Ảnh có chứa khung ảnh, Hình chữ nhật, khung

Nội dung do AI tạo ra có thể không chính xác.

**BỘ NÔNG NGHIỆP VÀ PHÁT TRIỂN NÔNG THÔN**

**PHÂN HIỆU TRƯỜNG ĐẠI HỌC THỦY LỢI**

-----🙞🙜🕮🙞🙜-----



BÁO CÁO MÔN HỌC KHAI PHÁ DỮ LIỆU

**CREDIT CARD FRAUD DETECTION**

SINH VIÊN THỰC HIỆN :

2251068227 – Bùi Thiện Phát

2251068255 – Nguyễn Phúc Thịnh

GIÁO VIÊN HƯỚNG DẪN : Ths.Vũ Thị Hạnh

Hồ Chí Minh, ngày 20 tháng 10 năm 2025

# **I. GIỚI THIỆU ĐỀ TÀI**

Trong thời đại công nghệ số, giao dịch trực tuyến và thanh toán điện tử đã trở thành một phần không thể thiếu trong đời sống hiện đại. Tuy nhiên, song song với sự phát triển này là sự gia tăng nhanh chóng của các **hoạt động gian lận tài chính**, đặc biệt trong lĩnh vực **thẻ tín dụng**. Các hành vi gian lận ngày càng tinh vi, khó phát hiện bằng các phương pháp truyền thống, gây thiệt hại lớn cho cả ngân hàng, tổ chức tài chính và khách hàng cá nhân. Vì vậy, việc áp dụng **các phương pháp khai phá dữ liệu và học máy** để tự động phát hiện giao dịch bất thường là một hướng đi quan trọng và có tính ứng dụng cao.

Bài toán **phát hiện gian lận thẻ tín dụng (Credit Card Fraud Detection)** là một ví dụ điển hình của lĩnh vực **phân loại nhị phân mất cân bằng (imbalanced classification)**, trong đó số lượng giao dịch gian lận chiếm tỷ lệ rất nhỏ so với giao dịch hợp lệ. Điều này đặt ra thách thức lớn cho các mô hình học máy, vì nếu không xử lý đúng cách, mô hình dễ thiên lệch và dự đoán tất cả giao dịch là “hợp lệ”, dẫn đến bỏ sót các trường hợp gian lận thực sự. Do đó, cần có các chiến lược tiền xử lý dữ liệu, lựa chọn mô hình và đánh giá phù hợp để đảm bảo mô hình vừa chính xác, vừa nhạy trong việc phát hiện gian lận.

Trong dự án này, nhóm tập trung vào việc **xây dựng, huấn luyện và đánh giá nhiều mô hình học máy khác nhau** nhằm phát hiện các giao dịch gian lận trong bộ dữ liệu thực tế từ Kaggle. Thông qua việc áp dụng các kỹ thuật như **chuẩn hóa dữ liệu, xử lý mất cân bằng bằng SMOTE, sử dụng trọng số lớp (class\_weight)** và **so sánh hiệu năng của nhiều mô hình (Logistic Regression, Random Forest, XGBoost, LightGBM, CatBoost, SVM, Decision Tree)**, nhóm hướng đến việc tìm ra mô hình có khả năng nhận diện chính xác nhất các giao dịch gian lận.

**II. MỤC TIÊU VÀ BÀI TOÁN ĐẶT RA**

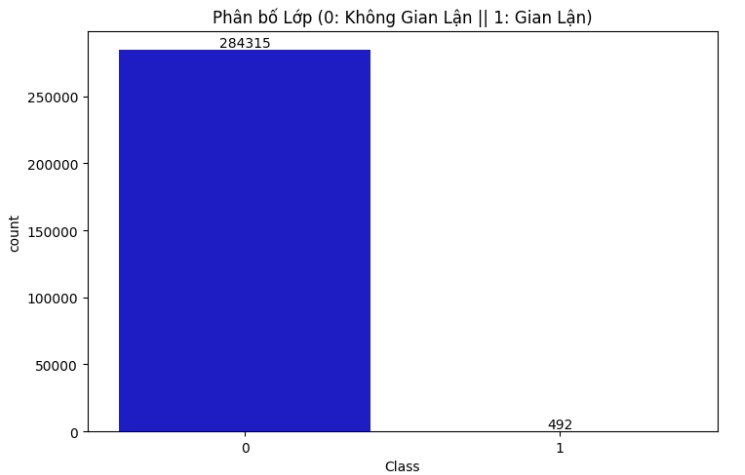
**2.1 Mục tiêu của đề tài**  
Mục tiêu của đề tài là **xây dựng một hệ thống phát hiện gian lận thẻ tín dụng tự động** bằng cách áp dụng các thuật toán học máy (Machine Learning) trên bộ dữ liệu thực tế. Hệ thống này phải có khả năng **phân biệt chính xác giữa giao dịch hợp lệ và giao dịch gian lận**, đồng thời **giảm thiểu rủi ro bỏ sót các trường hợp gian lận thật**.  
Cụ thể, nhóm hướng tới việc:

