

MỤC LỤC

CHƯƠNG I: TỔNG QUAN	4
1.1. Lý do chọn đề tài	4
1.2. Mục tiêu nghiên cứu	4
1.2.1. Mục tiêu tổng quát	4
1.2.2. Mục tiêu cụ thể	4
CHƯƠNG II: CƠ SỞ LÝ THUYẾT	6
2.1. Giới thiệu về bài toán phân loại	6
2.1.1. Khái niệm	6
2.1.2. Các loại bài toán phân loại	6
2.1.3. Quy trình giải quyết bài toán phân loại	6
2.1.4. Các thách thức trong bài toán phân loại	7
2.1.5. Ứng dụng của phân loại	7
2.2. Lý thuyết mô hình Random Forest	7
2.2.1. Giới thiệu về Random Forest	7
2.2.2. Cấu trúc và cách thức hoạt động của Random Forest	8
2.2.3. Ưu nhược điểm của Random Forest	9
2.2.4. Các tham số quan trọng của mô hình Random Forest	10
2.2.5. Ứng dụng của Random Forest	10
2.3. Chỉ số Kurtosis và Skew	10
2.3.1. Kurtosis (Độ nhọn)	10
2.3.2. Skew (Độ lệch)	11
2.3.3. Ý nghĩa và ứng dụng của Kurtosis và Skew	11
CHƯƠNG III: XÂY DỰNG MÔ HÌNH PHÁT HIỆN GIAN LẬN TÀI CHÍNH DỰA TRÊN HỌC MÁY CÓ GIÁM SÁT	13
3.1. Bài toán phát hiện gian lận tài chính	13
3.2. Thu thập dữ liệu và làm sạch	13
3.3. Exploratory Data Analysis (EDA) – Khai phá phân tích dữ liệu	14
3.4. Data preprocessing (Tiền xử lý dữ liệu)	20
3.4.1. Hiệu chỉnh chỉ số Kurtosis and skew của tập dữ liệu	20
3.4.2. Giảm kích thước của tập dữ liệu	21

3.4.3.	Chia tập dữ liệu Df_Balanced để huấn luyện và đánh giá	24
3.5.	Huấn luyện 3 mô hình (Random Forest, Logistic Regression và Gradient Boosting) và đánh giá hiệu suất.	25
3.6.	Tối ưu hóa tham số của mô hình Random Forest Classifier và đưa ra model cuối cùng.	28
3.7.	Xây dựng trang web phát hiện gian lận dựa trên mô hình đã được huấn luyện	31
CHƯƠNG IV: KẾT LUẬN.....		33
4.1.	Kết luận chung	33
4.2.	Hướng phát triển cho tương lai	33
TÀI LIỆU THAM KHẢO		35

DANH MỤC HÌNH ẢNH

Hình 1: Luồng hoạt động của học máy có giám sát	7
Hình 2: Mô hình Random Forest.....	9
Hình 3: Mô hình hoạt động của hệ thống.....	13
Hình 4: Số lượng giá trị duy nhất (unique values) cho từng cột trong tập dữ liệu	14
Hình 5: Phân phối của các giá trị trong các cột số liệu của tập dữ liệu.....	14
Hình 6: Đóng góp của các đặc trưng.....	15
Hình 7: Biểu đồ ma trận tương quan Pearson	16
Hình 8: Biểu đồ ma trận tương quan Spearman	17
Hình 9: Biểu đồ hộp theo step (thời gian giao dịch)	18
Hình 10: Chỉ số Kurtosis and skew của tập dữ liệu	19
Hình 11: Hiệu chỉnh chỉ số Kurtosis and skew của tập dữ liệu	21
Hình 12: Kích thước của tập dữ liệu	22
Hình 13: Giảm kích thước của tập dữ liệu	22
Hình 14: Biểu đồ đóng góp của các đặc trưng sau khi điều chỉnh dữ liệu.....	23
Hình 15: Chỉ số Kurtosis and skew của tập dữ liệu cuối cùng.....	24
Hình 16: Chia tập dữ liệu Df_Balanced để huấn luyện và đánh giá.....	24
Hình 17: Huấn luyện 3 mô hình	26
Hình 18: Chỉ số đánh giá của 3 mô hình.....	26
Hình 19: Biểu đồ chỉ số đánh giá của 3 mô hình.....	27
Hình 20: Tối ưu hóa tham số của mô hình Random Forest Classifier	28
Hình 21: Huấn luyện mô hình cuối cùng	29
Hình 22: Chỉ số đánh giá của mô hình cuối cùng	29
Hình 23: Biểu đồ chỉ số đánh giá của mô hình cuối cùng	30
Hình 24: Lưu trữ mô hình	31
Hình 25: Giao diện trang web.....	32

CHƯƠNG I: TỔNG QUAN

1.1. Lý do chọn đề tài

- **Tính cấp thiết của vấn đề:** Hiện nay, gian lận (fraud) là một vấn đề nghiêm trọng trong nhiều lĩnh vực, đặc biệt là tài chính, thương mại điện tử và bảo hiểm. Gian lận không chỉ gây tổn thất kinh tế lớn cho doanh nghiệp mà còn ảnh hưởng đến lòng tin của khách hàng. Do đó, việc phát hiện và ngăn chặn gian lận là một yêu cầu cấp thiết.
- **Ứng dụng thực tiễn cao:** Các mô hình phát hiện gian lận sử dụng khoa học dữ liệu và trí tuệ nhân tạo (AI) đã chứng minh hiệu quả trong việc xử lý dữ liệu lớn và phức tạp. Kỹ thuật này không chỉ tự động hóa việc phát hiện mà còn giảm thiểu sai sót so với các phương pháp truyền thống.
- **Kết hợp kiến thức liên ngành:** Đề tài yêu cầu kết hợp các kiến thức về khoa học dữ liệu, trí tuệ nhân tạo, xử lý dữ liệu lớn và lĩnh vực kinh doanh cụ thể (tài chính, thương mại, bảo hiểm,...). Điều này mang lại cơ hội để mở rộng và áp dụng các kỹ năng đã học vào các tình huống thực tế.
- **Học tập và phát triển kỹ năng chuyên sâu:** Làm việc với dữ liệu gian lận thường gặp nhiều thách thức như dữ liệu mất cân đối, yêu cầu xử lý dữ liệu lớn, và tối ưu hóa mô hình. Việc giải quyết những vấn đề này sẽ giúp phát triển các kỹ năng chuyên sâu về xử lý dữ liệu và xây dựng mô hình AI/ML.
- **Tính đổi mới và tiềm năng nghiên cứu:** Lĩnh vực phát hiện gian lận luôn đổi mới, với các thuật toán và phương pháp tiên tiến không ngừng được phát triển. Đề tài này không chỉ có giá trị nghiên cứu mà còn có thể mở rộng để áp dụng trong các lĩnh vực khác.

1.2. Mục tiêu nghiên cứu

1.2.1. Mục tiêu tổng quát

- Xây dựng và đánh giá hiệu quả của các phương pháp khoa học dữ liệu trong việc phát hiện gian lận, từ đó đề xuất một mô hình tối ưu giúp cải thiện độ chính xác và hiệu suất trong việc nhận diện các hành vi gian lận trên dữ liệu thực tế.

1.2.2. Mục tiêu cụ thể

- **Tìm hiểu và phân tích vấn đề gian lận:** Nghiên cứu các loại gian lận phổ biến trong lĩnh vực được chọn (ví dụ: tài chính, thương mại điện tử, bảo hiểm) và cách chúng được thực hiện. Từ đó xác định các đặc điểm dữ liệu và dấu hiệu nhận biết.
- **Thu thập và xử lý dữ liệu:** Thu thập một tập dữ liệu phù hợp, thực hiện các bước tiền xử lý như làm sạch dữ liệu, xử lý mất cân đối dữ liệu, và tạo các đặc trưng liên quan để phục vụ cho việc xây dựng mô hình.
- **Áp dụng các kỹ thuật khoa học dữ liệu:** Thử nghiệm các mô hình học máy (Machine Learning) hoặc học sâu (Deep Learning) phổ biến, bao gồm các thuật toán như Logistic Regression, Random Forest, Gradient Boosting, Xgboost hoặc Neural Networks để phát hiện gian lận.

- **Đánh giá hiệu quả mô hình:** Sử dụng các chỉ số đánh giá hiệu quả như độ chính xác (Accuracy), độ nhạy (Recall), Precision, F1-score và RocAuc để so sánh các mô hình. Tập trung vào việc giảm thiểu các trường hợp báo động sai (False Positives) và bỏ sót gian lận (False Negatives).
- **Đề xuất giải pháp tối ưu:** Dựa trên kết quả đánh giá, đề xuất một giải pháp hoặc mô hình có thể ứng dụng hiệu quả trong thực tế, đồng thời đảm bảo tính khả thi và khả năng triển khai.
- **Phân tích ý nghĩa và ứng dụng:** Đưa ra các khuyến nghị về cách tích hợp mô hình phát hiện gian lận vào hệ thống hiện tại, cùng với đánh giá tác động kinh tế và xã hội của giải pháp này.

Kết quả mong đợi:

- Xây dựng được một mô hình phát hiện gian lận chính xác và hiệu quả.
- Đóng góp vào việc phát triển các phương pháp khoa học dữ liệu trong lĩnh vực nhận diện gian lận.
- Tạo nền tảng cho các nghiên cứu sâu hơn hoặc ứng dụng thực tế trong tương lai.

CHƯƠNG II: CƠ SỞ LÝ THUYẾT

2.1. Giới thiệu về bài toán phân loại

2.1.1. Khái niệm

- Phân loại là một bài toán trong học máy (Machine Learning) với mục tiêu phân nhóm hoặc phân loại các đối tượng vào các lớp (class) cụ thể dựa trên các đặc trưng (features) của chúng. Mỗi đối tượng trong dữ liệu đầu vào sẽ được gán nhãn (label) thuộc một lớp trong số các lớp đã xác định trước.
- Ví dụ, trong bài toán phân loại email, các email có thể được phân loại thành các lớp như "spam" và "not spam". Một ví dụ khác là phân loại bệnh nhân vào các nhóm "bị bệnh" hoặc "không bị bệnh" dựa trên các chỉ số y tế.

2.1.2. Các loại bài toán phân loại

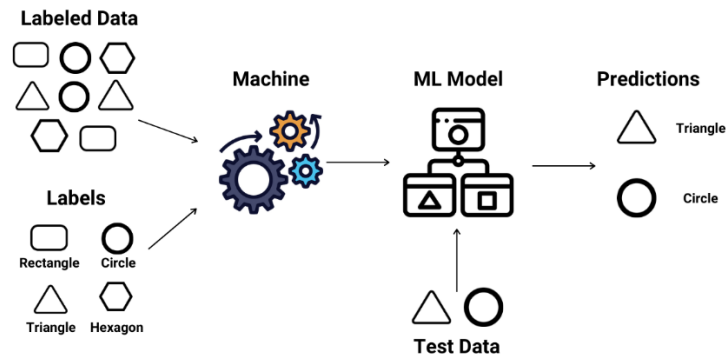
- **Phân loại nhị phân (Binary Classification):** Bài toán phân loại có hai lớp. Ví dụ, phân loại email thành spam hoặc không spam.
- **Phân loại đa lớp (Multiclass Classification):** Bài toán phân loại có hơn hai lớp. Ví dụ, phân loại các loại trái cây thành các nhóm như táo, chuối, cam.
- **Phân loại đa nhãn (Multilabel Classification):** Mỗi đối tượng có thể thuộc nhiều lớp cùng lúc. Ví dụ, phân loại một bài viết thành nhiều chủ đề như "khoa học", "công nghệ", và "giáo dục".

