

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN  
KHOA HỆ THÔNG THÔNG TIN**

**NGUYỄN HOÀNG HIỆP - 15520214  
TRẦN HOÀNG LUÂN - 15520452**

**KHÓA LUẬN TỐT NGHIỆP  
NGHIÊN CỨU MÔ HÌNH PHÁT HIỆN BẤT  
THƯỜNG TRONG GIAO DỊCH TÀI CHÍNH  
Research on fraud detection models in financial transactions**

**KỸ SỰ NGÀNH HỆ THÔNG THÔNG TIN**

**TP. HỒ CHÍ MINH, NĂM 2020**

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN  
KHOA HỆ THÔNG THÔNG TIN**

**NGUYỄN HOÀNG HIỆP - 15520214  
TRẦN HOÀNG LUÂN - 15520452**

**KHÓA LUẬN TỐT NGHIỆP  
NGHIÊN CỨU MÔ HÌNH PHÁT HIỆN BẤT  
THƯỜNG TRONG GIAO DỊCH TÀI CHÍNH  
Research on fraud detection models in financial transactions**

**KỸ SƯ NGÀNH HỆ THÔNG THÔNG TIN**

**GIẢNG VIÊN HƯỚNG DẪN  
TS. CAO THỊ NHẠN**

**TP. HỒ CHÍ MINH, NĂM 2020**

## **DANH SÁCH HỘI ĐỒNG BẢO VỆ KHÓA LUẬN**

Hội đồng chấm khóa luận tốt nghiệp, thành lập theo Quyết định số .....  
ngày ..... của Hiệu trưởng Trường Đại học Công nghệ Thông tin.

1. ..... – Chủ tịch.
2. ..... – Thư ký.
3. ..... – Ủy viên.
4. ..... – Ủy viên.

TP. HCM, ngày... tháng... năm 2020

**NHẬN XÉT KHÓA LUẬN TỐT NGHIỆP  
CỦA CÁN BỘ HƯỚNG DẪN**

**Tên khóa luận:**

**NGHIÊN CỨU MÔ HÌNH PHÁT HIỆN BẤT THƯỜNG  
TRONG GIAO DỊCH TÀI CHÍNH**

**Nhóm SV thực hiện:**

Nguyễn Hoàng Hiệp - 15520214  
Trần Hoàng Luân - 15520452

**Cán bộ hướng dẫn:**

TS. Cao Thị Nhạn

**Đánh giá Khóa luận**

1. Về cuốn báo cáo:

Số trang	_____	Số chương	_____
Số bảng số liệu	_____	Số hình vẽ	_____
Số tài liệu tham khảo	_____	Sản phẩm	_____

- Một số nhận xét về hình thức cuốn báo cáo:

.....  
.....  
.....  
.....

2. Về nội dung nghiên cứu:

.....  
.....

.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....

3. Về chương trình ứng dụng:

.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....

4. Về thái độ làm việc của sinh viên:

**Đánh giá chung:**

**Điểm từng sinh viên:**

Nguyễn Hoàng Hiệp: \_\_\_\_/10

Trần Hoàng Luân: \_\_\_\_/10

**Người nhận xét**

(Ký và ghi rõ họ tên)

TP. HCM, ngày... tháng... năm 2020

**NHẬN XÉT KHÓA LUẬN TỐT NGHIỆP  
CỦA CÁN BỘ PHẢN BIỆN**

Tên khóa luận:

**NGHIÊN CỨU MÔ HÌNH PHÁT HIỆN BẤT THƯỜNG  
TRONG GIAO DỊCH TÀI CHÍNH**

Nhóm SV thực hiện:

Nguyễn Hoàng Hiệp – 15520214  
Trần Hoàng Luân - 15520452

Cán bộ hướng dẫn:

TS. Cao Thị Nhạn

**Đánh giá Khóa luận**

1. Về cuốn báo cáo:

Số trang	_____	Số chương	_____
Số bảng số liệu	_____	Số hình vẽ	_____
Số tài liệu tham khảo	_____	Sản phẩm	_____

- Một số nhận xét về hình thức cuốn báo cáo:

.....  
.....  
.....  
.....

2. Về nội dung nghiên cứu:

.....  
.....

### 3. Về chương trình ứng dụng:

#### 4. Về thái độ làm việc của sinh viên:

## Đánh giá chung:

## Điểm từng sinh viên:

Nguyễn Hoàng Hiệp: \_\_\_\_/10

Trần Hoàng Luân: \_\_\_/10

## **Người nhận xét**

(Ký và ghi rõ họ tên)

## LỜI CẢM ƠN

Trong cuộc sống của chúng ta, có lẽ ai cũng đã từng thất bại hoặc thành công, dù như thế nào thì đó cũng là kết quả nỗ lực của mỗi cá nhân cũng như tập thể. Và đằng sau đó chính là sự hỗ trợ giúp đỡ từ mọi người. Hôm nay, để có thể hoàn thành được khóa luận tốt nghiệp này, nhóm tác giả xin gửi lời cảm ơn chân thành đến quý Thầy Cô trường Đại học Công nghệ Thông tin, đặc biệt là quý Thầy Cô Khoa Hệ thống Thông Tin - những người đã dùng tri thức và tâm huyết của mình để truyền đạt cho nhóm vốn kiến thức vô cùng quý báu trong khoảng thời gian học tập tại trường. Những kiến thức mà Thầy Cô truyền đạt là bước đệm quan trọng giúp nhóm có thể hoàn thành đề tài tốt hơn.

Đặc biệt, nhóm tác giả xin gửi lời cảm ơn chân thành và lòng biết ơn sâu sắc đến TS. Cao Thị Nhạn, cảm ơn Cô đã tận tình hướng dẫn và tạo điều kiện tốt nhất cho nhóm hoàn thành đề tài này. Những lời động viên, góp ý chân tình của Cô là động lực quý báu để nhóm tiếp thu được nhiều kiến thức bổ ích cũng như vượt qua những khó khăn khi tìm hiểu và thực hiện khóa luận.

Trong thời gian thực hiện đề tài, nhóm tác giả đã cố gắng vận dụng những kiến thức nền tảng đã học, kết hợp học hỏi và tìm hiểu các kiến thức mới và công nghệ để ứng dụng xây dựng đề tài khóa luận tốt nghiệp. Tuy nhiên trong quá trình thực hiện, do kiến thức và kinh nghiệm còn nhiều hạn chế, nên khó tránh khỏi những thiếu sót. Chính vì vậy, nhóm tác giả rất mong nhận được sự góp ý từ quý Thầy Cô để hoàn thiện thêm những kiến thức và kỹ năng cần thiết, làm hành trang quý báu để nhóm thực hiện các dự án thực tế trong tương lai.

Xin chân thành cảm ơn quý Thầy Cô!

Nhóm tác giả

TP.HCM, ngày 12 tháng 09 năm 2019

## ĐỀ CƯƠNG CHI TIẾT

### Tên đề tài:

- Tên Tiếng Việt: Nghiên cứu mô hình phát hiện bất thường trong giao dịch tài chính.
- Tên Tiếng Anh: Research on fraud detection models in financial transactions.

Cán bộ hướng dẫn: TS. Cao Thị Nhạn.

Địa chỉ liên hệ:

- Email: [nhanct@uit.edu.vn](mailto:nhanct@uit.edu.vn)
- Số điện thoại: 090 8822441

Thời gian thực hiện: Từ tháng 09 năm 2019 đến tháng 01 năm 2020.

### Sinh viên thực hiện:

- Trần Hoàng Luân - 15520452
- Nguyễn Hoàng Hiệp - 15520214

### Nội dung đề tài:

#### - Mục tiêu:

- Nghiên cứu, tìm hiểu phương pháp phát hiện bất thường trong giao dịch tài chính. Đề tài tiến hành nghiên cứu thực hiện các thuật toán phát hiện bất thường kết quả trên bộ dữ liệu giao dịch tài chính.
- Sau khi tạo ra các mô hình tiến hành chạy thử nghiệm để so sánh, đánh giá và chọn ra mô hình phù hợp trong việc phát hiện bất thường giao dịch.

- Đánh giá chi tiết kết quả thu được khi sử dụng mô hình trên tập dữ liệu.
- **Phạm vi:**
  - Bộ dữ liệu về các giao dịch tài chính.
  - Các kỹ thuật, thuật toán trong data mining và machine learning.
  - Platform: Colab Google.
  - Ngôn ngữ lập trình: Python.
  - Thư viện: Numpy, Pandas, Matplotlib, Sklearn, Tensorflow, Keras.
- **Đối tượng:**
  - Dữ liệu nghiên cứu là bộ dữ liệu được thu thập từ nhiều nguồn phục vụ cho mục đích nghiên cứu.
  - Các kỹ thuật khai phá dữ liệu, công cụ khai phá dữ liệu và các module lập trình trong khai phá dữ liệu.
  - Sử dụng các thuận toán trong machine learning.
- **Phương pháp thực hiện:**
  - **Khảo sát:**
    - Dựa trên tài liệu: Thu thập, đọc hiểu thông tin từ các tài liệu, giáo trình liên quan đến khai phá dữ liệu, machine learning và vấn đề liên quan đến bất thường trong tài chính.
    - Dựa trên yêu cầu người dùng.
    - Dựa trên các hệ thống sẵn có.
  - **Phân tích lý luận:**
    - Các thuật toán, phương pháp liên quan.
    - Các công nghệ liên quan.
  - **Đánh giá:**
    - Sử dụng các kỹ thuật đánh giá kết quả thực nghiệm.
- **Kết quả mong đợi:**

- Sử dụng các thuật toán machine learning để tạo ra mô hình phát hiện bất thường.
  - Đánh giá kết quả thực nghiệm trên 2 bộ dữ liệu như đã nêu trên.
- **Nội dung thực hiện:**
- **Nội dung nghiên cứu:**
    - Tìm hiểu thực trạng của đè tài và các nghiên cứu liên quan đến đè tài.
    - Phân tích đè tài: chọn mô hình để nghiên cứu và giải quyết vấn đề.
    - Tìm dữ liệu và tiền xử lý dữ liệu.
    - Nghiên cứu tìm hiểu các thuật toán lựa chọn tiêu chuẩn, kỹ thuật phù hợp cho tình huống dữ liệu mất cân bằng. Cụ thể đó là: Oversampling - SMOTE, Undersampling - RandomUnderSampler và Phương pháp kết hợp SMOTE + ENN.
    - Một số thuật toán phân lớp dự kiến nghiên cứu sử dụng: Random Forest, Multi-layer Perceptron (Artificial Neural Network Algorithms), Extreme Gradient Boosting (XGB).
    - Chạy thực nghiệm.
    - Đánh giá kết quả thực nghiệm.
- **Hướng phát triển:**
- Cải tiến những hạn chế (nếu có) của phương pháp sử dụng.
  - Chọn ra mô hình có kết quả tốt, hiệu quả cao và khả thi khi triển khai thực tế.
- **Kế hoạch thực hiện:**
- **Giai đoạn 1 (17/08/2019 - 15/09/2019):** Thảo luận chọn đè tài, tìm hiểu thực trạng, các bài viết chủ đề liên quan, bộ dữ liệu sẽ thực hiện và tiến hành chỉnh sửa, làm sạch bộ dữ liệu đã chọn và viết đề cương khóa luận.
  - **Giai đoạn 2 (16/09/2019 - 11/10/2019):** Kiểm tra lại bộ dữ liệu và xử lý dữ liệu (nếu cần). Nghiên cứu lựa chọn tiêu chuẩn, kỹ thuật:

Oversampling - SMOTE Undersampling - RandomUnderSampler và Combined Class Methods - SMOTE + ENN. Các mô hình lựa chọn cho bộ dữ liệu và tìm hiểu các thuật toán dự kiến.

- Giai đoạn 3 (12/10/2019 - 01/12/2019): Cài đặt môi trường và viết code để chạy các thuật toán đã tìm hiểu. Sử dụng các bộ dữ liệu đã chọn. Tìm hiểu các kỹ thuật đánh giá hệ thống phân lớp như: True/False Positive/Negative, Precision-Recall, F1 score. Đánh giá kết quả thực nghiệm qua các phương pháp đánh giá mô hình phân lớp.
- **Giai đoạn 4: (02/12/2019 - 20/12/2019):** Tìm hiểu các kỹ thuật nâng cao độ chính xác của phương pháp như: Gradient Descent, Back-Propagation (Artificial Neural Network Algorithms).
- **Giai đoạn 5 (21/12/2019 - 12/01/2020):** Chính sửa, bổ sung theo góp ý. Hoàn chỉnh báo cáo và tiến hành ra phản biện và bảo vệ trước hội đồng.

<b>Xác nhận của CBHD</b> <i>(Ký tên và ghi rõ họ tên)</i>	<b>TP. HCM, ngày 12 tháng 09 năm 2019</b> <b>Sinh viên</b> <i>(Ký tên và ghi rõ họ tên)</i>
TS. Cao Thị Nhạn	Trần Hoàng Luân    Nguyễn Hoàng Hiệp

## MỤC LỤC

<b>TÓM TẮT KHÓA LUẬN .....</b>	<b>1</b>
<b>CHƯƠNG 1: GIỚI THIỆU .....</b>	<b>2</b>
1.1. Đặt vấn đề .....	2
1.2. Mục tiêu luận văn.....	4
1.3. Đối tượng, phạm vi .....	5
1.3.1. Đối tượng .....	5
1.3.2. Phạm vi.....	5
1.4. Công cụ sử dụng.....	5
1.5. Bố cục báo cáo .....	5
<b>CHƯƠNG 2: CƠ SỞ LÝ THUYẾT .....</b>	<b>7</b>
2.1. Quy trình phát hiện bất thường trong giao dịch tài chính .....	7
2.2. Học máy (Machine learning) .....	8
2.2.1. Giới thiệu.....	8
2.2.2. Máy học.....	8
2.2.3. Ứng dụng của machine learning .....	9
2.2.4. Supervised Learning (Học có giám sát).....	10
2.3. Các thuật toán được sử dụng trong bài toán xác định bất thường trong giao dịch tài chính .....	11
2.3.1. Thuật toán Rừng ngẫu nhiên (Random forest).....	11
2.3.1.1. Bootstrapping và Bagging.....	12
2.3.1.1.1. Bootstrapping .....	12
2.3.1.1.2. Bagging .....	12

2.3.1.2. Ưu điểm.....	12
2.3.1.3. Cách hoạt động.....	13
2.3.2. Thuật toán tăng cường độ dốc cực cao (XGBoost).....	14
2.3.2.1. Giới thiệu.....	14
2.3.2.2. Lý do chọn XGBoost .....	15
2.3.2.3. Mô hình và thông số (Model and Parameters) .....	15
2.3.2.4. Hàm mất mát và chính quy hóa (Training Loss - Regularization) ..	16
2.3.2.5. Cây quyết định kết hợp (Decision Tree Ensembles).....	17
2.3.3. Thuật toán Multi-layer Perceptron (Artificial Neural Network Algorithms) .....	18
2.3.3.1. Giới thiệu về Perception, bài toán Perceptron và thuật toán Perceptron .....	18
2.3.3.2. Multi-layer perceptron (MLP) .....	19
2.4. Các phương pháp đánh giá một hệ thống phân lớp.....	25
2.4.1. True/False Positive/Negative .....	27
2.4.1.1. True/False Positive/Negative .....	27
2.4.1.2. Receiver Operating Characteristic curve .....	28
2.4.1.3. Area Under the Curve .....	29
2.4.1.4. Precision và Recall.....	30
2.5. Các phương pháp/kỹ thuật xử lý dữ liệu mất cân bằng .....	33
2.5.1. Phương pháp Random Undersampling .....	34
2.5.2. Phương pháp Random Oversampling – SMOTE.....	34
2.5.3. Phương pháp kết hợp SMOTE + ENN .....	34
<b>CHƯƠNG 3: MÔ HÌNH ĐỀ XUẤT.....</b>	<b>36</b>

3.1. Giới thiệu về bộ dữ liệu.....	36
3.1.1. Bộ dữ liệu Paysim .....	36
3.1.2. Bộ dữ liệu Credit Card Fraud Detection .....	38
3.1.3. Thao tác tiền xử lý dữ liệu .....	39
3.2. Ngôn ngữ lập trình được lựa chọn .....	39
3.2.1. Python .....	39
3.3. Google Colab .....	41
3.3.1. Lý do chọn Google colab .....	42
3.3.2. Hạn chế.....	43
<b>CHƯƠNG 4: THỰC NGHIỆM .....</b>	<b>45</b>
4.1. Bộ dữ liệu Paysim .....	45
4.1.1. Các bước thực hiện.....	45
4.1.2. Tìm hiểu tổng quan dữ liệu .....	45
4.1.3. Phân tích dữ liệu.....	47
4.1.3.1. Những loại giao dịch gian lận .....	47
4.1.3.2. Tìm hiểu ý nghĩa thuộc tính isFlaggedFraud .....	49
4.1.3.3. Tìm hiểu các tài khoản thương gia (bắt đầu bằng M) được gán nhãn .....	53
4.1.4. Tiền xử lý dữ liệu .....	56
4.1.5. Trực quan hóa dữ liệu – Data visualization .....	60
4.1.5.1. Phân tán theo thời gian.....	60
4.1.5.2. Phân tán theo số tiền giao dịch.....	60
4.1.5.3. Phân tán theo số tiền sai trong tài khoản đích.....	61
4.1.6. Xử lý dữ liệu mất cân bằng .....	62

4.1.6.1. Random UnderSampling .....	62
4.1.6.2. Random OverSampling .....	63
4.1.7. Ma trận tương quan (Correlation Matrices) .....	65
4.1.7.1. Dữ liệu chưa xử lý cân bằng .....	66
4.1.7.2. Dữ liệu cân bằng theo phương pháp UnderSampling .....	67
4.1.7.3. Dữ liệu cân bằng theo phương pháp Oversampling.....	68
4.1.8. Phát hiện bất thường (Anomaly detection) .....	68
4.1.8.1. Phân phối trước khi loại bỏ các giá trị ngoại vi .....	69
4.1.8.2. Loại bỏ các giá trị ngoại vi.....	70
4.1.8.2.1. Dữ liệu xử lý bằng phương pháp Undersampling.....	70
4.1.8.2.2. Dữ liệu xử lý bằng phương pháp Oversampling.....	71
4.1.8.3. Phân phối sau khi loại bỏ giá trị ngoại vi.....	71
4.1.8.3.1. Dữ liệu xử lý bằng phương pháp Undersampling.....	71
4.1.8.3.2. Dữ liệu xử lý bằng phương pháp Oversampling.....	72
4.1.9. Lưu dữ liệu sau khi thực hiện cân bằng dữ liệu .....	72
4.1.10. Mô hình phát hiện bất thường sử dụng thuật toán Random forest.....	72
4.1.10.1. Với bộ dữ liệu chưa xử lý mất cân bằng .....	72
4.1.10.2. Với bộ dữ liệu đã xử lý cân bằng .....	74
4.1.11. Mô hình phát hiện bất thường sử dụng thuật toán Extreme Gradient Boosting (EGB).....	79
4.1.11.1. Với bộ dữ liệu chưa xử lý mất cân bằng .....	79
4.1.11.2. Với bộ dữ liệu đã xử lý cân bằng .....	81
4.2. Bộ dữ liệu Credit Card Fraud Detection .....	83
4.2.1. Các bước thực hiện.....	83

4.2.2. Tìm hiểu tổng quan dữ liệu .....	83
4.2.3. Phân tích dữ liệu.....	84
4.2.3.1. Phân phối xác suất một số thuộc tính.....	84
4.2.4. Tiền xử lý dữ liệu .....	87
4.2.4.1. Scaling.....	87
4.2.4.2. Splitting Data (Tách khung dữ liệu gốc).....	88
4.2.5. Xử lý dữ liệu mất cân bằng .....	89
4.2.5.1. Random Undersampling .....	89
4.2.5.2. Random Oversampling (SMOTE) .....	90
4.2.5.3. Phương pháp kết hợp SMOTE + ENN .....	90
4.2.6. Ma trận tương quan (Correlation matrix).....	91
4.2.6.1. Dữ liệu chưa xử lý cân bằng .....	91
4.2.6.2. Dữ liệu cân bằng theo phương pháp Undersampling.....	91
4.2.6.3. Dữ liệu cân bằng theo phương pháp Oversampling.....	92
4.2.6.4. Dữ liệu cân bằng theo phương pháp kết hợp SMOTE + ENN .....	92
4.2.7. Tương quan giữa các thuộc tính.....	93
4.2.8. Phát hiện bất thường (Anomaly Detection) .....	94
4.2.8.1. Dữ liệu xử lý bằng phương pháp Undersampling.....	94
4.2.8.2. Dữ liệu xử lý bằng phương pháp Oversampling.....	97
4.2.8.3. Dữ liệu xử lý bằng phương pháp kết hợp SMOTE + ENN .....	98
4.2.9. Lưu dữ liệu sau khi thực hiện cân bằng dữ liệu .....	99
4.2.10. Mô hình phát hiện bất thường sử dụng thuật toán Random forest.....	99
4.2.10.1. Với bộ dữ liệu chưa xử lý mất cân bằng .....	99
4.2.10.2. Với bộ dữ liệu đã xử lý mất cân bằng .....	101

4.2.11. Mô hình phát hiện bất thường sử dụng thuật toán Extreme Gradient Boosting (EGB).....	107
4.2.11.1. Với bộ dữ liệu chưa xử lý mất cân bằng .....	107
4.2.11.2. Với bộ dữ liệu đã xử lý mất cân bằng .....	109
<b>CHƯƠNG 5: KẾT LUẬN – HƯỚNG PHÁT TRIỂN.....</b>	<b>116</b>
5.1. Kết quả đạt được .....	116
5.2. Hướng phát triển .....	116
<b>TÀI LIỆU THAM KHẢO .....</b>	<b>1</b>
<b>PHỤ LỤC .....</b>	<b>3</b>

## **DANH MỤC HÌNH ẢNH**

Hình 2.1. Quy trình phát hiện bất thường trong giao dịch tài chính .....	7
Hình 2.2. Quá trình phát triển của Machine learning.....	9
Hình 2.3. Một số thuật toán machine learning phổ biến .....	11
Hình 2.4. Quá trình hoạt động của thuật toán Random forest trong dự đoán .....	14
Hình 2.5. Minh họa boosting.....	17
Hình 2.6. Multi-layer Perceptron .....	19
Hình 2.7. Kí hiệu liên quan đến neuron thứ 1 của lớp thứ 2.....	21
Hình 2.8. Đồ thị của hàm số sigmoid.....	22
Hình 2.9. Minh họa confusion matrix .....	26
Hình 2.10. Minh họa confusion matrix bảng kết quả.....	27
Hình 2.11. Minh họa precision và recall .....	30
Hình 2.12. Quy trình thực hiện của thuật toán SMOTE + ENN .....	35
Hình 3.1. So sánh các ngôn ngữ lập trình .....	40
Hình 3.2. Bảng so sánh giữa tốc độ xử lý của CPU, GPU và TPU .....	42
Hình 3.3. Giới thiệu về Colaboratory 1 .....	43
Hình 3.4. Giới thiệu về colaboratory 2 .....	44
Hình 4.1. Kết quả đọc bộ dữ liệu Paysim sau khi kết nối .....	45
Hình 4.2. Tổng quan các giá trị của bộ dữ liệu Paysim .....	46
Hình 4.3. Biểu đồ thể hiện số lượng theo loại giao dịch.....	47
Hình 4.4. Biểu đồ thể hiện số giao dịch bất thường và giao dịch bình thường ở mỗi loại giao dịch .....	48
Hình 4.5. Biểu đồ thể hiện số giao dịch bất thường trên mỗi loại giao dịch theo thuộc tính isFlaggedFraud) .....	50
Hình 4.6. Kết quả kiểm tra các bộ dữ liệu khi phân tách.....	57
Hình 4.7. Biểu đồ thể hiện tổng quan bộ dữ liệu sau khi tiến hành xử lý.....	57
Hình 4.8. Kết quả sau khi tiến hành tạo mới 2 thuộc tính từ các thuộc tính đã có ...	59
Hình 4.9. Sự phân tán theo thời gian của giao dịch bình thường và giao dịch bất thường .....	60

Hình 4.10. Sự phân tán theo số tiền giao dịch .....	61
Hình 4.11. Sự phân tán theo số tiền sai trong giao dịch chuyển tiền và rút tiền.....	61
Hình 4.12. Kết quả khi tiến hành gộp dữ liệu giao dịch bình thường và bất thường	63
Hình 4.13. Biểu đồ phân tán giữa các giao dịch bình thường và bất thường.....	63
Hình 4.14. Kết quả khi gộp dữ liệu giữa giao dịch bình thường và bất thường sử dụng Random Oversampling.....	64
Hình 4.15. Biểu đồ phân tán các lớp của giao dịch bất thường và bình thường sau khi cân bằng dữ liệu .....	65
Hình 4.16. Ma trận tương quan đối với dữ liệu chưa xử lý cân bằng .....	66
Hình 4.17. Ma trận tương quan đối với dữ liệu xử lý cân bằng theo phương pháp Undersampling .....	67
Hình 4.18. Ma trận tương quan khi sử dụng dữ liệu cân bằng theo phương pháp oversampling .....	68
Hình 4.19. Thể hiện phân phối trước khi loại bỏ các giá trị ngoại vi của bộ dữ liệu dùng undersampling .....	69
Hình 4.20. Thể hiện phân phối trước khi loại bỏ các giá trị ngoại vi của bộ dữ liệu dùng oversampling .....	70
Hình 4.21. Hình phân phối của bộ dữ liệu xử lý cân bằng dùng Undersampling....	71
Hình 4.22. Bảng phân phối của bộ dữ liệu xử lý cân bằng dùng Oversampling .....	72
Hình 4.23. Ma trận nhầm lẫn tập dữ liệu test của bộ dữ liệu chưa xử lý cân bằng...74	
Hình 4.24. Mô phỏng bộ dữ liệu được xử lý cân bằng dùng Oversampling .....	75
Hình 4.25. Ma trận nhầm lẫn trên tập dữ liệu test dùng Oversampling .....	76
Hình 4.26. Mô phỏng bộ dữ liệu được xử lý cân bằng dùng Undersampling .....	77
Hình 4.27. Ma trận nhầm lẫn trên tập dữ liệu test dùng Undersampling .....	78
Hình 4.28. Ma trận tương quan trên tập dữ liệu test dùng bộ dữ liệu chưa cân bằng .....	80
Hình 4.29. Mô phỏng bộ dữ liệu được xử lý cân bằng dùng Oversampling .....	81
Hình 4.30. Ma trận tương quan trên bộ dữ liệu test dùng bộ dữ liệu đã thực hiện oversampling .....	82

Hình 4.31. Mô phỏng bộ dữ liệu Credit Card Fraud Detection ban đầu.....	83
Hình 4.32. Tổng quan giá trị của các thuộc tính .....	84
Hình 4.33. Đồ thị biểu diễn độ chênh lệch trong giao dịch bình thường và bất thường của thuộc tính Class .....	85
Hình 4.34. Phân phối xác suất số tiền giao dịch .....	86
Hình 4.35. Phân phối xác suất thời gian giao dịch.....	86
Hình 4.36. Kết quả sau khi thực hiện Splitting data đối với bộ dữ liệu.....	88
Hình 4.37. Kết quả sau khi thực hiện undersampling.....	89
Hình 4.38. Biểu đồ phân phối các nhãn của thuộc tính lớp trong bộ dữ liệu mới ...	90
Hình 4.39. Ma trận tương quan đối với dữ liệu mất cân bằng .....	91
Hình 4.40. Ma trận tương quan đối với bộ dữ liệu được xử lý cân bằng sử dụng undersampling .....	92
Hình 4.41. Ma trận tương quan đối với bộ dữ liệu được xử lý cân bằng sử dụng oversampling .....	92
Hình 4.42. Ma trận tương quan đối với bộ dữ liệu được xử lý cân bằng sử dụng phương pháp kết hợp SMOTE và ENN .....	93
Hình 4.43. Tương quan giữa các thuộc tính V với thuộc tính lớp .....	93
Hình 4.44. Tương quan giữa các thuộc tính V với thuộc tính lớp .....	93
Hình 4.45. Biểu đồ phân phối thuộc tính V14, V12, V10 của bộ dữ liệu dùng Undersampling .....	95
Hình 4.46. Biểu đồ hộp của các thuộc tính V14, V12, V10 sau khi loại bỏ các giá trị ngoại vi của bộ dữ liệu dùng Undersampling .....	96
Hình 4.47. Biểu đồ phân phối thuộc tính V14, V12, V10 của bộ dữ liệu dùng Oversampling .....	97
Hình 4.48. Biểu đồ hộp của các thuộc tính V14, V12, V10 sau khi loại bỏ các giá trị ngoại vi của bộ dữ liệu dùng Oversampling .....	98
Hình 4.49. Biểu đồ phân phối thuộc tính V14, V12, V10 của bộ dữ liệu dùng SMOTE + ENN .....	98

Hình 4.50. Biểu đồ hộp của các thuộc tính V14, V12, V10 sau khi loại bỏ các giá trị ngoại vi của bộ dữ liệu dùng SMOTE + ENN .....	99
Hình 4.51. Ma trận nhầm lẫn trên tập dữ liệu test của bộ dữ liệu chưa xử lý cân bằng .....	101
Hình 4.52. Ma trận nhầm lẫn trên tập dữ liệu test dùng Oversampling .....	103
Hình 4.53. Ma trận nhầm lẫn trên tập dữ liệu test dùng Undersampling .....	105
Hình 4.54. Ma trận nhầm lẫn trên tập dữ liệu test dùng SMOTE + ENN .....	107
Hình 4.55. Ma trận nhầm lẫn trên tập dữ liệu test của bộ dữ liệu chưa xử lý cân bằng .....	109
Hình 4.56. Ma trận nhầm lẫn trên tập dữ liệu test dùng Oversampling .....	111
Hình 4.57. Ma trận nhầm lẫn trên tập dữ liệu test dùng Undersampling .....	113
Hình 4.58. Ma trận nhầm lẫn trên tập dữ liệu test dùng SMOTE + ENN .....	115

## **DANH MỤC BẢNG BIỂU**

Bảng 3.1. Mô tả các thuộc tính của bộ dữ liệu Paysim.....	37
Bảng 3.2. Mô tả các thuộc tính của bộ dữ liệu Credit Card Fraud Detection.....	38
Bảng 4.1. Kiểu dữ liệu của các thuộc tính trong bộ dữ liệu Paysim.....	45
Bảng 4.2. Thống kê số lượng giao dịch bất thường và bình thường ở mỗi loại giao dịch.....	48
Bảng 4.3. Số lượng giao dịch bình thường và bất thường theo thuộc tính isFlaggedFraud của mỗi giao dịch .....	50
Bảng 4.4. Kết quả truy vấn theo giả thuyết d.....	53
Bảng 4.5. Thông tin truy vấn và kết quả trả về khi tìm các tài khoản thương gia có thực hiện một trong các giao dịch CASH_IN, CASH_OUT, PAYMENT.....	54
Bảng 4.6. Kết quả khi thực hiện tìm các giao dịch bất thườngs có tài khoản nguồn của giao dịch transfer là tài khoản nguồn của giao dịch cash_out .....	55
Bảng 4.7. Tên và kiểu dữ liệu của các thuộc tính sau khi thực hiện tiền xử lý dữ liệu .....	59
Bảng 4.8. Thống kê số lượng của mỗi giá trị ở thuộc tính isFraud theo bộ dữ liệu .	73
Bảng 4.9. Kết quả sau khi chạy thuật toán Random forest đối với bộ dữ liệu chưa cân bằng .....	73
Bảng 4.10. Thống kê số lượng của mỗi giá trị ở thuộc tính isFraud theo bộ dữ liệu .....	75
Bảng 4.11. Kết quả sau khi chạy thuật toán Random forest đối với bộ dữ liệu đã xử lý cân bằng dùng Oversampling.....	76
Bảng 4.12. Thống kê số lượng giao dịch của mỗi giá trị ở thuộc tính isFraud theo bộ dữ liệu.....	77
Bảng 4.13. Kết quả sau khi chạy thuật toán Random forest đối với bộ dữ liệu đã xử lý cân bằng dùng Undersampling.....	77
Bảng 4.14. Thống kê số lượng giao dịch của mỗi giá trị ở thuộc tính isFraud theo bộ dữ liệu.....	79

Bảng 4.15. Bảng kết quả sau khi chạy thuật toán EGB đối với bộ dữ liệu chưa thực hiện xử lý cân bằng .....	80
Bảng 4.16. Thống kê số lượng giao dịch của mỗi giá trị ở thuộc tính isFraud theo bộ dữ liệu.....	81
Bảng 4.17. Bảng kết quả sau khi chạy thuật toán EGB đối với bộ dữ liệu được xử lý dùng oversampling .....	82
Bảng 4.18. Số giao dịch của mỗi giá trị theo thuộc tính lớp.....	85
Bảng 4.19. Kết quả khi chia tỉ lệ các thuộc tính .....	87
Bảng 4.20. Kết quả sau khi thực hiện xóa các giá trị ngoại vi của các thuộc tính V14, V12, V10 của bộ dữ liệu dùng Undersampling .....	96
Bảng 4.22. Kết quả sau khi thực hiện xóa các giá trị ngoại vi của các thuộc tính V14, V12, V10 của bộ dữ liệu dùng SMOTE + ENN .....	98
Bảng 4.23. Thống kê số lượng của mỗi giá trị ở thuộc tính Class theo bộ dữ liệu.	100
Bảng 4.24. Hình kết quả sau khi chạy thuật toán Random forest đối với bộ dữ liệu chưa cân bằng.....	100
Bảng 4.25. Thống kê số lượng của mỗi giá trị ở thuộc tính Class .....	102
Bảng 4.26. Kết quả sau khi chạy thuật toán Random forest đối với bộ dữ liệu đã xử lý cân bằng dùng Oversampling.....	102
Bảng 4.27. Thống kê số lượng giao dịch của mỗi giá trị ở thuộc tính Class .....	104
Bảng 4.28. Hình kết quả sau khi chạy thuật toán Random forest đối với bộ dữ liệu đã xử lý cân bằng dùng Undersampling .....	104
Bảng 4.29. Thống kê số lượng giao dịch của mỗi giá trị ở thuộc tính Class .....	106
Bảng 4.30. Kết quả sau khi chạy thuật toán Random forest đối với bộ dữ liệu đã xử lý cân bằng dùng SMOTE + ENN .....	106
Bảng 4.31. Thống kê số lượng của mỗi giá trị ở thuộc tính Class theo bộ dữ liệu.	108
Bảng 4.32. Kết quả sau khi chạy thuật toán EGB đối với bộ dữ liệu chưa cân bằng .....	108
Bảng 4.33. Thống kê số lượng của mỗi giá trị ở thuộc tính Class .....	110

Bảng 4.34. Kết quả sau khi chạy thuật toán EGB đối với bộ dữ liệu đã xử lý cân bằng dùng Oversampling .....	110
Bảng 4.35. Thống kê số lượng giao dịch của mỗi giá trị ở thuộc tính Class .....	112
Bảng 4.36. Kết quả sau khi chạy thuật toán EGB đối với bộ dữ liệu đã xử lý cân bằng dùng Undersampling .....	112
Bảng 4.37. Thống kê số lượng giao dịch của mỗi giá trị ở thuộc tính Class .....	114
Bảng 4.38. Kết quả sau khi chạy thuật toán EGB đối với bộ dữ liệu đã xử lý cân bằng dùng SMOTE + ENN .....	114

## DANH MỤC TỪ VIẾT TẮT, THUẬT NGỮ

<b>STT</b>	<b>Thuật ngữ</b>	<b>Từ viết tắt</b>	<b>Nghĩa/Nội dung</b>
1	Boolean		Kiểu dữ liệu true/false hoặc 1/0
2	Classification and Regression Trees	CART	Cây phân loại và hồi quy
3	Cross validation		Xác thực chéo
4	Data		Dữ liệu
5	Data minining		Khai phá dữ liệu
6	Deep learning		Học sâu
7	Edited Nearest Neighbours	ENN	Nhằm mục đích tăng khả năng khai quát hóa trình phân loại bằng cách loại bỏ các trường hợp nhiễu khỏi tập huấn luyện
8	Extreme Gradient Boosting	EGB	Tăng cường độ dốc cực cao
9	Gradient Boosting Machines	GBM	Máy tăng cường độ dốc
10	Insights		Cái nhìn sâu sắc, sự thật ngầm hiểu
11	Label		Nhãn
12	Machine learning	ML	Máy học
13	Outcome		Đầu ra
14	Test set		Tập dữ liệu dùng để kiểm thử
15	Train		Huấn luyện
16	Train set		Tập dữ liệu dùng để huấn luyện

## LỜI CAM ĐOAN

Nhóm tác giả xin cam đoan đây là công trình nghiên cứu khoa học của Nguyễn Hoàng Hiệp và Trần Hoàng Luân và được sự hướng dẫn của TS. Cao Thị Nhạn. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau và có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong luận văn còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc. Nếu phát hiện có bất kỳ sự bất thường nào tôi xin hoàn toàn chịu trách nhiệm về nội dung luận văn của mình.

Sinh viên: Nguyễn Hoàng Hiệp

Trần Hoàng Luân

## TÓM TẮT KHÓA LUẬN

Đề tài khóa luận: “Nghiên cứu mô hình phát hiện bất thường trong giao dịch tài chính” là đề tài hướng đến việc nghiên cứu và chỉ ra các mô hình áp dụng trong việc phát hiện các bất thường trong giao dịch tài chính. Phát hiện bất thường là một vấn đề đầy thách thức tuy chỉ chiếm tỉ lệ thấp nhưng có thể gây ra nhiều tổn thất lớn. Hiện tại, tội phạm rất tinh vi, khi các kế hoạch lừa đảo thông thường không thành công, những kẻ lừa đảo đã học cách thay đổi chiến thuật bằng nhiều hành vi thông minh hơn.

Đề tài sử dụng machine learning trong việc sử dụng các thuật toán xử lý các bộ dữ liệu lớn với nhiều biến số và giúp tìm ra các mối tương quan giữa các hành vi người dùng và các hành động bất thường so với việc sử dụng các hệ thống dựa trên quy tắc khi quá trình xử lý dữ liệu chậm hơn và thực hiện khá nhiều các thao tác thủ công. Bằng việc sử dụng machine learning, đề tài nhằm đưa ra các thao tác thu thập, xử lý dữ liệu để tiến hành đưa ra các bộ huấn luyện và kiểm tra từ đó tạo ra các mô hình thích hợp. Cuối cùng so sánh đánh giá và chọn ra mô hình phù hợp trong việc phát hiện bất thường trong giao dịch. Mở rộng sẽ tìm hiểu các kỹ thuật nâng cao độ chính xác cho các thuật toán.

Sau khi tìm hiểu, khảo sát và nghiên cứu các bài báo khoa học liên quan, nhóm tác giả đưa ra hướng thực hiện cho đề tài như sau:

- Xử lý dữ liệu mất cân bằng cho các bộ dữ liệu bằng việc áp dụng các kỹ thuật phù hợp: Oversampling, Undersampling và Combined class methods SMOTE + ENN (Phương pháp kết hợp).
- Nghiên cứu và chạy các thuật toán phân lớp: Random forest, multi-layer perceptron, extreme gradient boosting.
- Tìm hiểu các kỹ thuật nâng cao độ chính xác của kết quả như: Gradient descent, back-propagation.
- Platform sử dụng: Colab Google.
- Ngôn ngữ lập trình: Python.

## CHƯƠNG 1: GIỚI THIỆU

### 1.1. Đặt vấn đề

Ngành dịch vụ tài chính và các giao dịch tài chính đang phải chịu những tổn thất và thiệt hại nặng nề liên quan đến bất thường. Nó gây ra những hậu quả nghiêm trọng trong ngành tài chính, chính phủ, cho cả doanh nghiệp và đối với người tiêu dùng thông thường.

Theo tạp chí Infosecurity, các bất thường tài chính gây thiệt hại cho nền kinh tế toàn cầu khoảng 3,2 nghìn tỉ bảng Anh vào năm 2018 [1]. Đối với một số doanh nghiệp, tổn thất do bất thường đạt hơn 10% tổng chi tiêu của họ. Những tổn thất lớn như vậy thúc đẩy các công ty tìm kiếm các giải pháp mới để ngăn chặn, phát hiện và loại bỏ các bất thường.

Nếu như trước đây, những tên tội phạm phải làm giả mã khách hàng, thì bây giờ để đánh cắp tiền thì điều cần thiết chính là chuyển sang đánh cắp mật khẩu tài khoản của người dùng. Theo chiến lược nghiên cứu của Javelin, phải mất hơn 40 ngày để phát hiện bất thường đối với các tổ chức tài chính chính thống [2]. Bất thường cũng tác động đến các ngân hàng cung cấp dịch vụ thanh toán trực tuyến. Chẳng hạn, 20% khách hàng sẽ thay đổi ngân hàng của họ sau khi trải qua các vụ lừa đảo [3].

Tội phạm mạng ngày càng thông minh hơn và họ đang tận dụng những tiến bộ trong công nghệ vì lợi ích của họ. Các ngân hàng và tổ chức tài chính không có lựa chọn nào khác ngoài việc thắt chặt bảo vệ và phát triển khả năng của chính họ nhanh hơn.

Theo truyền thống, các ngân hàng và tổ chức tài chính đã tiếp cận phát hiện bất thường bằng các thủ tục thủ công hoặc các giải pháp dựa trên các quy tắc, vốn bị hạn chế trong thành công của họ. Cách tiếp cận dựa trên quy tắc có nghĩa là một bộ tiêu chí phức tạp để gắn cờ các giao dịch đáng ngờ phải được thiết lập và xem xét thủ công.

Mặc dù điều này có thể có hiệu quả trong việc phát hiện ra sự bất thường phù hợp với các mẫu đã biết, nhưng nó không có khả năng phát hiện bất thường theo các mẫu dữ liệu mới hoặc chưa biết. Điều này mang lại cho các tội phạm động lực để

phát triển các kỹ thuật tinh vi hơn bao giờ hết để hiểu hết các quy tắc và chính những tội phạm cũng đang tận dụng các công nghệ mới để đạt được điều này.

Vì thế một thách thức đặt ra cho các tổ chức tài chính, ngân hàng, doanh nghiệp là cần phải sử dụng phương thức khác trong việc phát hiện bất thường tài chính một cách tốt hơn và hiệu quả hơn. Khi đó, các phương pháp máy học (machine learning) cho kết quả tốt.

Khối lượng lớn dữ liệu giao dịch và khách hàng có sẵn trong lĩnh vực tài chính được sử dụng để tìm ra những quy luật hay bất thường trong giao dịch. Từ đó, có thể tự động hóa việc phân tích các hành vi của khách hàng, tiên đoán các bất thường trong giao dịch, từ đó mở ra các cách thức nhằm xử lý bất thường theo thời gian thực.

Theo từ điển Cambridge Dictionary, gian lận hay sự bất thường trong tài chính là “tội ác của việc kiếm tiền bằng cách lừa đảo người khác”. Điều này đã xảy ra từ rất lâu, kể từ khi con người bắt đầu biết trao đổi hàng hóa và dịch vụ, thì đã có sẵn một nguy cơ là một trong hai bên sẽ lừa đảo đối phương. Và luôn có nguy cơ có thêm một bên thứ ba lừa đảo cả người bán và người mua. Với sự phát triển và mở rộng của thương mại điện tử, gian lận đã xuất hiện trên các hình thức mới và trở nên mạnh mẽ hơn bao giờ hết. Khi quy mô của mua sắm điện tử, ngân hàng trực tuyến và bảo hiểm trực tuyến tăng lên, những kẻ lừa đảo đã tận dụng tối đa mọi điểm yếu trong các hệ thống mà họ có thể tìm thấy. Điều này thường xuyên xảy ra, trước khi các chuyên gia có thể vá lỗi một hệ thống, các dữ liệu nhạy cảm đã bị đánh cắp và hàng triệu đô la cũng mất theo. Gian lận đã trở thành một vấn đề lớn và việc bỏ ra một khoản tiền lớn để ngăn chặn và phát hiện gian lận mà các nhà bán lẻ thương mại điện tử ở toàn cầu đã và đang thực hiện.

Ngăn chặn, phát hiện và loại bỏ bất thường là một số mối quan tâm chính của ngành thương mại điện tử và ngân hàng hiện nay, và một trong những phương pháp được lựa chọn để giải quyết vấn đề là áp dụng các thuật toán machine learning.

Tóm lại, machine learning (ML) là khoa học tạo và áp dụng các thuật toán có khả năng học hỏi từ dữ liệu quá khứ. Machine learning có thể chống lại bất thường

tài chính bằng cách sử dụng dữ liệu lớn tốt hơn và nhanh hơn con người từng có thể thực hiện.

Ưu điểm của việc sử dụng machine learning trong việc phát hiện các giao dịch bất thường là:

- Machine learning thực hiện hiệu quả hơn con người với lượng dữ liệu lớn, đảm bảo hoạt động xuyên suốt.
- ML đánh bại các hệ thống phát hiện bất thường truyền thống (rule-based system) [3] [4].