* Xây dựng quy trình xử lý dữ liệu hiệu quả, bao gồm chuẩn hóa và xử lý mất cân bằng lớp.
* Huấn luyện, đánh giá và so sánh hiệu năng của nhiều mô hình khác nhau để chọn ra mô hình tối ưu.
* Triển khai kết quả thành công cụ trực quan, dễ sử dụng cho việc dự đoán gian lận.

**2.2 Bài toán đặt ra**  
Bài toán được xác định là **bài toán phân loại nhị phân (Binary Classification)**, trong đó:

* **Đầu vào (Input):** Các đặc trưng mô tả thông tin giao dịch, bao gồm 28 đặc trưng ẩn danh (V1–V28) cùng hai biến gốc Time và Amount.
* **Đầu ra (Output):** Biến nhãn Class với hai giá trị:
  + 0: giao dịch hợp lệ (non-fraud).
  + 1: giao dịch gian lận (fraud).

Thách thức chính của bài toán là dữ liệu **mất cân bằng nghiêm trọng**, khi số lượng giao dịch gian lận chỉ chiếm khoảng **0.17%** tổng số mẫu. Do đó, nhóm cần kết hợp các kỹ thuật như **SMOTE**, **class\_weight='balanced'**, cùng quy trình tiền xử lý dữ liệu và đánh giá bằng các chỉ số phù hợp như **AUC**, **F1-score**, **Precision**, và **Recall** để đảm bảo mô hình không bị lệch về lớp chiếm đa số.



Phân bố lớp trong dữ liệu (mất cân bằng nghiêm trọng giữa hai loại giao dịch)

**2.3 Kết quả mong đợi**  
Sau khi hoàn thành, hệ thống phải:

* Dự đoán chính xác các giao dịch gian lận với **AUC và F1-score cao**, đảm bảo cân bằng giữa độ nhạy (Recall) và độ chính xác (Precision).
* Có khả năng **phát hiện sớm các hành vi gian lận** nhằm hỗ trợ ngân hàng và người dùng trong việc giảm thiểu rủi ro tài chính.
* Cung cấp **biểu đồ trực quan** (ROC Curve, Precision-Recall Curve, Confusion Matrix) để minh họa kết quả mô hình và có thể tích hợp trên **giao diện web hoặc dashboard** cho phép nhập dữ liệu và xem kết quả dự đoán trực tiếp.

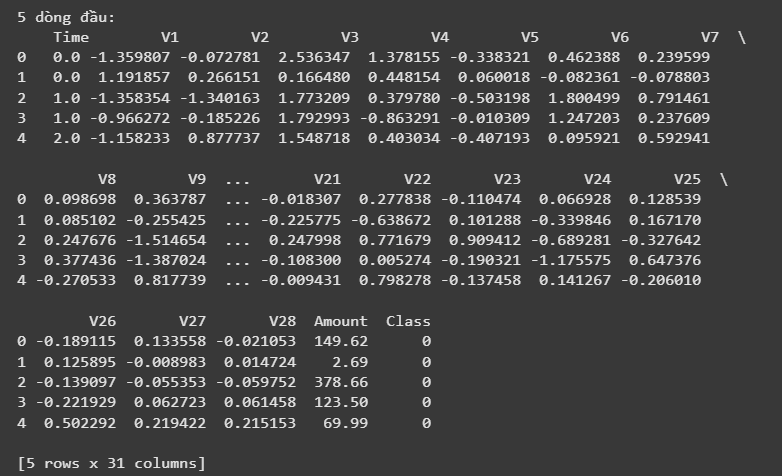
**III. MÔ TẢ DỮ LIỆU VÀ CÁC BƯỚC TIỀN XỬ LÝ**

**3.1 Mô tả dữ liệu**

Bộ dữ liệu được sử dụng trong đề tài là **Credit Card Fraud Detection Dataset**, được lấy từ trang **Kaggle**. Dữ liệu bao gồm **284.807 giao dịch thẻ tín dụng** được thực hiện trong hai ngày, trong đó chỉ có **492 giao dịch được gắn nhãn là gian lận**, chiếm khoảng **0.172%** tổng số giao dịch.

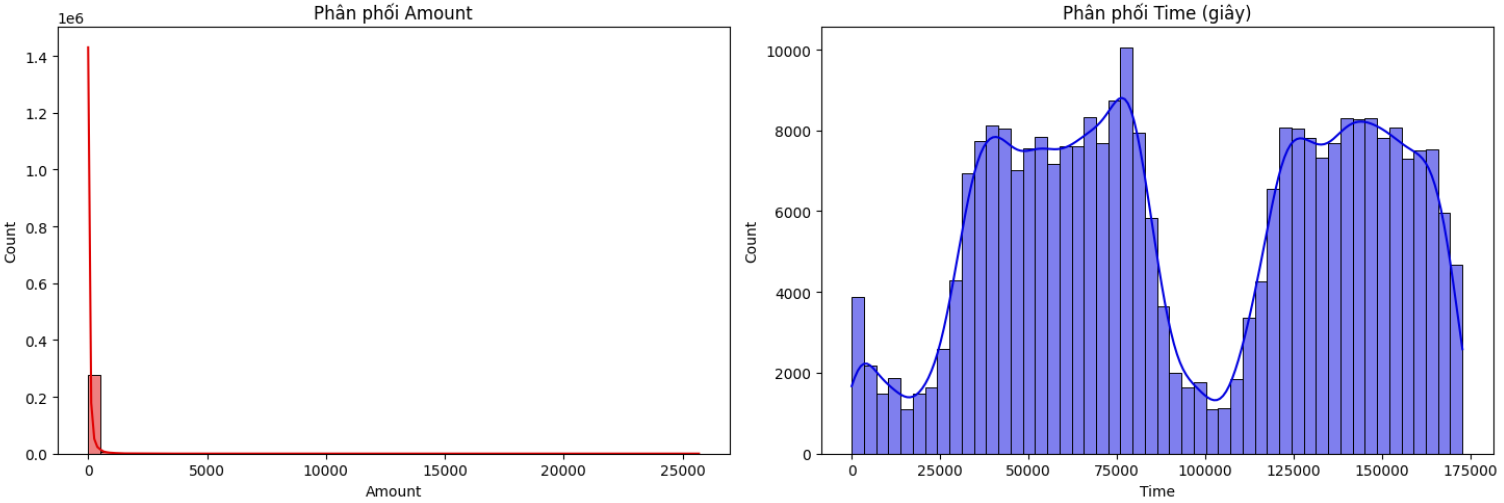
Dữ liệu bao gồm **31 thuộc tính**, trong đó:

* Time: thời gian tính bằng giây giữa giao dịch hiện tại và giao dịch đầu tiên.
* Amount: số tiền của giao dịch.
* Class: nhãn mục tiêu — 0 là giao dịch bình thường, 1 là giao dịch gian lận.
* V1 → V28: các đặc trưng đã được trích xuất bằng **PCA (Principal Component Analysis)** để ẩn thông tin gốc vì lý do bảo mật.



Bộ dữ liệu này có đặc điểm quan trọng là mất cân bằng nghiêm trọng về số lượng giữa hai lớp, đây chính là yếu tố khiến việc huấn luyện và đánh giá mô hình trở nên khó khăn hơn.

Nhờ vào đặc trưng đã được xử lý bằng PCA, các biến đầu vào không chứa thông tin nhận dạng cá nhân, đồng thời giúp giảm tương quan giữa các thuộc tính, hỗ trợ cho việc huấn luyện mô hình học máy.



Phân phối giá trị Amount và Time trong dữ liệu

**3.2 Các bước tiền xử lý**

Chia train/test

Chuẩn hóa

Kiểm tra

Đọc dữ liệu

PCA/t- NSE

Cân bằng lớp

**Bước 1. Đọc và kiểm tra dữ liệu**

Dữ liệu được đọc từ file **creditcard.csv** bằng thư viện Pandas. Nhóm tiến hành kiểm tra kích thước, 5 dòng đầu tiên, thống kê mô tả (describe()), và giá trị thiếu (isnull().sum().max()).

Kết quả cho thấy:

* Dữ liệu gồm 284.807 bản ghi và 31 cột.
* Không có giá trị bị thiếu (NaN).

→ Có thể sử dụng trực tiếp cho bước xử lý tiếp theo.

**Bước 2. Phân tích và trực quan hóa dữ liệu (EDA)**

Nhóm tiến hành vẽ các biểu đồ để quan sát cấu trúc và phân bố dữ liệu:

* Biểu đồ cột (Countplot): thể hiện sự mất cân bằng lớp giữa “gian lận” và “hợp lệ”.
* Boxplot: so sánh giá trị Amount giữa hai lớp.
* Histogram: biểu diễn phân phối của Amount và Time.
* Heatmap: hiển thị ma trận tương quan giữa các đặc trưng.

Qua đó nhận thấy dữ liệu PCA đã loại bỏ tương quan cao, tuy nhiên Time và Amount có thang đo khác biệt → cần chuẩn hóa.

**Bước 3. Chuẩn hóa dữ liệu**

* Tách đặc trưng và nhãn:
* X = df.drop('Class', axis=1)
* y = df['Class']
* Sử dụng StandardScaler để chuẩn hóa:
  + Trước tiên scale hai cột Time và Amount.
  + Sau đó scale toàn bộ dữ liệu (bao gồm V1–V28) để đưa các đặc trưng về cùng thang đo.

Kết quả được xác nhận bằng việc in ra giá trị trung bình ≈ 0 và độ lệch chuẩn ≈ 1.

**Bước 4. Chia dữ liệu huấn luyện và kiểm tra**

* Sử dụng hàm train\_test\_split để chia dữ liệu thành 80% train – 20% test, với tham số stratify=y để giữ nguyên tỷ lệ lớp gian lận trong hai tập.

**Bước 5. Xử lý mất cân bằng lớp**

Do tỷ lệ giao dịch gian lận rất thấp, nhóm áp dụng hai hướng xử lý:

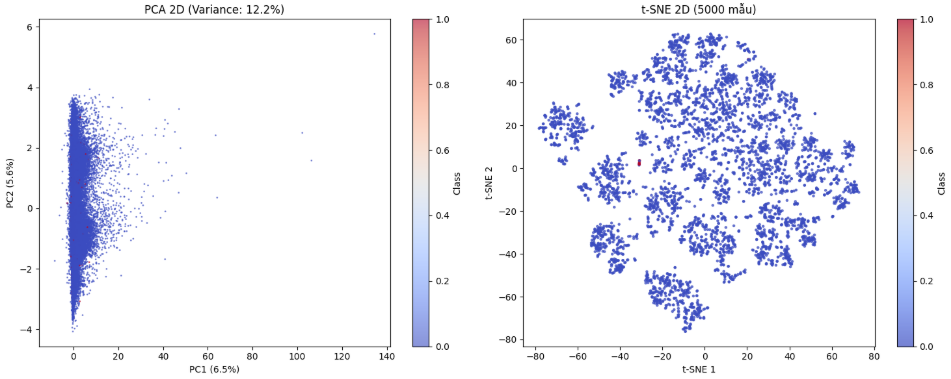
* Dùng class\_weight='balanced' trong các mô hình học máy để cân bằng tầm quan trọng giữa hai lớp.
* Áp dụng SMOTE (Synthetic Minority Oversampling Technique) trong pipeline khi thực hiện Cross-Validation (5-Fold), giúp sinh thêm các mẫu giả cho lớp gian lận, cải thiện khả năng nhận diện.

Bước 6. Giảm chiều và trực quan hóa đặc trưng

Sau khi chuẩn hóa, dữ liệu được giảm chiều bằng:

* PCA (2 thành phần chính) để trực quan hóa trong không gian 2D.
* t-SNE (5000 mẫu) để thể hiện cấu trúc phi tuyến giữa các lớp.

Các biểu đồ giúp nhận thấy rằng các điểm gian lận phân tán rời rạc và khó phân tách hoàn toàn tuyến tính, điều này giải thích vì sao cần thử nghiệm nhiều mô hình học máy khác nhau.



Trực quan dữ liệu sau khi giảm chiều bằng PCA và t-SNE

**IV. PHƯƠNG PHÁP KHAI PHÁ DỮ LIỆU**

**Tổng quan**

Nhóm thực hiện khai phá và so sánh nhiều mô hình phân loại để giải bài toán phát hiện gian lận thẻ tín dụng. Quy trình chính bao gồm: chuẩn hóa dữ liệu, chia train/test (stratify), huấn luyện nhiều mô hình với chiến lược cân bằng lớp, tối ưu ngưỡng phân loại, đánh giá bằng các chỉ số thích hợp và so sánh kết quả (AUC, F1). Ngoài ra nhóm áp dụng cross-validation kết hợp SMOTE để đánh giá ổn định với dữ liệu mất cân bằng và vẽ learning curve để kiểm tra hiện trạng học của mô hình.