2.1.3. Quy trình giải quyết bài toán phân loại

Quy trình để giải quyết bài toán phân loại bao gồm các bước sau:

- **Thu thập dữ liệu:** Thu thập và chuẩn bị dữ liệu đầu vào, đảm bảo rằng dữ liệu có đủ thông tin để phân loại.
- **Tiền xử lý dữ liệu:** Làm sạch dữ liệu, xử lý thiếu sót, loại bỏ nhiễu và chuyển đổi dữ liệu vào dạng có thể sử dụng cho mô hình học máy.
- **Chọn mô hình phân loại:** Lựa chọn một hoặc nhiều mô hình phân loại phù hợp để huấn luyện.
- **Huấn luyện mô hình:** Sử dụng dữ liệu đã được phân chia thành các bộ huấn luyện và kiểm tra để huấn luyện mô hình.
- **Đánh giá mô hình:** Đánh giá hiệu quả của mô hình bằng các chỉ số như độ chính xác, độ nhạy, độ đặc hiệu và F1-score.

Supervised Learning



Hình 1: Luồng hoạt động của học máy có giám sát

2.1.4. Các thách thức trong bài toán phân loại

- **Dữ liệu mất cân đối:** Khi các lớp trong dữ liệu không phân bố đều, các mô hình phân loại có thể bị thiên lệch và không phản ánh đúng thực tế. Các kỹ thuật như cân bằng lớp (balancing) hoặc sử dụng trọng số cho các lớp có thể giúp cải thiện mô hình.
- **Overfitting và Underfitting:** Mô hình có thể học quá chi tiết từ dữ liệu huấn luyện (overfitting) hoặc không học đủ thông tin (underfitting). Việc lựa chọn mô hình phù hợp và sử dụng các kỹ thuật như cross-validation là cần thiết để tránh hai vấn đề này.
- **Dữ liệu phức tạp:** Khi dữ liệu có nhiều đặc trưng, mối quan hệ giữa các đặc trưng có thể rất phức tạp, đòi hỏi các mô hình phức tạp hơn để xử lý tốt.

2.1.5. Ứng dụng của phân loại

Phân loại được ứng dụng rộng rãi trong nhiều lĩnh vực:

- **Chẩn đoán y tế:** Phân loại bệnh nhân thành các nhóm bệnh khác nhau dựa trên các chỉ số y tế.
- **Phân loại khách hàng:** Xác định phân khúc khách hàng trong marketing dựa trên hành vi mua sắm.
- **Phát hiện gian lận (Fraud Detection):** Phân loại các giao dịch hợp lệ và gian lận trong ngành tài chính và thương mại điện tử.

2.2. Lý thuyết mô hình Random Forest

2.2.1. Giới thiệu về Random Forest

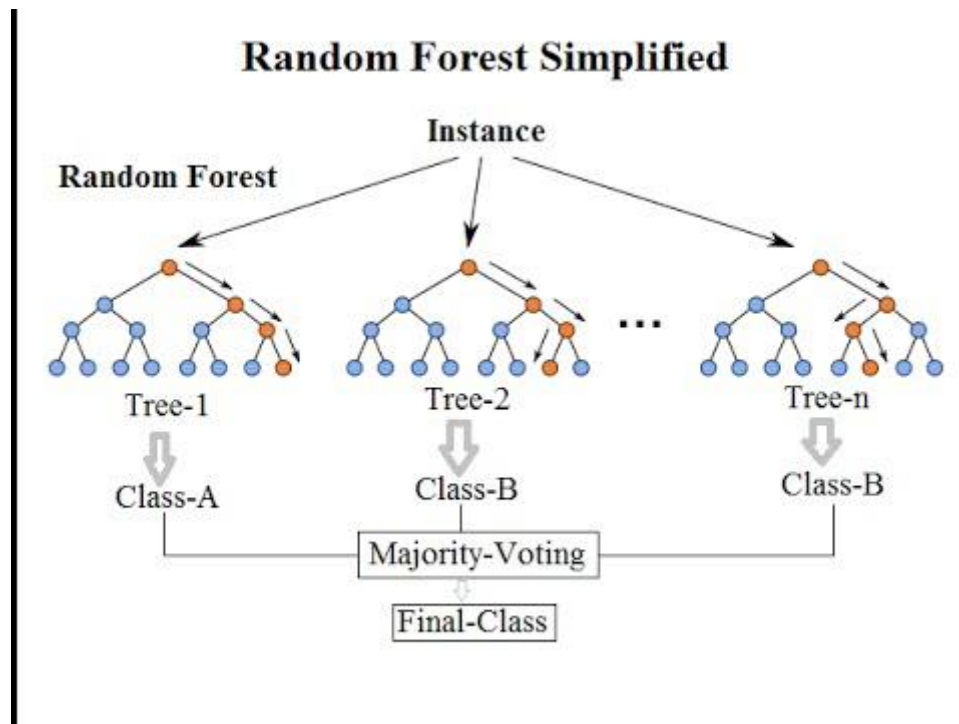
- Random Forest là một thuật toán học máy mạnh mẽ trong nhóm các phương pháp ensemble learning (học máy tập hợp), được sử dụng chủ yếu cho các bài toán phân loại và hồi quy. Nó được xây dựng từ một tập hợp các mô hình cây quyết định (Decision Trees) và sử dụng sự kết hợp của chúng để đưa ra dự đoán chính xác hơn so với một cây quyết định đơn lẻ.

- Mô hình Random Forest hoạt động dựa trên nguyên lý "bootstrap aggregating" (hay còn gọi là **bagging**), trong đó nhiều mô hình được huấn luyện và kết hợp lại để cải thiện hiệu suất và giảm thiểu sai số.

2.2.2. Cấu trúc và cách thức hoạt động của Random Forest

Random Forest là một tập hợp (ensemble) của nhiều cây quyết định, mỗi cây quyết định trong rừng này được huấn luyện trên một phần mẫu ngẫu nhiên khác nhau của dữ liệu huấn luyện. Sau khi tất cả các cây được huấn luyện, kết quả dự đoán của rừng sẽ là kết quả của việc kết hợp dự đoán của từng cây. Cụ thể:

- **Bootstrap Sampling:** Đầu tiên, Random Forest sẽ tạo ra nhiều bộ dữ liệu con bằng cách chọn ngẫu nhiên các điểm dữ liệu từ tập huấn luyện với sự thay thế (sampling with replacement). Mỗi bộ dữ liệu con này sẽ được sử dụng để huấn luyện một cây quyết định.
- **Ngẫu nhiên hóa các đặc trưng:** Khi mỗi cây quyết định được xây dựng, không phải tất cả các đặc trưng (features) trong dữ liệu đều được xem xét. Thay vào đó, tại mỗi nút phân chia trong cây quyết định, Random Forest sẽ chọn một tập con ngẫu nhiên của các đặc trưng để phân chia. Điều này giúp tăng tính đa dạng giữa các cây và giảm thiểu khả năng cây bị overfitting.
- **Kết hợp kết quả của các cây:** Sau khi các cây quyết định được huấn luyện, kết quả của Random Forest được đưa ra bằng cách kết hợp các dự đoán của từng cây. Đối với bài toán phân loại, kết quả sẽ là "quyết định phổ biến nhất" (majority vote) từ tất cả các cây. Đối với bài toán hồi quy, kết quả sẽ là trung bình của các giá trị dự đoán từ các cây.



Hình 2: Mô hình Random Forest

2.2.3. Ưu nhược điểm của Random Forest

- Ưu điểm:

- **Giảm thiểu overfitting:** Vì Random Forest sử dụng nhiều cây quyết định và kết hợp các dự đoán của chúng, nó có thể giảm thiểu overfitting, một vấn đề thường gặp trong cây quyết định đơn lẻ. Khi nhiều cây cùng đưa ra quyết định, ảnh hưởng của một cây bị overfitting sẽ bị giảm đi.
- **Khả năng xử lý dữ liệu lớn và đa dạng:** Random Forest có thể xử lý một lượng lớn dữ liệu với nhiều đặc trưng mà không cần quá nhiều công đoạn xử lý dữ liệu phức tạp.
- **Khả năng phân loại tốt ngay cả với dữ liệu thiếu:** Mặc dù Random Forest không yêu cầu dữ liệu phải hoàn chỉnh, nó vẫn có thể hoạt động hiệu quả khi gặp dữ liệu thiếu.
- **Độ chính xác cao:** Vì sử dụng nhiều cây và các phương pháp ngẫu nhiên hóa, Random Forest có thể cho kết quả chính xác cao trong hầu hết các trường hợp.
- **Khả năng phát hiện các đặc trưng quan trọng:** Random Forest có thể đánh giá được mức độ quan trọng của từng đặc trưng trong việc phân loại hoặc dự đoán.

- Nhược điểm:

- **Khó giải thích:** Mặc dù Random Forest rất hiệu quả trong việc đưa ra dự đoán, nhưng do mô hình này bao gồm nhiều cây quyết định, nó trở nên

khó giải thích và khó hiểu đối với người dùng, đặc biệt là khi so với một cây quyết định đơn giản.

- **Chi phí tính toán cao:** Việc huấn luyện nhiều cây quyết định có thể tốn nhiều tài nguyên tính toán, đặc biệt khi dữ liệu lớn và số lượng cây trong rừng tăng lên.
- **Kích thước mô hình lớn:** Một rừng quyết định có thể tạo ra một mô hình rất lớn, điều này có thể dẫn đến khó khăn trong việc lưu trữ và triển khai nếu bộ dữ liệu hoặc số cây quá lớn.

2.2.4. Các tham số quan trọng của mô hình Random Forest

- **Số lượng cây ($n_estimators$):** Số lượng cây quyết định trong rừng. Số lượng này có ảnh hưởng lớn đến độ chính xác và tốc độ huấn luyện của mô hình. Thông thường, số cây càng nhiều thì mô hình càng chính xác, nhưng lại tốn nhiều tài nguyên tính toán hơn.
- **Số lượng đặc trưng tại mỗi phân chia ($max_features$):** Đây là số lượng đặc trưng tối đa mà mỗi cây sẽ xem xét tại mỗi nút phân chia. Việc giảm số lượng đặc trưng sẽ làm tăng tính đa dạng của các cây nhưng cũng có thể làm giảm độ chính xác của mỗi cây.
- **Độ sâu tối đa của cây (max_depth):** Giới hạn độ sâu của cây quyết định. Cây quá sâu có thể dễ dàng bị overfitting, trong khi cây quá nông có thể thiếu khả năng phân chia dữ liệu một cách chính xác.
- **Số mẫu tối thiểu tại một nút phân chia ($min_samples_split$):** Đây là số mẫu tối thiểu cần thiết để tiếp tục phân chia một nút. Nếu giá trị này quá cao, cây sẽ bị hạn chế trong việc phân chia dữ liệu, trong khi nếu quá thấp, cây có thể dễ dàng bị overfitting.
- **Số mẫu tối thiểu tại một nút lá ($min_samples_leaf$):** Số mẫu tối thiểu tại mỗi nút lá (đầu ra) của cây quyết định. Điều này giúp tránh tình trạng cây quá phức tạp và dễ bị overfitting.

2.2.5. Ứng dụng của Random Forest

Random Forest có thể được áp dụng trong nhiều bài toán khác nhau, bao gồm:

- **Phân loại:** Phân loại bệnh nhân vào các nhóm bệnh, phân loại email là spam hay không, phân loại khách hàng trong marketing.
- **Hồi quy:** Dự đoán giá trị bất động sản, dự đoán số lượng bán hàng, dự đoán giá trị cổ phiếu.
- **Phát hiện bất thường (Anomaly Detection):** Phát hiện gian lận trong các giao dịch tài chính, phát hiện hành vi bất thường trong mạng máy tính.

2.3. Chỉ số Kurtosis và Skew

2.3.1. Kurtosis (Độ nhọn)

- Kurtosis là một chỉ số trong thống kê đo lường độ "nhọn" của phân phối xác suất, tức là mức độ tập trung của các giá trị dữ liệu quanh trung bình. Chỉ số này cho

biết sự phân bố của các điểm dữ liệu trong một phân phối so với một phân phối chuẩn (normal distribution).