Như vậy, nghiên cứu để phát hiện bất thường trong giao dịch tài chính có ý nghĩa ứng dụng quan trọng trong thực tế. Do vậy, nhóm quyết định thực hiện đề tài nghiên cứu và tìm hiểu việc áp dụng machine learning trong bài toán phát hiện giao dịch bất thường.

## 1.2. Mục tiêu luận văn

Nghiên cứu, tìm hiểu các thuật data mining và machine learning trong việc phát hiện bất thường trong giao dịch tài chính. Đề tài tiến hành nghiên cứu thực hiện các thuật toán phát hiện bất thường kết quả trên bộ dữ liệu giao dịch tài chính. Sau đó tiến hành chạy thử nghiệm để so sánh, đánh giá kết quả đạt được. Cụ thể, nhóm thực hiện theo trình tự sau:

- Trước hết, nhóm tìm hiểu các công trình nghiên cứu liên quan của các tác giả khác. Tiến hành các thực nghiệm tương tự để hiểu rõ thuật toán.
- Tìm hiểu và chọn dữ liệu thực nghiệm: Dựa trên các bài báo liên quan về phát hiện các giao dịch tài chính bất thường của các tác giả khác, nhằm hiểu rõ thuật toán và áp dụng vào thực nghiệm, sử dụng hai bộ dữ liệu: Credit Card Fraud Detection [5] và bộ Paysim [6] được giới thiệu tại Hội nghị chuyên đề lần thứ 28 European Modeling và Simulation Symposium-EMSS, Larnaca, Cyprus năm 2016 [7].
- Áp dụng một số thuật toán trong machine learning để đưa ra đánh giá, nhận xét.
- Ngoài ra, còn tìm hiểu các kỹ thuật đánh giá phương pháp máy học.

- Mở rộng: Thực hiện mô hình đã chọn với các bộ dữ liệu tương tự. Đánh giá kết quả thu được khi sử dụng mô hình trên nhiều tập dữ liệu khác nhau.

### 1.3. Đối tượng, phạm vi

#### 1.3.1. Đối tượng

- Dữ liệu nghiên cứu là bộ dữ liệu được thu thập từ nhiều nguồn phục vụ cho mục đích nghiên cứu.
- Các kỹ thuật khai phá dữ liệu, công cụ khai phá dữ liệu và các module lập trình trong khai phá dữ liệu.

#### 1.3.2. Phạm vi

- Bộ dữ liệu về các giao dịch tài chính.
- Các kỹ thuật, thuật toán trong data mining và machine learning.
- Platform: Google Colab.
- Ngôn ngữ lập trình: Python.
- Thư viện: Numpy, Pandas, Matplotlib, Sklearn, Tensorflow, Keras.

### 1.4. Công cụ sử dụng

Trong quá trình thực hiện đề tài, nhóm đã sử dụng một số công cụ phục vụ cho việc tìm hiểu và xây dựng đề tài, bao gồm:

- Google Colab: hỗ trợ thực hành nghiên cứu chạy các thuật toán.
- Python: Ngôn ngữ sử dụng để thực hiện các thuật toán machine learning.

Tất cả các phần mềm và công cụ trên được nhóm cài đặt và sử dụng trên hệ điều hành Windows 10.

### 1.5. Bô cục báo cáo

#### *Chương 1: Giới thiệu*

Giới thiệu tổng quan về đề tài. Nội dung chương 1 bao gồm: Đặt vấn đề và lý do chọn đề tài, giới thiệu về phát hiện bất thường và lý do sử dụng machine learning trong phát hiện bất thường, mục tiêu và phạm vi nghiên cứu, công cụ xây dựng hệ thống và bô cục của báo cáo.

#### *Chương 2: Cơ sở lý thuyết*

Giới thiệu và đưa lý do chọn các công cụ, ngôn ngữ lập trình sử dụng trong đề tài.  
Trình bày cơ sở lý thuyết của các mô hình, thuật toán, và các cải tiến đã được sử dụng trong đề tài.

*Chương 3: Mô hình phát hiện bất thường trong giao dịch tài chính*

Đề xuất mô hình phát hiện bất thường trong giao dịch tài chính. Chọn lựa các tập dữ liệu cho thực nghiệm và xử lý tiền dữ liệu.

*Chương 4: Thực nghiệm*

Trình bày kết quả áp dụng các mô hình đề xuất, đưa ra nhận xét.

*Chương 5: Kết luận*

Tóm tắt lại những kết quả mà nhóm đã đạt được khi thực hiện đề tài, và định hướng phát triển cho đề tài.

## CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

### 2.1. Quy trình phát hiện bất thường trong giao dịch tài chính



Hình 2.1. Quy trình phát hiện bất thường trong giao dịch tài chính<sup>1</sup>

Trong giao dịch tài chính, quy trình phát triển bất thường được minh họa như trong hình 2.1, bao gồm 4 bước.

Quá trình phát hiện bất thường bằng cách sử dụng machine learning bắt đầu bằng việc thu thập dữ liệu. Sau đó, dựa trên dữ liệu học để tìm ra mô hình machine learning nhằm dự đoán giao dịch có phải là giao dịch bất thường.

Trích xuất dữ liệu (Extract data): Nói chung, dữ liệu được chia thành ba phân đoạn khác nhau – train set – test set – cross-validation. Thuật toán được dùng để train cho trên bộ dữ liệu huấn luyện, độ chính xác đo bằng cách sử dụng cross-validation. Sau cùng sử dụng bộ liệu kiểm thử để kiểm tra mô hình có dự đoán tốt đối với các dữ liệu chưa từng nhìn thấy.

Cung cấp bộ đào tạo: Ứng dụng chính của machine learning được sử dụng trong phát hiện bất thường là dự đoán. Dự đoán giá trị của một số đầu ra (trong trường hợp bài toán phát hiện bất thường thì là giá trị nhị phân (Boolean) là đúng nếu giao dịch được cho là bất thường và sai là trường hợp giao dịch bình thường) được đưa ra một số giá trị đầu vào (ví dụ: Thời gian thực hiện giao dịch, số tiền giao dịch, loại giao dịch, ...). Dữ liệu được sử dụng để training các mô hình machine learning bao gồm các bản ghi với các giá trị đầu ra cho các giá trị đầu vào khác nhau. Các bản ghi thường được lấy từ dữ liệu lịch sử.

<sup>1</sup> Nguồn ảnh: <https://www.intellias.com/how-to-use-machine-learning-in-fraud-detection/>

Xây dựng mô hình (Building Models): Xây dựng mô hình là một bước thiết yếu để dự đoán sự bất thường hoặc bất thường trong các bộ dữ liệu. Xác định cách đưa ra dự đoán đó dựa trên các ví dụ trước về dữ liệu đầu vào và đầu ra.

## 2.2. Học máy (Machine learning)

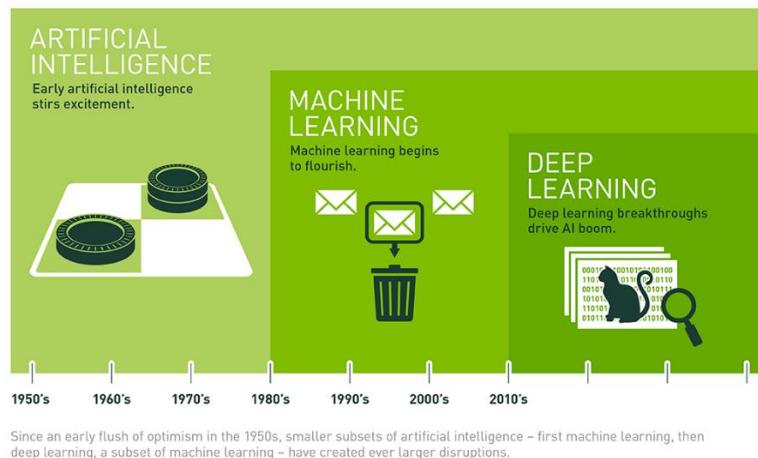
### 2.2.1. Giới thiệu

Những năm gần đây, Machine Learning (học máy) nổi lên như một bằng chứng của cuộc cách mạng công nghiệp lần thứ tư. Trí tuệ nhân tạo đang len lỏi vào mọi lĩnh vực trong đời sống xã hội. Xe tự hành của Google và Tesla, hệ thống tự đính thông tin vào khuôn mặt trong ảnh của Facebook, trợ lý ảo Siri của Apple, hệ thống gợi ý sản phẩm của Amazon, hệ thống gợi ý phim của Netflix, máy chơi cờ vây AlphaGo của Google DeepMind, ..., là những ứng dụng nổi bật của AI/Machine Learning.

### 2.2.2. Máy học

Máy học (Machine Learning) là một tập con của Artificial Intelligence (AI). Theo định nghĩa của Wikipedia, *Machine learning is the subfield of computer science that “gives computers the ability to learn without being explicitly programmed”*. Nói đơn giản, ML là một lĩnh vực của Khoa học máy tính, nó có khả năng tự học hỏi dựa trên dữ liệu đưa vào mà không cần phải được lập trình cụ thể.

Những năm gần đây, khi mà khả năng tính toán của các máy tính được nâng lên một tầm cao mới và lượng dữ liệu khổng lồ được thu thập bởi các hãng công nghệ lớn, ML đã tiến thêm một bước dài và một lĩnh vực mới được ra đời gọi là Deep Learning (Học sâu). Deep Learning đã giúp máy tính thực thi những việc tưởng chừng như không thể vào 10 năm trước, ví dụ như phân loại cả ngàn vật thể khác nhau trong các bức ảnh, tự tạo chú thích cho ảnh, bắt chước giọng nói và chữ viết của con người, giao tiếp với con người, hay thậm chí cả sáng tác văn hay âm nhạc. Hình 2.2 minh họa các giai đoạn phát triển của machine learning.



Hình 2.2. Quá trình phát triển của Machine learning<sup>2</sup>

### 2.2.3. Ứng dụng của machine learning

Machine learning có ứng dụng trong mọi lĩnh vực, ngành nghề. Đặc biệt là những ngành hiện đang phải làm việc với lượng dữ liệu lớn, nhằm giúp các tổ chức vận hành hiệu quả hơn, đồng thời tạo lợi thế cạnh tranh so với các đối thủ. Có thể liệt kê một số ví dụ sau:

**Các dịch vụ tài chính:** Ngân hàng và những doanh nghiệp hoạt động trong lĩnh vực tài chính sử dụng công nghệ Machine Learning với hai mục đích chính: xác định Insights<sup>3</sup> trong dữ liệu và ngăn chặn lừa đảo. Insights sẽ biết được các cơ hội đầu tư hoặc thông báo đến nhà đầu tư thời điểm giao dịch hợp lý. Thông qua các kỹ thuật khai thác dữ liệu, tổ chức có thể tìm được những khách hàng đang có hồ sơ rủi ro cao hoặc sử dụng giám sát mạng để chỉ rõ những tín hiệu lừa đảo.

**Tiếp thị và bán hàng:** Dựa trên hành vi mua hàng trước đây, các trang web sử dụng Machine Learning phân tích lịch sử mua hàng, từ đó giới thiệu những vật dụng mà khách hàng có thể sẽ quan tâm và yêu thích. Khả năng tiếp nhận dữ liệu,

<sup>2</sup> Nguồn ảnh: <https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/>

<sup>3</sup> Insights (hay còn được gọi là customer insights) là các “sự thật ngầm hiểu” của khách hàng giúp doanh nghiệp có thể thấu hiểu một cách sâu sắc mong muốn và nhu cầu của họ. Việc phân tích hành vi khách hàng có thể giúp doanh nghiệp liệt kê được những insights nói trên, và điều chỉnh các chiến lược kinh doanh phù hợp.

phân tích và sử dụng những dữ liệu đó để cá nhân hóa trải nghiệm mua sắm (hoặc thực hiện chiến dịch Marketing) chính là tương lai của ngành bán lẻ.

**Vận tải:** Phân tích dữ liệu để tối ưu hóa vận tải hàng hóa là trọng tâm trong ngành vận tải. Xác định khả năng tận dụng hiệu quả trên mỗi tuyến đường và dự đoán các vấn đề tiềm tàng để gia tăng lợi nhuận. Các chức năng phân tích dữ liệu và mô hình hóa của Machine Learning đóng vai trò quan trọng với các doanh nghiệp vận chuyển.

#### 2.2.4. Supervised Learning (Học có giám sát)

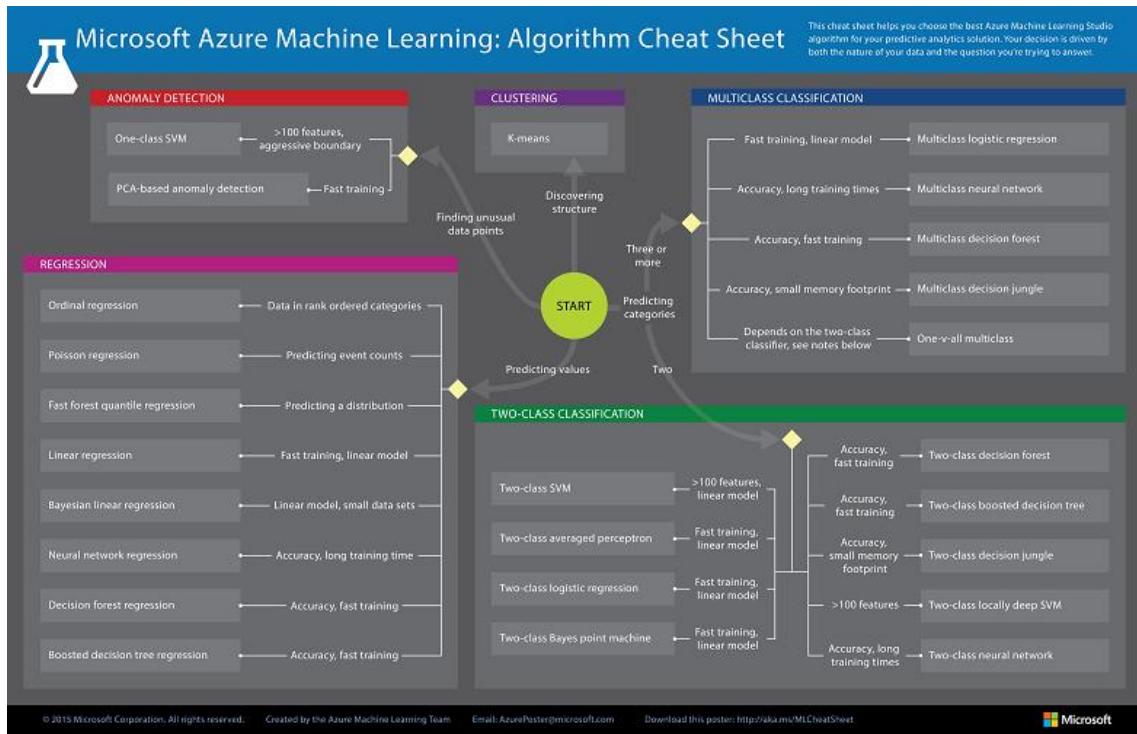
**Supervised learning** là thuật toán dự đoán đầu ra (outcome) của một dữ liệu mới (new input) dựa trên các cặp (input, outcome) đã biết từ trước. Cặp dữ liệu này còn được gọi là (data, label), tức (dữ liệu, nhãn). Supervised learning là nhóm phổ biến nhất trong các thuật toán Machine Learning.

**Ví dụ 1:** trong nhận dạng chữ viết tay, ta có ảnh của hàng nghìn ví dụ của mỗi chữ số được viết bởi nhiều người khác nhau. Chúng ta đưa các bức ảnh này vào trong một thuật toán và chỉ cho nó biết mỗi bức ảnh tương ứng với chữ số nào. Sau khi thuật toán tạo ra một mô hình, tức một hàm số mà đầu vào là một bức ảnh và đầu ra là một chữ số, khi nhận được một bức ảnh mới mà mô hình chưa nhìn thấy bao giờ, nó sẽ dự đoán bức ảnh đó chứa chữ số nào.

Thuật toán supervised learning còn được tiếp tục chia nhỏ ra thành hai loại chính:

- **Classification (Phân loại):** Một bài toán được gọi là classification nếu các label của input data được chia thành một số hữu hạn nhóm. Ví dụ: Gmail xác định xem một email có phải là spam hay không; các hãng tin dụng xác định xem một khách hàng có khả năng thanh toán nợ hay không. Ba ví dụ phía trên được chia vào loại này.
- **Regression (Hồi quy):** Nếu label không được chia thành các nhóm mà là một giá trị thực cụ thể. Ví dụ một căn nhà rộng  $x(m^2)$  và y phòng ngủ và cách trung tâm thành phố  $z(km)$  thì có tầm giá bao nhiêu?

**Một số thuật toán machine learning phổ biến:**



Hình 2.3. Một số thuật toán machine learning phổ biến<sup>4</sup>

### 2.3. Các thuật toán được sử dụng trong bài toán xác định bất thường trong giao dịch tài chính

Các thuật toán sử dụng để xác định bất thường trong giao dịch tài chính, được tìm hiểu trong khóa luận:

- Random forest.
- Extreme Gradient Boosting (EGB).
- Multi-layer Perceptron (Artificail Neural Network Algorithms).

#### 2.3.1. Thuật toán Rừng ngẫu nhiên (Random forest)

Random Forest (rừng ngẫu nhiên) là thuật toán học có giám sát (supervised learning),, được phát triển bởi Leo Breiman và Adele Cutler [8]. Thuật toán này có thể được sử dụng cho cả phân lớp và hồi quy. Random Forest là thuật toán linh hoạt và dễ sử dụng.

<sup>4</sup> Nguồn ảnh: <https://docs.microsoft.com/en-us/azure/machine-learning>

Random Forest được xây dựng dựa trên 3 thành phần chính là: CART, học toàn bộ, hội đồng các chuyên gia, kết hợp các mô hình, và tổng hợp bootstrap (bagging). Hình dưới đây thể hiện phương pháp phân lớp random forest.

### 2.3.1.1. Bootstrapping và Bagging

#### 2.3.1.1.1. Bootstrapping

Là một phương pháp rất nổi tiếng trong thống kê được giới thiệu bởi Bradley Efron vào năm 1979. Phương pháp này chủ yếu dùng để ước lượng lỗi chuẩn (standard errors), độ lệch (bias) và tính toán khoảng tin cậy (confidence interval) cho các tham số. Phương pháp này được thực hiện như sau: Từ một quần thể ban đầu lấy ra một mẫu  $L = (x_1, x_2, \dots, x_n)$  bao gồm  $n$  thành phần, tính toán các tham số mong muốn. Trong các bước tiếp theo lặp lại  $b$  lần việc tạo ra mẫu  $L_b$  cũng gồm  $n$  phần tử từ  $L$ . Bằng cách lấy lại mẫu với sự thay thế các thành phần trong mẫu ban đầu sau đó tính toán các tham số mong muốn.

#### 2.3.1.1.2. Bagging

Phương pháp này được xem như là một phương pháp tổng hợp kết quả có được từ các bootstrap. Tư tưởng chính của phương pháp này như sau: Cho một tập huấn luyện  $D = \{(x_i, y_i) : i=1, 2, \dots, n\}$  và giả sử chúng ta muốn có một dự đoán nào đó đối với biến  $x$ . Một mẫu gồm  $B$  tập dữ liệu, mỗi tập dữ liệu gồm  $n$  phần tử được chọn lựa ngẫu nhiên từ  $D$  với sự thay thế (giống như bootstrap). Do đó  $B = (D_1, D_2, \dots, D_B)$  trông giống như là một tập các tập huấn luyện được nhân bản.

Tập huấn một máy hoặc một mô hình đối với mỗi tập  $D_b$  ( $b=1, 2, \dots, B$ ) và lần lượt thu thập các kết quả dự báo có được trên mỗi tập  $D_b$ .

Kết quả tổng hợp cuối cùng được tính bằng cách trung bình hóa (regression) hoặc thông qua số phiếu bầu nhiều nhất (classification).

#### 2.3.1.2. Ưu điểm

Thuật toán Random Forest [9]:

- Có thể được sử dụng cho bài toán phân lớp (classification) và hồi quy (regression).
- Làm việc được với dữ liệu thiếu giá trị.

- Khi thuật toán có nhiều cây hơn, chúng ta có thể tránh được việc quá phù hợp (Overfitting<sup>5</sup>) với tập dữ liệu.
- Có thể tạo mô hình cho các giá trị phân loại.

### 2.3.1.3. Cách hoạt động

Random Forest hoạt động bằng cách đánh giá nhiều cây quyết định rồng ngẫu nhiên, và lấy ra kết quả được đánh giá tốt nhất trong số kết quả trả về.

Mã giả cho hoạt động của Random forest:

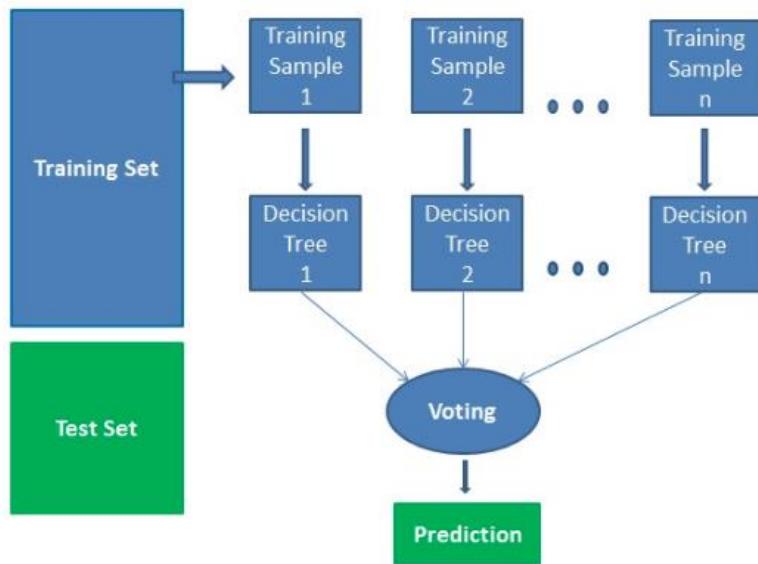
- Bước 1: Chọn ngẫu nhiên “k” features từ tập “m” features. Để ý  $k << m$ .
- Bước 2: Từ tập “k” features, tính toán ra nút (node) con theo nút tốt nhất vừa tìm được.
- Bước 3: Lặp lại bước 1 đến bước 3 cho đến khi đạt đến k nút.
- Bước 4: Lặp lại bước 1 đến bước 4 để tạo ra “n” cây.

Sau các bước trên, một random forest đã được tạo ra. Để biểu diễn dự đoán sử dụng random forest đã huấn luyện, sử dụng các bước sau đây [10]:

- Bước 1: Chọn các mẫu ngẫu nhiên từ tập dữ liệu đã cho.
- Bước 2: Thiết lập cây quyết định cho từng mẫu và nhận kết quả dự đoán từ mỗi quyết định cây.
- Bước 3: Hủy bỏ phiếu cho mỗi kết quả dự đoán.
- Bước 4: Chọn kết quả được dự đoán nhiều nhất là dự đoán cuối cùng.

---

<sup>5</sup> Một mô hình mô tốt là mô hình có tính tổng quát, tức mô tả được dữ liệu cả trong lẫn ngoài tập training. Mô hình chỉ mô tả tốt dữ liệu trong tập training được gọi là overfitting.



Hình 2.4. Quá trình hoạt động của thuật toán Random forest trong dự đoán<sup>6</sup>

### 2.3.2. Thuật toán tăng cường độ dốc cực cao (XGBoost)

#### 2.3.2.1. Giới thiệu

XGBoost, viết tắt của Extreme Extreme Boosting, được **Tianqi Chen** giới thiệu vào năm 2014 [11]. Kể từ khi được giới thiệu, XGBoost đã trở thành một trong những thuật toán học máy phổ biến. XGBoost thuộc về nhóm các thuật toán tăng cường và sử dụng khung tăng cường độ dốc (GBM) làm cốt lõi. GBM là một thư viện tăng cường tối ưu hóa.

XGBoost nổi tiếng là cung cấp các giải pháp tốt hơn các thuật toán học máy khác. Trên thực tế, kể từ khi ra đời, thuật toán này đã trở thành thuật toán học máy “tiên tiến” để đối phó với “dữ liệu có cấu trúc”. Đây là thuật toán mới nhằm giải quyết bài toán học có giám sát cho độ chính xác khá cao bên cạnh mô hình Deep learning.

Nếu Deep learning chỉ nhận đầu vào là dữ liệu thô dạng số (numeric) (ta thường phải chuyển đổi sang n-vector trong không gian số thực) thì XGBoost nhận đầu vào là bộ dữ liệu dạng bảng (tabular datasets) với mọi kích thước và dạng dữ liệu bao gồm cả categorical mà dạng dữ liệu này thường được tìm thấy nhiều hơn trong

<sup>6</sup> Nguồn ảnh: <https://viblo.asia/p/phan-lop-bang-random-forests-trong-python-djeZ1D2QKWz>

mô hình kinh doanh (business model), đây là một trong những lý do tại sao các cá nhân tham gia Kaggle thường sử dụng.

### 2.3.2.2. Lý do chọn XGBoost

#### a. Tốc độ thực hiện

XGBoost có tốc độ huấn luyện nhanh, có khả năng tăng quy mô (scale up) để tính toán song song trên nhiều server, có thể tăng tốc bằng cách sử dụng GPU, nhờ vậy mà Big Data không phải là vấn đề của mô hình này.

#### b. Hiệu suất mô hình

- XGBoost thống trị các bộ dữ liệu có cấu trúc hoặc dạng bảng về vấn đề mô hình dự báo phân loại và hồi quy. Bằng chứng là nó là thuật toán dành cho những người chiến thắng các thuộc thi trên nền tảng khoa học dữ liệu Kaggle<sup>7</sup>.
- Ví dụ, danh sách những người đứng thứ nhất, nhì và ba có sử dụng có tiêu đề XGBoost: [Machine Learning Challenge Winning Solutions](#).

#### c. Tính toán song song

- XGBoost có thể khai thác sức mạnh của tính đa lõi, có thể tính toán song song trên GPU và trên các mạng máy tính khiến việc huấn luyện trên các bộ dữ liệu rất lớn cũng khả thi.

#### d. Nhiều thông số tinh chỉnh mô hình

- XGBoost trong nội bộ có các tham số để xác thực chéo (cross-validation), chính quy hóa (regularization), các hàm mục tiêu do người dùng xác định, các giá trị bị thiếu (missing values), tham số cây (tree parameters), API tương thích scikit-learn, ...

### 2.3.2.3. Mô hình và thông số (Model and Parameters)

XGBoost được sử dụng cho các vấn đề học tập có giám sát, trong đó chúng tôi sử dụng dữ liệu đào tạo (có nhiều tính năng) “ $x_i$ ” để dự đoán một biến mục tiêu “ $y_i$ ”. Trước khi chúng ta tìm hiểu về cây cụ thể, chúng ta hãy bắt đầu bằng cách xem xét các yếu tố cơ bản trong học tập có giám sát:

---

<sup>7</sup> Trang chủ: <https://www.kaggle.com/>

$$\hat{y}_i = \sum_j \theta_j x_{ij} \quad (1)$$

Mô hình trong học tập có giám sát thường đề cập đến cấu trúc toán học mà theo đó dự đoán “ $y_i$ ” được thực hiện từ “ $x_i$ ” đầu vào. Một ví dụ phổ biến là một mô hình “tuyến tính”, trong đó dự đoán được đưa ra là (1), một tổ hợp tuyến tính của các tính năng đầu vào có trọng số. Giá trị dự đoán có thể có các cách hiểu khác nhau, tùy thuộc vào nhiệm vụ, tức là, hồi quy hoặc phân loại. Ví dụ, nó có thể được chuyển đổi logistic để có xác suất của lớp dương trong hồi quy logistic và nó cũng có thể được sử dụng làm điểm xếp hạng khi chúng ta muốn xếp hạng đầu ra.

Các tham số là phần không xác định mà chúng ta cần học từ dữ liệu. Trong các bài toán hồi quy tuyến tính, các tham số là các hệ số. Thông thường chúng ta sẽ sử dụng  $\theta$ . Để biểu thị các tham số (có nhiều tham số trong một mô hình).

#### 2.3.2.4. Hàm mất mát và chính quy hóa (Training Loss - Regularization)

Nhiệm vụ đào tạo mô hình hiệu quả việc tìm ra các tham số tốt nhất phù hợp nhất với dữ liệu đào tạo “ $x_i$ ” và nhãn “ $y_i$ ”. Để đào tạo mô hình, chúng ta cần xác định hàm mục tiêu (objective function) để đo mức độ phù hợp của mô hình với dữ liệu đào tạo.

Một đặc điểm nổi bật của các chức năng khách quan là chúng bao gồm hai phần: mất đào tạo và thời hạn chính quy:

$$\text{obj}(\theta) = L(\theta) + \Omega(\theta) \quad (2)$$

Trong đó  $L$  là hàm mất đào tạo và là thuật ngữ chính quy (regularization). Sự mất mát đào tạo đo lường mức độ dự đoán mô hình của chúng tôi đối với dữ liệu đào tạo. Một lựa chọn phổ biến của  $L$  là lỗi bình phương trung bình (mean squared error):

$$L(\theta) = \sum_i (y_i - \hat{y}_i)^2 \quad (3)$$

Một hàm mất mát thường được sử dụng khác là mất logistic, được sử dụng cho hồi quy logistic:

$$L(\theta) = \sum_i [y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i) \ln(1 + e^{\hat{y}_i})] \quad (4)$$

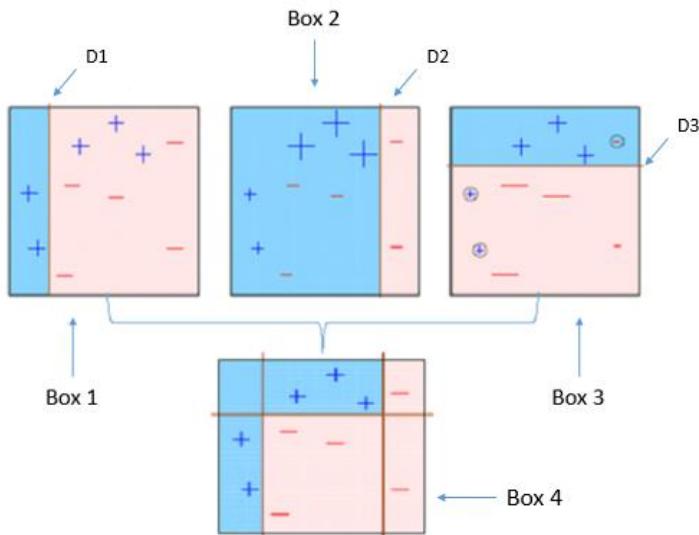
Thuật ngữ chính quy là những gì mọi người thường quên thêm vào. Thuật ngữ chính quy hóa kiểm soát sự phức tạp của mô hình, giúp chúng ta tránh bị quá mức(overfitting).

### 2.3.2.5. Cây quyết định kết hợp (Decision Tree Ensembles)

**Ý tưởng:** thay vì xây dựng một mô hình dự đoán (chẳng hạn descision tree) có độ chính xác tương đối, ta đi xây dựng nhiều mô hình dự đoán có độ chính xác kém hơn (weak learner) khi đi riêng lẻ nhưng lại cho độ chính xác cao khi kết hợp lại.

Có thể hình dung mỗi weak learner gồm học sinh yếu, khá, giỏi và thầy giáo. Trong đó, trọng số uy tín về kiến thức của thầy giáo sẽ là cao nhất và học sinh yếu sẽ là thấp nhất. Khi bạn đặt câu hỏi nào đó và cần những người này đưa ra kết luận, nếu nhiều người cùng có chung kết luận hoặc uy tín của những người đưa ra kết luận cao hơn tập thể thì ta có thể tin kết luận này là đúng.

Sau mỗi lần huấn luyện weak learner, mô hình sẽ tính lại trọng số cho các điểm dữ liệu đã bị phân lớp sai, để những lượt huấn luyện tiếp theo những điểm dữ liệu này sẽ có cơ hội nhiều hơn được phân lớp đúng. Dưới đây là mô hình dự đoán tổng quát.



Hình 2.5. Minh họa boosting

### Ví dụ minh họa:

4 Phân loại (trong 4 hộp), đang cố gắng phân loại “+” và “-” càng đồng nhất càng tốt.

- BOX 1: Trình phân loại đầu tiên (thường là gốc quyết định) tạo một đường thẳng đứng (tách) tại D1. Nó nói bất kỳ điều gì ở bên trái của D1 là “+” và bất cứ điều gì ở bên phải D1 là “-”. Tuy nhiên, bộ phân loại phân loại sai 3 điểm.
- BOX 2: Trình bày phân loại thứ 2 cho độ chính xác lớn hơn cho 3 điểm “+” phân loại sai (là các dấu “+” có kích thước lớn hơn) và tạo ra 1 đường thẳng đứng D2. Một lần nữa, bất cứ điều gì ở bên phải của D2 là “-”, bất cứ điều gì bên trái D2 là “+”. Tuy nhiên, nó vẫn phải sai lầm khi phân loại không chính xác 3 điểm.
- BOX 3: Bộ phân loại thứ ba mang lại độ chính xác lớn hơn cho ba điểm “-” phân loại sai và tạo ra một đường ngang tại D4. Tuy nhiên, trình phân loại này đã không phân loại chính xác các điểm (trong vòng tròn).
- BOX 4: Đây là sự kết hợp có trọng số của các phân loại yếu (Box 1,2 và 3). Như bạn có thể thấy, nó làm rất tốt trong việc phân loại tất cả các điểm một cách chính xác.

### **2.3.3. Thuật toán Multi-layer Perceptron (Artificial Neural Network Algorithms)**

#### **2.3.3.1. Giới thiệu về Perception, bài toán Perceptron và thuật toán Perceptron**

Perceptron là một thuật toán Classification cho trường hợp đơn giản nhất: chỉ có hai class (lớp) (*bài toán với chỉ hai class được gọi là binary classification*) và cũng chỉ hoạt động được trong một trường hợp rất cụ thể. Tuy nhiên, nó là nền tảng cho một mảng lớn quan trọng của Machine Learning là Neural Networks và sau này là Deep Learning.

Bài toán Perceptron được phát biểu như sau: *Cho hai class được gán nhãn, hãy tìm một đường phẳng sao cho toàn bộ các điểm thuộc class 1 nằm về 1 phía, toàn bộ các điểm thuộc class 2 nằm về phía còn lại của đường phẳng đó. Với giả định rằng tồn tại một đường phẳng như thế.*

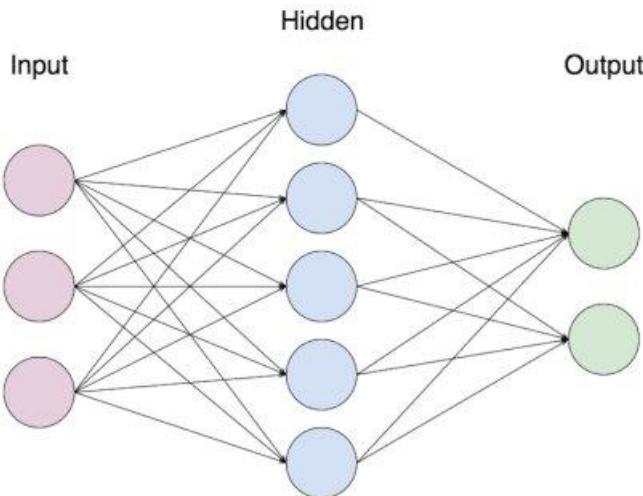
Nếu tồn tại một đường phẳng phân chia hai class thì ta gọi hai class đó là *linearly separable*. Các thuật toán classification tạo ra các boundary là các đường phẳng được gọi chung là phân loại tuyến tính (Linear Classifier).

### Thuật toán Perceptron (PLA)

1. Chọn ngẫu nhiên một vector hệ số  $w$  với các phần tử gần 0.
2. Duyệt ngẫu nhiên qua từng điểm dữ liệu  $x_i$ :
  - Nếu  $x_i$  được phân lớp đúng, tức  $\text{sgn}(w^T x_i) = y_i$ , không cần làm gì.
  - Nếu  $x_i$  bị misclassified, cập nhật  $w$  theo công thức:  $w = w + y_i x_i$
3. Kiểm tra xem có bao nhiêu điểm bị misclassified. Nếu không còn điểm nào, dừng thuật toán. Nếu còn quay lại bước 2.

#### 2.3.3.2. Multi-layer perceptron (MLP)

[12] Như tìm hiểu ở trên về perceptron – mô hình đơn giản nhất của một artificial neuron (tế bào thần kinh nhân tạo). Tiếp đến chúng ta sẽ tìm hiểu về một ví dụ của mạng neural network đơn giản được tạo ra từ các liên kết giữa các perceptron và là nền tảng để hiểu các mạng khác phức tạp hơn trong deep learning. Mạng neural network này có tên là Multi-layer Perceptron (MLP).



Hình 2.6. Multi-layer Perceptron<sup>8</sup>

<sup>8</sup> Nguồn ảnh: <https://dlapplications.github.io/2018-06-15-MLP/>

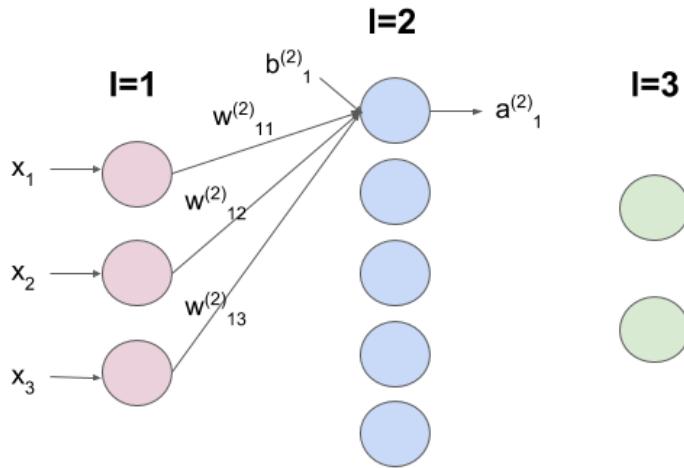
Gọi là Multi-layer Perceptron (perceptron nhiều lớp) bởi vì nó là tập hợp của các perceptron chia làm nhiều nhóm, mỗi nhóm tương ứng với một layer. Trong hình trên ta có một ANN với 3 lớp: Input layer (lớp đầu vào), Output layer (lớp đầu ra) và Hidden layer (lớp ẩn). Thông thường, khi giải quyết một bài toán ta chỉ quan tâm đến input và output của một model, do vậy trong MLP nói riêng và ANN nói chung, ngoài lớp Input và Output ra thì các lớp neuron ở giữa được gọi chung là Hidden (không quan tâm đến chúng).

Để dễ hình dung, người ta thường gọi lớp Input là lớp thứ nhất (kí hiệu  $l=1$ ) cứ thế tăng dần cho tới lớp output (trong Hình 1 là  $l=3$ ). Ta sẽ nói MLP này có số lớp là  $L=3$ . Mỗi hình tròn là biểu trưng cho một neuron (trong trường hợp này là perceptron) do đó đối với lớp thứ 2 ta nói nó có  $n=5$  neuron, lớp thứ 3 có  $n=2$  neuron. Đối với lớp input thì mỗi hình tròn chỉ là đại diện cho một giá trị input chứ không ta không tính toán gì ở đây và được gọi là input neuron. Để biểu diễn một MLP bằng công thức toán, người ta thường sử dụng các kí hiệu như sau (Các kí hiệu này cũng được sử dụng cho các mạng ANN khác):

- Các giá trị input thường được kí hiệu là  $x_1, x_2, x_3, \dots, x_{n_x}$  với MLP có  $n_x$  input.
- Đối với từng neuron, giá trị output của neuron thứ  $k$  của lớp thứ  $l$  được kí hiệu là  $a_k^l$ . Ví dụ output của neuron thứ 1 (tính từ trên xuống) của lớp thứ 2 sẽ là  $a_1^{(2)}$ , phía trên sẽ là số thứ tự lớp phía dưới là thứ tự neuron. Ngoài lớp Input ra thì ta có thể dễ dàng thấy là output của lớp này sẽ là input cho lớp kế tiếp.
- Như đã giới thiệu, mỗi mũi tên nối giữa các neuron của 2 lớp liền kề nhau đại diện cho một liên kết gửi - nhận thông tin với độ mạnh yếu được quyết định bởi một giá trị  $w$  (weight). Ta gọi giá trị  $w$  của liên kết giữa neuron thứ  $k$  ở lớp thứ  $l-1$  với neuron thứ  $j$  ở lớp kế tiếp  $l$  là  $w_{jk}^{(1)}$ . Chỗ này rất dễ nhầm lẫn nên các bạn chú ý, ví dụ liên kết giữa neuron thứ 1 ở lớp thứ 2 với neuron thứ 2 ở lớp thứ 3 sẽ viết là  $w_{21}^{(3)}$ , ở phía trên là thứ tự lớp của neuron nhận tín hiệu vào, phía dưới là thứ tự của mỗi neuron trong từng lớp. Số thứ tự của neuron nhận tín hiệu viết trước và neuron gửi tín hiệu viết sau.

- Cuối cùng, các giá trị output được kí hiệu  $y_1^{\wedge}, y_2^{\wedge}, \dots, y_{n_y}^{\wedge}$  với  $n_y$  là số output.

Từ input đến output:



Hình 2.7. Kí hiệu liên quan đến neuron thứ 1 của lớp thứ 2<sup>9</sup>

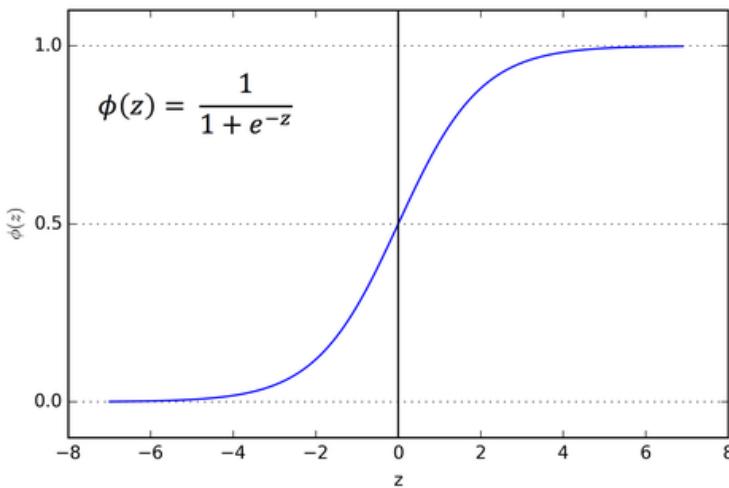
Sử dụng các kí hiệu như trên chúng ta sẽ biểu diễn quá trình tính toán từ input cho ra output của MLP 3 lớp trong Hình 2.9. Với lớp  $l = 1$  thì chỉ là lớp của các giá trị input nên ta không tính toán gì ở đây. Với lớp  $l = 2$  ta sẽ tính output của từng neuron. Bắt đầu với neuron thứ 1 của lớp  $l = 2$  như trong Hình 2.10, như đã tìm hiểu ở trên, đầu tiên chúng ta sẽ tính tổng ( $z$ , cách kí hiệu giống như với  $a$ ) của các input tới đã được điều chỉnh độ mạnh yếu nhờ các giá trị  $w$ :

$$z_1^{(2)} = w_0 + w_{11}^{(2)}x_1 + w_{12}^{(2)}x_2 + w_{13}^{(2)}x_3 = \sum_{k=1}^3 w_{1k}^{(2)}x_k + w_0 \quad (1)$$

Về giá trị  $w_0$  (gọi là bias, do đó còn được kí hiệu là  $b_1^{(2)}$  như Hình 2.10). Thường thì giá trị của bias là đặc trưng cho từng neuron. Ta đã biết nếu  $z$  vượt qua một ngưỡng nhất định thì neuron sẽ phát ra một output còn không thì sẽ không output gì cả. Để quyết định một neuron có output hay không thì trong Deep Learning người ta sử dụng một số hàm số được gọi chung là **Activation function**. Ở đây giả sử ta sử dụng một hàm số thông dụng khác là hàm Sigmoid như sau:

<sup>9</sup> Nguồn ảnh: <https://dlapplications.github.io/2018-06-15-MLP/>

$$\sigma(z) = \frac{1}{1 + e^{-z}} = \frac{e^z}{e^z + 1} \quad (2)$$



Hình 2.8. Đồ thị của hàm số sigmoid<sup>10</sup>

Nhìn đồ thị, nếu  $z$  nhỏ hơn một ngưỡng nhất định thì giá trị output của hàm sigmoid  $\sigma$  (đọc là sigma) sẽ gần với 0 và nếu  $z$  đủ lớn thì giá trị output sẽ gần với 1. Sử dụng hàm sigmoid này ta sẽ tính được giá trị output của neuron thứ 1 của lớp  $l = 2$  như sau:

$$a_1^{(2)} = \sigma(z_1^{(2)}) = \sigma\left(\sum_{k=1}^3 w_{1k}^{(l)} x_k + w_0\right) \quad (3)$$

Và ta có thể làm tương tự như vậy đối với các neuron tiếp theo trong lớp Hidden  $l = 2$  và lớp Output  $l = 3$ . Ta có thể thấy các tính toán trên giống hệt nhau cho các neuron trong cùng một lớp. Vì lý do đó mà ta có thể tính toán song song (parallel processing) nhiều neuron trong một lớp để tăng tốc độ. Điều đó dẫn tới việc trong DL GPU hay được sử dụng để tăng tốc độ tính toán (training + testing). Một CPU thông thường có khoảng vài đến vài chục core với xung nhịp (frequency) mỗi core lớn còn một GPU thông thường có khoảng vài trăm tới vài nghìn core với xung nhịp mỗi core nhỏ hơn. Với những tính toán đơn giản như trên thì không cần xung nhịp quá lớn do

<sup>10</sup> Nguồn ảnh: <https://dlapplications.github.io/2018-06-15-MLP/>

vậy GPU với số core gấp nhiều lần CPU sẽ giúp training nhanh hơn nhiều nhờ vào tính toán song song.