**Các mô hình thử nghiệm**

Trong notebook, nhóm đã khởi tạo và huấn luyện các mô hình sau (với một số tham số chính như trong code):

* Logistic Regression (class\_weight='balanced', max\_iter=1000)
* Decision Tree (class\_weight='balanced')
* Random Forest (class\_weight='balanced', n\_estimators=100)
* SVM (class\_weight='balanced', probability=True)
* XGBoost (scale\_pos\_weight được tính theo tỷ lệ lớp trong train, eval\_metric='logloss')
* LightGBM (class\_weight='balanced')
* CatBoost (auto\_class\_weights='Balanced', verbose=0)

**Quy trình huấn luyện và đánh giá**

1. **Huấn luyện trực tiếp trên X\_train / đánh giá trên X\_test**

* Mỗi mô hình được fit trên X\_train, y\_train rồi dự đoán xác suất (predict\_proba) trên X\_test.
* Tính AUC (ROC AUC) làm chỉ số chính cho khả năng phân biệt tổng quát.
* Tính F1 score theo hai cách: F1 mặc định (với ngưỡng 0.5) và F1 tối ưu tìm bằng hàm find\_best\_threshold (duyệt ngưỡng 0.10–0.89 bước 0.01) để cân bằng precision/recall theo dữ liệu thực tế.
* Vẽ confusion matrix ứng với ngưỡng tối ưu để minh họa số lượng TP/FP/TN/FN.

1. **So sánh mô hình**

* Lưu các chỉ số (AUC, F1\_default, F1\_optimal, Threshold, Time) vào bảng df\_results và vẽ biểu đồ so sánh AUC và F1 tối ưu để chọn mô hình tốt nhất.

1. **Tối ưu ổn định bằng Cross-Validation + SMOTE**

* Sử dụng StratifiedKFold(n\_splits=5) để giữ tỉ lệ lớp trong các fold.
* Xây pipeline ImbPipeline([('smote', SMOTE()), ('model', model)]) để oversample lớp thiểu số chỉ trên tập huấn luyện mỗi fold (tránh rò rỉ dữ liệu).
* Tính trung bình AUC từ cross\_val\_score và F1 tối ưu trên các fold để đánh giá ổn định hơn.

1. **Learning curve**

* Với pipeline (SMOTE + XGBoost), dùng learning\_curve (scoring = 'roc\_auc') để vẽ đường học (train vs validation) theo kích thước tập huấn luyện, giúp phát hiện underfitting/overfitting và ước lượng lợi ích khi thêm dữ liệu.

**Kỹ thuật cân bằng lớp và ngưỡng**

* Dùng class\_weight='balanced' ở những mô hình hỗ trợ để điều chỉnh lỗi học do mất cân bằng lớp.
* Dùng SMOTE để sinh mẫu cho lớp thiểu số trong cross-validation (thay vì oversample toàn cục) — cách này giúp mô hình học ranh giới tốt hơn mà không gây rò rỉ dữ liệu.
* Tìm ngưỡng phân loại tối ưu riêng cho từng mô hình bằng find\_best\_threshold, vì với dữ liệu mất cân bằng, ngưỡng 0.5 thường không phù hợp nếu mục tiêu ưu tiên recall (phát hiện gian lận).

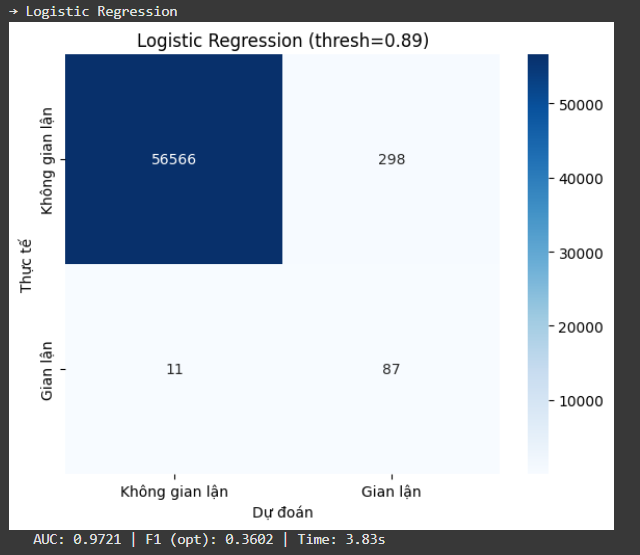
**Giải thích mô hình và phân tích đặc trưng**

