

```
In [2]: import numpy as np
import pandas as pd

df = pd.read_csv("iris.csv")
```

```
In [3]: df.head()
```

Out[3]:

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

```
In [4]: df.describe()
```

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

```
In [5]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
#   Column          Non-Null Count  Dtype
---  -
0   sepal_length    150 non-null   float64
1   sepal_width     150 non-null   float64
2   petal_length    150 non-null   float64
3   petal_width     150 non-null   float64
4   species         150 non-null   object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

```
In [6]: df.isnull().sum()
```

Out[6]:

```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

```
In [7]: df.shape
```

Out[7]: (150, 5)

```
In [8]: df['species'].unique()
```

Out[8]: array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)

```
In [9]: df.keys()
```

Out[9]: Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species'], dtype='object')

```
In [10]: X = df.iloc[:, :4].values
```

```
In [11]: y = df['species'].values
```

```
In [23]: X.shape
```

Out[23]: (150, 4)

In [24]: `y.shape`

Out[24]: (150,)

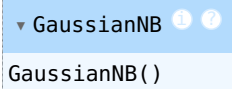
In [26]: `from sklearn.preprocessing import StandardScaler`
`scaler = StandardScaler()`
`x = scaler.fit_transform(X)`

In [27]: `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, random_state=42)`

In [28]: `x_train.shape, x_test.shape, y_train.shape, y_test.shape`

Out[28]: ((120, 4), (30, 4), (120,), (30,))

In [29]: `from sklearn.naive_bayes import GaussianNB`
`gaussian = GaussianNB()`
`gaussian.fit(x_train, y_train)`

Out[29]: 

In [34]: `Y_pred = gaussian.predict(x_test)`

In [35]: `gaussian.score(x_test, y_test)`

Out[35]: 1.0

In [46]: `from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix`
`accuracy_score(y_test, Y_pred)`

Out[46]: 1.0

In [47]: `precision = precision_score(y_test, Y_pred, average = 'micro')`
`print(precision)`
1.0

In [48]: `recall = recall_score(y_test, Y_pred, average = 'micro')`
`print(recall)`
1.0

In [51]: `cm = confusion_matrix(y_test, Y_pred)`
`print(cm)`
[[10 0 0]
 [0 9 0]
 [0 0 11]]

In [58]: `def get_confusion_matrix_values(y_test, y_pred):`
`cm = confusion_matrix(y_test, y_pred)`
`return(cm[0][0], cm[0][1], cm[1][0], cm[1][1])`

In [61]: `TP, FP, FN, TN = get_confusion_matrix_values(y_test, Y_pred)`
`print("TP : ", TP)`
`print("FP : ", FP)`
`print("FN : ", FN)`
`print("TN : ", TN)`
TP : 10
FP : 0
FN : 0
TN : 9

In [65]: `print("The accuracy is : ", (TP+TN)/(TP+TN+FP+FN))`
`print("The precision is : ", TP/(TP+FP))`
`print("The recall is : ", TP/(TP+FN))`
`print('Error Rate: ', (FP+FN)/(TP+TN+FP+FN))`
The accuracy is : 1.0
The precision is : 1.0
The recall is : 1.0
Error Rate: 0.0

In [64]: `F_measure = 2 * recall * precision / recall + precision`
`print(F_measure)`
3.0

