

Pipeline và data leakage

☑ Trắc nghiệm 18 📄 Thực hành 3 🔗 Chạy trên Kaggle ⬇ PDF bài

Trước khi đưa vào model, dữ liệu thường phải xử lý: điền ô trống, đổi cột chữ thành số, chuẩn hóa. Lỗi hay gặp là xử lý cả bảng rồi mới chia train/validation. Khi đó giá trị trung bình dùng để điền ô trống đã tính cả phần validation, nên điểm validation không còn đáng tin. Lỗi này gọi là data leakage (rò rỉ dữ liệu). Pipeline của scikit-learn ghép các bước xử lý với model thành một khối, để mỗi bước chỉ học từ train rồi áp nguyên cho dữ liệu mới.

MỤC TIÊU

- Phân biệt bước xử lý cố định với bước phải học từ train.
- Viết Pipeline và ColumnTransformer cho bảng có cột số, cột chữ và ô trống.
- Kiểm các bước xử lý được học lại trong mỗi fold.
- Nhận ra các kiểu data leakage hay gặp.

Tiền xử lý cũng là một phần của mô hình

Giả sử train có số đăng ký 10 và 20. Chuẩn hóa bằng trung bình $\mu = 15$, độ lệch chuẩn $\sigma = 5$ cho hai giá trị -1 và 1 . Validation có số đăng ký 30 phải được biến đổi thành $(30 - 15)/5 = 3$. Nếu chuẩn hóa validation riêng, 30 là giá trị duy nhất nên lại thành 0: ta đã đổi hệ tọa độ.

Chuẩn hóa dùng

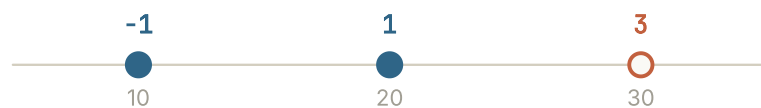
$$z_j = \frac{x_j - \mu_j}{\sigma_j}.$$

μ_j và σ_j là các con số **học từ dữ liệu**, dù không dùng nhãn. Median để điền ô trống, danh sách từ của văn bản, việc chọn feature cũng vậy. Vì thế chúng cũng là một phần của model: chỉ được học từ train, rồi giữ nguyên khi xử lý dữ liệu khác. Trong scikit-learn: `scaler.fit_transform(X_train)` rồi `scaler.transform(X_test)`; gọi `fit_transform` trên test là học lại μ, σ từ chính test.

Min-max đưa mỗi cột về $[0, 1]$ bằng $(x - \min)/(\max - \min)$, nên cần một cặp min, max cho mỗi cột, tính trên train. Giá trị test ngoài khoảng của train sẽ ra ngoài $[0, 1]$: train có tuổi 20–60 thì tuổi 70 thành $(70 - 20)/(60 - 20) = 1,25$, chuyện bình thường chứ không phải lỗi. Vì dựa vào cực trị, min-max dễ bị một điểm ngoại lai làm lệch cả thang; z-score chỉ dùng μ và σ .

Dùng scaler train

$\mu = 15, \sigma = 5$



Fit riêng valid

chỉ có một giá trị



Validation phải dùng μ, σ học từ train; fit riêng biến buổi đông nhất thành 0.

Biến đổi cố định, như đổi cm sang m bằng cách chia 100, thì không học gì từ dữ liệu, chỉ cần áp dụng giống nhau cho train và dữ liệu mới.

Python

```
import numpy as np
from sklearn.preprocessing import StandardScaler

small_train = np.array([[10.0], [20.0]])
small_valid = np.array([[30.0]])
scaler = StandardScaler().fit(small_train)
print('Train:', scaler.transform(small_train).ravel())
print('Valid dùng scaler train:', scaler.transform(small_valid).ravel())
wrong_valid = StandardScaler().fit_transform(small_valid)
print('Valid fit riêng, cách sai:', wrong_valid.ravel())
```

KẾT QUẢ

```
Train: [-1.  1.]
Valid dùng scaler train: [3.]
Valid fit riêng, cách sai: [0.]
```

Nhìn hai cách biến đổi 30. Cách đúng giữ quan hệ cao hơn trung bình train ba độ lệch chuẩn. Cách fit riêng đưa một buổi rất đông về ngay trung tâm thang đo, làm model nhận một ý nghĩa khác.