- **Kurtosis cao** ($Kurtosis > 3$): Phân phối có "đỉnh" cao và "đuôi" dày, tức là có nhiều giá trị cực đoan (outliers) và các giá trị gần trung bình hơn so với phân phối chuẩn. Các phân phối có kurtosis cao được gọi là **leptokurtic**.
- **Kurtosis thấp** ($Kurtosis < 3$): Phân phối có "đỉnh" thấp và "đuôi" mỏng, tức là dữ liệu phân tán đều hơn và ít có các giá trị ngoại lệ. Các phân phối có kurtosis thấp được gọi là **platykurtic**.
- **Kurtosis bằng 3** ($Kurtosis = 3$): Phân phối có dạng chuẩn (normal distribution). Một phân phối chuẩn có kurtosis bằng 3 và được gọi là **mesokurtic**.

2.3.2. Skew (Độ lệch)

- Skewness (độ lệch) là một chỉ số đo lường sự đối xứng của phân phối dữ liệu. Nếu phân phối dữ liệu lệch về phía bên trái hoặc bên phải của trung bình, chỉ số skew sẽ phản ánh điều đó.
 - **Skew dương (Right skew)**: Khi phân phối có độ lệch dương, đuôi bên phải của phân phối dài hơn, tức là phần lớn dữ liệu tập trung về phía bên trái của phân phối. Các giá trị lớn xuất hiện ít hơn nhưng có ảnh hưởng mạnh mẽ đến phân phối. Khi đó, chỉ số skew sẽ dương (> 0).
 - **Skew âm (Left skew)**: Khi phân phối có độ lệch âm, đuôi bên trái của phân phối dài hơn, tức là phần lớn dữ liệu tập trung về phía bên phải của phân phối. Các giá trị nhỏ xuất hiện ít hơn nhưng lại kéo phân phối về phía trái. Khi đó, chỉ số skew sẽ âm (< 0).
 - **Skew bằng 0**: Khi phân phối có độ lệch bằng 0, tức là phân phối là đối xứng xung quanh giá trị trung bình. Đây là đặc điểm của một phân phối chuẩn.

2.3.3. Ý nghĩa và ứng dụng của Kurtosis và Skew

- **Kurtosis**:
 - **Đánh giá sự phân bố dữ liệu**: Kurtosis giúp hiểu rõ hơn về cách thức dữ liệu phân bố, liệu có nhiều điểm ngoại lệ hay không.
 - **Phát hiện phân phối bất thường**: Một kurtosis cao có thể chỉ ra rằng có quá nhiều dữ liệu xa trung bình, điều này có thể là dấu hiệu của phân phối bất thường hoặc sự xuất hiện của outliers.
- **Skew**:
 - **Đánh giá đối xứng của phân phối**: Skewness cung cấp thông tin về việc phân phối có đối xứng hay không, và có thể chỉ ra các vấn đề trong dữ liệu như sự xuất hiện của các giá trị ngoại lệ (outliers).

- **Điều chỉnh cho mô hình dự báo:** Nếu dữ liệu lệch, các mô hình thống kê như hồi quy có thể gặp khó khăn trong việc đưa ra dự đoán chính xác. Thường xuyên điều chỉnh dữ liệu lệch thông qua phép biến đổi như log transformation có thể cải thiện kết quả mô hình.

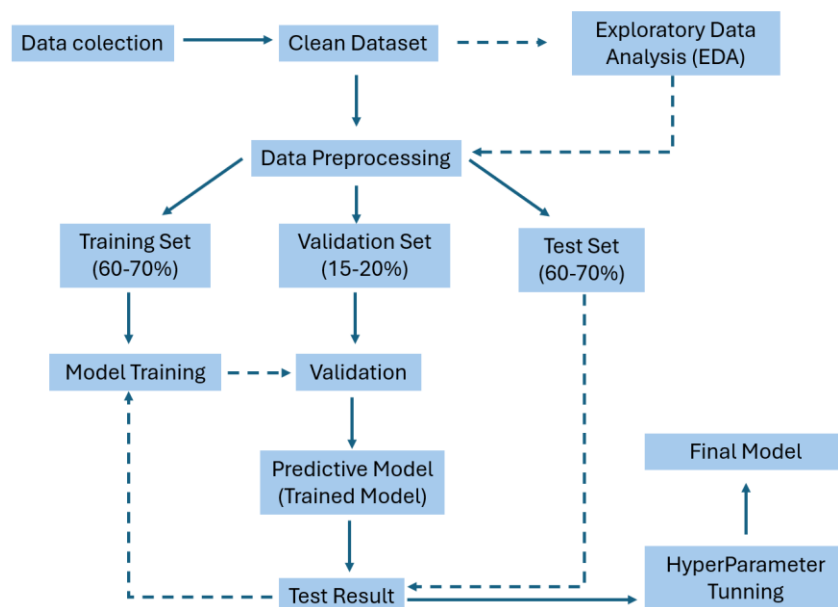
- **Ví dụ:**

- **Phân phối chuẩn:** Một phân phối chuẩn có kurtosis = 3 và skewness = 0, nghĩa là phân phối đối xứng và có đỉnh vừa phải.
- **Phân phối với độ lệch dương (skew dương):** Một phân phối có các giá trị nhỏ chiếm đa số, nhưng lại có một số giá trị lớn kéo dài đuôi phân phối về phía phải. Đây là phân phối với skew > 0, và có thể có kurtosis cao nếu có nhiều giá trị ngoại lệ.
- **Phân phối với độ lệch âm (skew âm):** Một phân phối có các giá trị lớn chiếm đa số và có một số giá trị nhỏ kéo dài đuôi phân phối về phía trái. Đây là phân phối với skew < 0, và có thể có kurtosis thấp nếu dữ liệu ít có giá trị ngoại lệ.

CHƯƠNG III: XÂY DỰNG MÔ HÌNH PHÁT HIỆN GIAN LẬN TÀI CHÍNH DỰA TRÊN HỌC MÁY CÓ GIÁM SÁT

3.1. Bài toán phát hiện gian lận tài chính

- Việc phát triển mô hình tiến hành như sau. Trước tiên, dữ liệu đầu vào biểu diễn dưới dạng các đặc trưng hay các thuộc tính. Tiếp theo, trong giai đoạn huấn luyện, tập dữ liệu đã được gán nhãn $\{0,1\}$ - gọi là dữ liệu huấn luyện hay dữ liệu mẫu - được sử dụng để huấn luyện một bộ phân loại. Sau khi huấn luyện xong, bộ phân loại được sử dụng để xác định giao dịch mới (giao dịch chưa biết nhãn) thuộc vào loại nào trong hai loại nói trên. Trong cả giai đoạn huấn luyện và phân loại, thuật toán phân loại chỉ làm việc với dữ liệu đã được biểu diễn dưới dạng các đặc trưng. Kết quả đầu ra của quá trình huấn luyện này là hai lớp: isFraud(gian lận), notFraud(không gian lận).
- Mô hình phát hiện gian lận có thể mô tả như hình sau:



Hình 3: Mô hình hoạt động của hệ thống

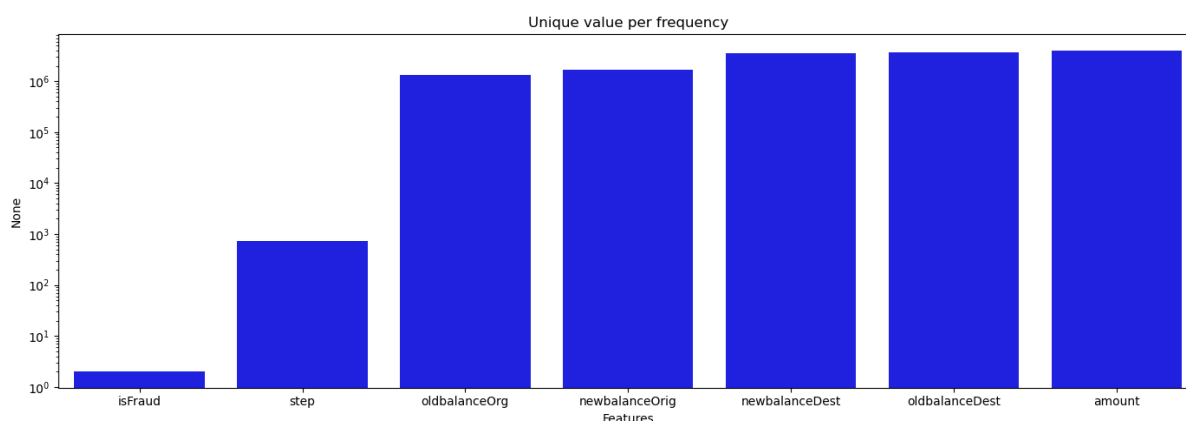
3.2. Thu thập dữ liệu và làm sạch

- Dữ liệu dùng để huấn luyện mô hình và kiểm thử mô hình được lấy từ tập dữ liệu lại kaggle, tại địa chỉ [dataset](#). Dữ liệu gồm 11 cột và 6362620 hàng.
- Các cột của dữ liệu bao gồm:
 - **step**: Thời gian giao dịch được ghi nhận, có thể là một đơn vị thời gian như giờ hoặc ngày.
 - **type**: Loại giao dịch
 - **amount**: Số tiền trong giao dịch.
 - **nameOrig**: ID của tài khoản thực hiện giao dịch.

- **oldbalanceOrig**: Số dư của tài khoản gửi trước giao dịch.
- **newbalanceOrig**: Số dư của tài khoản gửi sau giao dịch.
- **nameDest**: ID của tài khoản nhận giao dịch.
- **oldbalanceDest**: Số dư của tài khoản nhận trước giao dịch.
- **newbalanceDest**: Số dư của tài khoản nhận sau giao dịch.
- **isFraud**: Biến nhị phân (0 hoặc 1) biểu thị liệu giao dịch có phải là gian lận hay không.

3.3. Exploratory Data Analysis (EDA) – Khai phá phân tích dữ liệu

Số lượng giá trị duy nhất (unique values) cho từng cột trong tập dữ liệu:

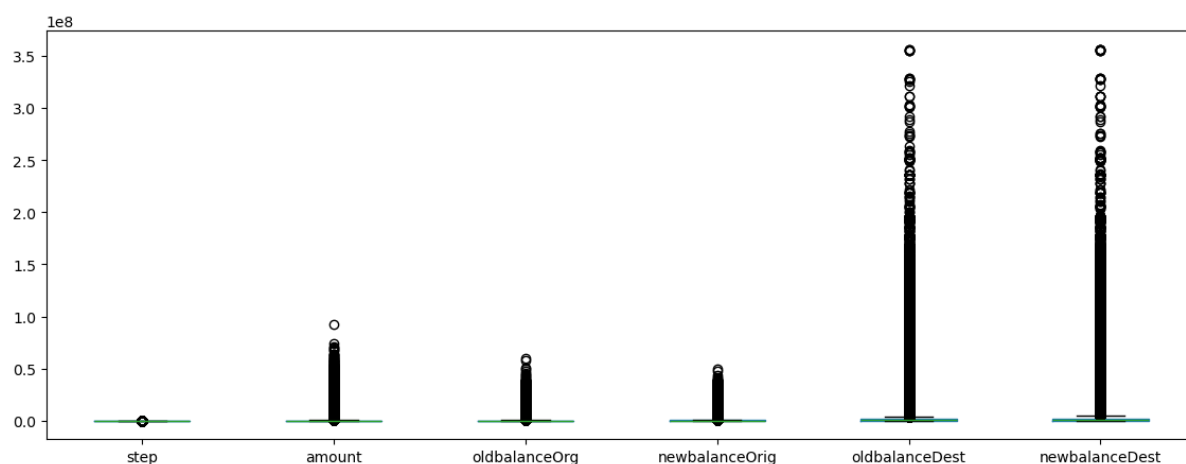


Hình 4: Số lượng giá trị duy nhất (unique values) cho từng cột trong tập dữ liệu

- Nhận xét:

- Dữ liệu của các cột như amount và các số dư có tính đa dạng cao, điều này rất hữu ích cho việc phân tích hoặc xây dựng mô hình.
- Cột isFraud có ít giá trị duy nhất, điều này có thể gây mất cân bằng dữ liệu nếu tỷ lệ giữa các giao dịch hợp lệ và gian lận chênh lệch lớn.