Để dễ hình dung về tính toán song song ta có thể biểu diễn các tính toán trên theo một dạng khác. Nếu bạn nào đã học đại số tuyến tính thì thay vì viết  $z_i$  là tổng của các  $w_i x_i$  thì ta có thể biểu diễn thành tích vô hướng của hai vector như sau:

$$z_1^{(2)} = [w_{11}^{(2)} \quad w_{12}^{(2)} \quad w_{13}^{(2)}] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + b_1^{(2)} = w_1^{(2)T} + b_1^{(2)} \quad (4)$$

Trong đó:  $w_1^{(2)} = \begin{bmatrix} w_{11}^{(2)} \\ w_{12}^{(2)} \\ w_{13}^{(2)} \end{bmatrix}$  và  $x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$  lần lượt là cách viết theo dạng vector gồm tất cả các weight của neuron thứ 1 thuộc lớp  $l = 2$  và tất cả input x. Áp dụng tương tự với các neuron khác trong lớp, ta có thể viết gộp lại cách tính các giá trị output của tất cả các neuron trong lớp  $l = 2$  như sau:

$$\begin{aligned} Z^{(2)} &= \begin{bmatrix} w_{11}^{(2)} & w_{12}^{(2)} & w_{13}^{(2)} \\ w_{21}^{(2)} & w_{22}^{(2)} & w_{23}^{(2)} \\ w_{31}^{(2)} & w_{32}^{(2)} & w_{33}^{(2)} \\ w_{41}^{(2)} & w_{42}^{(2)} & w_{43}^{(2)} \\ w_{51}^{(2)} & w_{52}^{(2)} & w_{53}^{(2)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{(2)} \\ b_2^{(2)} \\ b_3^{(2)} \\ b_4^{(2)} \\ b_5^{(2)} \end{bmatrix} b_1^{(2)} = \begin{bmatrix} -w_1^{(2)T} \\ -w_2^{(2)T} \\ -w_3^{(2)T} \\ -w_4^{(2)T} \\ -w_5^{(2)T} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{(2)} \\ b_2^{(2)} \\ b_3^{(2)} \\ b_4^{(2)} \\ b_5^{(2)} \end{bmatrix} \\ &= W^{(2)T} X + b^{(2)} \end{aligned} \quad (5)$$

Ở đây,  $W^{(2)}$  ( $W$  được viết hoa) được gọi là **weight matrix** (ma trận weight) và  $b^{(2)}$  là **bias vector** của lớp  $l = 2$ . Trong ví dụ ta đang xét thì  $W^{(2)}$  là ma trận có  $3 \times 5$  chiều (dimension), vector x có 3 chiều, vector b có 5 chiều (mỗi neuron có 1 bias) và vector z có 5 chiều. Có thể nhớ đơn giản số chiều của ma trận  $W^{(2)}$  là  $3 \times 5$  vì nó output ra một vector có chiều là 5 (lớp hidden  $l = 2$  có 5 neuron) từ một vector có chiều là 3 (có 3 input). Tương tự, đối với ma trận  $W^{(3)}$  của lớp Output sẽ có số chiều là  $5 \times 2$  như sau:

$$W^{(3)} = \begin{bmatrix} w_{11}^{(2)} & w_{21}^{(2)} \\ w_{12}^{(2)} & w_{22}^{(2)} \\ w_{13}^{(2)} & w_{23}^{(2)} \\ w_{14}^{(2)} & w_{24}^{(2)} \\ w_{15}^{(2)} & w_{25}^{(2)} \end{bmatrix} \quad (6)$$

Vì nó từ một input vector của 5 neuron lớp Hidden  $l = 2$  output ra một vector có chiều là 2 (2 neuron ở lớp Output). Lưu ý là input vector cho 2 neuron ở lớp Output không phải là  $z^{(2)}$  mà là  $a^{(2)}$  với:

$$a^{(2)} = \begin{bmatrix} \sigma(z_1^{(2)}) \\ \sigma(z_2^{(2)}) \\ \sigma(z_3^{(2)}) \\ \sigma(z_4^{(2)}) \\ \sigma(z_5^{(2)}) \end{bmatrix} = \sigma(z^{(2)}) = \sigma(W^{(2)T}x + b^{(2)}) \quad (7)$$

Tóm lại, toàn bộ các tính toán từ đầu vào cho tới đầu ra của một MLP 3 lớp như hình 1 có thể được tóm tắt như sau:

$$z^{(2)} = W^{(2)T}x + b^{(2)} \quad (8)$$

$$a^{(2)} = \sigma(z^{(2)}) \quad (9)$$

$$z^{(3)} = W^{(2)T}a^{(2)} + b^{(3)} \quad (10)$$

$$\hat{y} = a^{(3)} = f(z^{(3)}) \quad (11)$$

Chính vì cách tính toán theo một chiều từ đầu vào cho tới đầu ra như trên mà một MLP như đang xét còn được gọi là **Feed-forward Neural Network** (feedforward: truyền tới). Cần chú ý là đối với neuron ở lớp Output thì tùy vào output của bài toán mà sẽ chọn Activation function  $f$  cho phù hợp.

Vậy: Một *multilayer perceptron* (MLP) là việc implement nhiều layers perceptrons được kết nối liền kề nhau, tạo thành một bộ điều khiển (feedforward) thần kinh đơn giản. Mạng Feedforward là một mạng gồm một hay nhiều lớp neuron, trong đó các dây dẫn tín hiệu chỉ truyền theo một chiều từ input qua các lớp, cho đến

output. MLP này giúp ích trong những hàm phi tuyến tính mà perceptron đơn không thể thực hiện.

#### 2.4. Các phương pháp đánh giá một hệ thống phân lớp

Khi xây dựng một mô hình Machine learning, cần một phép đánh giá để xem mô hình sử dụng có hiệu quả không và để so sánh khả năng của các mô hình.

Hiệu năng của một mô hình thường được đánh giá dựa trên tập dữ liệu kiểm thử (test data). Cụ thể, giả sử đầu ra của mô hình khi đầu vào là tập kiểm thử được mô tả bởi vector  $y_{pred}$  là vector dự đoán đầu ra với mỗi phần tử là class được dự đoán của một điểm dữ liệu trong tập kiểm thử. Ta cần so sánh giữa vector dự đoán  $y_{pred}$  này với vector class thật của dữ liệu, được mô tả bởi vector  $y_{true}$ . Ví dụ với bài toán có 3 lớp dữ liệu được gán nhãn là 0, 1, 2. Trong bài toán thực tế, các class có thể có nhãn bất kỳ, không nhất thiết là số, và không nhất thiết bắt đầu từ 0. Chúng ta hãy tạm giả sử các class được đánh số từ 0 đến C-1 trong trường hợp có C lớp dữ liệu. Có 10 điểm dữ liệu trong tập kiểm thử với các nhãn thực sự được mô tả bởi  $y_{true} = [0, 0, 0, 0, 1, 1, 1, 2, 2, 2]$ . Giả sử bộ phân lớp chúng ta đang cần đánh giá dự đoán nhãn cho các điểm này là  $y_{pred} = [0, 1, 0, 2, 1, 1, 0, 2, 1, 2]$ .

Có rất nhiều cách đánh giá một mô hình phân lớp. Tùy vào những bài toán khác nhau mà sử dụng các phương pháp khác nhau. Các phương pháp thường được sử dụng là: accuracy score, confusion matrix, ROC curve, Area Under the Curve, Precision and Recall, F1 score, Top R error, etc.

##### Accuracy:

Cách đơn giản và hay được sử dụng nhất là accuracy (độ chính xác). Cách đánh giá này đơn giản tính tỉ lệ giữa số điểm được dự đoán đúng và tổng số điểm trong tập dữ liệu kiểm thử.

##### Confusion matrix:

Cách tính sử dụng accuracy như ở trên chỉ cho chúng ta biết được bao nhiêu phần trăm lượng dữ liệu được phân loại đúng mà không chỉ ra được cụ thể mỗi loại được phân loại như thế nào, lớp nào được phân loại đúng nhiều nhất, và dữ liệu thuộc

lớp nào thường bị phân loại nhầm vào lớp khác. Để có thể đánh giá được các giá trị này, chúng ta sử dụng một ma trận được gọi là ma trận tương quan (confusion matrix).

Về cơ bản, confusion matrix thể hiện có bao nhiêu điểm dữ liệu được dự đoán đúng, và bao nhiêu điểm dữ liệu cho kết quả dự đoán sai.

Total: 10	Predicted as: 0	Predicted as: 1	Predicted as: 2	
True: 0	2	1	1	4
True: 1	1	2	0	3
True: 2	0	1	2	3

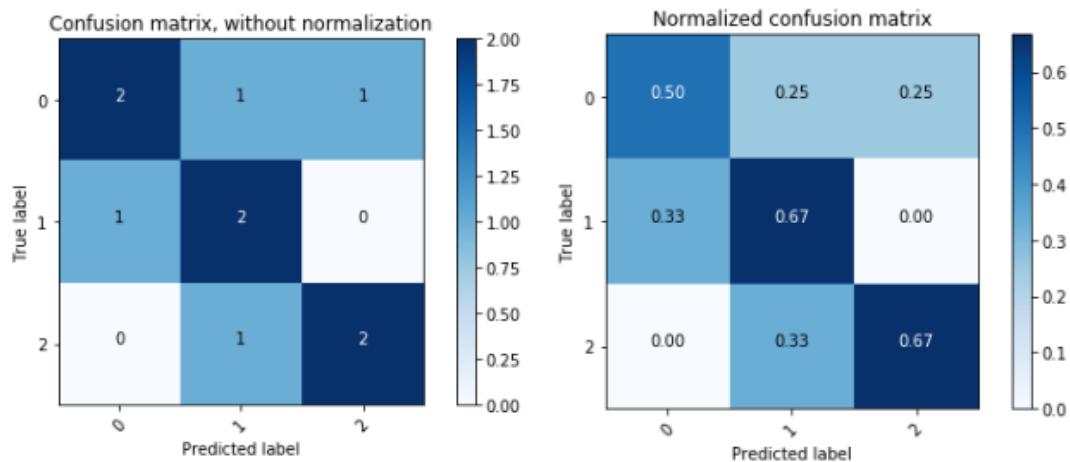
Hình 2.9. Minh họa confusion matrix

Có tổng cộng 10 điểm dữ liệu. Chúng ta xem xét ma trận tại vùng 3x3 trung tâm của bảng.

Ma trận thu được gọi là confusion matrix. Là một ma trận vuông với kích thước mỗi chiều bằng số lượng lớp dữ liệu. Giá trị tại hàng thứ i, cột thứ j là số lượng điểm lẻ ra thuộc vào class i nhưng lại được dự đoán là thuộc vào class j. Như vậy, nhìn vào hàng thứ nhất (0), ta có thể thấy được rằng trong số bốn điểm thực sự thuộc lớp 0, chỉ có hai điểm được phân loại đúng, hai điểm còn lại bị phân loại nhầm vào lớp 1 và lớp 2.

Các phần tử trên đường chéo của ma trận là số điểm được phân loại đúng của mỗi lớp dữ liệu. Từ đây có thể suy ra accuracy chính bằng tổng các phần tử trên đường chéo chia cho tổng các phần tử của toàn ma trận.

Confusion matrix thường được minh họa bằng màu sắc để có cái nhìn rõ ràng hơn.



Hình 2.10. Minh họa confusion matrix bảng kết quả

Với bài toán có nhiều lớp dữ liệu, cách biểu diễn bằng màu sẽ rất hữu ích. Các ô màu đậm thể hiện các giá trị cao. Một mô hình tốt sẽ cho một confusion matrix có các phần tử trên đường chéo chính có giá trị lớn, các phần tử còn lại có giá trị nhỏ. Nói cách khác, khi biểu diễn bằng màu sắc, đường chéo có màu càng đậm so với phần còn lại sẽ càng tốt. Từ hai hình trên ta thấy rằng confusion matrix đã chuẩn hóa mang nhiều thông tin hơn. Sự khác nhau được thấy ở ô trên cùng bên trái. Lớp dữ liệu 0 được phân loại không thực sự tốt nhưng trong ma trận tương quan không chuẩn hóa (Unnormalized confusion matrix), nó vẫn có màu đậm như hai ô còn lại trên đường chéo chính.

#### 2.4.1. True/False Positive/Negative

##### 2.4.1.1. True/False Positive/Negative

Cách đánh giá này thường được áp dụng cho các bài toán phân lớp có hai lớp dữ liệu. Cụ thể hơn, trong hai lớp dữ liệu này có một lớp nghiêm trọng hơn lớp kia và cần được dự đoán chính xác. Ví dụ, trong bài toán xác định có bệnh ung thư hay không thì việc không bị sót quan trọng hơn là việc chẩn đoán nhầm âm tính thành dương tính. Trong bài toán xác định có mìn dưới lòng đất hay không thì việc bỏ sót nghiêm trọng hơn việc báo động nhầm rất nhiều. Hay trong các bài toán email rác thì việc cho nhầm email quan trọng vào thùng rác nghiêm trọng hơn việc xác định một email rác là email thường.

Trong những bài toán này, người ta thường định nghĩa lớp dữ liệu quan trọng hơn cần được xác định đúng là lớp Positive (P-dương tính), lớp còn lại được gọi là Negative (N-âm tính). Ta định nghĩa True Positive (TP), False Positive (FP), True Negative (TN), False Negative (FN) dựa trên confusion matrix chưa chuẩn hóa như sau:

	Predicted as Positive	Predicted as Negative
Actual: Positive	True Positive (TP)	False Negative (FN)
Actual: Negative	False Positive (FP)	True Negative (TN)

Người ta thường quan tâm đến TPR, FNR, FPR, TNR (R-Rate) dựa trên normalized confusion matrix như sau:

	Predicted as Positive	Predicted as Negative
Actual: Positive	$TPR = TP/(TP+FN)$	$FNR = FN/(TP+FN)$
Actual: Negative	$FPR = FP/(FP+TN)$	$TNR = TN/(FP+TN)$

False Positive Rate còn được gọi là False Alarm Rate (tỉ lệ báo động nhầm), False Negative Rate còn được gọi là Miss Detection Rate (tỉ lệ bỏ sót). Trong bài toán dò mìn, thà báo nhầm còn hơn bỏ sót, tức là ta có thể chấp nhận False Alarm Rate cao để đạt được Miss Detection Rate thấp.

#### Chú ý:

- Việc biết một cột của confusion matrix này sẽ suy ra được cột còn lại vì tổng các hàng luôn bằng 1 và chỉ có hai lớp dữ liệu.
- Với các bài toán có nhiều lớp dữ liệu, ta có thể xây dựng bảng True/False Positive/Negative cho mỗi lớp nếu coi lớp đó là lớp Positive, các lớp còn lại gộp chung thành lớp Negative, giống như cách làm trong one-vs-rest.

#### 2.4.1.2. Receiver Operating Characteristic curve

Trong một số bài toán, việc tăng hay giảm FNR, FPR có thể được thực hiện bằng việc thay đổi một ngưỡng (threshold) nào đó. Lấy ví dụ khi ta sử dụng thuật toán Logistic Regression, đầu ra của mô hình có thể là các lớp cứng 0 hay 1, hoặc cũng có thể là các giá trị thể hiện xác suất để dữ liệu đầu vào thuộc vào lớp 1. Khi sử

dụng thư viện sklearn Logistic Regression, ta có thể lấy được các giá trị xác xuất này bằng phương thức predict\_proba(). Mặc định, ngưỡng được sử dụng là 0.5, tức là một điểm dữ liệu  $x$  sẽ được dự đoán rơi vào lớp 1 nếu giá trị predict\_proba( $x$ ) lớn hơn 0.5 và ngược lại.

Nếu bây giờ ta coi lớp 1 là lớp Positive, lớp 0 là lớp Negative, câu hỏi đặt ra là làm thế nào để tăng mức độ báo nhầm (FPR) để giảm mức độ bỏ sót (FNR)? Chú ý rằng tăng FNR đồng nghĩa với việc giảm TPR vì tổng của chúng luôn bằng 1.

Một kỹ thuật đơn giản là ta thay giá trị threshold từ 0.5 xuống một số nhỏ hơn. Chẳng hạn nếu chọn threshold = 0.3, thì mọi điểm được dự đoán có xác suất đầu ra lớn hơn 0.3 sẽ được dự đoán là thuộc lớp Positive. Nói cách khác, tỉ lệ các điểm được phân loại là Positive sẽ tăng lên, kéo theo cả False Positive Rate và True Positive Rate cùng tăng lên (cột thứ nhất trong ma trận tăng lên). Từ đây suy ra cả FNR và TNR đều giảm.

Ngược lại, nếu ta muốn bỏ sót còn hơn báo nhầm, tất nhiên là ở mức độ nào đó, như bài toán xác định email rác chẳng hạn, ta cần tăng threshold lên một số lớn hơn 0.5. Khi đó, hầu hết các điểm dữ liệu sẽ được dự đoán thuộc lớp 0, tức Negative, và cả TNF và FNR đều tăng lên, tức TPR và FPR giảm xuống

Như vậy, ứng với mỗi giá trị của threshold, ta sẽ thu được một cặp (FPR, TPR). Biểu diễn các điểm (FPR, TPR) trên đồ thị khi thay đổi threshold từ 0 tới 1 ta sẽ thu được một đường được gọi là Receiver Operating Characteristic curve hay ROC curve. (Chú ý rằng khoảng giá trị của threshold không nhất thiết từ 0 tới 1 trong các bài toán tổng quát. Khoảng giá trị này cần được đảm bảo có trường hợp TPR/FPR nhận giá trị lớn nhất hay nhỏ nhất mà nó có thể đạt được).

#### 2.4.1.3. Area Under the Curve

Dựa trên ROC curve, ta có thể chỉ ra rằng một mô hình có hiệu quả hay không. Một mô hình hiệu quả khi có FPR thấp và TPR cao, tức tồn tại một điểm trên ROC curve gần với điểm có tọa độ (0, 1) trên đồ thị (góc trên bên trái). Curve càng gần thì mô hình càng hiệu quả.

Có một thông số nữa dùng để đánh giá mà tôi đã sử dụng ở trên được gọi là Area Under the Curve hay AUC. Đại lượng này chính là diện tích nằm dưới ROC curve màu cam. Giá trị này là một số dương nhỏ hơn hoặc bằng 1. Giá trị này càng lớn thì mô hình càng tốt.

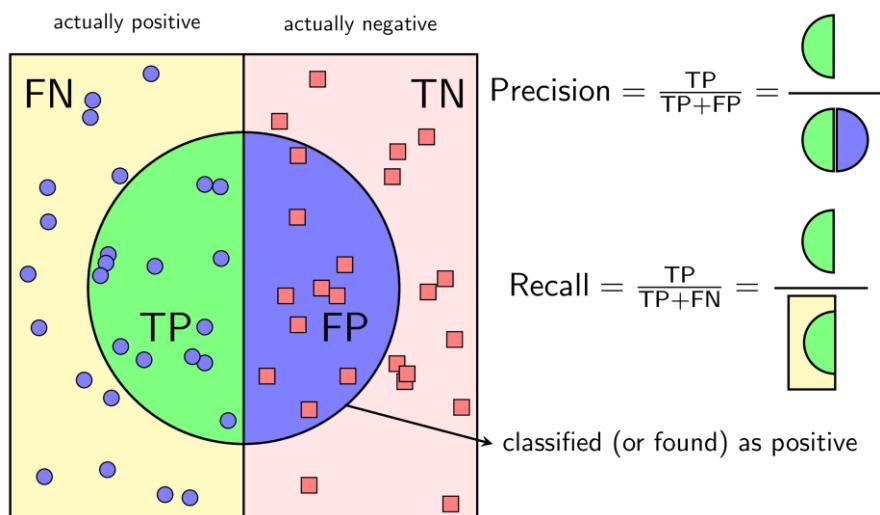
Chú ý: Cross validation cũng có thể được thực hiện bằng cách xác định ROC curve và AUC lên [validation set].

#### 2.4.1.4. Precision và Recall

##### Định nghĩa:

Với bài toán phân loại mà tập dữ liệu của các lớp là chênh lệch nhau rất nhiều, có một phép đo hiệu quả thường được sử dụng là Precision-Recall.

Trước hết xét bài toán phân loại nhị phân. Ta cũng coi một trong hai lớp là positive, lớp còn lại là negative.



Hình 2.11. Minh họa precision và recall

Với một cách xác định một lớp là positive, Precision được định nghĩa là tỉ lệ số điểm true positive trong số những điểm được phân loại là positive ( $TP+FP$ ).

Recall được định nghĩa là tỉ lệ số điểm true positive trong số những điểm thực sự là positive ( $TP+FN$ ).

Một cách toán học, Precision và Recall là hai phân số có tử số bằng nhau nhưng mẫu số khác nhau.

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

$$Recall = \frac{TP}{TP + EN} \quad (2)$$

Ta thấy được rằng TPR và Recall là hai đại lượng bằng nhau. Ngoài ra, cả Precision và Recall đều là các số không âm nhỏ hơn hoặc bằng một.

Precision cao đồng nghĩa với việc độ chính xác của các điểm tìm được là cao. Recall cao đồng nghĩa với việc True Positive Rate cao, tức tỉ lệ bỏ sót các điểm thực sự Positive là thấp.

### Precision-Recall curve và Average Precision

Tương tự như ROC curve, chúng ta cũng có thể đánh giá mô hình dựa trên việc thay đổi một ngưỡng và quan sát giá trị của Precision và Recall. Khái niệm Area Under the Curve (AUC) cũng được định nghĩa tương tự. Với Precision-Recall Curve, AUC còn có một tên khác là **Average precision (AP)**.

Giả sử có NN ngưỡng để tính precision và recall, với mỗi ngưỡng cho một cặp giá trị precision, recall là  $P_n, R_n$ ,  $n=1,2,\dots, N$ . Precision-Recall curve được vẽ bằng cách vẽ từng điểm có toạ độ  $(R_n, P_n)$  trên trực toạ độ và nối chúng với nhau. AP được xác định bằng:

$$AP = \sum_n (R_n - R_{n-1})P_n \quad (3)$$

Trong đó  $(R_n - R_{n-1})P_n$  chính là diện tích hình chữ nhật có chiều rộng  $(R_n - R_{n-1})$  và chiều cao  $P_n$ , đây cũng gần với cách tính tích phân dựa trên cách tính diện tích của từng hình chữ nhật nhỏ.

### F1-score:

\$F\\_1\$ score, hay F1-score, là *harmonic mean* của precision và recall (giả sử rằng hai đại lượng này khác không):

$$\begin{aligned} \frac{2}{F_1} &= \frac{1}{precision} + \frac{1}{recall} \text{ hay } F_1 = 2 \frac{1}{\frac{1}{precision} + \frac{1}{recall}} \\ &= 2 \frac{precision \cdot recall}{precision + recall} \end{aligned} \quad (4)$$

F1-score có giá trị nằm trong nửa khoảng (0,1]. F<sub>1</sub> càng cao, bộ phân lớp càng tốt. Khi cả recall và precision đều bằng 1 (tốt nhất có thể), F<sub>1</sub>=1. Khi cả recall và precision đều thấp, ví dụ bằng 0.1 thì F<sub>1</sub> = 0.1. Dưới đây là một vài ví dụ về F<sub>1</sub>:

precision	recall	F <sub>1</sub>
1	1	1
0.1	0.1	0.1
0.5	0.5	0.5
1	0.1	0.182
0.3	0.8	0.36

Như vậy, một bộ phân lớp với precision = recall = 0.5 tốt hơn một bộ phân lớp khác với precision = 0.3, recall = 0.8 theo cách đo này.

Trường hợp tổng quát của F<sub>1</sub>-score là  $F_\beta$  score:

$$F_\beta = (1 + \beta^2) \frac{precision \cdot recall}{\beta^2 \cdot precision + recall} \quad (5)$$

F<sub>1</sub> chính là một trường hợp đặc biệt của  $F_\beta$  khi  $\beta = 1$ . Khi  $\beta > 1$ , recall được coi trọng hơn precision, khi  $\beta < 1$ , precision được coi trọng hơn. Hai đại lượng  $\beta$  thường được sử dụng là  $\beta = 2$  và  $\beta = 0.5$ .

Precision-recall cho bài toán phân lớp nhiều lớp

Cũng giống như ROC curve, precision-recall curve ban đầu được định nghĩa cho bài toán phân lớp nhị phân. Để có thể áp dụng các phép đo này cho bài toán multi-class classification, các đại lượng đầu ra (ground truth và predicted output) cần được đưa về dạng nhị phân.

Bằng trực giác, ta có thể đưa bài toán phân lớp nhiều lớp về bài toán phân lớp nhị phân bằng cách xem xét từng lớp. Với mỗi lớp, ta coi dữ liệu thuộc lớp đó có label là *positive*, tất cả các dữ liệu còn lại có label là *negative*. Sau đó, giá trị Precision, Recall, và PR curve được áp dụng lên từng lớp. Với mỗi lớp, ta sẽ nhận được một cặp giá trị precision và recall. Với các bài toán có ít lớp dữ liệu, ta có thể minh họa PR curve cho từng lớp trên cùng một đồ thị. Tuy nhiên, với các bài toán có rất nhiều lớp dữ liệu, việc này đôi khi không khả thi. Thay vào đó, hai phép đánh giá dựa trên Precision-Recall được sử dụng là *micro-average* và *macro-average*.

**Micro-average:** Micro-average precision, micro-average recall đơn giản tính bằng:

$$\text{micro-average precision} = \frac{\sum_{c=1}^C TP_c}{\sum_{c=1}^C (TP_c + FP_c)} \quad (6)$$

$$\text{micro-average recall} = \frac{\sum_{c=1}^C TP_c}{\sum_{c=1}^C (TP_c + FN_c)} \quad (7)$$

Với  $TP_c$ ,  $FP_c$ ,  $FN_c$  lần lượt là TP, FP, FN của lớp c.

Tức TP được tính là tổng của toàn bộ TP của mỗi lớp. Tương tự với FP và FN.

Micro-average F-score cũng được tính tương tự như F-score nhưng dựa trên micro-average precision và micro-average recall.

**Macro-average:**

- Macro-average precision là trung bình cộng của các precision theo class, tương tự với Macro-average recall.
- Macro-average F-Score cũng được tính tương tự như F-score nhưng dựa trên macro-average precision và macro-average recall.

→ Sau khi thống nhất, đối với đề tài khóa luận tốt nghiệp, nhóm sẽ sử dụng ma trận tương quan (Confusion matrix) và F1-score để đánh giá hệ thống phân lớp.

## 2.5. Các phương pháp/kỹ thuật xử lý dữ liệu mất cân bằng

Trong các phương pháp tiếp cận dữ liệu rất mất cân bằng, bằng cách điều chỉnh sự phân bố lớp để làm giảm sự mất cân bằng dữ liệu, thì thuật toán Under-sampling (giảm phần tử) và Over-sampling (tăng phần tử) là hai thuật toán phổ biến. Bên cạnh

đó, người ta cũng có thể kết hợp cả hai phương pháp trên, tức là cùng lúc giảm số phần tử ở lớp đa số và tăng phần tử ở lớp thiểu số.

### 2.5.1. Phương pháp Random Undersampling

Phương pháp giảm phần tử ở lớp đa số để làm giảm tính mất cân bằng dữ liệu. Cách đơn giản nhất đó là loại bỏ các phần tử ở lớp đa số một cách ngẫu nhiên. Ngoài ra cũng có một số cách giảm phần tử một cách có chủ đích như: giảm phần tử nhiều ở vùng an toàn (safe level), giảm phần tử đường biên (borderline).

Trong random undersampling, thực hiện ngẫu nhiên loại bỏ phần tử của lớp đa số trong tập dữ liệu huấn luyện cho đến khi ra tỉ số giữa đa số và lớp thiểu số phù hợp. Do đó tổng số dữ liệu huấn luyện được giảm đáng kể.

### 2.5.2. Phương pháp Random Oversampling – SMOTE

SMOTE là viết tắt của Synthetic Minority Over-sampling Technique. Không giống như Random UnderSampling, SMOTE tạo các điểm tổng hợp mới để có sự cân bằng của các lớp. Đây là một cách khác để giải quyết “các vấn đề mất cân bằng”.

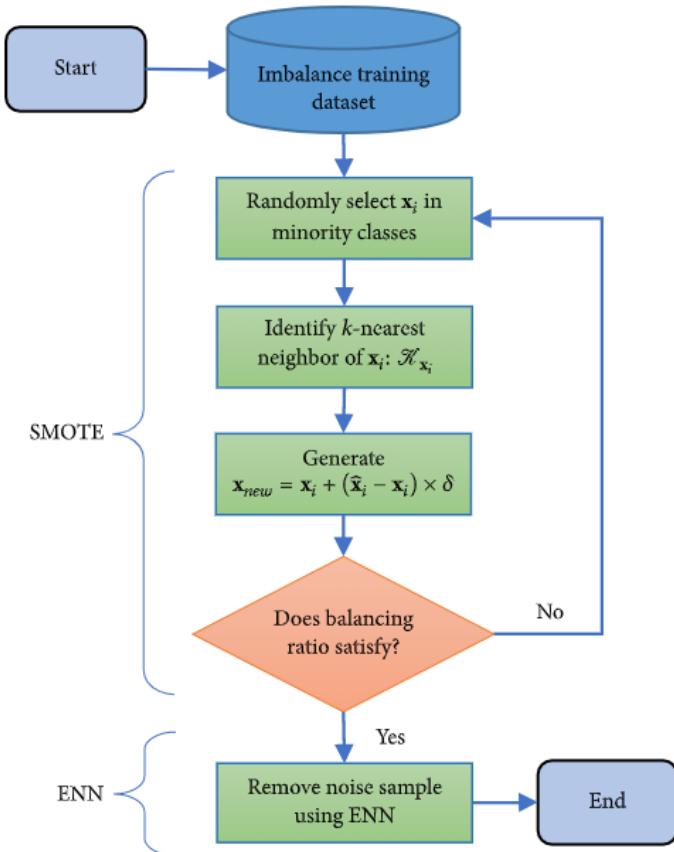
#### Hiểu về SMOTE:

- SMOTE tạo ra các dữ liệu tổng hợp (synthetic points) từ lớp thiểu số để đạt được sự cân bằng giữa nhóm thiểu số và đa số.
- Vị trí của các synthetic points: SMOTE chọn khoảng cách giữa các lân cận gần nhất của nhóm thiểu số, ở giữa các khoảng cách này, nó tạo ra các điểm tổng hợp.
- Nhiều thông tin được giữ lại vì chúng tôi không phải xóa bất kỳ hàng nào không giống như trong việc lấy mẫu ngẫu nhiên.
- Mặc dù có khả năng SMOTE sẽ chính xác hơn so với ly mẫu ngẫu nhiên, nhưng sẽ mất nhiều thời gian hơn để đào tạo vì không có hàng nào bị loại bỏ như đã nêu trước đây.

### 2.5.3. Phương pháp kết hợp SMOTE + ENN

- Sử dụng Smote để oversampling và sử dụng ENN để làm sạch dữ liệu
- ENN là viết tắt của (Edited Nearest Neighbours)

- ENN được sử dụng để loại bỏ các ví dụ từ cả hai lớp. Do đó, bất kỳ ví dụ nào bị phân loại sai bởi ba hàng xóm gần nhất của nó sẽ bị xóa khỏi tập huấn luyện.
- Phương pháp ENN có thể loại bỏ cả các ví dụ nhiễu như các ví dụ đường biên, cung cấp bề mặt quyết định mượt mà hơn. do đó, nó sẽ cung cấp khả năng làm sạch dữ liệu chuyên sâu hơn.



Hình 2.12. Quy trình thực hiện của thuật toán SMOTE + ENN<sup>11</sup>

<sup>11</sup> Nguồn ảnh:

## CHƯƠNG 3: MÔ HÌNH ĐỀ XUẤT

Qua quá trình tìm hiểu và nghiên cứu, nhóm tác giả sử dụng hai bộ dữ liệu được cung cấp bởi Kaggle:

- Các giao dịch thanh toán mô phỏng trên thiết bị di động – Paysim [6].
- Bộ dữ liệu về các giao dịch được thực hiện bằng thẻ tín dụng vào tháng 9 năm 2013 bởi các chủ thẻ châu Âu: Credit Card Fraud Detection [5].

### 3.1. Giới thiệu về bộ dữ liệu

#### 3.1.1. Bộ dữ liệu Paysim

Trong bối cảnh thiếu các bộ dữ liệu có sẵn công khai về các dịch vụ tài chính và đặc biệt trong lĩnh vực giao dịch tiền điện thoại di động. Các bộ dữ liệu tài chính rất quan trọng đối với nhiều nhà nghiên cứu và đặc biệt là trong việc phát hiện bất thường trong giao dịch tài chính. Một phần của vấn đề là bản chất riêng tư của các giao dịch tài chính, dẫn đến việc không có các bộ dữ liệu công khai.

Nhóm tác giả sử dụng bộ dữ liệu tổng hợp được tạo bằng trình giả lập PaySim – bộ dữ liệu được cung cấp bởi Kaggle. PaySim sử dụng dữ liệu được tổng hợp từ bộ dữ liệu riêng để tạo ra bộ dữ liệu tổng hợp giống với hai loại hoạt động của các giao dịch đó là: giao dịch hoạt động bình thường và thực hiện hành vi bất thường để sau đó đánh giá hiệu suất của phương pháp phát hiện bất thường trong giao dịch tài chính.

PaySim mô phỏng các giao dịch tiền điện thoại di động dựa trên một mẫu các giao dịch thực được trích từ một tháng nhật ký tài chính từ một dịch vụ tiền điện thoại di động được thực hiện ở một quốc gia châu Phi. Nhật ký ban đầu được cung cấp bởi một công ty đa quốc gia, nhà cung cấp dịch vụ tài chính di động hiện đang hoạt động tại hơn 14 quốc gia trên toàn thế giới.

Bộ dữ liệu tổng hợp này được thu nhỏ lại 1/4 bộ dữ liệu ban đầu và nó được tạo ra chỉ dành cho Kaggle.

Các thuộc tính và mô tả:

Bảng 3.1. Mô tả các thuộc tính của bộ dữ liệu Paysim

STT	Tên thuộc tính	Mô tả
1	step	Ánh xạ một đơn vị thời gian trong thế giới thực. Trong trường hợp này, step = 1 là 1 giờ. Tổng số step là 744 (mô phỏng trong 30 ngày)
2	type	Hình thức giao dịch, bao gồm: gửi tiền mặt, rút tiền mặt, ghi nợ, thanh toán và chuyển tiền.
3	amount	Số tiền của giao dịch bằng nội tệ.
4	nameOrig	Khách hàng ở tài khoản nguồn giao dịch
5	oldbalanceOrg	Số dư ban đầu trước khi giao dịch
6	newbalanceOrig	Số dư mới sau khi thực hiện giao dịch
7	nameDest	Khách hàng là người nhận giao dịch
8	oldbalanceDest	Số dư trước khi giao dịch của tài khoản đích. Lưu ý đối với khách hàng bắt đầu bằng M (thương gia) thì oldbalanceDest = 0
9	newbalanceDest	Số dư sau khi giao dịch của tài khoản đích. Lưu ý đối với khách hàng bắt đầu bằng M (thương gia) thì newbalanceDest = 0
10	isFraud	Hiển thị nhận biết là giao dịch bất thường hay không, với isFraud=1 là giao dịch bất thường, isFraud=0 là giao dịch bình thường.
11	isFlaggedFraud	isFlaggedFraud = 1 khi giao dịch chuyển tiền bất hợp pháp với số tiền chuyển hơn 200000 trong một giao dịch duy nhất. Trường này với mục đích kiểm soát việc chuyển tiền ô ạt sang một tài khoản khác và đánh dấu sự bất hợp pháp của giao dịch

### 3.1.2. Bộ dữ liệu Credit Card Fraud Detection

Bộ dữ liệu Credit Card Fraud Detection được chia sẻ trên Kaggle.

Bộ dữ liệu chứa các giao dịch được thực hiện bằng thẻ tín dụng vào tháng 9 năm 2013 bởi các chủ thẻ châu Âu. Bộ dữ liệu này trình bày các giao dịch xảy ra trong hai ngày, trong đó có 492 vụ lừa đảo trong số 284807 giao dịch. Bộ dữ liệu rất mất cân bằng, lớp positive (bất thường) chiếm 0,172% của tất cả các giao dịch.

Dữ liệu chỉ chứa các biến đầu vào số là kết quả của chuyển đổi PCA. Thật không may, do vấn đề bảo mật, không thể cung cấp các tính năng gốc và thông tin cơ bản hơn về dữ liệu. Các tính năng V1, V2, ... V28 là các thành phần chính thu được với PCA, các tính năng duy nhất chưa được chuyển đổi với PCA là 'Time' (thời gian) và 'Amount' (số tiền). Thuộc tính 'Time' chứa các giây trôi qua giữa mỗi giao dịch và giao dịch đầu tiên trong bộ dữ liệu. Thuộc tính 'Amount' là Số tiền giao dịch, thuộc tính này có thể được sử dụng cho việc học tập dựa trên chi phí phụ thuộc vào ví dụ. Thuộc tính 'Class' là biến phản hồi và nó nhận giá trị 1 trong trường hợp giao dịch bất thường và 0 là giao dịch bình thường.

Bộ dữ liệu chứa 284807 giao dịch. Giá trị trung bình của tất cả các giao dịch là 88,35\$ trong khi giao dịch lớn nhất được ghi lại trong bộ dữ liệu này lên tới 25691.16 \$.

Các thuộc tính và mô tả:

Bảng 3.2. Mô tả các thuộc tính của bộ dữ liệu Credit Card Fraud Detection

STT	Tên thuộc tính	Mô tả
1	class	Có hai giá trị: 0 là giao dịch bình thường, 1 là giao dịch bất thường.
2	Amount	Số tiền thực hiện giao dịch
3	V1, V2, ..., V28	Các thuộc tính ẩn danh do sự riêng tư. Giá trị của các thuộc tính này là số, là kết quả của quá trình chuyển đổi PCA.

4	Time	Lượng giây trôi qua giữa mỗi giao dịch, và giao dịch đầu tiên trong bộ dữ liệu.
---	------	---

### 3.1.3. Thao tác tiền xử lý dữ liệu

Đối với hai bộ dữ liệu đề cập ở trên, nhóm sẽ thực hiện quá trình tiền xử lý dữ liệu bằng việc phân tích, tách dữ liệu và lựa chọn các thuộc tính phù hợp với quá trình nghiên cứu.

Qua quá trình nghiên cứu, đối với bài toán phát hiện bất thường trong giao dịch tài chính này, nhóm sẽ sử dụng: Đối với dữ liệu mất cân bằng, tiến hành xử lý tiền dữ liệu bằng việc sử dụng một trong các phương pháp sau dành cho hai bộ dữ liệu trên:

- Random Undersampling.
- Random Oversampling.
- Phương pháp kết hợp sử dụng SMOTE + ENN.

## 3.2. Ngôn ngữ lập trình được lựa chọn

### 3.2.1. Python

Python là ngôn ngữ lập trình thông dịch, hướng đối tượng, ngôn ngữ lập trình, cấp cao được giải thích với ngữ nghĩa rộng lớn. Là một ngôn ngữ lập trình phổ biến, được tạo ra bởi Guido van và Ross vào năm 1991.

Theo thống kê đến thời điểm hiện tại của thì ngôn ngữ Python đang dẫn đầu so với các ngôn ngữ lập trình còn lại [13]:

Worldwide, Dec 2019 compared to a year ago:				
Rank	Change	Language	Share	Trend
1		Python	29.71 %	+4.1 %
2		Java	19.29 %	-2.2 %
3		Javascript	8.33 %	+0.0 %
4		C#	7.27 %	-0.4 %
5		PHP	6.32 %	-1.0 %
6		C/C++	6.0 %	-0.3 %
7		R	3.79 %	-0.2 %
8		Objective-C	2.61 %	-0.6 %
9		Swift	2.5 %	-0.1 %
10		Matlab	1.84 %	-0.2 %

Hình 3.1. So sánh các ngôn ngữ lập trình<sup>12</sup>

Hiện nay với khả năng xử lý các phép toán phức tạp của mình, Python đang được sử dụng nhiều trong việc phát triển trí tuệ nhân tạo và các nghiên cứu trong lĩnh vực machine learning vì tính đơn giản và nhỏ gọn của nó.

Ưu điểm khi sử dụng Python:

- Là ngôn ngữ dễ hiểu, cấu trúc rõ ràng, cú pháp ngắn gọn.
- Có trên tất cả các nền tảng hệ điều hành từ UNIX, MS – DOS, Mac OS, Windows và Linux và các OS khác thuộc Unix.
- Tương thích mạnh mẽ với Unix, hardware, third-party software với số lượng thư viện khổng lồ và mạnh mẽ để tính toán cũng như làm Machine learning như Numpy, Sympy, Scipy, Matplotlib, Pandas, TensorFlow, Keras, ...

Nhược điểm của Python:

- Mang đầy đủ điểm yếu của các ngôn ngữ thông dịch như tốc độ chậm, tiềm tàng lỗi trong quá trình thông dịch, source code dễ dàng bị dịch ngược.
- Ngôn ngữ có tính linh hoạt cao nên thiếu chặt chẽ.

<sup>12</sup> Nguồn ảnh: <http://pypl.github.io/PYPL.html>

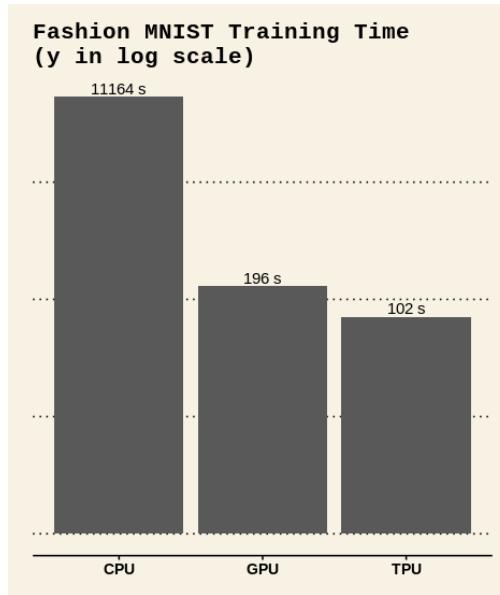
### 3.3. Google Colab

Machine learning/Deep learning đang phát triển với tốc độ rất nhanh. Để viết một chương trình sử dụng framework về Deeplearning như TensorFlow, Kere hay Pytorch, chúng ta có thể sử dụng bất kỳ Python IDE nào như PyCharm, Jupyter Notebook hay Atom. Tuy nhiên, do những thuật toán Machine learning/Deep learning yêu cầu hệ thống phải có tốc độ và khả năng xử lý cao (thông thường dựa trên GPU), mà máy tính của chúng ta thông thường không được trang bị GPU. Rất nhiều người học và nghiên cứu chọn giải pháp là thuê những dịch vụ tính toán trên Amazon Web Servies (AWS).

Từ đó, Google cho ra đời một dịch vụ hoàn toàn miễn phí dành cho cộng đồng nghiên cứu AI, phát triển các ứng dụng Deep learning bằng việc cung cấp GPU và TPU miễn phí: Google Colab.

Google colab là một dịch vụ đám mây miễn phí, hiện nay có hỗ trợ GPU (Tesla K80) và TPU (TPUv2). Do được phát triển dựa trên Jupiter Notebook nên việc sử dụng Google Colab cũng tương tự như việc sử dụng Jupyter Notebook. Google Colab là một công cụ lý tưởng để chúng ta rèn luyện kỹ năng lập trình với ngôn ngữ Python thông qua các thư viện của deep learning. Google Colab cài đặt sẵn cho chúng ta những thư viện rất phổ biến trong nghiên cứu Deep Learning như PyTorch, TensorFlow, Keras và OpenCV.

Hình 3.2 sau đây là Benchmark so sánh giữa tốc độ xử lý của CPU, GPU và TPU trong Google Colab. Căn cứ vào kết quả so sánh, ta thấy rằng TPU đạt tốc độ nhanh gần gấp đôi GPU và nhanh hơn gấp 10 lần so với CPU.



Hình 3.2. Bảng so sánh giữa tốc độ xử lý của CPU, GPU và TPU

Do được phát triển dựa trên Jupyter Notebook nên việc sử dụng Google Colab cũng tương tự như khi sử dụng Jupyter Notebook (file tạo ra trong Google Colab cũng có đuôi .ipynb như trong Jupyter Notebook).

### 3.3.1. Lý do chọn Google colab

Cung cấp máy ảo miễn phí:

- Lên đến 12gb ram.
- 50gb dung lượng ổ cứng.
- Cài đặt sẵn một số thư viện phổ biến: Numpy, Pandas, TensorFlow, ...
- Truy cập GPU, TPU miễn phí: GPU “NVIDIA Tesla K80”.
- Có hỗ trợ Python 3 và cả Python 2.
- Có hỗ trợ kết nối với local để làm việc.
- Cung cấp thư viện code example phong phú.
- Có tích hợp với Google Drive, Github.
- Có thể chia sẻ kiểm soát quyền và bạn sẽ có thể thấy các cộng tác viên khác làm việc ngay lập tức.
- Có lưu lại lịch sử thay đổi, một tính năng cực kỳ hữu ích khi làm việc nhóm.

- Bạn cũng có thể nhận xét trên từng ô, ví dụ ai đó được cấp quyền “comment-only”.
- Import Jupyter/IPython notebooks cá nhân vào.

### 3.3.2. Hạn chế

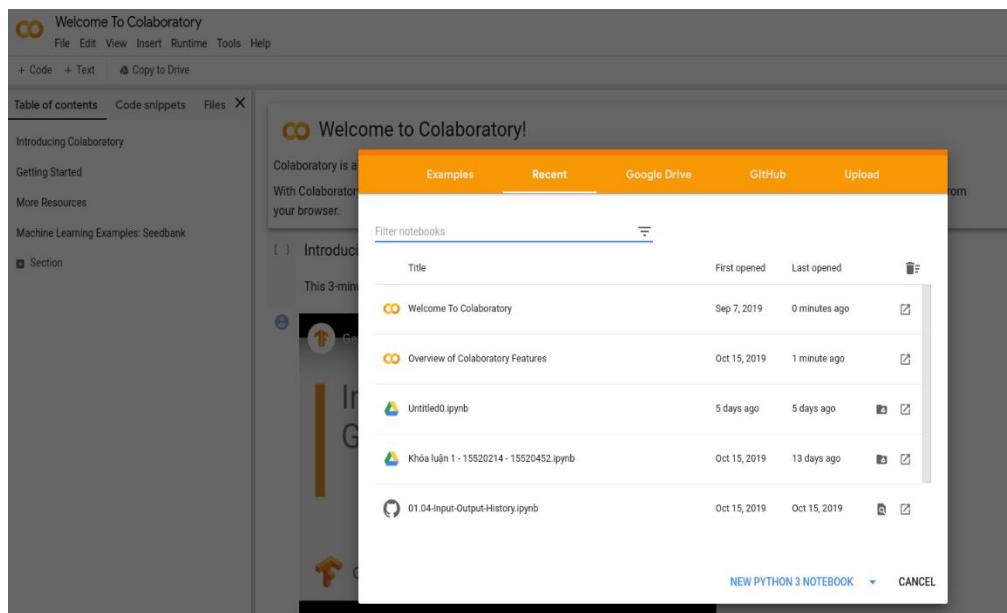
Tất cả các sổ ghi chép Colaboratory phải được lưu trữ trong Google Drive - vì vậy bạn sẽ cần phải đăng nhập vào tài khoản Google trước khi bạn có thể truy cập công cụ.

Việc tính toán nền chạy dài có thể bị dừng, người dùng muốn chạy tính toán liên tục hoặc chạy dài thì nên chạy thông qua giao diện người dùng của Colaboratory để sử dụng thời gian chạy cục bộ".

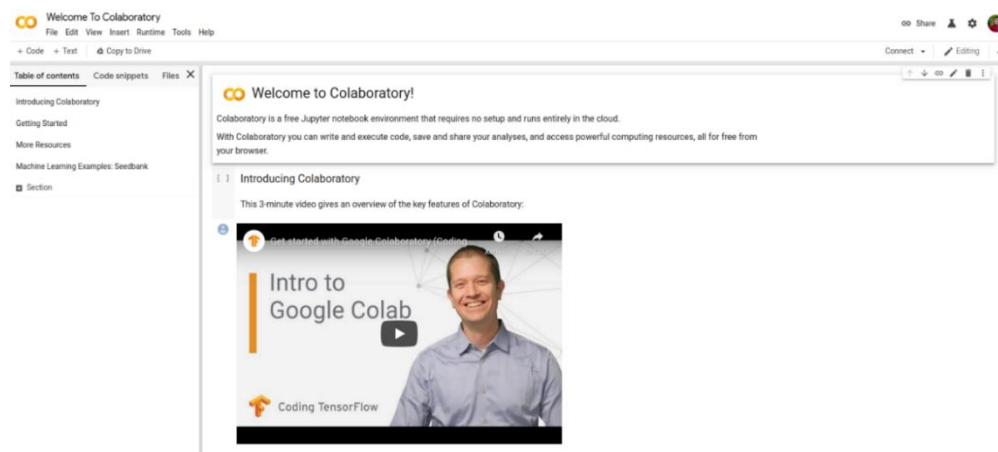
Bạn cần phải cài đặt 1 số thư viện python không phổ biến mà bạn cần dùng đến, và bạn phải lặp lại mỗi lần reset session.

Google Storage được sử dụng với session hiện tại của bạn, vì vậy muốn lưu trữ dữ liệu thì phải chuyển qua Google Driver trước khi đóng session.

Có thể khó khăn (và có khả năng tốn kém) khi làm việc với các bộ dữ liệu lớn hơn khi bạn phải tải xuống và lưu trữ chúng trong ổ Google (chỉ 15 GB là miễn phí trong Google Drive).



Hình 3.3. Giới thiệu về Colaboratory 1



Hình 3.4. Giới thiệu về colaboratory 2

## CHƯƠNG 4: THỰC NGHIỆM

Để tài nghiên cứu được trình bày như đã nói sử dụng Google Colab, do đó quá trình thực nghiệm và mô phỏng đề tài sẽ sử dụng Google Colab để trình bày.

### 4.1. Bộ dữ liệu Paysim

#### 4.1.1. Các bước thực hiện

- Bước 1: Import Library
- Bước 2: Connect to google drive.
- Bước 3: Đọc dữ liệu vừa kết nối

Chi tiết lệnh thực hiện xem phần phụ lục.

Ta được kết quả như sau:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYOUT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0	0
1	1	PAYOUT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0	0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1	0
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1	0
4	1	PAYOUT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0	0

Hình 4.1. Kết quả đọc bộ dữ liệu Paysim sau khi kết nối

#### 4.1.2. Tìm hiểu tổng quan dữ liệu

##### a. Kiểu dữ liệu các thuộc tính

Kiểu dữ liệu được mô tả như bảng dưới đây:

Bảng 4.1. Kiểu dữ liệu của các thuộc tính trong bộ dữ liệu Paysim

Lệnh	Thuộc tính	Kiểu dữ liệu
#Hiển thị kiểu dữ liệu của các thuộc tính data.dtypes	step	int (64)
	type	object
	amount	float (64)
	nameOrig	object
	oldbalanceOrg	float (64)
	nameDest	object
	oldbalanceDest	float (64)

	newbalanceDest	float (64)
	isFraud	int (64)
	isFlaggedFraud	int (64)

b. Kích thước dữ liệu

Tiến hành chạy lệnh sau:

Lệnh	Mô tả
# Kích thước dữ liệu data.shape	Dữ liệu có 6.362.620 dòng, và 11 cột

c. Số lượng giao dịch bình thường và giao dịch bất thường

Lệnh	Mô tả
# Số lượng giao dịch bình thường và giao dịch bất thường data.isFraud.value_counts()	Dữ liệu có 6354407 giao dịch bình thường và 8213 giao dịch bất thường

d. Số dòng dữ liệu bị thiếu

Lệnh	Mô tả
data.isnull().values.any()	Với True: dữ liệu bị thiếu, False: Có dữ liệu. Kết quả trả về: False. Do đó dữ liệu không bị thiếu.

e. Tổng quan bộ dữ liệu

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
count	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06
mean	2.433972e+02	1.798619e+05	8.338831e+05	8.551137e+05	1.100702e+06	1.224996e+06	1.290820e-03	2.514687e-06
std	1.423320e+02	6.038582e+05	2.888243e+06	2.924049e+06	3.399180e+06	3.674129e+06	3.590480e-02	1.585775e-03
min	1.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	1.560000e+02	1.338957e+04	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
50%	2.390000e+02	7.487194e+04	1.420800e+04	0.000000e+00	1.327057e+05	2.146614e+05	0.000000e+00	0.000000e+00
75%	3.350000e+02	2.087215e+05	1.073152e+05	1.442584e+05	9.430367e+05	1.111909e+06	0.000000e+00	0.000000e+00
max	7.430000e+02	9.244552e+07	5.958504e+07	4.958504e+07	3.560159e+08	3.561793e+08	1.000000e+00	1.000000e+00

Hình 4.2. Tổng quan các giá trị của bộ dữ liệu Paysim

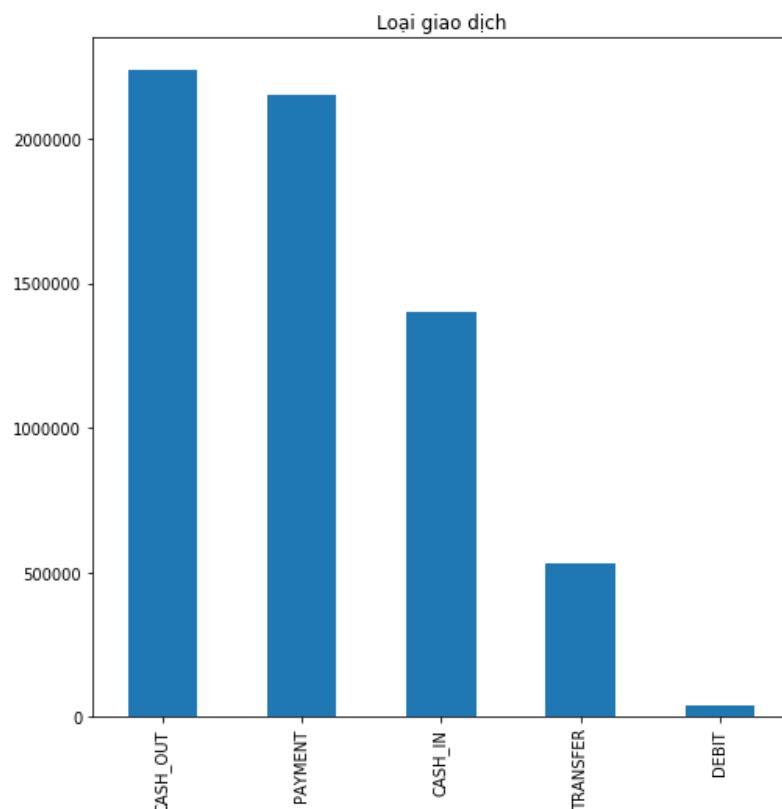
### 4.1.3. Phân tích dữ liệu

#### 4.1.3.1. Những loại giao dịch gian lận

##### a. Phân phối xác suất cột loại giao dịch

Loại giao dịch	Số lượng giao dịch
CASH_OUT	2237500
PAYMENT	2151495
CASH_IN	1399284
TRANSFER	532909
DEBIT	41432

Biểu đồ thể hiện số lượng theo loại giao dịch):



Hình 4.3. Biểu đồ thể hiện số lượng theo loại giao dịch

##### b. Phân phối xác suất giao dịch bình thường và giao dịch bất thường ở mỗi loại giao dịch

Ghi chú: Ở cột isFraud, trong tập dữ liệu:

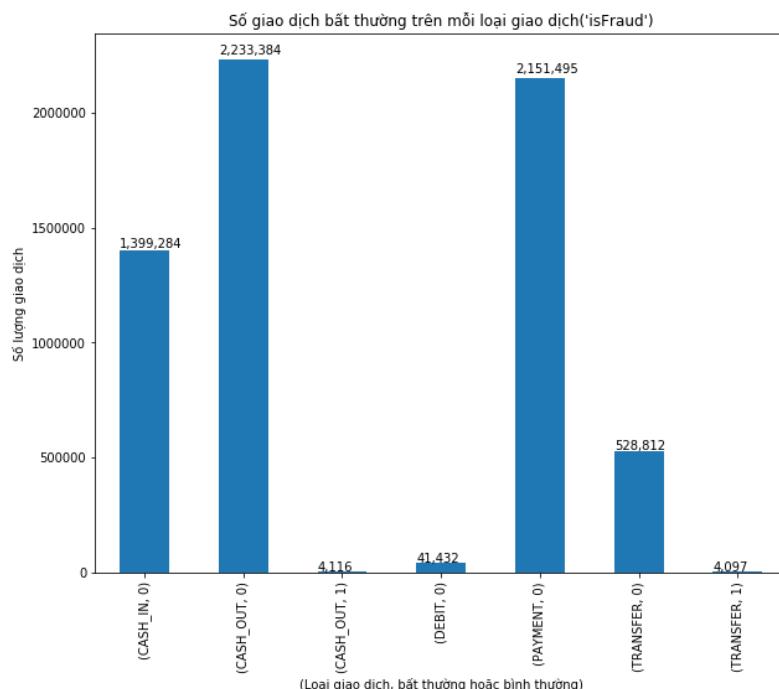
- 0: là giao dịch bình thường.
- 1: là giao dịch bất thường.

Thống kê số lượng giao dịch bất thường và giao dịch bình thường ở mỗi loại giao dịch:

Bảng 4.2. Thống kê số lượng giao dịch bất thường và bình thường  
ở mỗi loại giao dịch

STT	Loại giao dịch	Số lượng giao dịch bình thường	Số lượng giao dịch bất thường
1	PAYMENT	2151495	0
2	TRANSFER	528812	4097
3	CASH_OUT	2233384	4116
4	DEBIT	41432	0
5	CASH_IN	1399284	0

Biểu đồ thể hiện số giao dịch bất thường ở mỗi loại giao dịch:



Hình 4.4. Biểu đồ thể hiện số giao dịch bất thường và giao dịch bình thường  
ở mỗi loại giao dịch

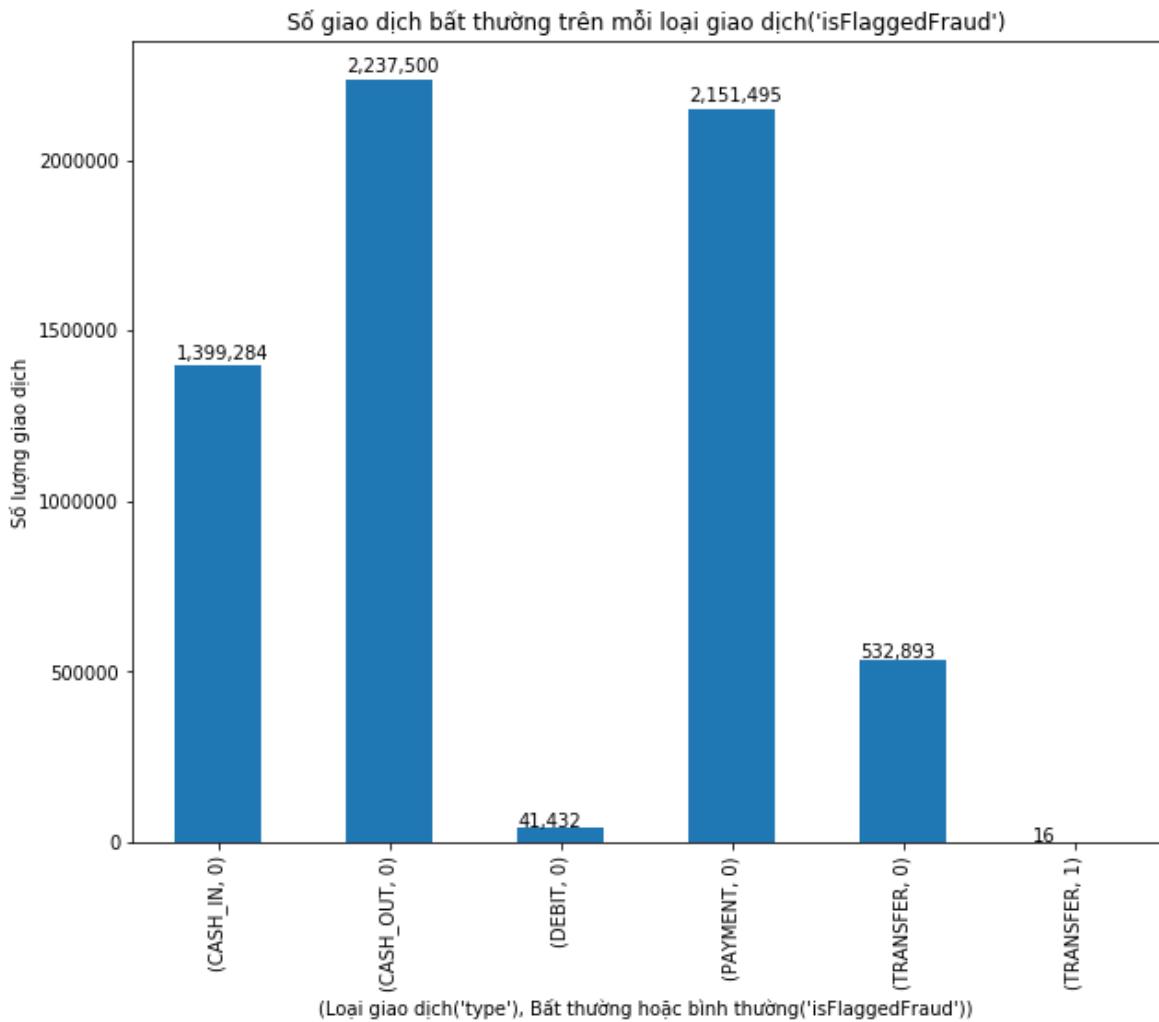
**Nhận xét:**

- Chỉ có 2 loại giao dịch: TRANSFER và CASH\_OUT là có giao dịch bất thường.
- Số lượng giao dịch bất thường ở 2 loại giao dịch này gần như bằng nhau.
- Theo như mô tả kèm theo bộ dữ liệu, mô giả định dự đoán cho **Giao dịch bất thường**: Tiền được chuyển từ tài khoản này sang tài khoản khác, sau đó rút tiền ra.

**4.1.3.2. Tìm hiểu ý nghĩa thuộc tính isFlaggedFraud**

Theo như thông tin được cung cấp, thuộc tính isFlaggedFraud = 1 khi giao dịch chuyển tiền có giá trị lớn hơn 200000. Tuy nhiên không đề cập đến giá trị nào có giá trị lớn hơn 200000, dựa theo một nghiên cứu đã có [14], chúng ta sẽ đi đặt các giả định.

Biểu đồ thể hiện số giao dịch bất thường trên mỗi loại giao dịch (xét theo thuộc tính isFlaggedFraud = 1) (chi tiết lệnh thực hiện xem phụ lục).



Hình 4.5. Biểu đồ thể hiện số giao dịch bất thường trên mỗi loại giao dịch theo thuộc tính isFlaggedFraud)

Phân loại các giao dịch theo giao dịch bình thường và bất thường của thuộc tính isFlaggedFraud được cho như bảng 4.3.

Bảng 4.3. Số lượng giao dịch bình thường và bất thường theo thuộc tính isFlaggedFraud của mỗi giao dịch

Loại giao dịch	Số lượng giao dịch theo thuộc tính isFlaggedFraud	
	Giao dịch bình thường	Giao dịch bất thường
CASH_IN	1399284	0
CASH_OUT	2237500	0

DEBIT	41432	0
PAYMENT	2151495	0
TRANSFER	532893	16

**Nhận xét:**

Chỉ có 16 giao dịch TRANSFER được đánh dấu là “isFlaggedFraud” trong toàn bộ dữ liệu.

**a. Giả thuyết: Thuộc tính ‘amount’  $\geq 200000$** 

Đầu tiên tìm những loại giao dịch có thuộc tính isFlaggedFraud được đánh dấu. Qua chạy lệnh thì giao dịch có thuộc tính isFlaggedFraud được đánh dấu là giao dịch loại Transfer.

Sau đó, ta tìm giá trị giao dịch nhỏ nhất khi isFlaggedFraud được đánh dấu ( $isFlaggedFraud = 1$ ) và tìm giá trị giao dịch lớn nhất có loại giao dịch là TRANSFER khi  $isFlaggedFraud$  không được đánh dấu ( $isFlaggedFraud = 0$ ), thu được kết quả.

- Giá trị giao dịch nhỏ nhất khi isFlaggedFraud của giao dịch Transfer được đánh dấu (1): 353874.22.
- Giá trị giao dịch lớn nhất khi isFlaggedFraud của giao dịch Transfer không được đánh dấu (0): 92445516.64.
- Để kiểm tra độ chênh lệch và kiểm tra đối với thuộc tính TRANSFER khi có amount lớn hơn bằng 200000 và  $isFlaggedFraud=0$  có đủ điều kiện sử dụng để làm tập huấn luyện, tiến hành chạy các lệnh sau:

Lệnh	Kết quả
<code>dfTransfer.isFlaggedFraud.sum()</code>	16
<code>len(dfTransfer.loc[(dfTransfer['amount'] &gt;= 200000) &amp; (dfTransfer['isFlaggedFraud'] == 0)])</code>	409094

**Nhận xét:**

- Trên thực tế, isFlaggedFraud vẫn không thể được đặt mặc dù điều kiện này được đáp ứng. Do đó loại trường hợp isFlaggedFraud phụ thuộc vào điều kiện thuộc tính amount  $\geq 200000$ .

**b. Giả thuyết isFlaggedFraud = 1 khi giao dịch chuyển tiền bị hủy**

Kiểm tra số giao dịch có số dư tài khoản nhận tiền trước và sau khi thực hiện giao dịch, ta có kết quả:

- Số giao dịch loại Transfer với isFlaggedFraud = 0, oldBalanceDest = 0 và newBalanceDest = 0 là: 4158 giao dịch.

**Nhận xét:**

- Điều kiện ở trên cũng không xác định được trạng thái của isFlaggedFraud, do đó loại giả thuyết này.

**c. Giả thuyết isFlaggedFraud = 1 khi oldbalanceOrg = newbalanceOrg**

Không xét thuộc tính newBalanceOrig vì nó chỉ được cập nhật sau khi giao dịch, trong khi isFlaggedFraud sẽ được đặt trước khi giao dịch diễn ra.

Một số kết quả sau khi chạy các lệnh liên quan:

- Giá trị nhỏ nhất của thuộc tính 'oldBalanceOrig' của isFlaggedFraud = 1 của giao dịch loại TRANSFER là: 353874.0.
- Giá trị lớn nhất của thuộc tính 'oldBalanceOrig' của isFlaggedFraud = 1 của giao dịch loại TRANSFER là: 19585040.0.
- Giá trị nhỏ nhất của thuộc tính 'oldBalanceOrig' của isFlaggedFraud = 0 khi oldBalanceOrig = newbalanceOrig của giao dịch loại TRANSFER là: 0.0.
- Giá trị nhỏ nhất của thuộc tính 'oldBalanceOrig' của isFlaggedFraud = 0 khi oldBalanceOrig = newbalanceOrig của giao dịch loại TRANSFER là: 575668.0.

**Nhận xét:**

- Có tồn tại trường hợp thỏa điều kiện những thuộc tính isFlaggedFraud = 0, đó là loại giả thuyết này.

**d. Giả thuyết IsFlaggedFraud có thể được đặt dựa trên việc một khách hàng giao dịch nhiều lần không?**

Bảng dưới đây thống kê các thông tin và kết quả trả về như sau:

Bảng 4.4. Kết quả truy vấn theo giả thuyết d

Thao tác truy vấn tìm	Kết quả
Có người khởi tạo các giao dịch được gắn cờ là bất thường có giao dịch nhiều lần	False
Có điểm đến của giao dịch được gắn cờ isFlaggedFraud nhiều hơn một lần	False
Số tài khoản đích của các giao dịch được gắn cờ là bất thường đã là tài khoản đích nhiều lần	2

**Nhận xét:**

- Thuộc tính isFlaggedFraud xảy ra ở tất cả các giá trị ở các thuộc tính khác nhau.
- IsFlaggedFraud dường như không liên quan đến bất kỳ biến hoặc tính năng giải thích nào trong dữ liệu.

**Kết luận:**

- Mặc dù isFraud, luôn được đặt khi 'isFlaggedFraud' được đặt, vì isFlaggedFraud được đặt chỉ 16 lần theo cách dường như vô nghĩa, chúng ta có thể coi tính năng này là không đáng kể và loại bỏ nó trong bộ dữ liệu mà không mất thông tin.

**4.1.3.3. Tìm hiểu các tài khoản thương gia (bắt đầu bằng M) được gán nhãn****a. Thực hiện kiểm tra đối với ba loại giao dịch là: CASH\_IN, CASH\_OUT, PAYMENT, xem có tài khoản thương gia nào thực hiện các giao dịch trên hay không?**

Kết quả được cho theo bảng dưới đây:

Bảng 4.5. Thông tin truy vấn và kết quả trả về khi tìm các tài khoản thương gia có thực hiện một trong các giao dịch CASH\_IN, CASH\_OUT, PAYMENT

Thao tác truy vấn tìm	Kết quả
Tìm tài khoản thương gia (đối với mã khách hàng bắt đầu bằng M) thực hiện giao dịch CASH_IN	False
Tìm tài khoản thương gia (đối với mã khách hàng bắt đầu bằng M) là tài khoản đích trong các thực hiện giao dịch CASH_OUT	False
Tìm bất kỳ tài khoản thương gia (đối với mã khách hàng bắt đầu bằng M) là tài khoản đích mà loại giao dịch khác ‘PAYMENT’	Flase

#### Nhận xét:

- Trong thực tế, không có tài khoản thương gia trong số bất kỳ tài khoản người khởi tạo. Thương gia chỉ có mặt trong tài khoản đích cho tất cả các thanh toán.

#### b. Có nhãn tài khoản xuất hiện nhiều lần trong giao dịch TRANSFER bất thường và CASH\_OUT không?

Theo mô tả được cung cấp:

- Phương thức hoạt động để thực hiện hành vi bất thường liên quan đến việc trước tiên thực hiện việc chuyển tiền vào tài khoản (lừa đảo), từ đó tiến hành CASH\_OUT. CAST\_OUT liên quan đến giao dịch với một thương gia thanh toán bằng tiền mặt.
- Do đó, trong quy trình gồm hai bước này, tài khoản bất thường sẽ là cả hai, đích đến trong việc chuyển tiền và người khởi tạo trong CASH\_OUT. Tuy nhiên, dữ liệu, cho thấy bên dưới rằng không có tài khoản xuất hiện nhiều lần như vậy trong số các giao dịch bất thường.

Ta thực hiện:

- Tìm trong các giao dịch bất thường có tài khoản đích của giao dịch TRANSFER là tài khoản nguồn của giao dịch rút tiền: Không có.
- Tìm trong các giao dịch bất thường có tài khoản nguồn của giao dịch TRANSFER (chuyển tiền) tài khoản nguồn của giao dịch CASH\_OUT (rút tiền), kết quả theo bảng sau đây:

Bảng 4.6. Kết quả khi thực hiện tìm các giao dịch bất thườngs có tài khoản nguồn của giao dịch transfer là tài khoản nguồn của giao dịch cash\_out

Thuộc tính			
step	65	486	738
type	TRANSFER	TRANSFER	TRANSFER
amount	1282971.57	214793.32	814689.88
nameOrig	C1175896731	C2140495649	C2029041842
oldbalanceOrg	1282971.57	214793.32	814689.88
newbalanceOrig	0.0	0.0	0.0
nameDest	C1714931087	C423543548	C1023330867
oldbalanceDest	0.0	0.0	0.0
NewbalanceDest	0.0	0.0	0.0
isFraud	1	1	1
isFlaggedFraud	0	0	0

Từ bảng trên, khi chạy lệnh trên bộ dữ liệu, tìm được giao dịch chuyển tiền đến tài khoản C423543548 vào thời gian bằng 486 trong khi giao dịch rút tiền từ tài khoản này đã diễn ra rất sớm vào thời gian bằng 185.

### Kết luận:

- Từ kết quả thu được ở phần 4.1.3.1 và 4.1.3.2 ở trên, các thuộc tính ‘nameOrig’ và ‘nameDest’ không thể hiện được tài khoản người bán theo cách dự kiến, nên **sẽ bỏ qua các thuộc tính này khỏi dữ liệu** vì chúng vô nghĩa.

#### 4.1.4. Tiền xử lý dữ liệu

Từ việc phân tích dữ liệu sơ bộ của phần 4.1.3. Phân tích dữ liệu, có thể thấy được rằng sự bất thường chỉ xảy ra đối với loại giao dịch là “TRANSFER” và “CASH\_OUT”.

Do đó, sẽ thực hiện chỉnh lại dữ liệu tương ứng để phân tích.

Thực hiện các thao tác sau:

- Trích ra những dòng dữ liệu loại ‘TRANSFER’ hoặc ‘CASH\_OUT’.
- Tách thuộc tính phụ thuộc (isFraud) và thuộc tính độc lập.
- Loại bỏ 3 thuộc tính không có ý nghĩa là: ‘nameOrig’, ‘nameDest’, ‘isFlaggedFraud’.
- Chuyển danh nhãn cho loại giao dịch TRANSFER = 0, và CASH\_OUT = 1.
- Chuyển kiểu dữ liệu thuộc tính ‘type’ về số nguyên.
- Sau đó tiến hành xáo trộn dữ liệu (hiện tại đang được sắp xếp theo thuộc tính step).

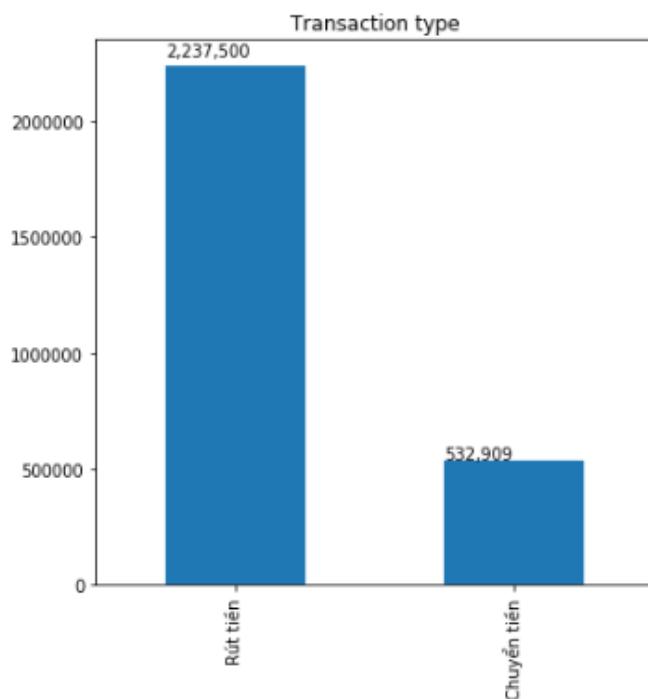
Kết quả:

- Dữ liệu sau khi xáo trộn có 2770409 dòng.
- Kiểm tra các bộ dữ liệu có phân tách ta được kết quả theo hình sau:

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest
1509681	145	0	104178.14	29943.0	0.0	4992785.43	5096963.56
4144975	303	1	77912.13	0.0	0.0	850455.85	928367.98
1640370	157	0	179519.26	0.0	0.0	5207670.43	5698247.60
2744120	212	1	195401.06	0.0	0.0	1219642.12	1415043.18
3300769	252	1	246354.91	0.0	0.0	3373089.40	5074355.54

Hình 4.6. Kết quả kiểm tra các bộ dữ liệu khi phân tách

Tổng quan bộ dữ liệu sau khi tiến hành lọc dữ liệu, được thể hiện qua biểu đồ bằng cách chạy lệnh dưới đây:



Hình 4.7. Biểu đồ thể hiện tổng quan bộ dữ liệu sau khi tiến hành xử lý

### Feature Engineering<sup>13</sup>:

Ta có:

<sup>13</sup> Feature Engineering là quá trình chuyển đổi tập dữ liệu ban đầu thành tập các thuộc tính (features) có thể giúp biểu diễn tập dữ liệu ban đầu tốt hơn, tạo điều kiện để giải quyết các bài toán dễ dàng hơn, giúp tương thích với từng mô hình dự đoán cụ thể, cũng như cải thiện độ chính xác của mô hình dự đoán hiện tại.

- Tỷ lệ giao dịch bất thường với ‘oldBalanceDest’ = ‘newBalanceDest’ = 0 mặt dù ‘amount’ = 0 là: 0.4955558261293072.
- Tỷ lệ giao dịch bình thường với oldBalanceDest’ = ‘newBalanceDest’ = 0 mặt dù ‘amount’ = 0 là: 0.0006176245277308345. (xem chi tiết lệnh và kết quả phần phụ lục).

### Nhận xét:

- Dữ liệu có một số giao dịch với số dư tài khoản đích trước và sau khi thực hiện giao dịch đều bằng 0 trong khi số tiền giao dịch khác 0.
- Tỷ lệ của các giao dịch trên trong dữ liệu:
  - Với giao dịch bất thường là khoảng 50%.
  - Với giao dịch bình thường là khoảng 0.06%.

### Kết luận:

- Có thể nhận thấy số dư tài khoản đích = 0 là một dấu hiệu mạnh mẽ để phát hiện sự bất thường khi giao dịch. Và việc giá trị bằng 0 làm cho các thuật toán machine learning không hoạt động tốt, do đó ta sẽ thay thế bằng -1 sẽ tốt hơn cho các thuật toán machine learning.

Thực hiện thay thế giá trị 0 thành -1 cho các thuộc tính: oldbalanceDest và newbalanceDest. Mục đích dấu khác thường của dữ liệu, có thể là một thông tin có ích khi huấn luyện.

- Tương tự, dữ liệu cũng có một số giao dịch với số dư bằng 0 trong tài khoản gốc cả trước và sau trong khi số tiền giao dịch khác 0.
- Trong trường hợp này, tỷ lệ của các giao dịch như vậy nhỏ hơn nhiều so với giao dịch có sự bất thường (0.3%), và so với giao dịch bình thường 47%.

Ta tiến hành tạo 2 thuộc tính mới dựa trên 5 thuộc tính đã có. Đó là errorBalanceOrig và errorBalanceDest:

step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	errorBalanceOrig	errorBalanceDest
355	0	88915.82	NaN	NaN	828181.60	917097.42	0	NaN	-1.164153e-10
15	1	479902.86	61068.00	0.0	4640292.80	5120195.66	0	418834.86	0.000000e+00
140	0	6609.59	NaN	NaN	174076.65	180686.23	0	NaN	1.000000e-02
16	1	235732.48	57143.63	0.0	1648427.18	2128627.11	0	178588.85	-2.444674e+05
328	1	141158.75	NaN	NaN	673931.18	815089.93	0	NaN	0.000000e+00

Hình 4.8. Kết quả sau khi tiến hành tạo mới 2 thuộc tính từ các thuộc tính đã có

Kích thước của bộ dữ liệu sau khi tạo hai thuộc tính mới là 2770409 dòng.

Ta thấy được các thuộc tính mới này hóa ra rất quan trọng trong việc đạt được hiệu suất tốt từ thuật toán Machine learning mà chúng ta sẽ sử dụng. Bảng tương quan giữa các thuộc tính ở bên dưới sẽ chứng minh điều này.

Công việc kế đến là ta sẽ lưu dữ liệu đã được làm sạch để sử dụng khi xây dựng mô hình.

Sau khi thực hiện tiền xử lý dữ liệu, bộ dữ liệu mới thu được bao gồm các thuộc tính sau:

Bảng 4.7. Tên và kiểu dữ liệu của các thuộc tính sau khi thực hiện tiền xử lý dữ liệu

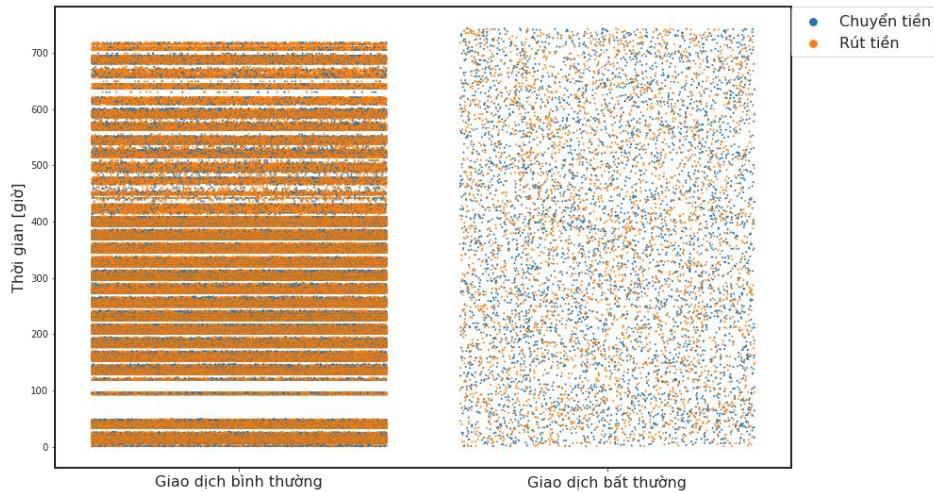
STT	Tên thuộc tính	Kiểu
1	step	int (64)
2	type	object
3	amount	float (64)
4	oldbalanceOrg	float (64)
5	newbalanceOrig	float (64)
6	oldbalanceDest	float (64)
7	newbalanceDest	float (64)
8	isFraud	int (64)
9	errorBalanceOrig	float (64)

10	errorBalanceD	float (64)
----	---------------	------------

#### 4.1.5. Trực quan hóa dữ liệu – Data visualization

##### 4.1.5.1. Phân tán theo thời gian

Biểu đồ thể hiện sự phân tán theo thời gian của giao dịch bình thường và giao dịch bất thường.



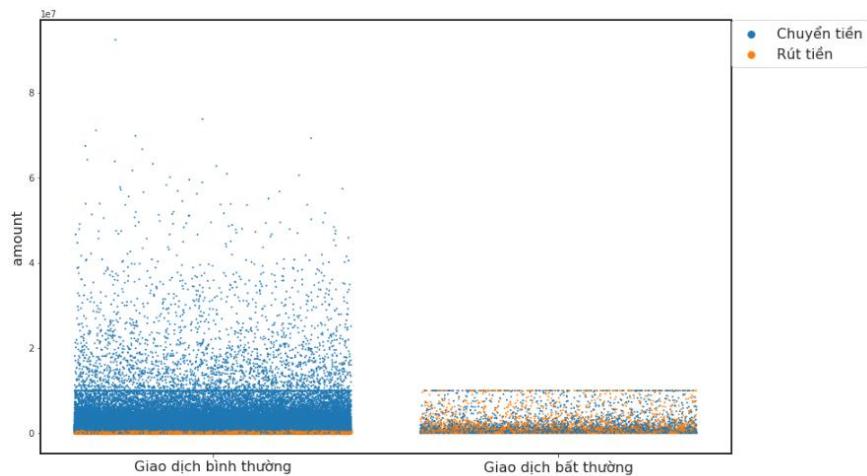
Hình 4.9. Sự phân tán theo thời gian của giao dịch bình thường và giao dịch bất thường

##### Nhận xét:

- Đối với dữ liệu giao dịch bình thường:
  - Rút tiền vượt trội hơn chuyển tiền.
  - Phân phối không đều theo thời gian.
- Đối với dữ liệu giao dịch bất thường:
  - Số lượng chuyển và rút tiền không chênh lệch nhiều.
  - Phân phối đều theo thời gian.

##### 4.1.5.2. Phân tán theo số tiền giao dịch

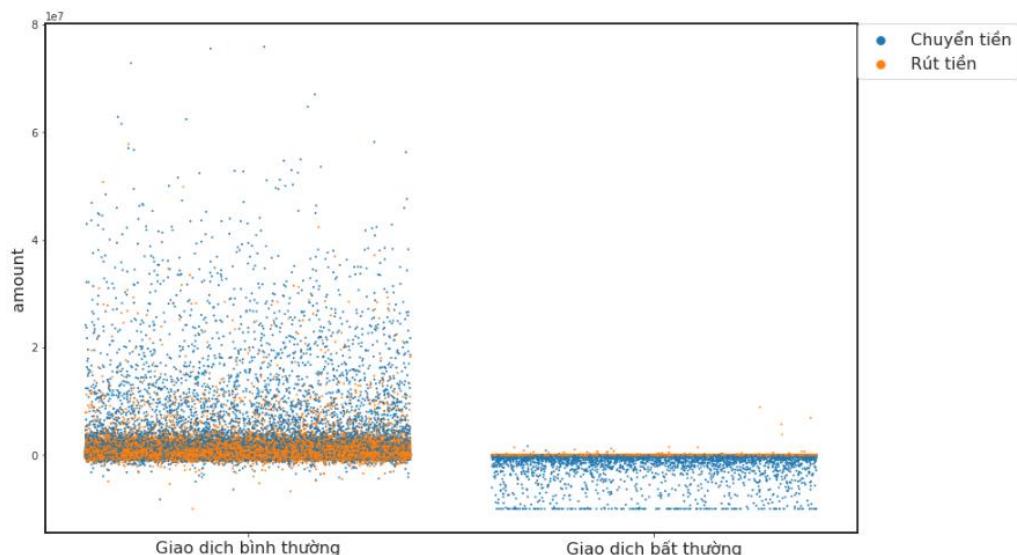
Biểu đồ thể hiện mật độ của các giao dịch theo số tiền:



Hình 4.10. Sự phân tán theo số tiền giao dịch

#### 4.1.5.3. Phân tán theo số tiền sai trong tài khoản đích

Biểu đồ thể hiện số tiền bị sai trong tài khoản đích khi thực hiện giao dịch:



Hình 4.11. Sự phân tán theo số tiền sai trong giao dịch chuyển tiền và rút tiền

**Nhận xét:**

- Với giao dịch không bình thường thì thuộc tính ‘amount’ có mối tương quan lớn với thuộc tính ‘errorBalanceOrig’.

**4.1.6. Xử lý dữ liệu mất cân bằng**

Thực hiện tách dữ liệu các giao dịch bình thường và bất thường, sau đó lấy ra các dòng dữ liệu là giao dịch bình thường. Từ kết quả chạy lệnh, ta thấy được **Số giao dịch bất thường** là 8213 giao dịch, độ mất cân bằng chiếm tỉ lệ 0.296%.

Có thể thấy, sau khi loại bỏ các giao dịch không liên quan và chỉ giữ lại những giao dịch có sự bất thường. Thực tế chỉ có gần 0.3% trên tổng số giao dịch xảy ra sự bất thường, quá chênh lệch so với số lượng giao dịch bình thường, do đó có thể thấy đây là tập dữ liệu rất mất cân bằng.

Qua quá trình tìm hiểu, nhóm tác giả lựa chọn các phương pháp sau để xử lý dữ liệu mất cân bằng:

- Oversampling.
- Undersampling.
- Phương pháp kết hợp (combined class methods) SMOTE + ENN.

**4.1.6.1. Random UnderSampling**

Trong giai đoạn này của dự án, nhóm sẽ triển khai “Lấy mẫu ngẫu nhiên”, về cơ bản bao gồm việc xóa dữ liệu để có bộ dữ liệu cân bằng hơn và do đó tránh các mô hình đưa ra quá phù hợp với các giao dịch bình thường.

Nhược điểm của việc lấy mẫu dưới là một mô hình được đào tạo theo cách này sẽ không hoạt động tốt trên dữ liệu kiểm tra sai lệch trong thế giới thực vì hầu như tất cả các thông tin đã bị loại bỏ.

Các bước thực hiện:

- Đầu tiên, xác định mức độ mất cân bằng của các lớp.
- Lấy số lượng mẫu giao dịch bình thường bằng số lượng mẫu giao dịch bất thường (50/50), trong trường hợp này cụ thể là lấy số giao dịch bình thường = số giao dịch bất thường = 8213 giao dịch.

- Sau khi có tập dữ liệu, tiến hành xáo trộn dữ liệu, mục đích để xem dữ liệu mô hình có duy trì được độ chính xác ổn định hay không.