HỎI NHANH

Ta cần điền các ô thiếu ở validation bằng median. Median ấy phải tính từ đâu?

- A Từ chính các hàng validation
- B Từ train của lần học đang đánh giá
- C Từ train và validation gộp lại, trước khi chia
- D Từ test, vì test giống dữ liệu lúc dùng thật nhất

Data leakage: thông tin vượt qua ranh giới

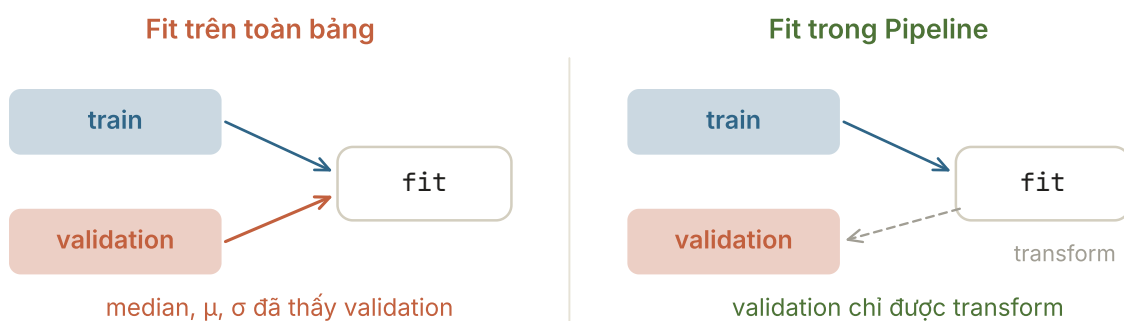
Rò rỉ dữ liệu (data leakage) xuất hiện khi quy trình xây mô hình dùng thông tin không được phép có trong tình huống đang đánh giá. Có nhiều đường đi, và không phải đường nào Pipeline cũng chặn được.

Đường rò rỉ	Ví dụ ở câu lạc bộ	Cách chặn
Feature có sau lúc dự đoán	Số vé đã quét sau buổi	Kiểm thời điểm cột được biết
Preprocessing	Median tính trên toàn bảng	Fit trong Pipeline và từng fold
Mẫu gán trùng	Một phiếu lưu hai lần	Gom bản trùng, chia group đúng
Nhãn tương lai	Tổng tham gia tuần sau trong feature tuần này	Kiểm lag/cửa sổ/thời điểm
Quyết định theo test	Chọn model vì test đẹp hơn	Dùng validation, giữ test độc lập

Dùng feature của test để học bước tiền xử lý, dù không đụng tới nhãn, vẫn làm sai phép đánh giá: lúc dùng thật, dữ liệu mới chưa có khi ta học. Có đề cho phép dùng cả input của tập test (không có nhãn) và sẽ ghi rõ trong thể lệ; nếu đề không nói gì thì không làm.

Bước nào học từ nhãn càng phải làm lại trong từng fold. Ví dụ chọn 50 feature tương quan mạnh nhất với nhãn trên cả bảng rồi mới cross-validation có thể cho 92%, trong khi chọn lại trong từng fold chỉ còn 78%. Khi điểm cao bất thường so với độ khó của bài, như 99,99% ở một bài nổi tiếng là khó, việc đầu tiên là tìm chỗ rò rỉ, rồi xem các lớp có lệch cực mạnh không.

Đường rò rỉ đầu bảng dễ thấy nhất khi tính tay. Train có số người đến (20, 30), validation là 40, và bảng có thêm một cột ghi sau buổi, đếm đúng số người đã vào cửa. Model $\hat{y} = x_{\text{final_count}}$ cho lỗi 0 trên cả ba hàng. Phép tính không sai, nhưng model vô dụng: chiều thứ Sáu chưa có cột ấy. Vì vậy trước khi mừng vì điểm đẹp, hãy kiểm từng feature có sẵn vào lúc dự đoán không.