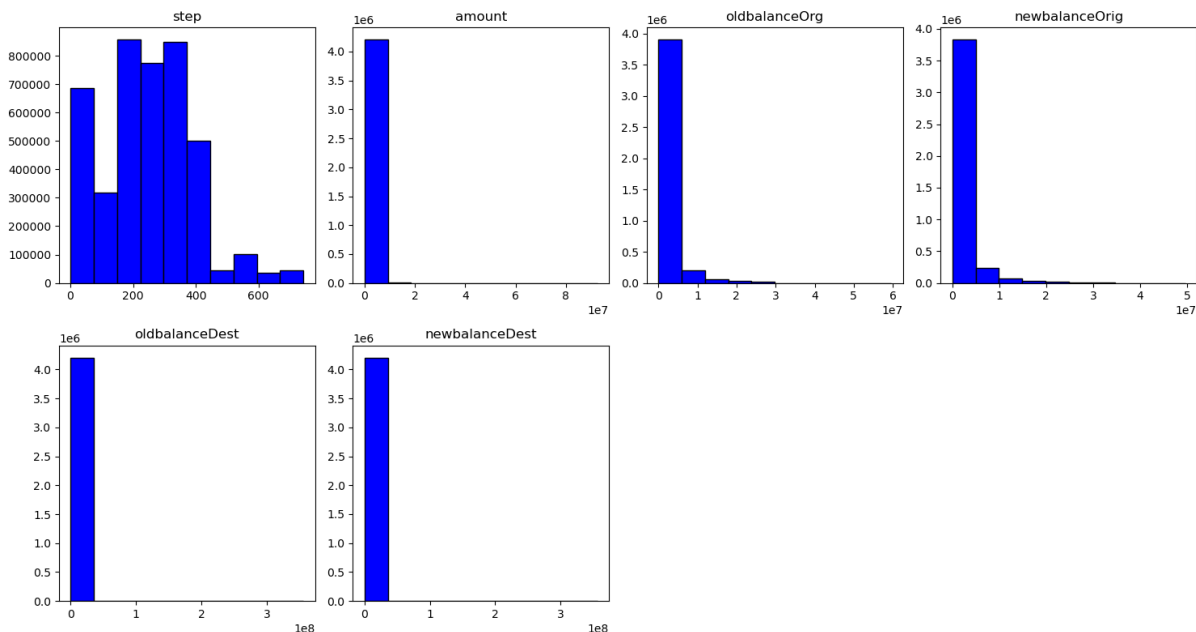
Phân phối của các giá trị trong các cột số liệu của tập dữ liệu:



Hình 5: Phân phối của các giá trị trong các cột số liệu của tập dữ liệu

- Nhận xét:
 - Cột `amount`, `oldbalanceOrg`, `newbalanceOrig`, `oldbalanceDest`, `newbalanceDest`:
 - Có rất nhiều giá trị bất thường (outliers), được thể hiện bởi các điểm tròn nằm xa râu của biểu đồ.
 - Phân phối các giá trị bị lệch mạnh, với phần lớn dữ liệu tập trung ở các giá trị nhỏ, trong khi một số ít giao dịch có giá trị cực lớn.
 - Cột `step`: Phân phối của cột này khác biệt, không có outliers rõ ràng, và các giá trị dường như khá đồng đều.
 - Cột `amount` và các số dư: Sự xuất hiện nhiều giá trị lớn bất thường có thể đại diện cho các giao dịch lớn, một số trong đó có thể liên quan đến gian lận.

Đóng góp của các đặc trưng:

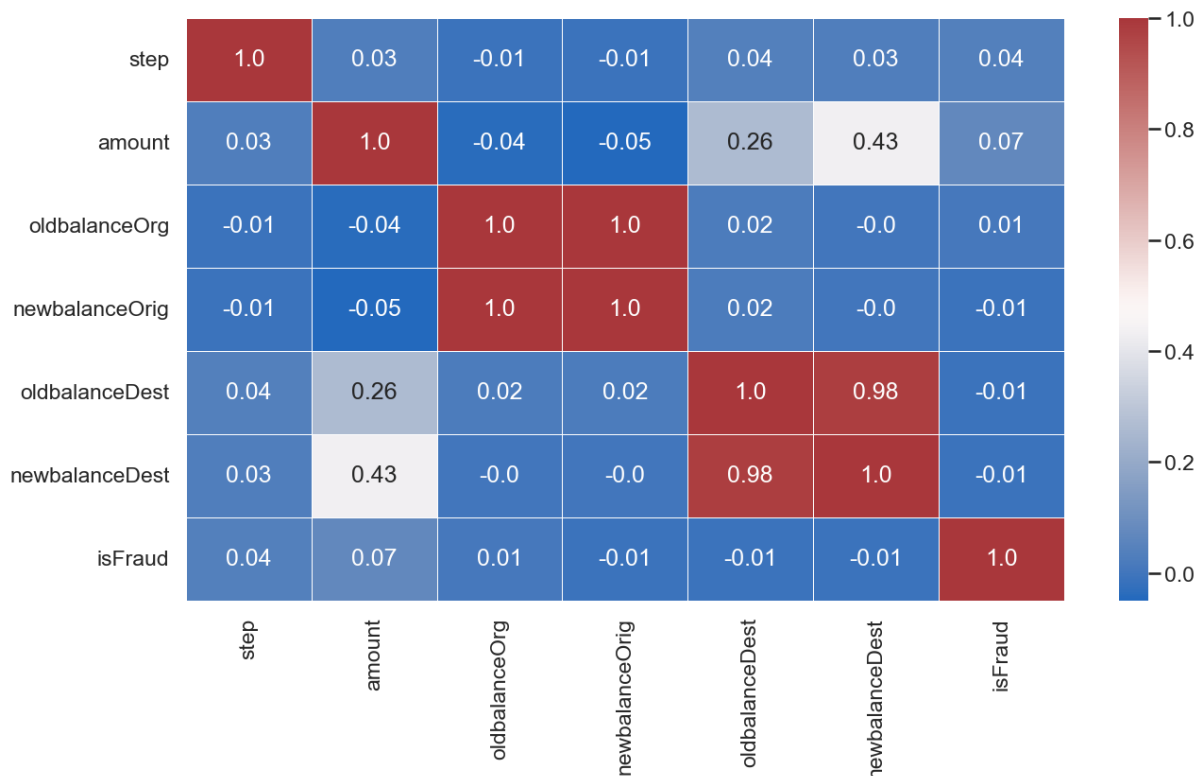


Hình 6: Đóng góp của các đặc trưng

- Nhận xét:
 - **step**: Phân phối dường như khá đồng đều ở một số khoảng giá trị thấp hơn, với lượng giao dịch lớn ở các khoảng thấp hơn (cụ thể là dưới 400). Số lượng giao dịch giảm dần ở các giá trị cao hơn.
 - **amount**: Phân phối rất lệch phải, với đa số giá trị ở mức thấp. Có một số giá trị cực đại lớn (có thể là giao dịch với số tiền lớn)
 - **oldbalanceOrg** và **newbalanceOrg**: Phân phối lệch phải, cho thấy phần lớn giá trị số dư gốc ban đầu và sau giao dịch của tài khoản gốc nhỏ, nhưng vẫn có một số giao dịch có giá trị rất lớn.

- **oldbalanceDest và newbalanceDest:** Cả hai biến này có một số lượng lớn giá trị bằng hoặc gần 0, nhưng cũng xuất hiện các giá trị rất lớn nằm ở đuôi bên phải.
- Các biểu đồ như amount, oldbalanceOrg, và newbalanceOrg cho thấy sự tập trung mạnh mẽ ở giá trị thấp, nhưng lại có một số giao dịch với giá trị lớn bất thường (outlier) cho thấy dữ liệu bị mất cân đối.

Biểu đồ ma trận tương quan Pearson:

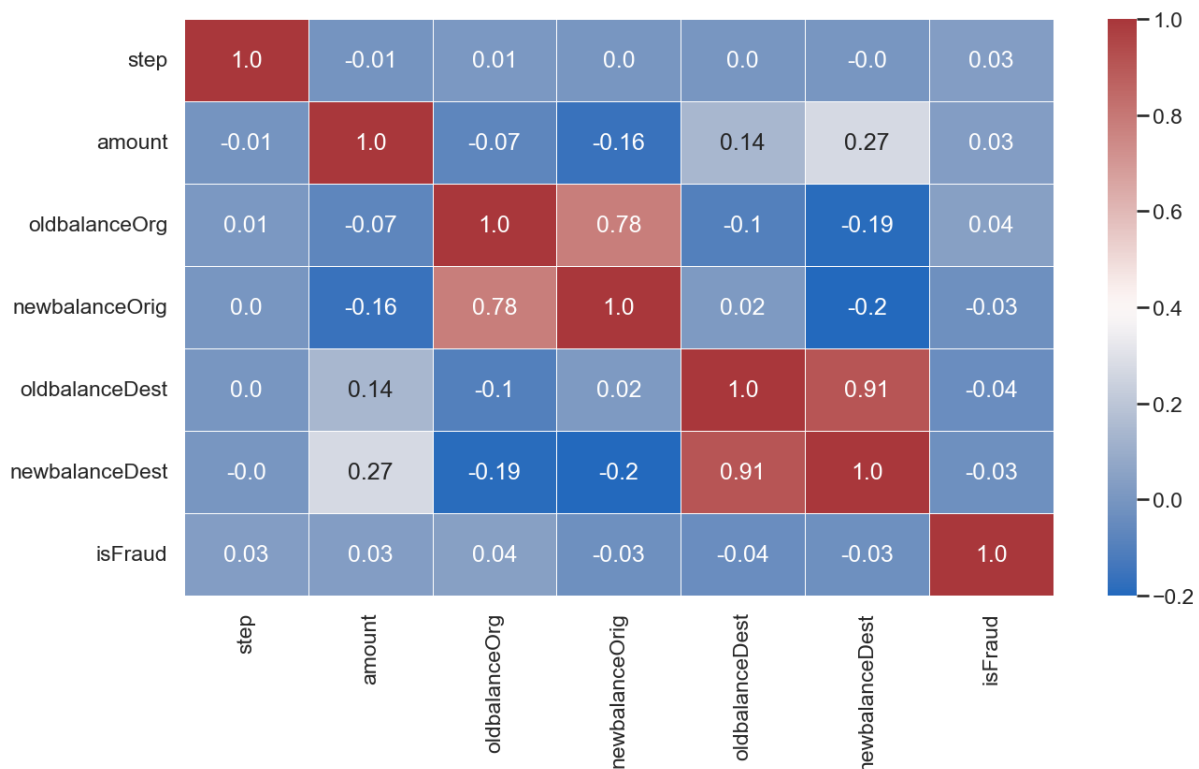


Hình 7: Biểu đồ ma trận tương quan Pearson

- Nhận xét:
 - **oldbalanceOrg và newbalanceOrg (0.98):** Có mối tương quan rất cao (gần như tuyệt đối). Điều này cho thấy sự phụ thuộc mạnh giữa số dư tài khoản trước và sau giao dịch của tài khoản nguồn (Origin).
 - **oldbalanceDest và newbalanceDest (0.98):** Tương tự, mối quan hệ giữa số dư tài khoản trước và sau giao dịch của tài khoản đích (Destination) cũng rất chặt chẽ.
 - **amount và oldbalanceDest (0.26):** Có một mức tương quan vừa phải. Điều này có thể lý giải rằng giao dịch lớn hơn (amount lớn) có khả năng làm tăng đáng kể số dư tại tài khoản đích.
 - **amount và newbalanceDest (0.43):** Tương quan mạnh hơn một chút so với oldbalanceDest. Điều này là do giao dịch lớn (amount) thường tạo ra thay đổi rõ rệt trên số dư cuối cùng của tài khoản đích.

