ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC VÀ KỸ THUẬT THÔNG TIN

PHẠM THẢO NHI – 19520815

TRỊNH LINH CHI – 19521285

ĐỒ ÁN CUỐI KỲ

MÔN: HỌC MÁY THỐNG KÊ

LỚP: DS102.L21

PHÂN LOẠI GIAO DỊCH THẺ TÍN DỤNG GIAN LẬN

CREDIT CARD FRAUD DETECTION PROJECT

SINH VIÊN NGÀNH KHOA HỌC DỮ LIỆU

GIẢNG VIÊN HƯỚNG DẪN

TS. NGUYỄN TẤN TRẦN MINH KHANG

ThS. VÕ DUY NGUYÊN

TP. HỒ CHÍ MINH, 2021

LỜI CẢM ƠN

Lời đầu tiên, nhóm em xin gửi lời cảm ơn chân thành đến Trường Đại học Công nghệ Thông tin, Khoa Khoa học và Kỹ thuật Thông tin đã tạo điều kiện cho chúng em có cơ hội được tiếp cận với môn học "Học máy thống kê". Đặc biệt, chúng em xin gửi lời cảm ơn sâu sắc đến Thầy Nguyễn Tấn Trần Minh Khang và Thầy Võ Duy Nguyên – giảng viên bộ môn "Học máy thống kê" đã tận tình hướng dẫn, truyền đạt cho chúng em những kiến thức quý báu cũng như kỹ năng cơ bản cần có để có thể học tập hiệu quả trong suốt quá trình học tập và thực hiện đề tài nghiên cứu này.

Tuy nhiên, trong quá trình nghiên cứu và thực hiện đề tài, vì kiến thức chuyên môn vẫn còn nhiều hạn chế cùng với việc trình độ bản thân còn hạn hẹp nên nhóm em có thể vẫn còn nhiều thiếu sót khi tìm hiểu, đánh giá và trình bày về đề tài. Rất mong nhận được sự quan tâm, ý kiến đóng góp và sự đánh giá của Thầy để chúng em có thể hoàn thiện và phát triển hơn trong tương lai.

Cuối cùng, chúng em xin kính chúc Thầy nhiều sức khoẻ, thành công và hạnh phúc.

Chúng em xin chân thành cảm ơn!

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG

DANH MỤC TỪ VIẾT TẮT

STT	Từ viết tắt	Ý nghĩa
1	CP	Card-present
2	CNP	Card-not-present
3	EMV	Europay Mastercard và Visa
4	LR	Logistic Regression
5	RF	Random Forest
6	KNN	K-Nearest Neighbors
7	PCA	Principal Component Analysis
8	TP	True Positive
9	TN	True Negative
10	FP	False Positive
11	FN	False Negative
12	TPR	True Positive Rate
13	FPR	False Positive Rate
14	TNR	True Negative Rate
15	FNR	False Negative Rate

Bảng 1. *Bảng danh mục từ viết tắt*

CHƯƠNG I. GIỚI THIỆU

1.1. Giới thiệu tổng quan về đề tài

1.1.1. Khái niệm giao dịch gian lận qua thẻ tín dụng (Credit Card Fraud Detection)

Thẻ tín dụng (Credit Card) là một loại thẻ ngân hàng mà người sở hữu được phép sử dụng để thanh toán mà không cần tiền trong thẻ. Có thể hiểu thẻ tín dụng cho phép chủ thẻ "mượn" tiền của ngân hàng trong một hạn mức nhất định để thanh toán và họ sẽ phải hoàn trả lại số tiền ấy cho ngân hàng vào cuối mỗi kỳ.

Hiện nay, các giao dịch bằng thẻ tín dụng đang trở nên phổ biến và quan trọng trong đời sống. Bằng chứng là, con người ngày càng có xu hướng hạn chế mang theo và sử dụng tiền mặt. Khi tần suất giao dịch bằng thẻ ngày càng tăng, số lượng các giao dịch gian lận cũng tăng lên nhanh chóng. Gian lận trong các giao dịch bằng thẻ tín dụng (Credit Card Fraud) là việc sử dụng trái phép tài khoản bởi một người không phải là chủ sở hữu của tài khoản đó.

1.1.2. Các hình thức giao dịch bằng thẻ

Có hai hình thức giao dịch bằng thẻ:

- Hình thức giao dịch bằng thẻ vật lý, hay còn được gọi là card-present (CP), là tình huống giao dịch cần thẻ vật lý, chẳng hạn như giao dịch tại cửa hàng (còn được gọi là điểm bán hàng - POS) hoặc giao dịch tại điểm rút tiền (ví dụ: tại máy rút tiền tự động - ATM).
- Hình thức giao dịch không cần thẻ vật lý, hay còn được gọi là card-not-present (CNP). Như tên gọi của nó, đây là các tình huống không cần sử dụng thẻ vật lý, có thể kể đến các khoản thanh toán được thực hiện trên Internet, qua điện thoại hoặc qua thư.

Đối với mỗi hình thức, sẽ có các kỹ thuật khác nhau được sử dụng để thực hiện hành vi gian lận. Quan trọng hơn, những kẻ lừa đảo gần đây có nhiều khả năng khai thác các khiếm khuyết của các hình thức CNP hơn là các hình thức CP, có thể là do các CP đã tồn tại hơn hai thập kỷ nay và đã trở nên khá mạnh mẽ đối với các cuộc tấn công nhằm thực hiện gian lận, đặc biệt là nhờ công nghệ EMV (Europay Mastercard và Visa, tức là thẻ được nhúng chip) [1]. Trong những năm gần đây, gian lận trực tuyến là một vấn đề ngày càng gia tăng. Nhất là vào thời điểm hiện tại, khi cả người tiêu dùng và doanh nghiệp đang phải đối mặt với đại dịch toàn cầu

COVID-19, bắt buộc họ phải thích ứng và thực hiện nhiều giao dịch qua thẻ tín dụng bằng hình thức CNP hơn. Điều này dẫn đến sự gia tăng mua sắm trực tuyến và thương mại điện tử, đồng thời mở ra một sân chơi lớn hơn cho những kẻ gian lận thử sức những thủ thuật mới [2].

Các biện pháp ngăn chặn hành vi gian lận bằng thẻ tín dụng này có thể được nghiên cứu và thực hiện nhờ vào Machine Learning cùng với các thuật toán của nó.

Trong báo cáo này, chúng tôi dựa trên bộ dữ liệu có sẵn, huấn luyện các mô hình Machine Learning để phân loại giao dịch gian lận bằng thẻ tín dụng. Đồng thời phân tích hiệu suất của các phương pháp phân loại khác nhau bao gồm Random Forest (RF), Logistic Regression (LR), K-Nearest Neighbors (KNN). Báo cáo này được chia thành các phần khác nhau bao gồm các nội dung: giới thiệu về đề tài, trình bày một số cơ sở lý thuyết liên quan, tìm hiểu và phân tích bộ dữ liệu, huấn luyện các mô hình máy học. Sau đó, chúng tôi tiến hành đánh giá các mô hình và so sánh đối chiếu giữa các mô hình để tìm ra mô hình nào hoạt động tốt nhất.

1.1.3. Thách thức trong việc phát hiện gian lận thẻ tín dụng

Thách thức là nhận ra các giao dịch thẻ tín dụng gian lận để những khách hàng của các công ty thẻ tín dụng không bị tính phí cho các giao dịch mà họ không thực hiện.

Những thách thức chính liên quan đến phát hiện gian lận thẻ tín dụng là:

- Lượng dữ liệu lớn được xử lý hàng ngày và quá trình xây dựng mô hình phải đủ nhanh để phản hồi kịp thời với hành vi lừa đảo.
- Dữ liệu không cân bằng: Hầu hết các giao dịch (99.827%) là không gian lận, điều này khiến cho việc phát hiện các giao dịch gian lận thật sự khó khăn.
- Dữ liệu bị phân loại sai có thể là một vấn đề lớn khác vì không phải mọi giao dịch gian lận đều bị bắt hoặc ghi lại.
- Các kỹ thuật thích ứng được những kẻ lừa đảo sử dụng để chống lại mô hình. [3]

1.1.4. Cách giải quyết thách thức trong việc phát hiện gian lận thẻ tín dụng

- Mô hình được sử dụng phải đơn giản và đủ nhanh để phát hiện sự bất thường và phân loại được đó là giao dịch gian lận càng nhanh càng tốt.
- Sự mất cân bằng dữ liệu có thể được xử lý bằng cách sử dụng đúng cách một số phương pháp mà chúng ta sẽ đề cập đến trong phần tiếp theo.
- Để bảo vệ quyền riêng tư của người dùng, kích thước dữ liệu có thể được giảm bớt.
- Làm cho mô hình trở nên đơn giản và dễ hiểu để khi kẻ lừa đảo thích nghi với nó, chỉ cần với một số chỉnh sửa, ta có thể thiết lập và triển khai một mô hình mới.
- Phải lấy một nguồn dữ liệu đáng tin cậy hơn để kiểm tra lại dữ liệu, ít nhất là để đào tạo mô hình. [4]

1.2. Mục tiêu đề tài

- Phân tích được các đặc điểm của bộ dữ liệu.
- Xây dựng các mô hình học máy hiện đại phát hiện gian lận trên thẻ tín dụng bằng cách sử dụng các giao dịch và nhãn của chúng là gian lận hoặc không gian lận để phát hiện xem các giao dịch mới do khách hàng thực hiện có phải là gian lận hay không.
- Đánh giá kết quả đạt được từ các mô hình đã xây dựng.

