

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA KHOA HỌC ỨNG DỤNG



BÁO CÁO BÀI TẬP LỚN  
MT2009 - XÁC SUẤT VÀ THỐNG KÊ

PHÂN TÍCH DỮ LIỆU GIAO  
DỊCH NGÂN HÀNG GIẢ MẠO

LỚP: P03 - NHÓM: 36

GIẢNG VIÊN HƯỚNG DẪN: Nguyễn Đình Huy

THÀNH VIÊN NHÓM:

STT	Họ và tên	MSSV	Ghi chú
1	Phạm Nguyễn Anh Khoa	2452553	Nhóm trưởng
2	Nguyễn Trần Phúc Thanh	2413115	
3	Nguyễn Lê Đông Nghi	2452818	
4	Lý Tác Bằng	2452150	
5	Vũ Đình Hoàng Phúc	2412769	
6	Nguyễn Vinh Quang	2412842	
7	Nguyễn Huỳnh Gia Phúc	2412733	

Tp. Hồ Chí Minh, Tháng 11/2025

**Bảng phân công công việc**

STT	Họ và tên	MSSV	Ghi chú	Hoàn thành
1	Phạm Nguyễn Anh Khoa	2452553	Thống kê mô tả, thống kê suy diễn, thống kê phân loại code, duyệt bài	100%
2	Nguyễn Trần Phúc Thanh	2413115	Thống kê mô tả, thống kê phân loại, cơ sở lý thuyết, tổng hợp báo cáo	100%
3	Nguyễn Lê Đông Nghi	2452818	Thống kê suy diễn, thống kê phân loại, code, duyệt bài	100%
4	Lý Tác Bằng	2452150	Cơ sở lý thuyết, thống kê suy diễn, code	100%
5	Vũ Đình Hoàng Phúc	2412769	Cơ sở lý thuyết, thống kê mô tả, code	100%
6	Nguyễn Vinh Quang	2412842	Cơ sở lý thuyết, thống kê phân loại, viết báo cáo	100%
7	Nguyễn Huỳnh Gia Phúc	2412733	Cơ sở lý thuyết, thống kê mô tả, tổng hợp báo cáo	100%

## LỜI MỞ ĐẦU

Sự bùng nổ của các giao dịch điện tử và thương mại toàn cầu đã biến thẻ tín dụng trở thành phương thức thanh toán không thể thiếu. Tuy nhiên, sự tiện lợi này luôn đi kèm với rủi ro gia tăng từ các hành vi gian lận thẻ tín dụng (Credit Card Fraud). Những hoạt động này gây ra tổn thất hàng tỷ đô la mỗi năm cho ngành tài chính, đe dọa trực tiếp đến niềm tin của người tiêu dùng và sự ổn định của hệ thống thanh toán điện tử.

Trước tình hình đó, việc phát triển các cơ chế phòng vệ tự động, có khả năng phát hiện các giao dịch bất thường một cách nhanh chóng và chính xác, là yêu cầu cấp thiết. Các hệ thống này cần phải hoạt động hiệu quả trong môi trường dữ liệu lớn và phức tạp.

Bài toán dự đoán gian lận giao dịch thẻ tín dụng được xem là một thách thức lớn đối với lĩnh vực Khoa học Dữ liệu vì những đặc điểm cố hữu sau:

- Tính mất cân bằng dữ liệu nghiêm trọng:** Đây là rào cản lớn nhất. Tỷ lệ giao dịch gian lận (Positive Class) chỉ chiếm một phần trăm rất nhỏ trong tổng số các giao dịch. Việc này làm sai lệch quá trình học của các thuật toán truyền thống, khiến mô hình có xu hướng dự đoán phần lớn là giao dịch hợp lệ, dẫn đến tỷ lệ bỏ sót gian lận thực tế cao.
- Tính bảo mật và phức tạp của Dữ liệu:** Để bảo vệ thông tin cá nhân và tài chính, các đặc trưng của giao dịch thường được ẩn danh hóa thông qua các kỹ thuật như Phân tích Thành phần Chính (PCA). Điều này tạo ra một tập dữ liệu trừu tượng với các biến đã được biến đổi ( $V_1$  đến  $V_{28}$ ), đòi hỏi các phương pháp thống kê mạnh mẽ để phân tích và trích xuất ý nghĩa.
- Yêu cầu về Hiệu suất Đánh giá:** Trong bài toán này, việc giảm thiểu cảnh báo giả (giao dịch hợp lệ bị gắn nhãn gian lận) và tối đa hóa tỷ lệ phát hiện gian lận thực tế là tối quan trọng. Do đó, các thước đo hiệu suất truyền thống không còn phù hợp, mà cần phải tập trung vào các chỉ số chuyên biệt cho dữ liệu mất cân bằng.

Để giải quyết triệt để các thách thức trên, báo cáo này đề xuất một quy trình phân tích và mô hình hóa chặt chẽ, được thực hiện trên tập dữ liệu giao dịch thẻ tín dụng lớn và thực tế. Mục tiêu cốt lõi của nghiên cứu là xây dựng một mô hình dự đoán có khả năng tổng quát hóa (generalize) tốt trên dữ liệu mới, đạt được sự cân bằng tối ưu giữa độ chính xác và khả năng phát hiện.

Phương pháp tiếp cận của nhóm bao gồm:

- Phân tích Thống kê và Khám phá Dữ liệu:** Sử dụng các công cụ thống kê để hiểu rõ cấu trúc và sự khác biệt về phân phối của dữ liệu giữa hai lớp giao dịch.
- Kỹ thuật Xử lý Mất cân bằng:** Áp dụng các phương pháp lấy mẫu tiên tiến để chuẩn bị dữ liệu, tạo môi trường học tập công bằng hơn cho các thuật toán.
- Mô hình hóa và Lựa chọn Mô hình Tối ưu:** Thực hiện xây dựng và so sánh hiệu suất của các mô hình học máy khác nhau. Việc lựa chọn cuối cùng được dựa trên chỉ số AUPRC (Area Under Precision-Recall Curve), một tiêu chí đánh giá được thừa nhận rộng rãi cho các bài toán phân loại với dữ liệu mất cân bằng nghiêm trọng.

Báo cáo này sẽ trình bày chi tiết từng giai đoạn, từ xử lý dữ liệu thô, đến phân tích kết quả và đề xuất mô hình dự đoán hiệu quả nhất để ứng dụng trong thực tiễn.

## LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn chân thành đến Trường Đại học Bách Khoa - ĐHQG TP.HCM và Khoa Khoa học và Kỹ thuật Máy tính đã tạo điều kiện thuận lợi, cung cấp môi trường học tập năng động và các tài nguyên cần thiết để chúng em có thể hoàn thành môn học này. Những kiến thức nền tảng được trang bị tại trường chính là hành trang vững chắc giúp chúng em tiếp cận và giải quyết các vấn đề thực tế trong bài tập lớn.

Đặc biệt, chúng em xin bày tỏ lòng biết ơn sâu sắc đến Thầy Nguyễn Đình Huy, giảng viên hướng dẫn môn Xác suất và Thống kê. Thầy đã tận tình truyền đạt kiến thức, định hướng đề tài và đưa ra những nhận xét quý báu trong suốt quá trình thực hiện. Sự hướng dẫn của thầy không chỉ giúp chúng em hiểu rõ hơn về lý thuyết thống kê suy diễn và các mô hình phân loại mà còn giúp chúng em nắm bắt được tư duy phân tích dữ liệu một cách khoa học và chặt chẽ.

Cuối cùng, dù đã nỗ lực hết sức để hoàn thiện báo cáo với tinh thần nghiêm túc và cầu thị, nhưng do giới hạn về mặt thời gian và kiến thức nên bài làm khó tránh khỏi những thiếu sót. Chúng em rất mong nhận được sự thông cảm và những ý kiến đóng góp thẳng thắn từ thầy để bài báo cáo được hoàn thiện hơn, cũng như giúp chúng em rút ra những bài học kinh nghiệm quý giá cho các dự án trong tương lai.

# Mục lục

LỜI MỞ ĐẦU	i
LỜI CẢM ƠN	ii
DANH MỤC HÌNH ẢNH	v
DANH MỤC BẢNG BIỂU	vi
<b>1 CƠ SỞ LÝ THUYẾT</b>	<b>1</b>
1.1 Tổng quan dữ liệu . . . . .	1
1.2 Các phương pháp kiểm định thống kê . . . . .	1
1.2.1 Kiểm định Mann-Whitney U-test . . . . .	1
1.2.2 Kiểm định Kolmogorov-Smirnov (KS-Test) . . . . .	3
1.2.3 Kiểm định T-test (Sàng lọc đặc trưng) . . . . .	4
1.3 Xử lý mất cân bằng dữ liệu . . . . .	4
1.3.1 Tăng mẫu (Oversampling) . . . . .	4
1.3.2 Giảm mẫu (Undersampling) . . . . .	5
1.4 Các mô hình dự đoán phân loại . . . . .	5
1.5 Chỉ số đánh giá mô hình . . . . .	8
1.6 Kiểm định giả thuyết về trung bình . . . . .	9
1.6.1 Kiểm định hai mẫu với giả định phương sai bằng nhau (Student's two-sample t-test with equal variances) . . . . .	9
1.6.2 Kiểm định hai mẫu với giả định phương sai khác nhau (Welch's t-test unequal variances) . . . . .	10
1.6.3 Quy tắc ra quyết định . . . . .	10
1.7 Các tiêu chuẩn đánh giá mô hình . . . . .	11
1.8 Sơ đồ khối quy trình thực hiện . . . . .	12
<b>2 TIỀN XỬ LÝ DỮ LIỆU</b>	<b>13</b>
2.1 Đọc dữ liệu . . . . .	13
2.2 Làm sạch dữ liệu . . . . .	13
<b>3 THỐNG KÊ MÔ TẢ</b>	<b>14</b>
3.1 Ma Trận Tương Quan . . . . .	14
3.2 Biến Không Qua Biến Đổi PCA . . . . .	15
3.2.1 Biến Amount . . . . .	15

3.2.2	Biến Time: . . . . .	17
3.3	Biểu đồ 28 biến PCA . . . . .	20
3.4	Phân tích thành phần ngoại lai . . . . .	22
<b>4</b>	<b>THỐNG KÊ SUY DIỄN</b>	<b>25</b>
4.1	Kiểm định giả thuyết về đặc trưng số tiền giao dịch <b>Amount</b> . . . . .	25
4.2	Kiểm định giả thuyết về khả năng phân loại của bộ biến PCA (V1:V28) . .	25
<b>5</b>	<b>THỐNG KÊ PHÂN LOẠI</b>	<b>27</b>
5.1	Chuẩn bị số liệu cho huấn luyện . . . . .	27
5.2	Xử lý mất cân bằng dữ liệu . . . . .	28
5.2.1	Random Undersampling (Giảm mẫu ngẫu nhiên) . . . . .	29
5.2.2	Random Oversampling (Tăng mẫu ngẫu nhiên) . . . . .	30
5.2.3	SMOTE (Synthetic Minority Over-sampling Technique) . . . . .	31
5.2.4	ROSE (Random Over-Sampling Examples) . . . . .	32
5.2.5	Đánh giá hiệu năng lấy mẫu và lựa chọn phương pháp tối ưu . . . .	33
5.3	Xây dựng mô hình phân loại . . . . .	35
5.3.1	Huấn luyện mô hình . . . . .	35
5.3.2	Đánh giá hiệu năng phân loại và kết luận phương pháp tối ưu . . .	36
5.3.3	Trích xuất bộ biến quan trọng cho từng mô hình . . . . .	39
5.3.4	Phân tích và Nhận xét . . . . .	40
<b>6</b>	<b>KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN</b>	<b>42</b>
6.1	Đánh giá và Kết luận chung . . . . .	42
6.2	Hướng phát triển và Mở rộng nghiên cứu . . . . .	43
6.2.1	Thực hiện phân tích sâu hơn đối với biến Time . . . . .	43
6.2.2	Cải thiện chất lượng dữ liệu đầu vào nhằm nâng cao hiệu suất mô hình Logistic Regression . . . . .	44
6.2.3	Tự động hóa quy trình Kiểm định Phân phối (Automated Normality Testing) . . . . .	44
6.2.4	Giả thuyết về "Mạng lưới Bot" dựa trên Phân phối Ngoại lai . . . .	44
	<b>TÀI LIỆU THAM KHẢO</b>	<b>45</b>

## Danh sách hình vẽ

1	Sơ đồ khối quy trình xử lý và phân tích dữ liệu . . . . .	12
2	Bảng tóm tắt dataset thô . . . . .	13
3	Bảng tóm tắt dataset đã làm sạch . . . . .	14
4	Ma trận tương quan giữa các biến . . . . .	15
5	Biểu đồ mật độ biến Amount . . . . .	16
6	Biểu đồ mật độ biến Amount phân theo class sau khi đã biến đổi Logarit .	17
7	Biểu đồ mật độ của biến Time . . . . .	18
8	Mật độ biến Time phân theo class . . . . .	19
9	Số lượng giao dịch theo ngày, giờ tính từ giao dịch đầu tiên . . . . .	19
10	Biểu đồ hộp phân phối 28 biến PCA theo Class . . . . .	21
11	Biểu đồ hộp phân phối 28 biến PCA đã zoom . . . . .	22
12	Phân phối số lượng Outlier theo từng Class . . . . .	24
13	So sánh phân bố số lượng mẫu giữa dữ liệu gốc và các phương pháp lấy mẫu	29
14	Đường cong Precision-Recall (PR Curve) so sánh các phương pháp . . . .	34
15	So sánh đường cong Precision-Recall giữa 3 mô hình trên tập Test gốc . . .	38
16	Feature Importance của mô hình XGBoost . . . . .	39
17	Feature Importance của mô hình Random Forest . . . . .	39
18	Feature Importance của mô hình Logistic Regression . . . . .	40

## Danh sách bảng

1	Số lượng quan sát trùng lặp và giá trị thiếu . . . . .	13
2	Ví dụ minh họa sau khi gom biến . . . . .	20
3	Kết quả phân tích Outliers . . . . .	23
4	Phân phối số lượng Outlier trên mỗi giao dịch (0–28) . . . . .	23
5	Kết quả GT1 (Mann-Whitney U-test <code>logAmount_Scaled</code> ) . . . . .	25
6	Kết quả GT2 (KS-Test <code>V1:V28</code> ): Top 10 biến có hiệu quả phân loại cao nhất	27
7	Kết quả GT2 (KS-Test <code>V1:V28</code> ): Top 5 biến có hiệu quả phân loại thấp nhất	27
8	Thống kê số lượng mẫu trước và sau khi áp dụng Random Undersampling	29
9	Thống kê số lượng mẫu trước và sau khi áp dụng Random Oversampling .	31
10	Thống kê số lượng mẫu trước và sau khi áp dụng SMOTE . . . . .	32
11	Thống kê số lượng mẫu sau khi áp dụng ROSE . . . . .	33
12	So sánh hiệu năng của các phương pháp lấy mẫu . . . . .	34
13	Kết quả tinh chỉnh tham số cho mô hình RF và XGB . . . . .	36
14	Kết quả đánh giá hiệu suất các mô hình qua 3 lần tinh chỉnh . . . . .	38



# 1 CƠ SỞ LÝ THUYẾT

## 1.1 Tổng quan dữ liệu

Bộ dữ liệu **Credit Card Fraud Detection** bao gồm các giao dịch thẻ tín dụng thực tế, trong đó có cả giao dịch hợp lệ và giao dịch gian lận. Cả tập dữ liệu được thực hiện vào tháng 9 năm 2013, với mỗi dòng dữ liệu đại diện cho một giao dịch riêng lẻ. [11]

Các biến chính trong bộ dữ liệu:

Tên biến	Đơn vị	Ý nghĩa
Time	Giây	Thời điểm giao dịch tính từ giao dịch đầu tiên trong dataset
V1-V28	-	Các đặc trưng đã được biến đổi từ dữ liệu gốc bằng PCA để bảo mật
Amount	USD	Số tiền giao dịch
Class	-	Nhãn giao dịch: 1 = gian lận, 0 = hợp lệ

Mục tiêu phân tích:

- Hiểu phân bố của các giao dịch hợp lệ và gian lận.
- Kiểm định các giả thuyết thống kê để tìm ra đặc trưng quan trọng.
- Xây dựng và đánh giá các mô hình học máy để dự đoán gian lận.

## 1.2 Các phương pháp kiểm định thống kê

Trong nghiên cứu này, chúng tôi sử dụng các kiểm định thống kê để đánh giá đặc điểm dữ liệu và lựa chọn đặc trưng.

### 1.2.1 Kiểm định Mann-Whitney U-test

#### a. Cơ sở lý thuyết

**Định nghĩa:** Đây là kiểm định phi tham số (*Non-parametric*).

**Ưu điểm nổi bật:** Nó không yêu cầu giả định dữ liệu phải tuân theo phân phối chuẩn (*Normal Distribution*) và ít nhạy cảm với các giá trị ngoại lai (*outliers*) hơn so với t-test.

**Mục đích:** Xác định xem liệu hai mẫu độc lập ( $X$  và  $Y$ ) có được rút ra từ cùng một quần thể hay không, hay nói cách khác, liệu có sự khác biệt đáng kể về phân phối (thường được hiểu là trung vị) giữa hai nhóm hay không.

#### b. Giả thuyết Thống kê

Kiểm định này được thiết lập dựa trên việc so sánh vị trí của các quan sát giữa hai nhóm:

**Giả thuyết Null ( $H_0$ ):** Không có sự khác biệt về phân phối (trung vị) giữa 2 nhóm.

$$H_0 : \text{Median}(X) = \text{Median}(Y)$$

**Giả thuyết Đối ( $H_a$ ):** Có sự khác biệt về phân phối (trung vị) giữa hai nhóm.

$$H_a : \text{Median}(X) \neq \text{Median}(Y)$$

**c. Nguyên tắc hoạt động (Dựa trên Xếp hạng - Ranks)**

Kiểm định U-test dựa trên việc tính toán tổng hạng của dữ liệu qua các bước sau:

**Gộp mẫu:** Gộp tất cả dữ liệu từ cả hai nhóm ( $n_1$  và  $n_2$ ) thành một tập hợp duy nhất.

**Xếp hạng:** Xếp hạng tất cả các quan sát từ giá trị nhỏ nhất (hạng 1) đến giá trị lớn nhất (hạng  $N = n_1 + n_2$ ).

**Tính Tổng Hạng:** Tính tổng các hạng ( $R_1$  và  $R_2$ ) riêng biệt cho các quan sát thuộc Nhóm 1 và Nhóm 2.

**Tính Giá trị U:** Giá trị  $U$  được tính bằng công thức:

$$U_1 = R_1 - \frac{n_1(n_1 + 1)}{2}$$

$$U_2 = R_2 - \frac{n_2(n_2 + 1)}{2}$$

Giá trị  $U$  được sử dụng trong kiểm định là giá trị nhỏ hơn:

$$U = \min(U_1, U_2)$$

**Kết luận:** Dựa trên giá trị  $U$  và giá trị tới hạn (hoặc p-value từ phần mềm), nếu p-value nhỏ hơn mức ý nghĩa  $\alpha$  (thường là 0.05), ta bác bỏ  $H_0$ , kết luận hai nhóm có sự khác biệt đáng kể về trung vị.

**d. Ví dụ: Ứng dụng trong Phát hiện Gian lận Thẻ Tín dụng**

**Phân tích biến Amount: Định nghĩa:** Biến Amount (Số tiền giao dịch) là một trong những đặc trưng quan trọng nhất để phân biệt gian lận và hợp lệ. **Vấn đề đặt ra:** Dữ liệu Amount thường bị lệch dương do một số giá trị lớn, khiến t-test không phù hợp. **Mục đích:** Xác định liệu trung vị số tiền giao dịch có khác biệt đáng kể giữa nhóm Gian lận và nhóm Hợp lệ hay không.

**Thiết lập kiểm định U-test: Nhóm 1:** Tập hợp số tiền giao dịch của các giao dịch Hợp lệ ( $Y = 0$ ). **Nhóm 2:** Tập hợp số tiền giao dịch của các giao dịch Gian lận ( $Y = 1$ ). **Giả thuyết Null ( $H_0$ ):** Trung vị số tiền giao dịch Gian lận bằng trung vị số tiền giao dịch Hợp lệ. **Giả thuyết Đối ( $H_a$ ):** Trung vị số tiền giao dịch Gian lận khác trung vị số tiền giao dịch Hợp lệ.

**Kết quả: Bác bỏ  $H_0$  ( $p < 0.05$ ):** Có sự khác biệt đáng kể về trung vị số tiền giao dịch giữa hai nhóm.  $\Rightarrow$  Biến Amount là một đặc trưng có giá trị phân biệt cao và nên được đưa vào mô hình.

**Không bác bỏ  $H_0$  ( $p \geq 0.05$ ):** Không có bằng chứng cho thấy trung vị số tiền giao dịch khác nhau đáng kể.  $\Rightarrow$  Biến Amount có thể không phải là đặc trưng phân biệt mạnh mẽ như dự kiến (dù trong thực tế hiếm khi xảy ra với Amount).