Kết quả theo hình dưới đây (chi tiết lệnh xem phần phụ lục):

Số giao dịch còn lại: 16426										
	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	errorBalanceOrig	errorBalanceDest
615727	277	1	322907.44	-1.00	-1.0	593694.99	916602.43	0	322907.44	-1.164153e-10
497782	336	0	195255.28	8875.00	0.0	92278.38	287533.66	0	186380.28	5.820766e-11
400705	343	1	802417.45	802417.45	0.0	49645.31	852062.77	1	0.00	-1.000000e-02
2018388	241	0	57734.59	57734.59	0.0	-1.00	-1.00	1	0.00	5.773459e+04
2172362	99	1	286670.90	286670.90	0.0	1028298.11	1314969.01	1	0.00	0.000000e+00

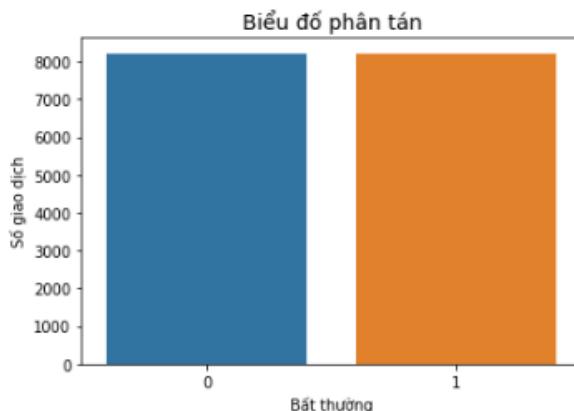
Hình 4.12. Kết quả khi tiến hành gộp dữ liệu giao dịch bình thường và bất thường

### Nhận xét:

- Phương pháp lấy mẫu ngẫu nhiên có thể xử lý dữ liệu mất cân bằng. Tuy nhiên, nó cũng mang lại nhiều rủi ro cho mô hình xây dựng vì có rất nhiều thông tin bị mất trong dữ liệu lượt bỏ (information loss).

Sau khi tiến hành cân bằng dữ liệu, tiếp đến sẽ bắt đầu phân tích và tiền xử lý trên dữ liệu mới.

Phân phối các lớp trong tập dữ liệu mẫu được thể hiện qua biểu đồ cột dưới đây:



Hình 4.13. Biểu đồ phân tán giữa các giao dịch bình thường và bất thường

#### 4.1.6.2. Random OverSampling

Oversampling: làm tăng trọng số của lớp thiểu số bằng cách sao chép các ví dụ của lớp thiểu số. Mặc dù nó không làm tăng thông tin, nhưng nó làm tăng vấn đề khớp

dữ liệu, khiến cho mô hình trở nên quá cụ thể. Nó cũng có thể là trường hợp độ chính xác cho tập huấn luyện cao, nhưng hiệu suất cho các bộ dữ liệu mới thực sự kém hơn. Random oversampling đơn giản là:

- Sao chép ngẫu nhiên các ví dụ lớp thiểu số.
- Làm tăng khả năng xảy ra quá mức.
- Một khía cạnh, nhược điểm lớn của việc lấy mẫu ngẫu nhiên là phương pháp này có thể loại bỏ dữ liệu hữu ích.

Tiến hành thực hiện các bước sau:

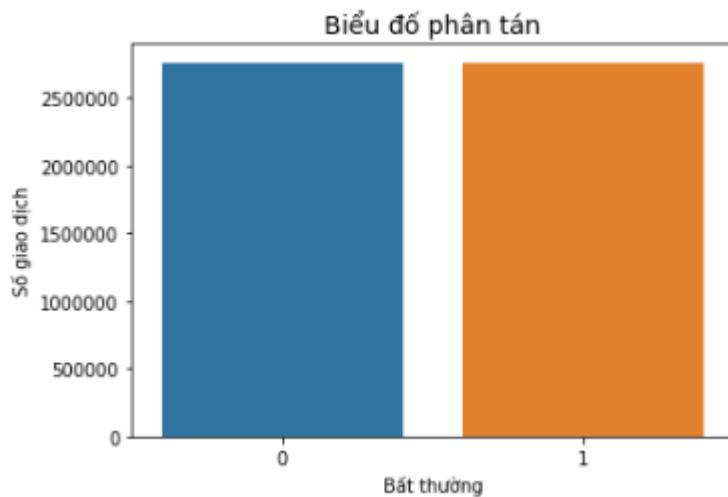
- Tách dữ liệu giao dịch bình thường và giao dịch bất thường.
- Lấy ra các dòng là giao dịch bình thường.
- Chạy thuật toán Random undersampling.
- Chuyển dữ liệu.
- Gộp dữ liệu giao dịch bất thường thành giao dịch bình thường.
- Xáo trộn dữ liệu.

Kết quả sau khi thực hiện được cho trong hình dưới đây (chi tiết lệnh xem phần phụ lục):

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	errorBalanceOrig	errorBalanceDest	isFraud	
1980434	331.0	1.0	139424.89		-1.00	-1.0	553115.08	692539.97	139424.89	0.00	0
4643802	26.0	1.0	37065.55	37065.55		0.0	1396327.66	1433393.21	0.00	0.00	1
1934508	135.0	1.0	520593.63		-1.00	-1.0	3780877.21	4557483.47	520593.63	-256012.63	0
5442216	390.0	0.0	1084144.37	1084144.37		0.0	-1.00	-1.00	0.00	1084144.37	1
2906542	520.0	0.0	5469910.99	5469910.99		0.0	-1.00	-1.00	0.00	5469910.99	1

Hình 4.14. Kết quả khi gộp dữ liệu giữa giao dịch bình thường và bất thường sử dụng Random Oversampling

Biểu đồ thể hiện phân phối các lớp trong tập dữ liệu mẫu:



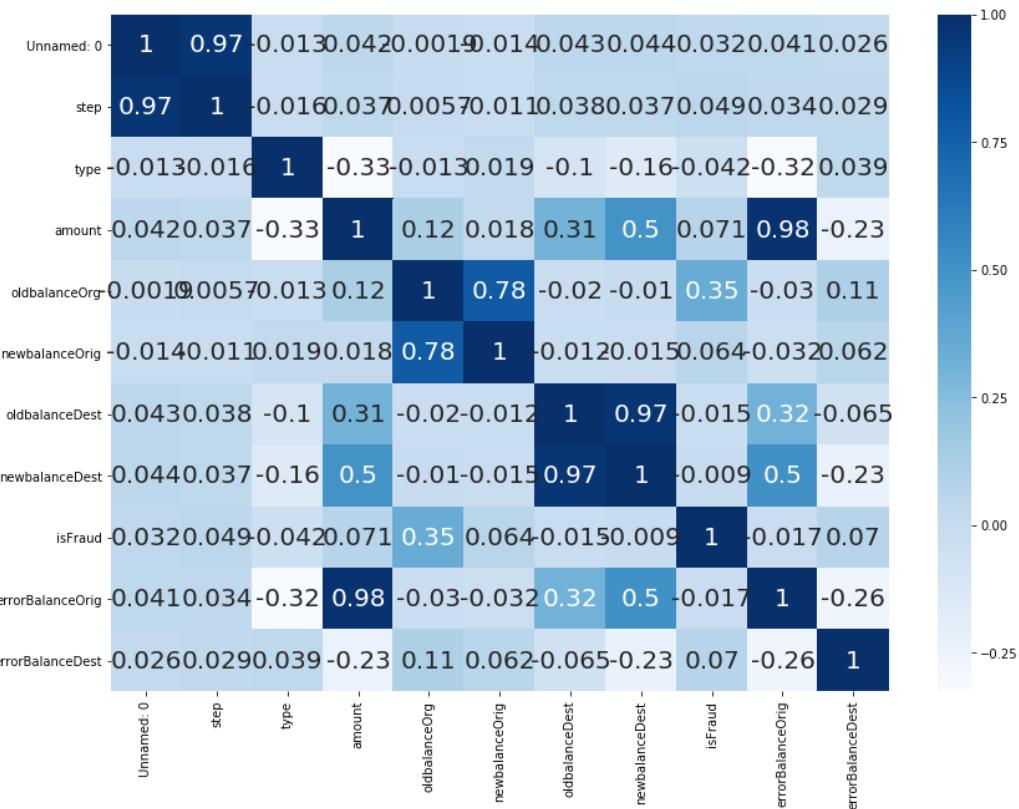
Hình 4.15. Biểu đồ phân tán các lớp của giao dịch bất thường và bình thường sau khi cân bằng dữ liệu

#### 4.1.7. Ma trận tương quan (Correlation Matrices)

- Mục đích để biết liệu có những thuộc tính nào có ảnh hưởng lớn đến việc một giao dịch cụ thể có phải là giao dịch bất thường hay không. Tuy nhiên, điều quan trọng ở đây là sẽ sử dụng đúng khung dữ liệu (tập mẫu) để xem các tính năng nào có mối tương quan tích cực hoặc tiêu cực cao liên quan đến các giao dịch bất thường.
- Ma trận tương quan là một ma trận thể hiện các hệ số tương quan giữa các biến. Mỗi ô trong bảng hiển thị mối tương quan giữa hai biến.
- Một ma trận tương quan được sử dụng để tóm tắt dữ liệu, làm đầu vào cho một phân tích nâng cao hơn và như một chẩn đoán cho các phân tích nâng cao.

#### 4.1.7.1. Dữ liệu chưa xử lý cân bằng

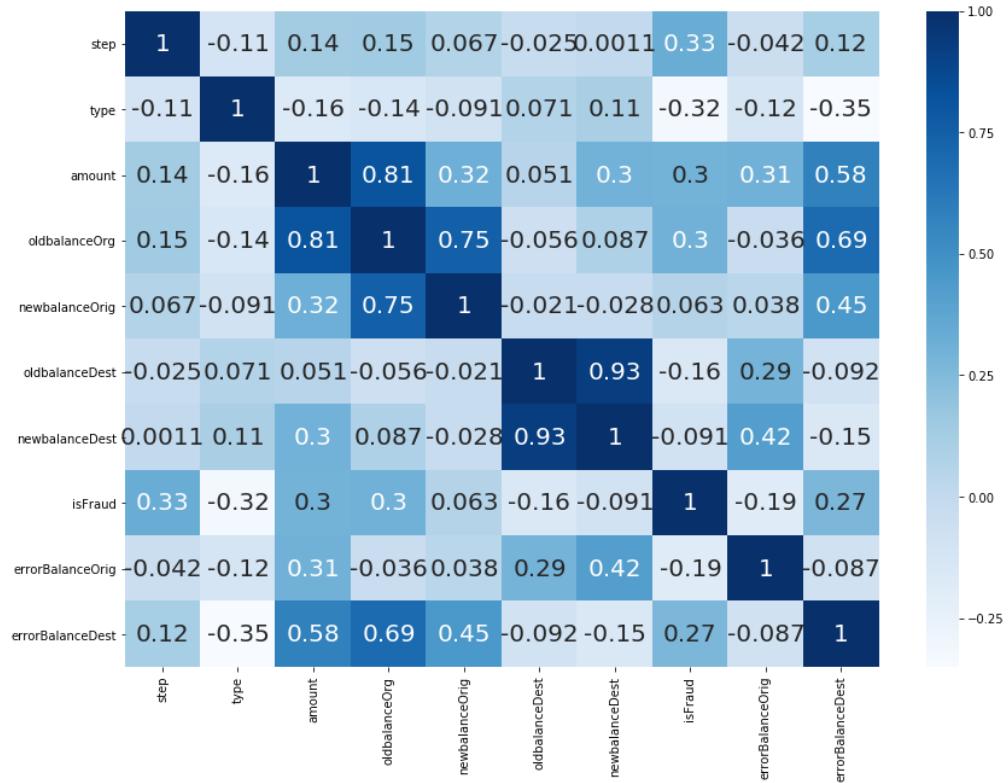
Ma trận tương quan các thuộc tính đối với dữ liệu chưa xử lý cân bằng:



Hình 4.16. Ma trận tương quan đối với dữ liệu chưa xử lý cân bằng

#### 4.1.7.2. Dữ liệu cân bằng theo phương pháp UnderSampling

Ma trận tương quan đối với dữ liệu cân bằng sử dụng UnderSampling được hiển thị:



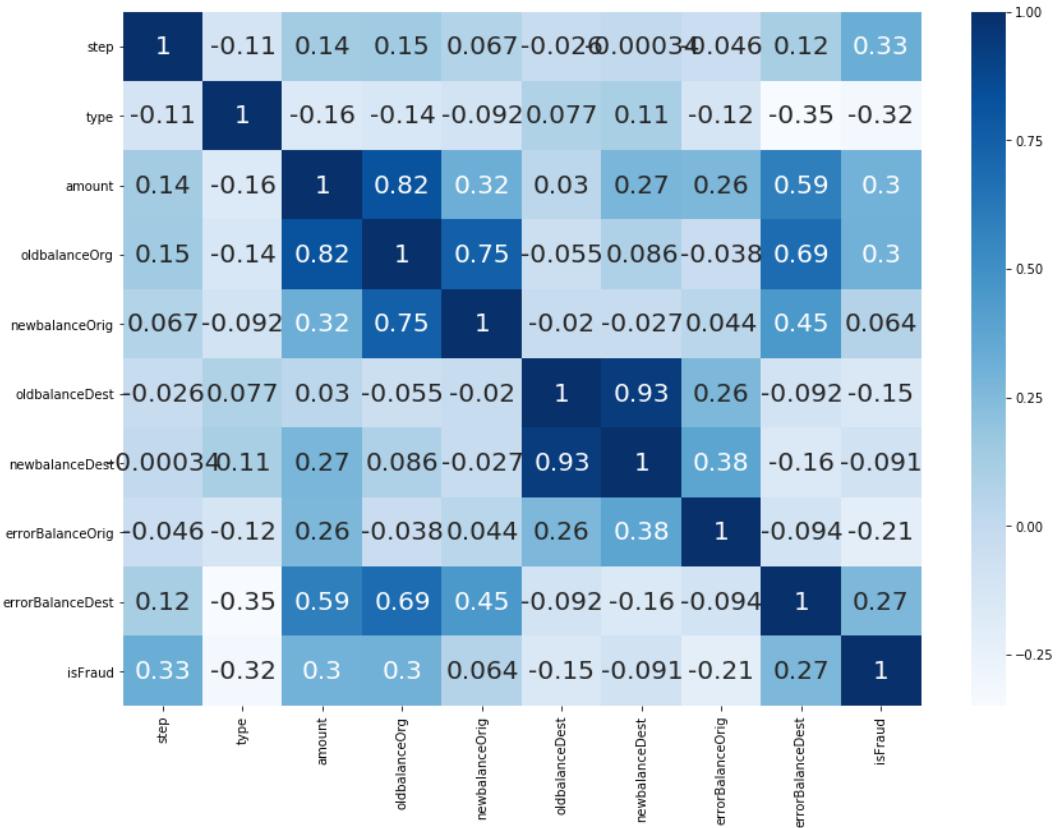
Hình 4.17. Ma trận tương quan đối với dữ liệu xử lý cân bằng theo phương pháp Undersampling

#### Nhận xét:

- Tương quan âm: có thể thấy thuộc tính ‘errorBalanceOrig’, ‘newbalanceDest’ có tương quan ngược cao nhất với biến phân loại mà nhóm đã chọn.
- Lưu ý rằng các giá trị này càng thấp thì kết quả cuối cùng sẽ là một giao dịch bất thường.
- BoxPlots: Nhóm sẽ sử dụng biểu đồ boxplots để hiểu rõ hơn về việc phân phối các thuộc tính này trong các giao dịch bất thường và giao dịch bình thường.

#### 4.1.7.3. Dữ liệu cân bằng theo phương pháp Oversampling

Mã trận tương quan của dữ liệu cân bằng theo phương pháp Oversampling:



Hình 4.18. Ma trận tương quan khi sử dụng dữ liệu cân bằng theo phương pháp oversampling

#### 4.1.8. Phát hiện bất thường (Anomaly detection)

Mục đích trong phần này sẽ loại bỏ các ‘dữ liệu ngoại vi (outliers)’ khỏi các thuộc tính có mối tương quan cao với các lớp giao dịch bình thường và bất thường. Điều này sẽ có tác động tích cực đến độ chính xác của các mô hình khi thực hiện phát hiện sự bất thường.

Tuy nhiên, cần phải cẩn thận về ngưỡng mà chúng ta muốn loại bỏ để loại bỏ các ngoại lệ.

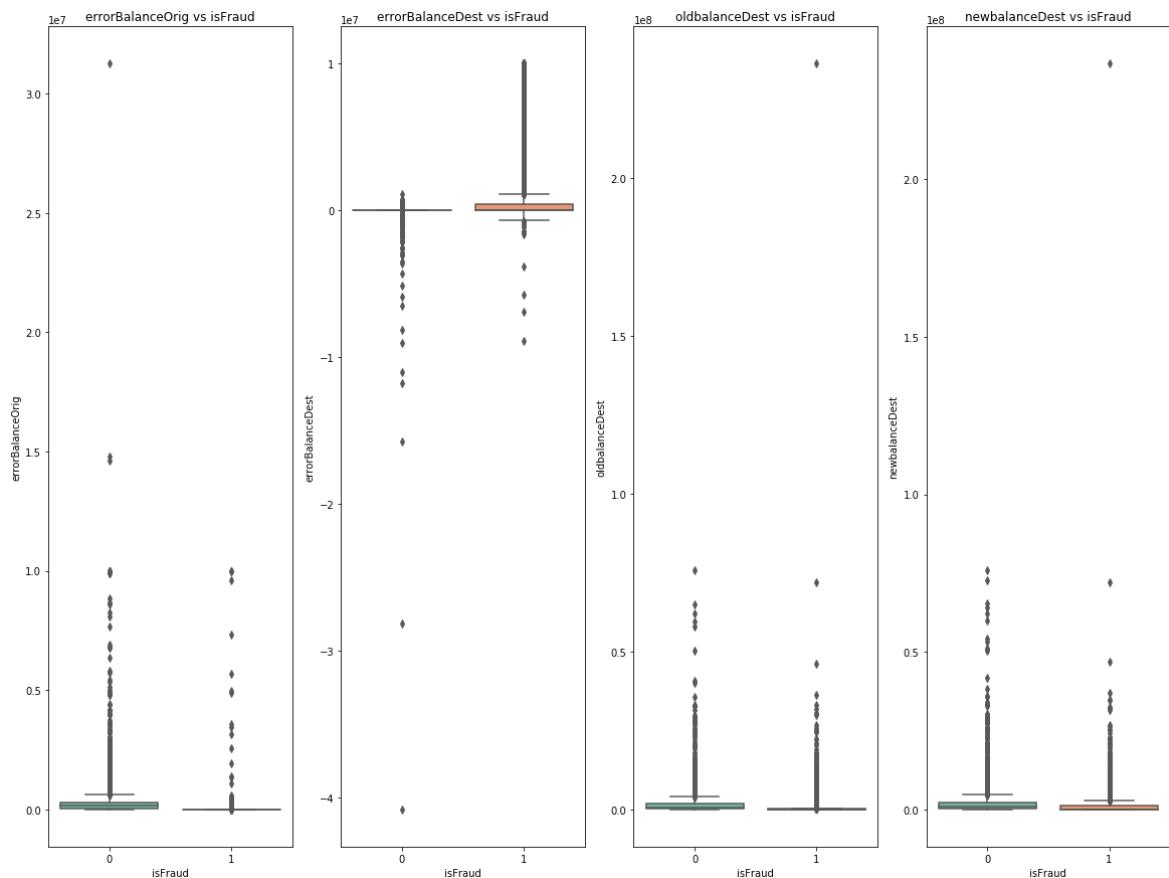
Việc xác định ngưỡng bằng cách nhân cho một số (ví dụ: 1.5) với phạm vi liên dải. Ngưỡng này càng cao, càng ít ngoại lệ sẽ phát hiện và ngưỡng này càng thấp thì càng phát hiện ra nhiều ngoại lệ.

## Mục tiêu:

- Nhóm sẽ tập trung vào các ngoại lệ cực đoan thay vì chỉ là các ngoại lệ. Bởi vì, có thể gặp rủi ro mất thông tin, điều này sẽ khiến các mô hình mô phỏng có độ chính xác thấp hơn.
- Xem ngưỡng sẽ ảnh hưởng đến độ chính xác của các mô hình phân loại.

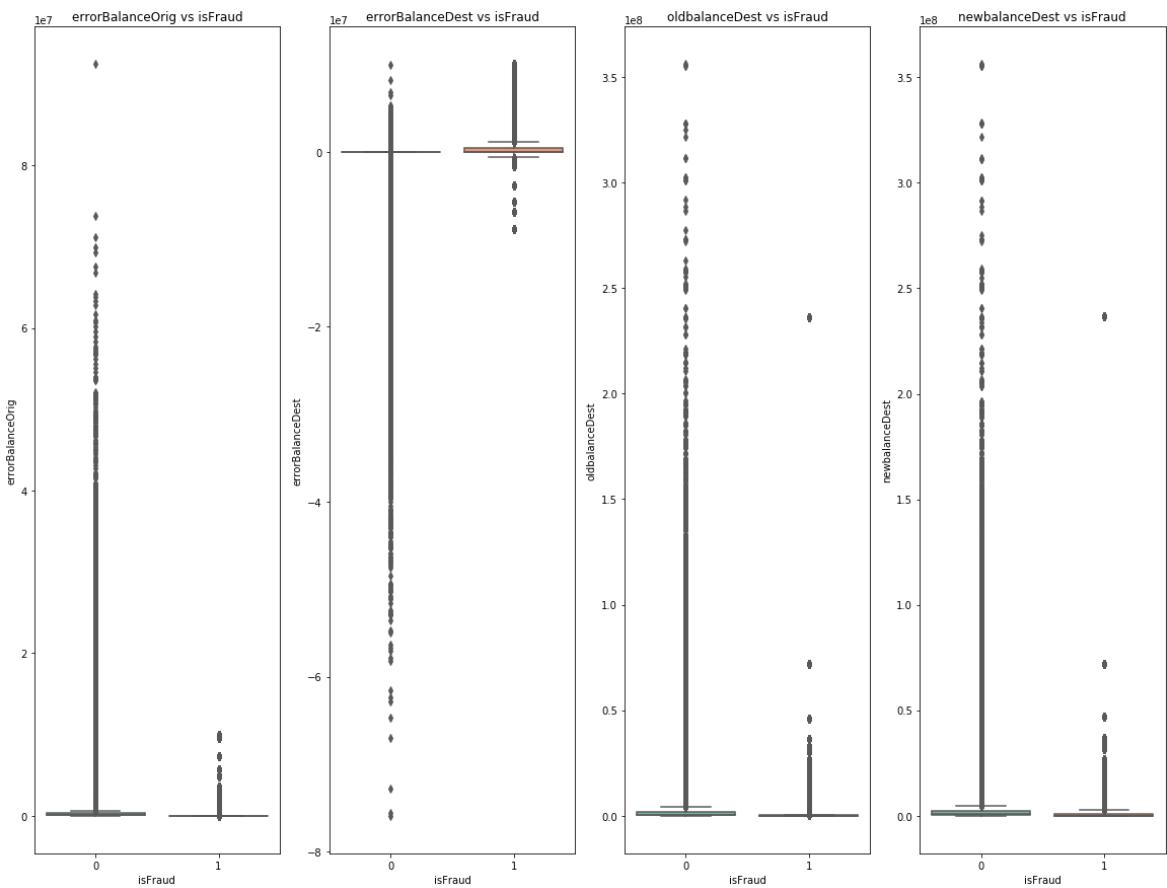
### 4.1.8.1. Phân phối trước khi loại bỏ các giá trị ngoại vi

Đối với dữ liệu được xử lý cân bằng dùng phương pháp Undersampling:



Hình 4.19. Thể hiện phân phối trước khi loại bỏ các giá trị ngoại vi của bộ dữ liệu dùng undersampling

Đối với dữ liệu được xử lý cân bằng dùng phương pháp Oversampling:



Hình 4.20. Thể hiện phân phối trước khi loại bỏ các giá trị ngoại vi của bộ dữ liệu dùng oversampling

#### 4.1.8.2. Loại bỏ các giá trị ngoại vi

##### 4.1.8.2.1. Dữ liệu xử lý bằng phương pháp Undersampling

Thực hiện tìm giá trị ngoại vi và loại bỏ giá trị ngoại vi của các thuộc tính errorBalanceOrig, errorBalanceDest, oldbalanceDest cho dữ liệu xử lý bằng phương pháp Undersampling.

Quá trình thực hiện cho thuộc tính errorBalanceOrig, các thuộc tính còn lại thực hiện tương tự, thu được kết quả như sau:

```
Bách phân vị 25: 0.0 | Bách phân vị 75: 0.0
iqr: 0.0
Biên dưới: 0.0
Biên trên: 0.0
```

Tính năng 'errorBalanceOrig' có giá trị ngoại vi cho trường hợp là giao dịch gian lận: 57

Giá trị ngoại vi:[3441041.46, -3.725290298461914e-09, 43728.5, 229909.57, 23292.3, 1395850.55, 10000000.0, -7.450580596923828e-09, 94372.61, 40611.22, 3171085.59, 181728.11, 149668.66, 4953893.08, 536624.41, 202978.65, 3.725290298461914e-09, 95428.32, 454859.39, 9585040.370000001, 85065.86, 122101.57, 1078013.76, -2.3283064365386963e-10, 416001.33, 277970.88, 9999999.99999998, 508782.2, 353874.22, 399045.0800000001, 577418.98, 3.725290298461914e-09, 2542664.27, 3.725290298461914e-09, 39713.28, 238295.11000000002, 234377.29, 4892193.09, 1.862645149230957e-09, 112280.88, 7.450580596923828e-09, 7316255.050000001, -3.725290298461914e-09, 291519.84, -1.862645149230957e-09, 3.725290298461914e-09, 407005.78, 128343.56000000001, 332729.54, 112486.46, 1933920.8, 10000000.0, 222048.71, 3.725290298461914e-09, 3576297.1, 5674547.89, 1343002.08]

#### 4.1.8.2.2. Dữ liệu xử lý bằng phương pháp Oversampling

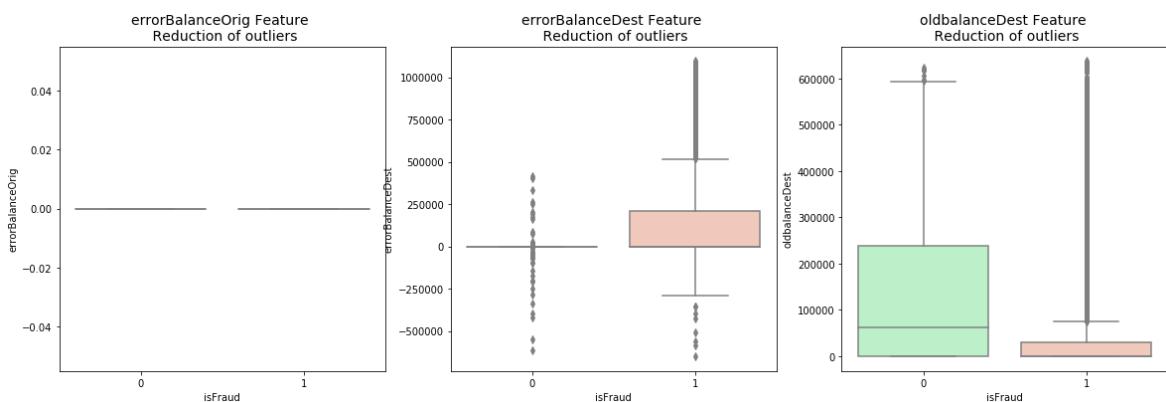
Thực hiện tìm giá trị ngoại vi và loại bỏ giá trị ngoại vi của các thuộc tính errorBalanceOrig, errorBalanceDest, oldbalanceDest cho dữ liệu xử lý bằng phương pháp Oversampling.

Tương tự như thực hiện đối với bộ dữ liệu xử lý bằng Undersampling, chi tiết quá trình thực hiện và kết quả cho các thuộc tính tham khảo phần phụ lục.

#### 4.1.8.3. Phân phối sau khi loại bỏ giá trị ngoại vi

##### 4.1.8.3.1. Dữ liệu xử lý bằng phương pháp Undersampling

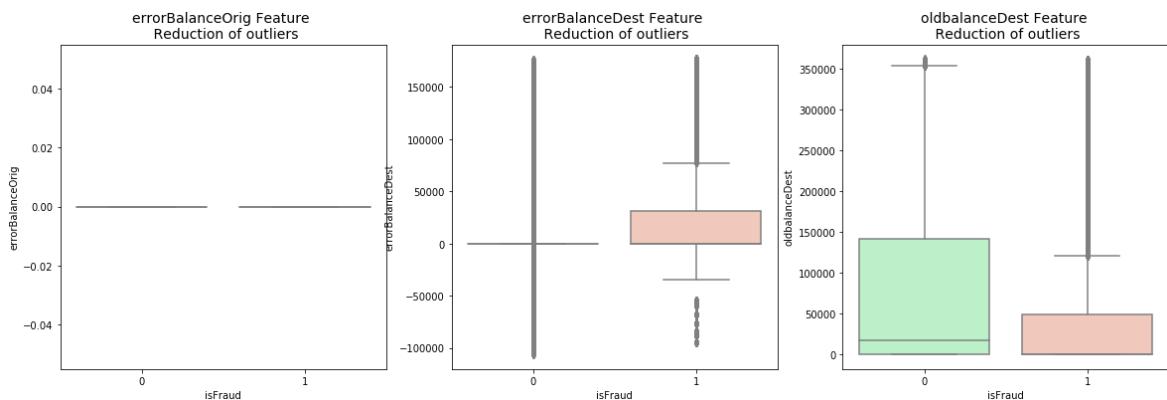
Hình 4.21 cho thấy phân phối của dữ liệu xử lý cân bằng dùng Undersampling:



Hình 4.21. Hình phân phối của bộ dữ liệu xử lý cân bằng dùng Undersampling

#### 4.1.8.3.2. Dữ liệu xử lý bằng phương pháp Oversampling

Hình 4.22 cho thấy phân phối của dữ liệu xử lý cân bằng dùng Oversampling:



Hình 4.22. Bảng phân phối của bộ dữ liệu xử lý cân bằng dùng  
Oversampling

#### 4.1.9. Lưu dữ liệu sau khi thực hiện cân bằng dữ liệu

Tiến hành lưu dữ liệu sau khi thực hiện cân bằng dữ liệu bằng ba phương pháp trên, dữ liệu này được sử dụng để huấn luyện (train) (chi tiết lệnh xem phần phụ lục).

Sau đó ta tiến hành thực hiện các bước sau:

- Tách các thuộc tính độc lập và phụ thuộc ra.
- Chia tập dữ liệu thành 2 phần: train và test với tỷ lệ 80% : 20%.
- Chuyển các giá trị thành một mảng để đưa vào các thuật toán (thuật toán XGBoost không cần thực hiện bước này).

#### 4.1.10. Mô hình phát hiện bất thường sử dụng thuật toán Random forest

##### 4.1.10.1. Với bộ dữ liệu chưa xử lý mất cân bằng

Thực hiện tách dữ liệu thành hai tập: tập train và test, sau đó sử dụng xác thực chéo trên tập train để điều chỉnh các tham số, số lượng các giao dịch của mỗi phân lớp theo thuộc tính isFraud trong trường hợp bộ dữ liệu chưa được xử lý cân bằng được liệt kê theo bảng sau:

Bảng 4.8. Thống kê số lượng của mỗi giá trị ở thuộc tính isFraud theo bộ dữ liệu

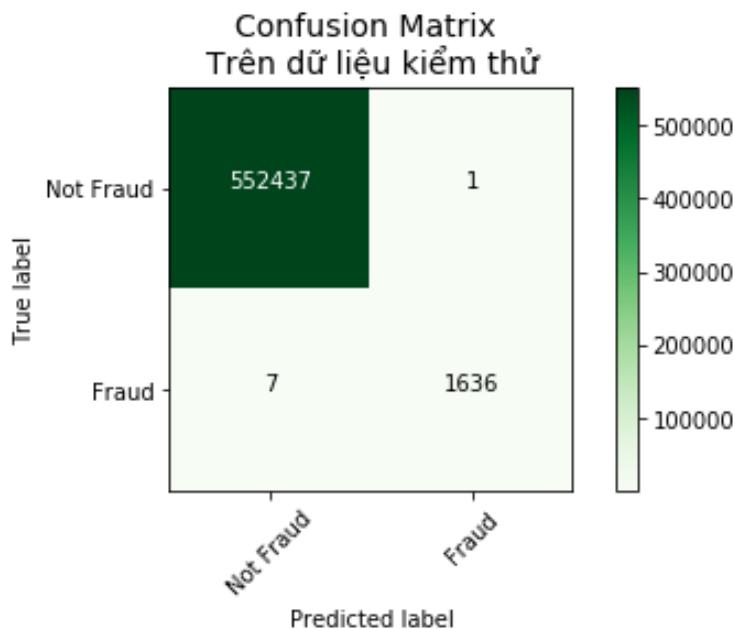
Loại xử lý của bộ dữ liệu	Tập dữ liệu	Số lượng của mỗi phân lớp của thuộc tính isFraud	
		isFraud = 0 (giao dịch bình thường)	isFraud = 1 (giao dịch bất thường)
Chưa xử lý cân bằng	Ban đầu	2762196	8213
	Tập train	2209757	6570
	Tập test	552439	1643

Ta tiến hành chạy mô hình bằng thuật toán Random forest cho bộ dữ liệu, sử dụng 3 fold-cross validation, ta có kết quả được cho theo bảng dưới đây (chi tiết xem phần phụ lục):

Bảng 4.9. Kết quả sau khi chạy thuật toán Random forest đối với bộ dữ liệu chưa cân bằng

Lần chạy	Độ chính xác trên tập train	Độ chính xác trên tập test	F1-score trên tập train	F1-score trên tập test	Thời gian
1	99.9993%	99.9986%			
2	99.9981%	99.9984%			
3	99.9986%	99.9986%			
<b>Trung bình</b>	<b>99.9987%</b>	<b>99.9985%</b>	0.9978	0.9975	399.3637 giây

Ma trận nhầm lẫn trên tập dữ liệu test (chi tiết lệnh xem ở phụ lục):



Hình 4.23. Ma trận nhầm lẫn tập dữ liệu test của bộ dữ liệu chưa xử lý cân bằng

### Kết quả:

- Có 552437 giao dịch bình thường được dự đoán đúng là giao dịch bình thường.
- 1 lần giao dịch bình thường được dự đoán sai là giao dịch bất thường.
- 7 lần giao dịch bất thường được dự đoán sai là giao dịch bình thường.
- 1636 lần giao dịch bất thường được dự đoán chính xác là giao dịch bất thường.

#### 4.1.10.2. Với bộ dữ liệu đã xử lý cân bằng

Lấy bộ dữ liệu được xử lý cân bằng dùng phương pháp Oversampling, tiến hành xem lại các thông tin:

	Unnamed: 0	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	errorBalanceOrig	errorBalanceDest	isFraud
0	3481951	576.0	1.0	451399.66	451399.66	0.00	215513.02	666912.68	0.0	-1.164153e-10	1
1	1494917	178.0	1.0	408401.45	689392.00	280990.55	147126.43	555527.88	0.0	0.000000e+00	0
2	4494752	496.0	0.0	174725.11	174725.11	0.00	-1.00	-1.00	0.0	1.747251e+05	1
3	4345240	273.0	0.0	76642.85	76642.85	0.00	-1.00	-1.00	0.0	7.664285e+04	1
4	4593205	4.0	0.0	169941.73	169941.73	0.00	-1.00	-1.00	0.0	1.699417e+05	1

Hình 4.24. Mô phỏng bộ dữ liệu được xử lý cân bằng dùng Oversampling

Tiếp theo thực hiện tách các thuộc tính độc lập và thuộc tính phụ thuộc, sau đó thực hiện chia làm hai phần: tập train và test với tỉ lệ: 80% train và 20% test, chạy lệnh để kiểm tra dữ liệu ta được kết quả:

Bảng 4.10. Thông kê số lượng của mỗi giá trị ở thuộc tính isFraud theo bộ dữ liệu

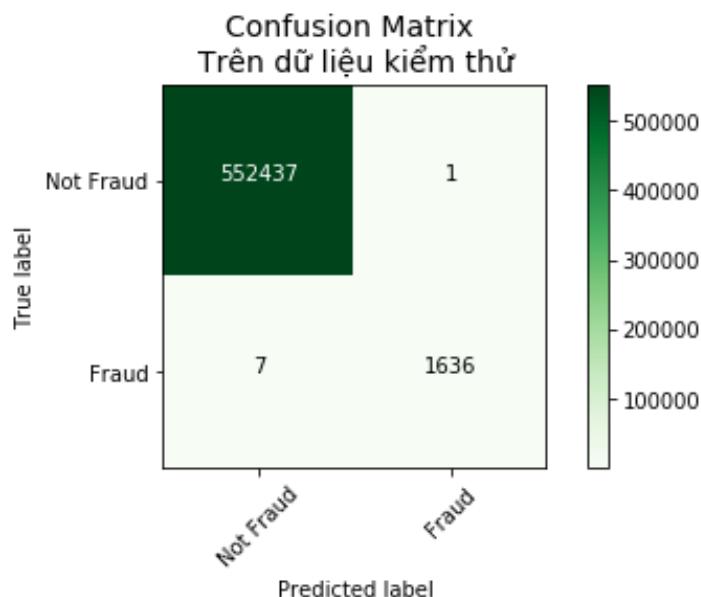
Loại xử lý của bộ dữ liệu	Tập dữ liệu	Số lượng của mỗi phân lớp của thuộc tính isFraud	
		isFraud = 0 (giao dịch bình thường)	isFraud = 1 (giao dịch bất thường)
Đã xử lý cân bằng theo phương pháp random over sampling.	Ban đầu đã xử lý cân bằng	135626	1283313
	Tập train	108569	1026582
	Tập test	27057	256731

Ta tiến hành chạy mô hình bằng thuật toán Random forest cho bộ dữ liệu, thu được kết quả (chi tiết xem phần phụ lục):

Bảng 4.11. Kết quả sau khi chạy thuật toán Random forest đối với bộ dữ liệu đã xử lý cân bằng dùng Oversampling

Lần chạy	Độ chính xác tập train	Độ chính xác tập test	F1-score trên tập train	F1-score trên tập test	Thời gian
1	99.9997%	100.0%			
2	100.0%	100.0%			
3	100.0%	100.0%			
<b>Trung bình</b>	<b>99.9999%</b>	<b>100.0%</b>	<b>1</b>	<b>1</b>	<b>50.1045 giây</b>

Ma trận trên tập dữ liệu test:



Hình 4.25. Ma trận nhầm lẫn trên tập dữ liệu test dùng Oversampling

### Kết quả:

- Có 552437 giao dịch bình thường được dự đoán đúng là giao dịch bình thường.
- 1 lần giao dịch bình thường được dự sai đoán là giao dịch bất thường.

- 7 lần giao dịch bất thường được dự đoán sai là giao dịch bình thường.
- 1636 lần giao dịch bất thường được dự đoán chính xác là giao dịch bất thường.

Lấy bộ dữ liệu được xử lý cân bằng dùng phương pháp Undersampling, tiến hành xem lại các thông tin, xem một số dòng của bộ dữ liệu:

	Unnamed: 0	Unnamed: 0.1	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	errorBalanceOrig	errorBalanceDest
0	400705	4785673	343	1	802417.45	802417.45	0.0	49645.31	852062.77	1	0.0	-0.01
1	2018388	3192449	241	0	57734.59	57734.59	0.0	-1.00	-1.00	1	0.0	57734.59
2	1062282	142088	11	1	12461.00	12461.00	0.0	27964.11	40425.11	1	0.0	0.00
3	2561624	5099867	355	1	42483.97	42483.97	0.0	51495.75	93979.72	1	0.0	0.00
4	71426	6118821	537	0	34238.20	34238.20	0.0	-1.00	-1.00	1	0.0	34238.20

Hình 4.26. Mô phỏng bộ dữ liệu được xử lý cân bằng dùng Undersampling

Tiếp theo thực hiện tách các thuộc tính độc lập và thuộc tính phụ thuộc, sau đó thực hiện chia làm hai phần: tập train và test với tỉ lệ: 80% train và 20% test, chạy lệnh để kiểm tra dữ liệu ta được kết quả:

Bảng 4.12. Thống kê số lượng giao dịch của mỗi giá trị ở thuộc tính isFraud theo bộ dữ liệu

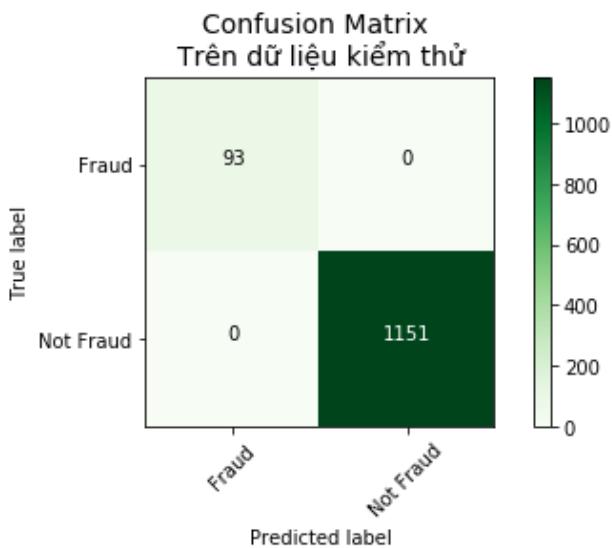
Loại xử lý của bộ dữ liệu	Tập dữ liệu	Số lượng của mỗi phân lớp của thuộc tính isFraud	
		isFraud = 0 (giao dịch bình thường)	isFraud = 1 (giao dịch bất thường)
Đã xử lý cân bằng theo phương pháp random undersampling.	Ban đầu đã xử lý cân bằng	492	5733
	Tập train	398	4582
	Tập test	94	1151

Ta tiến hành chạy mô hình bằng thuật toán Random forest sử dụng 3 fold-cross validation cho bộ dữ liệu, thu được kết quả (chi tiết xem phần phụ lục):

Bảng 4.13. Kết quả sau khi chạy thuật toán Random forest đối với bộ dữ liệu đã xử lý cân bằng dùng Undersampling

Lần chạy	Độ chính xác trên tập train	Độ chính xác trên tập test	F1-score trên tập train	F1-score trên tập test	Thời gian
1	99.9398%	99.9197%			
2	100%	100%			
3	99.9398%	100%			
<b>Trung bình</b>	<b>99.9598%</b>	<b>99.9732%</b>	0.9998	0.9999	1.4243 giây

Ma trận trên tập dữ liệu test:



Hình 4.27. Ma trận nhầm lẫn trên tập dữ liệu test dùng Undersampling

### Kết quả:

- Có 1151 giao dịch bình thường được dự đoán đúng là giao dịch bình thường.
- 0 lần giao dịch bình thường được dự sai đoán là giao dịch bất thường.
- 0 lần giao dịch bất thường được dự đoán sai là giao dịch bình thường.
- 93 lần giao dịch bất thường được dự đoán chính xác là giao dịch bất thường.

#### 4.1.11. Mô hình phát hiện bất thường sử dụng thuật toán Extreme Gradient Boosting (EGB)

##### 4.1.11.1. Với bộ dữ liệu chưa xử lý mất cân bằng

Lấy trên bộ dữ liệu chưa xử lý cân bằng, tiếp theo thực hiện tách các thuộc tính độc lập và thuộc tính phụ thuộc, sau đó thực hiện chia làm hai phần: tập train và test với tỉ lệ: 70% train và 30% test, chạy lệnh để kiểm tra dữ liệu ta được kết quả:

Bảng 4.14. Thống kê số lượng giao dịch của mỗi giá trị ở thuộc tính isFraud  
theo bộ dữ liệu

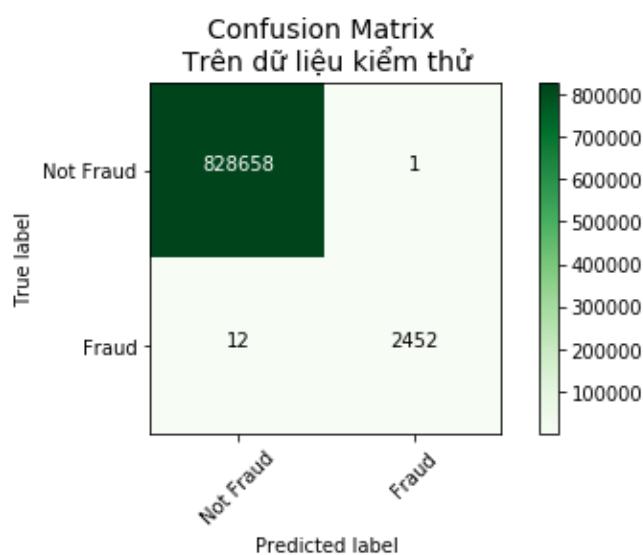
Loại xử lý của bộ dữ liệu	Tập dữ liệu	Số lượng của mỗi phân lớp của thuộc tính isFraud	
		isFraud = 0 (giao dịch bình thường)	isFraud = 1 (giao dịch bất thường)
Chưa xử lý cân bằng	Ban đầu đã xử lý cân bằng	2762196	8213
	Tập train	1933537	5749
	Tập test	828659	2464

Tiến hành chạy mô phỏng thuật toán extreme gradient boosting, sử dụng 3-fold cross validation, thu được kết quả sau (chi tiết lệnh thực thi xem phần phụ lục):

Bảng 4.15. Bảng kết quả sau khi chạy thuật toán EGB đối với bộ dữ liệu chưa thực hiện xử lý cân bằng

Lần chạy	Độ chính xác trên tập train	Độ chính xác trên tập test	F1-score trên tập train	F1-score trên tập test	Thời gian
1	99.9980%	99.9983%			
2	99.9988%	99.9986%			
3	99.9989%	99.9984%			
<b>Trung bình</b>	<b>99.9986%</b>	<b>99.9984%</b>	<b>0.9976</b>	<b>0.9974</b>	<b>257.5319 giây</b>

Ma trận trên tập dữ liệu test như sau:



Hình 4.28. Ma trận tương quan trên tập dữ liệu test dùng bộ dữ liệu chưa cân bằng

### Nhận xét:

- Có 828658 giao dịch bình thường được dự đoán đúng là giao dịch bình thường.
- 1 lần giao dịch bình thường được dự đoán sai là giao dịch bất thường.
- 12 lần giao dịch bất thường được dự đoán sai là giao dịch bình thường.
- 2452 lần giao dịch bất thường được dự đoán chính xác là giao dịch bất thường.

