February 21, 2025

[1]:
```python
# Step 1: Import necessary libraries
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
```

[2]:
```python
# Step 2: Load the Iris dataset
# Using sklearn's built-in dataset for Iris
from sklearn.datasets import load_iris
iris = load_iris()

# Convert to DataFrame for ease of use
data = pd.DataFrame(data=iris.data, columns=iris.feature_names)
data['species'] = iris.target

# Display first few rows of the dataset
data.head()
```

[2]:
```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
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

   species
0        0
1        0
2        0
3        0
4        0
```

[3]:
```python
# Step 3: Split the data into training and testing sets
X = data[iris.feature_names]      # Features
y = data['species']     # Target variable
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)

# Display shapes of the training and test sets
print(f"Training data shape: {X_train.shape}")
print(f"Test data shape: {X_test.shape}")
```

```
Training data shape: (105, 4)
Test data shape: (45, 4)
```

[4]:

```
# Step 4: Train the Naïve Bayes classifier
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)

# Confirm model training
print("Model trained successfully.")
```

```
Model trained successfully.
```

[5]:

```
# Step 5: Make predictions
y_pred = nb_model.predict(X_test)

# Display first few predictions
y_pred[:10]
```

[5]:

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

[6]:

```
# Step 6: Compute the Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
```

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

[7]:

```
# Step 7: Calculate additional metrics
TP = cm[0][0] # True Positive
FP = cm[0][1] # False Positive
TN = cm[1][1] # True Negative
FN = cm[1][0]    # False Negative

# Accuracy
accuracy = accuracy_score(y_test, y_pred)

# Error rate
```

```python
error_rate = 1 - accuracy

# Precision
precision = precision_score(y_test, y_pred, average='weighted')

# Recall
recall = recall_score(y_test, y_pred, average='weighted')

# F1 Score (Optional but a good metric)
f1 = f1_score(y_test, y_pred, average='weighted')

# Print metrics
print(f"Accuracy: {accuracy:.4f}")
print(f"Error Rate: {error_rate:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1 Score: {f1:.4f}")

# Output TP, FP, TN, FN
print(f"TP: {TP}, FP: {FP}, TN: {TN}, FN: {FN}")
```

```
Accuracy: 0.9778
Error Rate: 0.0222
Precision: 0.9794
Recall: 0.9778
F1 Score: 0.9777
TP: 19, FP: 0, TN: 12, FN: 0
```