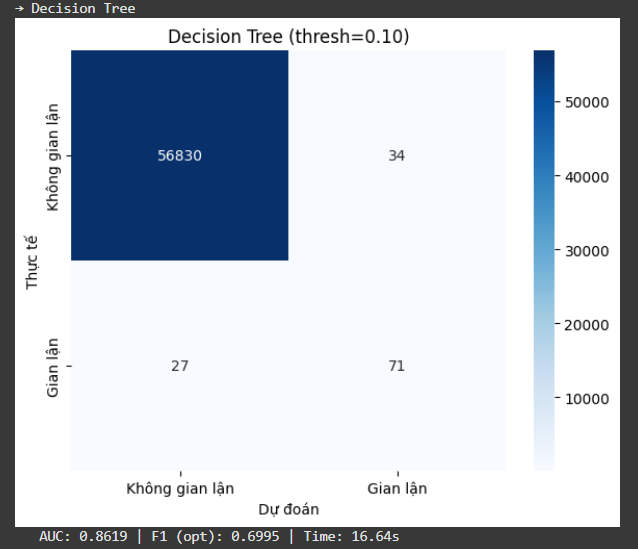
* Lấy feature importance từ Random Forest để xác định các đặc trưng có ảnh hưởng nhất; vẽ barplot top-6 features.
* Vẽ KDE plots cho các đặc trưng hàng đầu để so sánh phân phối giữa lớp gian lận và không gian lận.
* Notebook đã import shap và đặt mục tiêu dùng SHAP để giải thích mô hình; tuy nhiên phần tính toán explainer/SHAP values không được triển khai chi tiết trong mã hiện có — việc bổ sung SHAP là bước tiếp theo hữu ích để hiểu quyết định của mô hình.

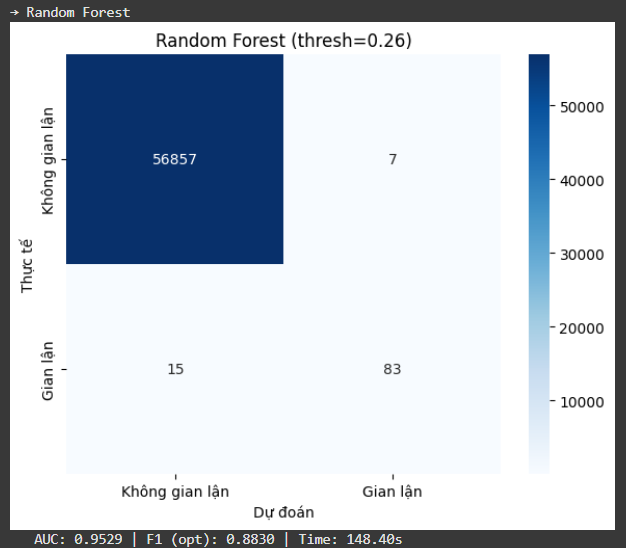
**Đánh giá sử dụng các chỉ số phù hợp**

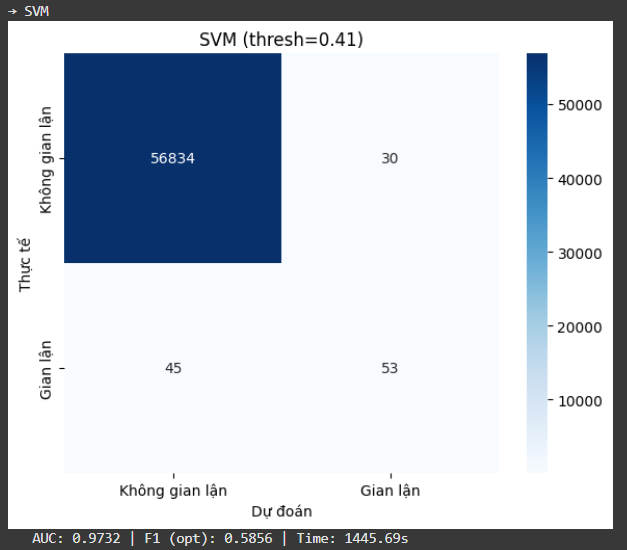
* AUC (ROC AUC): chỉ số tổng quát cho khả năng phân biệt model giữa hai lớp.
* Precision / Recall / F1: đặc biệt quan trọng trong bài toán mất cân bằng — nhóm sử dụng F1 tối ưu sau khi tìm ngưỡng để cân bằng precision và recall theo mục tiêu (ưu tiên giảm False Negatives).
* Confusion Matrix: minh họa trực quan số TP/FP/TN/FN với ngưỡng tối ưu.
* Cross-validation scores: đánh giá tính ổn định của mô hình qua nhiều split.