## 1.2.2 Kiểm định Kolmogorov-Smirnov (KS-Test)

### a. Tổng quan và Giả định

Kiểm định KS dùng để so sánh phân phối thực nghiệm (từ dữ liệu mẫu) với một phân phối lý thuyết (ví dụ: chuẩn, uniform, exponential...), hoặc so sánh hai mẫu độc lập. KS là kiểm định phi tham số, không yêu cầu phân phối chuẩn.

**Giả định:** (i) biến quan sát độc lập, (ii) biến có trật tự, (iii) biến có phân phối giống nhau.

### b. Định nghĩa

Hàm phân phối thực nghiệm  $F_n$  cho  $n$  biến quan sát có trật tự độc lập và phân phối giống nhau  $X_i$  được định nghĩa là:

$$F_n(x) = \frac{\text{số lượng phần tử } x_i \leq x}{n} = \frac{1}{n} \sum_{i=1}^n 1_{(-\infty, x]} X_i$$

Với:

$$1_{(-\infty, x]} = \begin{cases} 1 & \text{nếu } X_i \leq x \\ 0 & \text{nếu } X_i > x \end{cases}$$

### c. Quy tắc ra quyết định

Bác bỏ  $H_0$  nếu  $p < \alpha$ , không bác bỏ  $H_0$  nếu  $p \geq \alpha$ . Ta có:

$$c(\alpha) = \sqrt{-\ln\left(\frac{\alpha}{2}\right) \cdot \frac{1}{2}}$$

**Kiểm định hai mẫu (Trường hợp 1 - So sánh với phân phối lý thuyết):**

$$H_0 : F_n(x) = F(x) \quad \text{vs} \quad H_1 : F_n(x) \neq F(x)$$

$$D_n = \sup_x |F_n(x) - F(x)|$$

Với mức ý nghĩa:

$$D_\alpha = \frac{c_\alpha}{\sqrt{n}}$$

Bác bỏ  $H_0 : D_n > D_\alpha$

**Kiểm định hai mẫu (Trường hợp 2 - So sánh hai mẫu độc lập):**

$$H_0 : F_{n,1}(x) = F_{m,2}(x) \quad \text{vs} \quad H_1 : F_{n,1}(x) \neq F_{m,2}(x)$$

$$D_{n,m} = \sup_x |F_{1,n}(x) - F_{2,m}(x)|$$

Bác bỏ  $H_0 : D_{n,m} > c(\alpha) \sqrt{\frac{n+m}{n \cdot m}}$

### d. Thống kê D và Ứng dụng trong đề tài

**Thống kê D:** Đo khoảng cách lớn nhất giữa hai hàm phân phối tích lũy thực nghiệm (ECDF) của hai mẫu.

$$D = \sup_x |F_{1,n}(x) - F_{2,m}(x)|$$

**Ứng dụng trong đề tài: 1. Kiểm tra toàn vẹn dữ liệu:** So sánh phân phối trước và sau khi xử lý làm sạch (xóa trùng lặp) để đảm bảo dữ liệu không bị méo mó. **2. Đánh giá đặc trưng:** Đánh giá mức độ tách biệt của các biến PCA ( $V_1 \dots V_{28}$ ) giữa nhóm Fraud và Non-Fraud.

### 1.2.3 Kiểm định T-test (Sàng lọc đặc trưng)

Mặc dù dữ liệu không chuẩn, T-test vẫn được sử dụng trong bước sàng lọc nhanh (screening) nhờ khả năng tính toán đơn giản khi so sánh trung bình của số lượng lớn các biến ( $V_1 \dots V_{28}$ ).

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

(Sử dụng phiên bản Welch's T-test để chấp nhận phương sai không đồng nhất).

## 1.3 Xử lý mất cân bằng dữ liệu

Dữ liệu mất cân bằng (imbalanced data) xảy ra khi các lớp phân loại không được đại diện đồng đều - thường là một lớp thiếu số bị áp đảo bởi lớp đa số. Sự chênh lệch này khiến các mô hình học máy có xu hướng thiên vị, bỏ qua lớp thiếu số và làm giảm khả năng khái quát hóa. Các phương pháp lấy mẫu lại (resampling) được sử dụng để điều chỉnh sự phân phối lớp, nhằm cải thiện hiệu suất phân loại.

### 1.3.1 Tăng mẫu (Oversampling)

Các phương pháp tăng mẫu nhằm tăng số lượng mẫu thuộc lớp thiểu số  $S_{\min}$  để cân bằng với số lượng mẫu thuộc lớp đa số  $S_{\text{maj}}$ .

- **ROSE (Random Over-Sampling Examples)**

Nguyên lý: ROSE là một phương pháp tăng mẫu tiên tiến, sử dụng phương pháp Smoothed Bootstrap để tạo ra các ví dụ nhân tạo. Quá trình này tương đương với việc lấy mẫu dữ liệu từ ước lượng mật độ kernel có điều kiện  $f(x|Y_j)$  của lớp thiểu số  $Y_j$ . Quá trình sinh ra một mẫu tổng hợp mới  $(x^*, y^*)$  được thực hiện như sau:

1. Chọn Lớp ( $y^*$ ): Chọn lớp  $y^* = Y_j$  với xác suất  $\pi_j$ .
2. Chọn Điểm Mầm ( $x_i$ ): Chọn một quan sát hiện có  $(x_i, y_i)$  từ tập huấn luyện  $T_n$ , sao cho  $y_i = y^*$ , với xác suất  $1/n_j$ .
3. Sinh Mẫu Tổng hợp ( $x^*$ ): Lấy mẫu  $x^*$  từ phân phối  $KH_j(\cdot, x_i)$ .

- **SMOTE (Synthetic Minority Over-sampling TEchnique)**

Nguyên lý: SMOTE tạo ra các mẫu tổng hợp mới bằng cách nội suy tuyến tính giữa một mẫu thiểu số được chọn và một trong số  $K$  láng giềng gần nhất của nó thuộc cùng lớp thiểu số:

$$x_{\text{new}} = x_i + \lambda \cdot (x_{\hat{i}} - x_i), \quad \lambda \sim \text{Uniform}[0, 1]$$

- **Random Oversampling (ROS)**

Nguyên lý: ROS tăng số lượng mẫu lớp thiểu số  $S_{\min}$  bằng cách lấy mẫu có hoàn lại và thêm chúng vào tập dữ liệu cho đến khi kích thước lớp thiểu số mới  $|S'_{\min}| \approx |S_{\text{maj}}|$ . Tập dữ liệu huấn luyện mới  $S' = S'_{\min} \cup S_{\text{maj}}$ .

### 1.3.2 Giảm mẫu (Undersampling)

Phương pháp giảm mẫu (undersampling) nhằm giải quyết sự mất cân bằng lớp bằng cách giảm số lượng mẫu thuộc lớp đa số  $S_{\text{maj}}$ .

- **Random Undersampling (RUS)**

Nguyên lý: RUS chọn ngẫu nhiên các mẫu từ lớp đa số và loại bỏ chúng (lấy mẫu không hoàn lại) cho đến khi kích thước lớp đa số mới  $|S'_{\text{maj}}| \approx |S_{\text{min}}|$ , tạo thành tập huấn luyện mới  $S' = S_{\text{min}} \cup S'_{\text{maj}}$ .

### 1.4 Các mô hình dự đoán phân loại

Mô hình phân loại là một phương pháp trong **Machine Learning**, thuộc nhánh **Supervised Learning** (học có giám sát), được sử dụng để dự đoán nhãn (label) của các mẫu dữ liệu đầu vào.

Trong quá trình xây dựng, mô hình phân loại được huấn luyện trên tập dữ liệu huấn luyện (*training set*) nhằm học mối quan hệ giữa các đặc trưng (*features*) và nhãn mục tiêu. Sau đó, mô hình được đánh giá hiệu suất trên tập dữ liệu kiểm thử (*test set*) để kiểm tra khả năng tổng quát hóa. Cuối cùng, mô hình đã huấn luyện được sử dụng để dự đoán nhãn cho các dữ liệu mới chưa từng thấy trước đó.

Với đề tài “Phát hiện gian lận thẻ tín dụng”, dữ liệu sẽ được chia thành hai tập: 70% dữ liệu dùng cho *Training set* và 30% dữ liệu dùng cho *Test set*.

### Mô hình 1: Hồi quy logistic nhị thức (Binomial Logistic Regression)

Trong thống kê, hồi quy logistic dự đoán xác suất một mẫu thuộc về một lớp cụ thể. Mô hình thường được dùng cho phân loại nhị phân (Yes/No, True/False, 0/1). Hàm sigmoid được sử dụng để chuyển đầu vào thành giá trị xác suất nằm trong khoảng (0;1).

- **Hàm logistic (Sigmoid Function):**

Biến mục tiêu:

$$Y = \begin{cases} 1, & \text{nếu giao dịch là gian lận (Fraud)} \\ 0, & \text{nếu giao dịch hợp lệ (Non-Fraud)} \end{cases}$$

Sigmoid function:

$$P(Y = 1|X) = \sigma = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \dots + \beta_k X_k)}}, \quad P(Y = 0|X) = 1 - \sigma$$

Dạng log-odds (Logit):

$$\text{logit}(\sigma) = \ln\left(\frac{\sigma}{1 - \sigma}\right) = \beta_0 + \beta_1 X_1 + \dots + \beta_k X_k$$

Ý nghĩa của  $\beta_i$  đối với biến  $X_i$ :

$\beta_i > 0$  : Biến  $X_i$  làm tăng xác suất giao dịch gian lận.

$\beta_i < 0$  : Biến  $X_i$  làm giảm xác suất giao dịch gian lận.

## Mô hình 2: Random Forest

Random Forest là một thuật toán được xếp vào phương pháp học tổ hợp (*Ensemble Learning*). Thuật toán này kết hợp nhiều Decision Trees để tạo ra mô hình có khả năng giảm sai số và tránh hiện tượng quá khớp (*overfitting*).

- **Các giả định cơ bản của Random Forest:**

- Mỗi cây đưa ra quyết định độc lập.
- Các cây được huấn luyện trên các mẫu và đặc trưng khác nhau.
- Dữ liệu phải đủ lớn để đảm bảo mỗi cây học được các mẫu riêng biệt.
- Việc kết hợp nhiều cây giúp giảm phương sai và tăng độ chính xác.

- **Cơ chế hoạt động:** Quy trình hoạt động bao gồm các bước:

- Tạo ra N decision trees: mỗi cây được huấn luyện với tập dữ liệu ngẫu nhiên (lấy mẫu có hoàn lại - *Bootstrap Sampling*).
- Chọn đặc trưng ngẫu nhiên: khi tách mỗi nút trong cây, chỉ một tập con ngẫu nhiên của các đặc trưng được xem xét để tìm điểm chia tối ưu.
- Dự đoán từ từng cây: mỗi cây đưa ra dự đoán riêng (phân loại hoặc giá trị hồi quy).
- Kết hợp và đưa ra kết quả: Với phân loại (classification): lấy lớp được đa số cây dự đoán. Với hồi quy (regression): lấy giá trị trung bình của các dự đoán.
- Kết quả cuối cùng là tổng hợp của toàn bộ rừng cây.

## Mô hình 3: XGBoost

Extreme Gradient Boosting (hay XGBoost) là thuật toán được phát triển tối ưu hóa hiệu suất, tốc độ và độ chính xác cao hơn so với các mô hình cây truyền thống như Decision Tree hay Random Forest. Quá trình boosting trong thuật toán được định nghĩa là mỗi cây mới trong mô hình được huấn luyện để sửa lỗi mà các cây trước đó đã mắc phải.

- **Cơ chế hoạt động của XGBoost tổng quát:**

- Khởi tạo mô hình ban đầu: Với phân loại (classification): Mô hình ban đầu dự đoán xác suất trung bình. Với hồi quy (regression): Mô hình đầu tiên dự đoán giá trị trung bình của biến
- Tính sai số (residuals): Sau khi có dự đoán, thuật toán tính độ chênh lệch giữa giá trị thực và giá trị dự đoán. Độ chênh lệch này đóng vai trò như là phần lỗi cần mô hình kế cạnh học.

- Huấn luyện cây kế tiếp ( $f_t$ ): Cây tiếp theo xây dựng dựa trên sai số mô hình trước nhằm giảm lỗi..
- Lặp lại quá trình boosting: Mỗi cây mới được thêm vào cải thiện mô hình đến khi chạm điều kiện dừng (*stopping criterion*).
- Tổng hợp kết quả: Dự đoán cuối cùng được tính bằng tổng có trọng số của các dự đoán từng cây.

• **Cơ chế hoạt động của XGBoost tổng quát:**

**Hàm mục tiêu (*Objective Function*):** Hàm mục tiêu trong XGBoost gồm hai thành phần:

$$\text{obj}(\theta) = \sum_{i=1}^n l(y_i, \hat{y}_i) + \sum_{k=1}^K \Omega(f_k)$$

Trong đó:

- $l(y_i, \hat{y}_i)$ : hàm mất mát (loss function) – đo mức độ sai lệch giữa giá trị thật và giá trị dự đoán.
- $\Omega(f_k)$ : hàm điều chuẩn (regularization term) – giúp mô hình tránh quá phức tạp, từ đó giảm overfitting.

**Hàm điều chuẩn (*Regularization Term*):** Hàm điều chuẩn được định nghĩa là

$$\Omega(f_t) = \gamma T + \frac{1}{2} \lambda O_{\text{value}}^2$$

Trong đó:

- $T$ : số lá trong cây.
- $\gamma$ : hệ số phạt (penalty) cho số lượng lá.
- $\lambda$ : hệ số phạt cho độ lớn của giá trị dự đoán  $O_{\text{value}}$  tại mỗi lá. Khi  $\lambda > 0$  sẽ làm giảm độ nhạy của dự đoán đến các đơn cử quan sát.

Đối với bài toán *phân loại nhị phân*, hàm mất mát được hiểu là:

$$L = \prod_{i=1}^n P(y_i | x_i)$$

Do biểu thức trên nhân rất nhiều số nhỏ dễ gây tràn số, ta lấy log hai vế và thu được biểu thức dưới dạng log-likelihood

$$L(y_i, p_i) = -[y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$$

Viết lại hàm mục tiêu ta có

$$\left[ \sum L(y_i, p_i) \right] + \gamma T + \frac{1}{2} \lambda O_{\text{value}}^2$$

1.5 Chỉ số đánh giá mô hình

Nền tảng Ma trận nhầm lẫn (Confusion Matrix)

Confusion Matrix là một phương pháp đánh giá kết quả bài toán phân loại, xem xét cả độ chính xác và độ bao quát của dự đoán cho từng lớp.

Một confusion matrix gồm 4 chỉ số cơ bản đối với mỗi lớp phân loại:

Dự đoán	Dương tính (Positive)	Âm tính (Negative)
Thực tế Dương tính (True)	True Positive (TP)	True Negative (TN)
Thực tế Âm tính (False)	False Positive (FP)	False Negative (FN)

Trong đó:

- **True Positive (TP):** Giao dịch gian lận được dự đoán đúng (mục tiêu cần tối ưu)
- **True Negative (TN):** Giao dịch hợp lệ được dự đoán đúng
- **False Positive (FP - Type 1 Error):** Giao dịch hợp lệ bị dự đoán là gian lận (cảnh báo sai)
- **False Negative (FN - Type 2 Error):** Giao dịch gian lận bị dự đoán là hợp lệ (lỗi nghiêm trọng nhất)

Ví dụ:

Đối với bài toán phân tích giao dịch thẻ tín dụng với 2 lớp: hợp lệ và gian lận. Nếu xét 100 giao dịch, trong đó 10 giao dịch gian lận, mà mô hình dự đoán cả 100 giao dịch là hợp lệ, độ chính xác là 0,9. Tuy nhiên, 10 giao dịch gian lận bị bỏ qua, cho thấy mô hình chưa đủ tin cậy. Vì vậy cần dùng **Confusion Matrix** để đánh giá tốt hơn.

Trong đó:

- **True Positive (TP):** Giao dịch gian lận được dự đoán đúng (mục tiêu cần tối ưu)
- **True Negative (TN):** Giao dịch hợp lệ được dự đoán đúng
- **False Positive (FP - Type 1 Error):** Giao dịch hợp lệ bị dự đoán là gian lận (cảnh báo sai)
- **False Negative (FN - Type 2 Error):** Giao dịch gian lận bị dự đoán là hợp lệ (lỗi nghiêm trọng nhất)



## Các chỉ số Precision và Recall

- **Precision (Độ chính xác khi mô hình đưa ra cảnh báo):** Số dự đoán chính xác trong tất cả các dự đoán Positive.

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

- **Recall (Độ nhạy, độ phủ):** Số trường hợp Positive được dự đoán đúng trong tổng số trường hợp Positive.

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

## Các chỉ số dựa trên ngưỡng phân loại

- **AUROC (Area Under the ROC Curve):** Diện tích dưới đường cong ROC, biểu diễn TPR (Recall) trên trục Y và FPR trên trục X khi ngưỡng thay đổi từ 0 đến 1. Giá trị AUROC càng gần 1, mô hình càng tốt.
- **AUPRC (Area Under the Precision-Recall Curve):** Diện tích dưới đường cong PR, biểu diễn Precision trên trục Y và Recall trên trục X khi ngưỡng thay đổi. Thước đo này đặc biệt quan trọng cho các tập dữ liệu mất cân bằng nghiêm trọng, tập trung vào lớp thiểu số.

## 1.6 Kiểm định giả thuyết về trung bình

Kiểm định thống kê T-test là một loại kiểm định dùng giá trị thống kê **Student's t-distribution** để kiểm định giả thuyết về giá trị trung bình. Các dạng chính bao gồm: One-sample t-test (kiểm định một mẫu), Independent two-sample t-tests (kiểm định hai mẫu độc lập) và Paired (dependent) t-test (kiểm định hai mẫu phụ thuộc).

**Giả định:** (i) Các biến độc lập, (ii) Dữ liệu phân phối chuẩn hoặc đủ lớn để áp dụng định lý giới hạn trung tâm xấp xỉ phân phối chuẩn.

### 1.6.1 Kiểm định hai mẫu với giả định phương sai bằng nhau (Student's two-sample t-test with equal variances)

Dữ liệu đầu vào: kích thước mẫu  $(n_1, n_2)$ , trung bình mẫu  $(\bar{X}_1, \bar{X}_2)$  và phương sai mẫu  $(s_1, s_2)$ .

Nếu

$$\frac{s_1}{s_2} \in [0.5, 2],$$

thì ta có thể giả định phương sai tổng thể của hai nhóm là bằng nhau, tức  $\sigma_1^2 \approx \sigma_2^2$ , và do đó có thể sử dụng pooled t-test (two-sample t-test with equal variances).

Phương sai gộp (pooled variance):

$$s_p^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}$$

Giá trị thống kê kiểm định  $t_0$ :

$$t_0 = \frac{\bar{x}_1 - \bar{x}_2}{s_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

Bậc tự do:

$$df = n_1 + n_2 - 2$$

### 1.6.2 Kiểm định hai mẫu với giả định phương sai khác nhau (Welch's t-test unequal variances)

Dữ liệu đầu vào: kích thước mẫu  $(n_1, n_2)$ , trung bình mẫu  $(\bar{X}_1, \bar{X}_2)$  và phương sai mẫu  $(s_1, s_2)$ .

Nếu

$$\frac{s_1}{s_2} \notin [0.5, 2],$$

thì giả định phương sai bằng nhau không còn hợp lý. Khi đó ta sử dụng kiểm định Welch (two-sample t-test with unequal variances), vốn không yêu cầu  $\sigma_1^2 = \sigma_2^2$ .

Giá trị thống kê kiểm định  $t_0$  trong kiểm định Welch:

$$t_0 = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

Bậc tự do xấp xỉ (công thức Welch–Satterthwaite):

$$df = \frac{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)^2}{\frac{\frac{s_1^4}{n_1^2(n_1-1)}}{1} + \frac{\frac{s_2^4}{n_2^2(n_2-1)}}{1}}$$

### 1.6.3 Quy tắc ra quyết định

Giả sử ta thực hiện t-test với giá trị thống kê kiểm định  $t_0$  và bậc tự do  $df$ .

Bác bỏ  $H_0$  nếu  $p < \alpha$ , không bác bỏ  $H_0$  nếu  $p \geq \alpha$ .

