LAB - Tính toán các chỉ số đánh giá từ Ma trận Nhầm lẫn

Giới thiệu

Trong học máy và các bài toán phân loại, **ma trận nhầm lẫn** (Confusion Matrix) là một công cụ giúp đánh giá chất lượng của mô hình dự đoán. Ma trận này cho thấy kết quả dự đoán đúng và sai của mô hình.

Một ma trận nhầm lẫn cơ bản cho bài toán phân loại nhị phân có dạng:

$$\begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix}$$

Ý nghĩa các chỉ số trong ma trận nhầm lẫn

- **True Negative (TN)**: Số lượng mẫu thực tế là **âm** (Negative) và mô hình cũng dự đoán là **âm**.
- **False Positive (FP)**: Số lượng mẫu thực tế là **âm** nhưng mô hình lại dự đoán là **dương** (Positive). Đây còn gọi là **dương giả**.
- **False Negative (FN)**: Số lượng mẫu thực tế là **dương** nhưng mô hình lại dự đoán là **âm**. Đây còn gọi là **âm giả**.
- **True Positive (TP)**: Số lượng mẫu thực tế là **dương** và mô hình cũng dự đoán là **dương**.

Các giá trị này có thể tính toán được nhiều chỉ số quan trọng, giúp đánh giá mô hình một cách toàn diện.

Các Chỉ số Đánh giá Hiệu quả Mô Hình

1. Độ chính xác (Accuracy)

Độ chính xác là tỷ lệ số dự đoán đúng trên tổng số mẫu. Chỉ số này cho biết mô hình dự đoán chính xác bao nhiêu phần trăm.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

2. Độ nhạy (Recall)

Độ nhạy hay còn gọi là **Tỷ lệ phát hiện dương** (Sensitivity) cho biết mô hình phát hiện đúng bao nhiêu phần trăm các mẫu dương. Độ nhạy đặc biệt quan trọng khi chúng ta muốn giảm thiểu số trường hợp **âm giả**.

$$\text{Recall} = \frac{TP}{TP + FN}$$

Ví dụ, nếu đang xây dựng mô hình chẩn đoán bệnh, Recall cao có nghĩa là mô hình không bỏ sót quá nhiều ca bệnh.

Giải thích :

- recall là tỷ lệ phát hiện đúng các mẫu dương.
- Trong trường hợp chẩn đoán bệnh, các mẫu dương là các ca bệnh thực sự.
- recall cao có nghĩa là mô hình phát hiện được hầu hết các ca bệnh, tức là không bỏ sót nhiều ca bệnh

3. Độ đặc hiệu (Specificity)

Độ đặc hiệu là tỷ lệ dự đoán đúng các mẫu âm trên tổng số các mẫu âm thực tế. Chỉ số này cho biết mô hình có khả năng nhận diện đúng các mẫu âm tốt như thế nào.

$$\text{Specificity} = \frac{TN}{TN + FP}$$

Ví dụ, trong mô hình phát hiện gian lận, Specificity cao tránh được các trường hợp báo động sai (dương giả).

Giải thích:

- Specificity là tỷ lệ dự đoán đúng các mẫu âm
- Trong trường hợp phát hiện gian lận, các mẫu âm là các giao dịch không gian lận
- Specificity cao có nghĩa là mô hình nhận diện đúng hầu hết các giao dịch không gian lận, tức là giảm thiểu các trường hợp báo động sai (False Positive)

4. Giá trị dự đoán dương (Precision)

Precision hay còn gọi là **Độ chính xác của các dự đoán dương** là tỷ lệ dự đoán đúng trong số tất cả các mẫu được dự đoán là dương. Precision đặc biệt quan trọng khi chi phí của việc dương giả cao.

$$\text{Precision} = \frac{TP}{TP + FP}$$

Ví dụ, trong mô hình phát hiện ung thư, Precision cao đảm bảo rằng những trường hợp bị đánh dấu là dương (bệnh nhân mắc ung thư) có khả năng mắc bệnh thực sự.