#### 4.1.11.2. Với bộ dữ liệu đã xử lý cân bằng

Lấy bộ dữ liệu được xử lý cân bằng dùng phương pháp Oversampling, tiến hành xem lại các thông tin:

	Unnamed: 0	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	errorBalanceOrig	errorBalanceDest	isFraud
0	3481951	576.0	1.0	451399.66	451399.66	0.00	215513.02	666912.68	0.0	-1.164153e-10	1
1	1494917	178.0	1.0	408401.45	689392.00	280990.55	147126.43	555527.88	0.0	0.000000e+00	0
2	4494752	496.0	0.0	174725.11	174725.11	0.00	-1.00	-1.00	0.0	1.747251e+05	1
3	4345240	273.0	0.0	76642.85	76642.85	0.00	-1.00	-1.00	0.0	7.664285e+04	1
4	4593205	4.0	0.0	169941.73	169941.73	0.00	-1.00	-1.00	0.0	1.699417e+05	1

Hình 4.29. Mô phỏng bộ dữ liệu được xử lý cân bằng dùng Oversampling

Tiếp theo ta thực hiện tách các thuộc tính độc lập và thuộc tính phụ thuộc, sau đó thực hiện chia làm hai phần: tập train và test, chạy lệnh để kiểm tra dữ liệu ta được kết quả:

Bảng 4.16. Thống kê số lượng giao dịch của mỗi giá trị ở thuộc tính isFraud theo bộ dữ liệu

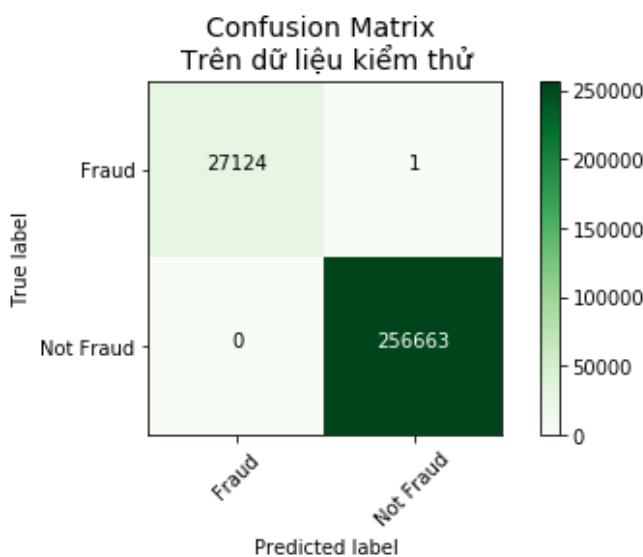
Loại xử lý của bộ dữ liệu	Tập dữ liệu	Số lượng của mỗi phân lớp của thuộc tính isFraud	
		isFraud = 0 (giao dịch bình thường)	isFraud = 1 (giao dịch bất thường)
Đã xử lý cân bằng theo phương pháp random over sampling.	Ban đầu đã xử lý cân bằng	135626	1283313
	Tập train	108501	1026650
	Tập test	27125	256663

Ta tiến hành chạy mô hình bằng thuật toán XGB sử dụng 3 fold-cross validation cho bộ dữ liệu, thu được kết quả (chi tiết lệnh xem phần phụ lục):

Bảng 4.17. Bảng kết quả sau khi chạy thuật toán EGB đối với bộ dữ liệu  
được xử lý dùng oversampling

Lần chạy	Độ chính xác trên tập train	Độ chính xác trên tập test	F1-score trên tập train	F1-score trên tập test	Thời gian
1	100%	99.9996%			
2	100%	99.9996%			
3	99.9997%	99.9996%			
<b>Trung bình</b>	<b>99.9999%</b>	<b>99.9996%</b>	<b>1</b>	<b>1</b>	<b>73.5196 giây</b>

Ma trận tương quan trên tập dữ liệu test đối với bộ dữ liệu được xử lý cân bằng dùng oversampling:



Hình 4.30. Ma trận tương quan trên bộ dữ liệu test dùng bộ dữ liệu đã thực hiện oversampling

### Nhận xét:

- Có 256663 giao dịch bình thường được dự đoán đúng là giao dịch bình thường.

- 0 lần giao dịch bình thường được dự đoán sai là giao dịch bất thường.
- 1 lần giao dịch bất thường được dự đoán sai là giao dịch bình thường.
- 27124 lần giao dịch bất thường được dự đoán chính xác là giao dịch bất thường.

## 4.2. Bộ dữ liệu Credit Card Fraud Detection

### 4.2.1. Các bước thực hiện

Tương tự như đối với bộ dữ liệu thứ nhất, trải qua các bước:

- Bước 1: Import library.
- Bước 2: Connect to google drive.
- Bước 3: Đọc dữ liệu vừa kết nối.

Ta được kết quả như sau:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	-0.551600	-0.617801	-0.991390	-0.311169	1.468177	-0.470401	0.207971
1	0.0	1.191857	0.266151	0.165480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	1.612727	1.065235	0.489095	-0.143772	0.635558	0.463917	-0.114805
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	0.624501	0.066084	0.717293	-0.165946	2.345865	-2.890083	1.109699
3	1.0	-0.9666272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	-0.226487	0.178228	0.507757	-0.287924	-0.631418	-1.059647	-0.684093
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	-0.822843	0.538196	1.345852	-1.119670	0.175121	-0.451449	-0.237033

Hình 4.31. Mô phỏng bộ dữ liệu Credit Card Fraud Detection ban đầu

### 4.2.2. Tìm hiểu tổng quan dữ liệu

#### a. Kiểu dữ liệu các thuộc tính

Lệnh	Thuộc tính	Kiểu dữ liệu
#Hiển thị kiểu dữ liệu của các thuộc tính data.dtypes	Time	float (64)
	V1, V2, V3, V4, ..., V28	float (64)
	Amount	float (64)
	Class	int (64)

## b. Kích thước dữ liệu

Tiến hành chạy lệnh sau:

Lệnh	Mô tả
# Kích thước dữ liệu data.shape	Dữ liệu có 284807 dòng, và 31 cột

## c. Tổng quan các giá trị trong bộ dữ liệu

Hình dưới đây mô tả giá trị của toàn bộ dữ liệu:

Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
count	284807.000000	2.848070e+05								
mean	94813.859575	3.919560e-15	5.688174e-16	-8.769071e-15	2.782312e-15	-1.552563e-15	2.010663e-15	-1.694249e-15	-1.927028e-16	-3.137024e-15
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00	1.098632e+00
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01	-1.343407e+01
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01	-6.430976e-01
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-02	-5.142873e-02
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-01	5.971390e-01
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.000721e+01	1.559499e+01

Hình 4.32. Tổng quan giá trị của các thuộc tính

## d. Số dòng dữ liệu bị thiếu

Lệnh	Mô tả
data_raw.isnull().sum().max()	0 Do đó dữ liệu không bị thiếu.

## e. Số lượng giao dịch trong lớp phân loại

Thông qua thực hiện chạy các lệnh, đối với thuộc tính Lớp (class) của bộ dữ liệu thì có:

- Giao dịch bình thường chiếm 99.83% của toàn bộ dữ liệu.
- Giao dịch bất thường chiếm 0.17% của toàn bộ dữ liệu.

### 4.2.3. Phân tích dữ liệu

#### 4.2.3.1. Phân phối xác suất một số thuộc tính

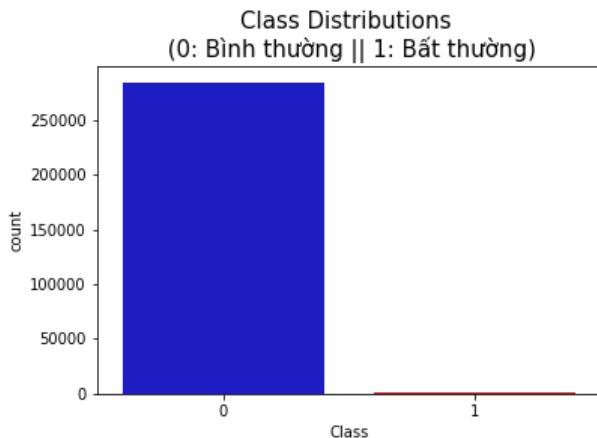
**Thuộc tính lớp (class):** Theo mô tả, ở thuộc tính Class có hai giá trị: 0 là các giao dịch bình thường, 1: là các giao dịch bất thường.

Bảng 4.18. Số giao dịch của mỗi giá trị theo thuộc tính lớp

Giá trị của thuộc tính	Số giao dịch
0: Giao dịch bình thường	284315
1: Giao dịch bất thường	492

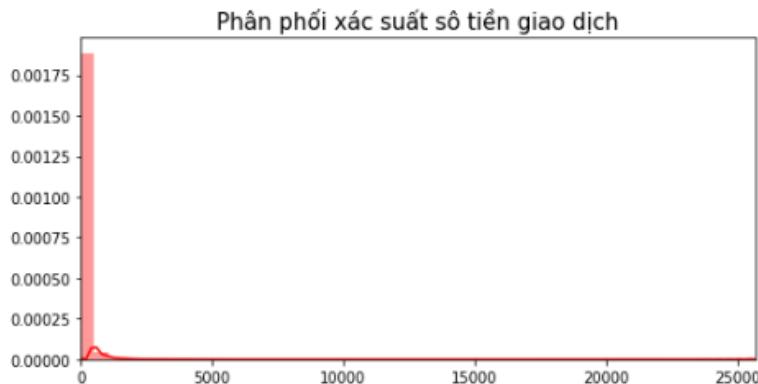
**Nhận xét:**

- Trong tập dữ liệu, số lượng giao dịch thành công (bình thường) chiếm đa số.
- Số lượng giao dịch bất thường là:  $492/284807 (0.17\%)$ . Từ đó ta thấy bộ dữ liệu đang bị mất cân bằng.
- Do đó cần tiến hành xử lý dữ liệu mất cân bằng này, vì nếu không xử lý thì khi sử dụng tập dữ liệu cho các mô hình dự đoán thì có thể sẽ không phù hợp vì nó sẽ cho rằng hầu hết các giao dịch đều bình thường.

**Đồ thị biểu diễn sự chênh lệch của thuộc tính Class:**

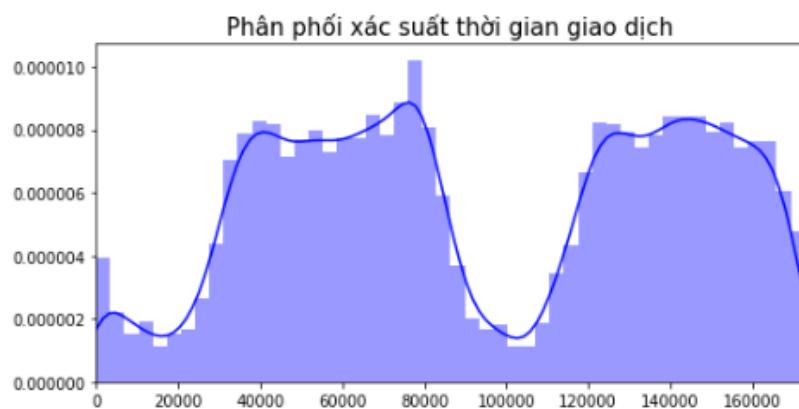
Hình 4.33. Đồ thị biểu diễn độ chênh lệch trong giao dịch bình thường và bất thường của thuộc tính Class

### Phân phối xác suất số tiền giao dịch:



Hình 4.34. Phân phối xác suất số tiền giao dịch

### Phân phối xác suất thời gian giao dịch:



Hình 4.35. Phân phối xác suất thời gian giao dịch

Nhìn vào các bảng phân phối trên, ta thấy được các thuộc tính Time, Amount, Class có độ chênh lệch rất nhiều, chứng tỏ chúng đang bị mất cân bằng.

### Nhận xét :

- Số tiền giao dịch tương đối nhỏ. Giá trị trung bình của tất cả các giao dịch thực hiện là khoảng 88 USD.
- Không có giá trị ‘Null’ trong bộ dữ liệu, vì vậy không thực hiện việc thay đổi giá trị.

#### 4.2.4. Tiền xử lý dữ liệu

##### 4.2.4.1. Scaling<sup>14</sup>

Trong bộ dữ liệu trên, thuộc tính Time và Amount không theo tỷ lệ với các cột V1, V2, ..., V28 do đó cần phải chia tỉ lệ cả hai để chúng được thu nhỏ như các cột khác. Mặt khác, cần phải tạo một mẫu phụ của khung dữ liệu để có số lượng các trường hợp giao dịch bình thường và bất thường tương đương, giúp thuật toán hiểu rõ hơn để xác định được các giao dịch có bất thường hay không.

Amount và Time là hai cột có giá trị tỷ lệ. Có 492 trường hợp giao dịch bất thường do đó sẽ lấy ngẫu nhiên 492 trường hợp giao dịch bình thường để tạo khung dữ liệu phụ mới. Rồi từ đó kết hợp các giá trị này với nhau để tạo ra một mẫu phụ mới.

Sau khi chạy các lệnh, ta thu được kết quả như sau:

Bảng 4.19. Kết quả khi chia tỉ lệ các thuộc tính

Scaled_amount	1.783274	-0.269825	4.983721	1.418291	0.670579
Scaled_time	-0.994983	-0.994983	-0.994972	-0.99497	-0.99496
V1	-1.359807	1.191857	-1.358354	-0.96627	-1.15823
V2	-0.072781	0.266151	-1.340163	-0.18523	0.877737
V3	2.536347	0.16648	1.773209	1.792993	1.548718
V4	1.378155	0.448154	0.37978	-0.86329	0.403034
V5	-0.338321	0.060018	-0.503198	-0.01031	-0.40719
V6	0.462388	-0.082361	1.800499	1.247203	0.095921
V7	0.239599	-0.078803	0.791461	0.237609	0.592941
V8	0.098698	0.085102	0.247676	0.377436	-0.27053
V9	0.363787	-0.255425	-1.514654	-1.38702	0.817739
V10	0.090794	-0.166974	0.207643	-0.05495	0.753074

<sup>14</sup> Scaling là một khái niệm chính của các mô hình học máy. Hầu hết các bộ dữ liệu chứa các tính năng thường khách nhau về cường độ và đơn vị. Một số mô hình học máy không bị ảnh hưởng bởi cường độ và đơn vị khác nhau này; tuy nhiên, hầu hết các mô hình học máy đều bị ảnh hưởng bởi sự thay đổi này vì chúng sử dụng một số liệu khoảng cách để tính khoảng cách giữa các điểm dữ liệu.

V11	-0.551600	1.612727	0.624501	-0.22649	-0.82284
V12	-0.617801	1.065235	0.066084	0.178228	0.538196
V13	-0.991390	0.489095	0.717293	0.507757	1.345852
V14	-0.311169	-0.143772	-0.165946	-0.28792	-1.11967
V15	1.468177	0.635558	2.345865	-0.63142	0.175121
V16	-0.470401	0.463917	-2.890083	-1.05965	-0.45145
V17	0.207971	-0.114805	1.109969	-0.68409	-0.23703
V18	0.025791	-0.183361	-0.121359	1.965775	-0.0382
V19	0.403993	-0.145783	-2.261857	-1.23262	0.803487
V20	0.251412	-0.069083	0.52498	-0.20804	0.408542
V21	-0.018307	-0.225775	0.247998	-0.1083	-0.00943
V22	0.277838	-0.638672	0.771679	0.005274	0.798278
V23	-0.110474	0.101288	0.909412	-0.19032	-0.13746
V24	0.066928	-0.339846	-0.689281	-1.17558	0.141267
V25	0.128539	0.16717	-0.327642	0.647376	0.20601
V26	-0.189115	0.125895	-0.139097	-0.22193	0.502292
V27	0.133558	-0.008983	-0.055353	0.062723	0.219422
V28	-0.021053	0.014724	-0.059752	0.061458	0.215153
Class	0	0	0	0	0

#### 4.2.4.2. Splitting Data (Tách khung dữ liệu gốc)

Mục tiêu là để so sánh giữa bộ dữ liệu gốc, và bộ dữ liệu sau khi thực hiện cân bằng dữ liệu, do đó cần phải tách khung dữ liệu gốc ban đầu.

No Frauds 99.83 % of the dataset  
Frauds 0.17 % of the dataset

-----  
Phân phối nhãn phân loại:  
[0.99827062 0.00172938]  
[0.99827636 0.00172364]

Hình 4.36. Kết quả sau khi thực hiện Splitting data đối với bộ dữ liệu

### 4.2.5. Xử lý dữ liệu mất cân bằng

#### 4.2.5.1. Random Undersampling

Tiến hành thực hiện kỹ thuật Lấy mẫu ngẫu nhiên (Random undersampling), về cơ bản bao gồm việc xóa dữ liệu để có một bộ dữ liệu cân bằng hơn và tránh được việc mô hình xây dựng bị quá mức.

Các bước thực hiện như sau:

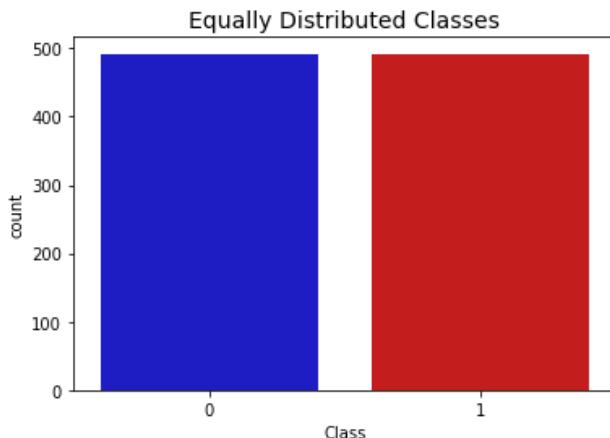
- Xác định độ mất cân bằng của thuộc tính lớp (sử dụng “value\_counts()” trên cột lớp để xác định số lượng cho mỗi nhãn).
- Xác định số trường hợp được coi là giao dịch bất thường (class = 1), thực hiện xóa các giao dịch bình thường cho đến khi cùng số lượng với các giao dịch bất thường (giả sử muốn tỷ lệ 50/50, tương đương với việc có 492 trường hợp giao dịch bình thường và 492 trường hợp giao dịch bất thường).
- Sau khi thực hiện kỹ thuật này, tạo một mẫu phụ của khung dữ liệu với tỷ lệ 50/50 liên quan đến các lớp. Sau đó, thực hiện việc xáo trộn dữ liệu để kiểm tra độ duy trì ổn định tính chính xác.

Kết quả thực hiện như sau (chi tiết lệnh xem phần phụ lục):

	scaled_amount	scaled_time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
271994	-0.098512	0.941811	1.952612	0.570243	-1.635657	3.611706	0.994265	-0.204485	0.630400	-0.118109	-1.317378	1.749963	-0.280662	-0.717751	-1.991944
100623	7.364773	-0.201142	-0.758469	-0.045410	-0.168438	-1.313275	-1.901763	0.739433	3.071892	-0.483422	0.618203	-1.769060	-0.651414	-0.005423	-0.517194
37421	17.622860	-0.537389	-1.245076	-4.678746	-0.509095	0.804979	-2.527825	0.327461	0.684009	-0.003134	0.981389	-0.658852	0.429093	-0.490091	-2.330822
151008	-0.293440	0.113606	-26.457745	16.497472	-30.177317	8.904157	-17.892600	-1.227904	-31.197329	-11.438920	-9.462573	-22.187089	4.419997	-10.592305	-0.703796
15225	1.089779	-0.682656	-19.856322	12.095893	-22.164083	6.115541	-15.148022	-4.346724	-15.648507	13.276805	-3.974162	-8.859194	5.730816	-8.088034	0.230825

Hình 4.37. Kết quả sau khi thực hiện undersampling

Biểu đồ các nhãn phân loại đối với dữ liệu mới của thuộc tính lớp như sau:



Hình 4.38. Biểu đồ phân phối các nhãn của thuộc tính lớp  
trong bộ dữ liệu mới

#### 4.2.5.2. Random Oversampling (SMOTE)

Tiến hành thực hiện kỹ thuật Random undersampling, dưới đây là kết quả so sánh trước và sau khi sử dụng kỹ thuật SMOTE:

- Trước khi thực hiện SMOTE:
  - o Số lượng giao dịch bất thường: 492.
  - o Số lượng giao dịch bình thường: 284315.
- Sau khi thực hiện SMOTE:
  - o Số lượng giao dịch bất thường: 284315.
  - o Số lượng giao dịch bình thường: 284315.

Chi tiết lệnh thực hiện tham khảo phần phụ lục.

#### 4.2.5.3. Phương pháp kết hợp SMOTE + ENN

Tiến hành áp dụng kết hợp giữa SMOTE và ENN, dưới đây là kết quả so sánh trước và sau khi sử dụng kỹ thuật SMOTE + ENN:

- Trước khi thực hiện SMOTE + ENN:
  - o Số lượng giao dịch bất thường: 492.
  - o Số lượng giao dịch bình thường: 284315.
- Sau khi thực hiện SMOTE + ENN:
  - o Số lượng giao dịch bất thường: 275740

- Số lượng giao dịch bình thường: 265395.

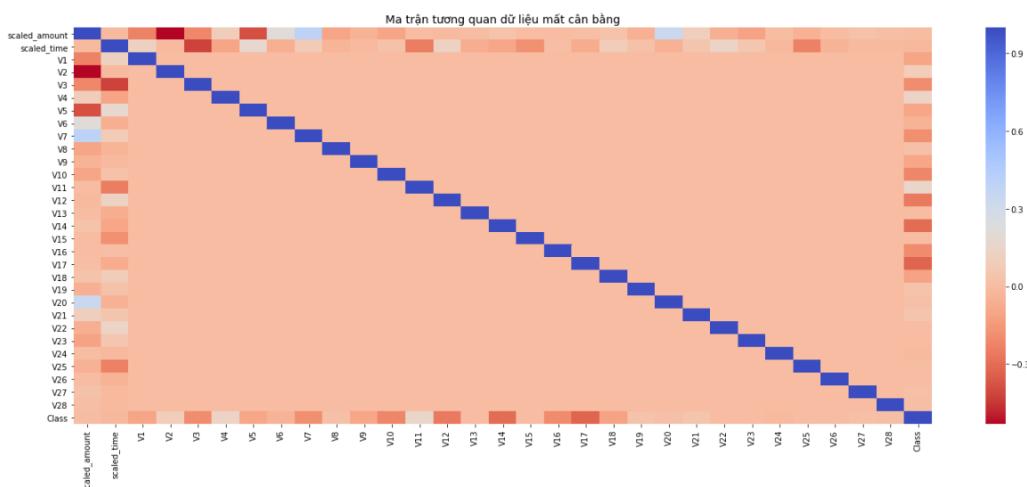
Chi tiết lệnh thực hiện tham khảo phần phụ lục.

#### 4.2.6. Ma trận tương quan (Correlation matrix)

Ma trận tương quan là bản chất của việc hiểu dữ bộ dữ liệu Credit Card Fraud Detection. Cần biết có những tính năng ảnh hưởng lớn đến việc một giao dịch cụ thể có phải là lừa đảo hay không. Tuy nhiên, điều quan trọng là sử dụng đúng tệp dữ liệu (undersampling) để xem các tính năng nào có mối tương quan thuận hoặc ngược cao liên quan đến các giao dịch bất thường.

##### 4.2.6.1. Dữ liệu chưa xử lý cân bằng

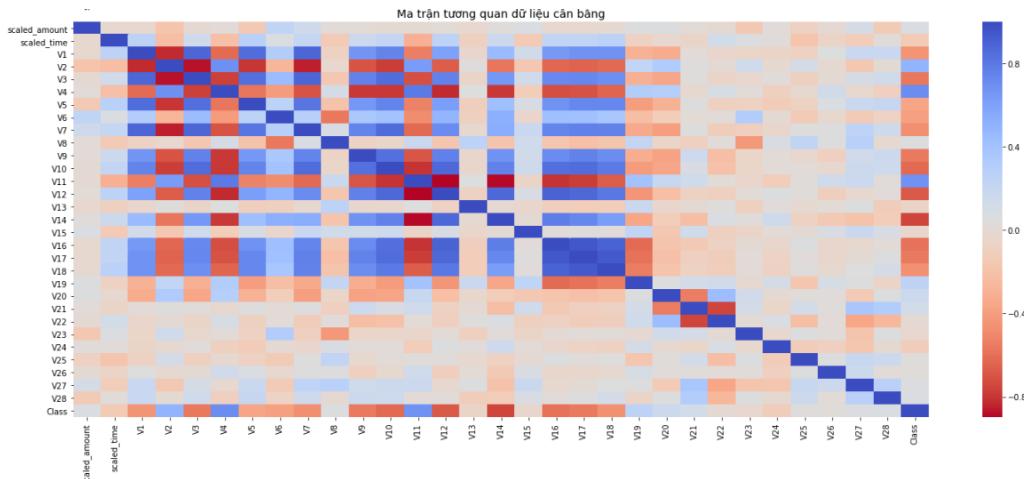
Ma trận tương quan dưới đây thể hiện trên bộ dữ liệu trước khi xử lý cân bằng:



Hình 4.39. Ma trận tương quan đối với dữ liệu mất cân bằng

##### 4.2.6.2. Dữ liệu cân bằng theo phương pháp Undersampling

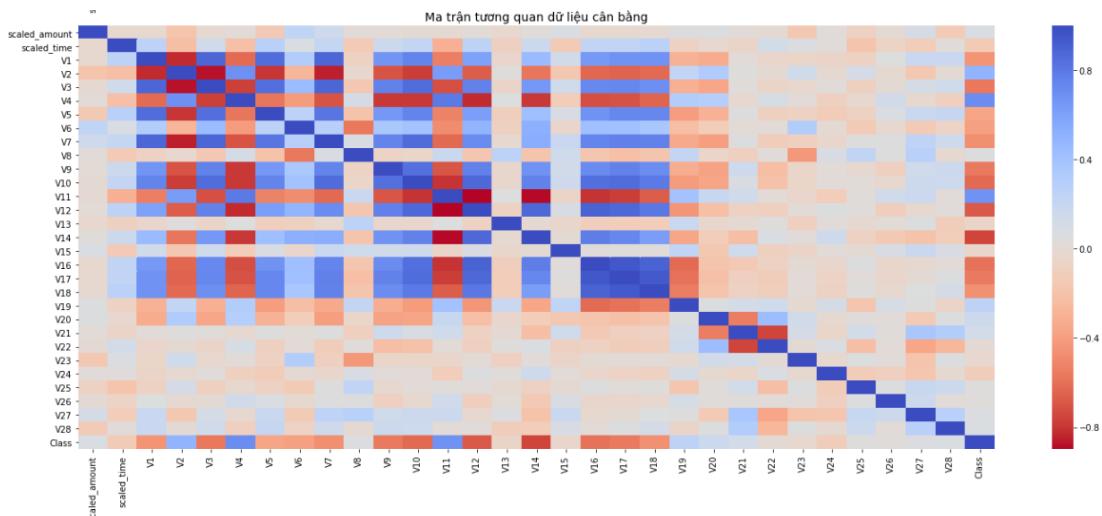
Dưới đây là ma trận tương quan của bộ dữ liệu sau khi xử lý cân bằng dùng phương pháp Undersampling:



Hình 4.40. Ma trận tương quan đối với bộ dữ liệu được xử lý cân bằng sử dụng undersampling

#### 4.2.6.3. Dữ liệu cân bằng theo phương pháp Oversampling

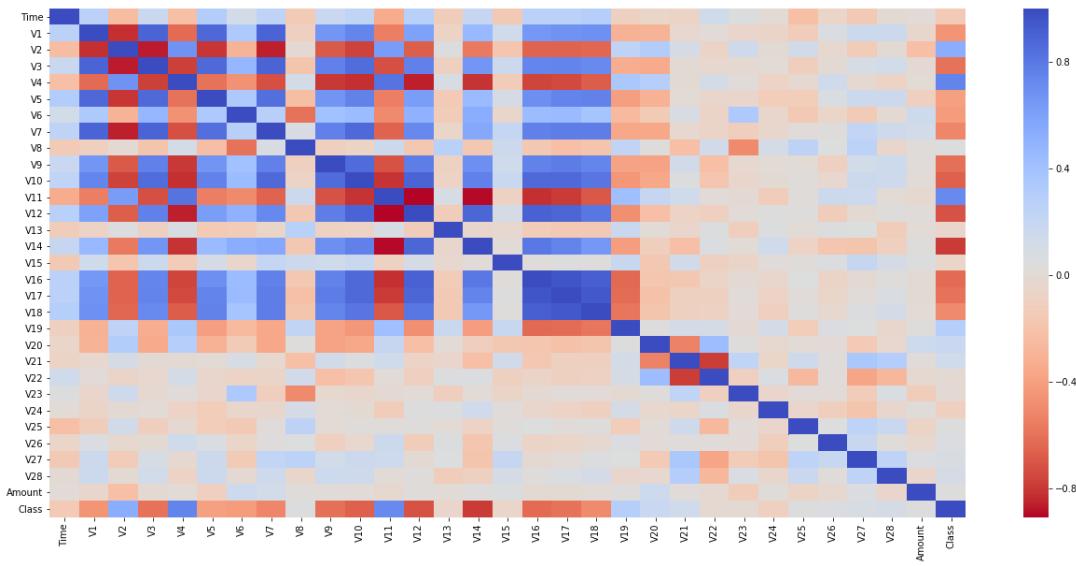
Dưới đây là ma trận tương quan của bộ dữ liệu sau khi xử lý cân bằng dùng phương pháp Oversampling:



Hình 4.41. Ma trận tương quan đối với bộ dữ liệu được xử lý cân bằng sử dụng oversampling

#### 4.2.6.4. Dữ liệu cân bằng theo phương pháp kết hợp SMOTE + ENN

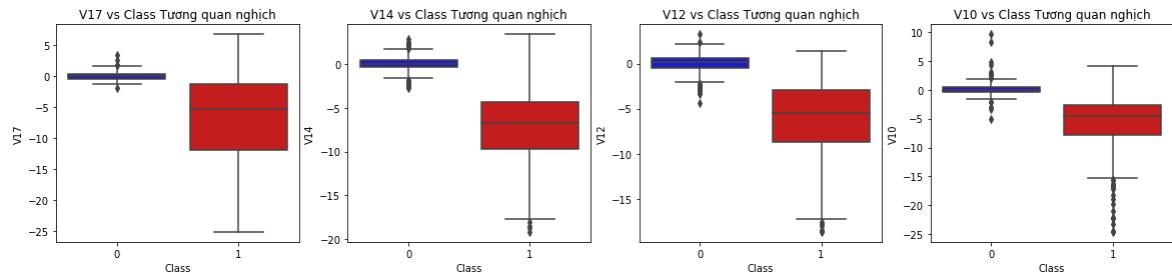
Dưới đây là ma trận tương quan của bộ dữ liệu sau khi xử lý cân bằng dùng phương pháp kết hợp SMOTE + ENN:



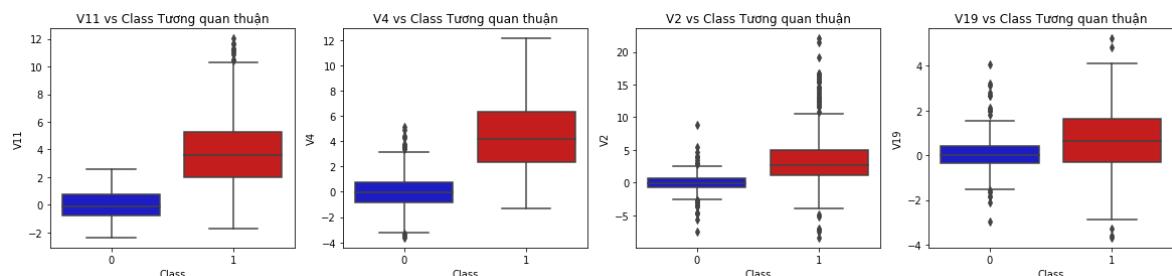
Hình 4.42. Ma trận tương quan đối với bộ dữ liệu được xử lý cân bằng sử dụng phương pháp kết hợp SMOTE và ENN

#### 4.2.7. Tương quan giữa các thuộc tính

Biểu đồ thể hiện mối tương quan giữa một số lớp ở bộ dữ liệu:



Hình 4.43. Tương quan giữa các thuộc tính V với thuộc tính lớp



Hình 4.44. Tương quan giữa các thuộc tính V với thuộc tính lớp

## Tóm tắt:

- Tương quan ngược: các thuộc tính V17, V14, V12 và V10 có mối tương quan ngược chiều. Lưu ý rằng các giá trị này càng thấp thì kết quả cuối cùng sẽ là một giao dịch bất thường.
- Tương quan tích cực: V2, V4, V11 và V19 có mối tương quan tích cực. Lưu ý rằng các giá trị này càng cao thì kết quả cuối cùng sẽ là một giao dịch bất thường.
- Biểu đồ hộp (BoxPlots): sử dụng biểu đồ hộp để hiểu rõ hơn về việc phân phối các tính năng này trong các giao dịch bình thường và giao dịch bất thường.

### 4.2.8. Phát hiện bất thường (Anomaly Detection)

Mục đích loại bỏ các “ngoại lệ cực đoan” khỏi các thuộc tính có mối tương quan cao với các nhãn. Điều này sẽ có tác động tích cực đến độ chính xác đến các mô hình đang thực hiện.

Phương pháp phạm vi liên vùng (Interquartile Range Method):

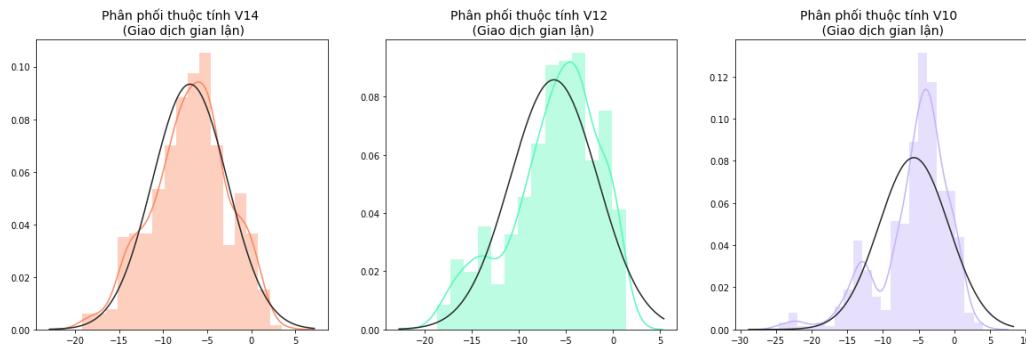
- Phạm vi liên vùng (Interquartile Range - IQR): Thực hiện tính toán giá trị này bằng sự khác biệt giữa phân vị (percentile) thứ 75 và phân vị thứ 25. Mục đích là tạo ra một ngưỡng vượt quá tỷ lệ phần trăm thứ 75 và 25 trong trường hợp một số giá trị vượt qua ngưỡng này có thể bị xóa.
- Biểu đồ hộp (Boxplots): Bên cạnh việc dễ dàng nhìn thấy các phần trăm thứ 25 và 75 (cả hai đầu của hình vuông), có thể dễ dàng nhìn thấy các ngoại lệ cực đoan (extreme outliers) (các điểm nằm ngoài cực thấp và cực cao hơn).

#### Lưu ý:

- Cần cẩn thận về ngưỡng để loại bỏ các ngoại lệ. Trong các lệnh dưới đây, ngưỡng được xác định bằng cách nhân một số (ví dụ: 1.5) với Phạm vi liên dải. Ngưỡng này càng cao, ngoại lệ sẽ ít được phát hiện và ngưỡng này càng thấp thì càng phát hiện ra nhiều ngoại lệ.

#### 4.2.8.1. Dữ liệu xử lý bằng phương pháp Undersampling

Dưới đây là các biểu đồ phân phối thuộc tính V14, V12, V10 (giao dịch gian lận):



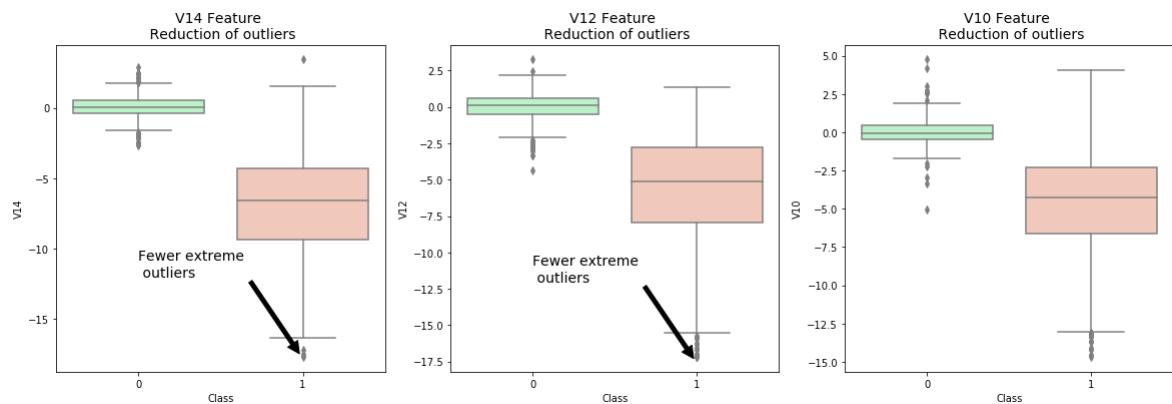
Hình 4.45. Biểu đồ phân phối thuộc tính V14, V12, V10 của bộ dữ liệu dùng Undersampling

Tiến hành xóa các giá trị ngoại vi khỏi các giao dịch gian lận cho các thuộc tính V14, V12, V10 ta thu được kết quả:

Bảng 4.20. Kết quả sau khi thực hiện xóa các giá trị ngoại vi của các thuộc tính V14, V12, V10 của bộ dữ liệu dùng Undersampling

	<b>Bách phân vị 25</b>	<b>Bách phân vị 75</b>	<b>iqr</b>	<b>Giá trị biên dưới</b>	<b>Giá trị biên trên</b>	<b>Giá trị ngoại vi</b>	<b>Số lượng dữ liệu còn lại sau khi loại bỏ các giá trị ngoại vi</b>
V14	-9.6927	-4.2828	5.4099	-17.8076	3.8320	4	980
V12				-17.3430	5.7770	4	976
V10				-14.8989	4.9203	27	947

Biểu đồ hộp sau khi loại bỏ các giá trị ngoại vi cho các thuộc tính V14, V12, V10:



Hình 4.46. Biểu đồ hộp của các thuộc tính V14, V12, V10 sau khi loại bỏ các giá trị ngoại vi của bộ dữ liệu dùng Undersampling

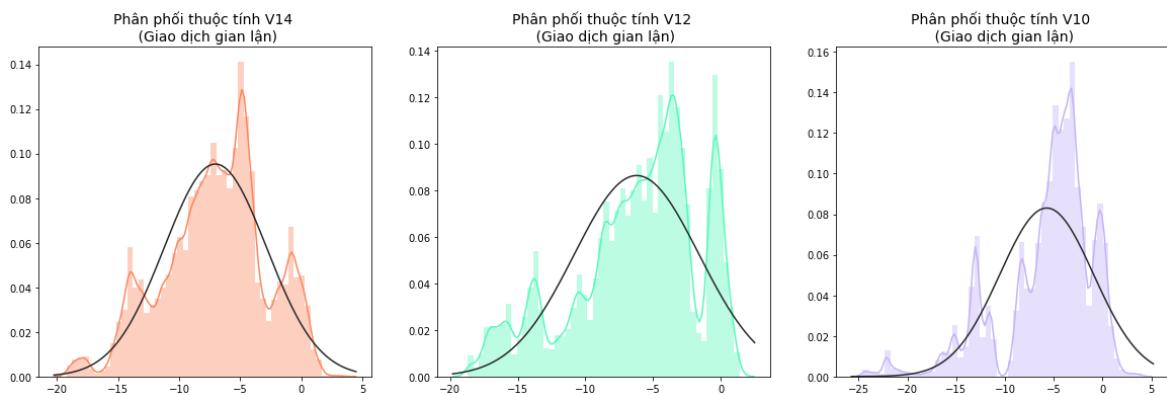
### Chú giải:

- Trực quan hóa phân phối: Trước tiên bắt đầu bằng cách hình dung phân phối thuộc tính mà sẽ sử dụng để loại bỏ một số ngoại lệ. V14 là thuộc tính duy nhất có phân phối Gaussian so với các thuộc tính V12 và V10.
- Xác định ngưỡng: Sau khi lựa chọn giá trị thích hợp, tiến hành nhân với iqr (loại bỏ nhiều ngoại lệ hơn), từ đó tiến hành xác định ngưỡng trên và ngưỡng

dưới bằng cách đặt ngưỡng q25 (ngưỡng cực thấp) và thêm ngưỡng q75 + (ngưỡng cực cao).