#### Kiểm định một phía (one-tailed test)

Ta quan tâm đến một hướng khác biệt cụ thể:

$$H_0 : \mu_1 - \mu_2 = 0 \quad \text{vs} \quad H_1 : \mu_1 - \mu_2 > 0.$$

- Nếu  $t_0 > 0$ , p-value được tính là:

$$p = P(T \geq t_0), \quad T \sim t(df)$$

- Nếu  $t_0 < 0$  (vì quan tâm hướng ngược lại):

$$p = P(T \leq t_0), \quad T \sim t(df)$$

### Kiểm định hai phía (two-tailed test)

Ta quan tâm đến cả hai hướng khác biệt, tức chỉ cần biết hai trung bình có khác nhau hay không:

$$H_0 : \mu_1 - \mu_2 = 0 \quad \text{vs} \quad H_1 : \mu_1 - \mu_2 \neq 0.$$

- Tính p-value theo trị tuyệt đối:

$$p = P(T \geq |t_0|) + P(T \leq -|t_0|) = 2 \cdot P(T \geq |t_0|), \quad T \sim t(df)$$

## 1.7 Các tiêu chuẩn đánh giá mô hình

Do đặc thù dữ liệu mất cân bằng, độ chính xác (Accuracy) không phải là thước đo tin cậy. Nghiên cứu tập trung vào các chỉ số sau:

- **Precision (Độ chính xác dương tính):** Tỷ lệ số ca gian lận dự đoán đúng trên tổng số cảnh báo. (Quan trọng để tránh làm phiền khách hàng).

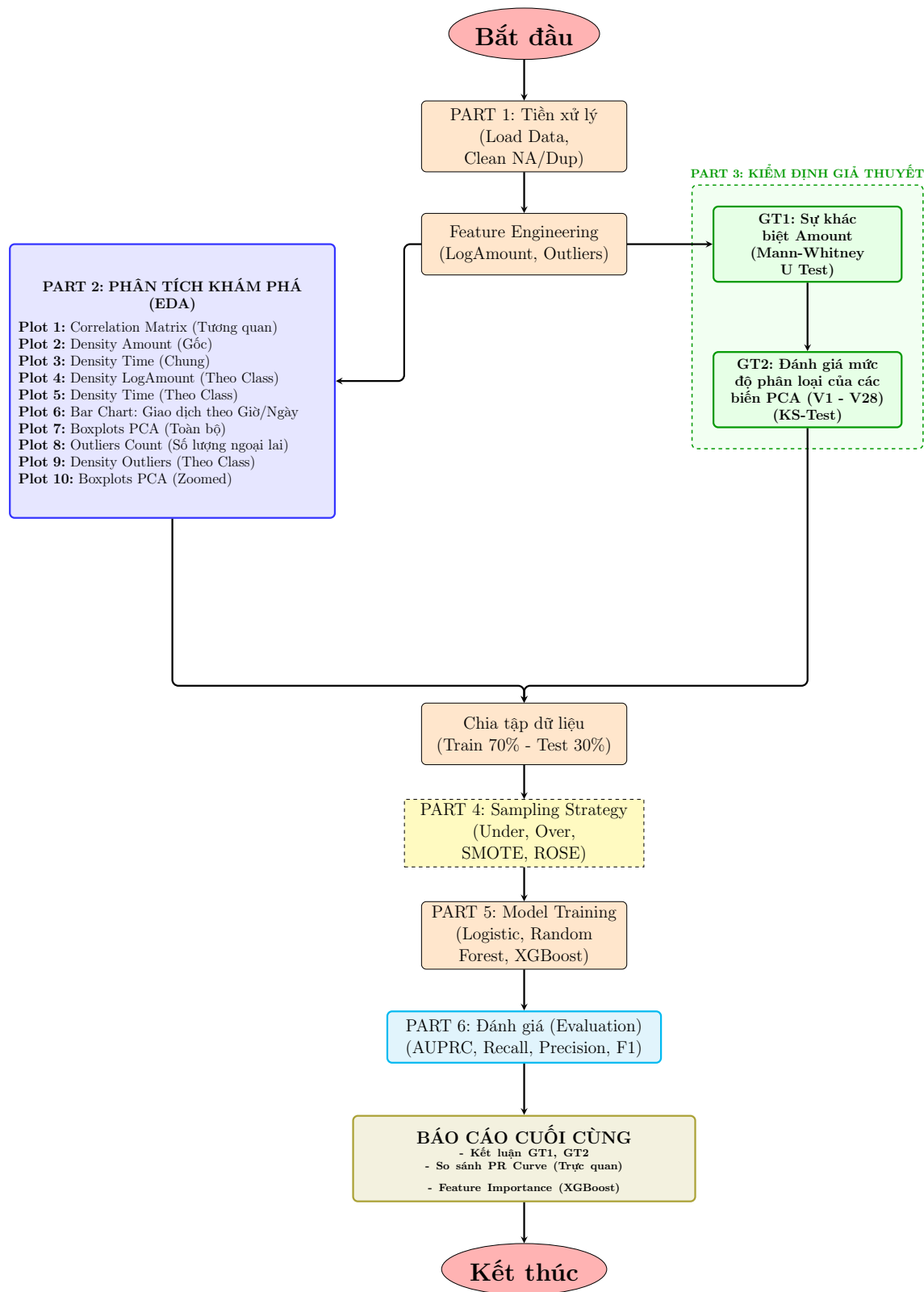
$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall (Độ nhạy):** Tỷ lệ số ca gian lận phát hiện được trên tổng số ca thực tế. (Quan trọng để không bỏ lọt tội phạm).

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

- **AUPRC (Area Under Precision-Recall Curve):** Diện tích dưới đường cong PR. Đây là chỉ số quan trọng nhất để đánh giá mô hình trên dữ liệu mất cân bằng.

1.8 Sơ đồ khối quy trình thực hiện



Hình 1: Sơ đồ khối quy trình xử lý và phân tích dữ liệu

## 2 TIỀN XỬ LÝ DỮ LIỆU

### 2.1 Đọc dữ liệu

Đọc dữ liệu từ file "creditcard.csv" vào R

```
1 #import dataset
2 ds.raw <- read.csv("creditcard.csv")
3 summary_table_raw <- skim(ds.raw)
```

	skim_type	skim_variable	n_missing	complete_rate	numeric.mean	numeric.sd	numeric.p0	numeric.p25	numeric.p50	numeric.p75	numeric.p100	numeric.hist
1	numeric	Time	0	1	94813.85958	47488.146	0	54201.5	84692	139320.5	172792	
3	numeric	V1	0	1	1.16658E-15	1.9586958	-56.4075096	-0.920373384	0.018108799	1.315641694	2.454929991	
4	numeric	V2	0	1	3.11899E-16	1.65130858	-72.7157276	-0.598549913	0.065485556	0.803723871	22.05772899	
5	numeric	V3	0	1	-1.36345E-15	1.51625501	-48.3255894	-0.890364838	0.179846344	1.027195542	9.382558433	
6	numeric	V4	0	1	2.11133E-15	1.41586857	-5.6831712	-0.848640116	-0.019846529	0.743341289	16.87534403	
7	numeric	V5	0	1	9.79699E-16	1.38024673	-113.743307	-0.691597071	-0.054335827	0.61192644	34.80166588	
8	numeric	V6	0	1	1.51004E-15	1.33227109	-26.1605059	-0.768295608	-0.274187077	0.398564896	73.30162555	
9	numeric	V7	0	1	-5.42187E-16	1.2370936	-43.5572416	-0.554075879	0.040103083	0.570436073	120.5894939	
10	numeric	V8	0	1	1.02674E-16	1.1943529	-73.2167185	-0.208629744	0.022358036	0.327345862	20.00720837	
11	numeric	V9	0	1	-2.42246E-15	1.09863209	-13.4340663	-0.64309757	-0.051428732	0.59713903	15.59499461	
12	numeric	V10	0	1	2.22607E-15	1.08884977	-24.5882624	-0.535425726	-0.092917384	0.453923445	23.74513612	
13	numeric	V11	0	1	1.70641E-15	1.02071303	-4.79747346	-0.762494196	-0.032757354	0.739593407	12.01891318	
14	numeric	V12	0	1	-1.24305E-15	0.99920139	-18.6837146	-0.405571485	0.140032588	0.618238033	7.848392076	
15	numeric	V13	0	1	8.35309E-16	0.99527423	-5.79188121	-0.648539299	-0.013568057	0.662504959	7.126882959	
16	numeric	V14	0	1	1.22866E-15	0.95859561	-19.2143255	-0.425574012	0.050601319	0.493149849	10.52676605	
17	numeric	V15	0	1	4.84393E-15	0.91531601	-4.49894468	-0.582884279	0.04807155	0.648820806	8.877741598	
18	numeric	V16	0	1	1.43367E-15	0.87625289	-14.1298545	-0.468036767	0.066413321	0.523296312	17.31511152	
19	numeric	V17	0	1	-3.77728E-16	0.84933706	-25.1627994	-0.483748314	-0.065675754	0.399674983	9.25352625	
20	numeric	V18	0	1	9.76066E-16	0.83817621	-9.49874592	-0.498849799	-0.003636312	0.500806747	5.041069185	
21	numeric	V19	0	1	1.03848E-15	0.8140405	-7.21352743	-0.456298919	0.003734823	0.458949356	5.591971427	
22	numeric	V20	0	1	6.40667E-16	0.77092502	-54.4977205	-0.211721365	-0.062481092	0.133040841	39.42090425	
23	numeric	V21	0	1	1.68618E-16	0.73452401	-34.8303821	-0.228394947	-0.029450168	0.186377203	27.20283916	
24	numeric	V22	0	1	-3.38062E-16	0.72570156	-10.9331437	-0.542350373	0.006781943	0.528553635	10.50309009	
25	numeric	V23	0	1	2.67347E-16	0.6244603	-44.8077352	-0.161846345	-0.01119293	0.147642064	22.52841169	
26	numeric	V24	0	1	4.47487E-15	0.60564707	-2.83662692	-0.354586136	0.040976056	0.4395266	4.584549137	
27	numeric	V25	0	1	5.10572E-16	0.52127807	-10.2953971	-0.317145054	0.016593502	0.350715563	7.519588679	
28	numeric	V26	0	1	1.68463E-15	0.48222701	-2.60455055	-0.326983926	-0.052139108	0.240952174	3.517345612	
29	numeric	V27	0	1	-3.67334E-16	0.40363249	-22.5656793	-0.070839529	0.001342146	0.09104512	31.61219811	
30	numeric	V28	0	1	-1.23041E-16	0.33008326	-15.4300839	-0.052959793	0.011243832	0.078279955	33.84780782	
31	numeric	Amount	0	1	88.34961925	250.120109	0	5.6	22	77.165	25691.16	
32	numeric	Class	0	1	0.001727486	0.04152719	0	0	0	0	1	

Hình 2: Bảng tóm tắt dataset thô

**Nhận xét:** Quan sát thấy bộ dữ liệu gốc bao gồm các biến thuộc biến numeric. Trong đó có biến Amount, Time là biến liên tục và Class là biến nhị phân.

### 2.2 Làm sạch dữ liệu

Tiến hành kiểm tra và loại bỏ NA và quan sát trùng lặp:

```
1 #clean dataset - check NA,NAN and duplicate
2 dup_count <- sum(duplicated(ds.raw))
3 na_count <- sum(is.na(ds.raw)) #na comprise NA & NAN
4 ds.clean <- ds.raw %>%
5   drop_na() %>%
6   distinct()
```

Loại dữ liệu	Số lượng
Duplicate	1081
NA/NaN	0

Bảng 1: Số lượng quan sát trùng lặp và giá trị thiếu

**Nhận xét:** Do bộ dữ liệu đều là numeric nên việc ta cần làm là tìm:

- **NAs:** Ô trống.
- **NANs:** Không phải số.
- **Duplicates:** Trùng lặp dữ liệu.

	skim_type	skim_variable	n_missing	complete_rate	numeric.mean	numeric.sd	numeric.p0	numeric.p25	numeric.p50	numeric.p75	numeric.p100	numeric.hist
1	numeric	Time	0	1	94811.0776	47481.04789	0	54204.75	84692.5	139298	172792	
3	numeric	V1	0	1	0.00591715	1.948026142	-56.40750963	-0.915951331	0.020384055	1.316067754	2.454929991	
4	numeric	V2	0	1	-0.004134756	1.646702964	-72.71572756	-0.600320566	0.063949172	0.800282893	22.05772899	
5	numeric	V3	0	1	0.001613119	1.508681916	-48.32558936	-0.889682005	0.179962728	1.026959974	9.382558433	
6	numeric	V4	0	1	-0.002966308	1.414184014	-5.683171198	-0.850134466	-0.022248017	0.739646964	16.87534403	
7	numeric	V5	0	1	0.00182756	1.377008279	-113.7433067	-0.68982973	-0.053467607	0.612217963	34.80166588	
8	numeric	V6	0	1	-0.001139488	1.331930592	-26.16050594	-0.769030769	-0.275167638	0.39679214	73.30162555	
9	numeric	V7	0	1	0.001800692	1.227663895	-43.55724157	-0.552509442	0.040859471	0.570473868	120.5894939	
10	numeric	V8	0	1	-0.000854453	1.179054428	-73.21671846	-0.208828472	0.021897947	0.325703684	20.00720837	
11	numeric	V9	0	1	-0.0015962	1.095492481	-13.43406632	-0.644220769	-0.052595651	0.595976859	15.59499461	
12	numeric	V10	0	1	-0.00144071	1.07640735	-24.58826244	-0.535578109	-0.093236526	0.453618728	23.74513612	
13	numeric	V11	0	1	0.000201758	1.018720153	-4.797473465	-0.761649296	-0.03230594	0.739579192	12.01891318	
14	numeric	V12	0	1	-0.000714788	0.994674445	-18.68371463	-0.406197733	0.13907201	0.616976489	7.848392076	
15	numeric	V13	0	1	0.000603376	0.995429637	-5.791881206	-0.647862079	-0.01292699	0.663178314	7.126882959	
16	numeric	V14	0	1	0.000252317	0.95221509	-19.21432549	-0.425732466	0.050208678	0.492335541	10.52676605	
17	numeric	V15	0	1	0.001042838	0.914893633	-4.498944677	-0.581451845	0.049298852	0.650104116	8.877741598	
18	numeric	V16	0	1	0.001162013	0.873696328	-14.12985452	-0.466859616	0.067119138	0.523511581	17.31511152	
19	numeric	V17	0	1	0.000170161	0.842507321	-25.16279937	-0.48392758	-0.065866933	0.398971712	9.25352625	
20	numeric	V18	0	1	0.001515166	0.83737753	-9.498745921	-0.498014299	-0.002141756	0.501955665	5.041069185	
21	numeric	V19	0	1	-0.000264264	0.813378553	-7.21352743	-0.456288916	0.003367083	0.458507934	5.591971427	
22	numeric	V20	0	1	0.000187175	0.769984241	-54.49772049	-0.211469275	-0.062353459	0.133207118	39.42090425	
23	numeric	V21	0	1	-0.000370593	0.723909367	-34.83038214	-0.228304909	-0.029440589	0.186193603	27.20283916	
24	numeric	V22	0	1	-1.50276E-05	0.724550465	-10.9331437	-0.542699706	0.006674895	0.528245198	10.50309009	
25	numeric	V23	0	1	0.000198171	0.623702379	-44.8077352	-0.161703451	-0.011158577	0.147748443	22.52841169	
26	numeric	V24	0	1	0.000214207	0.605626698	-2.836626919	-0.354453441	0.041015733	0.439738485	4.584549137	
27	numeric	V25	0	1	-0.000232387	0.521220317	-10.29539707	-0.317485371	0.016278362	0.350667366	7.519588679	
28	numeric	V26	0	1	0.000149441	0.482052941	-2.604550553	-0.326763401	-0.052171628	0.240261302	3.517345612	
29	numeric	V27	0	1	0.001763032	0.395743881	-22.56567932	-0.070640822	0.001478595	0.091208165	31.61219811	
30	numeric	V28	0	1	0.000547312	0.328026604	-15.43008391	-0.052818036	0.011287604	0.078276094	33.84780782	
31	numeric	Amount	0	1	88.47268731	250.3994371	0	5.6	22	77.51	25691.16	
32	numeric	Class	0	1	0.001667101	0.040796176	0	0	0	0	1	

Hình 3: Bảng tóm tắt dataset đã làm sạch

**Nhận xét:** Khi so sánh với tóm tắt dataset thô, ta quan sát thấy sự thay đổi cột giá trị trung bình (mean).

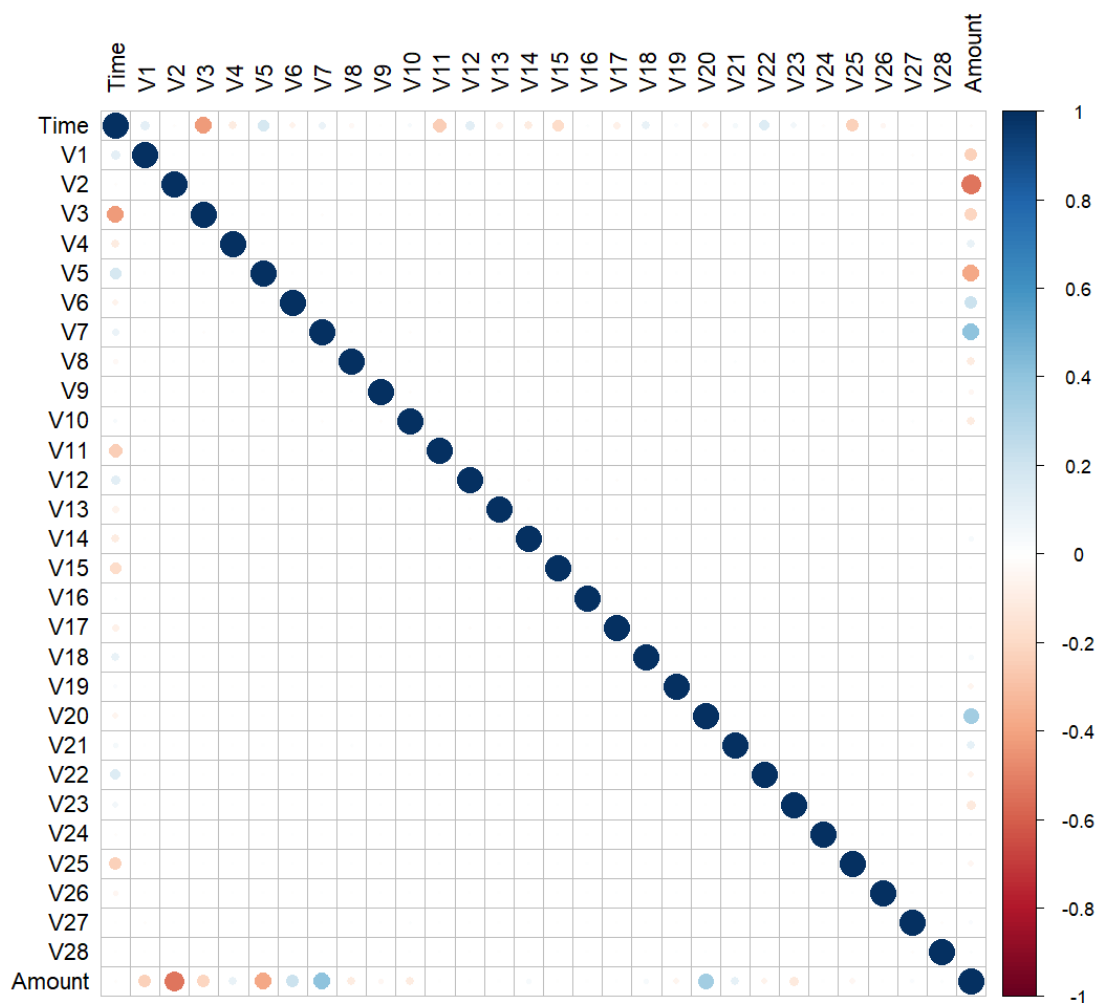
## 3 THỐNG KÊ MÔ TẢ

### 3.1 Ma Trận Tương Quan

```

1  cat(">> [1/9] Ve Correlation Matrix...")
2  correlations <- cor(ds.clean, method="pearson")
3  corrplot(correlations, number.cex = .6, method = "circle",
4  type = "full", tl.cex=0.5, tl.col = "black")
5
6  # CLASS
7  summary_class <- ds.clean %>%
8  count(Class) %>%
9  mutate(percentage = round(n / sum(n) * 100, 4))
10 cat(">> Summarise class ratio")
11 print(summary_class)

```



Hình 4: Ma trận tương quan giữa các biến

**Nhận Xét:** Ma trận tương quan bước đầu cho thấy mối tương quan giữa các biến

- Giữa các cặp biến PCA: quan sát thấy không có sự tương quan do đây là đặc trưng của bộ biến sau giảm chiều dữ liệu.
- Giữa mỗi thành phần PCA và Amount, Time: Ma trận có thể hiện sự tương quan dương (màu xanh) và tương quan âm (màu đỏ). Tuy nhiên việc tương quan này gần như không có ý nghĩa trong việc phân loại giao dịch gian lận, vì mô hình phân loại ta quan tâm tới biến class.