- **isFraud** (nhãn gian lận) hầu như không có tương quan đáng kể với bất kỳ biến liên tục nào trong dữ liệu. Điều này gợi ý rằng các đặc điểm như số dư ban đầu, số dư sau giao dịch, hoặc giá trị giao dịch có thể không đủ để trực tiếp phân biệt giao dịch gian lận.
- **step** (đại diện cho thời gian hoặc khoảng cách giữa các giao dịch) có tương quan rất yếu với các biến khác (0.03 đến 0.04), cho thấy thời gian giao dịch không ảnh hưởng đáng kể đến các yếu tố như số dư tài khoản hoặc giá trị giao dịch.

Biểu đồ ma trận tương quan Spearman:



Hình 8: Biểu đồ ma trận tương quan Spearman

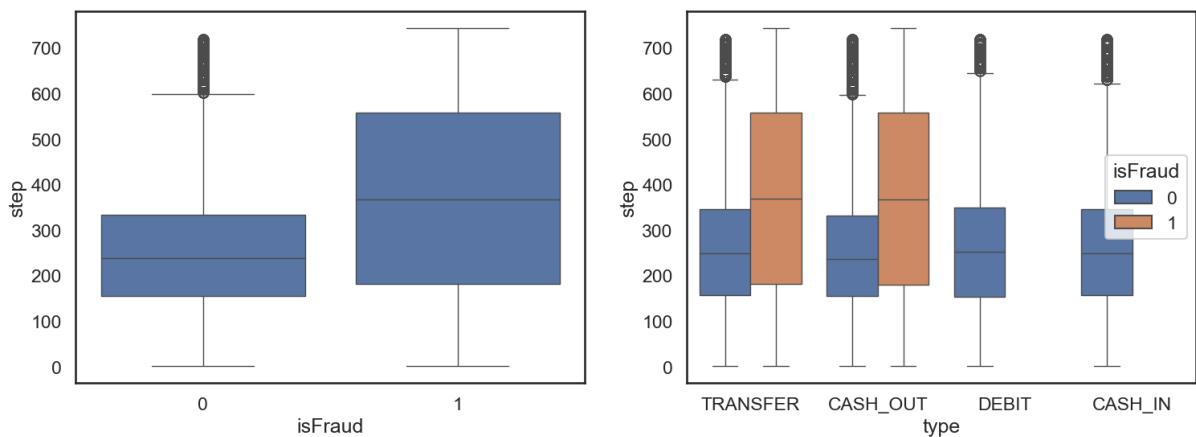
- Nhận xét:

- **oldbalanceOrg và newbalanceOrg (0.78):** Có mối tương quan cao giữa số dư trước và sau giao dịch của tài khoản nguồn. Điều này hợp lý, vì trong các giao dịch không gian lận, số tiền giao dịch thường được trừ trực tiếp từ số dư trước, dẫn đến tương quan này.
- **oldbalanceDest và newbalanceDest (0.91):** Số dư trước và sau giao dịch của tài khoản đích cũng có tương quan rất cao. Điều này cho thấy giao dịch thường làm thay đổi số dư tài khoản đích theo cách tuyến tính.
- **amount và newbalanceDest (0.27):** Số tiền giao dịch (amount) có tương quan trung bình với số dư cuối của tài khoản đích (newbalanceDest). Điều

này có thể do số tiền giao dịch lớn sẽ ảnh hưởng trực tiếp đến số dư của tài khoản đích.

- **amount và oldbalanceDest (0.14):** Mức tương quan yếu hơn so với newbalanceDest, nhưng vẫn có dấu hiệu cho thấy giá trị giao dịch ảnh hưởng đến số dư tài khoản đích ban đầu.
- **isFraud và các biến khác (-0.04 đến 0.04):** Nhân gian lận (isFraud) không có mối tương quan mạnh với bất kỳ biến số nào trong dữ liệu. Điều này cho thấy giao dịch gian lận không dễ dàng được xác định chỉ dựa trên các đặc điểm liên tục như số dư hoặc giá trị giao dịch.
- **step và các biến khác (-0.01 đến 0.03):** Biến step (thời gian hoặc chu kỳ giao dịch) cũng không có mối liên hệ đáng kể với bất kỳ biến nào khác.

Biểu đồ hộp theo step (thời gian giao dịch)



Hình 9: Biểu đồ hộp theo step (thời gian giao dịch)

- Nhận xét:

- **Biểu đồ bên trái:** Phân phối step (thời gian giao dịch) được tách theo hai nhóm isFraud = 0 (không gian lận) và isFraud = 1 (gian lận).
 - Đối với isFraud = 0 (không gian lận): Phân phối tập trung nhiều hơn ở khoảng từ 100 đến 300. Có một số giá trị ngoại lai ở mức rất cao (>600).
 - Đối với isFraud = 1 (gian lận): Phân phối cũng trải dài từ khoảng 100 đến 600, nhưng giá trị trung bình và phân vị (median) cao hơn so với isFraud = 0. Điều này cho thấy các giao dịch gian lận có xu hướng xảy ra vào các chu kỳ giao dịch muộn hơn.
- **Biểu đồ bên phải:** Phân phối của step được chia theo loại giao dịch (TRANSFER, CASH_OUT, DEBIT, CASH_IN) và phân biệt bởi isFraud.
 - TRANSFER: Giao dịch gian lận (isFraud = 1) xảy ra nhiều hơn ở khoảng step lớn (thường từ 400 trở lên). Giao dịch không gian lận tập trung ở khoảng từ 100 đến 300.

- CASH_OUT: Giao dịch gian lận có phân phối tương tự TRANSFER, với giá trị step cao hơn trung bình so với không gian lận. Xu hướng cho thấy gian lận có thể liên quan đến thời gian thực hiện giao dịch.
- DEBIT và CASH_IN: Không có giao dịch gian lận (isFraud = 1) xuất hiện ở các loại giao dịch này. Điều này phù hợp với các mẫu gian lận thực tế, khi gian lận thường xảy ra ở TRANSFER và CASH_OUT.
- Ta thấy được giao dịch TRANSFER và CASH_OUT là các loại giao dịch chính liên quan đến gian lận. DEBIT và CASH_IN ít hoặc không liên quan đến gian lận. Các giao dịch gian lận xảy ra ở các chu kỳ giao dịch muộn hơn (giá trị step cao hơn).

Chỉ số Kurtosis and skew của tập dữ liệu:

```
num_features = [col for col in df_m.select_dtypes("number").columns if col != "isFraud"]

for c in num_features:
    kurt = scipy.stats.mstats.kurtosis(df_m[c])
    Skew = scipy.stats.mstats.skew(df_m[c])

    print(f"{c} Skew: {np.round(Skew, 2)}, Kurtosis: {np.round(kurt, 2)}")

step Skew: 0.38, Kurtosis: 0.34
amount Skew: 26.42, Kurtosis: 1275.57
oldbalanceOrg Skew: 4.18, Kurtosis: 20.69
newbalanceOrig Skew: 4.13, Kurtosis: 20.11
oldbalanceDest Skew: 17.2, Kurtosis: 690.56
newbalanceDest Skew: 16.77, Kurtosis: 630.94
```

Hình 10: Chỉ số Kurtosis and skew của tập dữ liệu

- Chỉ số Skewness (Độ lệch):

- step (Skew = 0.38): Gần như đối xứng, phân phối không quá lệch. Điều này cho thấy thời gian thực hiện giao dịch (step) có xu hướng phân bố đồng đều trong khoảng quan sát.
- amount (Skew = 26.42): Phân phối cực kỳ lệch phải. Điều này chỉ ra rằng đa số giao dịch có số tiền nhỏ, trong khi một số ít giao dịch có số tiền rất lớn (các giá trị cực đại).
- oldbalanceOrg và newbalanceOrig (Skew ≈ 4.1): Phân phối cũng lệch phải. Tương tự amount, đa số tài khoản ban đầu có số dư thấp, nhưng có một số tài khoản có số dư rất cao.
- oldbalanceDest và newbalanceDest (Skew ≈ 17): Phân phối lệch phải rất mạnh, với một số đích đến có số dư cực kỳ lớn, nhưng phần lớn có số dư thấp.

- **Kurtosis (Độ nhọn):**

- step (Kurtosis = 0.34): Phân phối phẳng hơn phân phối chuẩn, dữ liệu phân tán nhiều hơn quanh trung tâm, không tập trung mạnh vào một khu vực cụ thể.
- amount (Kurtosis = 1275.57): Phân phối cực kỳ nhọn với đuôi dài, chỉ ra rằng hầu hết các giao dịch có số tiền rất nhỏ, trong khi một số ít giao dịch có số tiền cực lớn (giá trị ngoại lai).
- oldbalanceOrg và newbalanceOrig (Kurtosis ≈ 20): Đỉnh rất nhọn, cho thấy phần lớn dữ liệu tập trung quanh giá trị thấp và có một số giá trị ngoại lai lớn.
- oldbalanceDest và newbalanceDest (Kurtosis ≈ 690): Phân phối cực kỳ nhọn, cho thấy dữ liệu bị chi phối bởi số dư nhỏ và một số rất ít tài khoản có số dư cực lớn.

- **Kết luận:**

- Skewness: Hầu hết các biến, đặc biệt là amount, oldbalanceOrg, newbalanceOrig, oldbalanceDest, và newbalanceDest, đều lệch phải mạnh, cho thấy sự bất cân đối trong dữ liệu. Phần lớn các giao dịch hoặc số dư có giá trị nhỏ, nhưng tồn tại một số giá trị ngoại lai lớn.
- Kurtosis: Kurtosis rất cao ở các biến tài chính (amount, oldbalanceOrg, newbalanceOrig, oldbalanceDest, newbalanceDest), cho thấy sự tập trung dữ liệu quanh các giá trị thấp và đuôi phân phối dài do giá trị ngoại lai.

- **Hệ quả:**

- Tập dữ liệu không tuân theo phân phối chuẩn, đặc biệt là ở các biến tài chính, với nhiều giá trị ngoại lai (outliers) đáng kể.
- Khi phân tích hoặc xây dựng mô hình, cần sử dụng các phương pháp thích hợp (như chuẩn hóa log) để giảm ảnh hưởng của độ lệch và độ nhọn.

3.4. Data preprocessing (Tiền xử lý dữ liệu)

3.4.1. Hiệu chỉnh chỉ số Kurtosis and skew của tập dữ liệu

- Dựa vào nhận xét các chỉ số của Kurtosis and skew của tập dữ liệu trên, ta cần điều chỉnh lại 2 chỉ số này để làm giảm sự ảnh hưởng của 2 chỉ số của tập dữ liệu này, từ đó có thể tăng hiệu quả của mô hình huấn luyện.

```

df_processed = df_m.copy()

skewed_columns = ['amount', 'oldbalanceOrig', 'newbalanceOrig', 'oldbalanceDest', 'newbalanceDest']

for col in skewed_columns:
    df_processed[col] = np.log1p(df_processed[col])

for col in skewed_columns:
    lower_limit = df_processed[col].quantile(0.01)
    upper_limit = df_processed[col].quantile(0.99)
    df_processed[col] = np.clip(df_processed[col], lower_limit, upper_limit)

import scipy.stats
for col in skewed_columns:
    skew = scipy.stats.mstats.skew(df_processed[col])
    kurt = scipy.stats.mstats.kurtosis(df_processed[col])
    print(f"{col}: Skew: {np.round(skew, 2)}, Kurtosis: {np.round(kurt, 2)}")

amount: Skew: -0.87, Kurtosis: 1.58
oldbalanceOrig: Skew: -0.28, Kurtosis: -1.4
newbalanceOrig: Skew: 0.5, Kurtosis: -1.62
oldbalanceDest: Skew: -1.8, Kurtosis: 1.82
newbalanceDest: Skew: -2.59, Kurtosis: 6.25

```

Hình 11: Hiệu chỉnh chỉ số Kurtosis and skew của tập dữ liệu

- Hàm hiệu chỉnh chỉ số áp dụng:
 - **Biến đổi logarit:** Giảm ảnh hưởng của ngoại lai.
 - **Clipping (cắt giá trị):** Điều chỉnh giá trị ngoại lai về giới hạn hợp lý.
 - **Đánh giá hiệu quả:** Tính lại Skew và Kurtosis để kiểm tra sự cải thiện.
- Sau khi hiệu chỉnh các chỉ số:
 - **Tính phân phối:** Các cột trở nên cân đối hơn, ít bị ảnh hưởng bởi các giá trị ngoại lai.
 - **Ứng dụng trong mô hình:** Các thuật toán học máy trở nên ổn định và chính xác hơn vì:
 - Tránh được ảnh hưởng tiêu cực của giá trị ngoại lai.
 - Dữ liệu phân phối gần chuẩn giúp các thuật toán tuyến tính hoạt động tốt hơn.