#### 4.2.8.2. Dữ liệu xử lý bằng phương pháp Oversampling

Dưới đây là các biểu đồ phân phối thuộc tính V14, V12, V10 (giao dịch gian lận):



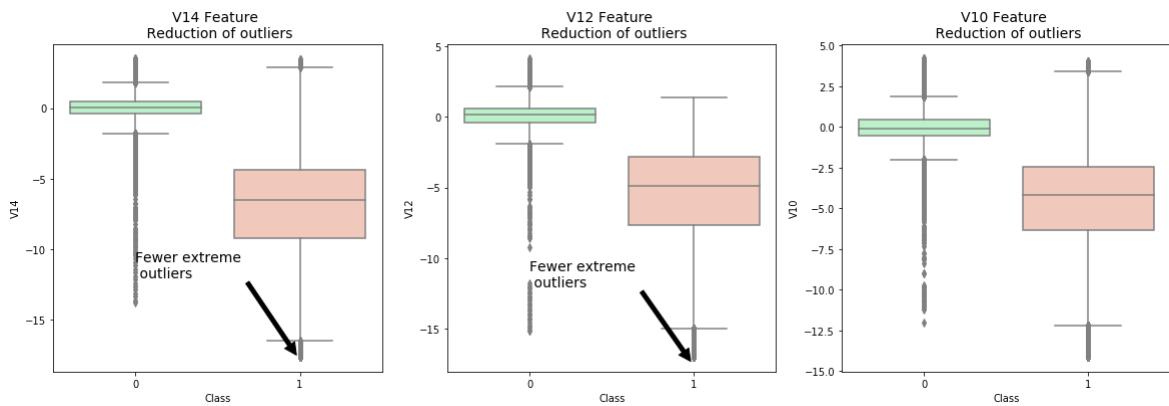
Hình 4.47. Biểu đồ phân phối thuần túy V14, V12, V10 của bộ dữ liệu dùng Oversampling

Tiến hành xóa các giá trị ngoại vi khỏi các giao dịch gian lận cho các thuộc tính V14, V12, V10 ta thu được kết quả:

khi thực hiện xóa các giá trị ngoại vi của các thuộc tính V14, V12, V10 của bộ dữ liệu dùng Oversampling

	Bách phân vị 25	Bách phân vị 75	iqr	Giá trị biên dưới	Giá trị biên trên	Giá trị ngoại vi	Số lượng dữ liệu còn lại sau khi loại bỏ các giá trị ngoại vi
V14				-17.6157	3.5048	3288	564955
V12	-9.6955	-4.4154	5.2801	-16.9713	5.3669	3762	561193
V10				-14.1180	4.1984	17970	541307

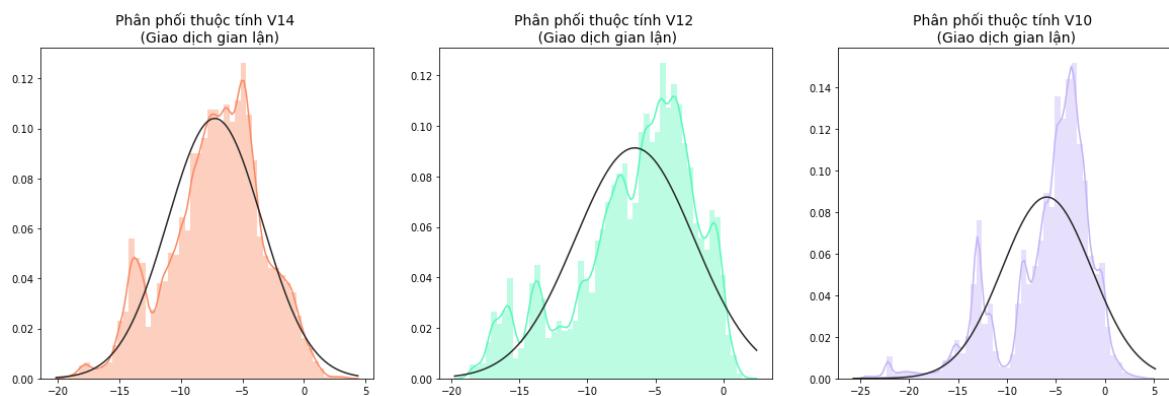
Biểu đồ hộp sau khi loại bỏ các giá trị ngoại vi cho các thuộc tính V14, V12, V10:



Hình 4.48. Biểu đồ hộp của các thuộc tính V14, V12, V10 sau khi loại bỏ các giá trị ngoại vi của bộ dữ liệu dùng Oversampling

#### 4.2.8.3. Dữ liệu xử lý bằng phương pháp kết hợp SMOTE + ENN

Dưới đây là các biểu đồ phân phối thuộc tính V14, V12, V10 (giao dịch gian lận):



Hình 4.49. Biểu đồ phân phối thuộc tính V14, V12, V10 của bộ dùng SMOTE + ENN

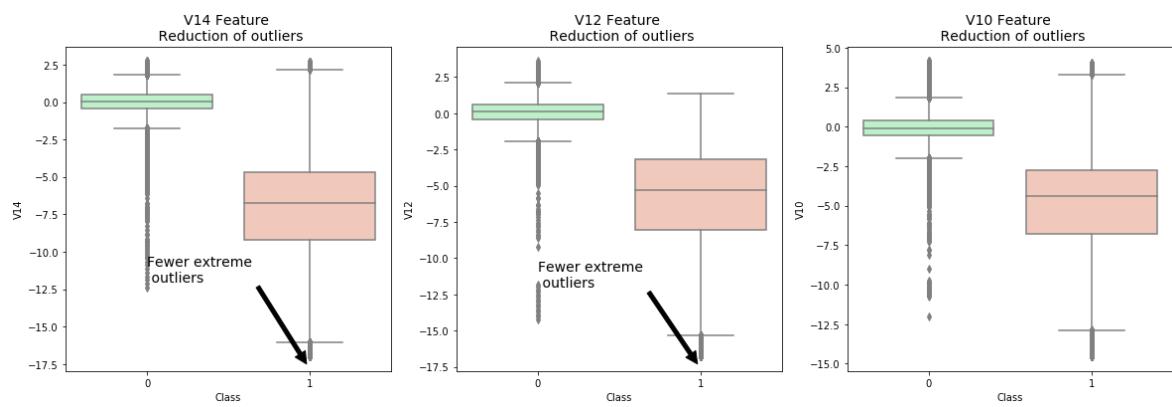
Tiến hành xóa các giá trị ngoại vi khỏi các giao dịch gian lận cho các thuộc tính V14, V12, V10 ta thu được kết quả:

Bảng 4.21. Kết quả sau khi thực hiện xóa các giá trị ngoại vi của các thuộc tính V14, V12, V10 của bộ dữ liệu dùng SMOTE + ENN

	<b>Bách phân vị 25</b>	<b>Bách phân vị 75</b>	<b>iqr</b>	<b>Giá trị biên dưới</b>	<b>Giá trị</b>	<b>Giá trị</b>	<b>Số lượng dữ liệu còn lại sau</b>

					biên trên	ngoại vi	khi loại bỏ các giá trị ngoại vi
V14	-9.5906	-4.6631	4.9275	- 16.9818	2.7282	2411	537886
V12				- 16.7700	4.7554	4326	533560
V10				- 14.5279	4.1496	13545	518153

Biểu đồ hộp sau khi loại bỏ các giá trị ngoại vi cho các thuộc tính V14, V12, V10:



Hình 4.50. Biểu đồ hộp của các thuộc tính V14, V12, V10 sau khi loại bỏ các giá trị ngoại vi của bộ dữ liệu dùng SMOTE + ENN

#### 4.2.9. Lưu dữ liệu sau khi thực hiện cân bằng dữ liệu

Tiến hành lưu dữ liệu sau khi thực hiện cân bằng dữ liệu bằng ba phương pháp trên, dữ liệu này được sử dụng để huấn luyện (train) (chi tiết lệnh phần phụ lục).

#### 4.2.10. Mô hình phát hiện bất thường sử dụng thuật toán Random forest

##### 4.2.10.1. Với bộ dữ liệu chưa xử lý mất cân bằng

Thực hiện tách dữ liệu thành hai tập: tập train và test, sau đó sử dụng xác thực chéo trên tập train để điều chỉnh các tham số, số lượng các giao dịch của mỗi phân lớp theo thuộc tính isFraud trong trường hợp bộ dữ liệu chưa được xử lý cân bằng được liệt kê theo bảng sau:

Bảng 4.22. Thống kê số lượng của mỗi giá trị ở thuộc tính Class theo bộ dữ liệu

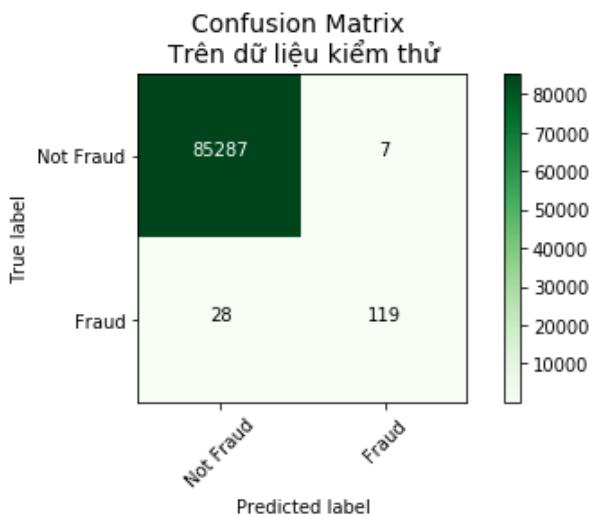
Loại xử lý của bộ dữ liệu	Tập dữ liệu	Số lượng của mỗi phân lớp của thuộc tính Class	
		Class = 0 (giao dịch bình thường)	Class = 1 (giao dịch bất thường)
Chưa xử lý cân bằng	Ban đầu	284315	492
	Tập train	199020	344
	Tập test	85295	148

Ta tiến hành chạy mô hình bằng thuật toán Random forest cho bộ dữ liệu, sử dụng 3 fold-cross validation, ta có kết quả được cho theo bảng dưới đây:

Bảng 4.23. Hình kết quả sau khi chạy thuật toán Random forest đối với bộ dữ liệu chưa cân bằng

Lần chạy	Độ chính xác trên tập train	Độ chính xác trên tập test	F1-score trên tập train	F1-score trên tập test	Thời gian
1	99.9398%	99.9579%			35.8072 giây
2	99.9564%	99.9579%			36.3180 giây
3	99.9564%	99.9579%			36.9021 giây
<b>Trung bình</b>	<b>99.9508%</b>	<b>99.9579%</b>	<b>0.8426</b>	<b>0.8689</b>	<b>36.3424 giây</b>

Ma trận nhầm lẫn trên tập dữ liệu test (chi tiết lệnh xem ở phụ lục):



Hình 4.51. Ma trận nhầm lẫn trên tập dữ liệu test của bộ dữ liệu chưa xử lý cân bằng

### Kết quả:

- Có 85287 giao dịch bình thường được dự đoán đúng là giao dịch bình thường.
- 7 lần giao dịch bình thường được dự đoán sai là giao dịch bất thường.
- 28 lần giao dịch bất thường được dự đoán sai là giao dịch bình thường.
- 119 lần giao dịch bất thường được dự đoán chính xác là giao dịch bất thường.

#### 4.2.10.2. Với bộ dữ liệu đã xử lý mất cân bằng

##### a. Đối với bộ dữ liệu xử lý cân bằng dùng phương pháp Oversampling

Thực hiện tách bộ dữ liệu thành hai tập train và test, số lượng giao dịch trên từng nhãn của thuộc tính isFraud trên tập train và test được cho theo bảng dưới đây:

Bảng 4.24. Thống kê số lượng của mỗi giá trị ở thuộc tính Class

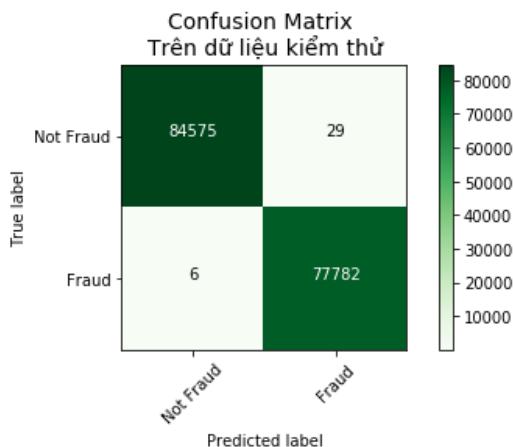
Loại xử lý của bộ dữ liệu	Tập dữ liệu	Số lượng của mỗi phân lớp của thuộc tính Class	
		Class = 0 (giao dịch bình thường)	Class = 1 (giao dịch bất thường)
Đã thực hiện xử lý cân bằng theo Oversampling	Ban đầu chưa tách	282012	259295
	Tập train	197408	181506
	Tập test	84604	77789

Ta tiến hành chạy mô hình bằng thuật toán Random forest cho bộ dữ liệu cân bằng theo phương pháp Oversampling, sử dụng 3 fold-cross validation, thu được kết quả (chi tiết lệnh xem phần phụ lục):

Bảng 4.25. Kết quả sau khi chạy thuật toán Random forest đối với bộ dữ liệu đã xử lý cân bằng dùng Oversampling

Lần chạy	Độ chính xác tập train	Độ chính xác tập test	F1-score trên tập train	F1-score trên tập test	Thời gian
1	99.9889%	99.9772%			67.8341 giây
2	99.9818%	99.9772%			67.8082 giây
3	99.9826%	99.9803%			67.4045 giây
<b>Trung bình</b>	<b>99.9844%</b>	<b>99.9782%</b>	<b>0.9998</b>	<b>0.9998</b>	<b>67.6823 giây</b>

Ma trận trên tập dữ liệu test:



Hình 4.52. Ma trận nhầm lẫn trên tập dữ liệu test dùng Oversampling

### Kết quả:

- Có 84575 giao dịch bình thường được dự đoán đúng là giao dịch bình thường.
- 29 lần giao dịch bình thường được dự đoán sai là giao dịch bất thường.
- 6 lần giao dịch bất thường được dự đoán sai là giao dịch bình thường.
- 77782 lần giao dịch bất thường được dự đoán chính xác là giao dịch bất thường.

### b. Đối với bộ dữ liệu xử lý cân bằng dùng phương pháp Undersampling

Thực hiện tách bộ dữ liệu thành hai tập train và test, số lượng giao dịch trên từng nhãn của thuộc tính isFraud trên tập train và test được cho theo bảng dưới đây:

Bảng 4.26. Thống kê số lượng giao dịch của mỗi giá trị ở thuộc tính Class

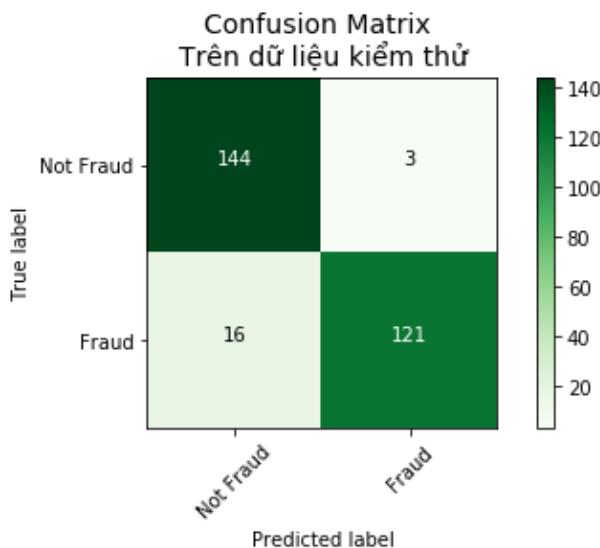
Loại xử lý của bộ dữ liệu	Tập dữ liệu	Số lượng của mỗi phân lớp của thuộc tính Class	
		Class = 0 (giao dịch bình thường)	Class = 1 (giao dịch bất thường)
Đã thực hiện xử lý cân bằng theo Undersampling	Ban đầu chưa tách	490	457
	Tập train	343	319
	Tập test	147	138

Ta tiến hành chạy mô hình bằng thuật toán Random forest sử dụng 3 fold-cross validation cho bộ dữ liệu, thu được kết quả (chi tiết lệnh xem phần phụ lục):

Bảng 4.27. Hình kết quả sau khi chạy thuật toán Random forest đối với bộ dữ liệu đã xử lý cân bằng dùng Undersampling

Lần chạy	Độ chính xác trên tập train	Độ chính xác trên tập test	F1-score trên tập train	F1-score trên tập test	Thời gian
1	95.4955%	93.3333%			0.4444 giây
2	94.5455%	92.6316%			0.4439 giây
3	93.6364%	93.6842%			0.4372 giây
<b>Trung bình</b>	<b>94.5591%</b>	<b>93.2164%</b>	<b>0.9406</b>	<b>0.9264</b>	<b>0.4418 giây</b>

Ma trận trên tập dữ liệu test:



Hình 4.53. Ma trận nhầm lẫn trên tập dữ liệu test dùng Undersampling

### Kết quả:

- Có 144 giao dịch bình thường được dự đoán đúng là giao dịch bình thường.
- 3 lần giao dịch bình thường được dự sai đoán là giao dịch bất thường.
- 16 lần giao dịch bất thường được dự đoán sai là giao dịch bình thường.
- 121 lần giao dịch bất thường được dự đoán chính xác là giao dịch bất thường.

### c. Đối với bộ dữ liệu xử lý cân bằng dùng phương pháp kết hợp SMOTE + ENN

Thực hiện tách bộ dữ liệu thành hai tập train và test, số lượng giao dịch trên từng nhãn của thuộc tính isFraud trên tập train và test được cho theo bảng dưới đây:

Bảng 4.28. Thống kê số lượng giao dịch của mỗi giá trị ở thuộc tính Class

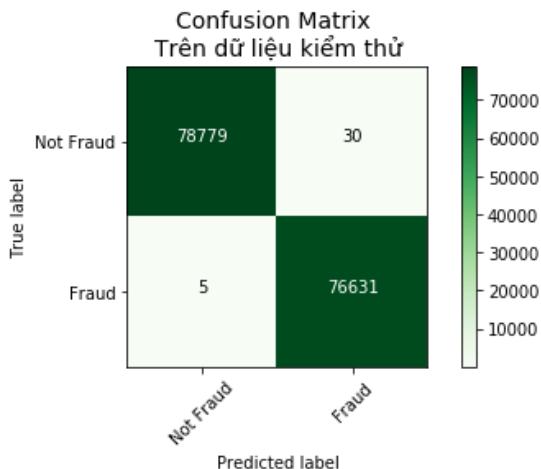
Loại xử lý của bộ dữ liệu	Tập dữ liệu	Số lượng của mỗi phân lớp của thuộc tính Class	
		Class = 0 (giao dịch bình thường)	Class = 1 (giao dịch bất thường)
Đã thực hiện xử lý cân bằng theo phương pháp kết hợp SMOTE + ENN	Ban đầu chưa tách	262695	255458
	Tập train	183886	178821
	Tập test	78809	76637

Ta tiến hành chạy mô hình bằng thuật toán Random forest sử dụng 3 fold-cross validation cho bộ dữ liệu, thu được kết quả (chi tiết xem phần phụ lục):

Bảng 4.29. Kết quả sau khi chạy thuật toán Random forest đối với bộ dữ liệu đã xử lý cân bằng dùng SMOTE + ENN

Lần chạy	Độ chính xác trên tập train	Độ chính xác trên tập test	F1-score trên tập train	F1-score trên tập test	Thời gian
1	99.9851%	99.9801%			70.5193 giây
2	99.9735%	99.9723%			69.4997 giây
3	99.9851%	99.9794%			71.1136 giây
<b>Trung bình</b>	<b>99.9813%</b>	<b>99.9773%</b>	<b>0.9998</b>	<b>0.9998</b>	<b>70.3775 giây</b>

Ma trận trên tập dữ liệu test::



Hình 4.54. Ma trận nhầm lẫn trên tập dữ liệu test dùng SMOTE + ENN

### Kết quả:

- Có 78779 giao dịch bình thường được dự đoán đúng là giao dịch bình thường.
- 30 lần giao dịch bình thường được dự đoán sai là giao dịch bất thường.
- 5 lần giao dịch bất thường được dự đoán sai là giao dịch bình thường.
- 76631 lần giao dịch bất thường được dự đoán chính xác là giao dịch bất thường.

#### 4.2.11. Mô hình phát hiện bất thường sử dụng thuật toán Extreme Gradient Boosting (EGB)

##### 4.2.11.1. Với bộ dữ liệu chưa xử lý mất cân bằng

Thực hiện tách dữ liệu thành hai tập: tập train và test, sau đó sử dụng xác thực chéo trên tập train để điều chỉnh các tham số, số lượng các giao dịch của mỗi phân lớp theo thuộc tính isFraud trong trường hợp bộ dữ liệu chưa được xử lý cân bằng được liệt kê theo bảng sau:

Bảng 4.30. Thống kê số lượng của mỗi giá trị ở thuộc tính Class theo bộ dữ liệu

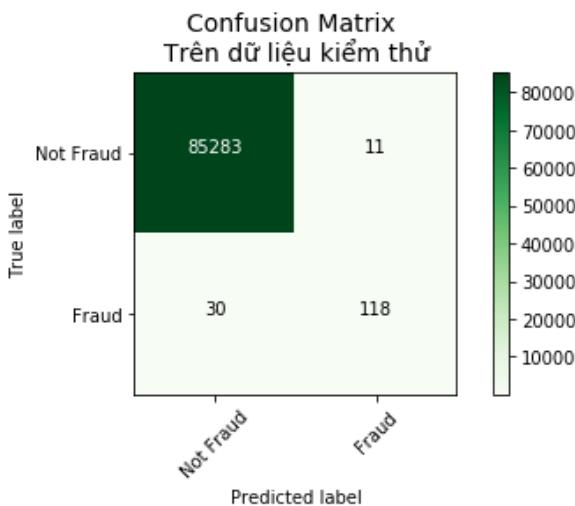
Loại xử lý của bộ dữ liệu	Tập dữ liệu	Số lượng của mỗi phân lớp của thuộc tính Class	
		Class = 0 (giao dịch bình thường)	Class = 1 (giao dịch bất thường)
Chưa xử lý cân bằng	Ban đầu	284315	492
	Tập train	199020	344
	Tập test	85295	148

Tiến hành chạy mô hình bằng thuật toán EGB cho bộ dữ liệu, sử dụng 3 fold-cross validation, ta có kết quả được cho theo bảng dưới đây (chi tiết lệnh xem phụ lục):

Bảng 4.31. Kết quả sau khi chạy thuật toán EGB đối với bộ dữ liệu chưa cân bằng

Lần chạy	Độ chính xác trên tập train	Độ chính xác trên tập test	F1-score trên tập train	F1-score trên tập test	Thời gian
1	99.9549%	99.9508%			31.7186 giây
2	99.9503%	99.9531%			34.8223 giây
3	99.9428%	99.9508%			34.7050 giây
<b>Trung bình</b>	<b>99.9493%</b>	<b>99.9516%</b>	<b>0.8397</b>	<b>0.8509</b>	<b>33.7486 giây</b>

Ma trận nhầm lẫn trên tập dữ liệu test:



Hình 4.55. Ma trận nhầm lẫn trên tập dữ liệu test của bộ dữ liệu chưa xử lý cân bằng

### Kết quả:

- Có 85283 giao dịch bình thường được dự đoán đúng là giao dịch bình thường.
- 11 lần giao dịch bình thường được dự đoán là giao dịch bất thường.
- 30 lần giao dịch bất thường được dự đoán sai là giao dịch bình thường.
- 118 lần giao dịch bất thường được dự đoán chính xác là giao dịch bất thường.

#### 4.2.11.2. Với bộ dữ liệu đã xử lý mất cân bằng

##### a. Đối với bộ dữ liệu xử lý cân bằng dùng phương pháp Oversampling

Thực hiện tách bộ dữ liệu thành hai tập train và test, số lượng giao dịch trên từng nhãn của thuộc tính isFraud trên tập train và test được cho theo bảng dưới đây:

Bảng 4.32. Thông kê số lượng của mỗi giá trị ở thuộc tính Class

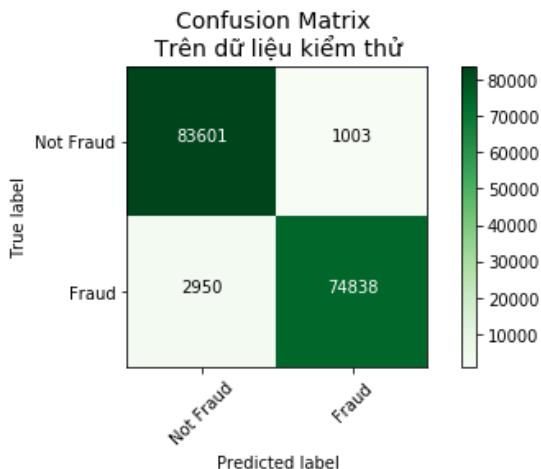
<b>Loại xử lý của bộ dữ liệu</b>	<b>Tập dữ liệu</b>	<b>Số lượng của mỗi phân lớp của thuộc tính Class</b>	
		<b>Class = 0 (giao dịch bình thường)</b>	<b>Class = 1 (giao dịch bất thường)</b>
Đã thực hiện xử lý cân bằng theo Oversampling	Ban đầu chưa tách	282012	259295
	Tập train	197408	181506
	Tập test	84604	77789

Ta tiến hành chạy mô hình bằng thuật toán EGB cho bộ dữ liệu cân bằng theo phương pháp Oversampling, sử dụng 3 fold-cross validation, thu được kết quả (chi tiết lệnh xem phần phụ lục):

Bảng 4.33. Kết quả sau khi chạy thuật toán EGB đối với bộ dữ liệu đã xử lý cân bằng dùng Oversampling

<b>Lần chạy</b>	<b>Độ chính xác tập train</b>	<b>Độ chính xác tập test</b>	<b>F1-score trên tập train</b>	<b>F1-score trên tập test</b>	<b>Thời gian</b>
1	97.7063%	97.5873%			70.5326 giây
2	97.6177%	97.5036%			70.5800 giây
3	97.6279%	97.6058%			70.5427 giây
<b>Trung bình</b>	<b>97.6506%</b>	<b>97.5656%</b>	<b>0.9752</b>	<b>0.9743</b>	<b>70.5518 giây</b>

Ma trận trên tập dữ liệu test:



Hình 4.56. Ma trận nhầm lẫn trên tập dữ liệu test dùng Oversampling

### Kết quả:

- Có 83601 giao dịch bình thường được dự đoán đúng là giao dịch bình thường.
- 1003 lần giao dịch bình thường được dự đoán sai là giao dịch bất thường.
- 2950 lần giao dịch bất thường được dự đoán sai là giao dịch bình thường.
- 74838 lần giao dịch bất thường được dự đoán chính xác là giao dịch bất thường.

### b. Đối với bộ dữ liệu xử lý cân bằng dùng phương pháp Undersampling

Thực hiện tách bộ dữ liệu thành hai tập train và test, số lượng giao dịch trên từng nhãn của thuộc tính isFraud trên tập train và test được cho theo bảng dưới đây:

Bảng 4.34. Thống kê số lượng giao dịch của mỗi giá trị ở thuộc tính Class

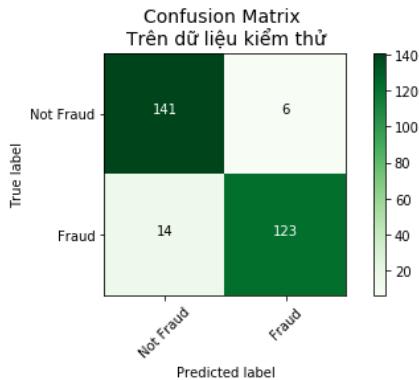
Loại xử lý của bộ dữ liệu	Tập dữ liệu	Số lượng của mỗi phân lớp của thuộc tính Class	
		Class = 0 (giao dịch bình thường)	Class = 1 (giao dịch bất thường)
Đã thực hiện xử lý cân bằng theo Undersampling	Ban đầu chưa tách	490	457
	Tập train	343	319
	Tập test	147	138

Tiến hành chạy mô hình bằng thuật toán EGB sử dụng 3 fold-cross validation cho bộ dữ liệu, thu được kết quả:

Bảng 4.35. Kết quả sau khi chạy thuật toán EGB đối với bộ dữ liệu đã xử lý cân bằng dùng Undersampling

Lần chạy	Độ chính xác trên tập train	Độ chính xác trên tập test	F1-score trên tập train	F1-score trên tập test	Thời gian
1	95.4955%	91.9298%			0.1213 giây
2	94.0909%	94.0351%			0.1056 giây
3	92.2727%	92.2807%			0.1086 giây
<b>Trung bình</b>	<b>93.9530%</b>	<b>92.7485%</b>	<b>0.9356</b>	<b>0.9227</b>	<b>0.1118 giây</b>

Ma trận trên tập dữ liệu test:



Hình 4.57. Ma trận nhầm lẫn trên tập dữ liệu test dùng Undersampling

### Kết quả:

- Có 141 giao dịch bình thường được dự đoán đúng là giao dịch bình thường.
- 6 lần giao dịch bình thường được dự đoán sai là giao dịch bất thường.
- 14 lần giao dịch bất thường được dự đoán sai là giao dịch bình thường.
- 123 lần giao dịch bất thường được dự đoán chính xác là giao dịch bất thường.

**c. Đối với bộ dữ liệu xử lý cân bằng dùng phương pháp kết hợp SMOTE + ENN**  
Thực hiện tách bộ dữ liệu thành hai tập train và test, số lượng giao dịch trên từng nhãn của thuộc tính isFraud trên tập train và test được cho theo bảng dưới đây:

Bảng 4.36. Thống kê số lượng giao dịch của mỗi giá trị ở thuộc tính Class

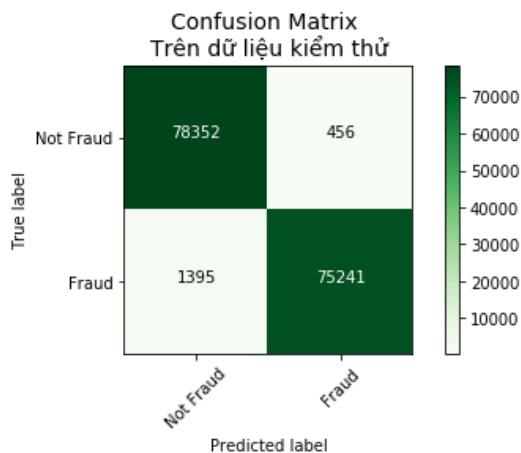
Loại xử lý của bộ dữ liệu	Tập dữ liệu	Số lượng của mỗi phân lớp của thuộc tính Class	
		Class = 0 (giao dịch bình thường)	Class = 1 (giao dịch bất thường)
Đã thực hiện xử lý cân bằng theo phương pháp kết hợp SMOTE + ENN	Ban đầu chưa tách	262695	255458
	Tập train	183886	178821
	Tập test	78809	76637

Tiến hành chạy mô hình bằng thuật toán EGB sử dụng 3 fold-cross validation cho bộ dữ liệu, thu được kết quả (chi tiết lệnh xem phần phụ lục):

Bảng 4.37. Kết quả sau khi chạy thuật toán EGB đối với bộ dữ liệu đã xử lý cân bằng dùng SMOTE + ENN

Lần chạy	Độ chính xác trên tập train	Độ chính xác trên tập test	F1-score trên tập train	F1-score trên tập test	Thời gian
1	98.8156%	98.7713%			64.2233 giây
2	98.7899%	98.8208%			63.9993 giây
3	98.8189%	98.8337%			63.6296 giây
<b>Trung bình</b>	<b>98.8081%</b>	<b>98.8086%</b>	<b>0.9878</b>	<b>0.9878</b>	<b>63.9507 giây</b>

Ma trận trên tập dữ liệu test:



Hình 4.58. Ma trận nhầm lẫn trên tập dữ liệu test dùng SMOTE + ENN

### Kết quả:

- Có 78352 giao dịch bình thường được dự đoán đúng là giao dịch bình thường.
- 456 lần giao dịch bình thường được dự sai đoán là giao dịch bất thường.
- 1395 lần giao dịch bất thường được dự đoán sai là giao dịch bình thường.
- 75241 lần giao dịch bất thường được dự đoán chính xác là giao dịch bất thường.

## CHƯƠNG 5: KẾT LUẬN – HƯỚNG PHÁT TRIỂN

### 5.1. Kết quả đạt được

Bài toán phát hiện giao dịch bất thường là một bài toán thú vị và thu hút được nhiều sự quan tâm của nhiều nhà đầu tư và các nhà nghiên cứu.

Trong khuôn khổ của một khóa luận tốt nghiệp đại học, nhóm tác giả tập trung nghiên cứu các kỹ thuật máy học được áp dụng cho phát hiện giao dịch bất thường. Cụ thể là Random Forest và XGBoost để dự báo rằng một giao dịch có bất thường hay không để phát hiện kịp thời và ngăn chặn qua đó giảm tổn thất và tăng độ uy tín của doanh nghiệp.

Nhóm tác giả cũng nghiên cứu và áp dụng nhiều kỹ thuật xử lý dữ liệu rất mảnh cân bằng, cũng như các phương pháp đánh giá mô hình có dữ liệu rất mảnh cân bằng.

Qua thực nghiệm áp dụng các kỹ thuật ở trên, kết quả thu được từ cả 2 mô hình Random Forest và XGBoost là đáng chú ý. Trong đó mô hình XGBoost có tốc độ thực hiện nhanh hơn nhiều và cho kết quả tương tự Random Forest. Kết quả thu được là khá tương đồng với các nghiên cứu khác mà chúng tôi tham khảo.

Kết quả mà khóa luận đạt được, tuy chưa thật sự xuất sắc nhưng cũng đã đạt được những yêu cầu ban đầu đặt ra và là nền tảng cho các nghiên cứu tiếp theo.

### 5.2. Hướng phát triển

Do còn nhiều hạn chế về thời gian và kiến thức: khóa luận còn một số vấn đề cần tiếp tục nghiên cứu, phát triển và hoàn thiện trong thời gian tới.

Trong tương lai mô hình cần bổ sung thêm các kỹ thuật đánh giá mô hình khác khác cũng như thực hiện thêm các kỹ thuật xử lý dữ liệu mảnh cân bằng để mô hình có thể dự đoán trên dữ liệu chưa từng nhìn thấy tốt hơn.

Tiếp tục nghiên cứu kỹ hơn về lý thuyết các mô hình máy học, tinh chỉnh các tham số của mô hình để tìm được mô hình các mô hình cho kết quả tốt hơn.

Tìm hiểu và cái đặt thêm các mô hình dự đoán khác để so sánh với mô hình hiện tại cũng như tìm mô hình phù hợp.

Tiếp tục phát triển và hoàn thiện, tích hợp vào ứng dụng cụ thể với hy vọng có thể ứng dụng mô hình vào sử dụng trong thực tế ở Việt Nam.

## **TÀI LIỆU THAM KHẢO**

### **Tiếng Việt:**

- [9] Couhp, "RANDOM FOREST, THẾ NÀO LÀ MỘT RỪNG NGẦU NHIÊN," 24 01 2018. [Trực tuyến]. Available: <https://couhpcode.wordpress.com/2018/01/24/random-forest-the-nao-la-mot-rung-ngau-nhien/>.
- [10] N. D. Sim, "Phân lớp bằng Random Forests trong Python," 23 11 2018. [Trực tuyến]. Available: <https://viblo.asia/p/phan-lop-bang-random-forests-trong-python-djeZ1D2QKWz>.
- [12] "Nền tảng của deep learning - Multi-layer Perceptron," 2017. [Trực tuyến]. Available: <https://dlapplications.github.io/2018-06-15-MLP/>.

### **Tiếng Anh:**

- [1] P. Muncaster, "Global Fraud Hits £3.2 Trillion," 22 May 2018. [Online]. Available: <https://www.infosecurity-magazine.com/news/global-fraud-hits-32-trillion/>.
- [2] Javelin, "Identity Fraud Hits Record High with 15.4 Million U.S. Victims in 2016, Up 16 Percent According to New Javelin Strategy & Research Study," [Online]. Available: <https://www.javelinstrategy.com/press-release/identity-fraud-hits-record-high-154-million-us-victims-2016-16-percent-according-new>.
- [3] AltexSoft, "Fraud Detection: How Machine Learning Systems Help Reveal Scams in Fintech, Healthcare, and eCommerce," [Online]. Available: <https://www.altexsoft.com/whitepapers/fraud-detection-how-machine-learning-systems-help-reveal-scams-in-fintech-healthcare-and-eCommerce/>.
- [4] A. Oleksiuk, "How to Use Machine Learning in Fraud Detection,"

- 30 July 2019. [Online]. Available: <https://www.intellias.com/how-to-use-machine-learning-in-fraud-detection/>.
- [5] Machine Learning Group - ULB, "Credit Card Fraud Detection," 23 March 2018. [Online]. Available: <https://www.kaggle.com/mlg-ulb/creditcardfraud>.
- [6] T. @. NTNU, "Synthetic Financial Datasets For Fraud Detection," 04 March 2017. [Online]. Available: <https://www.kaggle.com/ntnu-testimon/paysim1>.
- [7] Edgar Alonso Lopez-Rojas, Ahmad Elmir and Stefan Axelsson, "PAYSIM: A FINANCIAL MOBILE MONEY SIMULATOR FOR FRAUD DETECTION," pp. 249-255, 2016.
- [8] L. Breiman and A. Cutler, "Random Forests," [Online]. Available: [https://www.stat.berkeley.edu/~breiman/RandomForests/cc\\_home.htm#intro](https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#intro).
- [11] T. Chen, T. He and M. Benesty, "Extreme Gradient Boosting," *arXiv*, pp. 785-794, 2016.
- [13] Pierre Carbonnelle, "PYPL PopularitY of Programming Language," 2019. [Online]. Available: <http://pypl.github.io/PYPL.html>.
- [14] A. Joshua, "Predicting Fraud in Financial Payment Services," 2017. [Online]. Available: <https://www.kaggle.com/arjunjoshua/kernels?sortBy=dateRun&group=collaboration&pageSize=20&userId=779333>.
- [15] S. PAFKA, "BENCHMARKING RANDOM FOREST IMPLEMENTATIONS," 19 05 2015. [Online]. Available: <http://datascience.la/benchmarking-random-forest-implementations/>.

## PHỤ LỤC

### Chương 4: Thực nghiệm

#### 4.1.1. Các bước thực hiện

##### Bước 1: Import Library

Thực hiện import các thư viện của python, thư viện xử lý số, bẽ biểu đồ, xây dựng mô hình, và các thư viện về phân lớp dữ liệu.

```
# pip install prompt_toolkit
# ***
# Import library
# Thư viện sử lý số
import pandas as pd
import numpy as np
import time
from collections import Counter
from scipy.stats import norm
from sklearn.manifold import TSNE
from sklearn.decomposition import PCA, TruncatedSVD
import matplotlib.patches as mpatches
# Thu hiện vẽ biểu đồ
%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib.lines as mlines
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns
import itertools
# Thư viện xây dựng mô hình
from sklearn.model_selection import train_test_split, learning_curve
from sklearn.metrics import average_precision_score, auc, accuracy_score, confusion_matrix, mean_squared_error
from sklearn.model_selection import cross_val_score
from imblearn.over_sampling import (RandomOverSampler)
from sklearn.datasets import make_classification
# Classifier Libraries
from sklearn.ensemble import RandomForestClassifier
from xgboost.sklearn import XGBClassifier
```

```
from xgboost import plot_importance, to_graphviz
```

## Bước 2: Connect to google drive

Cách 1: Sử dụng dữ liệu cá nhân

Thao tác 1: Kết nối với Google driver

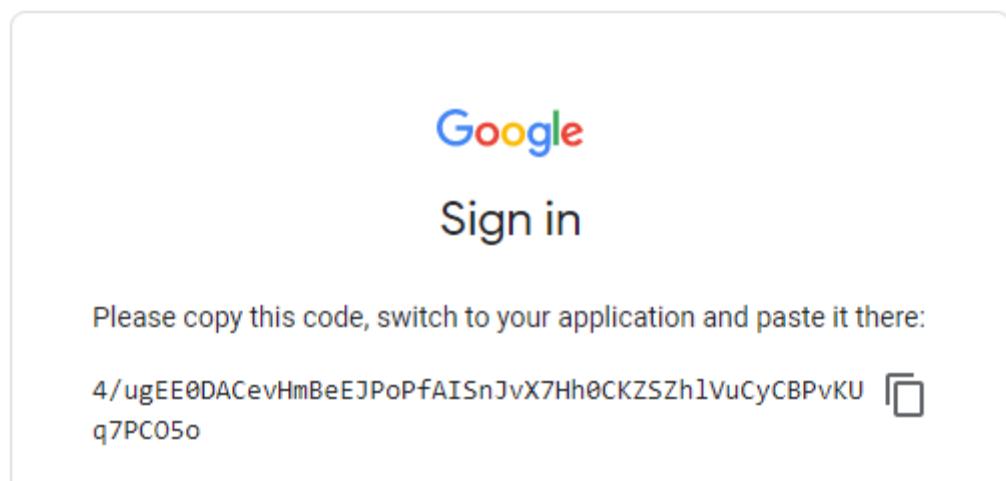
```
# Kết nối với gg driver
from google.colab import drive
drive.mount('drive')
```

Ta được kết quả như sau:

+ Code + Text

```
# Kết nối với gg driver
from google.colab import drive
drive.mount('drive')
# Cách pass auth: Nhấn vào link -> chọn tài khoản google uit -> copy auth -> dán vào ô input bên dưới -> nhấn enter
...
... Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0
Enter your authorization code:
```

Sau đó truy cập vào link như bên trên, đăng nhập bằng tài khoản đã được chia sẻ, sau đó trên màn hình hiển thị:



Ta tiến hành copy đoạn code hiển thị trên màn hình và nhập vào ô code bên google colab, sau đó nhấn enter để hoàn thành:

```
# Kết nối với gg driver
from google.colab import drive
drive.mount('drive')
# Cách pass auth: Nhấn vào link -> chọn tài khoản google uit -> copy auth -> dán vào ô input bên dưới -> nhấn enter
...
... Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0br4i.apps.googleusercontent.com&redirect_uri=
Enter your authorization code:
*****
```

Thao tác 2: Truy cập đường dẫn đến nơi lưu trữ dữ liệu trên Google Drive:

Lệnh	Kết quả
# Đường dẫn đến nơi lưu trữ dữ liệu trong drive path = '/content/drive/My Drive/Colab Notebooks/Graduation Thesis/\\ Data/data 1.csv' print('DONE')	DONE

Cách 2: Chia sẻ cho người khác chạy thử

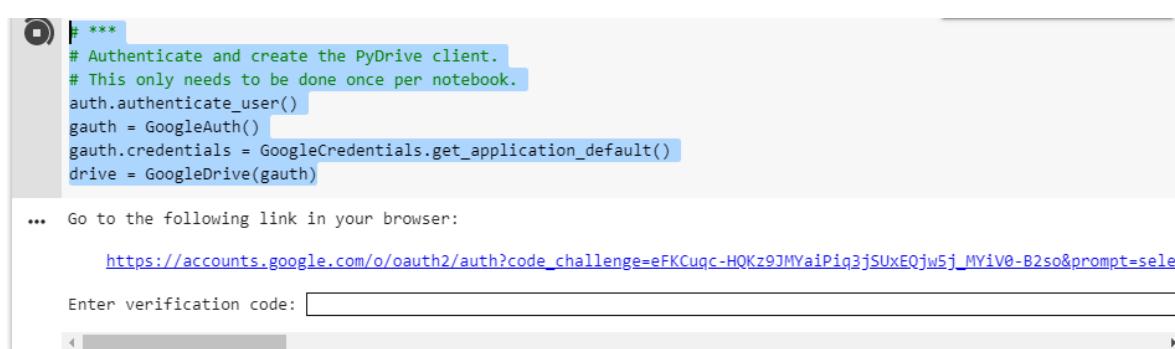
Thao tác 1: Import các thư viện:

```
# ***  
# Import PyDrive and associated libraries.  
# This only needs to be done once per notebook.  
from pydrive.auth import GoogleAuth  
from pydrive.drive import GoogleDrive  
from google.colab import auth  
from oauth2client.client import GoogleCredentials  
# ***
```

Thao tác 2: Kết nối với Google driver:

```
# Authenticate and create the PyDrive client.  
# This only needs to be done once per notebook.  
auth.authenticate_user()  
gauth = GoogleAuth()  
gauth.credentials = GoogleCredentials.get_application_default()  
drive = GoogleDrive(gauth)
```

Tương tự ở trên ta được như sau:



Nhấn vào được link để copy đoạn code sau đó nhập vào ô phía bên dưới, rồi nhấn enter để kết nối.

