pdc4pyvyr

April 29, 2025

```python
[1]: import pandas as pd
     import numpy as np
     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
```

```python
[2]: # Load the dataset
     df = pd.read_csv('Iris.csv')
```

```python
[3]: # Split the dataset into features and labels
     X = df.drop('Species', axis=1)
     y = df['Species']
```

```python
[4]: print(X)
```

```
      Id  SepalLengthCm  SepalWidthCm  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

[150 rows x 5 columns]
```

```python
[5]: print(y)
```

```
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
Name: Species, Length: 150, dtype: object
```

[6]:
```python
# Split the dataset into training and testing sets

"""
""
The train_test_split function of the sklearn.model_selection package in Python␣
 ↪splits arrays or matrices into random subsets for train and test data,␣
 ↪respectively.
"""

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[7]:
```python
# Create a Gaussian Naive Bayes classifier
classifier = GaussianNB()
```

[8]:
```python
# Train the classifier
classifier.fit(X_train, y_train)
```

[8]:
```
GaussianNB()
```

[9]:
```python
# Make predictions on the test set
y_pred = classifier.predict(X_test)
```

[10]:
```python
# Compute the confusion matrix

"""
A confusion matrix is a matrix that summarizes the performance of a machine␣
 ↪learning model on a set of test data. It is often used to measure the␣
 ↪performance of classification models, which aim to predict a categorical␣
 ↪label for each input instance.
"""

confusion_mat = confusion_matrix(y_test, y_pred)
print("Confusion Matrix : ")
print(confusion_mat)
```

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

```python
[11]: # Extract TN, FP, FN, TP from the confusion matrix

      """
      True Positive (TP): It is the total counts having both predicted and actual␣
       ↪values are Dog.
      True Negative (TN): It is the total counts having both predicted and actual␣
       ↪values are Not Dog.
      False Positive (FP): It is the total counts having prediction is Dog while␣
       ↪actually Not Dog.
      False Negative (FN): It is the total counts having prediction is Not Dog while␣
       ↪actually, it is Dog.
      """

      tn, fp, fn, tp = confusion_mat[0, 0], confusion_mat[0, 1], confusion_mat[1, 0],␣
       ↪confusion_mat[1, 1]
```

```python
[12]: # Compute evaluation metrics
      accuracy = (tp + tn) / (tp + tn + fp + fn)
      error_rate = 1 - accuracy
      precision = tp / (tp + fp)
      recall = tp / (tp + fn)
```

```python
[13]: # Print evaluation metrics
      print("Accuracy:", accuracy)
      print("Error Rate:", error_rate)
      print("Precision:", precision)
      print("Recall:", recall)
```

```
Accuracy: 1.0
Error Rate: 0.0
Precision: 1.0
Recall: 1.0
```