1.3. Các vấn đề liên quan

Từ khi gian lận thẻ tín dụng trở thành vấn đề lớn đối với thị trường tài chính, nhiều tổ chức tài chính đã phải chi một số tiền lớn và tập hợp các đội ngũ chuyên gia về con người để phát triển các hệ thống phát hiện gian lận. Nhiều nhà nghiên cứu đã và đang tích cực làm việc để giảm thiểu các vấn đề đang phải đối mặt trong khi xây dựng hệ thống phát hiện gian lận như vấn đề mất cân bằng lớp dữ liệu, thay đổi hành vi gian lận,… Trong phần này, chúng tôi sẽ liệt kê một số công trình nghiên cứu ở lĩnh vực này

Trong một nghiên cứu gần đây, Pozzolo và cộng sự đã tập trung vào 2 cách phát hiện gian lận khác nhau: Cách tiếp cận tĩnh (Static approach), trong đó, mô hình phát hiện gian lận được đào tạo theo mùa (ví dụ mỗi tháng hoặc một năm một lần) và cách tiếp cận trực tuyến, trong đó mô hình được cập nhật ngay sau khi dữ liệu giao dịch được ghi nhận. Họ đã tuyên bố rằng phương pháp tiếp cận trực tiếp là phương pháp tốt hơn vì các hành vi gian lận thay đổi theo thời gian. Pozzolo và cộng sự cũng đã đề xuất rằng Average Precision (AP), Area Under Curve (AUC), PrecisionRank là các độ đo tốt nhất

trong việc phát hiện gian lận. Trong một nghiên cứu khác, Pozzolo cùng các cộng sự đã kết luận rằng rừng ngẫu nhiên (Random Forest) là cách tiếp cận tốt nhất trong việc phát hiện gian lận [5].

Awoyemi và cộng sự đã thực hiện huấn luyện các mô hình K-Nearest Neighbors, Logistic Regression, Naïve Bayes trên dữ liệu giao dịch thẻ tín dụng, dữ liệu này đã được lấy mẫu quá mức bằng cách sử dụng kĩ thuật Synthetic Minority Over-sampling Technique (SMOTE). Kết quả cho thấy KNN cho kết quả tốt hơn so với hai phương pháp còn lại. Hiệu suất mô hình được đo bằng Recall, Precision, balanced classification rate, hệ số tương quan Mathews (Matthews correlation coefficient) [5].

Aleskerov và cộng sự đã đề xuất một hệ thống khai thác cơ sở dữ liệu sử dụng Neural Networks để phát hiện gian lận. Srivastava và cộng sự đã đề xuất mô hình Hidden Markov Hidden (HMM) phân tích thói quen chi tiêu của khách hàng để phát hiện hành vi gian lận. Wheeler và Aitken đã nghiên cứu nhiều thuật toán để phát hiện gian lận và kết quả cho thấy rằng phương pháp tiếp cận thích ứng (Adaptive approach) có thể lọc và sắp xếp các trường hợp gian lận để giảm số lượng các cuộc điều tra gian lận. Trong một nghiên cứu khác, Baader và Krcmar đã đề xuất một phương pháp kỹ thuật tính năng tự động (Automated feature engineering approach) để giảm tỉ lệ dương tính giả (false positive rate) thường thấy trong các kết quả dự đoán gian lận [5].

CHƯƠNG II. BỘ DỮ LIỆU

2.1. Mô tả tập dữ liệu

Tập dữ liệu được thu thập và phân tích trong quá trình hợp tác nghiên cứu của Worldline và Machine Learning Group của ULB (Université Libre de Bruxelles) về khai thác dữ liệu lớn và phát hiện gian lận, có thể được tìm thấy trên trang web của kaggle. Tập dữ liệu chứa các giao dịch thẻ tín dụng được thực hiện bởi các chủ thẻ Châu Âu vào khoảng tháng 9 năm 2013 và các giao dịch xảy ra trong hai ngày được trình bày trong tập dữ liệu này, bao gồm 284807 giao dịch.

Tập dữ liệu rất mất cân bằng và nghiêng về lớp tích cực (positive class), lớp tích cực là các trường hợp gian lận chiếm 0,173% dữ liệu giao dịch.

Tập dữ liệu có tổng số 31 thuộc tính đầu vào được sử dụng trong nghiên cứu này. Trong đó có chứa các biến đầu vào số là kết quả của phép biến đổi PCA (Principal Component Analysis) gồm các thuộc tính V1, V2,…V28, các thuộc tính không được chuyển đổi PCA là Time (Thời gian), Amount (Số lượng), Class (Lớp).

Đặc điểm hành vi của thẻ được thể hiện qua một số biến số của từng hồ sơ, thể hiện thói quen chi tiêu của khách hàng cùng với các ngày trong tháng, giờ trong ngày, vị trí địa lý hoặc loại hình thương nhân nơi giao dịch diễn ra. Sau đó, các biến này được sử dụng để tạo ra một mô hình phân biệt các giao dịch gian lận. Các chi tiết và thông tin cơ bản của các thuộc tính không thể được trình bày do các vấn đề bảo mật. [6]

2.2. Chi tiết tập dữ liệu

- Tên tập dữ liệu: Credit Card.
- Nguồn: https://www.kaggle.com/mlg-ulb/creditcardfraud
- Tập dữ liệu gồm 1 file csv: creditcard.csv
- Tập dữ liệu chứa 284807 điểm dữ liệu, mỗi điểm dữ liệu có 31 thuộc tính gồm:
    - Time: Thời gian giữa lần giao dịch đó với giao dịch lần đầu tiên của cùng một tài khoản.
    - Amount: Số tiền giao dịch.
    - Class: Nhận giá trị bằng 1 trong trường hợp giao dịch gian lận, bằng 0 trong trường hợp giao dịch bình thường.
    - V1, V2,…V28: Kết quả của phép chuyển đổi PCA, do tính bảo mật nên dữ liệu được ẩn đi dưới dạng này.

2.3. Thống kê dữ liệu

2.3.1. Thống kê dữ liệu trên bộ dữ liệu tổng

STT	Nhãn	Số lượng	Tỉ lệ
1	0	284315	99.827%
2	1	492	0.173%
Trung bình		142403.5	50%
Tổng cộng		284807	100%

Bảng 2.1. *Bảng thống kê dữ liệu trên bộ dữ liệu tổng*

2.3.2. Thống kê dữ liệu về số tiền giao dịch gian lận và không gian lận

```
NON-FRAUD CASE AMOUNT STATS
count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
```