3.4.2. Giảm kích thước của tập dữ liệu

- Kích thước của tập dữ liệu rất lớn (hàng triệu bản ghi), nhãn dữ liệu (label) là isfraud phân phối không đồng đều:

```
df_processed['isFraud'].value_counts()
```

isFraud	
0	4202912
1	8213

Hình 12: Kích thước của tập dữ liệu

- Có 8213 giá trị 1 mà có tới tận 4202912 giá trị 0. Chính vì thế cần giảm kích thước của nhãn không xuống để tập dữ liệu có thể trở nên cân bằng hơn. Từ đó có thể huấn luyện được mô hình hiệu quả hơn, tránh bị thiên vị.

```
from sklearn.utils import resample
import pandas as pd

df_majority = df_processed[df_processed['isFraud'] == 0]
df_minority = df_processed[df_processed['isFraud'] == 1]

df_majority_downsampled = resample(df_majority,
                                   replace=False,
                                   n_samples=len(df_minority)*2,
                                   random_state=42)

df_balanced = pd.concat([df_majority_downsampled, df_minority])
print(df_balanced['isFraud'].value_counts())
```

isFraud	
0	16426
1	8213

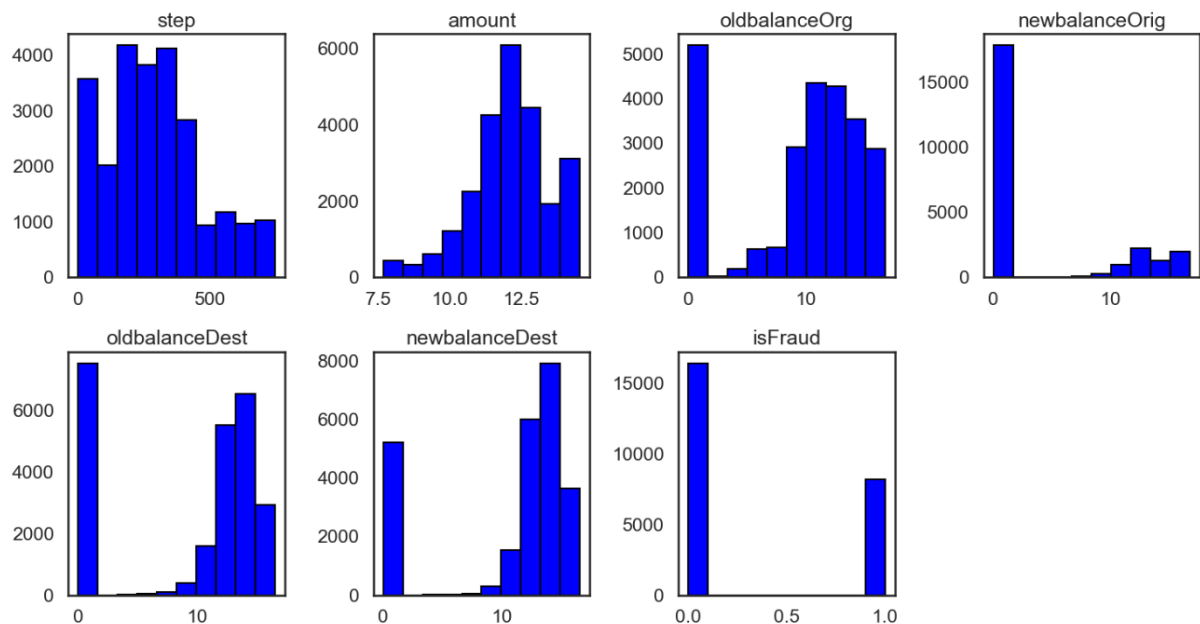
Name: count, dtype: int64

Hình 13: Giảm kích thước của tập dữ liệu

- Đoạn code trên có công dụng chính là thực hiện tái cân bằng dữ liệu (data balancing) để cải thiện hiệu quả của các mô hình học máy, đặc biệt trong trường hợp dữ liệu mất cân đối (imbalanced).
 - df_majority: Tập dữ liệu thuộc lớp chiếm đa số (isFraud = 0 - giao dịch hợp lệ).
 - df_minority: Tập dữ liệu thuộc lớp thiểu số (isFraud = 1 - giao dịch gian lận).

- Sử dụng kỹ thuật resample đối với df_majority, ta được df_majority_downsampled là phiên bản rút gọn của lớp đa số.
 - Sau khi rút gọn lớp đa số, dữ liệu của cả hai lớp (majority và minority) được ghép lại thành một tập mới df_balanced.
- **Tập dữ liệu sau khi giảm kích thước này cân bằng hơn**, với 16426 nhãn 0 và 8231 nhãn 1. Điều này giúp các mô hình học máy không bị lệch về lớp chiếm đa số.

Nhận xét về biểu đồ đóng góp của các đặc trưng sau khi điều chỉnh dữ liệu:



Hình 14: Biểu đồ đóng góp của các đặc trưng sau khi điều chỉnh dữ liệu

- **amount**: Phân phối sau khi áp dụng log-transformation trở nên gần giống phân phối chuẩn hơn. Các giá trị bị lệch về phía cao đã được thu hẹp, giảm đi sự bất đối xứng trước đó.
- **oldbalanceOrig** và **newbalanceOrig**: Phân phối của các cột này cũng cải thiện đáng kể, với độ lệch được giảm đi nhờ phép log và cắt tỉa (clipping). Tuy nhiên, vẫn còn một số giá trị tập trung lớn gần mức thấp.
- **oldbalanceDest** và **newbalanceDest**: Sau điều chỉnh, các cột này cũng có dạng phân phối gần đối xứng hơn. Tuy nhiên, có sự tập trung lớn ở giá trị thấp (như 0), có thể liên quan đến các giao dịch không hợp lệ hoặc không được khởi tạo số dư.
- **Step**: Phân phối của cột này không có sự biến đổi đáng kể vì không thuộc danh sách cột bị điều chỉnh. Nó vẫn thể hiện sự phân phối tự nhiên của các bước thời gian trong dữ liệu.

Kurtosis and skew của Df_Balanced

```
num_features = [colm for colm in df_balanced.select_dtypes("number").columns if colm != "isFraud"]

for c in num_features:
    kurt = scipy.stats.mstats.kurtosis(df_balanced[c])
    Skew = scipy.stats.mstats.skew(df_balanced[c])

    print(f"{c} Skew: {np.round(Skew, 2)}, Kurtosis: {np.round(kurt, 2)}")

step Skew: 0.54, Kurtosis: -0.24
amount Skew: -0.47, Kurtosis: 0.58
oldbalanceOrg Skew: -0.82, Kurtosis: -0.71
newbalanceOrig Skew: 1.1, Kurtosis: -0.65
oldbalanceDest Skew: -0.7, Kurtosis: -1.34
newbalanceDest Skew: -1.22, Kurtosis: -0.25
```

Hình 15: Chỉ số Kurtosis and skew của tập dữ liệu cuối cùng

- **Hiệu quả của việc điều chỉnh dữ liệu:** Sự bất đối xứng (skewness) của các cột số liệu ban đầu đã giảm đi đáng kể, làm cho phân phối dữ liệu trở nên "mềm mại" và dễ sử dụng cho các thuật toán học máy. Việc giảm kurtosis cũng làm giảm tầm ảnh hưởng của các giá trị ngoại lệ. Các biểu đồ sau điều chỉnh cho thấy dữ liệu đã được cải thiện đáng kể về tính phân phối, giúp mô hình hóa và phân tích chính xác hơn.

3.4.3. Chia tập dữ liệu Df_Balanced để huấn luyện và đánh giá

```
X = df_balanced[['step', 'type', 'amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest', 'newbalanceDest']]
y = df_balanced['isFraud']
split_train = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
split_val = StratifiedShuffleSplit(n_splits=1, test_size=0.25, random_state=42)
def train_test_val_split(X, y, split_train, split_val):
    train_idx_first, test_idx = next(split_train.split(X, y))

    X_train_first = X.iloc[train_idx_first]
    y_train_first = y.iloc[train_idx_first]

    X_test = X.iloc[test_idx]
    y_test = y.iloc[test_idx]

    train_idx, val_idx = next(split_val.split(X_train_first, y_train_first))

    X_train = X_train_first.iloc[train_idx]
    y_train = y_train_first.iloc[train_idx]

    X_val = X.iloc[val_idx]
    y_val = y.iloc[val_idx]

    return X_train, y_train, X_test, y_test, X_val, y_val
X_train, y_train, X_test, y_test, X_val, y_val = train_test_val_split(X, y, split_train, split_val)
```

Hình 16: Chia tập dữ liệu Df_Balanced để huấn luyện và đánh giá

- Mục đích là thực hiện chia dữ liệu thành các tập *train*, *test*, và *validation*, sử dụng phương pháp **StratifiedShuffleSplit** từ thư viện **sklearn**
 - **Train:** Dùng để huấn luyện mô hình.

- **Validation (val):** Dùng để đánh giá hiệu năng mô hình trong quá trình tối ưu hóa siêu tham số.
 - **Test:** Dùng để đánh giá mô hình sau khi đã tối ưu.
 - **Phân tầng (Stratification):** Việc sử dụng **StratifiedShuffleSplit** đảm bảo rằng tỷ lệ của các lớp trong biến mục tiêu (y - cụ thể là isFraud) được giữ nguyên trong cả ba tập dữ liệu. Điều này rất quan trọng đối với bài toán mất cân bằng dữ liệu hoặc dữ liệu nhị phân (như trong trường hợp này, isFraud có giá trị 0 hoặc 1).
- Kết quả sau khi chạy hàm:
- **X_train, y_train:** Tập dữ liệu chính để huấn luyện mô hình (60%)
 - **X_val, y_val:** Tập dữ liệu để kiểm tra và tối ưu mô hình trong quá trình phát triển (20%)
 - **X_test, y_test:** Tập dữ liệu độc lập để đánh giá hiệu năng cuối cùng của mô hình (20%)

```
print(f"X_train shape: {X_train.shape}")
print(f"X_val shape:{X_val.shape}")
print(f"X_test shape:{X_test.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"y_val shape: {y_val.shape}")
print(f"y_test shape: {y_test.shape}")
```

```
X_train shape: (14783, 7)
X_val shape:(4928, 7)
X_test shape:(4928, 7)
y_train shape: (14783,)
y_val shape: (4928,)
y_test shape: (4928,)
```

- Ý nghĩa: Cách chia dữ liệu này có ý nghĩa rất quan trọng trong quá trình phát triển mô hình học máy, đặc biệt là trong bối cảnh bài toán phân loại với dữ liệu **mất cân bằng** (như trong trường hợp phát hiện giao dịch gian lận).

