ass6_mnish_iris

February 10, 2026

[1]:
```
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
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



iris = load_iris()

# Convert to DataFrame
data = pd.DataFrame(iris.data, columns=iris.feature_names)
data['target'] = iris.target

# Class names
class_names = iris.target_names

# Split features and target
X = data.iloc[:, :-1]   # all feature columns
y = data.iloc[:, -1]    # class label

# Train-test split (70% train, 30% test)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# Train Naïve Bayes model
model = GaussianNB()
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)
```

[2]:
```
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)
```

```
# [[TP1  FP12 FP13]
#  [FP21 TP2  FP23]
#  [FP31 FP32 TP3 ]]
```

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

[3]:
```
accuracy = accuracy_score(y_test, y_pred)
error_rate = 1 - accuracy

print("Accuracy:", accuracy)
print("Error Rate:", error_rate)
```

```
Accuracy: 0.9777777777777777
Error Rate: 0.022222222222222254
```

[4]:
```
precision = precision_score(y_test, y_pred, average=None)
recall = recall_score(y_test, y_pred, average=None)

classes = model.classes_

for i, cls in enumerate(classes):
    print(f"\nClass: {cls}")
    print("Precision:", precision[i])
    print("Recall:", recall[i])
```

```
Class: 0
Precision: 1.0
Recall: 1.0

Class: 1
Precision: 1.0
Recall: 0.9230769230769231

Class: 2
Precision: 0.9285714285714286
Recall: 1.0
```

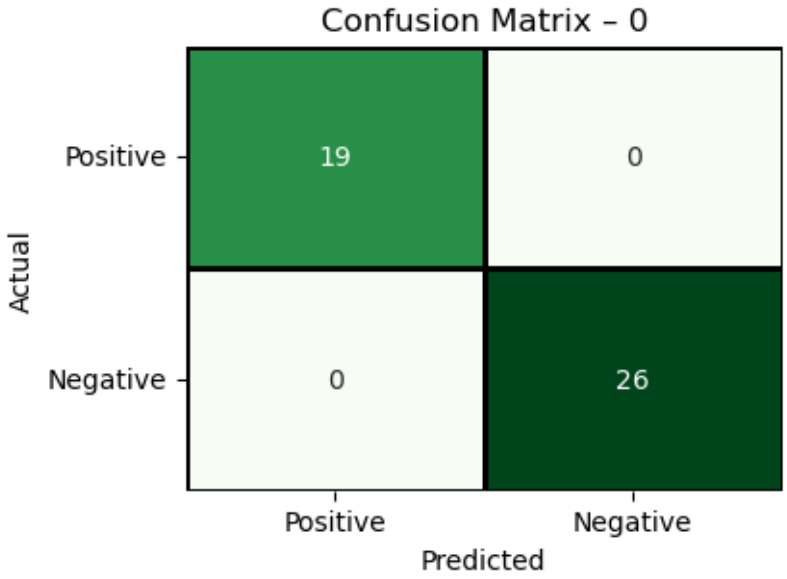
[5]:
```
for i, cls in enumerate(classes):
    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = cm.sum() - (TP + FP + FN)
```

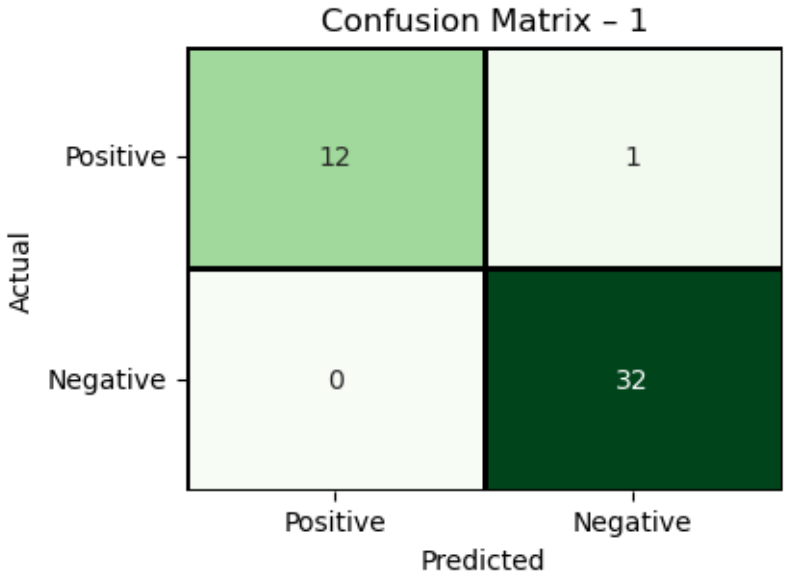
```
    print(f"\nClass: {cls}")
    print("TP:", TP)
    print("FP:", FP)
    print("FN:", FN)
    print("TN:", TN)
```

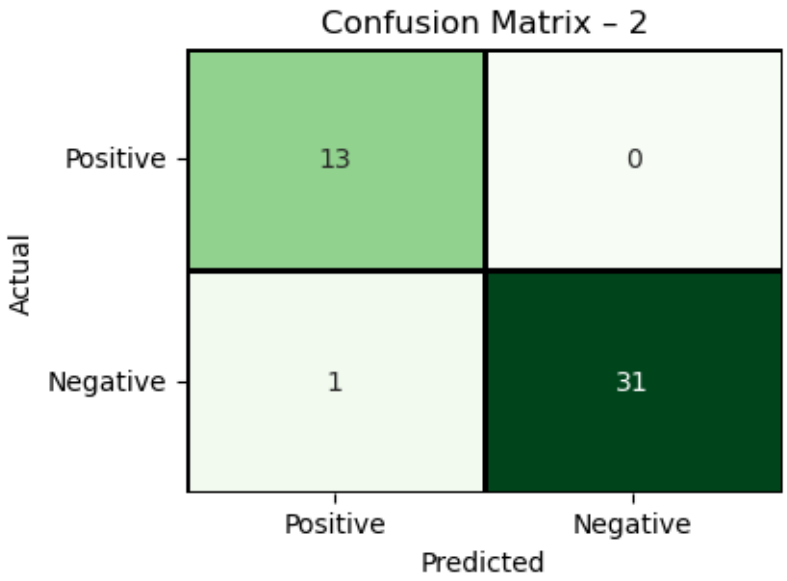
```
Class: 0
TP: 19
FP: 0
FN: 0
TN: 26

Class: 1
TP: 12
FP: 0
FN: 1
TN: 32

Class: 2
TP: 13
FP: 1
FN: 0
TN: 31
```

[17]:
```
for i, cls in enumerate(classes):
    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = cm.sum() - (TP + FP + FN)

    class_cm = [[TP, FN],
                [FP, TN]]

    plt.figure(figsize=(4,3))
    sns.heatmap(
        class_cm,
        annot=True,
        fmt="d",
        cmap="Greens",
        cbar=False,
        linewidths=1,
        linecolor="black"
    )
    plt.title(f"Confusion Matrix - {cls}")
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
```

```
    plt.xticks([0.5, 1.5], ["Positive", "Negative"])
    plt.yticks([0.5, 1.5], ["Positive", "Negative"], rotation=0)
    plt.show()
```






Confusion Matrix – 2
Positive
Negative
Actual
13
0
1
31
Positive
Negative
Predicted

[ ]: