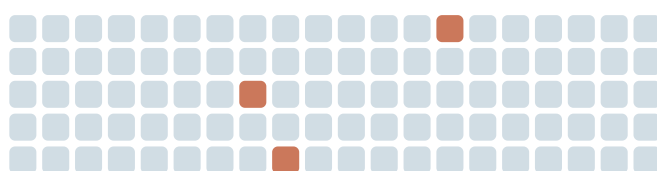


Classification metrics

☑ Trắc nghiệm 40 📧 Thực hành 11 🔗 Chạy trên Kaggle ⬇ PDF bài

Câu lạc bộ chỉ có vài buổi cần thêm người hỗ trợ, và muốn model báo trước những buổi ấy. Trong 100 buổi chỉ có ba buổi cần hỗ trợ, nên một model lúc nào cũng đoán “không cần” vẫn đúng 97 buổi. Accuracy 97% nghe tốt, nhưng model bỏ sót cả ba buổi quan trọng.



100 buổi, 3 buổi cần hỗ trợ

luôn đoán “không cần”

accuracy **0.97**

recall **0**

F1 **0**

Đoán toàn lớp đồng: accuracy 0.97 mà bỏ sót cả ba buổi cần hỗ trợ.

Vì vậy phải xem model sai kiểu nào: bỏ sót buổi cần hỗ trợ, hay báo nhầm buổi không cần. Bài này học confusion matrix, precision, recall, F1, ROC-AUC và cách chọn ngưỡng xác suất.

MỤC TIÊU

- Tính confusion matrix, accuracy, precision, recall, F1.
- Hiểu ROC-AUC và khi nào dùng nó.
- Chọn ngưỡng trên validation.
- Xử lý dữ liệu mất cân bằng lớp; dùng macro và weighted average.

Bốn ô trước một con số tổng

Quy ước lớp dương (positive) là **cần hỗ trợ**, nhãn 1. Lớp âm (negative), nhãn 0, là chưa cần hỗ trợ. Dương không có nghĩa tốt, âm không có nghĩa xấu; chúng là tên hai vai trò trong phép đo.

	đoán 0	đoán 1	
thật 0	TN 3	FP 3	báo nhầm
thật 1	FN 1	TP 3	bỏ sót

Mười buổi validation ở ngưỡng 0.5: hàng là nhãn thật, cột là dự đoán.

True positive (TP) là báo cần và thật sự cần. False positive (FP) là báo cần nhưng thật sự chưa cần. False negative (FN) là báo chưa cần nhưng thật sự cần. True negative (TN) là báo chưa cần và đúng là chưa cần.

Cách nhớ: chữ thứ hai (positive, negative) là điều model **báo**; chữ đầu (true, false) cho biết báo **đúng hay sai**. Mỗi buổi rơi vào đúng một ô, và tổng bốn ô bằng số buổi. Bảng bốn ô này gọi là **confusion matrix**: hai ô trên đường chéo (TP, TN) là đoán đúng, hai ô còn lại cho biết model nhầm lớp nào thành lớp nào. Trong scikit-learn, với `labels=[0, 1]`, nó được in theo thứ tự `[[TN, FP], [FN, TP]]`; tài liệu khác có thể vẽ theo chiều khác, nên luôn đọc nhãn hàng, cột trước.

Khi hai loại lỗi có giá khác nhau, nhân số lỗi với giá rồi so tổng: nếu một FP tốn 1 và một FN tốn 10, thì 20 FP và 30 FN tốn $20 + 300 = 320$.

HỎI NHANH

Mô hình báo buổi ấy không cần hỗ trợ, nhưng thực tế lại cần. Trường hợp này nằm ở ô nào?

A TP

B FN

C FP

D TN

Precision, recall và F1 nhìn những nhóm khác nhau

Accuracy đo tỉ lệ dự đoán đúng trên mọi mẫu:

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

Precision hỏi: trong những buổi model **đã báo cần**, bao nhiêu buổi thật sự cần? **Recall** hỏi: trong những buổi **thật sự cần**, model phát hiện được bao nhiêu?

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}.$$

Precision thấp thì người hỗ trợ phải đến nhiều buổi không cần. Recall thấp thì nhiều buổi cần hỗ trợ bị bỏ sót. Model luôn báo “cần” có recall bằng 1, nhưng precision chỉ bằng tỉ lệ buổi cần trong dữ liệu. Ngoài ra còn **specificity** = $TN / (TN + FP)$, tức recall của lớp âm.