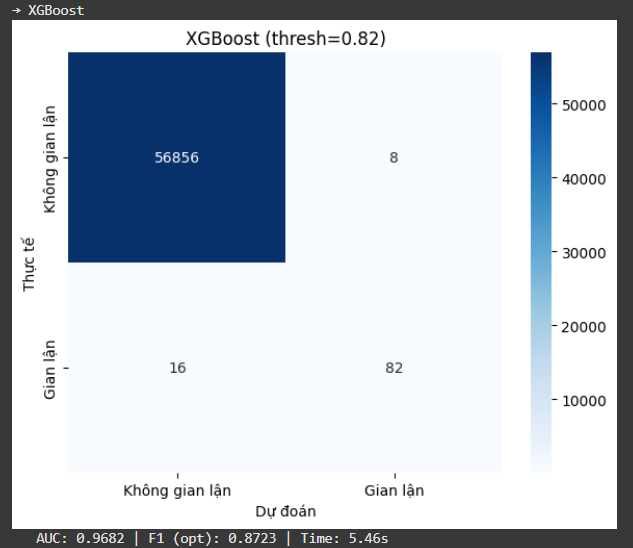
**Kết quả đánh giá các mô hình**

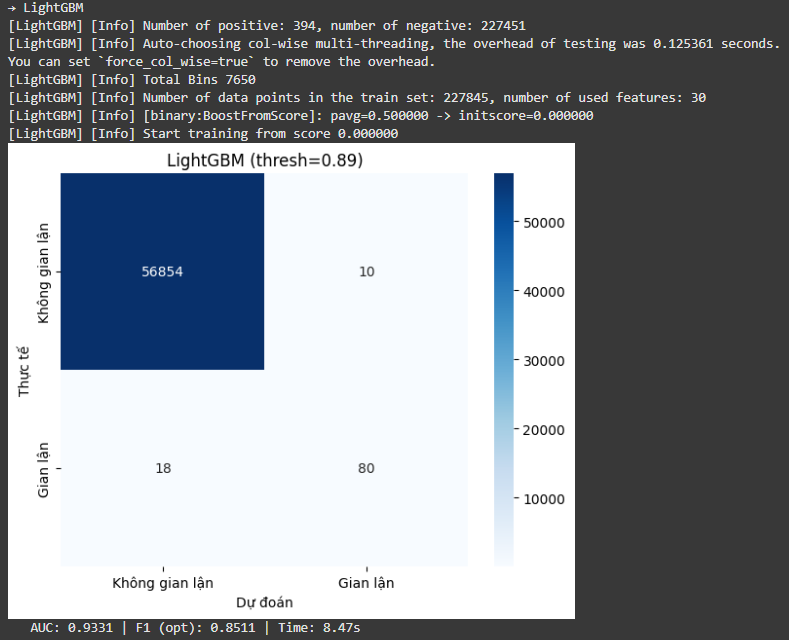
****

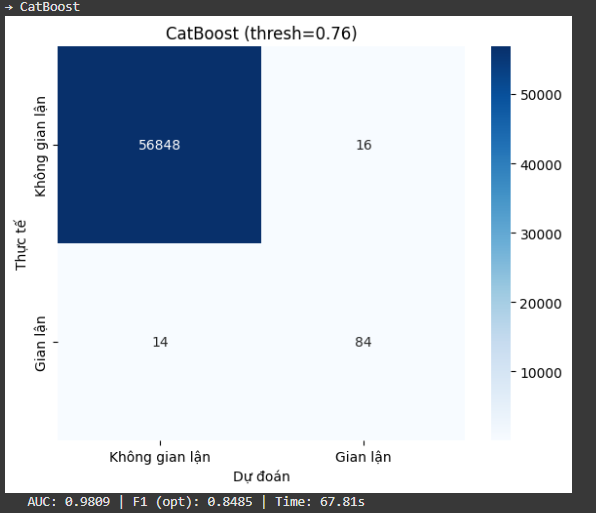
****

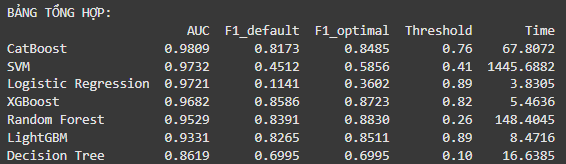
****

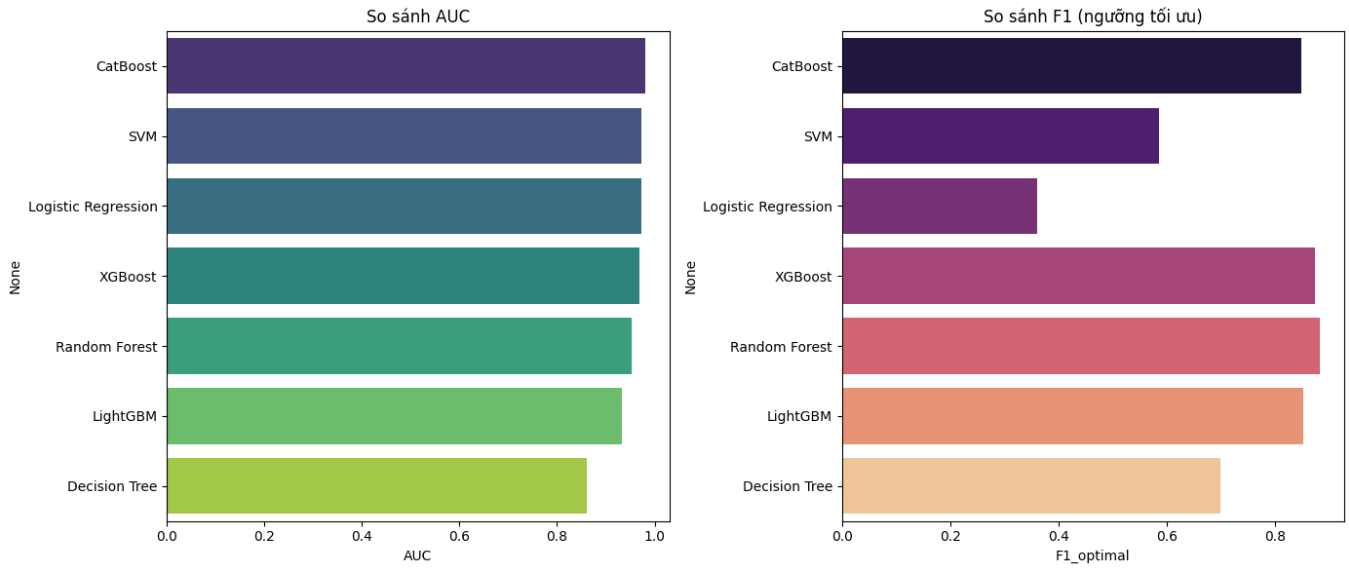
****

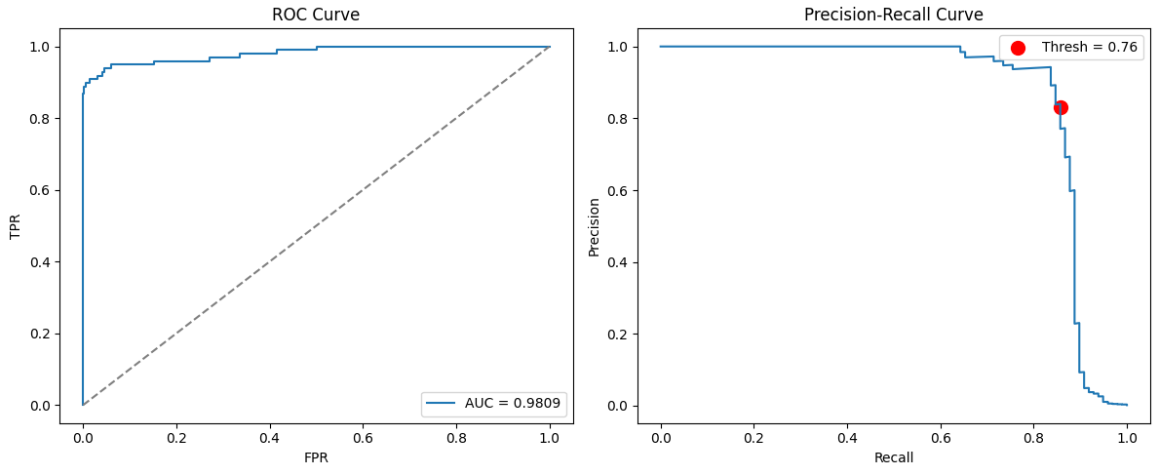
****

****

****

****

****

****

**Kết luận tóm tắt**

Phương pháp của nhóm kết hợp các bước tiền xử lý phù hợp (scale, stratify), chiến lược cân bằng lớp (class\_weight và SMOTE), so sánh đa mô hình và sử dụng các chỉ số/biểu đồ phù hợp để chọn mô hình tối ưu. Các bước bổ sung khả thi: tìm kiếm siêu-tham số (hyperparameter tuning), triển khai SHAP để giải thích quyết định, và thử thêm kỹ thuật ensemble/stacking để cải thiện hiệu năng trên lớp gian lận hiếm.