

```
In [100... import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
```

```
In [102... df = pd.read_csv("Iris.csv")

if 'Id' in df.columns:
    df = df.drop(columns=['Id'])

df.head()
```

```
Out[102...      SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  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 [104... X = df.iloc[:, [0,1]].values  # SepalLength & SepalWidth
y = df.iloc[:, -1].values
```

```
In [106... from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
y = le.fit_transform(y)
```

```
In [108... X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=0
)
```

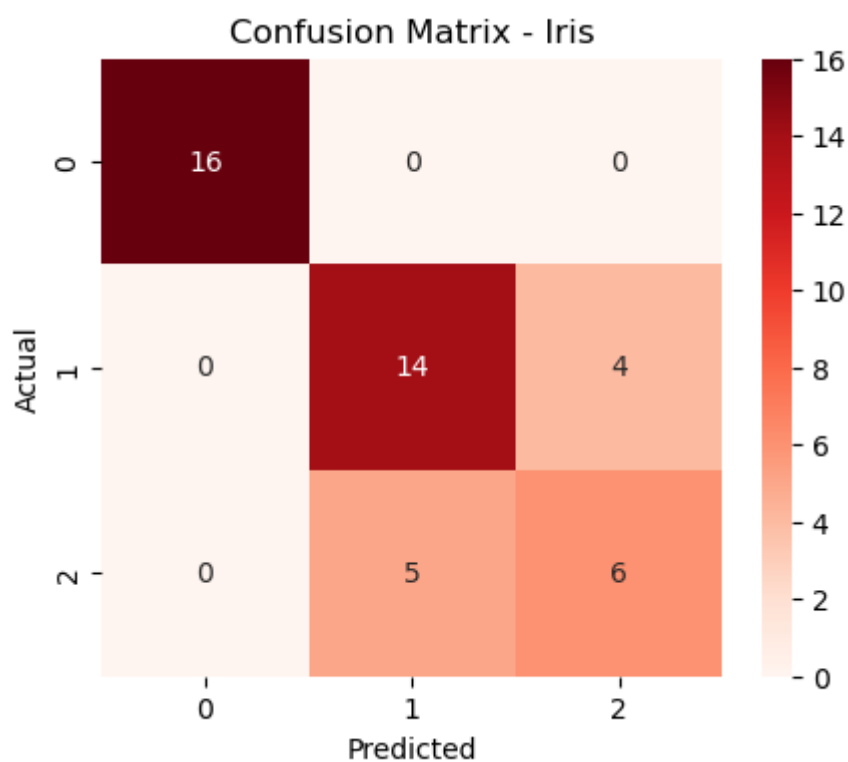
```
In [110... model = GaussianNB()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
```

```
In [112... cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)
```

```
Confusion Matrix:
[[16  0  0]
 [ 0 14  4]
 [ 0  5  6]]
```

```
In [114... plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Reds')
plt.title("Confusion Matrix - Iris")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



```
In [116... for i in range(len(cm)):  
    TP = cm[i, i]  
    FN = sum(cm[i, :]) - TP  
    FP = sum(cm[:, i]) - TP  
    TN = sum(sum(cm)) - (TP + FP + FN)  
  
    print(f"\nClass {i}")  
    print("TP:", TP)  
    print("FP:", FP)  
    print("FN:", FN)  
    print("TN:", TN)
```

Class 0

TP: 16

FP: 0

FN: 0

TN: 29

Class 1

TP: 14

FP: 5

FN: 4

TN: 22

Class 2

TP: 6

FP: 4

FN: 5

TN: 30

```
In [118... accuracy = accuracy_score(y_test, y_pred)  
error_rate = 1 - accuracy  
  
print("\nAccuracy:", accuracy)  
print("Error Rate:", error_rate)
```

Accuracy: 0.8

Error Rate: 0.19999999999999996

```
In [120... print("\nClassification Report:\n")  
print(classification_report(y_test, y_pred))
```

# Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	16
1	0.74	0.78	0.76	18
2	0.60	0.55	0.57	11
accuracy			0.80	45
macro avg	0.78	0.77	0.78	45
weighted avg	0.80	0.80	0.80	45

In [122...