Ưu tiên cái nào là tùy bài toán. Sàng lọc bệnh hay quét mã độc: bỏ sót rất đắt, còn báo nhầm thì đã có người kiểm lại, nên ưu tiên recall. Lọc thư rác mà vứt nhầm thư công việc, hay ẩn nhầm bình luận bình thường: báo nhầm gây hại, nên ưu tiên precision.

F1 là trung bình điều hòa của precision P và recall R :

$$F_1 = \frac{2PR}{P + R} = \frac{2TP}{2TP + FP + FN}.$$

Chỉ cần một trong hai thấp là F1 bị kéo thấp: $P = 0.5, R = 1$ cho $F_1 = 2 \cdot 0.5 \cdot 1 / 1.5 \approx 0.67$, thấp hơn trung bình cộng 0.75. F1 không dùng đến TN. Khi model không báo dương lần nào, precision có mẫu số 0; khi dữ liệu không có mẫu dương nào, recall có mẫu số 0. Code trong bài ghi rõ `zero_division=0` để những trường hợp đó tính là 0.

HỎI NHANH

Có TP = 3, FP = 3, FN = 1. Precision và recall lần lượt bằng bao nhiêu?

A 0.5 và 0.75

B 0.75 và 0.5

C 0.6 và 0.6

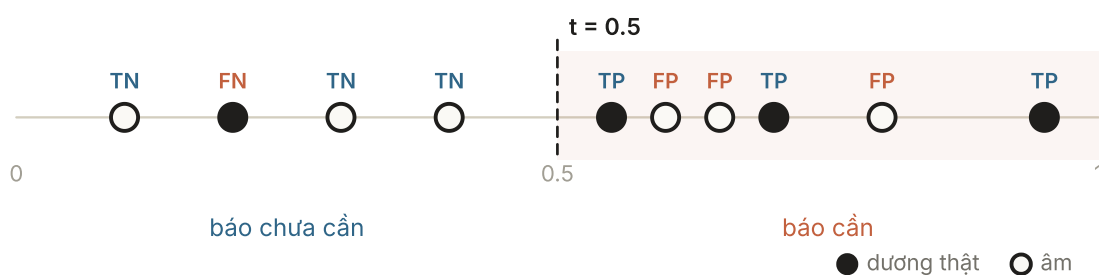
D 3/7 và 0.75

Một validation nhỏ để tính và đối chiếu

Giả sử mô hình đã trả xác suất cần hỗ trợ cho mười buổi validation. Ta cố ý dùng một mảng nhỏ để tính tay; code không huấn luyện mô hình trong bài này. Score lớn hơn nghĩa là model đánh giá khả năng cần hỗ trợ cao hơn. Threshold t biến score thành nhãn:

$$\hat{y}_i = \mathbf{1}[p_i \geq t].$$

$\mathbf{1}[\cdot]$ bằng 1 khi điều kiện đúng, bằng 0 khi sai. Quy tắc dùng dấu lớn hơn hay lớn hơn hoặc bằng phải thống nhất, nhất là khi score đúng bằng threshold.



Ngưỡng 0.5 chia mười score thành bốn ô của confusion matrix.

Python

```

import numpy as np
from sklearn.metrics import (
    confusion_matrix, accuracy_score, precision_score, recall_score, f1_score,
)

y_valid = np.array([1, 0, 1, 0, 0, 1, 0, 0, 1, 0])
p_valid = np.array([0.95, 0.80, 0.70, 0.65, 0.60, 0.55, 0.40, 0.30, 0.20, 0.10])
prediction = (p_valid >= 0.5).astype(int)
cm = confusion_matrix(y_valid, prediction, labels=[0, 1])
tn, fp, fn, tp = cm.ravel()
print(cm)
print('TN, FP, FN, TP:', tn, fp, fn, tp)
print(f'Accuracy={accuracy_score(y_valid, prediction):.3f}')
print(f'Precision={precision_score(y_valid, prediction, zero_division=0):.3f}')
print(f'Recall={recall_score(y_valid, prediction, zero_division=0):.3f}')
print(f'F1={f1_score(y_valid, prediction, zero_division=0):.3f}')

```

KẾT QUẢ

```

[[3 3]
 [1 3]]
TN, FP, FN, TP: 3 3 1 3
Accuracy=0.600
Precision=0.500
Recall=0.750
F1=0.600

```

Ở ngưỡng 0.5, sáu buổi được báo cần; trong đó ba buổi thật sự cần. Có bốn buổi dương thật và ta tìm được ba. Vì vậy precision 0.5, recall 0.75, $F1 = 6 / (6 + 3 + 1) = 0.6$. Accuracy cũng bằng 0.6 ở dữ liệu này, nhưng hai metric trùng số chỉ là tình cờ.