Giải thích:

- Precision là tỷ lệ dự đoán đúng trong số tất cả các mẫu được dự đoán là dương
- Trong trường hợp phát hiện ung thư, các mẫu dương là các bệnh nhân được dự đoán mắc ung thư
- Precision cao có nghĩa là hầu hết các bệnh nhân được dự đoán mắc ung thư thực sự mắc bệnh, tức là giảm thiểu các trường hợp dương giả (False Positive)

5. F1-Score

F1 Score là trung bình điều hòa giữa Precision và Recall. Chỉ số này hữu ích khi cần cân

bằng giữa Recall và Precision, đặc biệt trong các bài toán mà một chỉ số cao hơn có thể dẫn đến một chỉ số khác bị giảm.

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

F1 Score giúp đánh giá mô hình với dữ liệu không cân bằng, chẳng hạn như khi số lượng mẫu dương và âm chênh lệch đáng kể.

Tóm tắt

Chỉ số	Công thức	Ý nghĩa
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$	Tỷ lệ dự đoán đúng trên tổng số mẫu
Recall	$\frac{TP}{TP + FN}$	Tỷ lệ phát hiện đúng các mẫu dương
Specificity	$\frac{TN}{TN + FP}$	Tỷ lệ phát hiện đúng các mẫu âm
Precision	$\frac{TP}{TP + FP}$	Tỷ lệ các dự đoán dương chính xác
F1-Score	$\frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$	Cân bằng giữa Precision và Recall

Bài tập nhẹ nhàng

Cho ma trận nhầm lẫn sau:

$$\begin{bmatrix} 50 & 10 \\ 5 & 30 \end{bmatrix}$$

1. Định nghĩa ma trận nhầm lẫn
2. Tính toán các chỉ số
3. In kết quả

In [4]:
```
# Định nghĩa ma trận nhầm lẫn
TN = 50 #code here
FP = 10#code here
FN =  5#code here
TP =  30#code here
```

```
# Tính toán các chỉ số
accuracy = (TN + TP)/sum([TN, FN, TP, FP])#code here
recall = (TP)/(FN+TP) #code here
specificity = (TN)/(TN+FP) #code here
precision = (TP)/(TP+FP)#code here
f1 = (2*precision*recall)/(precision*recall)#code here
# In kết quả
cf_matrix = [[TN, FP],\
             [FN, TP]]
print(f"Confusion Matrix:\n{cf_matrix}")#Code here
print(f"Accuracy: {accuracy  :.2f}")#Code here
print(f"Recall: {   recall:.2f}")#Code here
print(f"Specificity: {    specificity:.2f}")#Code here
print(f"Precision: {    precision:.2f}")#Code here
print(f"F1 Score: {    f1:.2f}")#Code here
```

```
Confusion Matrix:
[[50, 10], [5, 30]]
Accuracy: 0.84
Recall: 0.86
Specificity: 0.83
Precision: 0.75
F1 Score: 2.00
```

In [ ]:
```
# Nhận xét ở đây:
---
----
```

Bài tập nâng cao

Sủ dụng Markfown để viết ra 4 chỉ số tính độ chính xác, và viết định nghĩa cho các công thức sau:

1. Balanced Accuracy
2. Matthews Correlation Coefficient (MCC)
3. Fowlkes-Mallows Index (FMI)
4. Bias

Ứng dụng 4 chỉ số này để tính toán cho bài tập nhẹ nhàng ở trên

In [ ]:
```
#Code here
```

Bài tập vận dụng

In [5]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split

# Tạo dữ liệu giả định cho KNN
np.random.seed(42)
data_size = 1000
# Tạo các đặc trưng ngẫu nhiên giữa các lớp
X_class0 = np.random.multivariate_normal([2, 2], [[1.5, 0.75], [0.75, 1.5
```