Fit trên toàn bảng để validation lọt vào tiền xử lý; Pipeline chỉ fit trên train.

HỎI NHANH

Cột `final_attendance` chỉ có sau khi buổi sinh hoạt kết thúc. Đặt cột này vào Pipeline thì nó có thành feature hợp lệ không?

- A Không, lúc dự đoán vẫn chưa có cột này
- B Có, vì Pipeline tự chặn mọi kiểu rò rỉ
- C Có, miễn là không gọi `fit` trên test
- D Không, vì Pipeline chỉ nhận cột số

Bảng có số, chữ và ô thiếu

Ta tạo 80 buổi giả. Cột slot là sáng hoặc chiều; target tăng thêm năm người cho buổi chiều theo công thức giả lập. Một số feature được che thành missing sau khi sinh target để mô phỏng việc ghi chép thiếu. Ta giữ 60 buổi đầu làm train, 20 buổi cuối làm validation.

Python

```
import pandas as pd

rng = np.random.default_rng(14)
n = 80
registered = rng.integers(15, 65, size=n).astype(float)
rain = rng.uniform(0, 1, size=n)
slot = rng.choice(['sáng', 'chiều'], size=n)
y = 4 + 0.7 * registered - 5 * rain + 5 * (slot == 'chiều')
y = y + rng.normal(0, 1.5, size=n)
X = pd.DataFrame({'registered': registered, 'rain': rain, 'slot': slot})
X.loc[[2, 13, 44, 64], 'registered'] = np.nan
X.loc[[6, 50, 75], 'rain'] = np.nan
X_train, X_valid = X.iloc[:60], X.iloc[60:]
y_train, y_valid = y[:60], y[60:]
print(X_train.isna().sum().to_dict())
print(X_valid.isna().sum().to_dict())
```

KẾT QUẢ

```
{'registered': 3, 'rain': 2, 'slot': 0}
{'registered': 1, 'rain': 1, 'slot': 0}
```

Missing là thiếu thông tin, không mặc nhiên là giá trị 0. Đăng ký chưa ghi lại khác với chắc chắn không ai đăng ký. Median là baseline có thể dùng cho cột số; nó không khôi phục chính xác dữ liệu bị mất. Có thể thêm cột chỉ báo missing nếu validation đúng cách cho thấy hữu ích.

Với chữ, one-hot encoding tạo một cột cho mỗi danh mục đã thấy: sáng thành vector đánh dấu sáng, chiều thành vector đánh dấu chiều. Không mã hóa sáng=0, chiều=1, tối=2 rồi mặc nhiên coi khoảng cách giữa chúng có ý nghĩa: với mã 1, 2, 3, mô hình tuyến tính coi danh mục thứ hai nằm “giữa” hai danh mục kia. Ô thiếu ở cột chữ có thể điền bằng

danh mục gặp nhiều nhất trong train, hoặc thêm hẳn một danh mục “thiếu”. Bài 2.11 (Feature engineering) sẽ mở rộng phần này.

HỎI NHANH

Thay vì one-hot, ta mã hóa slot thành sáng = 0, chiều = 1, tối = 2, rồi đưa vào LinearRegression. Làm vậy là ngầm giả định điều gì?

- A Không giả định gì, vì số chỉ là tên của từng buổi
- B Đủ dữ liệu thì mô hình sẽ tự học ra thứ tự đúng
- C Cách mã hóa chỉ ảnh hưởng khi cột có ô thiếu
- D Sáng, chiều, tối nằm trên một đường thẳng, cách đều nhau