Thao tác 3: Chạy lệnh bên dưới để tiến hành lấy dữ liệu:

```
# ***
# DỮ LIỆU THÔ
# Download a file based on its file ID.
# là id đường dẫn chia sẻ file tài liệu
# example: https://drive.google.com/file/d/1rNwi9KilujzHH4Q0rzXImW47tbZmR3Ds/view
# ID : 1rNwi9KilujzHH4Q0rzXImW47tbZmR3Ds
file_id = '1iGVOvK2Td5qZdRkv76cMeMGEWX-0P_aN'
downloaded = drive.CreateFile({'id': file_id})
downloaded.GetContentFile('financial.csv') # download file
```

### Bước 3: Đọc dữ liệu vừa kết nối

Đọc dữ liệu theo cách 1:

```
# Đọc dữ liệu cho cách 1
from pandas import read_csv
filename = path
# data = read_csv(filename, names=names)
data = read_csv(filename)
# peek = data.head(10)
```

Đọc dữ liệu theo cách 2:

```
# Đọc dữ liệu cho cách 2:
data = pd.read_csv('creditcard.csv')
```

Hiển thị 5 dòng đầu tiên trong bộ dữ liệu, tiến hành chạy lệnh:

```
data.head(5)
```

Ta được kết quả như sau:

step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0

#### 4.1.6. Xử lý dữ liệu mất cân bằng

```
X = data_clean
```

```

# ***
# Tách dữ liệu giao dịch bình thường và giao dịch bất thường
Xfraud = X.loc[X.isFraud == 1]
# Lấy ra các dòng là giao dịch không gian lận
XnonFraud = X.loc[X.isFraud == 0]
XnonFraud = XnonFraud.sample(frac=1, random_state=42)
print('Số giao dịch bất thường', len(Xfraud))
# Tỷ lệ giao dịch bất thường trên tổng số lượng sao dịch
print('Mất cân bằng = {}%'.format(
    round(len(Xfraud) / float(len(X)), 5)*100))

```

#### 4.1.6.1. Random UnderSampling

```

# ***
# lấy ra 8213 giao dịch bình thường
non_fraud_df = XnonFraud[:8213]
# Gộp dữ liệu bất thường là bình thường
normal_distributed_df = pd.concat([Xfraud, non_fraud_df])
# xáo trộn dữ liệu
df_with_undersampling = normal_distributed_df.sample(frac=1, random_
state=42)
df_with_undersampling = df_with_undersampling.iloc[:,1:]
print('Số giao dịch còn lại: {}'.format(len(df_with_undersampling)))
df_with_undersampling.head()

```

#### 4.1.6.2. Random OverSampling

Thực hiện các bước theo thứ tự để chạy thuật toán Random oversampling:

```

# ***
# Dữ liệu đã làm sạch: data_clean
# Tách dữ liệu giao dịch bình thường và giao dịch bất thường
y_rs = data_clean['isFraud']
# Lấy ra các dòng là giao dịch bình thường
X_rs = data_clean.drop(['isFraud'], axis = 1)
name_x = X_rs.columns
# RandomOverSampler: reference https://imbalanced-
# learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.Rando
# mOverSampler.html
sampler = RandomOverSampler(random_state=42)
X_rs, y_rs = sampler.fit_sample(X_rs, y_rs)

```

```

print('RandomOverSampler {}'.format(Counter(y_rs)))
# Chuyển dữ liệu ndarray to DataFrame
X_rs = pd.DataFrame(X_rs)
y_rs = pd.DataFrame(y_rs)
X_rs = X_rs.iloc[:,1:]
X_rs.columns = name_x[1:]
y_rs.columns = ['isFraud']
# Gộp dữ liệu bất thường và bình thường
normal_distributed_df = pd.concat([X_rs, y_rs], axis=1)
# xáo trộn dữ liệu
df_with_oversampling = normal_distributed_df.sample(frac=1, random_state=42)
df_with_oversampling.head(5)

```

#### 4.1.9. Lưu dữ liệu sau khi thực hiện cân bằng

```

from google.colab import drive
drive.mount('drive')
# Dự liệu sau khi làm sạch bằng undersampling, dùng để train.
new_df.to_csv('/content/drive/My Drive/Colab Notebooks/Graduation Thesis/Data/data_under_sampling.csv')
print('=====DONE=====')
# Dự liệu sau khi làm sạch bằng oversampling, dùng để train.
df_with_oversampling.to_csv('/content/drive/My Drive/Colab Notebooks/Graduation Thesis/Data/data_over_sampling.csv')
print('=====DONE=====')

```

#### 4.1.10. Mô hình phát hiện bất thường sử dụng thuật toán Random forest

##### 4.1.10.1. Với bộ dữ liệu chưa xử lý mất cân bằng

Ta tiến hành chạy mô hình bằng thuật toán Random forest cho bộ dữ liệu, sử dụng 3 fold-cross validation:

```

# Load data
# data = pd.read_csv('data_name.csv', index_col=0)
data_clean = data_clean.iloc[:, 1:]
X = data_clean.drop('isFraud', axis=1)
y = data_clean['isFraud']
# Tách dữ liệu thành train set và test set
# Chúng tôi sẽ sử dụng xác thực chéo trên tập huấn luyện để điều chỉnh các tham số, sau đó kiểm tra dữ liệu chưa xem

```

```

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0
.5, shuffle=True, stratify=y, random_state=42)
y_test.value_counts()
# Mô hình
model = RandomForestClassifier(n_estimators=50, n_jobs=-1,
max_features='sqrt', random_state=42)
# Danh sách confusion matrix của từng lần KFold
conf_matrix_list_of_arrays = []
conf_matrix_list_of_arrays_test_set = []
# Danh sách độ chính xác của từng lần KFold trên dữ liệu test
accuracy_models = []
accuracy_test_set = []
f1_score_train = []
f1_score_test = []
# KFold = 3
kf = StratifiedKFold(n_splits=3)
kf.get_n_splits(X_train, y_train)
# Thời gian bắt đầu train
start_ho = timer()
for train_index, test_index in kf.split(X_train, y_train):
    X_tr, X_te = X_train.iloc[train_index], X_train.iloc[test_index]
    y_tr, y_te = y_train.iloc[train_index], y_train.iloc[test_index]
    model.fit(X_tr, y_tr)
    y_pre_train = model.predict(X_te)
    y_pre_test = model.predict(X_test)
    # Print the accuracy
    accuracy_models.append(
        accuracy_score(y_te, y_pre_train, normalize=True)*100)
    accuracy_test_set.append(
        accuracy_score(y_test, y_pre_test, normalize=True)*100)
    # Confusion matrix
    conf_matrix = confusion_matrix(y_te, y_pre_train)
    conf_matrix_list_of_arrays.append(conf_matrix)
    conf_matrix_2 = confusion_matrix(y_test, y_pre_test)
    conf_matrix_list_of_arrays_test_set.append(conf_matrix_2)
    # F1 score
    f1_score_train.append(f1_score(y_te, y_pre_train))

```

```

f1_score_test.append(f1_score(y_test, y_pre_test))

# Độ chính xác của mô hình
print('Độ chính xác của mô hình trên dữ liệu train: ', accuracy_model_s)
print('Độ chính xác của mô hình trên dữ liệu test: ', accuracy_test_s)
print('Độ chính xác trung bình dữ liệu train: ', np.mean(accuracy_mode_ls))
print('Độ chính xác trung bình dữ liệu test: ', np.mean(accuracy_test_set))
print('F1 score trung bình trên dữ liệu train: ', np.mean(f1_score_train))
print('F1 score trung bình trên dữ liệu test: ', np.mean(f1_score_test))

# Thời gian kết thúc train
end_ho = timer()
# Tổng thời thực hiện
time_ho = (end_ho - start_ho)
print('Thời gian thực hiện: ', time_ho, ' giây')

```

#### 4.1.10.2. Với bộ dữ liệu đã xử lý mất cân bằng

Với dữ liệu đã xử lý mất cân bằng – overfitting. Ta tiến hành chạy mô hình bằng thuật toán Random forest cho bộ dữ liệu, sử dụng 3 fold-cross validation:

```

print(df_with_oversampling['isFraud'].value_counts())
df_with_oversampling.head()

# df_with_oversampling = df_with_oversampling.iloc[:, 1:]
# Tách các thuộc tính độc lập và thuộc tính phụ thuộc
y = df_with_oversampling['isFraud']
X = df_with_oversampling.drop('isFraud', axis=1)
# Chia tập dữ liệu 2 phần: tập huấn luyện và tập kiểm thử
# Tỷ lệ: train 60%, test 40%
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=42)
# Mô hình

model = RandomForestClassifier(n_estimators=50, n_jobs=-1, max_features='sqrt', random_state=42)
# Danh sách confusion matrix của từng lần KFold

```

```

conf_matrix_list_of_arrays = []
conf_matrix_list_of_arrays_test_set = []
# Danh sách độ chính xác của từng lần KFold trên dữ liệu test
accuracy_models = []
accuracy_test_set = []
f1_score_train = []
f1_score_test = []
# KFold = 3
kf = KFold(n_splits=3)
kf.get_n_splits(X_train)
# Thời gian bắt đầu train
start_ho = timer()
for train_index, test_index in kf.split(X_train):
    X_tr, X_te = X_train.iloc[train_index], X_train.iloc[test_index]
    y_tr, y_te = y_train.iloc[train_index], y_train.iloc[test_index]
    model.fit(X_tr, y_tr)
    # Print the accuracy
    y_pre_train = model.predict(X_te)
    y_pre_test = model.predict(X_test)
    accuracy_models.append(
        accuracy_score(y_te, y_pre_train, normalize=True)*100)
    accuracy_test_set.append(
        accuracy_score(y_test, y_pre_test, normalize=True)*100)
    # Confusion matrix
    conf_matrix = confusion_matrix(y_te, y_pre_train)
    conf_matrix_list_of_arrays.append(conf_matrix)
    conf_matrix_2 = confusion_matrix(y_test, y_pre_test)
    conf_matrix_list_of_arrays_test_set.append(conf_matrix_2)
    # F1 score
    f1_score_train.append(f1_score(y_te, y_pre_train))
    f1_score_test.append(f1_score(y_test, y_pre_test))
# Độ chính xác của mô hình
print('Độ chính xác của mô hình trên dữ liệu train: ', accuracy_models)
print('Độ chính xác của mô hình trên dữ liệu test: ', accuracy_test_set)

```

```

print('Độ chính xác trung bình dữ liệu train:', np.mean(accuracy_modes))
print('Độ chính xác trung bình dữ liệu test:', np.mean(accuracy_test_set))
print('F1 score trung bình trên dữ liệu train:', np.mean(f1_score_train))
print('F1 score trung bình trên dữ liệu test:', np.mean(f1_score_test))
# Thời gian kết thúc train
end_ho = timer()
# Tổng thời thực hiện
time_ho = (end_ho - start_ho)
print('Thời gian thực hiện: ', time_ho, ' giây')

```

Với dữ liệu đã xử lý mất cân bằng – underfitting: Ta tiến hành chạy mô hình bằng thuật toán Random forest cho bộ dữ liệu, sử dụng 3 fold-cross validation:

```

print(data_under_sampling['isFraud'].value_counts())
data_under_sampling.head()
# data_under_sampling = data_under_sampling.iloc[:, 1:]
# Tách các thuộc tính độc lập và thuộc tính phụ thuộc
y = data_under_sampling['isFraud']
X = data_under_sampling.drop('isFraud', axis=1)
# Chia tập dữ liệu 2 phần: tập huấn luyện và tập kiểm thử
# Tỷ lệ: train 70%, test 30%
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
print(y_train.value_counts())
print(y_test.value_counts())
# Mô hình
model = RandomForestClassifier(n_estimators=50, n_jobs=-1, max_features='sqrt', random_state=42)
# Danh sách confusion matrix của từng lần KFold
conf_matrix_list_of_arrays = []
conf_matrix_list_of_arrays_test_set = []
# Danh sách độ chính xác của từng lần KFold trên dữ liệu test
accuracy_models = []
accuracy_test_set = []

```

```

f1_score_train = []
f1_score_test = []
# KFold = 3
kf = KFold(n_splits=3)
kf.get_n_splits(X_train)
# Thời gian bắt đầu train
start_ho = timer()
for train_index, test_index in kf.split(X_train):
    X_tr, X_te = X_train.iloc[train_index], X_train.iloc[test_index]
    y_tr, y_te = y_train.iloc[train_index], y_train.iloc[test_index]
    model.fit(X_tr, y_tr)
    # Print the accuracy
    y_pre_train = model.predict(X_te)
    y_pre_test = model.predict(X_test)
    accuracy_models.append(
        accuracy_score(y_te, y_pre_train, normalize=True)*100)
    accuracy_test_set.append(
        accuracy_score(y_test, y_pre_test, normalize=True)*100)
    # Confusion matrix
    conf_matrix = confusion_matrix(y_te, y_pre_train)
    conf_matrix_list_of_arrays.append(conf_matrix)
    conf_matrix_2 = confusion_matrix(y_test, y_pre_test)
    conf_matrix_list_of_arrays_test_set.append(conf_matrix_2)
    # F1 score
    f1_score_train.append(f1_score(y_te, y_pre_train))
    f1_score_test.append(f1_score(y_test, y_pre_test))
# Độ chính xác của mô hình
print('Độ chính xác của mô hình trên dữ liệu train: ', accuracy_models)
print('Độ chính xác của mô hình trên dữ liệu test: ', accuracy_test_set)
print('Độ chính xác trung bình dữ liệu train:', np.mean(accuracy_models))
print('Độ chính xác trung bình dữ liệu test:', np.mean(accuracy_test_set))
print('F1 score trung bình trên dữ liệu train:', np.mean(f1_score_train))

```

```

print('F1 score trung bình trên dữ liệu test:', np.mean(f1_score_test
))
# Thời gian kết thúc train
end_ho = timer()
# Tổng thời thực hiện
time_ho = (end_ho - start_ho)
print('Thời gian thực hiện: ', time_ho, ' giây')

```

#### 4.1.11. Mô hình phát hiện bát thường sử dụng thuật toán Extreme gradient boosting (EGB)

##### 4.1.11.1. Với bộ dữ liệu chưa xử lý mất cân bằng

Ta tiến hành chạy mô hình bằng thuật toán EGB cho bộ dữ liệu, sử dụng 3 fold-cross validation:

```

data_clean['isFraud'].value_counts()
# Load data
data_clean = data_clean.iloc[:, 1:]
X = data_clean.drop('isFraud', axis=1)
y = data_clean['isFraud']
# Tách dữ liệu thành train set và test set
# Chúng tôi sẽ sử dụng xác thực chéo trên tập huấn luyện để điều chỉnh các tham số, sau đó kiểm tra dữ liệu chưa xem
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, shuffle=True, stratify=y, random_state=42)
# Mô hình
model = XGBClassifier(objective="binary:logistic", random_state=42)
# Danh sách confusion matrix của từng lần KFold
conf_matrix_list_of_arrays = []
conf_matrix_list_of_arrays_test_set = []
# Danh sách độ chính xác của từng lần KFold trên dữ liệu test
accuracy_models = []
accuracy_test_set = []
f1_score_train = []
f1_score_test = []
# KFold = 3
kf = StratifiedKFold(n_splits=3)
kf.get_n_splits(X_train, y_train)

```

```

# Thời gian bắt đầu train
start_ho = timer()
for train_index, test_index in kf.split(X_train, y_train):
    X_tr, X_te = X_train.iloc[train_index], X_train.iloc[test_index]
    y_tr, y_te = y_train.iloc[train_index], y_train.iloc[test_index]
    model.fit(X_tr, y_tr)
    y_pre_train = model.predict(X_te)
    y_pre_test = model.predict(X_test)
    # Print the accuracy
    accuracy_models.append(
        accuracy_score(y_te, y_pre_train, normalize=True)*100)
    accuracy_test_set.append(
        accuracy_score(y_test, y_pre_test, normalize=True)*100)
    # Confusion matrix
    conf_matrix = confusion_matrix(y_te, y_pre_train)
    conf_matrix_list_of_arrays.append(conf_matrix)
    conf_matrix_2 = confusion_matrix(y_test, y_pre_test)
    conf_matrix_list_of_arrays_test_set.append(conf_matrix_2)
    # F1 score
    f1_score_train.append(f1_score(y_te, y_pre_train))
    f1_score_test.append(f1_score(y_test, y_pre_test))
# Độ chính xác của mô hình
print('Độ chính xác của mô hình trên dữ liệu train: ', accuracy_models)
print('Độ chính xác của mô hình trên dữ liệu test: ', accuracy_test_set)
print('Độ chính xác trung bình dữ liệu train:', np.mean(accuracy_models))
print('Độ chính xác trung bình dữ liệu test:', np.mean(accuracy_test_set))
print('F1 score trung bình trên dữ liệu train:', np.mean(f1_score_train))
print('F1 score trung bình trên dữ liệu test:', np.mean(f1_score_test))
# Thời gian kết thúc train
end_ho = timer()
# Tổng thời thực hiện

```

```

time_ho = (end_ho - start_ho)
print('Thời gian thực hiện: ', time_ho, ' giây')

```

#### 4.1.11.2. Với bộ dữ liệu đã xử lý mất cân bằng

a. Đối với bộ dữ liệu xử lý cân bằng dùng phương pháp Oversampling

Ta tiến hành chạy mô hình bằng thuật toán EGB cho bộ dữ liệu cân bằng theo phương pháp Oversampling, sử dụng 3 fold-cross validation:

```

print(df_with_oversampling['isFraud'].value_counts())
df_with_oversampling.head()
# Load data
# df_with_oversampling = df_with_oversampling.iloc[:, 1:]
X = df_with_oversampling.drop('isFraud', axis=1)
y = df_with_oversampling['isFraud']
# Tách dữ liệu thành train set và test set
# Chúng tôi sẽ sử dụng xác thực chéo trên tập huấn luyện để điều chỉnh các tham số, sau đó kiểm tra dữ liệu chưa xem
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=True, stratify=y, random_state=42)
# Mô hình
model = XGBClassifier(objective="binary:logistic", random_state=42)
# Danh sách confusion matrix của từng lần KFold
conf_matrix_list_of_arrays = []
conf_matrix_list_of_arrays_test_set = []
# Danh sách độ chính xác của từng lần KFold trên dữ liệu test
accuracy_models = []
accuracy_test_set = []
f1_score_train = []
f1_score_test = []
# KFold = 3
kf = StratifiedKFold(n_splits=3)
kf.get_n_splits(X_train, y_train)
# Thời gian bắt đầu train
start_ho = timer()
for train_index, test_index in kf.split(X_train, y_train):
    X_tr, X_te = X_train.iloc[train_index], X_train.iloc[test_index]
    y_tr, y_te = y_train.iloc[train_index], y_train.iloc[test_index]
    model.fit(X_tr, y_tr)

```

```

y_pre_train = model.predict(X_te)
y_pre_test = model.predict(X_test)
# Print the accuracy
accuracy_models.append(
    accuracy_score(y_te, y_pre_train, normalize=True)*100)
accuracy_test_set.append(
    accuracy_score(y_test, y_pre_test, normalize=True)*100)

# Confusion matrix
conf_matrix = confusion_matrix(y_te, y_pre_train)
conf_matrix_list_of_arrays.append(conf_matrix)
conf_matrix_2 = confusion_matrix(y_test, y_pre_test)
conf_matrix_list_of_arrays_test_set.append(conf_matrix_2)

# F1 score
f1_score_train.append(f1_score(y_te, y_pre_train))
f1_score_test.append(f1_score(y_test, y_pre_test))

# Độ chính xác của mô hình
print('Độ chính xác của mô hình trên dữ liệu train: ', accuracy_models)
print('Độ chính xác của mô hình trên dữ liệu test: ', accuracy_test_set)
print('Độ chính xác trung bình dữ liệu train:', np.mean(accuracy_models))
print('Độ chính xác trung bình dữ liệu test:', np.mean(accuracy_test_set))
print('F1 score trung bình trên dữ liệu train:', np.mean(f1_score_train))
print('F1 score trung bình trên dữ liệu test:', np.mean(f1_score_test))

# Thời gian kết thúc train
end_ho = timer()
# Tổng thời thực hiện
time_ho = (end_ho - start_ho)
print('Thời gian thực hiện: ', time_ho, ' giây')

```

## 4.2. Bộ dữ liệu Credit Card Fraud Detection

### 4.2.1. Các bước thực hiện

Tương tự như phụ lục 4.1.1

#### 4.2.5. Xử lý dữ liệu mất cân bằng

##### 4.2.5.1. Random Undersampling

```
# Xáo trộn dữ liệu
data_raw = data_raw.sample(frac=1)
# Số lượng giao dịch gian lận là 492 dòng.
fraud_df = data_raw.loc[data_raw['Class'] == 1]
non_fraud_df = data_raw.loc[data_raw['Class'] == 0][:492]
normal_distributed_df = pd.concat([fraud_df, non_fraud_df])
# Xáo trộn dữ liệu một lần nữa
new_df = normal_distributed_df.sample(frac=1, random_state=42)
new_df.head()
```

##### 4.2.5.2. Random Oversampling (SMOTE)

```
X = data_raw.drop('Class', axis=1)
y = data_raw['Class']
print("Trước SMOTE, số lượng giao dịch gian lận là: {}".format(sum(y==1)))
print("Trước SMOTE, số lượng giao dịch không gian lận là: {} \n".format(sum(y==0)))
sm = SMOTE(random_state=42)
X_res, y_res = sm.fit_sample(X, y.ravel())
print('Sau SMOTE, Kích thước của dữ liệu train_X: {}'.format(X_res.shape))
print('Sau SMOTE, Kích thước của dữ liệu train_y: {} \n'.format(y_res.shape))
print("Sau SMOTE, Số lượng giao dịch gian lận là: {}".format(sum(y_res==1)))
print("Sau SMOTE, Số lượng giao dịch không gian lận là: {}".format(sum(y_res==0)))
# Convert 'ndarray' to 'DataFrame'
X_res = pd.DataFrame(X_res, columns=X.columns)
y_res = pd.DataFrame(y_res, columns=['Class'])
print(X_train.shape)
print(y_train.shape)
data_smote = pd.concat([X_res, y_res], axis=1)
print(data_smote.shape)
```

##### 4.2.5.3. Phương pháp kết hợp SMOTE + ENN

```

X = data_raw.drop('Class', axis=1)
y = data_raw['Class']
print("Truóc SMOTE + ENN, số lượng giao dịch gian lận là: {}".format(
sum(y==1)))
print("Truóc SMOTE + ENN, số lượng giao dịch không gian lận là: {} \n".format(sum(y==0)))
sm = SMOTEENN(random_state=42)
X_res, y_res = sm.fit_sample(X, y.ravel())
print('Sau SMOTE + ENN, Kích thước của dữ liệu train_X: {}'.format(X_res.shape))
print('Sau SMOTE + ENN, Kích thước của dữ liệu train_y: {} \n'.format(y_res.shape))
print("Sau SMOTE + ENN, Số lượng giao dịch gian lận là: {}".format(sum(y_res==1)))
print("Sau SMOTE + ENN, Số lượng giao dịch không gian lận là: {}".format(sum(y_res==0)))
# Convert 'ndarray' to 'DataFrame'
X_res = pd.DataFrame(X_res, columns=X.columns)
y_res = pd.DataFrame(y_res, columns=['Class'])
print(X_res.shape)
print(y_res.shape)
data_smote_enn = pd.concat([X_res, y_res], axis=1)
print(data_smote_enn.shape)

```

#### 4.2.9. Lưu dữ liệu sau khi thực hiện cân bằng dữ liệu

```

from google.colab import drive
drive.mount('drive')
# Dự liệu sau khi làm sạch bằng undersampling, dùng để train.
new_df.to_csv('/content/drive/My Drive/Colab Notebooks/Graduation Thesis/Data/data_credit_undersampling.csv')
print('=====DONE=====')
# Dự liệu sau khi làm sạch bằng undersampling, dùng để train.
new_df.to_csv('/content/drive/My Drive/Colab Notebooks/Graduation Thesis/Data/data_credit_smote.csv')
print('=====DONE=====')
# Dự liệu sau khi làm sạch bằng undersampling, dùng để train.

```

```

data_smote_enn.to_csv('/content/drive/My Drive/Colab Notebooks/Graduation Thesis/Data/data_credit_smote_enn.csv')
print('=====DONE=====')

```

#### 4.2.10. Mô hình phát hiện bất thường sử dụng thuật toán Random forest

##### 4.2.10.1. Với bộ dữ liệu chưa xử lý mất cân bằng

Ta tiến hành chạy mô hình bằng thuật toán Random forest cho bộ dữ liệu, sử dụng 3 fold-cross validation:

```

# Mô hình

model = RandomForestClassifier(n_estimators=50, n_jobs=-1,
max_features='sqrt', random_state=42)
# Danh sách confusion matrix của từng lần KFold
conf_matrix_list_of_arrays = []
conf_matrix_list_of_arrays_test_set = []
# Danh sách độ chính xác của từng lần KFold trên dữ liệu test
accuracy_train_set = []
accuracy_test_set = []
f1_score_train = []
f1_score_test = []
# Cross Validation: KFold = 3
cv = StratifiedKFold(n_splits=3, random_state=None, shuffle=True)
cv.get_n_splits(X_train, y_train)
# List time train
times_train = []
for train_index, test_index in cv.split(X_train, y_train):
    undersample_Xtrain, undersample_Xtest = X_train.iloc[train_index],
    X_train.iloc[test_index]
    undersample_ytrain, undersample_ytest = y_train.iloc[train_index],
    y_train.iloc[test_index]
    # Thời gian bắt đầu train
    start_time = timer()
    model.fit(undersample_Xtrain, undersample_ytrain)
    y_pre_train = model.predict(undersample_Xtest)
    y_pre_test = model.predict(X_test)
    # Print the accuracy
    accuracy_train_set.append(

```

```

accuracy_score(undersample_ytest, y_pre_train, normalize=True
)*100)

accuracy_test_set.append(
    accuracy_score(y_test, y_pre_test, normalize=True)*100)

# Confusion matrix
conf_matrix = confusion_matrix(undersample_ytest, y_pre_train)
conf_matrix_list_of_arrays.append(conf_matrix)
conf_matrix_2 = confusion_matrix(y_test, y_pre_test)
conf_matrix_list_of_arrays_test_set.append(conf_matrix_2)

# F1 score
f1_score_train.append(f1_score(undersample_ytest, y_pre_train))
f1_score_test.append(f1_score(y_test, y_pre_test))

# Thời gian kết thúc train
end_time = timer()

# Tính thời gian thực hiện
time_train = (end_time - start_time)
times_train.append(time_train)

# Độ chính xác của mô hình
print('Độ chính xác của mô hình trên dữ liệu train: ', accuracy_train_set)
print('Độ chính xác của mô hình trên dữ liệu test: ', accuracy_test_set)

print('Độ chính xác trung bình dữ liệu train:', np.mean(accuracy_train_set))
print('Độ chính xác trung bình dữ liệu test:', np.mean(accuracy_test_set))

print('F1 score trung bình trên dữ liệu train:', np.mean(f1_score_train))
print('F1 score trung bình trên dữ liệu test:', np.mean(f1_score_test))

print('Thời gian thực hiện: ', times_train)

```

#### 4.2.10.2. Với bộ dữ liệu đã xử lý mất cân bằng

a. Đối với bộ dữ liệu xử lý cân bằng dùng phương pháp Oversampling

```

X = data_smote.drop('Class', axis=1)
y = data_smote['Class']

# Tách dữ liệu thành train set và test set

```

```

# Chúng tôi sẽ sử dụng xác thực chéo trên tập huấn luyện để điều chỉnh
# các tham số, sau đó kiểm tra dữ liệu chưa xem
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, shuffle=True, stratify=y, random_state=42)
print('Số lượng giao dịch trên từng nhãn trên dữ liệu huấn luyện:\n')
)
print(y_train.value_counts())
print('Số lượng giao dịch trên từng nhãn trên dữ liệu kiểm thử:\n')
print(y_test.value_counts())
# Mô hình
model = RandomForestClassifier(n_estimators=50, n_jobs=-1,
max_features='sqrt', random_state=42)
# Danh sách confusion matrix của từng lần KFold
conf_matrix_list_of_arrays = []
conf_matrix_list_of_arrays_test_set = []
# Danh sách độ chính xác của từng lần KFold trên dữ liệu test
accuracy_train_set = []
accuracy_test_set = []
f1_score_train = []
f1_score_test = []
# Cross Validation: KFold = 3
cv = StratifiedKFold(n_splits=3, random_state=None, shuffle=True)
cv.get_n_splits(X_train, y_train)
# List time train
times_train = []
for train_index, test_index in cv.split(X_train, y_train):
    undersample_Xtrain, undersample_Xtest = X_train.iloc[train_index],
    X_train.iloc[test_index]
    undersample_ytrain, undersample_ytest = y_train.iloc[train_index],
    y_train.iloc[test_index]
    # Thời gian bắt đầu train
    start_time = timer()

    model.fit(undersample_Xtrain, undersample_ytrain)
    y_pre_train = model.predict(undersample_Xtest)
    y_pre_test = model.predict(X_test)
    # Print the accuracy

```

```

accuracy_train_set.append(
    accuracy_score(undersample_ytest, y_pre_train, normalize=True
)*100)

accuracy_test_set.append(
    accuracy_score(y_test, y_pre_test, normalize=True)*100)

# Confusion matrix
conf_matrix = confusion_matrix(undersample_ytest, y_pre_train)
conf_matrix_list_of_arrays.append(conf_matrix)
conf_matrix_2 = confusion_matrix(y_test, y_pre_test)
conf_matrix_list_of_arrays_test_set.append(conf_matrix_2)

# F1 score
f1_score_train.append(f1_score(undersample_ytest, y_pre_train))
f1_score_test.append(f1_score(y_test, y_pre_test))

# Thời gian kết thúc train
end_time = timer()

# Tính thời gian thực hiện
time_train = (end_time - start_time)
times_train.append(time_train)

# Độ chính xác của mô hình
print('Độ chính xác của mô hình trên dữ liệu train: ', accuracy_train_set)
print('Độ chính xác của mô hình trên dữ liệu test: ', accuracy_test_set)
print('Độ chính xác trung bình dữ liệu train:', np.mean(accuracy_train_set))
print('Độ chính xác trung bình dữ liệu test:', np.mean(accuracy_test_set))
print('F1 score trung bình trên dữ liệu train:', np.mean(f1_score_train))
print('F1 score trung bình trên dữ liệu test:', np.mean(f1_score_test))
print('Thời gian thực hiện: ', times_train)

```

## b. Đối với bộ dữ liệu xử lý cân bằng dùng phương pháp Undersampling

```

X = new_df.drop('Class', axis=1)
y = new_df['Class']
# X = data_undersampling.drop('Class', axis=1)

```

```

# y = data_undersampling['Class']
print('Số lượng giao dịch trên từng nhãn:\n', y.value_counts())
# Tách dữ liệu thành train set và test set
# Chúng tôi sẽ sử dụng xác thực chéo trên tập huấn luyện để điều chỉnh
# các tham số, sau đó kiểm tra dữ liệu chưa xem
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, shuffle=True, stratify=y, random_state=42)
print('Số lượng giao dịch trên từng nhãn trên dữ liệu huấn luyện:\n')
)
print(y_train.value_counts())
print('Số lượng giao dịch trên từng nhãn trên dữ liệu kiểm thử:\n')
print(y_test.value_counts())
# Mô hình
model = RandomForestClassifier(n_estimators=50, n_jobs=-1,
max_features='sqrt', random_state=42)
# Danh sách confusion matrix của từng lần KFold
conf_matrix_list_of_arrays = []
conf_matrix_list_of_arrays_test_set = []
# Danh sách độ chính xác của từng lần KFold trên dữ liệu test
accuracy_train_set = []
accuracy_test_set = []
f1_score_train = []
f1_score_test = []
# Cross Validation: KFold = 3
cv = StratifiedKFold(n_splits=3, random_state=None, shuffle=True)
cv.get_n_splits(X_train, y_train)
# List time train
times_train = []
for train_index, test_index in cv.split(X_train, y_train):
    undersample_Xtrain, undersample_Xtest = X_train.iloc[train_index],
    X_train.iloc[test_index]
    undersample_ytrain, undersample_ytest = y_train.iloc[train_index],
    y_train.iloc[test_index]
    # Thời gian bắt đầu train
    start_time = timer()
    model.fit(undersample_Xtrain, undersample_ytrain)
    y_pre_train = model.predict(undersample_Xtest)

```

```

y_pre_test = model.predict(X_test)
# Print the accuracy
accuracy_train_set.append(
    accuracy_score(undersample_ytest, y_pre_train, normalize=True
)*100)
accuracy_test_set.append(
    accuracy_score(y_test, y_pre_test, normalize=True)*100)
# Confusion matrix
conf_matrix = confusion_matrix(undersample_ytest, y_pre_train)
conf_matrix_list_of_arrays.append(conf_matrix)
conf_matrix_2 = confusion_matrix(y_test, y_pre_test)
conf_matrix_list_of_arrays_test_set.append(conf_matrix_2)
# F1 score
f1_score_train.append(f1_score(undersample_ytest, y_pre_train))
f1_score_test.append(f1_score(y_test, y_pre_test))
# Thời gian kết thúc train
end_time = timer()
# Tính thời gian thực hiện
time_train = (end_time - start_time)
times_train.append(time_train)
# Độ chính xác của mô hình
print('Độ chính xác của mô hình trên dữ liệu train: ', accuracy_train_set)
print('Độ chính xác của mô hình trên dữ liệu test: ', accuracy_test_set)
print('Độ chính xác trung bình dữ liệu train:', np.mean(accuracy_train_set))
print('Độ chính xác trung bình dữ liệu test:', np.mean(accuracy_test_set))
print('F1 score trung bình trên dữ liệu train:', np.mean(f1_score_train))
print('F1 score trung bình trên dữ liệu test:', np.mean(f1_score_test))
print('Thời gian thực hiện: ', times_train)
c. Đối với bộ dữ liệu xử lý cân bằng dùng phương pháp kết hợp SMOTE + ENN
X = data_smote_enn.drop('Class', axis=1)

```

```

y = data_smote_enn['Class']
print('Số lượng giao dịch trên từng nhãn:\n', y.value_counts())
# Tách dữ liệu thành train set và test set
# Chúng tôi sẽ sử dụng xác thực chéo trên tập huấn luyện để điều chỉnh
# các tham số, sau đó kiểm tra dữ liệu chưa xem
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, shuffle=True, stratify=y, random_state=42)
print('Số lượng giao dịch trên từng nhãn trên dữ liệu huấn luyện:\n')
)
print(y_train.value_counts())
print('Số lượng giao dịch trên từng nhãn trên dữ liệu kiểm thử:\n')
print(y_test.value_counts())
# Mô hình
model = RandomForestClassifier(n_estimators=50, n_jobs=-1,
max_features='sqrt', random_state=42)
# Danh sách confusion matrix của từng lần KFold
conf_matrix_list_of_arrays = []
conf_matrix_list_of_arrays_test_set = []
# Danh sách độ chính xác của từng lần KFold trên dữ liệu test
accuracy_train_set = []
accuracy_test_set = []
f1_score_train = []
f1_score_test = []
# Cross Validation: KFold = 3
cv = StratifiedKFold(n_splits=3, random_state=None, shuffle=True)
cv.get_n_splits(X_train, y_train)
# List time train
times_train = []
for train_index, test_index in cv.split(X_train, y_train):
    smote_enn_Xtrain, smote_enn_Xtest = X_train.iloc[train_index], X_
    train.iloc[test_index]
    smote_enn_ytrain, smote_enn_ytest = y_train.iloc[train_index], y_
    train.iloc[test_index]
    # Thời gian bắt đầu train
    start_time = timer()
    model.fit(smote_enn_Xtrain, smote_enn_ytrain)
    y_pre_train = model.predict(smote_enn_Xtest)

```

```

y_pre_test = model.predict(X_test)
# Print the accuracy
accuracy_train_set.append(
    accuracy_score(smote_enn_ytest, y_pre_train, normalize=True)*
100)
accuracy_test_set.append(
    accuracy_score(y_test, y_pre_test, normalize=True)*100)
# Confusion matrix
conf_matrix = confusion_matrix(smote_enn_ytest, y_pre_train)
conf_matrix_list_of_arrays.append(conf_matrix)
conf_matrix_2 = confusion_matrix(y_test, y_pre_test)
conf_matrix_list_of_arrays_test_set.append(conf_matrix_2)
# F1 score
f1_score_train.append(f1_score(smote_enn_ytest, y_pre_train))
f1_score_test.append(f1_score(y_test, y_pre_test))
# Thời gian kết thúc train
end_time = timer()
# Tính thời gian thực hiện
time_train = (end_time - start_time)
times_train.append(time_train)
# Độ chính xác của mô hình
print('Độ chính xác của mô hình trên dữ liệu train: ', accuracy_train_
_set)
print('Độ chính xác của mô hình trên dữ liệu test: ', accuracy_test_s_
et)
print('Độ chính xác trung bình dữ liệu train:', np.mean(accuracy_trai_
n_set))
print('Độ chính xác trung bình dữ liệu test:', np.mean(accuracy_test_-
set))
print('F1 score trung bình trên dữ liệu train:', np.mean(f1_score_trai_
in))
print('F1 score trung bình trên dữ liệu test:', np.mean(f1_score_test_))
print('Thời gian thực hiện: ', times_train)

```

#### 4.2.11. Mô hình phát hiện bất thường sử dụng thuật toán Extreme Gradient Boosting (EGB)

#### 4.2.11.1. Với bộ dữ liệu chưa xử lý mất cân bằng

```
# Mô hình
model = XGBClassifier(objective="binary:logistic", random_state=42)
# Danh sách confusion matrix của từng lần KFold
conf_matrix_list_of_arrays = []
conf_matrix_list_of_arrays_test_set = []
# Danh sách độ chính xác của từng lần KFold trên dữ liệu test
accuracy_train_set = []
accuracy_test_set = []
f1_score_train = []
f1_score_test = []
# Cross Validation: KFold = 3
cv = StratifiedKFold(n_splits=3, random_state=None, shuffle=True)
cv.get_n_splits(X_train, y_train)
# List time train
times_train = []
for train_index, test_index in cv.split(X_train, y_train):
    undersample_Xtrain, undersample_Xtest = X_train.iloc[train_index],
    X_train.iloc[test_index]
    undersample_ytrain, undersample_ytest = y_train.iloc[train_index],
    y_train.iloc[test_index]
    # Thời gian bắt đầu train
    start_time = timer()
    model.fit(undersample_Xtrain, undersample_ytrain)
    y_pre_train = model.predict(undersample_Xtest)
    y_pre_test = model.predict(X_test)
    # Print the accuracy
    accuracy_train_set.append(
        accuracy_score(undersample_ytest, y_pre_train, normalize=True
    ) * 100)
    accuracy_test_set.append(
        accuracy_score(y_test, y_pre_test, normalize=True) * 100)
# Confusion matrix
conf_matrix = confusion_matrix(undersample_ytest, y_pre_train)
conf_matrix_list_of_arrays.append(conf_matrix)
conf_matrix_2 = confusion_matrix(y_test, y_pre_test)
```

```

conf_matrix_list_of_arrays_test_set.append(conf_matrix_2)
# F1 score
f1_score_train.append(f1_score(undersample_ytest, y_pre_train))
f1_score_test.append(f1_score(y_test, y_pre_test))
# Thời gian kết thúc train
end_time = timer()
# Tính thời gian thực hiện
time_train = (end_time - start_time)
times_train.append(time_train)
# Độ chính xác của mô hình
print('Độ chính xác của mô hình trên dữ liệu train: ', accuracy_train_set)
print('Độ chính xác của mô hình trên dữ liệu test: ', accuracy_test_set)
print('Độ chính xác trung bình dữ liệu train:', np.mean(accuracy_train_set))
print('Độ chính xác trung bình dữ liệu test:', np.mean(accuracy_test_set))
print('F1 score trung bình trên dữ liệu train:', np.mean(f1_score_train))
print('F1 score trung bình trên dữ liệu test:', np.mean(f1_score_test))
print('Thời gian thực hiện: ', times_train)
# Confusion matrix: dữ liệu test
actual_cm_test_set = np.mean(conf_matrix_list_of_arrays_test_set, axis=0).astype(int)
labels = ['Not Fraud', 'Fraud']
plot_confusion_matrix(actual_cm_test_set, labels, title="Confusion Matrix \nTrên dữ liệu kiểm thử", cmap=plt.cm.Greens)

```

#### 4.2.11.2. Với bộ dữ liệu đã xử lý mất cân bằng

##### a. Đối với bộ dữ liệu xử lý cân bằng dùng phương pháp Oversampling

```

# Mô hình
model = XGBClassifier(objective="binary:logistic", random_state=42)
# Danh sách confusion matrix của từng lần KFold
conf_matrix_list_of_arrays = []
conf_matrix_list_of_arrays_test_set = []

```

```

# Danh sách độ chính xác của từng lần KFold trên dữ liệu test
accuracy_train_set = []
accuracy_test_set = []
f1_score_train = []
f1_score_test = []
# Cross Validation: KFold = 3
cv = StratifiedKFold(n_splits=3, random_state=None, shuffle=True)
cv.get_n_splits(X_train, y_train)
# List time train
times_train = []
for train_index, test_index in cv.split(X_train, y_train):
    undersample_Xtrain, undersample_Xtest = X_train.iloc[train_index]
    , X_train.iloc[test_index]
    undersample_ytrain, undersample_ytest = y_train.iloc[train_index]
    , y_train.iloc[test_index]
    # Thời gian bắt đầu train
    start_time = timer()
    model.fit(undersample_Xtrain, undersample_ytrain)
    y_pre_train = model.predict(undersample_Xtest)
    y_pre_test = model.predict(X_test)
    # Print the accuracy
    accuracy_train_set.append(
        accuracy_score(undersample_ytest, y_pre_train, normalize=True
    ) * 100)
    accuracy_test_set.append(
        accuracy_score(y_test, y_pre_test, normalize=True) * 100)
    # Confusion matrix
    conf_matrix = confusion_matrix(undersample_ytest, y_pre_train)
    conf_matrix_list_of_arrays.append(conf_matrix)
    conf_matrix_2 = confusion_matrix(y_test, y_pre_test)
    conf_matrix_list_of_arrays_test_set.append(conf_matrix_2)
    # F1 score
    f1_score_train.append(f1_score(undersample_ytest, y_pre_train))
    f1_score_test.append(f1_score(y_test, y_pre_test))
    # Thời gian kết thúc train
    end_time = timer()
    # Tính thời gian thực hiện

```

```

        time_train = (end_time - start_time)
        times_train.append(time_train)

# Độ chính xác của mô hình
print('Độ chính xác của mô hình trên dữ liệu train: ', accuracy_train_set)
print('Độ chính xác của mô hình trên dữ liệu test: ', accuracy_test_set)
print('Độ chính xác trung bình dữ liệu train:', np.mean(accuracy_train_set))
print('Độ chính xác trung bình dữ liệu test:', np.mean(accuracy_test_set))
print('F1 score trung bình trên dữ liệu train:', np.mean(f1_score_train))
print('F1 score trung bình trên dữ liệu test:', np.mean(f1_score_test))

print('Thời gian thực hiện: ', times_train)

```

### b. Đối với bộ dữ liệu xử lý cân bằng dùng phương pháp Undersampling

```

# Mô hình
model = XGBClassifier(objective="binary:logistic", random_state=42)

# Danh sách confusion matrix của từng lần KFold
conf_matrix_list_of_arrays = []
conf_matrix_list_of_arrays_test_set = []

# Danh sách độ chính xác của từng lần KFold trên dữ liệu test
accuracy_train_set = []
accuracy_test_set = []
f1_score_train = []
f1_score_test = []

# Cross Validation: KFold = 3
cv = StratifiedKFold(n_splits=3, random_state=None, shuffle=True)
cv.get_n_splits(X_train, y_train)

# List time train
times_train = []
for train_index, test_index in cv.split(X_train, y_train):
    undersample_Xtrain, undersample_Xtest = X_train.iloc[train_index],
    X_train.iloc[test_index]

```

```

undersample_ytrain, undersample_ytest = y_train.iloc[train_index]
, y_train.iloc[test_index]
# Thời gian bắt đầu train
start_time = timer()
model.fit(undersample_Xtrain, undersample_ytrain)
y_pre_train = model.predict(undersample_Xtest)
y_pre_test = model.predict(X_test)
# Print the accuracy
accuracy_train_set.append(
    accuracy_score(undersample_ytest, y_pre_train, normalize=True
)*100)
accuracy_test_set.append(
    accuracy_score(y_test, y_pre_test, normalize=True)*100)
# Confusion matrix
conf_matrix = confusion_matrix(undersample_ytest, y_pre_train)
conf_matrix_list_of_arrays.append(conf_matrix)
conf_matrix_2 = confusion_matrix(y_test, y_pre_test)
conf_matrix_list_of_arrays_test_set.append(conf_matrix_2)
# F1 score
f1_score_train.append(f1_score(undersample_ytest, y_pre_train))
f1_score_test.append(f1_score(y_test, y_pre_test))
# Thời gian kết thúc train
end_time = timer()
# Tính thời gian thực hiện
time_train = (end_time - start_time)
times_train.append(time_train)
# Độ chính xác của mô hình
print('Độ chính xác của mô hình trên dữ liệu train: ', accuracy_train_
_set)
print('Độ chính xác của mô hình trên dữ liệu test: ', accuracy_test_s
et)
print('Độ chính xác trung bình dữ liệu train:', np.mean(accuracy_trai
n_set))
print('Độ chính xác trung bình dữ liệu test:', np.mean(accuracy_test_
set))
print('F1 score trung bình trên dữ liệu train:', np.mean(f1_score_trai
n))

```

```

print('F1 score trung bình trên dữ liệu test:', np.mean(f1_score_test))
)
print('Thời gian thực hiện: ', times_train)

c. Đối với bộ dữ liệu xử lý cân bằng dùng phương pháp kết hợp SMOTE + ENN

# Mô hình

model = XGBClassifier(objective="binary:logistic", random_state=42)

# Danh sách confusion matrix của từng lần KFold
conf_matrix_list_of_arrays = []
conf_matrix_list_of_arrays_test_set = []

# Danh sách độ chính xác của từng lần KFold trên dữ liệu test
accuracy_train_set = []
accuracy_test_set = []
f1_score_train = []
f1_score_test = []

# Cross Validation: KFold = 3
cv = StratifiedKFold(n_splits=3, random_state=None, shuffle=True)
cv.get_n_splits(X_train, y_train)

# List time train
times_train = []
for train_index, test_index in cv.split(X_train, y_train):
    smote_enn_Xtrain, smote_enn_Xtest = X_train.iloc[train_index], X_train.iloc[test_index]
    smote_enn_ytrain, smote_enn_ytest = y_train.iloc[train_index], y_train.iloc[test_index]

    # Thời gian bắt đầu train
    start_time = timer()
    model.fit(smote_enn_Xtrain, smote_enn_ytrain)
    y_pre_train = model.predict(smote_enn_Xtest)
    y_pre_test = model.predict(X_test)

    # Print the accuracy
    accuracy_train_set.append(
        accuracy_score(smote_enn_ytest, y_pre_train, normalize=True) *
        100)

    accuracy_test_set.append(
        accuracy_score(y_test, y_pre_test, normalize=True) * 100)

# Confusion matrix

```

```

conf_matrix = confusion_matrix(smote_enn_ytest, y_pre_train)
conf_matrix_list_of_arrays.append(conf_matrix)
conf_matrix_2 = confusion_matrix(y_test, y_pre_test)
conf_matrix_list_of_arrays_test_set.append(conf_matrix_2)

# F1 score
f1_score_train.append(f1_score(smote_enn_ytest, y_pre_train))
f1_score_test.append(f1_score(y_test, y_pre_test))

# Thời gian kết thúc train
end_time = timer()

# Tính thời gian thực hiện
time_train = (end_time - start_time)
times_train.append(time_train)

# Độ chính xác của mô hình
print('Độ chính xác của mô hình trên dữ liệu train: ', accuracy_train_set)
print('Độ chính xác của mô hình trên dữ liệu test: ', accuracy_test_set)
print('Độ chính xác trung bình dữ liệu train:', np.mean(accuracy_train_set))
print('Độ chính xác trung bình dữ liệu test:', np.mean(accuracy_test_set))
print('F1 score trung bình trên dữ liệu train:', np.mean(f1_score_train))
print('F1 score trung bình trên dữ liệu test:', np.mean(f1_score_test))

print('Thời gian thực hiện: ', times_train)

```