week6-b

September 13, 2024

[10]:
```
import numpy as np

def confusion_matrix(y_true, y_pred):
    classes = np.unique(y_true)
    matrix = np.zeros((len(classes), len(classes)), dtype=int)
    for i in range(len(y_true)):
        matrix[classes == y_true[i], classes == y_pred[i]] += 1
    return matrix

def accuracy(y_true, y_pred):
    return np.sum(np.array(y_true) == np.array(y_pred)) / len(y_true)

def precision(y_true, y_pred):
    matrix = confusion_matrix(y_true, y_pred)
    return np.diag(matrix) / np.sum(matrix, axis=0)

def recall(y_true, y_pred):
    matrix = confusion_matrix(y_true, y_pred)
    return np.diag(matrix) / np.sum(matrix, axis=1)

def f1_score(y_true, y_pred):
    p = precision(y_true, y_pred)
    r = recall(y_true, y_pred)
    return 2 * (p * r) / (p + r)

y_true = list(map(int, input("Enter the true labels (comma-separated): ").
 ↪split(',')))
y_pred = list(map(int, input("Enter the predicted labels (comma-separated): ").
 ↪split(',')))

print("\nConfusion Matrix:")
conf_matrix = confusion_matrix(y_true, y_pred)
for row in conf_matrix:
    print(' '.join(map(str, row)))

print("\nAccuracy: {:.2f}".format(accuracy(y_true, y_pred)))
```

```
print("Precision: ", np.array2string(precision(y_true, y_pred), precision=2,␣
 ↪separator=', '))
print("Recall: ", np.array2string(recall(y_true, y_pred), precision=2,␣
 ↪separator=', '))
print("F1 Score: ", np.array2string(f1_score(y_true, y_pred), precision=2,␣
 ↪separator=', '))
```

```
Confusion Matrix:
3 0
0 6

Accuracy: 1.00
Precision:  [1., 1.]
Recall:  [1., 1.]
F1 Score:  [1., 1.]
```

- Created a confusion matrix to show true vs. predicted values.
- Calculated accuracy to measure overall correctness.
- Calculated precision to measure the correctness of positive predictions.
- Calculated recall to measure how well all positive cases are identified.
- Calculated F1 score to combine precision and recall into a single metric.