One-hot

sáng	1	0	0
chiều	0	1	0
tối	0	0	1

mỗi buổi một hệ số riêng

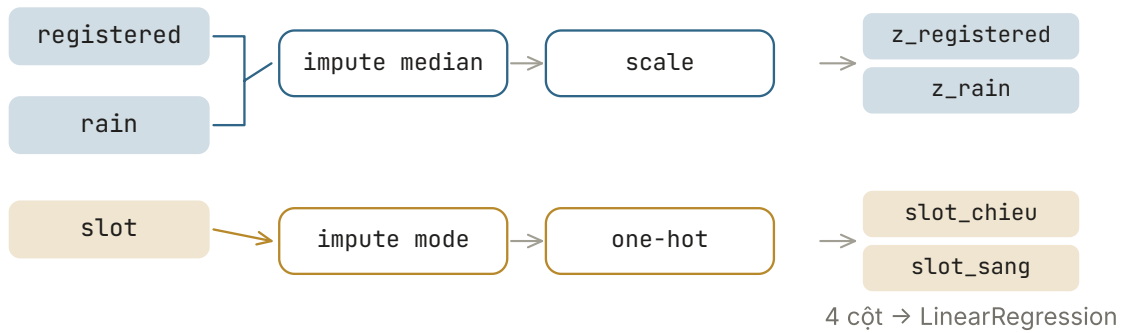
Mã số áp một thứ tự cách đều; one-hot cho mỗi danh mục một cột riêng.

Ghép ColumnTransformer với Pipeline

ColumnTransformer chia các cột thành nhóm, mỗi nhóm đi một nhánh xử lý riêng:

- Nhánh số: điền ô trống bằng median, rồi chuẩn hóa.
- Nhánh chữ: điền ô trống bằng danh mục hay gặp nhất, rồi one-hot.

Pipeline ngoài cùng ghép phần xử lý với LinearRegression thành một model. Nếu cột số có vài giá trị cực lớn, như thu nhập, thì nên điền bằng median thay vì trung bình, và có thể chặn (clip) giá trị theo ngưỡng tính trên train trước khi chuẩn hóa.



Cột số và cột chữ đi hai nhánh riêng rồi ghép thành 4 cột.

registered	rain	slot
58.00	0.08	chieu
61.00	0.15	chieu
20.00	0.26	chieu
52.00	0.08	chieu
—	0.99	chieu
23.00	—	sang

6 hàng validation thô: có ô trống (—) và cột chữ slot

registered	rain	slot
58.00	0.08	chieu
61.00	0.15	chieu
20.00	0.26	chieu
52.00	0.08	chieu
40.00	0.99	chieu
23.00	0.46	sang

SimpleImputer: ô trống thay bằng trung vị học từ train (40.0 và 0.46)

z_registered	z_rain	chieu	sang
+1.30	-1.41	1	0
+1.51	-1.14	1	0
-1.28	-0.78	1	0
+0.89	-1.41	1	0
+0.08	+1.75	1	0
-1.08	-0.07	0	1

OneHotEncoder: slot thành hai cột 0/1

z_registered	z_rain	chieu	sang	dự đoán
+1.30	-1.41	1	0	49.9
+1.51	-1.14	1	0	51.5
-1.28	-0.78	1	0	22.5
+0.89	-1.41	1	0	45.8
+0.08	+1.75	1	0	31.8
-1.08	-0.07	0	1	18.8

LinearRegression đọc 4 cột và cho dự đoán

Sáu hàng validation của bài đi qua Pipeline đã fit trên train: điền trung vị, chuẩn hóa, one-hot, rồi model dự đoán; mọi con số dùng thống kê của train.

Python

```

from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error

numeric = Pipeline([
    ('impute', SimpleImputer(strategy='median')),
    ('scale', StandardScaler()),
])

category = Pipeline([
    ('impute', SimpleImputer(strategy='most_frequent')),
  
```

```

    ('onehot', OneHotEncoder(handle_unknown='ignore', sparse_output=False)),
])
preprocess = ColumnTransformer([
    ('num', numeric, ['registered', 'rain']),
    ('cat', category, ['slot']),
])
model = Pipeline([
    ('preprocess', preprocess),
    ('regressor', LinearRegression()),
])
model.fit(X_train, y_train)
prediction = model.predict(X_valid)
print('Sau biến đổi:', model.named_steps['preprocess'].transform(X_valid).shape)
print(f'Validation MAE={mean_absolute_error(y_valid, prediction):.3f}')

```

KẾT QUẢ

Sau biến đổi: (20, 4)
Validation MAE=1.707

Nhìn shape sau biến đổi: hai cột số cộng hai cột one-hot. Tên preprocess và regressor do ta đặt để tra thành phần và chỉnh tham số sau này. Khi gọi `model.fit`, mọi bước cần fit đều học trên `X_train`; khi gọi `predict`, chỉ transform theo trạng thái đã lưu rồi dự đoán.

`sparse_output=False` giúp ví dụ nhỏ để nhìn dạng mảng đầy. Khi một cột có hàng nghìn danh mục, giữ mặc định mảng thưa để tiết kiệm bộ nhớ.

HỎI NHANH

Imputer nằm trong Pipeline, nhưng giá trị median đã được tính sẵn từ toàn bảng rồi truyền cố định vào. Chạy cross-validation thì còn sai ở đâu?

- A Median ấy đã tính cả các hàng validation của từng fold
- B Chỉ sai khi mô hình phía sau không tuyến tính
- C Không sai, vì mỗi fold vẫn gọi fit trên Pipeline
- D Phải dùng mean thay cho median thì mới đúng

Đọc trạng thái đã học và thử danh mục mới

Có thể in ra median mà Pipeline đã học để kiểm, giống in biến trung gian khi debug một thuật toán.

Python

```
fitted_numeric = model.named_steps['preprocess'].named_transformers_['num']
medians = fitted_numeric.named_steps['impute'].statistics_
print('Median đã học:', np.round(medians, 3))
print('Median tính từ train:', np.round(X_train[['registered', 'rain']].median().to_numpy(),
3))
new_session = pd.DataFrame({'registered': [30.0], 'rain': [0.2], 'slot': ['toi']})
new_features = model.named_steps['preprocess'].transform(new_session)
print('Cột one-hot của danh mục chưa thấy:', new_features[0, 2:])
print('Dự đoán buổi mới:', np.round(model.predict(new_session), 3))
```

KẾT QUẢ

```
Median đã học: [40.      0.462]
Median tính từ train: [40.      0.462]
Cột one-hot của danh mục chưa thấy: [0. 0.]
Dự đoán buổi mới: [27.493]
```

Median đã học khớp median tính tay trên train. Còn buổi 'toi' chưa từng có trong train: nhờ `handle_unknown='ignore'`, nó được mã hóa thành toàn số 0 ở hai cột one-hot. Không đặt tham số này thì `OneHotEncoder` báo lỗi khi gặp danh mục lạ. Code chạy được, nhưng model chưa hề học gì về buổi tối, nên con số dự đoán chỉ là đoán mò (câu hỏi dưới giải thích vì sao).

Một lỗi hay gặp khác là gọi `pd.get_dummies` riêng cho train và test. Train có các thành phố HN, HCM, DN, test có HN, HUE, nên hai bảng ra hai bộ cột khác nhau. Cách đúng là học một bộ mã hóa trên train rồi dùng nó để biến đổi test.

HỎI NHANH

Buổi “toi” chưa từng có trong train, nên nhận vector one-hot $[0, 0]$ và được dự đoán 27.49 người. Con số này thực chất là dự đoán cho cái gì?

- A Mô hình đã học được buổi tối đông hay vắng
- B Một buổi không phải sáng, cũng không phải chiều
- C Mô hình coi buổi tối như buổi chiều vì gần nhau
- D Không có nghĩa gì, lẽ ra phải báo lỗi

Cùng Pipeline, fit lại trong mỗi fold

Khi chạy cross-validation, đưa **cả Pipeline** vào `cross_val_score`, đừng đưa bảng đã xử lý sẵn trên toàn bộ dữ liệu. Ở mỗi fold, hàm này tạo một bản Pipeline mới và fit mọi bước chỉ trên train của fold đó.

Muốn chỉnh tham số của một bước bên trong Pipeline, ghi tên bước, hai dấu gạch dưới, rồi tên tham số, như `regressor__fit_intercept`. Nếu tạo Pipeline bằng `make_pipeline`, tên bước là tên lớp viết thường, nên tham số C của SVC là `svc__C`.