HỎI NHANH

Vẫn mười score ở trên, giờ đổi ngưỡng thành $t = 0.7$. Precision và recall lần lượt bằng bao nhiêu?

A 1/2 và 2/3

B 2/3 và 1/2

C 3/4 và 3/4

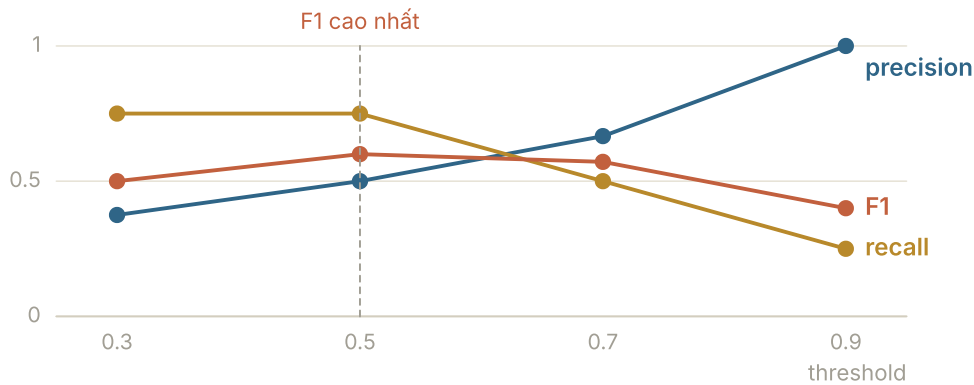
D 1 và 1/4

Quay lại ví dụ 100 buổi đầu bài: luôn đoán 0 cho $TP = 0$, $FN = 3$, $FP = 0$, $TN = 97$. Accuracy 0.97, nhưng recall và F1 đều bằng 0. Accuracy không tính sai, chỉ là nó không đo điều ta cần: tìm ra lớp hiếm.

Threshold là một lựa chọn phải đánh giá

Hạ threshold thì nhiều mẫu được báo dương hơn: recall không bao giờ giảm, còn precision lên hay xuống tùy mẫu mới được thêm vào là dương thật hay âm thật. Vì vậy không có công thức chung; ta thử vài ngưỡng trên validation và chọn theo metric của đề.

Nếu biết giá của hai loại lỗi và xác suất của model đáng tin, có một cách tính thẳng: báo dương khi chi phí kỳ vọng của việc không báo, $p \cdot c_{FN}$, lớn hơn chi phí kỳ vọng của việc báo, $(1 - p) \cdot c_{FP}$. Tức là báo dương khi $p > c_{FP} / (c_{FP} + c_{FN})$. Bỏ sót đất gấp 10 lần báo nhầm thì ngưỡng là $1/11 \approx 0.09$.



Tăng ngưỡng: precision lên, recall xuống; F1 cao nhất ở 0.5.

Python

```
thresholds = np.array([0.3, 0.5, 0.7, 0.9])
valid_f1 = []
for threshold in thresholds:
    prediction = p_valid >= threshold
    score = f1_score(y_valid, prediction, zero_division=0)
    valid_f1.append(score)
    print(f't={threshold:.1f}: F1={score:.3f}')
chosen_threshold = float(thresholds[np.argmax(valid_f1)])
print('Chọn trên validation:', chosen_threshold)
```

KẾT QUẢ

```
t=0.3: F1=0.500
t=0.5: F1=0.600
t=0.7: F1=0.571
t=0.9: F1=0.400
Chọn trên validation: 0.5
```

`argmax` chọn ngưỡng có F1 lớn nhất; nếu hòa thì lấy ngưỡng đầu tiên. Threshold cũng là một thứ ta chọn bằng dữ liệu, giống các tham số khác (bài 1.7 gọi là hyperparameter), dù nó không đổi trọng số nào của model.

Chọn xong, ta áp **đúng ngưỡng đó** lên một tập test riêng. Không thử lại bốn ngưỡng trên test rồi đổi ý.

