

# Compute performance metrics
accuracy = accuracy_score(Y_test, Y_predict)
report = classification_report(Y_test, Y_predict)

# Display results
print("Confusion Matrix:\n", cm)  # ✅ Show confusion matrix
print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:\n", report)  # ✅ Show precision, rec
```

```
Confusion Matrix:
 [[27  0  0]
 [ 0 22  1]
 [ 0  3 22]]
Accuracy: 0.95
Classification Report:
                  precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        27
Iris-versicolor       0.88      0.96      0.92        23
 Iris-virginica       0.96      0.88      0.92        25

       accuracy                           0.95        75
      macro avg       0.95      0.95      0.94        75
   weighted avg       0.95      0.95      0.95        75
```

In [ ]:
```
```