Python

```
from sklearn.model_selection import TimeSeriesSplit, cross_val_score

cv = TimeSeriesSplit(n_splits=3, test_size=10)
scores = cross_val_score(
    model, X_train, y_train, cv=cv,
    scoring='neg_mean_absolute_error', n_jobs=1,
)
print('Fold MAE:', np.round(-scores, 3))
print(f'Mean CV MAE={-scores.mean():.3f}')
```

KẾT QUẢ

Fold MAE: [1.087 1.995 1.314]

Như ở bài 1.2, scikit-learn trả MAE mang dấu âm (`neg_mean_absolute_error`), nên lấy `-scores` để đọc. Chỉ `X_train` được đưa vào cross-validation; 20 buổi validation bên ngoài chưa tham gia fit lần nào. Đây là cách bố trí hay dùng: cross-validation trên train để chọn tham số, rồi một phần giữ riêng bên ngoài để kiểm lại model đã chọn. Ba fold chên nhau khá nhiều (1.09 đến 2.00); nên xem số mẫu và số ô trống của từng fold để hiểu vì sao.

HỎI NHANH

Muốn cross-validation không bị rò rỉ qua bước điền median, ta đưa gì vào `cross_val_score`?

- A Bảng đã điền median tính từ mọi hàng
- B Chỉ đưa regressor, còn scaler đã fit trên toàn bảng
- C Bảng mà mỗi fold validation tự điền median của nó
- D Pipeline gồm imputer, scaler và regressor

Khi đi thi

- Với mỗi cột, hỏi: lúc dự đoán đã có cột này chưa?
- Mọi bước học từ dữ liệu (điền ô trống, chuẩn hóa, mã hóa) nằm trong Pipeline; đưa cả Pipeline vào cross-validation.
- Điểm tăng vọt sau khi thêm một cột thì kiểm cột ấy trước.

Tóm tắt

- Median, trung bình, độ lệch chuẩn, bộ mã hóa đều học từ train, rồi giữ nguyên cho dữ liệu khác.
- ColumnTransformer xử lý từng nhóm cột; Pipeline ghép xử lý với model thành một khối.
- Cross-validation phải fit lại cả Pipeline trong từng fold.
- Pipeline không chặn được mọi rò rỉ: cột có sau lúc dự đoán, bản ghi trùng, chọn theo test phải tự kiểm.

Kiểm tra cuối bài

CÂU 1

Bảng dữ liệu cho vay có cột số (thu nhập, tuổi) và cột chữ (nghề nghiệp, học vấn). 15% số dòng có ô thiếu, và cột thu nhập có vài giá trị cực lớn. Chuỗi tiền xử lý nào hợp lý nhất?

- A Xóa mọi dòng có ô thiếu, đưa cột số về khoảng 0 đến 1 theo min và max, đổi cột chữ thành 0, 1, 2...
- B Điền 0 vào ô thiếu, xóa dòng có thu nhập quá xa trung bình, giữ nguyên thang cột số, đổi cột chữ thành 0, 1, 2...
- C Điền median cho cột số và giá trị hay gặp nhất cho cột chữ, chặn bớt giá trị cực lớn, chuẩn hóa cột số, one-hot cột chữ
- D Giữ nguyên ô thiếu, chuẩn hóa cột số, one-hot cột chữ, rồi đưa thẳng vào LinearRegression

ĐỀ THI NOAI 2026 Singapore · Vòng sơ loại · Câu 304

CÂU 2

Thao tác nào phải học một con số từ train trước khi dùng?

- A Đổi cm sang m, tức chia cho 100
- B Điền ô thiếu bằng median
- C Đổi tên cột theo một danh sách cố định
- D Nhân một cột với hằng số mà đề đã cho

CÂU 3

Bảng có cột tuổi là số và cột loại trường là chữ. ColumnTransformer chuẩn hóa cột tuổi, one-hot cột loại trường, rồi ghép kết quả lại. Cách tổ chức này bảo đảm được điều gì?

- A Các cột không được chỉ định luôn được giữ lại, bất kể remainder
- B Mỗi nhánh nhận đúng nhóm cột của nó, rồi các đầu ra được ghép lại
- C Scaler và encoder dùng chung một bộ trung bình và phương sai
- D Encoder tự chọn những loại trường liên quan nhất đến nhãn

CÂU 4

OneHotEncoder đã học hai loại A và B, với `drop=None` và `handle_unknown="ignore"`. Gặp loại C chưa từng thấy, nhóm cột này sẽ cho vector nào?

A (1, 0)

B (0, 1)

C (1, 1)

D (0, 0)

CÂU 5

Một mô hình được học lại mỗi tuần. Sau 6 tuần, có người cần dựng lại đúng mô hình của tuần 3, nhưng nhóm chỉ còn file mô hình mới nhất. Muốn dựng lại được một mô hình đã học, ta cần lưu những gì?

A Dữ liệu đã dùng, cấu hình, seed, phiên bản thư viện, và cả Pipeline đã fit

B File mô hình và đoạn code huấn luyện

C Code, điểm validation và nhật ký huấn luyện

D File mô hình, các tham số đã chọn và điểm trên test

ĐỀ THI NOAI 2026 Singapore · Vòng sơ loại · Câu 308

CÂU 6

Ta cần dự báo trước buổi sinh hoạt một ngày. Cột `actual_attendance` trong bảng được ghi sau sự kiện, và có nó thì MAE gần bằng 0. Nên xử lý thế nào?

- A Bỏ cột này khỏi feature, vì lúc dự báo chưa có nó
- B Giữ cột này, nhưng đặt scaler vào Pipeline
- C Giữ cột này, nhưng tăng số fold validation
- D Giữ cột này, miễn là không dùng nó làm nhãn

CÂU 7

Mỗi học sinh được chụp ảnh nhiều lần, và một số ảnh còn bị lưu lại dưới tên khác. Muốn đánh giá trên học sinh mới, ta cần làm gì trước khi chạy cross-validation?

- A Đặt scaler vào Pipeline, rồi chia ngẫu nhiên từng ảnh
- B Giữ tỉ lệ nhãn ở hai phía, thay cho việc chia theo học sinh
- C Tìm các ảnh trùng, rồi chia theo nhóm học sinh
- D Đổi tên file, rồi chia ngẫu nhiên từng ảnh

CÂU 8

Scaler fit trên train có mean 10, std 2, và mô hình học trên thang đã chuẩn hóa ấy. Validation có mean 20, std 5. Giá trị $x = 22$ ở validation phải được đưa vào mô hình thành bao nhiêu?

A $(22 - 10)/5 = 2.4$

B $(22 - 10)/2 = 6$

C $(22 - 20)/5 = 0.4$

D $22 - 20 = 2$

CÂU 9

Một nhóm chọn 20 feature từ cả 100 dòng, rồi mới chạy 5-fold cross-validation, và thấy điểm rất cao. Kiểm lại thể nào cho đáng tin?

A Ở mỗi fold, chọn feature trên phần validation

B Dùng test để chọn feature, miễn là không fit mô hình trên test

C Ở mỗi fold, chỉ chọn feature từ phần train của fold đó

D Chọn một lần trên cả 100 dòng, dùng chung cho mọi fold

PHÒNG THEO [NOAI 2026 Singapore](#) · [Chung kết](#) · Câu 4

CÂU 10

Ta đã dùng kết quả trên test để chọn số feature, rồi chấm lại cấu hình thắng trên chính test đó. Muốn báo một con số độc lập, ta phải làm gì?

- A Fit lại mô hình từ đầu với seed khác rồi chấm lại
- B Đổi tên test thành validation, giữ nguyên điểm cũ
- C Bỏ bước chọn feature khỏi Pipeline rồi chấm lại
- D Chấm trên dữ liệu mới, chưa tham gia quyết định nào

Đọc thêm

- [scikit-learn — Common pitfalls](#). Đọc các ví dụ về preprocessing không nhất quán và leakage.
- [scikit-learn — Pipelines and composite estimators](#). Tra cách ghép Pipeline, ColumnTransformer và tên bước.
- [scikit-learn — OneHotEncoder](#). Xem cách xử lý danh mục mới và lựa chọn mảng thưa.