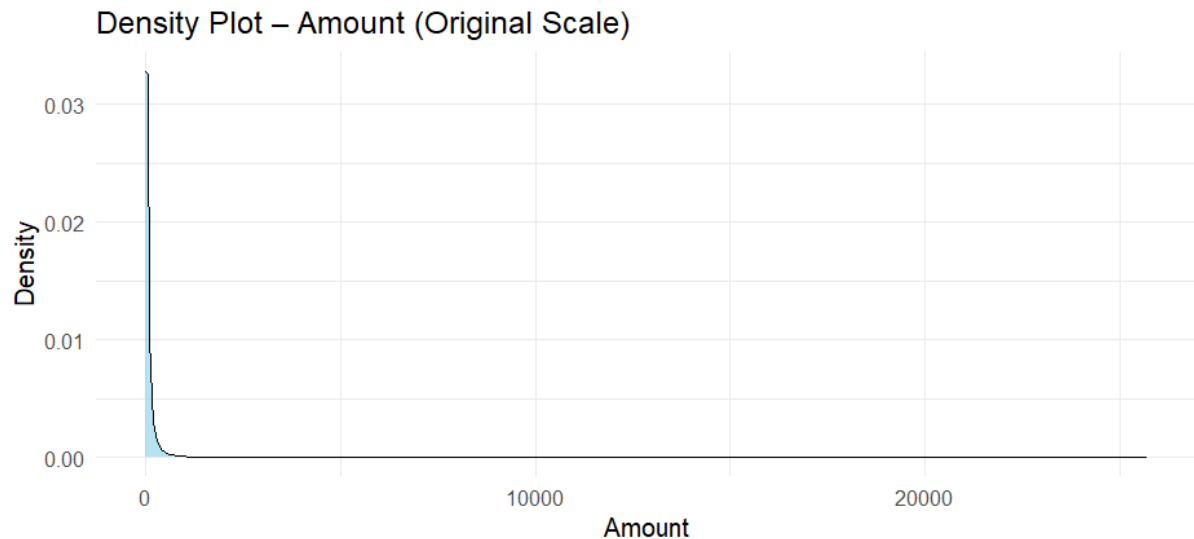
## 3.2 Biến Không Qua Biến Đổi PCA

### 3.2.1 Biến Amount

Vẽ biểu đồ mật độ của biến amount:

```
1 cat(">> [2/9] Ve Density Amount (Goc)...")
```

```
2 ggplot(ds.clean, aes(x = Amount)) +  
3 geom_density(fill = "skyblue", alpha = 0.6) +  
4 labs(title = "Density Plot - Amount (Original Scale)",  
5 x = "Amount", y = "Density") +  
6 theme_minimal()
```



Hình 5: Biểu đồ mật độ biến Amount

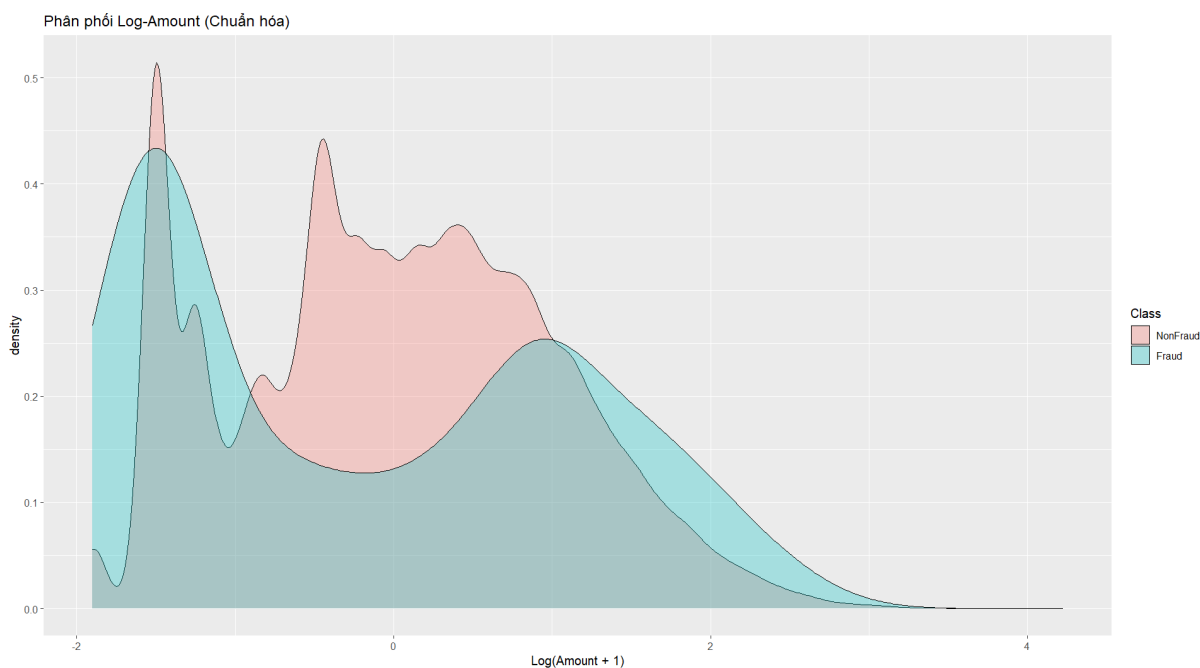
**Nhận Xét:** Biểu đồ cho thấy một **độ lệch dương cực kỳ nghiêm trọng** (Highly Positive Skew). Phần lớn mật độ tập trung sát trục Y, nghĩa là hầu hết các giao dịch có giá trị rất nhỏ (gần 0), trong khi các giao dịch lớn kéo dài đuôi đồ thị ra xa.

**Đề xuất:** Chuẩn hóa, biến đổi dữ liệu biến Amount bằng phương pháp Logarit biến đổi. Tuy nhiên ở đây sẽ sử dụng  $\log(\text{amount}+1)$  do dữ liệu Amount chứa giá trị 0.

Thực hiện chuẩn hóa Amount và khảo sát biểu đồ phân theo class:

```
1 ds.clean <- ds.clean %>%  
2 mutate(  
3   logAmount = log1p(Amount),  
4   logAmount_Scaled = as.numeric(scale(logAmount)),  
5   Class = factor(Class, levels = c(0, 1), labels = c("NonFraud", "Fraud"))  
6 )  
7  
8 cat(">> [4/9] Ve Phan phoi Log-Amount theo Class...")  
9 ggplot(ds.clean, aes(x = logAmount_Scaled, fill = Class)) +  
10 geom_density(alpha = 0.3) +  
11 scale_fill_manual(values = c("#F8766D", "#00BFC4")) +  
12 labs(title = "Phan phoi Log-Amount (Chuan hoa)",  
13 x = "Log(Amount + 1)")
```





Hình 6: Biểu đồ mật độ biến Amount phân theo class sau khi đã biến đổi Logarit

**Nhận Xét:** Nhìn về tổng thể, phân phối mật độ lớp fraud và nonfraud chồng chất lên nhau khá nhiều.

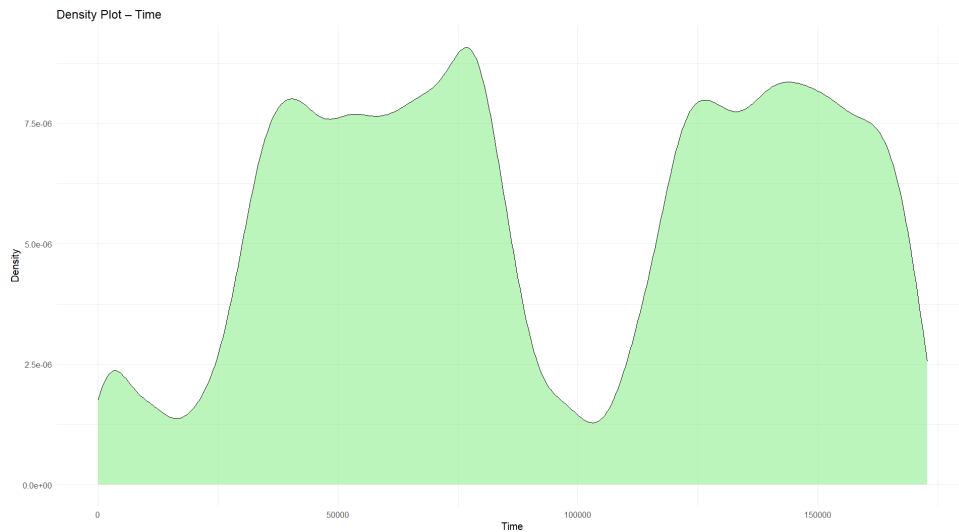
- Với  $\log(\text{amount}+1) \in [-1, 1]$  : Đồ thị thể hiện sự khác biệt khi nonfraud tập trung nhiều so với fraud.
- Đối với các khoảng còn lại, đồ thị không xuất hiện bất kỳ sự khác biệt rõ rệt nào về mật độ fraud và nonfraud.

Do đó, biến  $\log(\text{amount}+1)$  không thể là đặc trưng duy nhất để mô hình phát hiện gian lận hiệu quả.

### 3.2.2 Biến Time:

Vẽ biểu đồ mật độ của biến amount:

```
1 cat(">> [3/9] Ve Density Time (Chung)...")
2 ggplot(ds.clean, aes(x = Time)) +
3   geom_density(fill = "lightgreen", alpha = 0.6) +
4   labs(title = "Density Plot - Time",
5         x = "Time", y = "Density") +
6   theme_minimal()
```



Hình 7: Biểu đồ mật độ của biến Time

### Nhận Xét

- Biểu đồ thể hiện một phân bố **đa phương thức** (multimodal) với hai đỉnh chính rõ rệt, khoảng cách giữa hai đỉnh là khoảng 86400 giây (tương đương 24 giờ). Điều này cho thấy dữ liệu có thể được thu thập trong khoảng thời gian khoảng hai ngày.
- Hình dạng chung của biểu đồ thể hiện mô hình giao dịch theo thời gian trong ngày (ví dụ: hoạt động cao điểm vào ban ngày và giảm xuống vào ban đêm).

**Đề xuất:** Thực hiện thêm 2 biểu đồ, nhằm tìm hiểu sự khác biệt của biến theo class và liệu giao dịch gian lận có tập trung vào một khung giờ nhất định không:

- Biểu đồ mật độ Time phân theo class.
- Biểu đồ mật Time đã phân theo khung giờ theo class.

Chuyển biến Time và khảo sát biểu đồ phân theo class trong 2 ngày:

```

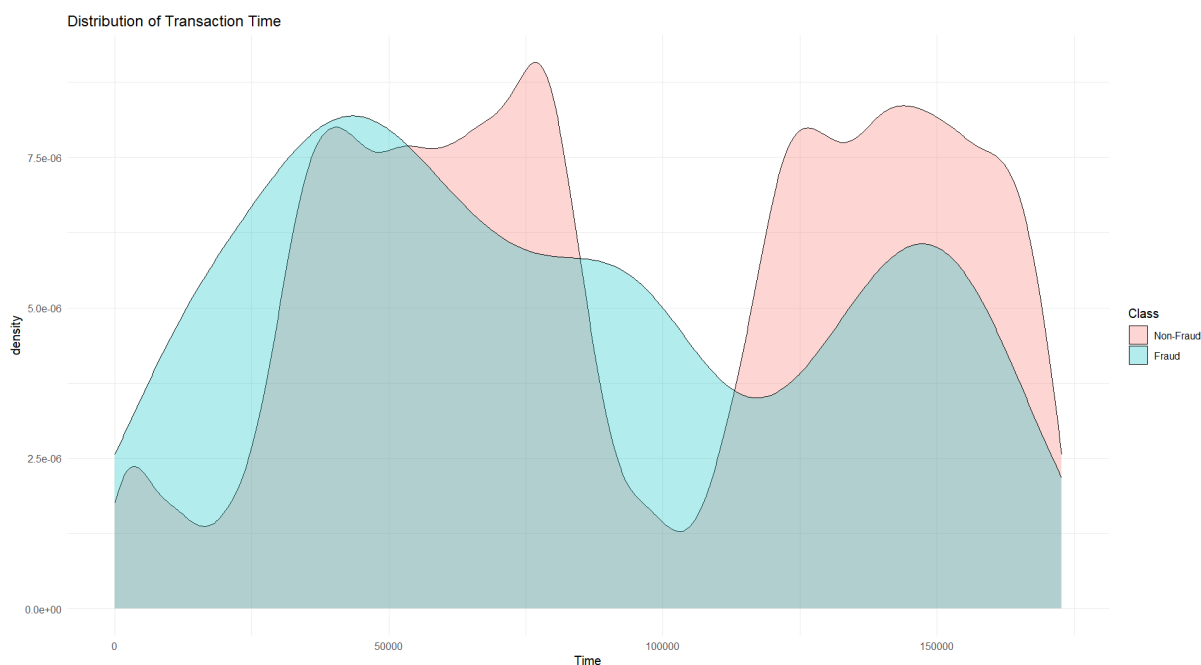
1 ds.clean <- ds.clean %>%
2 mutate(
3   Time_Z = as.numeric(scale(Time)),
4   Hour = floor(Time / 3600) %% 24,
5   Day = paste("Day", floor((Time / 3600) / 24) + 1),
6   Hour_Range = cut(Hour, breaks = 0:24, right = FALSE,
7   labels = paste0(0:23, "-", 1:24)),
8   Class = factor(Class, levels = c(0, 1),
9   labels = c("NonFraud", "Fraud"))
10 )
11
12 cat(">> [5/9] Ve Distribution of Transaction Time...")
13 ggplot(ds.clean, aes(x = Time, fill = Class)) +
14   geom_density(alpha = 0.3) +
15   scale_fill_manual(values = c("#F8766D", "#00BFC4"),
16   labels = c("Non-Fraud", "Fraud")) +
17   labs(title = "Distribution of Transaction Time",
18   x = "Time", y = "density") +
19   theme_minimal()
20
21 cat(">> [6/9] Ve So giao dich theo ngay, gio...")
22 ggplot(ds.clean, aes(x = Hour_Range, fill = Class)) +
23   geom_bar(alpha = 0.6, position = "identity") +
24   facet_wrap(Day ~ Class, scales = "free_y", ncol = 2) +

```

```

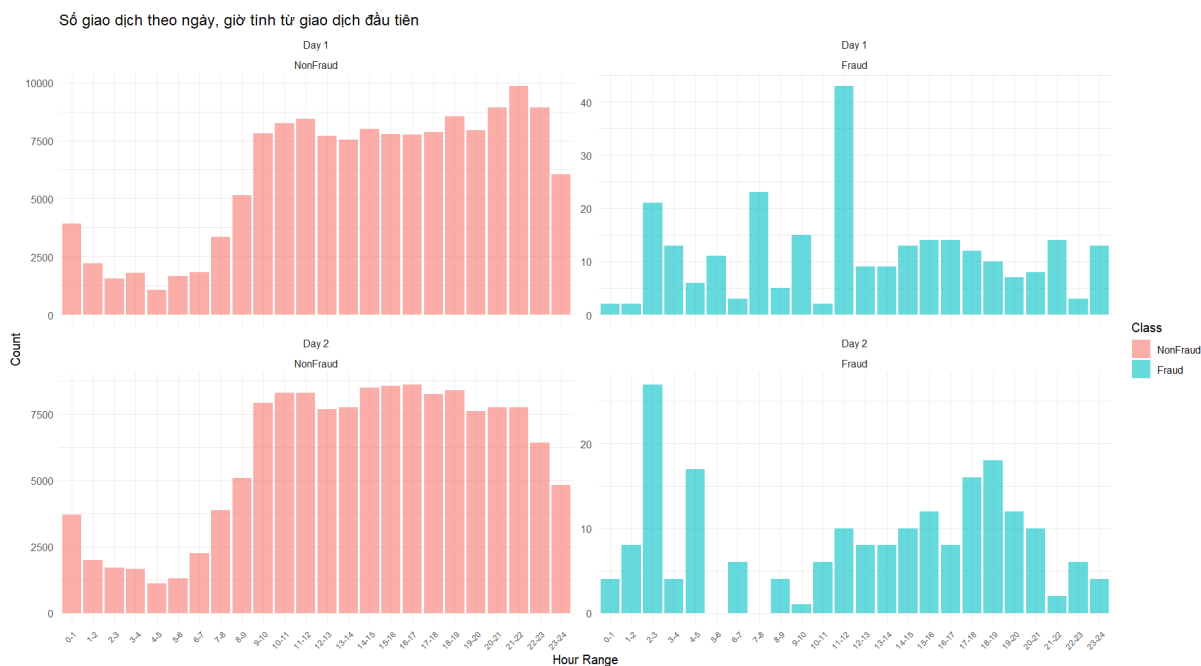
25 scale_fill_manual(values = c("#F8766D", "#00BFC4")) +
26 theme_minimal() +
27 theme(axis.text.x = element_text(angle = 45, hjust = 1,
28 size = 7)) +
29 labs(title = "Số giao dịch theo ngày, giờ",
30 x = "Hour Range", y = "Count", fill = "Class")

```



Hình 8: Mật độ biến Time phân theo class

**Nhận Xét:** Biểu đồ biến Time gốc theo class không mô tả được sự khác biệt rõ rệt nào về tỷ lệ giao dịch gian lận và gian lận.



Hình 9: Số lượng giao dịch theo ngày, giờ tính từ giao dịch đầu tiên

**Nhận Xét:** Mặc dù với số lượng nhỏ, tỷ lệ giao dịch fraud cao hơn tỷ lệ giao dịch nonfraud trong 5 giờ đầu tiên mỗi ngày (mốc tham chiếu từ lần giao dịch đầu tiên). Tuy nhiên với khác biệt nhỏ như vậy, biến Time cũng không thể là đặc trưng duy nhất để xây dựng mô hình phân loại hiệu quả mà cần phải kết hợp với những đặc trưng khác.

### 3.3 Biểu đồ 28 biến PCA

Chuẩn bị dữ liệu để trực quan hóa:

```
1 #select V features and Class, then gather them into a long format
2 correct_order <- paste0("V", 1:28)
3 ds.pca <- ds.scaled %>%
4 select(Class, V1:V28) %>%
5 gather(key = "Feature", value = "Value", -Class) %>%
6 mutate(Feature = factor(Feature, levels = correct_order))
```

Class	Feature	Value
0	V1	1.2
0	V2	0.5
1	V1	0.8
1	V2	1.0

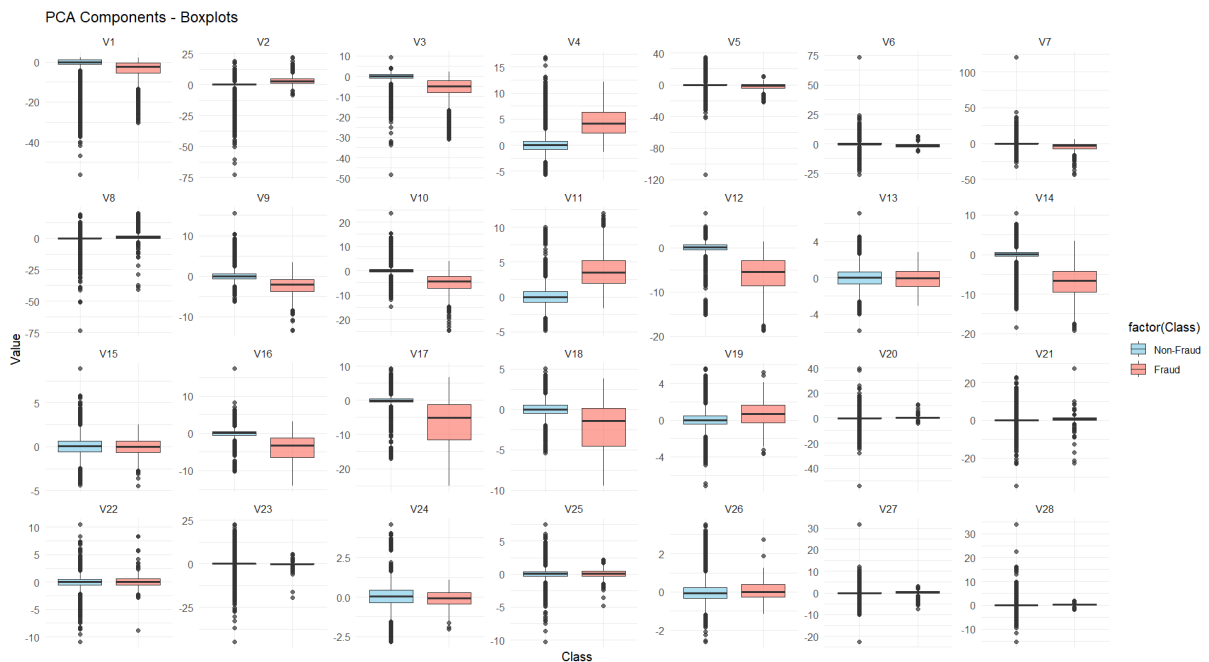
Bảng 2: Ví dụ minh họa sau khi gom biến

Việc chuyển các biến quan tâm từ dạng *wide* (dữ liệu gốc) sang dạng *long* nhằm tính toán thống kê và trực quan hóa cho từng biến PCA.

**Nhận xét:** Mỗi hàng khi này là một giá trị PCA của một biến cụ thể (V1:V28) của một quan sát.

#### Biểu đồ hộp (Boxplots):

Vẽ biểu đồ hộp thể hiện phân phối giá trị của một biến PCA (V1:V28) cho hai Class: **Non-Fraud (0)** và **Fraud (1)**.