3.5. Huấn luyện 3 mô hình (Random Forest, Logistic Regression và Gradient Boosting) và đánh giá hiệu suất.

- Sau khi đã chia dữ liệu thành 3 tập Train, Validation và Test. Ta sẽ chọn ra 3 mô hình để huấn luyện (dùng tập Train) và đánh giá hiệu suất dựa trên 5 chỉ số Accuracy, Recall, Precision, F1 và Roc_Auc (dùng tập Validation).

```
def model_performance_val(model, X_train, y_train, X_val, y_val):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_val)
    accuracy = accuracy_score(y_val, y_pred)
    recall = recall_score(y_val, y_pred)
    precision = precision_score(y_val, y_pred)
    f1 = f1_score(y_val, y_pred)
    roc_auc = roc_auc_score(y_val, y_pred)
    return accuracy, recall, precision, f1, roc_auc

rdf_model = RandomForestClassifier(random_state=42, class_weight="balanced", n_jobs=-1)
lg_model = LogisticRegression(penalty='l2', C=1.0, solver='lbfgs', max_iter=200, class_weight='balanced', random_state=42)
xgb_model = xgb.XGBClassifier(seed=42)

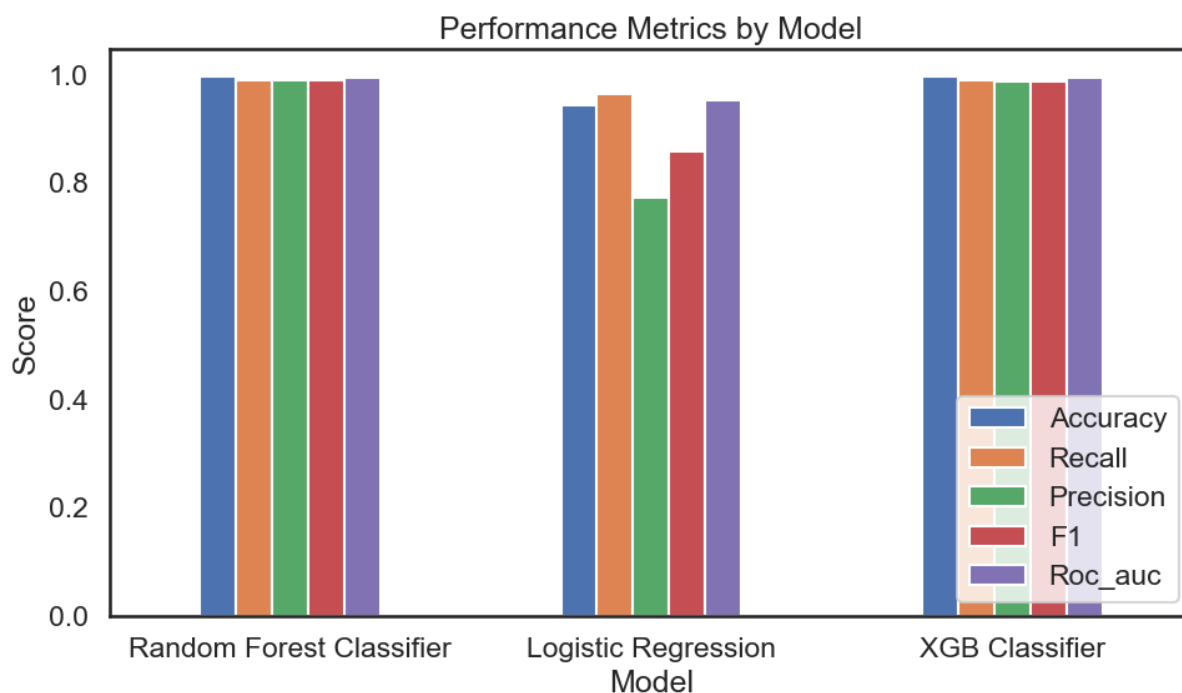
model_per_val_rdf = model_performance_val(rdf_model, X_train, y_train, X_val, y_val)
model_per_val_lg = model_performance_val(lg_model, X_train, y_train, X_val, y_val)
model_per_val_xgb = model_performance_val(xgb_model, X_train, y_train, X_val, y_val)
```

Hình 17: Huấn luyện 3 mô hình

- Sau khi huấn luyện 3 mô hình, ta thu được các chỉ số đánh giá của 3 mô hình như sau:

	Model	Accuracy	Recall	Precision	F1	Roc_auc
0	Random Forest Classifier	0.996753	0.990783	0.990783	0.990783	0.994406
1	Logistic Regression	0.944196	0.965438	0.773777	0.859047	0.952546
2	XGB Classifier	0.995942	0.990783	0.986239	0.988506	0.993914

Hình 18: Chỉ số đánh giá của 3 mô hình



Hình 19: Biểu đồ chỉ số đánh giá của 3 mô hình

- Đối với **Random Forest Classifier**: Hiệu suất tốt nhất về hầu hết các chỉ số, với độ chính xác (Accuracy), Recall, Precision, F1 và ROC AUC đều rất cao. ROC AUC gần đạt giá trị tối đa (0.994406), cho thấy mô hình này phân biệt tốt giữa các lớp.
- Đối với **Logistic Regression**: Hiệu suất thấp hơn đáng kể so với hai mô hình còn lại, đặc biệt là Precision (0.773777) và F1 (0.859047). ROC AUC (0.952546) cho thấy khả năng phân biệt giữa các lớp vẫn ổn, nhưng thấp hơn các mô hình còn lại.
- Đối với **XGB Classifier (XGBoost)**: Cạnh tranh với Random Forest và chỉ thua kém một chút về ROC AUC (0.993914 so với 0.994406). Precision thấp hơn một chút so với Random Forest nhưng vẫn cao (0.986239). Hiệu suất tổng thể gần tương đương với Random Forest.

Qua đánh giá các chỉ số của 3 mô hình dựa trên tập dữ liệu Validation, ta có thể thấy hiệu suất của mô hình **Random Forest Classifier** là tốt nhất trong 3 mô hình. Từ đó ta sẽ chọn mô hình **Random Forest Classifier** làm mô hình huấn luyện chính cho bài toán phát hiện gian lận.

3.6. Tối ưu hóa tham số của mô hình Random Forest Classifier và đưa ra model cuối cùng.

```
from hyperopt import hp, fmin, tpe, STATUS_OK, Trials
from hyperopt.pyll.base import scope
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score, StratifiedKFold
import numpy as np
strkf = StratifiedKFold(n_splits=4)
space = {
    'n_estimators': scope.int(hp.uniform('n_estimators', 50, 300)),
    'max_depth': scope.int(hp.uniform('max_depth', 3, 20)),
    'min_samples_split': hp.randint('min_samples_split', 2, 10),
    'min_samples_leaf': hp.randint('min_samples_leaf', 1, 10),
    'max_features': hp.uniform('max_features', 0.5, 1.0),
    'bootstrap': hp.choice('bootstrap', [True, False]),
    'random_state': 42,
}
def objective(param):
    model = RandomForestClassifier(**param)
    roc_auc = cross_val_score(model, X_train, y_train, scoring="roc_auc", cv=strkf).mean()
    return {'loss': -roc_auc, 'status': STATUS_OK}
trials = Trials()
best = fmin(fn=objective,          # Hàm mục tiêu
            space=space,          # Không gian tìm kiếm
            algo=tpe.suggest,     # Thuật toán tối ưu hóa (TPE)
            max_evals=70,        # Số lượng thử nghiệm tối đa
            trials=trials)        # Lưu lại thông tin về các thử nghiệm
print("Best hyperparameters:", best)
```

Hình 20: Tối ưu hóa tham số của mô hình Random Forest Classifier

- Mục tiêu là sử dụng Đoạn mã này sử dụng **Hyperopt** và thuật toán **TPE** để tối ưu hóa các siêu tham số của **RandomForestClassifier** nhằm cải thiện chỉ số **ROC AUC** trong bài toán phát hiện gian lận này.
- Quá trình tối ưu hóa được thực hiện qua 70 thử nghiệm, với việc đánh giá mô hình bằng **cross-validation**.
- Kết quả cuối cùng là bộ siêu tham số tốt nhất cho mô hình **RandomForestClassifier**.

Sau khi đã tìm được siêu tham số tốt nhất cho mô hình, ta tiến hành huấn luyện mô hình cuối cùng dựa trên các tham số đã được tối ưu:

```
final_rdf_model = RandomForestClassifier(
    bootstrap=False,
    max_depth=int(best['max_depth']),
    max_features=best['max_features'],
    min_samples_leaf=best['min_samples_leaf'],
    min_samples_split=best['min_samples_split'],
    n_estimators=int(best['n_estimators'])
)
final_rdf_model.fit(X_train, y_train)
```

RandomForestClassifier

RandomForestClassifier(bootstrap=False, max_depth=12, max_features=0.7589920792536271, min_samples_leaf=2, min_samples_split=9, n_estimators=182)

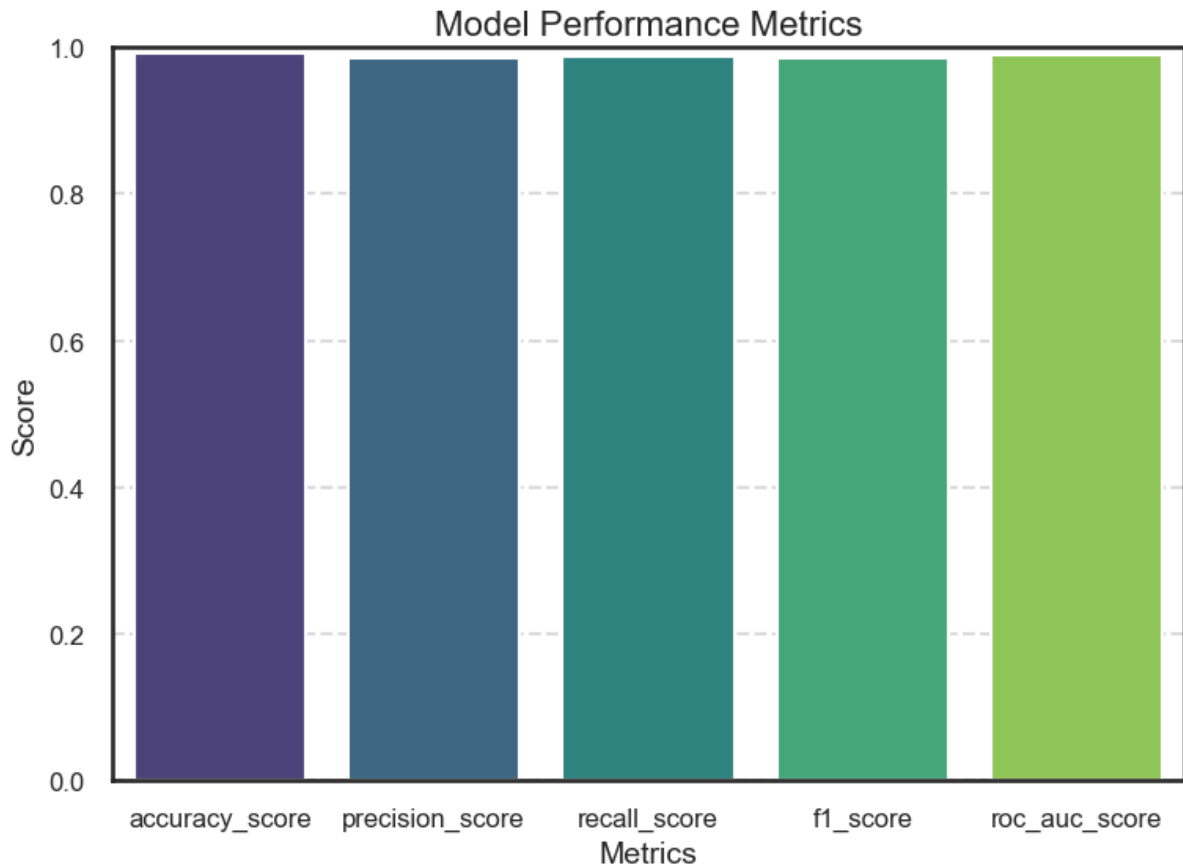
Hình 21: Huấn luyện mô hình cuối cùng

Sau khi đã huấn luyện mô hình cuối cùng, ta đánh giá các chỉ số đánh giá của mô hình dựa trên tập dữ liệu Test:

```
y_pred = final_rdf_model.predict(X_test)
rdf_per_test = model_performance_test(y_test, y_pred)
print(rdf_per_test)
plot_model_performance(rdf_per_test)
```

	accuracy_score	precision_score	recall_score	f1_score	roc_auc_score
0	0.991071	0.984839	0.988436	0.986634	0.990413

