In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df = sns.load_dataset('iris')
df
```

Out[2]:

	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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [3]:
```
x = df[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]
y = df[['species']]
print(x)
print(y)

# x = df.iloc[:,:4]
# y = df['species']
# print(x)
# print(y)
```

```
     sepal_length  sepal_width  petal_length  petal_width
0             5.1          3.5           1.4          0.2
1             4.9          3.0           1.4          0.2
2             4.7          3.2           1.3          0.2
3             4.6          3.1           1.5          0.2
4             5.0          3.6           1.4          0.2
..            ...          ...           ...          ...
145           6.7          3.0           5.2          2.3
146           6.3          2.5           5.0          1.9
147           6.5          3.0           5.2          2.0
148           6.2          3.4           5.4          2.3
149           5.9          3.0           5.1          1.8

[150 rows x 4 columns]
       species
0       setosa
1       setosa
2       setosa
3       setosa
4       setosa
..         ...
145  virginica
146  virginica
147  virginica
148  virginica
149  virginica

[150 rows x 1 columns]
```

In [4]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_st
```

In [5]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_test = sc.fit_transform(x_test)
x_train = sc.fit_transform(x_train)

# print(x_test)
# print(x_train)
```

In [6]:
```
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(x_train, y_train)
```

Out[6]:

▾ GaussianNB

```
GaussianNB()
```

In [7]:
```
y_predict = nb.predict(x_test)
print(y_predict)
```

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

In [8]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, reca

acc = accuracy_score(y_test, y_predict)
print(acc)

pr = precision_score(y_test, y_predict, average='macro') # Using average because we
print(pr)

rc = recall_score(y_test, y_predict, average='macro')
print(rc)

cm = confusion_matrix(y_test, y_predict)
print(cm)

cr = classification_report(y_test, y_predict)
print(cr)
```

```
0.8947368421052632
0.9010989010989011
0.9010989010989011
[[11  0  0]
 [ 0 12  2]
 [ 0  2 11]]
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        11
  versicolor       0.86      0.86      0.86        14
   virginica       0.85      0.85      0.85        13

    accuracy                           0.89        38
   macro avg       0.90      0.90      0.90        38
weighted avg       0.89      0.89      0.89        38
```

In [9]:
```
w = [[2.7,4.6,3.3,1.9]]
w = sc.fit_transform(w)

q = nb.predict(w)
print(q)
```

```
['versicolor']
```