```
X_class1 = np.random.multivariate_normal([4, 4], [[1.5, 0.75], [0.75, 1.5
X = np.vstack((X_class0, X_class1))
y = np.hstack((np.zeros(data_size // 2), np.ones(data_size // 2)))
# Chia dữ liệu thành tập huấn luyện và tập kiểm tra với test = 30 và rand
X_train, X_test, y_train, y_test = train_test_split(X, y)#code here
# Hiển thị một vài mẫu dữ liệu
print(pd.DataFrame(X_train[:5], columns=["Feature 1", "Feature 2"]))
print("Nhãn tương ứng:", y_train[:5])
```

```
   Feature 1  Feature 2
0   3.995138   2.704972
1  -0.820087   0.827710
2   1.900471   2.434856
3   2.846560   3.891776
4   4.570493   5.069028
Nhãn tương ứng: [1. 0. 1. 1. 1.]
```

In [10]:
```
def euclidean_distance(a, b):
    return (a[0] - b[0])**2 + (a[1] - b[1])**2 #code here
def knn_predict(X_train, y_train, X_test, k=5):
    y_pred = []
    for test_point in X_test:
        distances = [euclidean_distance(test_point, x) for x in X_train]
        k_indices = np.argsort(distances)[:k]
        k_nearest_labels = [y_train[i] for i in k_indices]
        most_common = max(set(k_nearest_labels), key=k_nearest_labels.cou
        #code here
        y_pred.append(most_common)
    return np.array(y_pred)
# Dự đoán trên tập kiểm tra với k = 5
y_pred_knn = knn_predict(X_train, y_train, X_test)
```

In [11]:
```
y_pred_knn
```

Out[11]:
```
array([1., 1., 1., 0., 0., 1., 1., 1., 0., 1., 0., 1., 1., 1., 1., 0.,
0.,
       1., 1., 0., 0., 0., 1., 0., 1., 0., 0., 1., 1., 0., 0., 0., 0.,
1.,
       0., 1., 1., 0., 1., 1., 1., 1., 0., 0., 0., 1., 1., 0., 0., 0.,
0.,
       0., 0., 0., 0., 1., 1., 1., 0., 0., 1., 1., 1., 0., 0., 0., 0.,
0.,
       1., 0., 0., 0., 1., 1., 0., 0., 1., 0., 1., 1., 0., 0., 0., 1.,
0.,
       0., 1., 1., 0., 0., 1., 1., 0., 1., 0., 0., 0., 1., 1., 0., 0.,
0.,
       0., 0., 1., 1., 1., 0., 0., 0., 1., 0., 1., 0., 0., 1., 0., 1.,
1.,
       0., 0., 0., 1., 0., 0., 0., 1., 1., 0., 0., 0., 1., 1., 0., 0.,
1.,
       1., 0., 1., 0., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.,
0.,
       0., 0., 0., 0., 1., 1., 0., 0., 0., 0., 1., 0., 1., 1., 1., 1.,
0.,
       1., 0., 0., 1., 0., 1., 1., 0., 1., 0., 0., 0., 1., 1., 1., 0.,
1.,
       0., 1., 0., 0., 1., 0., 1., 1., 1., 1., 1., 1., 0., 0., 0., 1.,
1.,
       0., 0., 0., 0., 0., 1., 0., 1., 0., 1., 0., 0., 1., 1., 1., 1.,
0.,
       1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 0., 0., 0., 1., 1., 1.,
0.,
       0., 0., 0., 0., 1., 1., 0., 0., 1., 0., 1., 1.])
```

In [12]:
```
y_test
```

Out[12]:
```
array([1., 1., 0., 1., 0., 0., 1., 0., 0., 1., 0., 1., 1., 1., 1., 1.,
0.,
       1., 1., 0., 0., 0., 0., 1., 1., 0., 0., 1., 1., 1., 0., 1., 0.,
1.,
       0., 1., 0., 0., 1., 0., 1., 1., 1., 0., 0., 0., 1., 0., 1., 0.,
0.,
       0., 0., 0., 0., 1., 1., 0., 0., 0., 1., 1., 1., 0., 0., 0., 0.,
0.,
       1., 1., 0., 1., 1., 1., 0., 0., 0., 0., 1., 1., 0., 0., 0., 1.,
0.,
       1., 0., 0., 0., 1., 0., 1., 0., 1., 0., 0., 0., 0., 1., 0., 0.,
0.,
       0., 1., 1., 1., 1., 0., 0., 0., 1., 0., 1., 0., 1., 1., 1., 1.,
0.,
       0., 0., 0., 1., 1., 0., 0., 1., 1., 0., 0., 0., 1., 0., 1., 0.,
0.,
       0., 0., 1., 0., 1., 1., 0., 0., 0., 0., 1., 0., 1., 1., 0., 1.,
0.,
       0., 0., 0., 0., 0., 1., 1., 0., 0., 0., 1., 1., 1., 1., 0., 1.,
0.,
       1., 0., 0., 1., 0., 1., 0., 0., 0., 0., 0., 1., 1., 1., 0., 0.,
1.,
       0., 0., 1., 0., 1., 0., 1., 1., 1., 1., 0., 1., 0., 0., 0., 0.,
1.,
       0., 1., 1., 0., 0., 1., 0., 0., 0., 1., 0., 0., 0., 0., 1., 1.,
1.,
       1., 1., 0., 1., 1., 1., 0., 1., 1., 1., 0., 0., 0., 1., 1., 1.,
0.,
       0., 0., 1., 0., 0., 1., 0., 0., 1., 0., 0., 1.])
```

In [ ]:
```python
# Định nghĩa hàm confusion_matrix
def confusion_matrix(y_true, y_pred):
    n = len(y_true)
    TP = sum([ 1 if (y_true[x]==y_pred[x] == 1.) else 0 for x in range(n)
    TN = sum([ 1 if (y_true[x]==y_pred[x] == 0.) else 0 for x in range(n)
    FP = sum([ 1 if (y_true[x] == y_pred[x] == 1.) else 0 for x in range(
    FN = sum([ 1 if (y_true[x]==y_pred[x] == 1.) else 0 for x in range(n)
    return #code here