Python

```
y_test = np.array([0, 1, 0, 0, 1, 0, 1, 0])
p_test = np.array([0.70, 0.80, 0.20, 0.40, 0.55, 0.10, 0.30, 0.65])
test_prediction = p_test >= chosen_threshold
```

```
print(f'Test F1 với ngưỡng đã chốt={f1_score(y_test, test_prediction,
zero_division=0):.3f}')
```

KẾT QUẢ

Test F1 với ngưỡng đã chốt=0.571

Điểm test khác validation vì là những mẫu khác. Nếu xem test xong lại thử ngưỡng khác rồi báo điểm tốt nhất, test đã thành một phần của việc chọn. Khi validation quá nhỏ, ngưỡng chọn được dễ chạy theo nhiễu; có thể chọn ngưỡng trên dự đoán OOF (bài 1.3) để có nhiều mẫu hơn.

HỎI NHANH

Trọng số của mô hình chỉ học trên train, nhưng ngưỡng phân lớp lại được chọn bằng nhãn test. Điểm trên test còn độc lập không?

- A Còn, vì không fit lại trọng số trên test
- B Còn, vì ngưỡng không phải tham số của mô hình
- C Không, vì ngưỡng đã được chọn bằng nhãn test
- D Không, vì phải dùng ngưỡng mặc định 0.5

ROC-AUC: thứ tự trước khi cắt ngưỡng

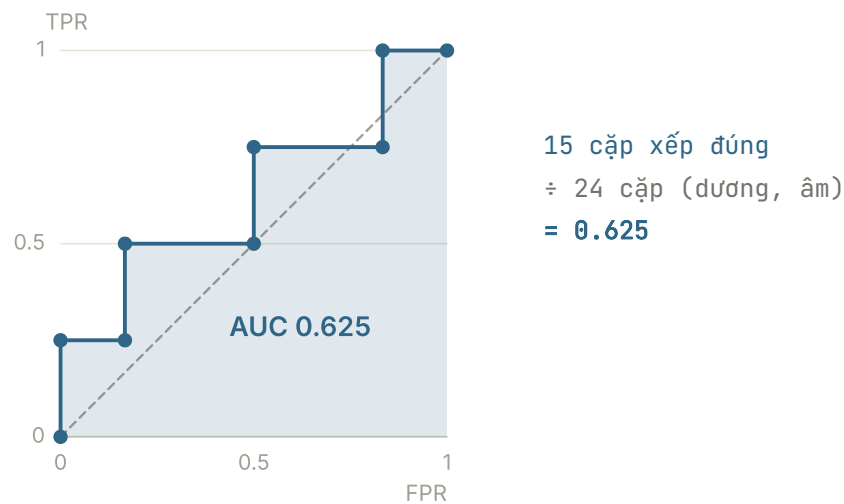
ROC dùng trục ngang false positive rate (FPR) và trục dọc true positive rate (TPR, chính là recall):