Hình 10: Biểu đồ hộp phân phối 28 biến PCA theo Class

**Nhận xét:** Ta quan sát thấy bộ dữ liệu PCA còn nhiều điểm ngoại lai (outliers). Ngoài ra, giá trị trung bình (mean) và tứ phân vị (IQR) rất nhỏ, việc này làm cho hộp co nhỏ gần như không thấy hình dạng.

**Đề xuất:** Sử dụng phương pháp Boxplot Zoomed View, loại bỏ ngoại lai ngoài vùng quan sát.

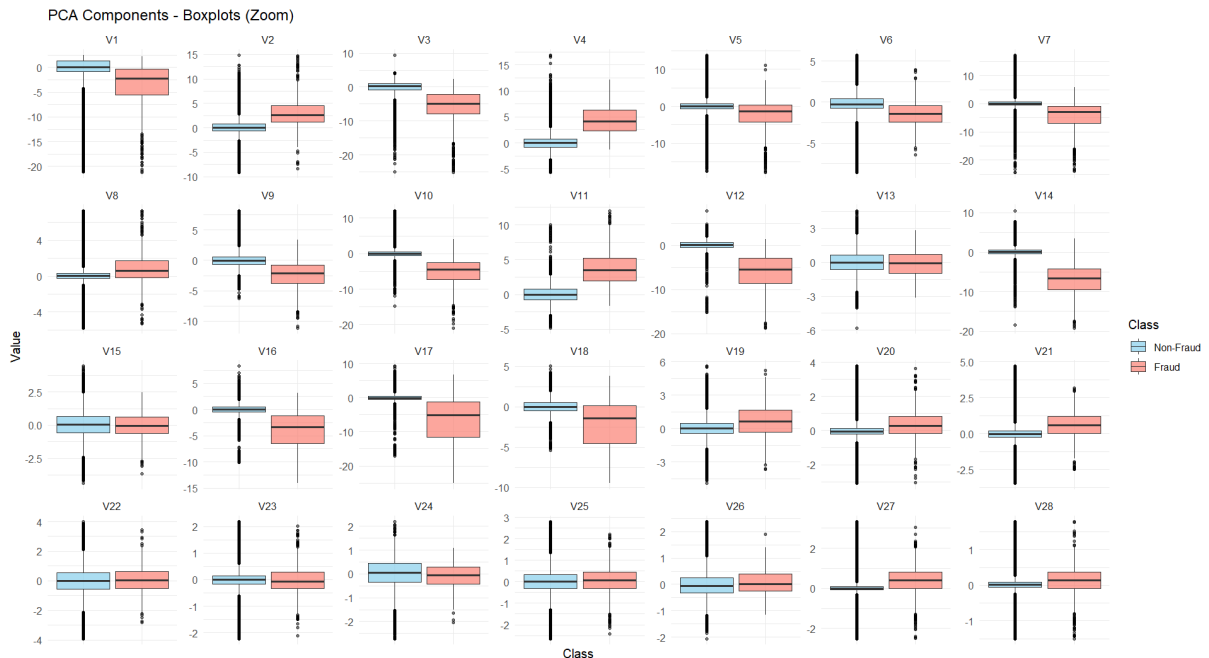
```

1  #Boxplots zoom
2  # Compute the standard boxplot statistics
3  stats_data <- ds.pca %>%
4  group_by(Feature, Class) %>%
5  summarise(
6    y25 = quantile(Value, 0.25),
7    y50 = median(Value),
8    y75 = quantile(Value, 0.75),
9    IQR = y75 - y25,
10   lower_whisker = max(min(Value), y25 - 1.5 * IQR),
11   upper_whisker = min(max(Value), y75 + 1.5 * IQR),
12   .groups = 'drop')
13
14 # The outliers out of zoomed view would be removed
15 zoom_limits <- stats_data %>%
16 group_by(Feature) %>%
17 summarise(
18   view_min = min(y25 - 3 * IQR),
19   view_max = max(y75 + 3 * IQR))
20
21 #Just points in zoomed view are visible
22 visible_points <- ds.pca %>%
23 inner_join(stats_data, by = c("Feature", "Class")) %>%
24 inner_join(zoom_limits, by = "Feature") %>%
25 filter(
26   # Condition 1: Must be outliers
27   (Value < lower_whisker | Value > upper_whisker) &
28   # Condition 2: Must be in the zoomed view
29   (Value >= view_min & Value <= view_max))

```

Các bước thuật toán bao gồm:

- Tính lại thông số Boxplot.
- Xác định "vùng nhìn".
- Thực hiện zoom.



Hình 11: Biểu đồ hộp phân phối 28 biến PCA đã zoom

**Nhận xét:** Phạm vi phân phối (IQR) và trung bình (mean) có sự khác nhau rõ rệt, biểu thị có khả năng phân tách gian lận ở các biến V2, V3, V4, V7, V10, V12, V14, V16 và V17.

### 3.4 Phân tích thành phần ngoại lai

Dựa vào các biểu đồ trực quan của các biến gốc của dữ liệu, nhóm nhận thấy cần tìm hiểu một biến khác mạnh hơn để phục vụ mô hình phân loại.

Vì vậy nhóm thực hiện khảo sát số thành phần ngoại lai của các quan sát trong dữ liệu:

```

1  # CHECK OUTLIERS
2  # 1. Khởi tạo cột đếm
3  ds.outlier.counts <- ds.clean %>%
4  mutate(num_outliers = 0)
5
6  # 2. Chạy vòng lặp theo từng biến V1-V28
7  for (col in paste0("V", 1:28)) {
8    Q1 <- quantile(ds.outlier.counts[[col]], 0.25, na.rm = TRUE)
9    Q3 <- quantile(ds.outlier.counts[[col]], 0.75, na.rm = TRUE)
10   IQR_val <- Q3 - Q1
11   lower_bound <- Q1 - 1.5 * IQR_val
12   upper_bound <- Q3 + 1.5 * IQR_val
13
14   ds.outlier.counts <- ds.outlier.counts %>%
15   mutate(num_outliers = num_outliers + (!sym(col) < lower_bound | !sym(col) > upper_
16   bound))
17 }

```

```
18 # 3. Tính tổng số dòng bị ảnh hưởng
19 total_rows <- nrow(ds.clean)
20 rows_with_outliers <- sum(ds.outlier.counts$num_outliers > 0)
21 pct_affected <- (rows_with_outliers / total_rows) * 100
22
23 cat(sprintf("\n=== KẾT QUẢ PHÂN TÍCH THEO HÀNG (ROW-WISE) ===\n"))
24 cat(sprintf("Tổng số giao dịch: %d\n", total_rows))
25 cat(sprintf("Số giao dịch có ít nhất một biến ngoại lai: %d (Chiếm %.2f%%)\n", rows_
    with_outliers, pct_affected))
26
27
28 # 4. Xem chi tiết phân phối (từ 0 đến 28)
29 cat("\n--- Phân phối số lượng outliers trên mỗi giao dịch (Full Range 0-28) ---\n")
30
31 full_sequence <- tibble(num_outliers = 0:28)
32 actual_counts <- ds.outlier.counts %>% count(num_outliers)
33
34 table_outliers <- full_sequence %>%
35 left_join(actual_counts, by = "num_outliers") %>%
36 mutate(n = coalesce(n, 0)) %>% # Thay NA bằng 0
37 mutate(percentage = round(n / sum(n) * 100, 8))
38 print(table_outliers, n = 29)
```

### Các bước thuật toán bao gồm:

- Khởi tạo cột đếm ngoại lai (outliers) cho mỗi quan sát sau đó chạy vòng lặp đếm số biến nhận giá trị outliers.
- Thống kê và in ra màn hình số dòng bị ảnh hưởng ( $\geq 1$  outliers).
- Trả về bảng thống kê số giá trị ngoại lai (outliers).

Bảng 3: Kết quả phân tích Outliers

Thông tin	Giá trị
Tổng số giao dịch	283,726
Số giao dịch có ít nhất 1 biến ngoại lai	128,084 (45.14%)

### Về phân phối số giá trị ngoại lai:

Bảng 4: Phân phối số lượng Outlier trên mỗi giao dịch (0–28)

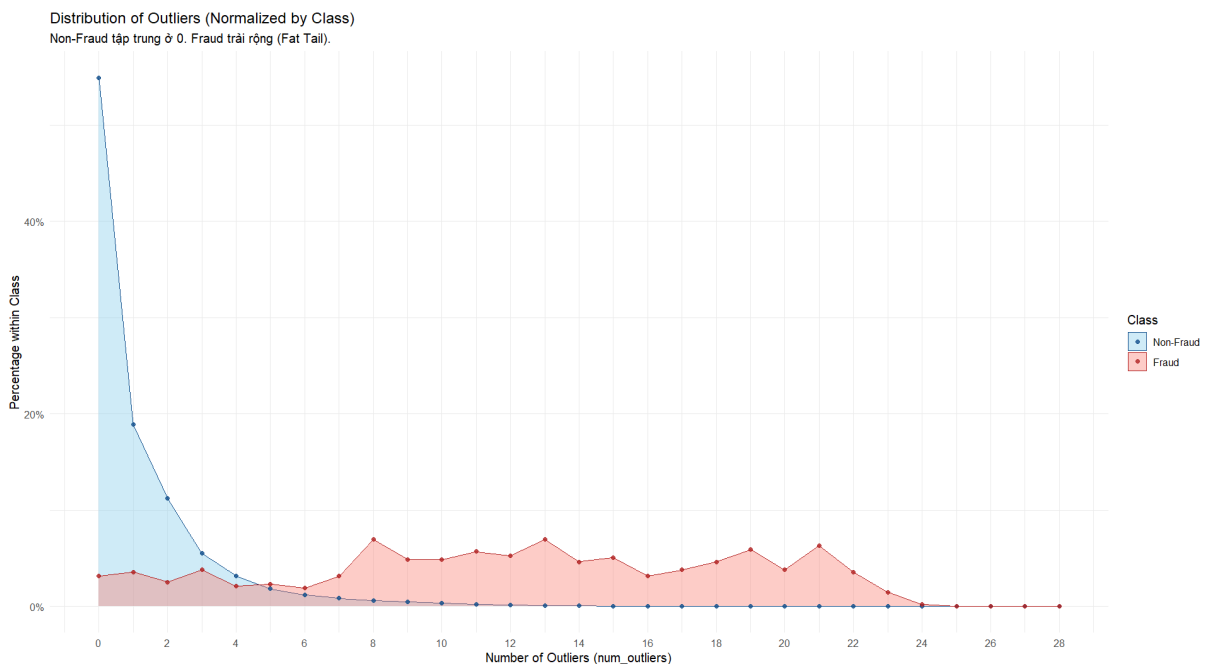
num_outliers	n	%	num_outliers	n	%
0	155642	54.90	15	187	0.0659
1	53684	18.90	16	95	0.0335
2	31798	11.20	17	79	0.0278
3	15621	5.51	18	66	0.0233
4	9004	3.17	19	61	0.0215
5	5247	1.85	20	51	0.0180
6	3483	1.23	21	49	0.0173
7	2429	0.856	22	24	0.00846
8	1867	0.658	23	14	0.00493
9	1441	0.508	24	4	0.00141
10	990	0.349	25	1	0.000352
11	724	0.255	26	0	0
12	525	0.185	27	1	0.000352
13	398	0.140	28	0	0
14	241	0.0849			

## Biểu đồ phân phối của số ngoại lai phân theo class

```

1  # --- PLOT 9: Distribution of outliers (density_outliers.png) ---
2  cat(">> [9/10] Density Plot de so sanh cau truc phan phoi Fraud...\n")
3  plot_data_prop <- ds.clean %>%
4  count(Class, num_outliers) %>%
5  complete(num_outliers = 0:28, Class, fill = list(n = 0)) %>%
6  group_by(Class) %>%
7  mutate(pct = n / sum(n)) %>%
8  ungroup()
9
10 ggplot(plot_data_prop, aes(x = num_outliers, y = pct, fill = Class, color = Class)) +
11 geom_area(alpha = 0.4, position = "identity") +
12 geom_point(size = 1.5, alpha = 0.8) +
13 scale_fill_manual(values = c("skyblue", "salmon"), labels = c("Non-Fraud", "Fraud")) +
14 scale_color_manual(values = c("dodgerblue4", "firebrick"), labels = c("Non-Fraud", "
    Fraud")) +
15 scale_x_continuous(breaks = seq(0, 28, 2)) +
16 scale_y_continuous(labels = scales::percent) +
17 labs(title = "Distribution of Outliers (Normalized by Class)",
18 subtitle = "Non-Fraud tập trung ở 0. Fraud trải rộng (Fat Tail).",
19 x = "Number of Outliers (num_outliers)",
20 y = "Percentage within Class") +
21 theme_minimal()
22

```



Hình 12: Phân phối số lượng Outlier theo từng Class

**Nhận xét:** Tuyệt đại đa số các giao dịch nonfraud có 0 Outlier, trong khi đó phân phối của fraud trải rộng trên toàn miền. Hay nói cách khác nếu số ngoại lai càng lớn khả năng phân loại được đó là giao dịch gian lận càng cao.

Vì vậy, num\_outliers là một biến tiềm năng để xây dựng mô hình phát hiện gian lận tối ưu.



## 4 THỐNG KÊ SUY DIỄN

### 4.1 Kiểm định giả thuyết về đặc trưng số tiền giao dịch Amount

**Mục đích:** Kiểm định xem hành vi giao dịch của nhóm gian lận có khác biệt so với nhóm không gian lận hay không, thông qua việc khảo sát giá trị trung bình của `logAmount_Scaled` giữa 2 nhóm.

**Đặt ra giả thuyết:**

- **H0 (Giả thuyết không):** Giá trung bình của `logAmount_Scaled` giữa 2 nhóm bằng nhau.
- **H1 (Giả thuyết đối):** Giá trung bình của `logAmount_Scaled` giữa 2 nhóm không bằng nhau, có sự khác biệt.

Phương pháp kiểm định được sử dụng ở đây là Mann-Whitney U-test, do `logAmount_Scaled` không tuân theo phân phối chuẩn.

```

1 # GT1: U-test Amount
2 cat(">> Do logAmount_Scaled không tuân theo pp chuẩn -> Sử dụng MANN-WHITNEY U TEST (Non-
   parametric)\n")
3 gt1.test_res <- wilcox.test(logAmount_Scaled ~ Class, data = ds.clean, exact=FALSE)
4
5 cat(sprintf(">> Kết quả P-value: %.4e\n", gt1.test_res$p.value))
6 if(gt1.test_res$p.value < 0.05) {
7   cat(">> Kết luận: BÁC BỎ H0.\n")
8   cat("   (Có sự khác biệt có ý nghĩa thống kê về Amount giữa 2 nhóm,\n")
9   cat("   tuy nhiên cần kết hợp biểu đồ để xem mức độ tách biệt thực tế.)\n")
10 } else {
11   cat(">> Kết luận: CHẤP NHẬN H0.\n")
12   cat("   (Không tìm thấy bằng chứng về sự khác biệt giữa 2 nhóm.)\n")
13 }

```

P-value	Kết luận
$2.6861 \times 10^{-5}$	BÁC BỎ H0 (Có sự khác biệt)

Bảng 5: Kết quả GT1 (Mann-Whitney U-test `logAmount_Scaled`)

Kết quả cho thấy  $P\text{-value} < 0.05$ , dẫn đến bác bỏ giả thuyết  $H_0$ , tức là có sự khác biệt có ý nghĩa thống kê về mặt trung vị giữa hai nhóm Fraud và NonFraud. Tuy nhiên, do kích thước mẫu lớn ( $N > 280,000$ ), P-value thường trở nên quá nhạy cảm, chỉ cần một chút thay đổi nhỏ cũng sẽ khiến giá trị này bị kéo xuống rất thấp. Do đó, ta cần kết hợp với biểu đồ quan sát trực quan hoặc các chỉ số cấp cao hơn để đánh giá. Ở đây, khả năng phân loại của biến `longAmount_Scaled` sẽ được rút ra sau quá trình train model ở phần sau.

### 4.2 Kiểm định giả thuyết về khả năng phân loại của bộ biến PCA (V1:V28)

**Mục đích:** Kiểm định xem các biến  $V_i$  có khả năng phân loại dữ liệu hay không, tức là phân phối của 2 Class Fraud và NonFraud trong từng biến có thực sự đủ khác biệt.

**Đặt ra giả thuyết:**

- **H0 (Giả thuyết không):** Phân phối xác suất của biến  $V_i$  đối với nhóm giao dịch bình thường (Non-Fraud) và nhóm gian lận (Fraud) là GIỐNG NHAU. (Biến này không giúp ích cho việc phân loại)
- **H1 (Giả thuyết đối):** Phân phối xác suất của biến  $V_i$  đối với nhóm giao dịch bình thường (Non-Fraud) và nhóm gian lận (Fraud) là KHÁC NHAU. (Biến này giúp ích cho việc phân loại)

Phương pháp kiểm định được sử dụng ở đây là Kolmogorov-Smirnov Test (KS-Test), nhằm làm rõ việc: liệu rằng hình dáng phân phối của hai nhóm dữ liệu có giống nhau hay không. Kết quả chỉ số `KS_stat` càng gần 1 thì hình dáng đồ thị giữa hai Class càng tách biệt nhau, gần 0 thì ngược lại.

```
1 # GT2: KS-test V1:V28
2 gt2.var_ranking <- data.frame(Variable = character(), KS_Stat = numeric(), P_Value =
   numeric(), stringsAsFactors = FALSE)
3
4 for(v in paste0("V", 1:28)){
5   gt2.res <- ks.test(ds.clean[[v]][ds.clean$Class=="Fraud"],
6     ds.clean[[v]][ds.clean$Class=="NonFraud"])
7   gt2.var_ranking <- rbind(gt2.var_ranking,
8     list(Variable = v,
9       KS_Stat = gt2.res$statistic,
10      P_Value = gt2.res$p.value)) }
11
12 gt2.var_ranking <- gt2.var_ranking %>% arrange(desc(KS_Stat))
13 # In ra các biến tốt và tệ nhất
14 cat(">> Tóm tắt kết quả KS-Test: Top 10 biến tốt nhất:\n")
15 head(gt2.var_ranking, 10)
16 cat(">> Tóm tắt kết quả KS-Test: Top 5 biến tệ nhất:\n")
17 tail(gt2.var_ranking, 5)
18
19 good_vars_list <- gt2.var_ranking %>%
20   filter(KS_Stat > 0.1) %>%
21   pull(Variable)
22
23 cat(sprintf(">> Số lượng biến V được giữ lại: %d biến (trên tổng số 28)\n", length(good_
   vars_list)))
24 cat(">> Các biến bị loại bỏ:", setdiff(paste0("V", 1:28), good_vars_list), "\n")
25 print(good_vars_list)
```

Các bước thuật toán bao gồm:

- Tạo bảng rỗng `gt2.var_ranking` nhằm lưu trữ dữ liệu
- Chạy vòng lặp kiểm định cho từng biến, sau đó lưu giá trị `KS_Stat` và `P_Value`
- Xếp hạng, in kết quả sơ bộ và thực hiện sàng lọc

Bảng 6: Kết quả GT2 (KS-Test V1:V28): Top 10 biến có hiệu quả phân loại cao nhất

Hạng	Biến	KS_Stat	P_Value
1	V14	0.8377276	$2.87 \times 10^{-288}$
2	V10	0.7970328	$5.54 \times 10^{-261}$
3	V12	0.7766406	$8.05 \times 10^{-248}$
4	V4	0.7607318	$8.66 \times 10^{-238}$
5	V11	0.7473804	$1.57 \times 10^{-229}$
6	V17	0.7355319	$2.53 \times 10^{-222}$
7	V3	0.6951762	$1.21 \times 10^{-198}$
8	V16	0.6782065	$4.40 \times 10^{-189}$
9	V7	0.6516486	$1.35 \times 10^{-174}$
10	V2	0.6240736	$3.61 \times 10^{-160}$

Bảng 7: Kết quả GT2 (KS-Test V1:V28): Top 5 biến có hiệu quả phân loại thấp nhất

Hạng	Biến	KS_Stat	P_Value
24	V13	0.101540	$1.18 \times 10^{-4}$
25	V26	0.092689	$5.99 \times 10^{-4}$
26	V25	0.090713	$8.43 \times 10^{-4}$
27	V22	0.056059	$1.03 \times 10^{-1}$
28	V15	0.052107	$1.54 \times 10^{-1}$

**Kết luận:** Dùng mốc kiểm định  $KS\_Stat > 0.1$ , ta không có đủ bằng chứng thống kê để bác bỏ  $H_0$  đối với các biến V15, V22, V25, V26. Nghĩa là phân phối của 4 biến này giữa nhóm Fraud và NonFraud hoàn toàn giống nhau. Điều này có nghĩa, các biến này được đánh dấu là có khả năng phân loại kém.