Hình 22: Chỉ số đánh giá của mô hình cuối cùng



Hình 23: Biểu đồ chỉ số đánh giá của mô hình cuối cùng

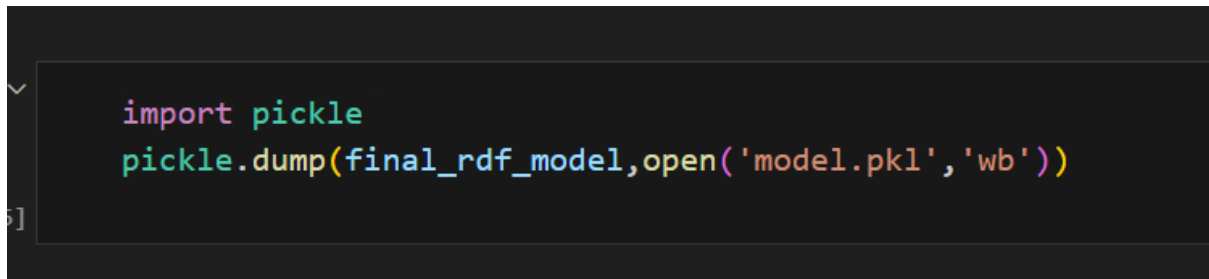
- Nhận xét:

- Accuracy (Độ chính xác): Với giá trị **0.9911** (hoặc 99.11%), mô hình của bạn đạt được độ chính xác rất cao, có nghĩa là mô hình dự đoán đúng gần 99% tổng số mẫu. Điều này cho thấy mô hình hoạt động rất tốt về mặt tổng quát.
- Precision (Độ chính xác của lớp dương): Với **precision = 0.9848**, mô hình có độ chính xác cao trong việc phân loại đúng các giao dịch gian lận (fraud). Điều này có nghĩa là khi mô hình dự đoán một giao dịch là gian lận, khả năng cao là giao dịch đó thực sự gian lận.
- Recall (Tỷ lệ phát hiện đúng): Với **recall = 0.9884**, mô hình có khả năng phát hiện gần như tất cả các giao dịch gian lận, chỉ bỏ sót rất ít. Điều này quan trọng trong các bài toán như phát hiện gian lận, vì việc phát hiện ra gian lận sớm là rất quan trọng.
- F1-Score: **F1-score = 0.9866** cho thấy mô hình có sự cân bằng tốt giữa việc không dự đoán sai quá nhiều giao dịch là gian lận và khả năng phát hiện gian lận cao. Đây là một chỉ số mạnh cho thấy mô hình đã đạt được sự cân bằng tốt giữa việc không bỏ sót gian lận và không dự đoán quá nhiều giao dịch là gian lận.

- ROC AUC Score: **ROC AUC = 0.9904** cho thấy mô hình có khả năng phân biệt giữa các giao dịch gian lận và không gian lận cực kỳ tốt, với khoảng cách rõ rệt giữa các lớp.
- Ta có thể thấy mô hình tối ưu hóa rất tốt các tham số và có hiệu suất vượt trội trong việc phát hiện gian lận.

3.7. Xây dựng trang web phát hiện gian lận dựa trên mô hình đã được huấn luyện

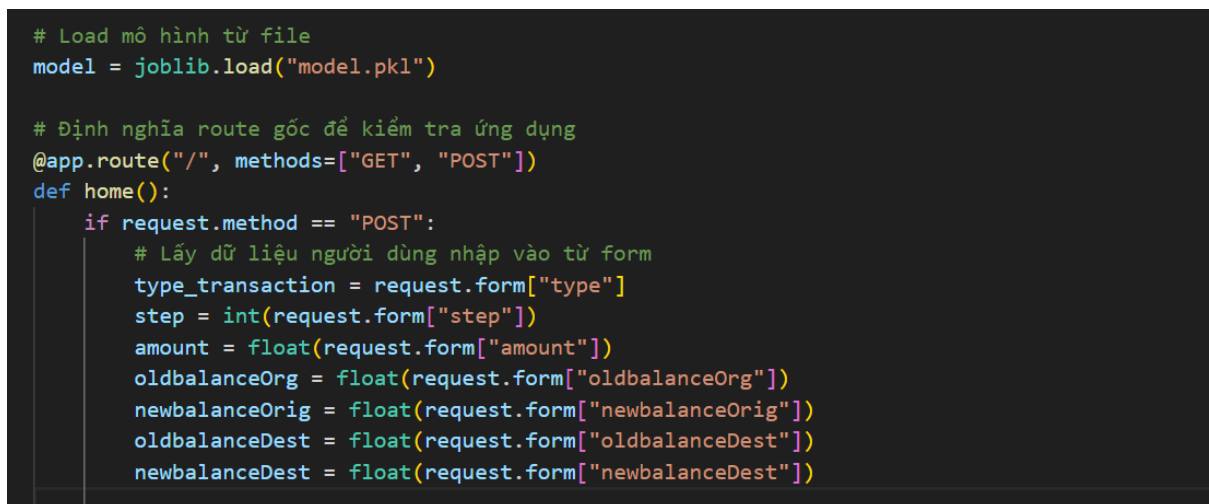
- Sau khi đã huấn luyện được mô hình cuối cùng, ta lưu trữ mô hình học máy vào tệp model.pkl:



```
import pickle
pickle.dump(final_rdf_model, open('model.pkl', 'wb'))
```

Hình 24: Lưu trữ mô hình

- Xây dựng trang web, mục tiêu là lấy dữ liệu người dùng nhập vào các form và đưa ra kết quả dự đoán:



```
# Load mô hình từ file
model = joblib.load("model.pkl")

# Định nghĩa route gốc để kiểm tra ứng dụng
@app.route("/", methods=["GET", "POST"])
def home():
    if request.method == "POST":
        # Lấy dữ liệu người dùng nhập vào từ form
        type_transaction = request.form["type"]
        step = int(request.form["step"])
        amount = float(request.form["amount"])
        oldbalanceOrg = float(request.form["oldbalanceOrg"])
        newbalanceOrig = float(request.form["newbalanceOrig"])
        oldbalanceDest = float(request.form["oldbalanceDest"])
        newbalanceDest = float(request.form["newbalanceDest"])
```

- Giao diện trang web:

Nhập thông tin đầu vào:

Loại giao dịch:
PAYMENT

Thời gian giao dịch:
23

Số tiền trong giao dịch:
11223

Số dư của tài khoản gửi trước giao dịch:
23223

Số dư của tài khoản gửi sau giao dịch:
22

Số dư của tài khoản nhận trước giao dịch:
22

Số dư của tài khoản nhận sau giao dịch:
3333

Prediction

Prediction Result

Result:

Kết quả giao dịch là: **không có khả năng gian lận** với xác suất không gian lận là 92.23%.

Hình 25: Giao diện trang web

CHƯƠNG IV: KẾT LUẬN

4.1. Kết luận chung

- Bài toán phát hiện gian lận tài chính là một trong những thách thức quan trọng trong lĩnh vực khoa học dữ liệu và học máy, đặc biệt trong bối cảnh sự gia tăng nhanh chóng của các giao dịch tài chính số. Với mục tiêu bảo vệ các tổ chức tài chính và khách hàng khỏi những tổn thất nặng nề do các hành vi gian lận, việc xây dựng các hệ thống phát hiện gian lận hiệu quả đã trở thành yêu cầu cấp thiết.
- Thông qua nghiên cứu và áp dụng các phương pháp hiện đại như học máy, học sâu, và phân tích dữ liệu lớn, các hệ thống phát hiện gian lận ngày càng được cải thiện, với khả năng nhận diện chính xác các giao dịch bất thường và giảm thiểu cảnh báo sai. Tuy nhiên, thách thức vẫn tồn tại do tính không cân bằng trong dữ liệu, sự thay đổi liên tục của hành vi gian lận, và yêu cầu cân đối giữa hiệu quả phát hiện và chi phí vận hành.
- Đề tài này không chỉ cung cấp một cái nhìn toàn diện về các kỹ thuật phát hiện gian lận hiện đại mà còn đề xuất các hướng tiếp cận phù hợp để giải quyết bài toán trong môi trường thực tiễn. Qua đó, nghiên cứu hy vọng đóng góp một phần vào việc tăng cường khả năng phòng chống gian lận, bảo vệ hệ thống tài chính và xây dựng lòng tin cho các bên tham gia trong nền kinh tế số hóa ngày càng phát triển.

4.2. Hướng phát triển cho tương lai

- **Tích hợp công nghệ tiên tiến:**
 - Học sâu (Deep Learning): Sử dụng các mô hình học sâu như mạng nơ-ron tích chập (CNN), mạng nơ-ron hồi tiếp (RNN) hoặc các mô hình tự mã hóa (Autoencoder) để nhận diện các mối quan hệ phức tạp và phát hiện các hành vi gian lận tinh vi hơn.
 - AI Generative Models: Các mô hình như GANs (Generative Adversarial Networks) có thể tạo ra các mẫu dữ liệu giả để kiểm tra và cải thiện khả năng phát hiện của hệ thống.
- **Xử lý dữ liệu lớn và thời gian thực:**
 - Hệ thống xử lý thời gian thực: Phát triển các giải pháp có khả năng phát hiện gian lận ngay khi giao dịch diễn ra, sử dụng công nghệ xử lý dữ liệu thời gian thực như Apache Kafka, Spark Streaming, hoặc Flink.
 - Phân tích dữ liệu lớn (Big Data Analytics): Tích hợp các nền tảng dữ liệu lớn như Hadoop hoặc Elasticsearch để xử lý khối lượng giao dịch khổng lồ với tốc độ cao và khả năng mở rộng.
- **Cải thiện độ chính xác của mô hình:**
 - Xử lý dữ liệu không cân bằng: Nghiên cứu các kỹ thuật mới để giải quyết bài toán dữ liệu không cân bằng, chẳng hạn như sử dụng kỹ thuật

oversampling, undersampling hoặc các thuật toán cải tiến như SMOTE và ADASYN.

- Học chuyển giao (Transfer Learning): Tận dụng mô hình đã được huấn luyện trên các tập dữ liệu tương tự để áp dụng vào các lĩnh vực mới mà không cần phải thu thập một lượng lớn dữ liệu huấn luyện.

- **Ứng dụng đa lĩnh vực:**

- Bảo hiểm và chứng khoán: Mở rộng ứng dụng phát hiện gian lận sang các lĩnh vực như bảo hiểm (phát hiện yêu cầu bồi thường giả) và giao dịch chứng khoán (phát hiện thao túng thị trường).
- Thương mại điện tử: Tăng cường phát hiện gian lận trong các giao dịch trực tuyến như giả mạo danh tính, mua hàng bất thường hoặc sử dụng thẻ tín dụng bị đánh cắp.

Phát hiện gian lận tài chính trong tương lai sẽ không chỉ dừng lại ở việc áp dụng các công nghệ hiện có mà còn mở rộng sang việc tích hợp các xu hướng công nghệ mới, nâng cao hiệu quả xử lý dữ liệu, và tối ưu hóa trải nghiệm người dùng. Với sự phát triển mạnh mẽ của trí tuệ nhân tạo và dữ liệu lớn, các hệ thống phát hiện gian lận sẽ ngày càng trở nên thông minh hơn, đáp ứng tốt hơn các thách thức và yêu cầu của một thế giới tài chính số hóa không ngừng thay đổi.

Link project github: [github](#)

TÀI LIỆU THAM KHẢO

- [1] Ian H. Witten, Eibe Frank, Mark A. Hall: “Data Mining: Practical Machine Learning Tools and Techniques”.
- [2] Christopher M. Bishop: "Pattern Recognition and Machine Learning".
- [3] Fabrizio Sebastiani (2002): "Machine Learning in Automated Text Categorization".
- [4] Wei-Meng Lee: “Python Machine Learning”.
- [5] Scikit-learn Documentation: <https://scikit-learn.org/stable/>