Hình 2.1. *Chi tiết số tiền (Amount) của các giao dịch không gian lận*

```
FRAUD CASE AMOUNT STATS
count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

Hình 2.2. *Chi tiết số tiền (Amount) của các giao dịch gian lận*

Như chúng ta có thể nhận thấy, giá trị tiền trung bình (mean) của những giao dịch gian lận (122.211321) nhiều hơn so với những giao dịch không gian lận (88.291022).

CHƯƠNG III. CƠ SỞ LÝ THUYẾT

3.1. Các phương pháp học máy

3.1.1. Logistic Regression (Hồi quy logistic)

Logistic Regression về cơ bản là một thuật toán phân loại có giám sát. Trong một bài toán phân loại, biến phụ thuộc (hoặc đầu ra), y, chỉ có thể nhận các giá trị rời rạc cho tập hợp các biến đặc trưng (hoặc đầu vào) đã cho, x.

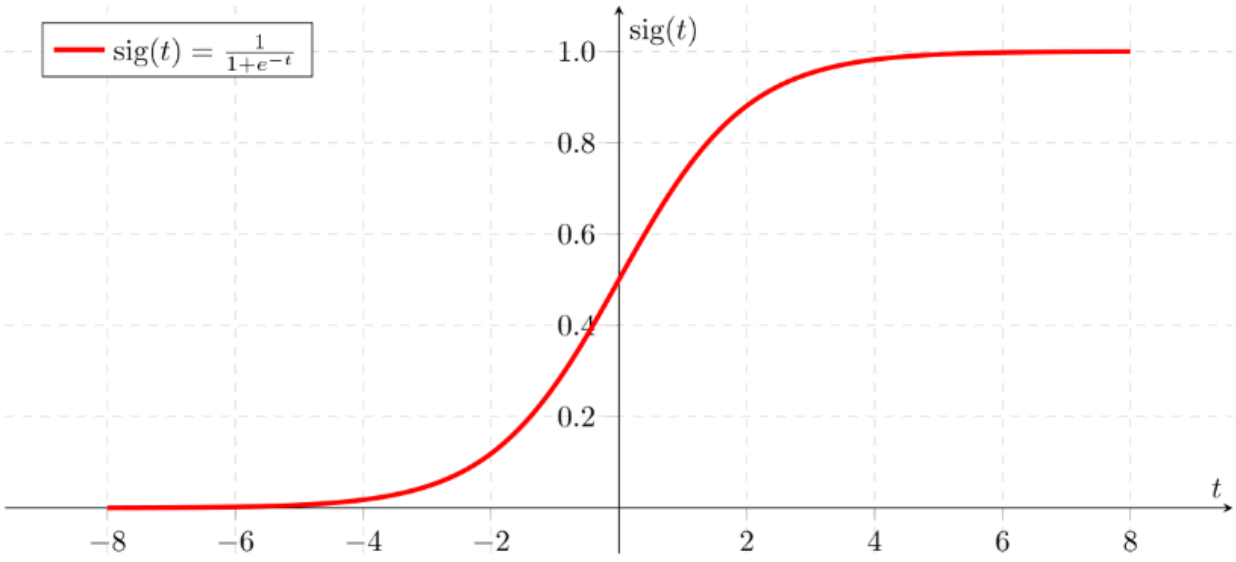
Logistic Regression dự đoán xác suất của một kết quả có hai giá trị bằng không hoặc một, có hoặc không, sai hoặc đúng.

Hồi quy Logistic được biến đổi một chút từ Linear Regression (hồi quy tuyến tính đa biến), bằng cách cho kết quả của Linear Regression vào hàm *sigmoid*, cụ thể:

$y = \text{sigmoid}(f(X)) = \text{sigmoid}\ (w_0 + w_1x_1 + \ldots + w_nx_n)$

Trong đó:

- $w_0, w_1, \ldots, w_n$ là các tham số của mô hình.
- $x_1, x_2, \ldots, x_n$ là các biến (đặc trưng – biến độc lập) đầu vào.
- y là kết quả đầu ra (biến phụ thuộc).
- Hàm sigmoid có dạng: $\text{sigmoid}(x) = \frac{1}{1+e^{-(x)}}$



Hình 3.1. *Đồ thị hàm sigmoid*

Kết quả của hàm *sigmoid* là một số thực trong khoảng (0,1), ta có thể xem kết quả này là một xác suất.

Để có thể chuyển xác suất này về một trong hai giá trị 0 hoặc 1:

- Ta đặt một ngưỡng xác suất nào đó (chẳng hạn 0.5).
- Nếu kết quả nằm dưới ngưỡng này, ta cho kết quả có giá trị 0.
- Nếu kết quả nằm trên ngưỡng này, ta cho kết quả có giá trị 1.

Điểm khác biệt giữa Linear Regression và Logistic Regression là Linear Regression dự báo giá trị của biến phụ thuộc y dựa trên mối quan hệ giữa nó với các biến độc lập x. Còn Logistic Regression hướng đến việc dự báo xác suất, khả năng biến y đạt một trong hai giá trị của nó dựa trên mối quan hệ giữa nó với các biến độc lập x [7].

Logistic Regression được ứng dụng trong nhiều ngành và lĩnh vực khác nhau, ví dụ như:

- Dự báo hay phân loại email có phải là spam hay không spam.
- Dự báo khả năng rời dịch vụ của khách hàng.
- Dự báo tình trạng khối u ung thư là ác tính hay lành tính trong y học.
- Dự báo khả năng khách hàng sẽ mua sản phẩm bất kỳ hay đăng ký dịch vụ.
- Dự báo khả năng trả nợ của khách hàng.

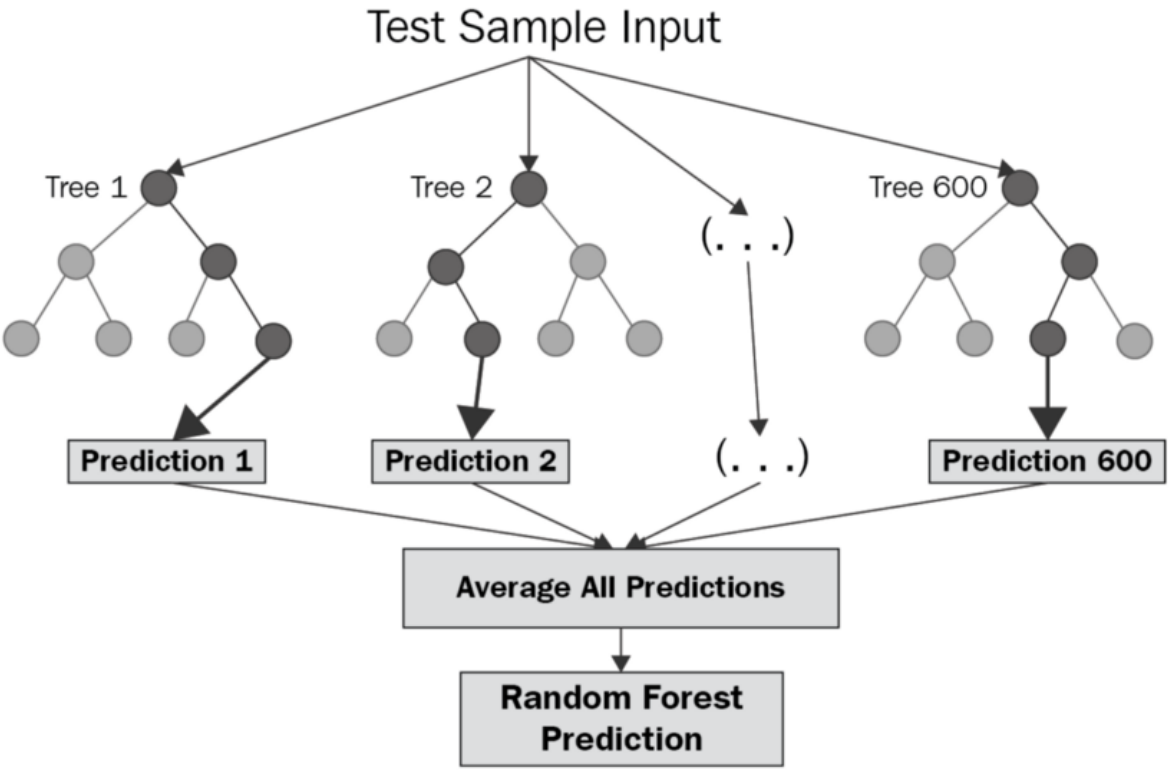
3.1.2. Random Forest (Rừng ngẫu nhiên)

3.1.2.1. Khái niệm

Random Forest [8] là một thuật toán học máy phổ biến thuộc về kỹ thuật học có giám sát (Supervised Learning). Nó có thể được sử dụng cho cả vấn đề phân loại và hồi quy trong Machine Learning. Nó dựa trên khái niệm học tập theo nhóm (Ensemble Learning), là một quá trình kết hợp nhiều bộ phân loại để giải quyết một vấn đề phức tạp và để cải thiện hiệu suất của mô hình.

Rừng ngẫu nhiên là một bộ phân loại chứa một số cây quyết định (Decision Tree) trên các tập con khác nhau của tập dữ liệu đã cho và lấy giá trị trung bình để cải thiện độ chính xác dự đoán của tập dữ liệu đó. Thay vì dựa vào một cây quyết định, rừng ngẫu nhiên lấy dự đoán từ mỗi cây và dựa trên đa số phiếu dự đoán, và nó dự đoán kết quả cuối cùng.

Khi Forest có nhiều cây hơn sẽ dẫn đến độ chính xác cao hơn và chúng ta có thể tránh được việc Overfitting với tập dữ liệu.



Hình 3.2. *Cấu trúc Rừng ngẫu nhiên [9]*

3.1.2.2. Cách Random Forest hoạt động

- Bước 1: Chọn số lượng cây quyết định muốn tạo, gọi là n.
- Bước 2: Xây dựng n cây quyết định, với mỗi cây:

  + Bước 2.1: Chọn K điểm dữ liệu ngẫu nhiên trong tập dữ liệu.

  + Bước 2.2: Xây dựng cây quyết định dựa trên K điểm dữ liệu được chọn.
- Bước 3: Đối với một điểm dữ liệu mới, ta thực hiện dự đoán trên tất cả cây quyết định xây dựng được. Điểm dữ liệu này sẽ được dự đoán là loại c nếu loại này chiếm đa số trong những lần dự đoán.

3.1.2.3. Ưu điểm, nhược điểm của Random Forest

- Ưu điểm:
  - Random Forest tạo ra những dự đoán tốt có thể hiểu được một cách dễ dàng.
  - Nó có thể xử lý các tập dữ liệu lớn một cách hiệu quả.

- Thuật toán Random Forest cung cấp mức độ chính xác cao hơn trong việc dự đoán kết quả so với thuật toán cây quyết định (Decision Tree).

- Nhược điểm:
  - Khi sử dụng Random Forest, cần nhiều tài nguyên hơn vì các khu rừng ngẫu nhiên xử lý các tập dữ liệu lớn hơn, chúng sẽ yêu cầu nhiều tài nguyên hơn để lưu trữ dữ liệu đó.
  - Tiêu tốn nhiều thời gian.

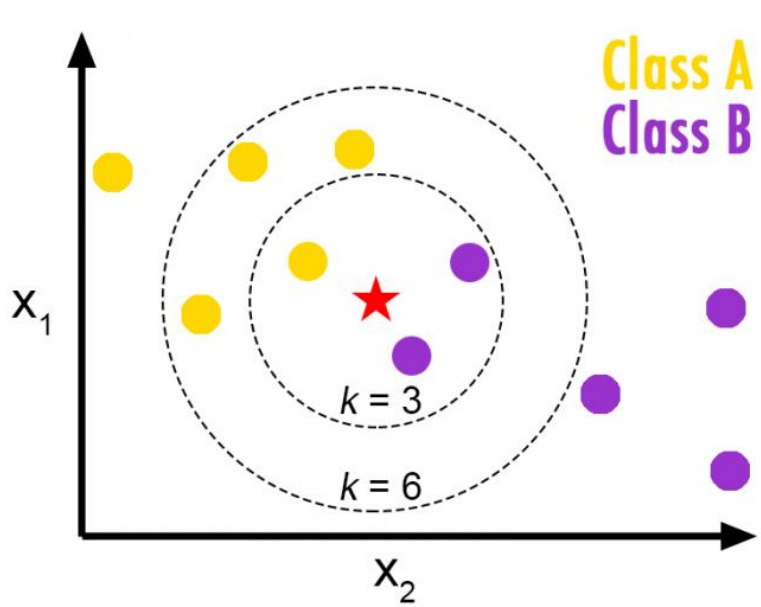
3.1.3. K-Nearest Neighbors (KNN)

3.1.3.1. Khái niệm

KNN (K-Nearest Neighbors) [10] là một trong những thuật toán học có giám sát đơn giản nhất được sử dụng nhiều trong khai phá dữ liệu và học máy. Ý tưởng của thuật toán này là nó không học một điều gì từ tập dữ liệu học (nên KNN được xếp vào loại lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán nhãn của dữ liệu mới.

Thuật toán này sẽ tìm *k* điểm có khoảng cách gần với điểm cần dự đoán nhất. Nhãn đầu ra của điểm cần dự đoán là nhãn có số lần xuất hiện nhiều nhất trong *k* điểm được chọn.

Nói cách khác, lớp (nhãn) của một đối tượng dữ liệu mới có thể dự đoán từ các lớp (nhãn) của k hàng xóm gần nó nhất.



Hình 3.3. *Ví dụ minh hoạ thuật toán KNN [11]*

3.1.3.2. Cách KNN hoạt động

- Chọn một số K bất kỳ, K là một số nguyên, tức là số điểm dữ liệu đã phân loại có khoảng cách gần nhất (láng giềng gần nhất) với điểm dữ liệu chưa phân loại.
- Tính toán khoảng cách (Euclidian, Manhattan, Minkowski,…) giữa điểm dữ liệu chưa phân loại với các điểm dữ liệu đã được phân loại.
- Với kết quả có được sắp xếp theo thứ tự với giá trị khoảng cách từ bé nhất đến lớn nhất.
- Chọn ra các điểm dữ liệu có giá trị khoảng cách bé nhất với điểm dữ liệu cần phân loại dựa trên K cho trước, ví dụ nếu K = 2 tức chọn ra 2 điểm dữ liệu gần nhất, K = 3 là 3 điểm dữ liệu gần nhất.