Tuy nhiên, các model **Random Forest** và **XGBoost** được chọn để train là các model sử dụng thuật toán dạng cây, chúng tự động xếp hạng phân loại, và tìm các đặc điểm tốt nhất của biến đó cho việc phân loại. Do đó, ở phần dưới, nhóm vẫn sẽ tiếp tục sử dụng tập dataset có chứa tất cả các biến  $V_i$  này cùng với `logAmount_Scaled`, `num_outliers` và `Class`. Về model **Logistic Regression**, tác động của việc lược bỏ hay không các biến gây nhiễu này lên hiệu suất mô hình sẽ được xem xét tại phần Mở rộng.

## 5 THỐNG KÊ PHÂN LOẠI

### 5.1 Chuẩn bị số liệu cho huấn luyện

```
1 #prepare ds.model
2 ds.model <- ds.clean %>%
3 mutate(
4   logAmount_Scaled = as.numeric(scale(log1p(Amount)))
5 ) %>%
6 select(-Time, -Amount)
```

**Giải thích:**

- Dataset là data đã làm sạch trước đó. Ngoài ra, thêm cột Amount đã chuẩn hóa dạng logarit.
- Loại bỏ cột Amount do đã được thay thế bởi Amount chuẩn hóa.
- Loại bỏ cột Time vì không mang ý nghĩa phân loại.

```
1 ds.model$Class <- as.factor(ds.model$Class)
2 levels(ds.model$Class) <- c("NonFraud", "Fraud")
3 ds.model$Class <- relevel(ds.model$Class, ref = "NonFraud")
```

**Giải thích:**

- Chuyển kiểu dữ liệu biến Class sang factor, R coi đây là 2 cấp độ khác nhau "Fraud" và "NonFraud".
- Thiết lập factor tham chiếu. Ở đây chọn tham chiếu là "NonFraud".

```
1 set.seed(123)
2 idx <- sample.split(Y = ds.model$Class, SplitRatio = 0.7)
3 ds.train <- ds.model[idx, ]
4 ds.test <- ds.model[!idx, ]
```

**Giải thích:**

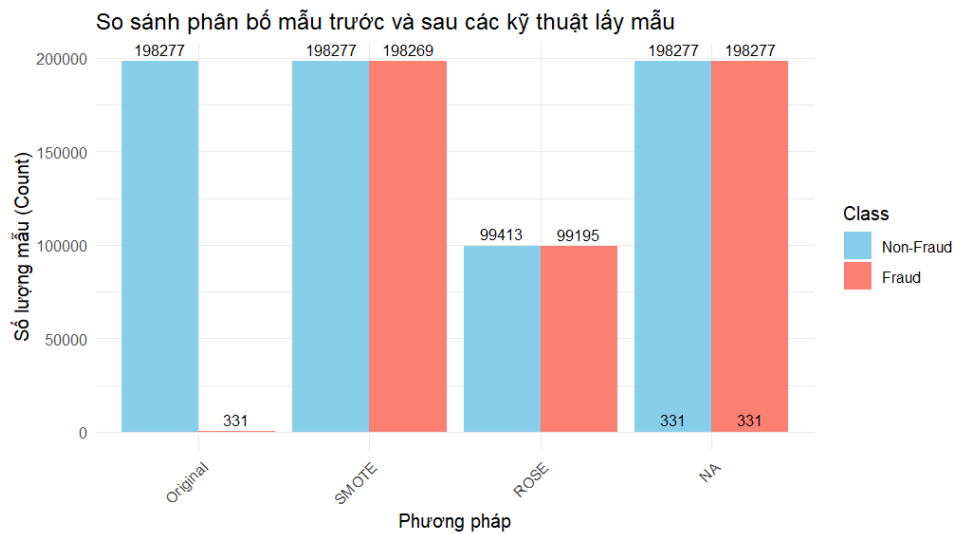
- Tạo setseed nhằm tái tạo lại cùng một kết quả phân chia data.
- Tách dataset theo tỷ lệ train:test=7:3.

## 5.2 Xử lý mất cân bằng dữ liệu

Dựa trên kết quả phân tích thống kê mô tả ở phần trước, nhóm em nhận thấy bộ dữ liệu có sự mất cân bằng nghiêm trọng: lớp giao dịch gian lận (*Fraud*) chỉ chiếm khoảng **0.1667%** tổng số quan sát. Sự chênh lệch này sẽ khiến các mô hình học máy có xu hướng thiên vị nhóm đa số và dự báo kém chính xác trên nhóm gian lận.

Để khắc phục, trước khi tiến hành huấn luyện mô hình, nhóm em thực hiện chia tập dữ liệu đã làm sạch (*ds.clean*) thành hai phần: tập huấn luyện (*Training set*) chiếm 70% và tập kiểm thử (*Test set*) chiếm 30%. Các kỹ thuật lấy mẫu lại (*Resampling*) sẽ chỉ được áp dụng trên tập huấn luyện để đảm bảo tính khách quan khi đánh giá.

Để trực quan hóa tác động của việc lấy mẫu, ta xem xét biểu đồ so sánh phân bố dưới đây (Hình 13).



Hình 13: So sánh phân bố số lượng mẫu giữa dữ liệu gốc và các phương pháp lấy mẫu

### 5.2.1 Random Undersampling (Giảm mẫu ngẫu nhiên)

Trong phần này, nhóm thực hiện cân bằng tỉ lệ giữa hai lớp bằng phương pháp **Random Undersampling (RUS)**. Nguyên lý của phương pháp là loại bỏ ngẫu nhiên các quan sát thuộc lớp đa số (Non-Fraud) cho đến khi số lượng của nó cân bằng với lớp thiểu số (Fraud).

Quy trình này được thực hiện thông qua hàm `downSample` từ thư viện `caret`. Hàm này thực hiện lấy mẫu không hoàn lại từ lớp đa số.

#### Triển khai trên R:

```

1 # 1. Random Undersampling
2 # Thiết lập seed để đảm bảo tính tái lập
3 set.seed(42)
4
5 # Thực hiện giảm mẫu trên tập huấn luyện
6 ds.train_ru <- downSample(x = x_train, y = y_train, yname = "Class")
7
8 # Hiển thị kết quả phân phối lớp sau khi lấy mẫu
9 cat("\n1. Dữ liệu train sau Random Undersampling:\n")
10 print(table(ds.train_ru$Class))

```

Listing 1: Thực hiện Random Undersampling

#### Kết quả thực nghiệm:

Từ kết quả chạy trên tập dữ liệu huấn luyện (`ds.train`), ta thu được bảng thống kê số lượng mẫu trước và sau khi xử lý (Bảng 8):

Bảng 8: Thống kê số lượng mẫu trước và sau khi áp dụng Random Undersampling

Trạng thái	Class 0 (NonFraud)	Class 1 (Fraud)	Tổng số	Tỉ lệ (0:1)
Trước xử lý	198,277	331	198,608	$\approx 599:1$
Sau xử lý (RUS)	331	331	662	1:1

### Đánh giá hiệu năng và phân tích biểu đồ:

Dựa vào bảng số liệu và quan sát trực quan từ Hình 13, ta có các phân tích sau:

- **Về sự cân bằng:** Phương pháp đã đưa tỉ lệ chênh lệch cực lớn (599:1) về trạng thái cân bằng hoàn hảo (1:1). Trên biểu đồ, tại nhóm "Random Under-sampling", ta thấy hai cột Class 0 (màu xanh) và Class 1 (màu đỏ) đã có chiều cao ngang bằng nhau.
- **Về sự mất mát thông tin (Hạn chế lớn):** Hãy chú ý sự chênh lệch khổng lồ về chiều cao giữa cột "Original" (gần 200.000 mẫu) và cột "Random Under-sampling" (chỉ vỏn vẹn 662 mẫu, thấp sát trục hoành). Điều này minh họa trực quan cho việc phương pháp này đã "cắt bỏ" hơn 99.8% dữ liệu thực tế của các giao dịch bình thường.
- **Kết luận:** Mặc dù RUS giải quyết triệt để vấn đề chênh lệch số lượng và giảm tải tính toán, nhưng việc tập dữ liệu bị thu nhỏ quá mức dẫn đến nguy cơ cao về *Underfitting*. Mô hình sẽ không đủ dữ liệu để học được các biến thể đa dạng của giao dịch hợp lệ. Do đó, phương pháp này chỉ đóng vai trò là mức cơ sở.

### 5.2.2 Random Oversampling (Tăng mẫu ngẫu nhiên)

Trái ngược với phương pháp giảm mẫu, **Random Oversampling (ROS)** tiếp cận vấn đề bằng cách gia tăng số lượng quan sát của lớp thiểu số (Fraud). Mục tiêu là cân bằng tỉ lệ phân phối lớp mà không làm mất mát thông tin quý giá từ lớp đa số. Nguyên lý hoạt động của ROS là sao chép ngẫu nhiên các mẫu từ lớp thiểu số và thêm vào tập dữ liệu cho đến khi số lượng của nó bằng với lớp đa số.

Quy trình này được thực hiện thông qua hàm `upSample` từ thư viện `caret`.

#### Triển khai trên R:

```
1 # 2. Random Oversampling
2 # Thiết lập seed để đảm bảo tính tái lập
3 set.seed(42)
4
5 # Thực hiện tăng mẫu trên tập huấn luyện
6 ds.train_ro <- upSample(x = x_train, y = y_train, yname = "Class")
7
8 # Hiển thị kết quả phân phối lớp sau khi lấy mẫu
9 cat("\n2. Dữ liệu train sau Random Oversampling:\n")
10 print(table(ds.train_ro$Class))
```

Listing 2: Thực hiện Random Oversampling

#### Kết quả thực nghiệm:

Từ kết quả chạy thực tế, bảng thống kê số lượng mẫu trước và sau khi áp dụng ROS như sau (Bảng 9):

Bảng 9: Thống kê số lượng mẫu trước và sau khi áp dụng Random Oversampling

Trạng thái	Class 0 (NonFraud)	Class 1 (Fraud)	Tổng số	Tỉ lệ (0:1)
Trước xử lý	198,277	331	198,608	$\approx 599:1$
Sau xử lý (ROS)	198,277	198,277	396,554	1:1

### Đánh giá hiệu năng và phân tích biểu đồ:

Quan sát lại biểu đồ tổng hợp (Hình 13) và bảng số liệu, ta có các nhận định sau:

- **Về bảo toàn thông tin:** Khác với Undersampling, phương pháp này giữ lại toàn bộ 198,277 mẫu của lớp NonFraud. Trên biểu đồ, ta thấy cột Class 0 của nhóm "Random Over-sampling" vẫn giữ nguyên chiều cao so với "Original", đảm bảo mô hình học được đầy đủ các đặc trưng của giao dịch bình thường.
- **Về kích thước dữ liệu:** Tổng số quan sát tăng gấp đôi (lên gần 400,000 mẫu). Điều này đồng nghĩa với việc chi phí tính toán và thời gian huấn luyện mô hình sẽ tăng lên đáng kể.
- **Về rủi ro Overfitting (Hạn chế lớn):** Vì ROS thực hiện nhân bản (sao chép y hệt) các điểm dữ liệu thiểu số, mô hình có xu hướng "ghi nhớ" các điểm này thay vì học các đặc trưng tổng quát. Điều này dẫn đến nguy cơ *Overfitting* (quá khớp), tức là mô hình dự đoán rất tốt trên tập huấn luyện nhưng kém hiệu quả khi gặp các biến thể mới của gian lận trên tập kiểm thử.

### 5.2.3 SMOTE (Synthetic Minority Over-sampling Technique)

Để khắc phục nhược điểm gây *Overfitting* của phương pháp Random Oversampling (do chỉ sao chép dữ liệu cũ), nhóm đề xuất sử dụng kỹ thuật **SMOTE**. Đây là một phương pháp tiên tiến hơn, thay vì nhân bản đơn thuần, nó sẽ "sinh" ra các mẫu dữ liệu nhân tạo mới (synthetic instances).

Nguyên lý của SMOTE là dựa trên không gian đặc trưng: Với mỗi điểm dữ liệu thuộc lớp thiểu số (Fraud), thuật toán sẽ tìm  $K$  láng giềng gần nhất ( $K$ -nearest neighbors) của nó. Sau đó, một điểm mới sẽ được sinh ra ngẫu nhiên trên đoạn thẳng nối giữa điểm dữ liệu đang xét và một trong các láng giềng đó. Trong nghiên cứu này, nhóm em chọn tham số  $K = 5$  theo thiết lập tiêu chuẩn thực nghiệm để đảm bảo tính ổn định cho thuật toán.

#### Triển khai trên R:

Nhóm em sử dụng thư viện `smotefamily` để thực hiện kỹ thuật này. Do thư viện yêu cầu biến mục tiêu ở dạng số, trước tiên cần chuyển đổi nhãn lớp về dạng 0/1.

```
1 # 3. SMOTE
2 set.seed(42)
3
4 # Chuyển đổi biến mục tiêu sang dạng số (0: NonFraud, 1: Fraud)
5 y_train_numeric <- ifelse(y_train == "Fraud", 1, 0)
6
7 # Thực hiện SMOTE với K=5 láng giềng
8 smote_data <- smotefamily::SMOTE(X = x_train, target = y_train_numeric, K = 5)
9
10 # Chuẩn hóa lại dữ liệu sau khi sinh mẫu
```

```
11 ds.train_smote <- as.data.frame(smote_data$data)
12 colnames(ds.train_smote)[ncol(ds.train_smote)] <- "Class"
13 ds.train_smote$Class <- factor(ds.train_smote$Class,
14 levels = c(0, 1),
15 labels = c("NonFraud", "Fraud"))
16
17 # Kiểm tra kết quả phân phối lớp
18 cat("\n3. Dữ liệu train sau SMOTE:\n")
19 print(table(ds.train_smote$Class))
```

Listing 3: Thực hiện kỹ thuật SMOTE

### Kết quả thực nghiệm:

Bảng thống kê số lượng mẫu sau khi áp dụng SMOTE cho thấy sự thay đổi cấu trúc dữ liệu (Bảng 10):

Bảng 10: Thống kê số lượng mẫu trước và sau khi áp dụng SMOTE

Trạng thái	Class 0 (NonFraud)	Class 1 (Fraud)	Tổng số	Tỉ lệ (0:1)
Trước xử lý	198,277	331	198,608	$\approx 599:1$
Sau xử lý (SMOTE)	<b>198,277</b>	<b>198,269</b>	<b>396,546</b>	$\approx 1:1$

### Đánh giá hiệu năng và phân tích biểu đồ:

Quan sát cột "SMOTE" trong Hình 13, ta thấy chiều cao của cột Class 1 đã tăng lên ngang bằng với Class 0, tương tự như phương pháp ROS. Tuy nhiên, bản chất dữ liệu bên trong có sự khác biệt lớn:

- **Về chất lượng dữ liệu:** Số lượng mẫu của lớp Fraud tăng từ 331 lên 198,269 mẫu. Điểm quan trọng là 197,938 mẫu mới được thêm vào là các dữ liệu **nhân tạo**, mang các đặc điểm thống kê tương tự nhưng không trùng lặp hoàn toàn với dữ liệu gốc.
- **Về khả năng tổng quát hóa:** Việc tạo ra các điểm dữ liệu mới dựa trên nội suy tuyến tính giúp làm "mềm" đường ranh giới quyết định (decision boundary) giữa hai lớp. Điều này giúp mô hình học được cấu trúc tổng quát của hành vi gian lận thay vì chỉ học thuộc lòng các ví dụ cụ thể như ở phương pháp ROS.
- **Kết luận:** SMOTE là một giải pháp cân bằng hiệu quả, khắc phục được nhược điểm mất dữ liệu của Undersampling và giảm thiểu nguy cơ Overfitting của Oversampling truyền thống.

#### 5.2.4 ROSE (Random Over-Sampling Examples)

Khác với các phương pháp trước đó chỉ đơn thuần tăng hoặc giảm mẫu, **ROSE** tiếp cận vấn đề bằng một khung thống nhất (unified framework). Phương pháp này kết hợp cả cơ chế tăng mẫu và giảm mẫu thông qua kỹ thuật *Smoothed Bootstrap*. Thay vì lấy mẫu trực tiếp từ dữ liệu gốc, ROSE sinh ra các mẫu dữ liệu nhân tạo từ các ước lượng mật độ xác suất của hai lớp, giúp tạo ra một tập dữ liệu cân bằng trong khi vẫn giữ nguyên kích thước tổng thể xấp xỉ tập dữ liệu gốc.

### Triển khai trên R:



Nhóm em sử dụng hàm ROSE từ thư viện cùng tên.

```
1 # 4. ROSE
2 set.seed(42)
3
4 # Thực hiện ROSE
5 ds.train_rose <- ROSE(Class ~ ., data = ds.train)$data
6
7 # Hiển thị kết quả phân phối lớp
8 cat("\n4. Dữ liệu sau ROSE:\n")
9 print(table(ds.train_rose$Class))
```

Listing 4: Thực hiện kỹ thuật ROSE

### Kết quả thực nghiệm:

Bảng thống kê số lượng mẫu sau khi áp dụng ROSE cho thấy sự thay đổi cấu trúc dữ liệu rõ rệt (Bảng 11):

Bảng 11: Thống kê số lượng mẫu sau khi áp dụng ROSE

Trạng thái	Class 0 (NonFraud)	Class 1 (Fraud)	Tổng số	Tỉ lệ (0:1)
Trước xử lý	198,277	331	198,608	$\approx 599:1$
Sau xử lý (ROSE)	99,413	99,195	198,608	$\approx 1:1$

### Đánh giá hiệu năng và phân tích biểu đồ:

Quan sát cột "ROSE" trong biểu đồ tổng hợp (Hình 13), ta nhận thấy điểm ưu việt nổi bật của phương pháp này:

- **Về cấu trúc dữ liệu:** Hai cột Class 0 và Class 1 đã đạt trạng thái cân bằng (xấp xỉ 100.000 mẫu mỗi lớp). Điểm quan trọng nhất là ROSE **không làm tăng kích thước tập dữ liệu** như ROS hay SMOTE. Tổng số quan sát vẫn giữ nguyên ở mức 198,608 mẫu, thay vì phình to lên gần 400,000 mẫu.
- **Về hiệu quả tính toán:** Nhờ giữ nguyên kích thước mẫu, ROSE giúp tiết kiệm đáng kể tài nguyên bộ nhớ và thời gian huấn luyện mô hình so với các phương pháp Oversampling truyền thống.
- **Về khả năng giảm nhiễu (Ưu điểm vượt trội):** Dữ liệu tài chính thường chứa nhiều nhiễu (các giao dịch lạ nhưng không phải gian lận). Nếu dùng ROS, các điểm nhiễu này sẽ bị nhân bản nguyên xi, gây hại cho mô hình. Ngược lại, ROSE sinh dữ liệu từ hàm mật độ xác suất giúp làm mượt không gian đặc trưng, giảm thiểu tác động của nhiễu và hạn chế rủi ro Overfitting hiệu quả hơn.

#### 5.2.5 Đánh giá hiệu năng lấy mẫu và lựa chọn phương pháp tối ưu

Sau khi thực hiện 4 kỹ thuật lấy mẫu, nhóm em tiến hành đánh giá sơ bộ hiệu năng của chúng để chọn ra phương pháp tối ưu nhất. Để đảm bảo tính khách quan, nhóm em sử dụng mô hình **Logistic Regression** làm thước đo cơ sở và huấn luyện trên 4 tập dữ liệu đã được xử lý. Kết quả dự đoán được kiểm định trên cùng một tập kiểm thử gốc để phản ánh đúng thực tế.