```
# Decision Boundary with Legend

X_set, y_set = X_train, y_train

X1, X2 = np.meshgrid(
    np.arange(X_set[:, 0].min()-1, X_set[:, 0].max()+1, 0.01),
    np.arange(X_set[:, 1].min()-1, X_set[:, 1].max()+1, 0.01)
)

plt.figure(figsize=(7,6))

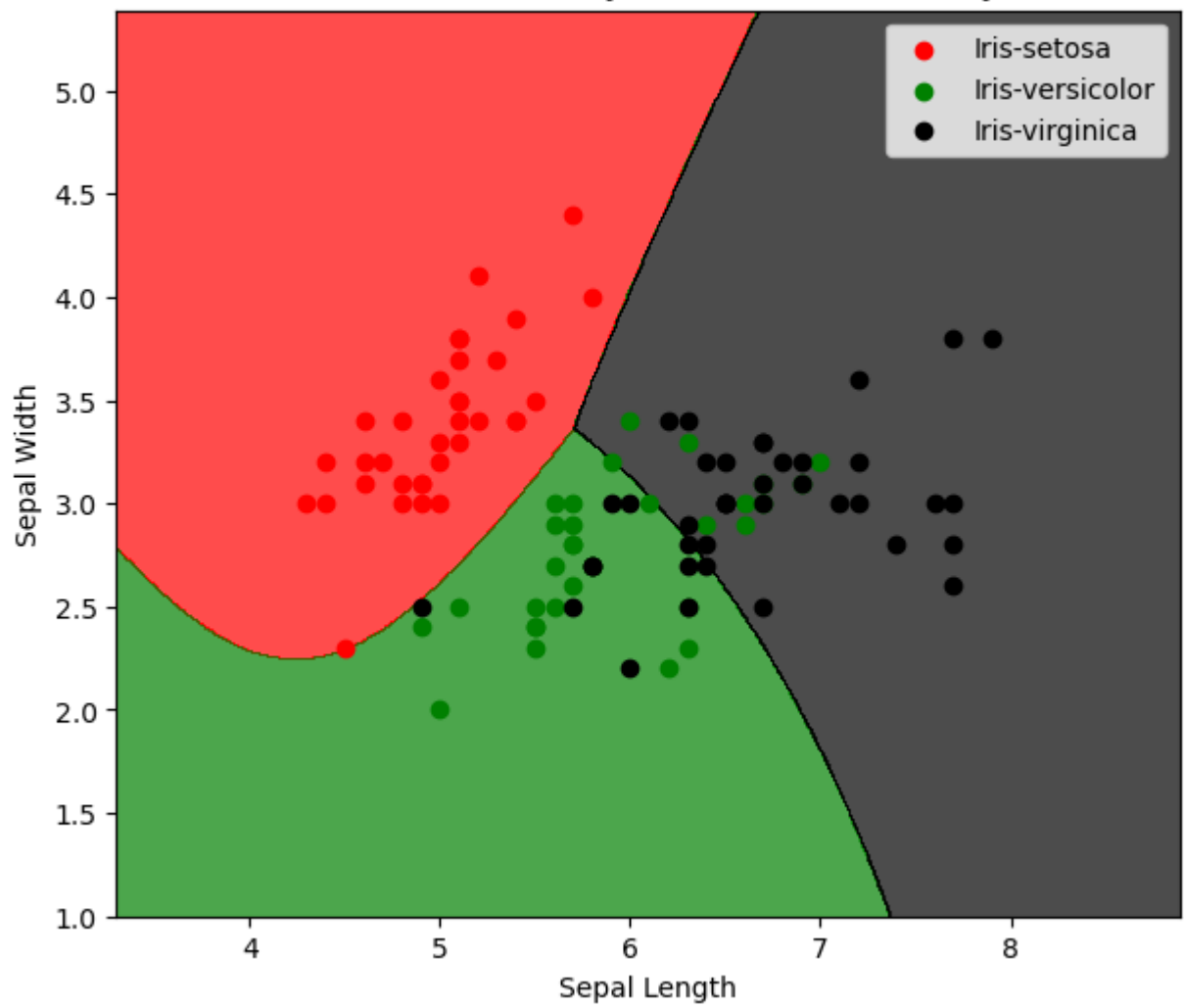
# Background prediction regions
plt.contourf(
    X1, X2,
    model.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
    alpha=0.7,
    cmap=ListedColormap(('red', 'green', 'black'))
)

# Plot each class separately for proper legend
colors = ('red', 'green', 'black')
labels = le.classes_

for i, color in enumerate(colors):
    plt.scatter(
        X_set[y_set == i, 0],
        X_set[y_set == i, 1],
        c=color,
        label=labels[i]
    )

plt.title("Gaussian Naïve Bayes - Decision Boundary")
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.legend()
plt.show()
```

Gaussian Naïve Bayes - Decision Boundary



In [ ]: