

PYTHON CHO KHOA HỌC DỮ LIỆU

Giảng viên: Hà Minh Tuấn

THAM SỐ TRUYỀN VÀO

Nhóm thực hiện:

Đặng Đỉnh Đoàn 23280046
Phạm Thanh Uy 23280097

TP. Hồ Chí Minh, tháng 12 năm 2025

Mục lục

1 Giới thiệu	2
2 Hàm build_preprocessor_config	2
2.1 Đường dẫn file và thư mục	2
2.2 Cột target và hidden missing	2
2.3 Chiến lược xử lý missing	2
2.4 Xử lý outlier	3
2.5 Chuẩn hóa	4
2.6 Encoding	4
2.7 Feature engineering	5
2.8 Split train/test	5
3 Hàm build_trainer_config	6
3.1 Target và random state	6
3.2 Scoring	6
3.3 Cross-validation và search	6
3.4 Imbalance và model list	7
3.5 Thư mục lưu model	7
4 Nhóm giải thích	7
4.1 summarize_test_metrics(...)	7
4.2 show_feature_importance_and_shap(...)	8
5 Nhóm main và cách pipeline chạy	8

1 Giới thiệu

Đây là các giá trị dùng để ghi đè lên mặc định, phục vụ cho mục đích của người dùng(user). Nằm trong file main.py.

2 Hàm build_preprocessor_config

Gồm tham số phục vụ cho việc tiền xử lý.

2.1 Đường dẫn file và thư mục

- raw_data_path = project_root / "data"/ "raw"/ "diabetes.csv"
 - Ví dụ điều chỉnh:
 - * Tên file: "diabetes.csv" → "pima.csv" hoặc file khác.
 - * Thư mục: "raw" → "input", v.v.
- processed_dir = project_root / "data"/ "processed"
 - Ví dụ điều chỉnh: tên thư mục processed.
- processed_train_path = processed_dir / "pima_train_processed.parquet"
- processed_test_path = processed_dir / "pima_test_processed.parquet"
 - Ví dụ điều chỉnh:
 - * Tên file: "pima_train_processed.parquet" → "train.parquet".
 - * Định dạng: .parquet → .csv (nếu DataPreprocessor hỗ trợ).
- scaler["save_scaler_path"] = processed_dir / "scaler.joblib"
 - Đổi tên file scaler hoặc thư mục lưu.

2.2 Cột target và hidden missing

- target_col = "Outcome"
 - Nếu sau này dùng dataset khác, ví dụ điều chỉnh thành tên cột nhãn khác:
Ví dụ: "Target", "HasDiabetes",...
- hidden_missing_cols=["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"]
=> Ta có thể thêm/ bớt cột trong list hoặc đổi tên nếu dataset có tên cột khác.

2.3 Chiến lược xử lý missing

- missing = {
 - "numeric_strategy": "median_by_outcome",

- "categorical_strategy": "most_frequent",
- **numeric_strategy – các tùy chọn hợp lý cho dạng numeric:**
 - "median_by_outcome": Tính median riêng cho từng nhóm Outcome (0 và 1). Giữ lại đặc trưng phân bố khác nhau giữa hai nhóm; thường phù hợp với dữ liệu y tế. (*Hiện đang sử dụng*)
 - "median_overall": Tính median trên toàn bộ dữ liệu, không phân nhóm. Đơn giản hơn nhưng có thể làm mất các tín hiệu quan trọng theo nhãn.
- **categorical_strategy – các tùy chọn hợp lý cho dạng categorical:**
 - "most_frequent": Diền bằng giá trị xuất hiện nhiều nhất (mode) trên toàn bộ tập dữ liệu. Là lựa chọn phổ biến, đơn giản và thường hiệu quả.

2.4 Xử lý outlier

- **outlier = {**
 - "numeric_cols": None,
 - "method": "iqr",
 - "strategy": "winsorize",
 - "iqr_factor": 1.5,
- **numeric_cols:**
 - None: tự động lấy tất cả các cột dạng numeric.
 - Danh sách cụ thể, ví dụ: ["Glucose", "Insulin"].
- **method – cách phát hiện outlier:**
 - "iqr": dùng khoảng tứ phân vị IQR (Q1, Q3, và ngưỡng Q1 - 1.5*IQR đến Q3 + 1.5*IQR). Hiện tại class chỉ hỗ trợ phương pháp này.
- **strategy – cách xử lý outlier:**
 - "winsorize": cắt đuôi dữ liệu về ngưỡng dưới/trên (đang sử dụng).
 - "flag": tạo thêm cột nhị phân để đánh dấu điểm outlier.
 - "none": không thay đổi dữ liệu.
- **iqr_factor:**
 - Mặc định: 1.5.
 - Có thể điều chỉnh:
 - * 1.0: nhạy hơn, nhiều giá trị bị coi là outlier.
 - * 3.0: “dễ dãi” hơn, ít điểm bị xem là outlier.

2.5 Chuẩn hóa

- **scaler = {**
 - "type": "standard",
 - "exclude_cols": ["Outcome"],
 - "save_scaler_path": processed_dir / "scaler.joblib",**}**
- **type – loại scaler:**
 - "standard": StandardScaler (chuẩn hóa dữ liệu sao cho mean = 0, std = 1).
 - "minmax": MinMaxScaler (chuẩn hóa dữ liệu về khoảng 0–1).
 - "none": Không scale dữ liệu.
 - Có thể có thêm các loại khác nếu class hỗ trợ, ví dụ "robust", ...
- **exclude_cols – các cột không scale:**
 - Mặc định là ["Outcome"].
 - Có thể thêm bớt các cột khác, ví dụ ["Outcome", "Age_group"] nếu muốn giữ nguyên một số cột.
- **save_scaler_path – đường dẫn lưu scaler:**
 - Ví dụ: processed_dir / "scaler.joblib".

2.6 Encoding

- **encoding = {**
 - "strategy": "onehot",
 - "handle_unknown": "ignore",**}**
- **strategy – cách encode dữ liệu categorical:**
 - "onehot": Sử dụng pd.get_dummies để tạo các cột nhị phân cho từng giá trị.
 - "label": Sử dụng LabelEncoder cho từng cột.
 - "none": Không encode, áp dụng khi tất cả dữ liệu đã ở dạng numeric.
- **handle_unknown – cách xử lý giá trị chưa gắp trong test set:**
 - "ignore": bỏ qua các giá trị chưa gắp.
 - Có thể có các tùy chọn khác tùy class encoder sử dụng.

2.7 Feature engineering

- `feature_engineering = {`
 - "enable": True,
 - "create_bmi_category": True,
 - "create_age_group": True,
 - "create_pregnancy_flag": True,
 - "create_interactions": True,
 - "bmi_col": "BMI",
 - "age_col": "Age",
 - "pregnancies_col": "Pregnancies",
 - "glucose_col": "Glucose",
 - "insulin_col": "Insulin",`}`
- **enable** – bật/tắt toàn bộ feature engineering:
 - True: bật
 - False: tắt
- **create_...** – bật/tắt từng feature:
 - `create_bmi_category`: tạo cột BMI category (True/False)
 - `create_age_group`: tạo cột Age group (True/False)
 - `create_pregnancy_flag`: tạo cột Pregnancy flag (True/False)
 - `create_interactions`: tạo các tương tác giữa biến (True/False)
- **..._col** – đổi tên cột nguồn:
 - `bmi_col`: cột BMI, ví dụ "BMI" → "BodyMassIndex"
 - `age_col`: cột Age
 - `pregnancies_col`: cột Pregnancies
 - `glucose_col`: cột Glucose
 - `insulin_col`: cột Insulin

2.8 Split train/test

- `split = {`
 - "test_size": 0.2,
 - "random_state": 42,
 - "stratify": True,`}`