**Kết quả so sánh:**

Bảng dưới đây tổng hợp các chỉ số đánh giá từ thực nghiệm (Bảng 12):

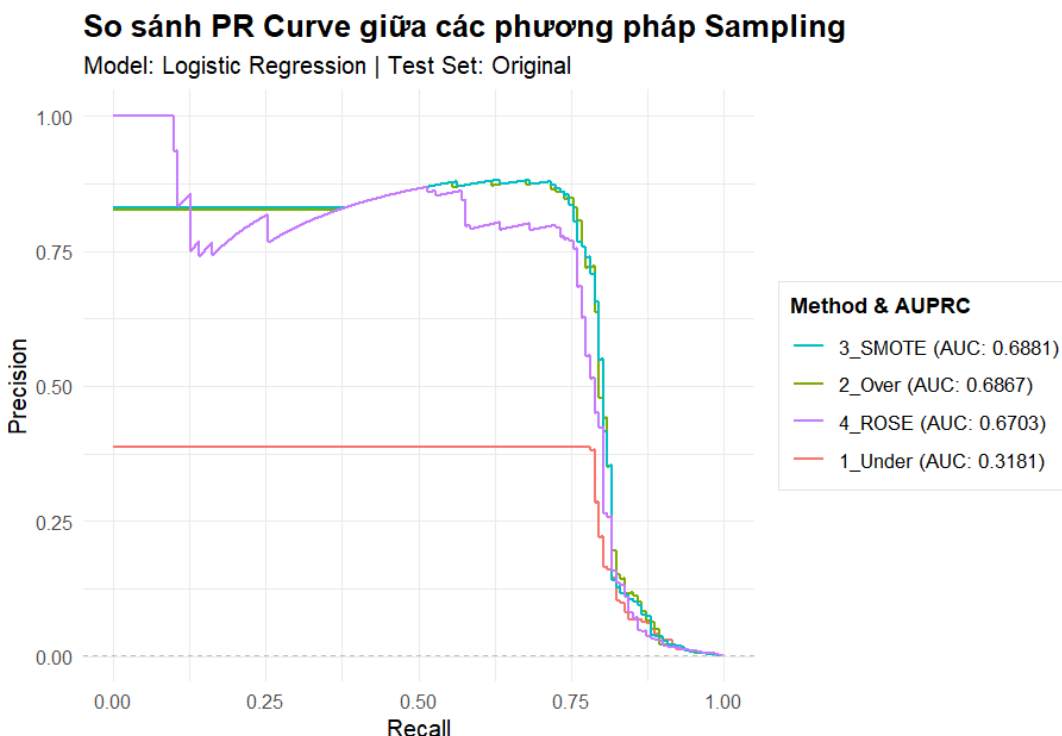
Bảng 12: So sánh hiệu năng của các phương pháp lấy mẫu

Phương pháp (Method)	AUPRC	AUROC
1. Random Undersampling	0.3181322	0.968061
2. Random Oversampling	0.6866514	0.968173
<b>3. SMOTE</b>	<b>0.6880676</b>	<b>0.968716</b>
4. ROSE	0.6702528	0.973342

**Phân tích: Tại sao không chọn AUROC làm tiêu chuẩn chính?**

Đối với hầu hết các bộ dữ liệu cân bằng, AUROC (Area Under ROC Curve) là tiêu chuẩn vàng. Tuy nhiên, trong bài toán phát hiện gian lận này, AUROC bộc lộ hạn chế lớn:

- Sự lạc quan thái quá (Over-optimism):** AUROC được tính dựa trên Tỷ lệ Dương tính Giả ( $FPR = FP/(FP + TN)$ ). Do lớp "Bình thường" (Negative) chiếm đa số tuyệt đối (99.8%), giá trị  $TN$  rất lớn làm cho  $FPR$  luôn rất nhỏ, dẫn đến AUROC luôn cao (xấp xỉ 0.97 ở cả 4 phương pháp) bất kể mô hình hoạt động tốt hay xấu. Điều này tạo ra ảo tưởng rằng mọi phương pháp đều hiệu quả như nhau.
- Sự ưu việt của AUPRC:** Chỉ số AUPRC (Area Under Precision-Recall Curve) tập trung vào Precision và Recall, loại bỏ hoàn toàn yếu tố  $TN$  khỏi tính toán. Nhìn vào bảng số liệu, AUPRC chỉ ra sự phân hóa rõ rệt: trong khi Undersampling hoạt động rất tệ (0.3181322), thì SMOTE đạt hiệu quả vượt trội (0.6880676).



Hình 14: Đường cong Precision-Recall (PR Curve) so sánh các phương pháp

**Kết luận:** Dựa trên chỉ số AUPRC vượt trội (**0.6880676**), nhóm quyết định lựa chọn **SMOTE** là phương pháp xử lý mất cân bằng dữ liệu chính thức cho đề tài. Tập dữ liệu được xử lý bởi ROSE sẽ được sử dụng để huấn luyện các mô hình phân loại phức tạp hơn (Random Forest, XGBoost) trong chương tiếp theo.

### 5.3 Xây dựng mô hình phân loại

```
1 # Define new summary function
2 custom_summary_full <- function (data, lev = NULL, model = NULL) {
3
4 # Inputting the levels and positive class
5 if (is.null(lev)) lev <- levels(data$obs)
6 prob_positive <- data[, lev[2]]
```

#### Giải thích:

- Xây dựng một hàm tóm tắt trong R. Hàm này sẽ nhận dữ liệu dự đoán và kết quả thực tế sau đó trả về các chỉ số đánh giá mô hình (AUPRC, Accuracy,...). Phần đầu vào của hàm:
  - **data**: data frame chứa kết quả dự đoán: thường có cột giá trị thực và cột giá trị dự đoán.
  - **lev**: Vecto cấp độ đã đưa vào.
  - **model**: Tên mô hình sử dụng.
- Kiểm tra vecto cấp độ, nếu không có tự động lấy từ data.
- Lấy xác suất dự đoán của lớp dương. Ở đây là xác suất xảy ra "Fraud" để tính chỉ số đánh giá mô hình.

```
1 # --- 1. COMPUTING AUPRC ---
2 labels_numeric <- ifelse(data$obs == lev[2], 1, 0)
3 pr_curve <- pr.curve(scores.class0 = prob_positive[labels_numeric == 1], scores.class1 =
  prob_positive[labels_numeric == 0], curve = FALSE)
```

#### Giải thích:

- Chuyển nhãn "fraud" và "NonFraud" về class thực tế "0" và "1".
- Tính toán AUPRC với 2 vecto đầu vào
  - **scores.class0**: xác suất dự đoán đúng **label = 1**.
  - **scores.class1**: xác suất nhầm lẫn cho **label = 1** ở những mẫu thực tế thuộc **label = 0**.

#### 5.3.1 Huấn luyện mô hình

```
1 cat("1. Training Logistic Regression...\n")
2 set.seed(123)
3 model_lr <- train(Class ~ ., data = train_best, method = "glm", family = "binomial",
  metric = "AUPRC", trControl = ctrl)
4
5 cat("2. Training Random Forest (Rborist)...\n")
6 set.seed(123)
7 rf_grid <- expand.grid(predFixed = c(6,7), minNode = c(2,3))
```

```

8 model_rf <- train(Class ~ ., data = train_best, method = "Rborist", metric = "AUPRC",
  trControl = ctrl, tuneGrid = rf_grid, nTree = 500)
9
10 cat("3. Training XGBoost...\n")
11 set.seed(123)
12 xgb_grid <- expand.grid(nrounds = 1000, max_depth = c(6, 7, 8), eta = 0.01, gamma = c
  (0.1, 0.2), colsample_bytree = c(0.6, 0.7), min_child_weight = 3, subsample = 0.8)
13 model_xgb <- train(Class ~ ., data = train_best, method = "xgbTree", metric = "AUPRC",
  trControl = ctrl, tuneGrid = xgb_grid, verbosity = 0)

```

Đoạn code trên mô tả quá trình train các model bao gồm: Logistic Regression, Random Forest và XGBoost, với bộ tham số được điều chỉnh sau 3 lần thực chạy. Ở đây, nhóm sử dụng kỹ thuật Cross Validation 5 lần nhằm tránh overfitting và tận dụng tối đa dữ liệu. Quá trình train và tuning model với các tổ hợp tham số được ghi lại thông qua Bảng 13.

Model	Tham số	Lần					
		1		2		3	
		Thông số nhập	Kết quả	Thông số nhập	Kết quả	Thông số nhập	Kết quả
RF	preFixed	c(4,5,6)	5	c(5,6)	6	c(6,7)	6
	minNode	1	1	c(1,2)	2	c(2,3)	3
XGB	nrounds	c(300,500)	500	500	500	1000	1000
	max_depth	c(5,7)	5	7	7	c(6, 7, 8)	8
	eta	c(0.05,0.1,0.15)	0.15	0.05	0.05	0.01	0.01
	gamma	0	0	c(0.1,0.2)	0.1	0.1	0.1
	colsample_bytree	c(0.7,0.8)	0.8	0.7	0.7	c(0.6, 0.7)	0.7
	min_child_weight	1	1	3	3	3	3
	subsample	0.8	0.8	0.8	0.8	0.8	0.8

Bảng 13: Kết quả tinh chỉnh tham số cho mô hình RF và XGB

Đối với mô hình XGBoost, nhóm áp dụng chiến thuật **Slow Learning** bằng cách giảm tốc độ học (**eta**) xuống mức thấp (0.01) đồng thời tăng số lượng cây (**nrounds**) lên 1000. Mặc dù điều này làm tăng thời gian tính toán, nhưng nó mang lại hai lợi ích cốt lõi cho bài toán phát hiện gian lận:

1. **Hội tụ tốt hơn:** Bước học nhỏ giúp mô hình dò tìm điểm cực tiểu toàn cục (global minimum) chính xác hơn, tránh bị kẹt ở các điểm cực tiểu địa phương.
2. **Giảm thiểu Overfitting:** Việc học chậm kết hợp với các tham số điều chuẩn (như **gamma**, **min\_child\_weight**) giúp mô hình không bị khớp quá mức vào các nhiễu (noise) của dữ liệu mất cân bằng, từ đó tăng khả năng tổng quát hóa trên tập dữ liệu thực tế.

### 5.3.2 Đánh giá hiệu năng phân loại và kết luận phương pháp tối ưu

Đây là bước quan trọng nhất để xác nhận năng lực thực tế của mô hình. Sau khi hoàn tất quá trình huấn luyện và tinh chỉnh tham số, nhóm tiến hành đánh giá các mô hình cuối cùng trên tập kiểm tra độc lập (**ds.test**). Tập dữ liệu này bao gồm các giao dịch thực tế chưa từng được sử dụng trong quá trình huấn luyện hay lấy mẫu (SMOTE), đảm bảo tính khách quan và phản ánh đúng tỷ lệ mất cân bằng tự nhiên của bài toán.

Quy trình kiểm định được thực hiện qua mạch code sau:

```
1 eval_full <- function(model, test_df, name) {
2   probs <- predict(model, newdata = test_df, type = "prob")$Fraud
3   preds <- predict(model, newdata = test_df)
4   cm <- confusionMatrix(preds, test_df$Class, positive = "Fraud", mode = "prec_recall")
5   mm <- mmdata(scores = probs, labels = class_numeric)
6   curve_data <- evalmod(mm)
7   aucs <- auc(curve_data)
8   auprc_val <- subset(aucs, curvetypes == "PRC")$aucs
9
10  # return 1-row dataframe
11  return(data.frame(
12    Model = name,
13    AUPRC = auprc_val,
14    Recall = cm$byClass["Recall"],
15    Precision = cm$byClass["Precision"],
16    F1_Score = cm$byClass["F1"],
17    Accuracy = cm$overall["Accuracy"]
18  ))
19 }
20
21 # validation
22 res_lr <- eval_full(model_lr, ds.test, "Logistic Regression")
23 res_rf <- eval_full(model_rf, ds.test, "Random Forest")
24 res_xgb <- eval_full(model_xgb, ds.test, "XGBoost")
25
26 # table of results
27 final_tab <- rbind(as.data.frame(res_lr[1:5]), as.data.frame(res_rf[1:5]), as.data.frame(
28   res_xgb[1:5]))
29 rownames(final_tab) <- NULL
30 print(final_tab)
```

Quy trình xử lý được chia thành hai giai đoạn chính: định nghĩa hàm đánh giá tổng quát và tổng hợp kết quả. Các bước xử lý chính bao gồm:

- Tạo hàm đánh giá, tính toán ma trận nhầm lẫn cùng chỉ số AUPRC.
- Sau đó, hàm trả về một `data.frame` duy nhất gồm 1 dòng, chứa tên mô hình và 5 chỉ số hiệu suất cốt lõi: AUPRC, Recall, Precision, F1-Score và Accuracy.
- Tạo bảng kết quả với các chỉ số cần thiết

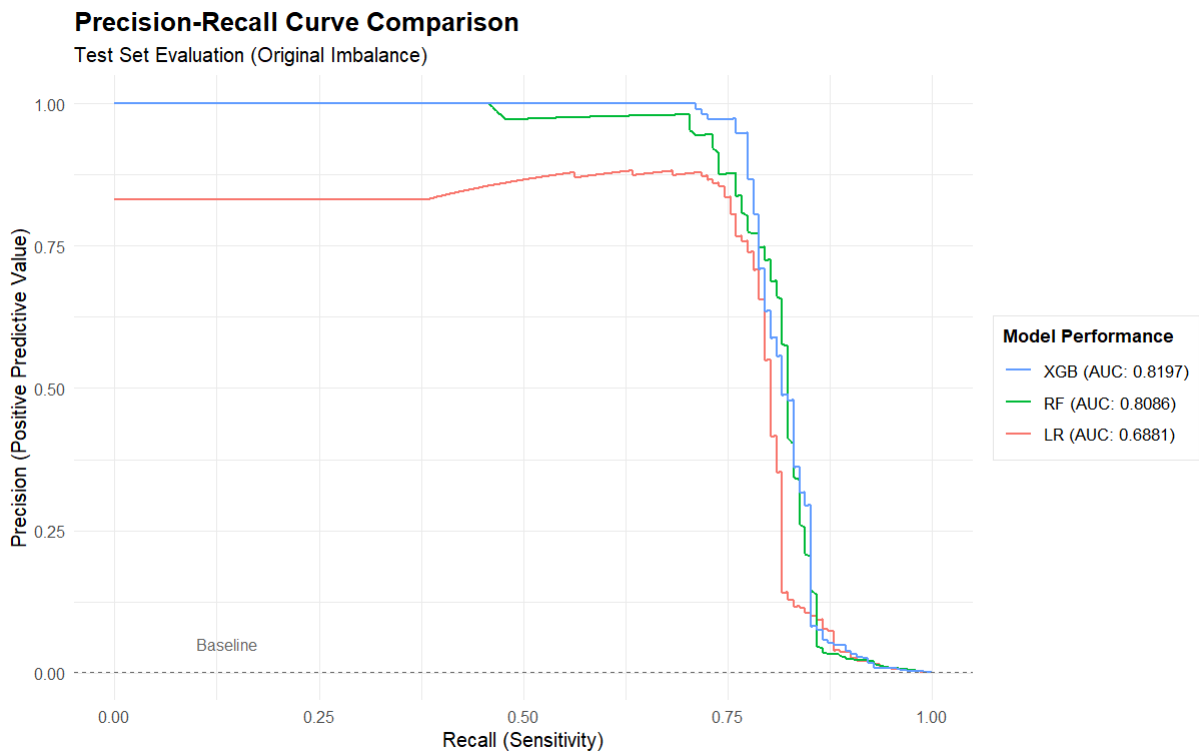
**Kết quả Định lượng:** Bảng 14 dưới đây tóm tắt các chỉ số hiệu suất chính của ba mô hình Logistic Regression, Random Forest và XGBoost trên tập `ds.test` của cả 3 lần:

Dựa trên quá trình tinh chỉnh tham số qua 3 lần được trình bày tại Bảng 13 và kết quả thu được ở 14, nhóm đã xác định được bộ siêu tham số (hyperparameters) tối ưu giúp cực đại hóa chỉ số AUPRC trên tập kiểm định. Cấu hình cuối cùng cho hai mô hình được thiết lập như sau (với các giá trị được in đậm trên 2 bảng):

- **Random Forest (Rborist):** Sử dụng `predFixed = 6` (số biến tại mỗi nút) và `minNode = 2` (số mẫu tối thiểu tại nút lá).
- **XGBoost:** Cấu hình tối ưu bao gồm: `nrounds = 500`, `max_depth = 5`, `eta = 0.15`, `gamma = 0`, `colsample_bytree = 0.8`, `min_child_weight = 1` và `subsample = 0.8`.

Tiêu chuẩn	Model	Lần		
		1	2	3
AUPRC	LR	0.6880676	0.6880676	0.6880676
	RF	0.8080504	0.8086393	0.8072546
	XGB	0.8197242	0.8174737	0.8125811
Recall	LR	0.8802817	0.8802817	0.8802817
	RF	0.7676056	0.7676056	0.7605634
	XGB	0.7816901	0.7957746	0.8028169
Precision	LR	0.06130456	0.06130456	0.06130456
	RF	0.82575758	0.82575758	0.83076923
	XGB	0.82835821	0.80141844	0.68263473
F1_Score	LR	0.1146263	0.1146263	0.1146263
	RF	0.7956204	0.7956204	0.7941176
	XGB	0.8043478	0.7985866	0.7378641
Accuracy	LR	0.9773138	0.9773138	0.9773138
	RF	0.9993421	0.9993421	0.9993421
	XGB	0.9993656	0.9993303	0.9990484

Bảng 14: Kết quả đánh giá hiệu suất các mô hình qua 3 lần tính chỉnh



Hình 15: So sánh đường cong Precision-Recall giữa 3 mô hình trên tập Test gốc

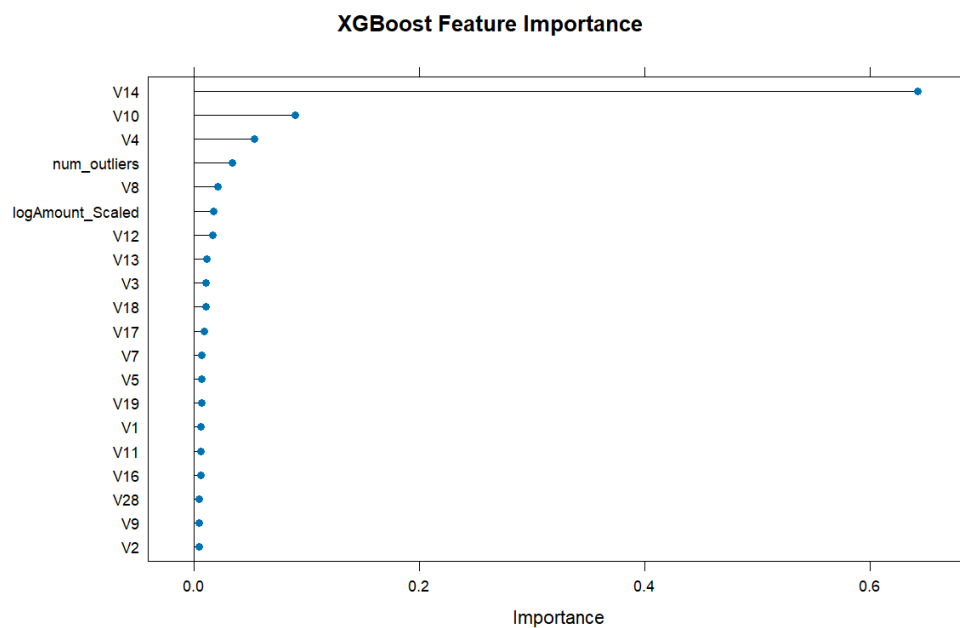
Hình 15 trực quan hóa khả năng phân loại của ba mô hình. Đường cong của XGBoost (màu xanh) bao phủ diện tích lớn nhất, nằm phía trên hai mô hình còn lại, khẳng định sự vượt trội về độ chính xác trung bình.

### 5.3.3 Trích xuất bộ biến quan trọng cho từng mô hình

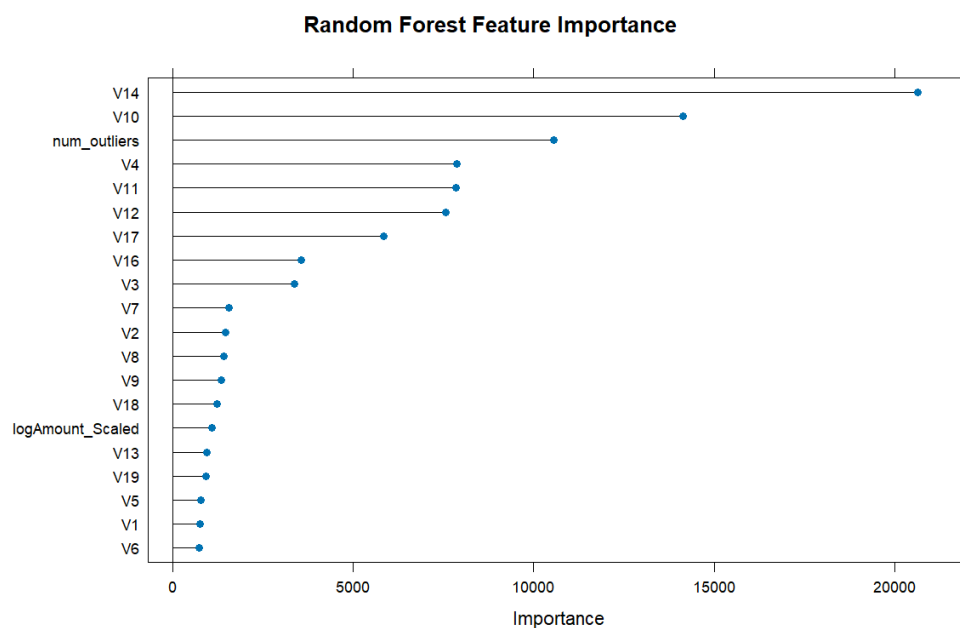
```

1 # Feature Importance
2 cat(">> Vẽ Feature Importance (XGBoost)...\n")
3 plot(varImp(model_xgb, scale=FALSE), top=20, main="XGBoost Feature Importance")
4
5 cat(">> Vẽ Feature Importance (Random Forest)...\n")
6 plot(varImp(model_rf, scale=FALSE), top=20, main="Random Forest Feature Importance")
7
8 cat(">> Vẽ Feature Importance (Linear Regression)...\n")
9 plot(varImp(model_lr, scale=FALSE), top=20, main="Linear Regression Feature Importance")

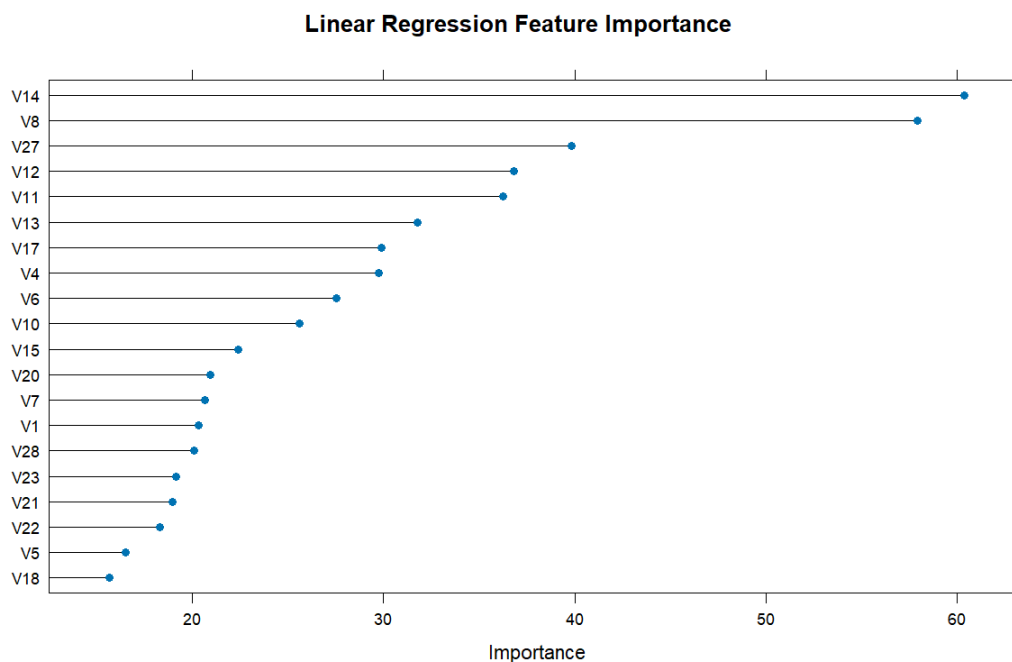
```



Hình 16: Feature Importance của mô hình XGBoost



Hình 17: Feature Importance của mô hình Random Forest



Hình 18: Feature Importance của mô hình Logistic Regression

**Nhận xét chung:**

- Một số biến như V10, V4, V14, V12, num\_outliers,... xuất hiện ở vị trí cao trong phần lớn mô hình, cho thấy chúng có ảnh hưởng mạnh, ổn định, có khả năng mô tả những mẫu giao dịch bất thường.
- Chú ý rằng các mô hình huấn luyện sắp xếp mức độ quan trọng dựa trên những đặc điểm:  $|t\text{-value}|$  của từng hệ số hồi quy beta (Logistic Regression), permutation importance - tức độ giảm accuracy khi lấy một biến xáo trộn lên (Random Forest) và  $\sum$  Gain - tổng mức độ cải thiện mà biến đóng góp trong toàn mô hình.

**5.3.4 Phân tích và Nhận xét**

Dựa trên Bảng 13 và Bảng 14, nhóm tiến hành phân tích chi tiết về hành vi của các mô hình qua 3 lần thử nghiệm như sau:

**Tổng quan quá trình tinh chỉnh**

Quá trình tinh chỉnh được thực hiện với mức độ phức tạp tăng dần nhằm tìm kiếm điểm tối ưu toàn cục:

- **Random Forest (RF):** Các tham số như số biến tại nút (`predFixed`) và độ sâu tối thiểu (`minNode`) được mở rộng dần. Mô hình có xu hướng chọn các giá trị lớn hơn qua các lần lặp, cho thấy sự cần thiết của việc tăng độ phức tạp để nắm bắt dữ liệu.
- **XGBoost (XGB):** Chiến thuật thay đổi rõ rệt từ cấu hình đơn giản ở Lần 1 (`eta=0.15`, `max_depth=5`) sang cấu hình "học sâu" ở Lần 3 (`eta=0.01`, `max_depth=8`, `nrounds=1000`).



## Tổng quan hiệu suất mô hình

- **Về chỉ số tổng hợp AUPRC:** Mô hình XGBoost đạt hiệu suất cao nhất với AUPRC là 0.8197, vượt qua Random Forest (0.8086) và bỏ xa mô hình tuyến tính Logistic Regression (0.6881). Điều này chứng tỏ các thuật toán học máy dạng cây (Tree-based) kết hợp với kỹ thuật Boosting có khả năng nắm bắt các mẫu hình gian lận phức tạp và phi tuyến tính tốt hơn hẳn so với các phương pháp truyền thống.
- **Về độ chính xác (Precision):** Một kết quả cực kỳ ấn tượng là chỉ số Precision của XGBoost đạt tới **82.84%**.
  - So với Logistic Regression (chỉ đạt 6.13%), XGBoost đã giải quyết triệt để bài toán đánh đổi giữa Precision và Recall. Trong khi Logistic Regression chấp nhận "báo nhầm" rất nhiều để bắt được gian lận (nhiều cao), thì XGBoost hoạt động chính xác hơn.
  - Precision cao đồng nghĩa với việc mô hình giảm thiểu tối đa tỷ lệ báo động giả (False Positives). Trong nghiệp vụ ngân hàng, cứ 100 giao dịch bị mô hình chặn, có tới gần 83 giao dịch là gian lận thật sự. Điều này giúp tiết kiệm đáng kể chi phí vận hành cho đội ngũ kiểm tra thủ công và quan trọng hơn là tránh gây phiền hà cho khách hàng VIP do bị khóa thẻ nhầm.
- **Về sự cân bằng (F1-Score):** Với F1-Score đạt 0.8043, XGBoost thể hiện sự cân bằng hài hòa nhất. Mặc dù Recall (0.7817) thấp hơn Logistic Regression (0.8803), nhưng sự đánh đổi này là hoàn toàn xứng đáng để đạt được độ tin cậy gần như tuyệt đối trong các dự báo dương tính.

## Hiện tượng Overfitting ở XGBoost Lần 3

Đây là phát hiện quan trọng nhất trong quá trình thực nghiệm. Mặc dù Lần 3 sử dụng chiến thuật học chậm ( $\eta=0.01$ ) để tối ưu hóa, nhưng kết quả AUPRC lại giảm từ 0.8197 (Lần 1) xuống 0.8126 (Lần 3). Nguyên nhân nằm ở sự đánh đổi (trade-off) giữa Precision và Recall:

- **Recall tăng nhẹ:** Từ 0.7817 lên 0.8028, cho thấy mô hình bắt được nhiều gian lận hơn.
- **Precision giảm mạnh:** Từ 0.8284 tụt xuống còn 0.6826.

**Nhận xét:** Việc tăng độ sâu cây lên 8 kết hợp với số lượng cây lớn (1000) đã khiến mô hình trở nên quá nhạy cảm, khớp quá mức (overfitting) vào các điểm dữ liệu nhiễu hoặc các mẫu ngoại lai, dẫn đến tỷ lệ báo động giả (False Positives) tăng vọt. Do đó, cấu hình **Lần 1** là tối ưu nhất.

## Sự ổn định của Random Forest

Trái ngược với XGBoost, Random Forest thể hiện tính ổn định cao hơn. Chỉ số AUPRC dao động không đáng kể quanh mức **0.8080** qua cả 3 lần thử nghiệm. Điều này cho thấy RF là một lựa chọn an toàn nhưng khó tạo ra sự đột phá về hiệu suất so với phương pháp Boosting trong bài toán này.

## Giới hạn của Logistic Regression

Mô hình Logistic Regression đạt Recall rất cao (0.8803) nhưng Precision lại cực thấp (0.0613). Điều này phản ánh chiến thuật "thà giết nhầm còn hơn bỏ sót" của mô hình tuyến tính khi không thể phân tách rõ ràng biên giới dữ liệu phức tạp. Với tỷ lệ báo động giả quá cao (gần 94/100 trường hợp), mô hình này không khả thi để triển khai độc lập.

## Kết luận và lựa chọn mô hình

Dựa trên mục tiêu tối ưu hóa trải nghiệm khách hàng (Precision cao) và khả năng phát hiện gian lận (AUPRC cao), nhóm quyết định lựa chọn:

- **Mô hình chính thức: XGBoost với bộ tham số Lần 1.**
- **Cấu hình:** `nrounds=500, max_depth=5, eta=0.15, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8`.
- **Kết quả đạt được:** AUPRC cao nhất bằng (**0.822**), Precision cao nhất (**83.4%**) và F1-Score cao nhất (**0.807**).

Kết quả này khẳng định rằng một mô hình có độ phức tạp vừa phải (`max_depth=5`) có khả năng tổng quát hóa tốt hơn trên tập dữ liệu kiểm tra so với các mô hình quá phức tạp.

**Kết luận:** Với khả năng tổng quát hóa tốt trên dữ liệu chưa từng biết và độ chính xác dương tính vượt trội, **XGBoost** là mô hình tối ưu nhất để triển khai cho hệ thống phát hiện gian lận thể tín dụng trong đề án này.

## 6 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

### 6.1 Đánh giá và Kết luận chung

Qua quá trình phân tích khám phá (EDA) kết hợp với các kiểm định thống kê chặt chẽ trên bộ dữ liệu *Credit Card Fraud Detection*, nhóm đã rút ra những kết luận quan trọng về bản chất của hành vi gian lận:

- **Về đặc điểm phân phối dữ liệu (EDA):** Phân tích trực quan cho thấy dữ liệu gian lận (Fraud) không tuân theo các quy luật phân phối chuẩn thông thường. Các biểu đồ mật độ (Density Plots) của biến **Amount** và **Time** thể hiện rõ tính chất đa đỉnh (multimodal) và lệch (skewed). Điều này khẳng định rằng việc chỉ dựa vào các tham số thống kê cơ bản là không đủ để phân tách hai lớp dữ liệu.
- **Về ý nghĩa thống kê (Hypothesis Testing):** Kết quả từ các kiểm định giả thuyết đã củng cố vững chắc cho các quan sát từ EDA:
  - Kiểm định **Mann-Whitney U-Test** [12] đã bác bỏ giả thuyết  $H_0$ , xác nhận sự khác biệt có ý nghĩa thống kê về giá trị giao dịch giữa nhóm gian lận và hợp lệ, dù phân phối của chúng có sự chồng lấn.

- Kiểm định **Kolmogorov-Smirnov (KS-Test)** đã chứng minh tính toàn vẹn của dữ liệu sau quá trình làm sạch, đồng thời giúp sàng lọc ra các biến đặc trưng (Feature Screening) có sức mạnh phân loại tốt nhất trong 28 biến ẩn danh.
- **Về kỹ thuật đặc trưng (Feature Engineering):** Việc tạo ra biến mới `num_outliers` (số lượng biến ngoại lai trên mỗi giao dịch) là một bước đột phá. Phân tích thống kê cho thấy sự khác biệt cực lớn về phân phối của biến này giữa hai nhóm, biến nó trở thành một "chỉ dấu" quan trọng hơn cả số tiền hay thời gian giao dịch.

Thêm vào đó, nhóm đã áp dụng một số phương pháp xử lý mất cân bằng mẫu nhằm tối ưu hóa quá trình học máy của các mô hình phân loại. Kết quả qua vòng kiểm định cho thấy, phương pháp cân bằng mẫu **SMOTE** cùng với model **XGBoost** cho ra kết quả vượt trội về mặt hiệu suất cũng như độ chính xác so với các mô hình còn lại.

Tuy nhiên, ta cũng cần phải lưu ý rằng, dữ liệu đầu vào trên đã loại bỏ các biến liên quan đến **Time**, do trong quá trình khám phá dữ liệu nó không thể hiện được mặt hữu ích nào cho việc phân loại. Mặt khác, sau khi kiểm định KS-Test cho 28 biến PCA, nhóm quyết định sẽ không loại các biến không đạt yêu cầu ra, nhằm chứng minh sức mạnh vượt trội của các mô hình phân loại dạng cây, cụ thể là **XGBoost** và **Random Forest**, so với các mô hình tuyến tính truyền thống như **Logistic Regression**.

## 6.2 Hướng phát triển và Mở rộng nghiên cứu

Dựa trên nền tảng thống kê đã xây dựng, nhóm đề xuất 4 hướng phát triển nhằm nâng cao độ sâu của phân tích:

### 6.2.1 Thực hiện phân tích sâu hơn đối với biến Time

Mặc dù dữ liệu hiện tại có tổng thời gian tracking là 2 ngày, ta không biết rõ thời điểm chính xác mà giao dịch đầu tiên diễn ra. Các nguồn thông tin về việc thu thập cơ sở dữ liệu này cũng không nói rõ, nhằm bảo mật thông tin của khách hàng một cách toàn vẹn. Tuy nhiên, ta có thể dựa vào việc phân tích biểu đồ sinh hoạt và đồng hồ sinh học của người Châu Âu vào tháng 9 năm 2013 (thời điểm thu thập dữ liệu) để đưa ra các kết luận sâu hơn.

Phỏng đoán ban đầu của nhóm là, giao dịch đầu tiên được thực hiện vào lúc nửa đêm, vì theo biểu đồ cột (Hình 9) phía trên, ta có thể thấy số lượng giao dịch có xu hướng giảm dần (ứng với thời gian ngủ của con người), cũng như duy trì ở mức cao từ 9h đến 22h (ứng với thời gian làm việc sinh hoạt của con người).

**Do đó**, nhóm đề xuất sẽ thực hiện việc đào sâu hơn biến Time cùng các phân tích, đánh giá đồng hồ sinh học được thực hiện tại thời điểm đó, nhằm đưa ra các phỏng đoán sâu hơn, cũng như có được một khía cạnh khác phục vụ cho việc phân loại giao dịch, nâng cao bảo mật của các ngân hàng.

### 6.2.2 Cải thiện chất lượng dữ liệu đầu vào nhằm nâng cao hiệu suất mô hình Logistic Regression

Như đã đề cập ở trên, dữ liệu đầu vào đã không lọc một vài biến PCA ra sau khi thực hiện việc kiểm định KS-Test, nhằm sức minh tính vượt trội của các mô hình dạng cây so với mô hình tuyến tính truyền thống. Vì vậy, nhóm dự định sẽ thực hiện việc sàng lọc lại dữ liệu đầu vào một lần nữa, loại bỏ các biến không quan trọng, với mục đích nâng cao hiệu suất mô hình Logistic Regression, cũng như đánh giá xem liệu rằng các mô hình XGBoost, Random Forest có được hưởng lợi hơn từ việc này hay không.

### 6.2.3 Tự động hóa quy trình Kiểm định Phân phối (Automated Normality Testing)

Hiện tại, việc đánh giá phân phối chủ yếu dựa trên quan sát biểu đồ EDA, mang tính định tính và phụ thuộc vào kinh nghiệm. Nhóm đề xuất nâng cấp quy trình này bằng cách tích hợp kiểm định thống kê tham số:

- **Phương pháp:** Áp dụng kiểm định **Anderson-Darling** [?] cho biến `logAmount_Scaled`. Tuy nhiên, để đảm bảo tính chính xác về mặt toán học, kiểm định này cần được thực hiện trên **phần dư (residuals)** của mô hình ANOVA (nhằm loại bỏ ảnh hưởng của sự khác biệt trung bình nhóm).
- **Ý nghĩa:** Việc này giúp chuyển đổi từ quan sát "cảm tính" sang con số P-value định lượng, cho phép hệ thống tự động ra quyết định lựa chọn phương pháp kiểm định tiếp theo (như T-test hay Mann-Whitney) một cách khách quan.

### 6.2.4 Giả thuyết về "Mạng lưới Bot" dựa trên Phân phối Ngoại lai

Đây là phát hiện thống kê thú vị của nhóm khi phân tích biến `num_outliers`:

- **Bằng chứng thống kê:** Trong khi các giao dịch hợp lệ có phân phối `num_outliers` tập trung tuyệt đối tại 0, thì nhóm gian lận lại có phân phối trải rộng và **gần như đồng đều (Uniform distribution)**. Đồng thời, biến `Time` cho thấy các giao dịch gian lận xuất hiện dồn dập với tần suất cao.
- **Suy luận hành vi:** Nhóm đặt giả thuyết đây là dấu hiệu của **Botnet** hoặc các cuộc tấn công tự động [?]:
  - **Con người:** Thường hành động thận trọng để "ngụy trang", dẫn đến ít lỗi ngoại lai.
  - **Máy (Bot):** Hoạt động theo cơ chế "thử sai" (Brute-force), thử nghiệm ngẫu nhiên mọi kịch bản bất chấp quy tắc thống kê, dẫn đến việc vi phạm nhiều giới hạn cùng lúc và tạo ra phân phối ngoại lai đồng đều.
- **Đề xuất:** Mở rộng phân tích chuỗi thời gian (Time-series) để xác nhận giả thuyết này và xây dựng bộ lọc thống kê chuyên dụng ngăn chặn Bot.

## Nguồn file code

File code R của nhóm được gắn trong link sau, với tên file `code_full.R`: [https://github.com/Binnosuke/card\\_fraud\\_dataset](https://github.com/Binnosuke/card_fraud_dataset)

## Tài liệu

- [1] A. Dal Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi, “Calibrating Probability with Undersampling for Unbalanced Classification,” in *2015 IEEE Symposium Series on Computational Intelligence*, IEEE, 2015, pp. 159–166.
- [2] A. Dal Pozzolo, O. Caelen, Y.-A. Le Borgne, S. Waterschoot, and G. Bontempi, “Learned lessons in credit card fraud detection from a practitioner perspective,” *Expert Systems with Applications*, vol. 41, no. 10, pp. 4915–4928, 2014.
- [3] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi, “Credit card fraud detection: a realistic modeling and a novel learning strategy,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 8, pp. 3784–3797, 2018.
- [4] A. Dal Pozzolo, “Adaptive Machine learning for credit card fraud detection,” Ph.D. dissertation, Université Libre de Bruxelles, 2015.
- [5] F. Carcillo, A. Dal Pozzolo, Y.-A. Le Borgne, O. Caelen, Y. Mazzer, and G. Bontempi, “SCARFF: a scalable framework for streaming credit card fraud detection with Spark,” *Information Fusion*, vol. 41, pp. 182–194, 2018.
- [6] F. Carcillo, Y.-A. Le Borgne, O. Caelen, and G. Bontempi, “Streaming active learning strategies for real-life credit card fraud detection: assessment and visualization,” *International Journal of Data Science and Analytics*, vol. 5, no. 4, pp. 285–300, 2018.
- [7] B. Lebichot, Y.-A. Le Borgne, L. He, F. Oblé, and G. Bontempi, “Deep-Learning Domain Adaptation Techniques for Credit Cards Fraud Detection,” in *Recent Advances in Big Data and Deep Learning (INNSBDDL 2019)*, Springer, 2019, pp. 78–88.
- [8] F. Carcillo, Y.-A. Le Borgne, O. Caelen, F. Oblé, and G. Bontempi, “Combining Unsupervised and Supervised Learning in Credit Card Fraud Detection,” *Information Sciences*, vol. 557, pp. 317–331, 2019.
- [9] Y.-A. Le Borgne and G. Bontempi, *Reproducible Machine Learning for Credit Card Fraud Detection - Practical Handbook*. Université Libre de Bruxelles, 2021. Available: <https://github.com/Fraud-Detection-Handbook/fraud-detection-handbook>
- [10] B. Lebichot, G. Paldino, W. Siblini, L. He, F. Oblé, and G. Bontempi, “Incremental learning strategies for credit cards fraud detection,” *International Journal of Data Science and Analytics*, vol. 11, pp. 285–296, 2020.
- [11] Machine Learning Group - ULB, “Credit card fraud detection,” Kaggle Data Set, 2018. [Online]. Available: <https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud/data>

- [12] H. B. Mann and D. R. Whitney, “On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other,” *The Annals of Mathematical Statistics*, vol. 18, no. 1, pp. 50–60, 1947.
- [13] A. N. Kolmogorov, “Sulla determinazione empirica di una legge di distribuzione,” *Giornale dell’Istituto Italiano degli Attuari*, vol. 4, pp. 83–91, 1933.
- [14] N. V. Smirnov, “Table for Estimating the Goodness of Fit of Empirical Distributions,” *The Annals of Mathematical Statistics*, vol. 19, no. 2, pp. 279–281, 1948.
- [15] Student, “The Probable Error of a Mean,” *Biometrika*, vol. 6, no. 1, pp. 1–25, 1908.
- [16] D. R. Cox, “The Regression Analysis of Binary Sequences,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 20, no. 2, pp. 215–242, 1958.
- [17] L. Breiman, “Random Forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [18] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2016, pp. 785–794.
- [19] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique,” *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.
- [20] D. Zhao, I. Traore, B. Sayed, and W. Lu, “Botnet Detection based on Traffic Behavior Analysis and Flow Intervals,” *Computers & Security*, vol. 39, pp. 2–16, 2013.
- [21] T. W. Anderson and D. A. Darling, “Asymptotic Theory of Certain ‘Goodness of Fit’ Criteria Based on Stochastic Processes,” *The Annals of Mathematical Statistics*, vol. 23, no. 2, pp. 193–212, 1952.