$$\text{FPR} = \frac{FP}{FP + TN}, \quad \text{TPR} = \frac{TP}{TP + FN}.$$

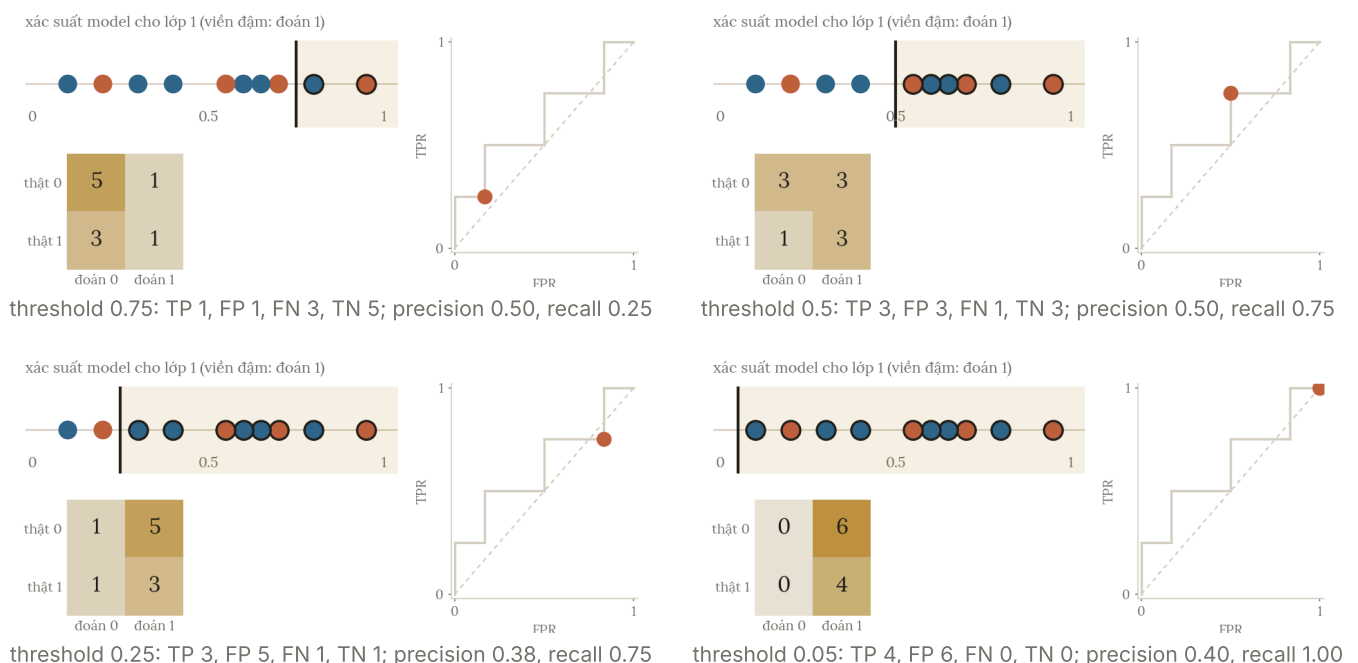
Hạ threshold từ rất cao xuống rất thấp, mỗi ngưỡng cho một điểm (FPR, TPR); nối các điểm lại được **đường ROC**. **ROC-AUC** là diện tích dưới đường đó.

Có một cách tính tay cho cùng kết quả: xét mọi cặp gồm một mẫu dương và một mẫu âm; cặp nào score của mẫu dương cao hơn thì được 1 điểm, bằng nhau được 0.5, thấp hơn được 0; AUC là trung bình số điểm. Vậy:

- AUC chỉ phụ thuộc **thứ tự** các score. Biến đổi score mà giữ nguyên thứ tự, như $\sigma(3s - 1)$, không đổi AUC.
- AUC bằng 0.5 là xếp hạng không hơn đoán bừa, bằng 1 là hoàn hảo. Nó không phải accuracy.



Diện tích dưới ROC bằng tỉ lệ cặp dương-âm được xếp đúng thứ tự.



Mười mẫu validation của bài khi hạ threshold từ 1 xuống 0: confusion matrix đổi theo, và điểm (FPR, TPR) chạy dọc đường ROC.

Với mảng validation, bốn score dương là 0.95, 0.70, 0.55, 0.20. Sáu score âm là 0.80, 0.65, 0.60, 0.40, 0.30, 0.10. Số cặp xếp đúng lần lượt là 6, 5, 3, 1: $AUC = 15 / (4 \cdot 6) = 0.625$.

Python

```
from sklearn.metrics import roc_auc_score, average_precision_score

positive_scores = p_valid[y_valid == 1]
negative_scores = p_valid[y_valid == 0]
pair_points = []
for positive in positive_scores:
    for negative in negative_scores:
        pair_points.append(float(positive > negative) + 0.5 * float(positive == negative))
print(f'AUC tính theo cặp={np.mean(pair_points):.3f}')
print(f'AUC thư viện={roc_auc_score(y_valid, p_valid):.3f}')
print(f'Average precision={average_precision_score(y_valid, p_valid):.3f}')
```

KẾT QUẢ

```
AUC tính theo cặp=0.625
AUC thư viện=0.625
Average precision=0.653
```

Hai dòng AUC trùng nhau, xác nhận cách đếm cặp. Với lớp hiếm, ROC có thể trông tốt dù số báo nhầm tuyệt đối vẫn lớn, vì FPR chia cho một số TN khổng lồ; khi ấy nên đọc thêm đường precision–recall và average precision (AP), một cách tổng hợp precision khi recall tăng dần.

AUC cần cả hai lớp và phải nhận score liên tục, không dùng nhãn đã cắt 0/1 để đo khả năng xếp hạng. Nếu một fold chỉ có một lớp, hãy sửa phép đánh giá hoặc ghi rõ metric không xác định ở đó, không lạng lẽ thay bằng 1.

HỎI NHANH

Hai mô hình xếp các mẫu theo đúng cùng một thứ tự score, chỉ khác nhau ở mức tự tin. ROC-AUC có phân biệt được hai mô hình này không?

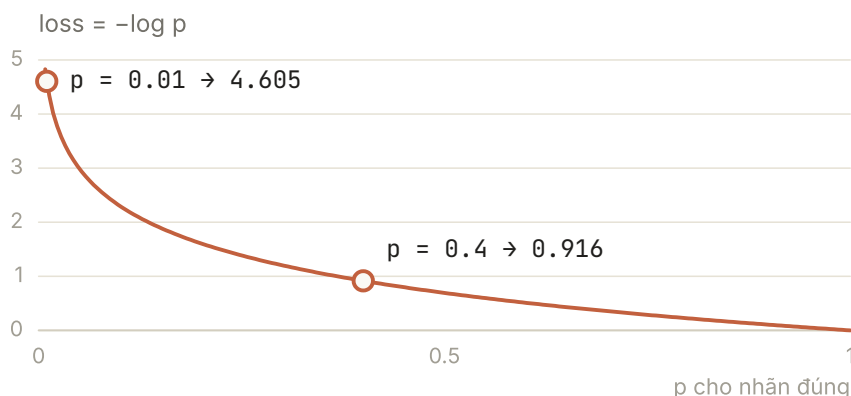
- A Có, vì AUC đo độ đúng của từng xác suất
- B Có, vì AUC là accuracy ở ngưỡng 0.5
- C Có, nếu một mô hình tự tin hơn hẳn
- D Không, vì AUC chỉ phụ thuộc thứ tự score

Xác suất và nhiều lớp

Log loss chấm mức xác suất chứ không chỉ nhãn. Với binary classification,

$$L = -\frac{1}{n} \sum_{i=1}^n [y_i \log p_i + (1 - y_i) \log(1 - p_i)].$$

log ở đây là logarithm tự nhiên. Đoán sai rất tự tin bị phạt nặng: với một mẫu dương, $p = 0.01$ cho loss $-\log 0.01 \approx 4.605$, còn $p = 0.4$ cho khoảng 0.916. Thư viện xử lý xác suất sát 0 hoặc 1 để tránh $\log(0)$; khi tự cài cần xem quy ước chính xác.



Sai càng tự tin càng bị phạt nặng: $p = 0.01$ mất 4.6, $p = 0.4$ chỉ mất 0.9.

Python

```
from sklearn.metrics import log_loss

truth = np.array([1, 0])
confident_wrong = np.array([0.01, 0.99])
less_confident_wrong = np.array([0.4, 0.6])
print(f'Sai rất tự tin: log loss={log_loss(truth, confident_wrong):.3f}')
print(f'Sai ít tự tin: log loss={log_loss(truth, less_confident_wrong):.3f}')
```

KẾT QUẢ

Sai rất tự tin: log loss=4.605
Sai ít tự tin: log loss=0.916

Ở ngưỡng 0.5, cả hai đều sai hai nhãn, nhưng log loss khác nhau rất xa.

Xác suất của model gọi là **hiệu chỉnh tốt** (well-calibrated) khi nó khớp tần suất thật: trong các mẫu được đoán khoảng 0.7, có khoảng 70% thật sự dương. Nếu trong 100 ca được đoán 0.9 mà chỉ 60 ca dương thật, model đang quá tự tin. Điều này phải kiểm trên nhiều mẫu, không nhìn một mẫu mà biết được. Hiệu chỉnh xác suất chỉ đổi các con số, không đổi thứ tự xếp hạng, nên không đổi AUC.

Với nhiều lớp, tính F1 cho từng lớp bằng cách coi lớp đó là dương, các lớp còn lại là âm. Trong confusion matrix nhiều lớp (hàng là nhãn thật, cột là dự đoán), precision của một lớp đọc theo cột của lớp ấy, recall đọc theo hàng. Rồi gộp lại thành một số theo một trong ba cách:

- **Macro-F1**: trung bình đều F1 của các lớp; lớp hiếm nặng ngang lớp đông.
- **Weighted-F1**: trung bình có trọng số theo số mẫu của từng lớp.
- **Micro-F1**: cộng TP, FP, FN của mọi lớp rồi mới tính; khi mỗi mẫu có đúng một nhãn, micro-F1 bằng accuracy.

Đề chấm macro-F1 thì báo macro-F1, đừng đổi sang weighted chỉ vì nó cao hơn.