- Tiếp theo xem xét giá trị của biến mục tiêu (biến phân loại) của các điểm dữ liệu gần nhất, chọn ra giá trị xuất hiện nhiều nhất và gán cho điểm dữ liệu chưa phân loại, ví dụ K = 3, trong đó có 2 điểm dữ liệu được phân loại là A, điểm còn lại là B thì điểm dữ liệu chưa phân loại lúc này sẽ được phân loại là A.
- Kiểm chứng lại độ hiệu quả của model trên test data set, và sử dụng các phương pháp đánh giá khác nhau
- Thay đổi giá trị K khác nhau và thực hiện lại quy trình để tìm được K tối ưu nhất cho tập dữ liệu.

3.1.3.3. Ưu điểm, nhược điểm của KNN

- Ưu điểm:
  - Thuật toán đơn giản, dễ dàng triển khai.
  - Độ phức tạp tính toán nhỏ.
  - Xử lý tốt với tập dữ liệu nhiễu.
- Nhược điểm:
  - Với K nhỏ dễ gặp nhiễu dẫn tới kết quả đưa ra không chính xác.
  - Cần nhiều thời gian để thực hiện do phải tính toán khoảng cách với tất cả các đối tượng trong tập dữ liệu.
  - Cần chuyển đổi kiểu dữ liệu thành các yếu tố định tính.

3.2. Đánh giá mô hình phân lớp

3.2.1. Độ chính xác (Accuracy)

Khi xây dựng mô hình phân loại chúng ta sẽ muốn biết một cách khái quát tỷ lệ các trường hợp được dự báo đúng trên tổng số các trường hợp là bao nhiêu. Tỷ lệ đó được gọi là độ chính xác [12].

Độ chính xác giúp ta đánh giá hiệu quả dự báo của mô hình trên một bộ dữ liệu. Độ chính xác càng cao thì mô hình của chúng ta càng chuẩn xác.

Độ chính xác của mô hình trên một tập dữ liệu được tính bằng công thức:

- $\text{Accuracy} = \frac{\text{Số lượng điểm dữ liệu dự đoán chính xác}}{\text{Tổng số lượng điểm dữ liệu}}$

Nghịch lý của độ chính xác (Accuracy Paradox):

- Đối với các bài toán phân loại mà số lượng điểm dữ liệu trong các lớp mất cân bằng, độ đo chính xác không còn hiệu quả để đánh giá mô hình nữa.
- Nguyên nhân là nếu mô hình của ta luôn dự đoán ra lớp mà có số lượng điểm dữ liệu lớn nhất, thì độ chính xác luôn luôn ở mức cao.

3.2.2. Confusion Matrix

Trong các bài toán phân loại, ta thường gặp những bài toán mà:

- Ta cần quan tâm đánh giá một lớp dữ liệu nào đó hơn các lớp dữ liệu còn lại.
- Số lượng điểm dữ liệu trong các lớp dữ liệu bị mất cân bằng.

Ta thường áp dụng các khái niệm True/False Positive/Negative để đánh giá mô hình trong các bài toán trên.

Đối với các bài toán mà ta xem có một lớp quan trọng hơn các lớp còn lại. Ta có thể:

- Xem lớp quan trọng hơn là Positive.
- Các lớp còn lại gộp chung thành Negative.

Từ đó ta định nghĩa các thuật ngữ sau:

	Dự đoán là Positive	**Dự đoán là Negative**
Thực sự là Positive	True Positive	False Negative
Thực sự là Negative	False Positive	True Negative

Bảng 3.1. *True/False Positive/Negative*

Trong đó:

- True Positive (TP): số lượng điểm của lớp positive được phân loại đúng là positive.
- True Negative (TN): số lượng điểm của lớp negative được phân loại đúng là negative.
- False Positive (FP): số lượng điểm của lớp negative bị phân loại sai thành positive.
- False Negative (FN): số lượng điểm của lớp positive bị phân loại sai thành negative. [13]

Confusion Matrix là trường hợp tổng quát của True/False Positive/Negative. Để đánh giá độ hiệu quả của một mô hình bằng Confusion Matrix, ta thường:

- Xây dựng các tỉ lệ như độ chính xác, độ lỗi,…
- Chuyển Confusion Matrix về dạng True/False Positive/Negative. Sau đó thực hiện đánh giá trên từng lớp dữ liệu.

Ta định nghĩa các loại tỉ lệ sau:

- True Positive Rate – TPR = $\frac{TP}{TP+FN}$
- False Positive Rate – FPR = $\frac{FP}{FP+TN}$
- True Negative Rate – TNR = $\frac{TN}{TN+FP}$
- False Negative Rate – FNR = $\frac{FN}{FN+TP}$

Thông thường, ta chỉ quan tâm đến False Positive Rate và False Negative Rate, ý nghĩa của chúng là:

- False Positive Rate có ý nghĩa là tỉ lệ xác định nhầm lớp dữ liệu ta quan tâm.
- False Negative Rate có ý nghĩa là tỉ lệ bỏ sót lớp dữ liệu ta quan tâm.

3.2.3. Precision

Precision [14] được định nghĩa là tỉ lệ số điểm Positive mô hình dự đoán đúng trên tổng số điểm mô hình dự đoán là Positive.

Precision càng cao, tức là số điểm mô hình dự đoán là positive đều là positive càng nhiều. Precision = 1, tức là tất cả số điểm mô hình dự đoán là Positive đều

đúng, hay không có điểm nào có nhãn là Negative mà mô hình dự đoán nhầm là Positive.

Công thức của Precision như sau:

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

3.2.4. Recall

Recall [14] được định nghĩa là tỉ lệ số điểm Positive mô hình dự đoán đúng trên tổng số điểm thật sự là Positive (hay tổng số điểm được gán nhãn là Positive ban đầu).

Recall càng cao, tức là số điểm Positive bị bỏ sót càng ít. Recall = 1, tức là tất cả số điểm có nhãn là Positive đều được mô hình nhận ra.

Công thức tính Recall như sau:

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

3.2.5. F1 Score

F1 Score [14] là trung bình điều hoà giữa Precision và Recall.

Công thức tính F1 Score như sau:

- F1 Score = $\frac{2 \times Precision \times Recall}{Precision+Recall}$

F1 Score càng cao tương ứng Precision và Recall càng cao, mô hình phân loại càng tốt.

CHƯƠNG IV. ĐÁNH GIÁ HIỆU SUẤT MÔ HÌNH

Khi xây dựng một mô hình Machine Learning, chúng ta 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. Có rất nhiều cách đánh giá một mô hình phân lớp. Tuỳ vào những bài toán khác nhau mà chúng ta sử dụng các phương pháp khác nhau. Trong bài báo cáo này, chúng tôi sẽ sử dụng 5 độ đo là accuracy, confusion matrix, precision, recall, F1 score để đánh giá ba mô hình là Logistic Regression, Random Forest, K-Nearest Neighbors.

4.1. Huấn luyện mô hình

Trong phần này, chúng tôi sẽ trình bày các bước chính để xây dựng và huấn luyện các mô hình cho phương pháp học máy Logistic Regression, Random Forest, K-Nearest Neighbors.

4.1.1. Tiền xử lý

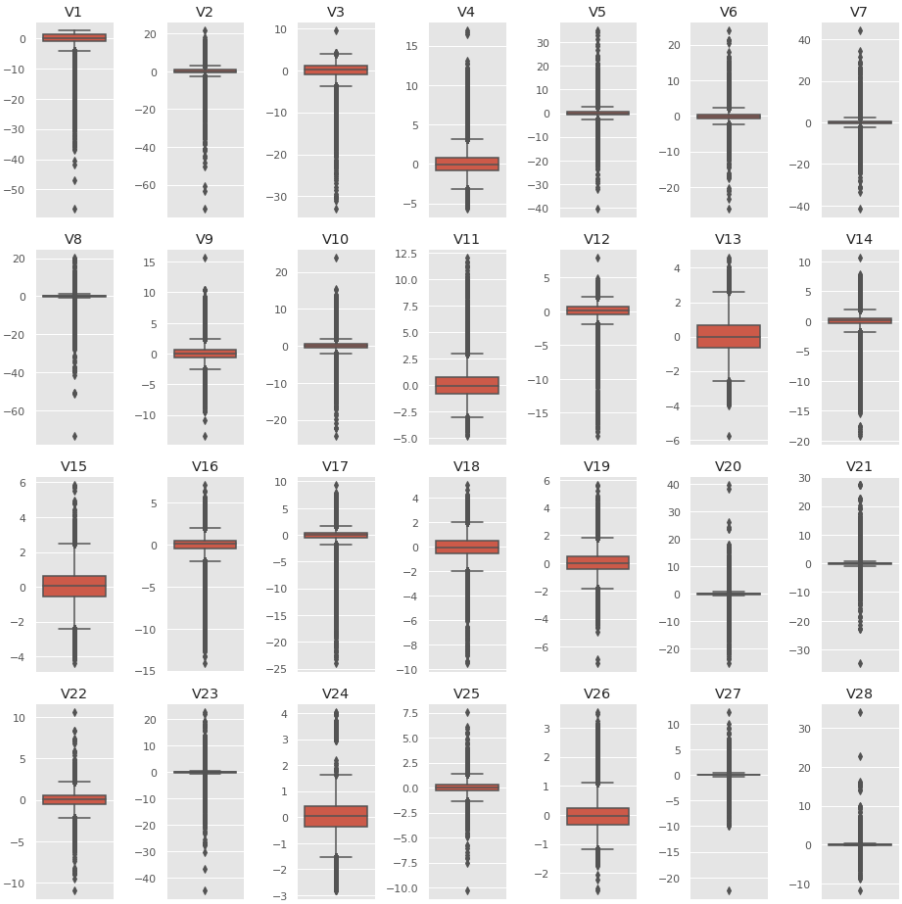
Đầu tiên, xác định hai biến X và Y. Trong đó, X là biến độc lập gồm các thuộc tính ‘Amount’, ‘Time’, ‘V1’,… ‘V28’; Y là biến phụ thuộc ‘Class’ chứa nhãn 0 hoặc 1. Sau đó, phân chia bộ dữ liệu thành tập huấn luyện (train) và tập kiểm thử (test) với tỉ lệ train:test = 8:2.