# Hàm tính toán và in các chỉ số
def evaluate_model(y_test, y_pred):
    cm = confusion_matrix(y_test, y_pred)
    TN, FP, FN, TP = cm.ravel()
    accuracy = #code here
    recall = #code here
    specificity =#code here
    precision = #code here
    f1 = #code here
    print(f"Confusion Matrix:\n{cm}")
    print(f"Accuracy: {accuracy:.2f}")
    print(f"Recall: {recall:.2f}")
    print(f"Specificity: {specificity:.2f}")
    print(f"Precision: {precision:.2f}")
    print(f"F1 Score: {f1:.2f}")
# Đánh giá mô hình KNN
print("KNN Model Evaluation:")
#code here
```

```
KNN Model Evaluation:
Confusion Matrix:
[[116  34]
 [ 16 134]]
Accuracy: 0.83
Recall: 0.89
Specificity: 0.77
Precision: 0.80
F1 Score: 0.84
```

Bài tập về nhà

Tải tập dữ liệu Wine từ sklearn.datasets và chia tập dữ liệu theo tỷ lệ 70:30. Xây dựng mô hình KNN để phân loại dữ liệu. Sử dụng k = 5. Tính toán và in ra độ chính xác, recall, và precision của mô hình

Xây dựng website để trực quan hóa kết quả và độ chính xác

In [ ]:
```
#Code here
```

In [ ]:
```
#Code here
```

In [ ]:
```
#Code here
```

In [ ]:
```
#Code here
```