```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, accuracy_score,
precision_score, recall_score

df=pd.read_csv(r"C:\Users\ASUS\Documents\pythonStack\DS_PR\IRIS.csv")
df.head()
```

	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

```python
x = df.drop('species',axis=1)
y = df['species']

encoder = LabelEncoder()
#Initializes the encoder. This is an object that will be used to
encode the target
#labels (i.e., species names) into numeric values.

ytrans = encoder.fit_transform(y)
#Transforms the original categorical labels (y) into numeric values
based on the learned mapping from fit().
#This means that the species names are converted to integers. For
example:
#'Iris-setosa' → 0
#'Iris-versicolor' → 1
#'Iris-virginica' → 2

ytrans
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

```
X_train, X_test, y_train, y_test = train_test_split(X, ytrans,
test_size=0.2, random_state=42)

model = GaussianNB()        # Create Naïve Bayes model
model.fit(X_train, y_train) # Train model on training data

GaussianNB()

y_pred = model.predict(X_test)
# ➤ Predict the class/species for each sample in the test set

cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)
# ➤ Shows how many predictions were correct/incorrect
# ➤ Each row = actual class, each column = predicted class

Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]

#  Confusion Matrix Explanation:
# Model predicted all 30 test samples correctly.
# Rows = actual class, columns = predicted class.
# Diagonal values (10, 9, 11) = correct predictions.
# Off-diagonal values (all 0) = no mistakes.
#  Accuracy = 100%, no errors.

# Accuracy = Total correct predictions / Total predictions
# Measures how often the classifier is correct overall
accuracy = accuracy_score(y_test, y_pred)

accuracy

1.0

#Measures how often the classifier is wrong
error_rate = 1 - accuracy

# average=None calculates precision/recall for each class separately,
without averaging across classes.
# Without average=None, precision/recall is calculated for each class
separately
precision = precision_score(y_test, y_pred, average=None)
# Output: precision per class, e.g., [0.9, 0.8, 1.0] for each class
print("Precision per class:", precision)

# With average='macro', precision/recall is averaged across all
classes
precision_macro = precision_score(y_test, y_pred, average='macro')
# Output: average precision across all classes, e.g., 0.9
print("Average Precision:", precision_macro)
```

```
Precision per class: [1. 1. 1.]
Average Precision: 1.0
```

```
# Recall = True Positives / (True Positives + False Negatives)
# Measures how many of the actual positives were correctly predicted
recall = recall_score(y_test, y_pred, average='macro')  # without average='macro'

recall
```

```
1.0
```