```
[ ]  x_train,x_test,y_train,y_test = train_test_split(x,y,random_state=42,test_size=0.2)

[ ]  x_train.shape, x_test.shape, y_train.shape

     ((227845, 30), (56962, 30), (227845,))
```

Hình 4.1. *Chia bộ dữ liệu với tỉ lệ train:test = 8:2*

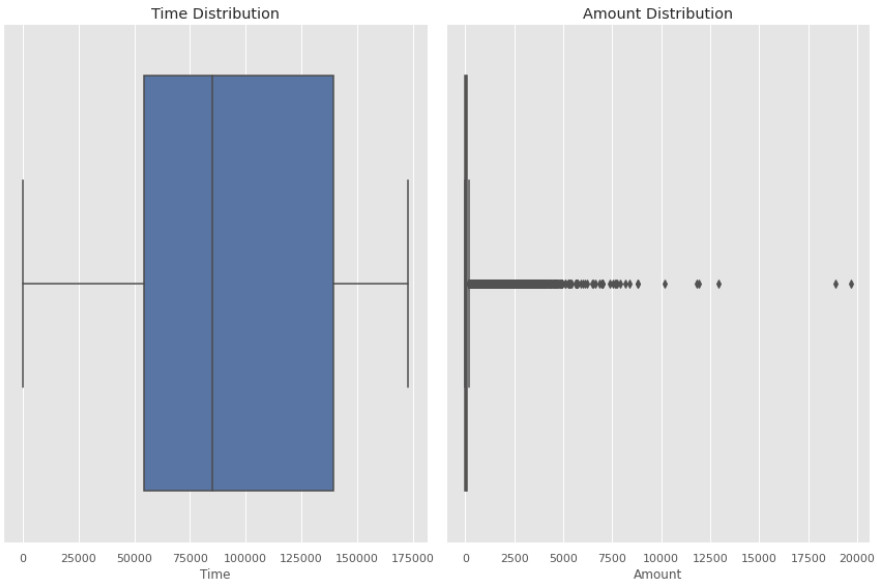
4.1.1.1. Chuẩn hoá dữ liệu

Tập X gồm 30 thuộc tính, trong đó 28 thuộc tính từ ‘V1’ đến ‘V28’ đã được biến đổi bằng thuật toán PCA.



Hình 4.2. *Phân phối của 28 thuộc tính từ V1 đến V28*

Như chúng ta có thể thấy, mỗi thuộc tính có sự phân bố khác nhau và nhiều thuộc tính xuất hiện outliers. Tất cả thuộc tính đều có giá trị tập trung vào 0 và mỗi thuộc tính có phạm vi giá trị khác nhau.



Hình 4.3. *Phân phối của thuộc tính Time và Amount*

Khác với ‘Time’, ‘Amount’ có nhiều outliers khi giá trị trung bình là 88 nhưng giá trị lớn nhất là hơn 25,000.

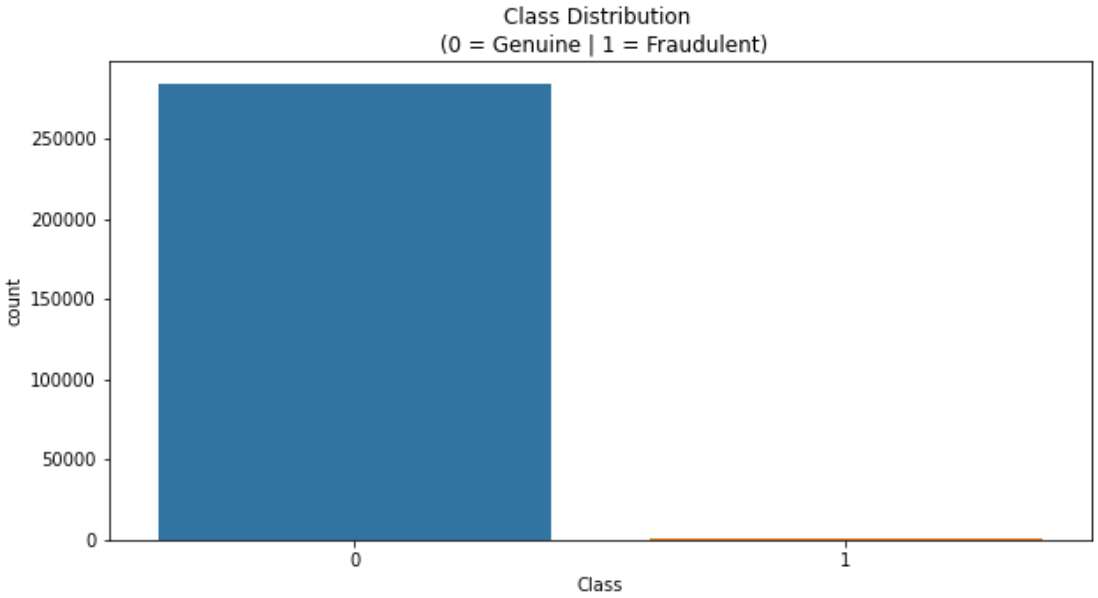
Ở phần này chúng tôi sẽ tiến hành chuẩn hóa 30 thuộc tính bao gồm cả các thuộc tính trên và hai thuộc tính còn lại là ‘Amount’ và ‘Time’. Mục đích là để đưa tất cả các thuộc tính về một quy mô tương tự, ngay cả khi chúng đã được chuẩn hóa trước đó.