HỎI NHANH

Bài phân loại 3 lớp, đề chấm macro-F1. Lớp A chiếm 60% số mẫu, lớp B 35%, lớp C 5%. F1 của mô hình trên từng lớp là 0.95, 0.90 và 0.20. Weighted-F1 khoảng 0.90, còn macro-F1 chỉ 0.68. Ta nên làm gì tiếp?

- A Báo weighted-F1 là 0.90, vì nó phản ánh phần lớn dữ liệu
- B Báo accuracy, vì nó không phụ thuộc cách lấy trung bình
- C Báo macro-F1 là 0.68, và tập trung cải thiện lớp C
- D Gộp lớp C vào lớp gần nhất cho điểm cao hơn

Lớp hiếm

Khi một lớp rất hiếm, model dễ lờ nó đi. Hai cách hay dùng:

- **class_weight**: lỗi trên lớp hiếm bị tính nặng hơn khi học, không thêm bớt mẫu nào. `class_weight='balanced'` đặt trọng số $n/(K \cdot n_c)$ cho lớp c , với n là tổng số mẫu, K là số lớp, n_c là số mẫu của lớp c . Ví dụ 900 và 100 mẫu cho trọng số 0.56 và 5.
- **Resampling**: nhân bản mẫu lớp hiếm hoặc bớt mẫu lớp đông. Chỉ làm trên train của mỗi fold; validation giữ nguyên tỉ lệ thật.

Cả hai đều không bảo đảm F1 tăng, nên vẫn phải đo trên validation và chọn lại ngưỡng.

HỎI NHANH

Bài phát hiện gian lận có 9800 giao dịch hợp lệ và 200 giao dịch gian lận. Mô hình để mặc định thì đoán gần như tất cả là hợp lệ. Trong bốn cách sửa sau, cách nào vừa dễ làm vừa hiệu quả nhất?

- A Dùng `class_weight` để lỗi trên lớp gian lận bị tính nặng hơn, giữ nguyên dữ liệu
- B Nhân bản 200 mẫu gian lận lên thành 9800 mẫu
- C Tạo thêm 9600 mẫu gian lận giả bằng cách pha trộn các mẫu thật
- D Bỏ bớt 9600 giao dịch hợp lệ, chỉ học trên 400 mẫu

ĐỀ THI NOAI 2026 Singapore · Vòng sơ loại · Câu 275

Khi đi thi

- Xem đề chấm metric nào: chấm ROC-AUC hay log loss thì nộp xác suất, đừng nộp nhãn 0/1.
- In confusion matrix và tỉ lệ lớp, không chỉ một con số tổng.
- Chọn threshold trên validation, rồi giữ nguyên threshold ấy khi dự đoán test.

Tóm tắt

- Precision: trong các cảnh báo, bao nhiêu đúng. Recall: trong các ca thật, phát hiện được bao nhiêu.
- F1 gộp precision và recall, không dùng TN.
- Chọn threshold trên validation; test dùng đúng ngưỡng đã chọn.

- ROC-AUC chấm thứ tự xếp hạng, log loss chấm xác suất, macro-F1 cho các lớp nặng ngang nhau.

Kiểm tra cuối bài

CÂU 1

Có 10 000 giao dịch, trong đó 200 gian lận. Sau khi dùng `class_weight`, mô hình cho TP = 180, FP = 294, FN = 20, TN = 9506. Mỗi vụ gian lận bị bỏ sót gây thiệt hại khoảng 500 đơn vị. Mỗi cảnh báo nhầm chỉ tốn 1 đơn vị cho một lần kiểm tra bằng tay. Mô hình này có dùng được không?

- A Rất tốt, vì accuracy là 96.9%, không cần xem gì thêm
- B Không tốt, vì precision chỉ 38%, báo nhầm quá nhiều
- C Chưa kết luận được khi chưa tính ROC-AUC
- D Dùng được, vì recall 90% và báo nhầm rẻ hơn bỏ sót rất nhiều

PHÒNG THEO [NOAI 2026 Singapore](#) · [Vòng sơ loại](#) · [Câu 276](#)

CÂU 2

Khi nào ta thường nên dùng F1 thay cho accuracy?

- A Khi dữ liệu rất lớn, tới hàng triệu mẫu
- B Khi số mẫu của các lớp gần bằng nhau
- C Khi mô hình là mô hình tuyến tính
- D Khi một lớp hiếm hơn hẳn lớp kia
- E Khi có rất nhiều feature

ĐỀ THI [USA-NA-AIO 2026 · Vòng 1 · Part 1.4](#)

CÂU 3

Có $TP = 8$, $FP = 4$, $FN = 2$. F1 của lớp dương bằng bao nhiêu?