}

- **test_size** – tỉ lệ dữ liệu test:
 - Ví dụ: 0.2 (20% test)
 - Có thể điều chỉnh: 0.1, 0.25, 0.3, ...
- **random_state** – seed cho việc shuffle dữ liệu:
 - Int bất kỳ: 42, 123, 2024, ...
 - Thay đổi giá trị sẽ thay đổi cách shuffle, nhưng nên cố định để reproducible
- **stratify** – stratify theo nhãn y:
 - True: giữ tỉ lệ 0/1 giống toàn dataset
 - False: split ngẫu nhiên, không stratify

3 Hàm build_trainer_config

3.1 Target và random state

- **target_col** – cột nhãn (label) của dataset:
 - Ví dụ: "Outcome"
 - Có thể đổi nếu tên cột label khác
- **random_state** – seed cho các thao tác ngẫu nhiên liên quan đến target:
 - Ví dụ: 42
 - Có thể đổi sang số khác nếu muốn kết quả random khác
 - Nên giữ cố định để kết quả reproducible

3.2 Scoring

- **scoring_primary** – metric chính dùng để đánh giá model:
 - Ví dụ: "f1"
 - Có thể điều chỉnh sang các metric khác: "roc_auc", "accuracy", "precision", "recall", ...
- **scoring_other** – các metric phụ:
 - Ví dụ: ["roc_auc", "accuracy", "precision", "recall"]
 - Có thể thêm/bớt metric phụ tùy nhu cầu, ví dụ: ["roc_auc", "balanced_accuracy"]

3.3 Cross-validation và search

- **cv_splits** – số fold cho cross-validation:
 - Ví dụ: 5

- Có thể đổi sang 3, 4, 10, ... Lưu ý: số fold lớn hơn → kết quả ổn định hơn nhưng tốn thời gian hơn
- **use_randomized_search – chọn loại search:**
 - True: dùng RandomizedSearchCV
 - False: dùng GridSearchCV
- **n_iter_random_search – số combination random khi dùng RandomizedSearchCV:**
 - Ví dụ: 20
 - Giảm để chạy nhanh (ví dụ 10)
 - Tăng để tìm kỹ hơn (ví dụ 50)

3.4 Imbalance và model list

- **use_smote – sử dụng SMOTE để xử lý imbalance:**
 - True: dùng SMOTE trong cross-validation để cân bằng dữ liệu nhãn
 - False: không dùng SMOTE
- **model_names – danh sách model mà Trainer hỗ trợ:**
 - Ví dụ: ["log_reg", "random_forest"]
 - Có thể thêm/bớt các model khác tùy nhu cầu

3.5 Thư mục lưu model

- **model_output_dir – thư mục lưu model đã train:**
 - Ví dụ: project_root / "models"
 - Có thể thay đổi sang các thư mục khác tùy nhu cầu, ví dụ: "artifacts", "outputs/models", ...

4 Nhóm giải thích

4.1 summarize_test_metrics(...)

- **Logic in/out – cố định, nhưng có thể tuỳ chỉnh:**
 - Có thể thêm/bớt metric cần log, ví dụ log thêm specificity nếu Trainer trả về
 - Có thể chỉnh nội dung giải thích trade-off: thay câu chữ cho phù hợp
- **Các key trong metrics đang đọc:**
 - "accuracy", "precision", "recall", "f1", "roc_auc", "tn", "fp", "fn", "tp"

4.2 show_feature_importance_and_shap(...)

- top_k – số lượng feature top cần in:
 - Ví dụ: 10
 - Có thể đổi sang 5, 15, 20, ... tùy nhu cầu
- max_samples – số lượng mẫu dùng để tính SHAP trong hàm compute_shap_values(...)
 - Ví dụ: 200
 - Ít hơn → chạy nhanh hơn nhưng kết quả noisy hơn
 - Nhiều hơn → chậm hơn nhưng ổn định hơn

5 Nhóm main và cách pipeline chạy

- Điều chỉnh việc gọi explain (SHAP / feature importance):

- Ví dụ trong code:

```
* if isinstance(X_train, pd.DataFrame):  
    *     show_feature_importance_and_shap(trainer, best_model_name, X_train)
```
- Có thể comment tạm nếu không muốn tính SHAP để tiết kiệm thời gian
- Có thể gọi thêm các hàm vẽ biểu đồ khác nếu muốn mở rộng