```
from sklearn.preprocessing import RobustScaler
robust_scaler = RobustScaler()
X_train = robust_scaler.fit_transform(x_train)
X_test = robust_scaler.transform(x_test)
```

Hình 4.4. *Chuẩn hoá dữ liệu với RobustScaler*

4.1.1.2. Xử lý dữ liệu mất cân bằng

Như đã đề cập trước đó, bộ dữ liệu bị mất cân bằng nghiêm trọng với số lượng nhãn 1 (giao dịch gian lận) chỉ chiếm 0.172%.



Hình 4.5. *Hai class 0, 1 của bộ dữ liệu bị mất cân bằng nghiêm trọng*

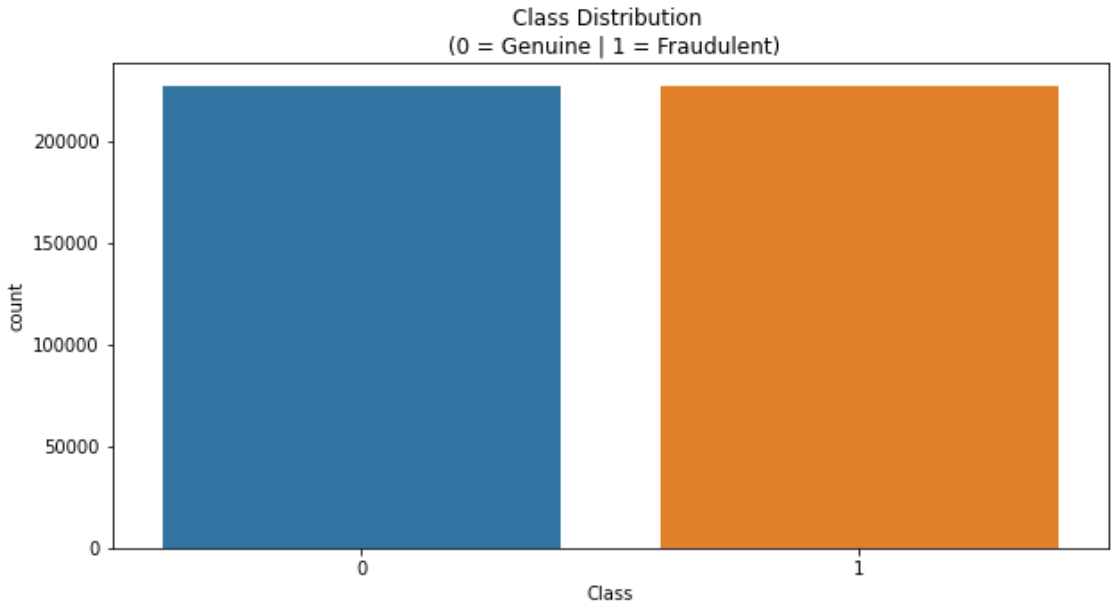
Điều này gây khó khăn đối với các mô hình dự đoán phân loại vì hầu hết các thuật toán học máy được sử dụng để phân loại được thiết kế dựa trên giả định về số lượng ví dụ bằng nhau cho mỗi nhãn. Điều này dẫn đến các mô hình có hiệu suất dự đoán kém, đặc biệt đối với nhãn thiểu số. Trong bài toán này, đây là một vấn đề cần xem xét vì giao dịch gian lận (lớp thiểu số) quan trọng hơn, do đó việc phân loại sai các giao dịch gian lận sẽ gây nhiều thiệt hại hơn giao dịch hợp pháp.

Để giải quyết vấn đề này, chúng tôi đề xuất kỹ thuật SMOTE (Synthetic Minority Oversampling Technique), một trong những cách tiếp cận phổ biến nhất để oversampling. Đây là một phương pháp tạo dữ liệu bằng cách tạo điểm dữ liệu mới của lớp thiểu số, cụ thể trong bài toán này là các điểm dữ liệu được gán nhãn '1'. SMOTE chọn ngẫu nhiên các điểm dữ liệu từ lớp thiểu số và sử dụng K Nearest Neighbors để tìm các điểm dữ liệu tương tự trong không gian đặc trưng. Sau đó, tạo một điểm dữ liệu tổng hợp tại điểm ở giữa điểm ngẫu nhiên và 'neighbor' gần nhất của nó [15].

