```
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
```

```
df=pd.read_csv('/IRIS.csv')
df
```

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

150 rows × 5 columns

```
print(df.head())
print(df.info())
# Step 3: View basic information
```

```
   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
<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
None
```

```
# Step 4: Split into features and target
X = df.drop('species', axis=1)
y = df['species']
```

```
# Step 5: Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
# Step 6: Initialize and train Naïve Bayes model
model = GaussianNB()
model.fit(X_train, y_train)
```

```
▾ GaussianNB
GaussianNB()
```

```
# Step 7: Predict on test data
y_pred = model.predict(X_test)
```

```
# Step 8: Confusion Matrix
cm = confusion_matrix(y_test, y_pred, labels=model.classes_)
print("Confusion Matrix:\n", cm)
```

```
Confusion Matrix:
 [[19  0  0]
 [ 0 12  1]
 [ 0  0 13]]
```

```
# Step 9: Evaluation Metrics
accuracy = accuracy_score(y_test, y_pred)
error_rate = 1 - accuracy
precision = precision_score(y_test, y_pred, average='macro')  # macro = class-wise average
recall = recall_score(y_test, y_pred, average='macro')
f1 = f1_score(y_test, y_pred, average='macro')
```

```
print("\nEvaluation Metrics:")
print(f"Accuracy      : {accuracy:.2f}")
print(f"Error Rate    : {error_rate:.2f}")
print(f"Precision     : {precision:.2f}")
print(f"Recall        : {recall:.2f}")
print(f"F1 Score      : {f1:.2f}")
```

```
Evaluation Metrics:
Accuracy      : 0.98
Error Rate    : 0.02
Precision     : 0.98
Recall        : 0.97
F1 Score      : 0.97
```