- A 11/15
- B 8/11
- C 2/3
- D 4/5

CÂU 4

Một hệ thống có 98% giao dịch hợp lệ và 2% gian lận. Mô hình đoán tất cả là hợp lệ, nên accuracy là 98%. Bỏ sót gian lận thì tổn kém hơn báo nhầm. Ta kết luận được gì?

- A Accuracy cao vẫn chưa đủ: recall trên lớp gian lận bằng 0, phải tính thiệt hại của từng loại lỗi
- B Accuracy thật ra là 0%, vì mô hình không tìm được vụ gian lận nào
- C Recall trên lớp gian lận cũng là 98%, vì bài hai lớp thì recall luôn bằng accuracy
- D Accuracy 98% là đủ để kết luận mô hình dùng được

PHÒNG THEO NOAI 2026 Singapore · Vòng sơ loại · Câu 274

CÂU 5

Quy tắc ra nhãn là: gán nhãn 1 khi $p \geq t$, với ngưỡng $t = 0.7$. Một mẫu có xác suất lớp dương $p = 0.7$ nhận nhãn gì?

- A Nhãn 0, vì chỉ gán 1 khi $p > t$
- B Nhãn 1, vì $p \geq t$
- C Nhãn 0.7, vì quy tắc trả lại chính score
- D Nhãn 1, vì mọi xác suất từ 0.5 trở lên đều được gán 1

CÂU 6

Trên cùng tập validation, ta thử ba ngưỡng. Ngưỡng 0.3 cho $FP = 30$, $FN = 2$. Ngưỡng 0.5 cho $FP = 12$, $FN = 6$. Ngưỡng 0.7 cho $FP = 4$, $FN = 12$. Mỗi FP tốn 1 đơn vị, mỗi FN tốn 5 đơn vị, ngoài ra không có chi phí nào khác. Ta chọn ngưỡng nào?

- A 0.5, vì tổng số lỗi ít nhất
- B 0.7, vì FP ít nhất
- C Chưa chọn được khi chưa biết accuracy
- D 0.3, vì tổng chi phí là 40, thấp nhất

CÂU 7

Xét 20 cặp gồm một mẫu dương và một mẫu âm. Có 12 cặp mẫu dương có score cao hơn, 4 cặp bằng nhau, 4 cặp thấp hơn. Cặp bằng nhau được tính nửa điểm. ROC-AUC bằng bao nhiêu?

- A 0.80
- B 0.75
- C 0.70
- D 0.60

CÂU 8

Đã có `p = model.predict_proba(X)[ :, 1]`, nhãn dương là 1. Tính ROC-AUC bằng dòng nào?

- A `roc_auc_score(y, p >= 0.5)`
- B `roc_auc_score(y, p)`
- C `roc_auc_score(p >= 0.5, y)`
- D `roc_auc_score(y, p.mean() * np.ones_like(p))`

CÂU 9

Một mẫu có nhãn thật là 1. Mô hình A cho xác suất lớp 1 là 0.40, mô hình B cho 0.01. Ở ngưỡng 0.5, cả hai đều xếp mẫu này vào lớp 0. Theo log loss, mô hình nào bị phạt nặng hơn ở mẫu này?

- A B, vì $-\log(0.01)$ lớn hơn $-\log(0.40)$
- B A, vì 0.40 gần ngưỡng hơn
- C Bằng nhau, vì cùng đoán ra nhãn 0
- D B, vì log loss chỉ đếm số lần đoán lớp 0

CÂU 10

Ba lớp có F1 là 0.90, 0.30 và 0.60. Số mẫu thật của chúng lần lượt là 80, 10 và 10. Macro-F1 bằng bao nhiêu?

A 0.81

B 0.90

C 0.60

D 0.30

Đọc thêm

- [scikit-learn — Classification metrics](#). Tra averaging, ROC-AUC, AP và log loss theo đúng dạng đầu vào.
- [scikit-learn — Tuning the decision threshold](#). Đọc vì sao chọn threshold là một quyết định cần dữ liệu validation riêng.
- James và cộng sự — [An Introduction to Statistical Learning](#), chương 4. Đọc ý nghĩa score phân loại và các loại lỗi.