```
smote = SMOTE(random_state=42)
X_train_re,Y_train_re  = smote.fit_resample(X_train, y_train)
```

Hình 4.6. *Cân bằng bộ dữ liệu với kĩ thuật SMOTE*

Sau khi áp dụng SMOTE, tập train từ 227845 điểm dữ liệu tăng lên thành 454902 điểm dữ liệu được chia đều cho hai nhãn, với 227451 điểm dữ liệu cho từng nhãn.



Hình 4.7. *Hai class 0, 1 trở nên cân bằng*

4.1.1.3. Tối ưu hoá mô hình

Một mô hình học máy có hai loại tham số. Loại tham số đầu tiên là các tham số (parameters) được học thông qua mô hình học máy, loại tham số thứ hai là các siêu tham số (hyperparameter) mà chúng ta truyền cho mô hình học máy. Điều chỉnh các siêu tham số này cho phù hợp với bộ dữ liệu có thể giúp các mô hình hoạt động tốt hơn. Tuy nhiên, rất khó để biết những giá trị nào phù hợp cho

các siêu tham số của một thuật toán nhất định trên một tập dữ liệu nhất định, do đó, người ta thường sử dụng các kỹ thuật tìm kiếm ngẫu nhiên hoặc tìm kiếm lưới cho các giá trị siêu tham số khác nhau [16]. Trong bước này, chúng tôi sẽ sử dụng thuật toán GridSearchCV trong thư viện sklearn.model_selection để đánh giá các mô hình cho các vectơ siêu tham số đã cho sử dụng cross-validation (xác thực chéo). Các tham số chúng tôi truyền vào thuật toán này bao gồm:

- Estimator: mô hình cần tối ưu hóa, trong trường hợp này là KNN, Logistic Regression và Random Forest.
- param_grid: các tham số cần khảo sát cho từng mô hình.
- Cv: số 'fold' được tạo để thực hiện cross-validation, trong trường hợp này là 10 cho cả 3 mô hình.
- Scoring: độ đo đánh giá hiệu suất mô hình, trong trường hợp này là 'accuracy' cho cả ba mô hình [17].

4.1.1.3.1. Logistic Regression

Đối với thuật toán Logistic Regression, chúng tôi quyết định khảo sát với ba tham số solver, penalty và C với các giá trị tương ứng:

- solver: 'newton-cg', 'lbfgs', 'sag', 'saga'
- penalty: 'none', 'l1', 'l2', 'elasticnet'
- C: '100', '10', '1.0', '0.1', '0.01'

Với các tham số như trên, khảo sát được 80 bộ tham số, với kết quả như bảng sau:

	mean_test_score	**params**
0	0.948508	{'C': 100, 'penalty': 'none', 'solver': 'newton-cg'}
1	0.948501	{'C': 100, 'penalty': 'none', 'solver': 'lbfgs'}
2	0.947384	{'C': 100, 'penalty': 'none', 'solver': 'sag'}
3	0.946687	{'C': 100, 'penalty': 'none', 'solver': 'saga'}

4		{‘C’: 100, ‘penalty’: ‘l1’, ‘solver’: ‘newton-cg’}
5		{‘C’: 100, ‘penalty’: ‘l1’, ‘solver’: ‘lbfgs’}
6		{‘C’: 100, ‘penalty’: ‘l1’, ‘solver’: ‘sag’}
…	….	…
25	***0.948512***	***{‘C’: 10, ‘penalty’: ‘l2’, ‘solver’: ‘lbfgs’}***
26	0.947384	{‘C’: 10, ‘penalty’: ‘l2’, ‘solver’: ‘sag’}
27	0.946685	{‘C’: 10, ‘penalty’: ‘l2’, ‘solver’: ‘saga’}
…	…	…
75	0.94656	{‘C’: 0.01, ‘penalty’: ‘l2’, ‘solver’: ‘saga’}
76		{‘C’: 0.01, ‘penalty’: ‘elasticnet’, ‘solver’: ‘newton-cg’}
77		{‘C’: 0.01, ‘penalty’: ‘elasticnet’, ‘solver’: ‘lbfgs’}
78		{‘C’: 0.01, ‘penalty’: ‘elasticnet’, ‘solver’: ‘sag’}
79		{‘C’: 0.01, ‘penalty’: ‘elasticnet’, ‘solver’: ‘saga’}

Bảng 4.1. *Hiệu suất của các bộ tham số - LR*

Với mean_test_score là hiệu suất trung bình của mô hình theo thang đo “accuracy”.

Lưu ý: Có những tham số không thể kết hợp với nhau, do đó không thể tính toán giá trị mean_test_score cũng như huấn luyện.

Từ bảng trên, ta thấy bộ tham số cho kết quả tốt nhất là: ‘C’: 10, ‘penalty’: ‘l2’, ‘solver’: ‘lbfgs’ với hiệu suất là 0.948512.

4.1.1.3.2. KNN

Đối với thuật toán KNN, chúng tôi quyết định khảo sát với hai tham số n_neighbors và weights với các giá trị tương ứng:

- n_neighbors: ‘1’, ‘2’, ‘3’,…‘20’
- weight_options: ‘uniform’, ‘distance’

Với các tham số như trên, khảo sát được 40 bộ tham số, với kết quả như bảng sau:

	mean_test_score	**params**
0	0.980322	{'n_neighbors': 1, 'weights': 'distance'}
1	0.980451	{'n_neighbors': 2, 'weights': 'distance'}
2	0.980541	{'n_neighbors': 3, 'weights': 'distance'}
3	0.980167	{'n_neighbors': 4, 'weights': 'distance'}
4	0.980525	{'n_neighbors': 5, 'weights': 'distance'}
5	0.971982	{'n_neighbors': 6, 'weights': 'distance'}
…	…	…
14	0.980166	{'n_neighbors': 15, 'weights': 'distance'}
15	0.961354	{'n_neighbors': 16, 'weights': 'distance'}
16	0.980154	{'n_neighbors': 17, 'weights': 'distance'}
17	0.980023	{'n_neighbors': 18, 'weights': 'distance'}
18	0.979723	{'n_neighbors': 19, 'weights': 'distance'}
19	0.980481	{'n_neighbors': 20, 'weights': 'distance'}
20	***0.980547***	***{'n_neighbors': 1, 'weights': 'uniform'}***
21	0.968395	{'n_neighbors': 2, 'weights': 'uniform'}
22	0.970818	{'n_neighbors': 3, 'weights': 'uniform'}
23	0.980439	{'n_neighbors': 4, 'weights': 'uniform'}
24	0.980104	{'n_neighbors': 5, 'weights': 'uniform'}
25	0.979512	{'n_neighbors': 6, 'weights': 'uniform'}

Bảng 4.2. *Hiệu suất của các bộ tham số - KNN*

Với mean_test_score là hiệu suất trung bình của mô hình theo thang đo “accuracy”.

Từ bảng trên ta thấy bộ tham số cho kết quả tốt nhất là: 'n_neighbors': 1, 'weights': 'uniform'.

4.1.1.3.3. Random Forest

Đối với thuật toán Random Forest, chúng tôi quyết định khảo sát với bốn tham số class_weight, solver, penalty và C với các giá trị tương ứng:

- criterion: ‘entropy’, ‘gini’
- max_features: ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’, ‘9’, ‘10’
- n_estimators: ‘10’, ‘100’, ‘1000’
- class_weight: ‘balanced’, ‘balanced_subsample’

Với các tham số như trên, khảo sát được 48 bộ tham số với kết quả như bảng sau:

	mean_test_score	**params**
0	0.997549	{‘class_weight’: ‘balanced’, ‘criterion’: ‘entropy’, ‘max_features’: ‘auto’, ‘n_estimators’: 1}
1	0.998496	{‘class_weight’: ‘balanced’, ‘criterion’: ‘entropy’, ‘max_features’: ‘auto’, ‘n_estimators’: 2}
2	0.999624	{‘class_weight’: ‘balanced’, ‘criterion’: ‘entropy’, ‘max_features’: ‘auto’, ‘n_estimators’: 3}
3	0.999752	{‘class_weight’: ‘balanced’, ‘criterion’: ‘entropy’, ‘max_features’: ‘auto’, ‘n_estimators’: 4}
4	0.997837	{‘class_weight’: ‘balanced’, ‘criterion’: ‘entropy’, ‘max_features’: ‘sqrt’, ‘n_estimators’: 1}
5	0.998371	{‘class_weight’: ‘balanced’, ‘criterion’: ‘entropy’, ‘max_features’: ‘sqrt’, ‘n_estimators’: 2}
…	…	…
35	***0.999769***	***{‘class_weight’: ‘balanced_subsample’, ‘criterion’: ‘entropy’, ‘max_features’: ‘log2’, ‘n_estimators’: 4}***
36	0.997294	{‘class_weight’: ‘balanced_subsample’, ‘criterion’: ‘gini’, ‘max_features’: ‘auto’, ‘n_estimators’: 1}

37	0.998147	{‘class_weight’: ‘balanced_subsample’, ‘criterion’: ‘gini’, ‘max_features’: ‘auto’, ‘n_estimators’: 2}
42	0.999618	{‘class_weight’: ‘balanced_subsample’, ‘criterion’: ‘gini’, ‘max_features’: ‘sqrt’, ‘n_estimators’: 3}
43	0.999666	{‘class_weight’: ‘balanced_subsample’, ‘criterion’: ‘gini’, ‘max_features’: ‘sqrt’, ‘n_estimators’: 4}
44	0.997283	{‘class_weight’: ‘balanced_subsample’, ‘criterion’: ‘gini’, ‘max_features’: ‘log2’, ‘n_estimators’: 1}
45	0.998079	{‘class_weight’: ‘balanced_subsample’, ‘criterion’: ‘gini’, ‘max_features’: ‘log2’, ‘n_estimators’: 2}
46	0.999580	{‘class_weight’: ‘balanced_subsample’, ‘criterion’: ‘gini’, ‘max_features’: ‘log2’, ‘n_estimators’: 3}
47	0.999635	{‘class_weight’: ‘balanced_subsample’, ‘criterion’: ‘gini’, ‘max_features’: ‘log2’, ‘n_estimators’: 4}

Bảng 4.3. *Hiệu suất của các bộ tham số - RF*

Từ bảng trên, ta thấy bộ tham số cho kết quả tốt nhất là: ‘class_weight’: ‘balanced_subsample’,‘criterion’: ‘entropy’, ‘max_features’: ‘log2’, ‘n_estimators’: 4 với hiệu suất là 0.999769.

4.1.2. Huấn luyện mô hình

Sau khi đã chọn được bộ tham số cho kết quả tốt nhất, chúng tôi tiến hành huấn luyện các mô hình sử dụng bộ dữ liệu đã được tiền xử lý trước đó.

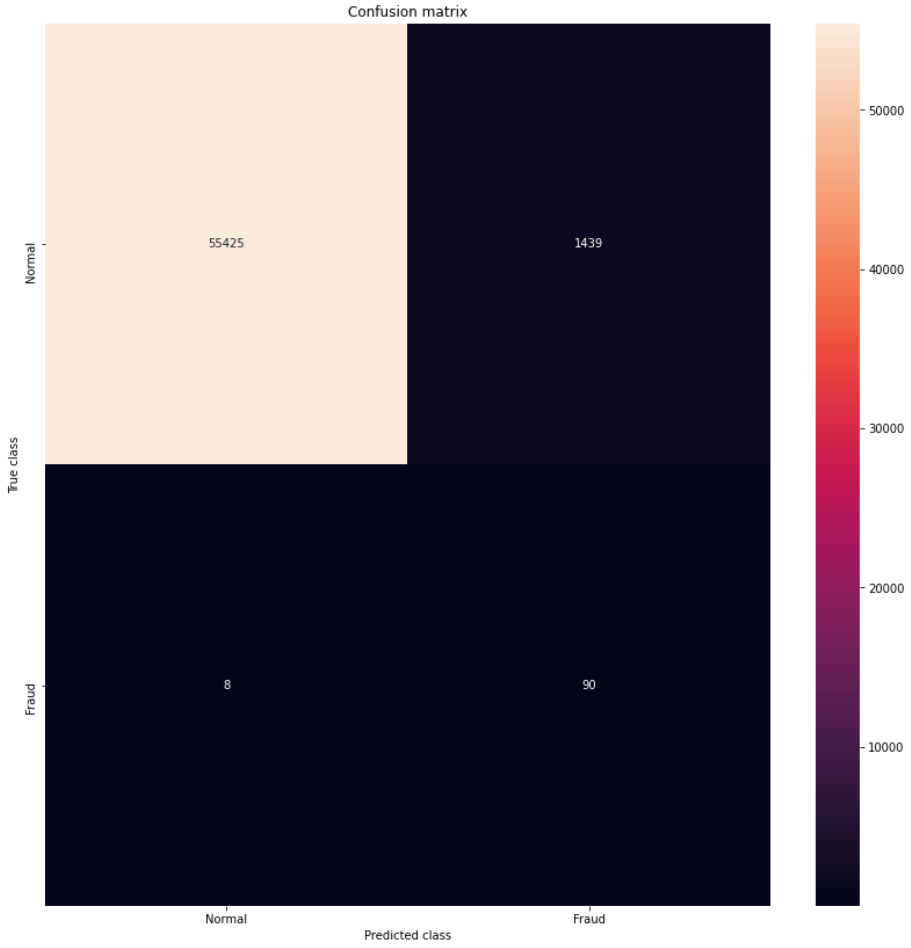
Hình 4.8. *Huấn luyện 3 mô hình LR, RF, KNN*

```
clf = LogisticRegression(C = 10, penalty = 'l2', solver = 'lbfgs')
clf.fit(X_train,y_train)

clf = RandomForestClassifier(class_weight = 'balanced_subsample', criterion = 'entropy', max_features = 'log2', n_estimators = 4)
clf.fit(X_train,y_train)

clf = KNeighborsClassifier(n_neighbors = 1, weights = 'uniform')
clf.fit(X_train,y_train)
```

4.2. Đánh giá hiệu suất mô hình

4.2.1. Logistic Regression

4.2.1.1. Confusion matrix và accuracy



Hình 4.9. *Confusion matrix trên tập Test – LR*

Theo ma trận trên, ta thấy:

- Số lượng điểm dữ liệu phân loại đúng là 55425 + 90 = 55515 điểm dữ liệu.
- Số lượng điểm dữ liệu phân loại sai là 1439 + 8 = 1447 điểm dữ liệu.
- Tỉ lệ điểm dữ liệu phân loại đúng là 55515/56962 = 0.98
- Số lượng điểm dữ liệu thực sự là giao dịch gian lận và được dự đoán là giao dịch hợp pháp là 8 điểm dữ liệu.
- Số lượng điểm dữ liệu thực sự là giao dịch hợp pháp và được dự đoán là giao dịch gian lận là 1439.

4.2.1.2. Classification report

	Precision	Recall	F1 score	Support
0 – Normal	1.00	0.97	0.99	56864
1 – Fraud	0.06	0.92	0.11	98
Avg/total	0.53	0.95	0.55	56962

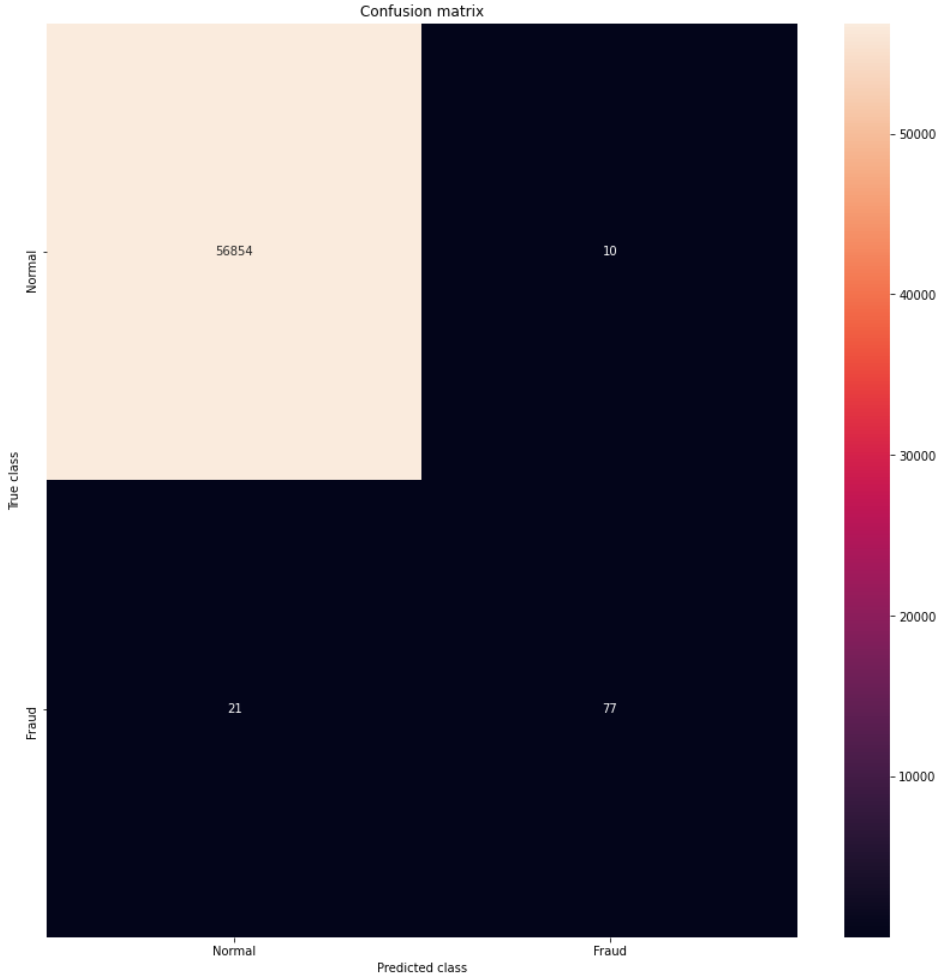
Bảng 4.4. *Classification report trên tập Test – LR*

Đối với mô hình Logistic Regression, ta nhận thấy:

- Mô hình phân loại tốt đối với nhãn "0" với kết quả F1 Score là 99%.
- Quan sát hiệu suất của nhãn "1" (được xem là nhãn quan trọng trong bài toán), thấy được kết quả Precision rất nhỏ (6%) nếu đem so với Recall (92%), dẫn đến kết quả F1 Score của nhãn "1" (11%) rất thấp.
- Từ kết quả precision, recall và F1 Score của mô hình sử dụng Logistic Regression, có thể thấy tuy mô hình bỏ sót khá ít các trường hợp giao dịch gian lận nhưng đã dự đoán sai khá nhiều trường hợp giao dịch hợp pháp thành giao dịch gian lận dẫn đến kết quả F1 score trung bình của mô hình (55%) ở mức trung bình, không cao.

4.2.2. Random Forest

4.2.2.1. Confusion matrix và accuracy



Hình 4.10. *Confusion matrix trên tập Test – RF*

Theo ma trận trên, ta thấy:

- Số lượng điểm dữ liệu phân loại đúng là 56854 + 77 = 56931 điểm dữ liệu.
- Số lượng điểm dữ liệu phân loại sai là 21 + 10 = 31 điểm dữ liệu.
- Tỉ lệ điểm dữ liệu phân loại đúng là 56931/56962 = 0.99
- Số lượng điểm dữ liệu thực sự là giao dịch gian lận và được dự đoán là giao dịch hợp pháp là 21 điểm dữ liệu.
- Số lượng điểm dữ liệu thực sự là giao dịch hợp pháp và được dự đoán là giao dịch gian lận là 10 điểm dữ liệu.

4.2.2.2. Classification Report

	Precision	Recall	F1 score	Support
0 – Normal	1.00	1.00	1.00	56864
1 – Fraud	0.89	0.79	0.83	98
Avg/total	0.94	0.89	0.92	56962

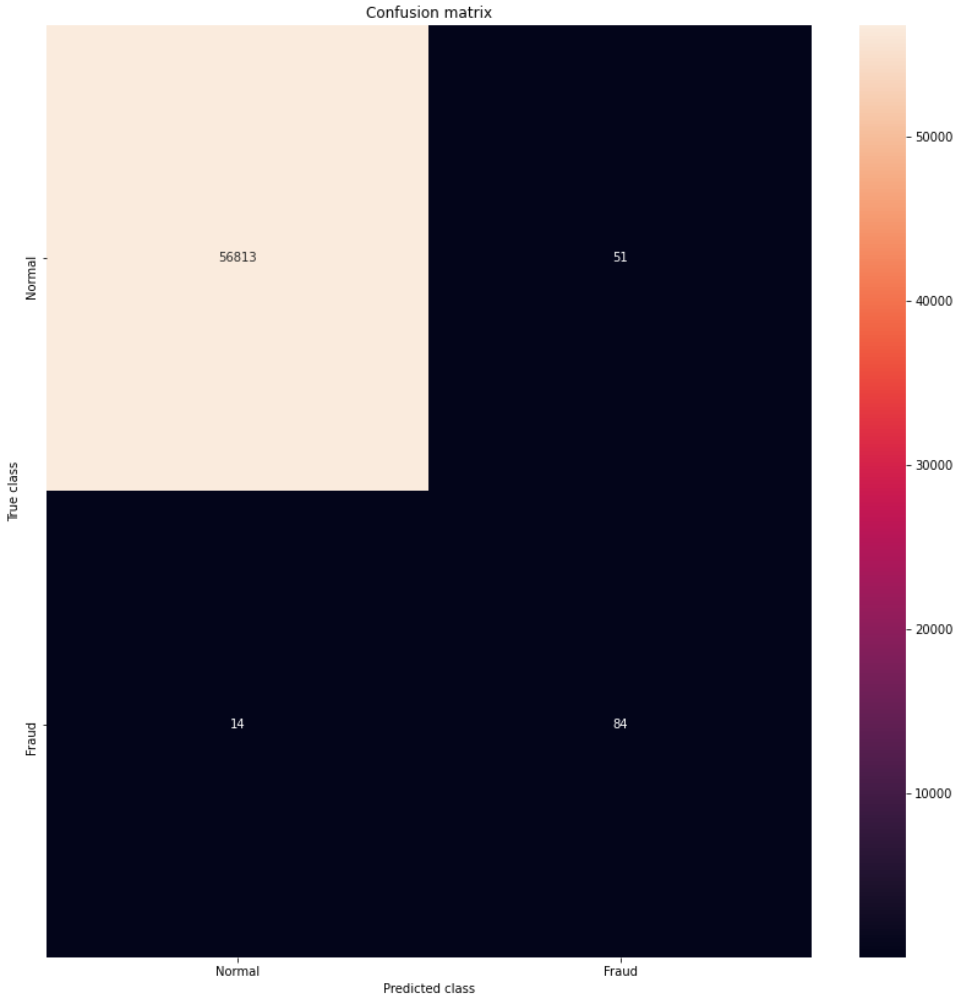
Bảng 4.5. *Classification report trên tập Test – RF*

Đối với mô hình Random Forest, ta nhận thấy:

- Mô hình phân loại rất tốt với nhãn "0" với kết quả Precision, Recall và F1 Score đều là 100%. Đồng thời, với nhãn "1" kết quả cũng tương đối tốt với kết quả F1 Score là 83%, dẫn đến kết quả F1 Score trung bình lên tới 92%.
- Quan sát hiệu suất của nhãn "1" (được xem là nhãn quan trọng trong bài toán), thấy được kết quả Precision (89%) và Recall (79%) tương đối tốt, dẫn đến kết quả F1 Score của nhãn "1" là 83%.

4.2.3. K-Nearest Neighbors

4.2.3.1. Confusion matrix và accuracy



Hình 4.11. *Confusion matrix trên tập Test – KNN*

Theo ma trận trên, ta thấy:

- Số lượng điểm dữ liệu phân loại đúng là 56813 + 84 = 56897 điểm dữ liệu.
- Số điểm dữ liệu phân loại sai là 51 + 14 = 65 điểm dữ liệu.
- Tỉ lệ điểm dữ liệu phân loại đúng là 56897/56962 = 0.99.
- Số lượng điểm dữ liệu thực sự là giao dịch gian lận và được dự đoán là giao dịch hợp pháp là 14 điểm dữ liệu.
- Số lượng điểm dữ liệu thực sự là giao dịch hợp pháp và được dự đoán là giao dịch gian lận là 51 điểm dữ liệu.

4.2.3.2. Classification Report

	Precision	Recall	F1 score	Support
0 – Normal	1.00	1.00	1.00	56864
1 – Fraud	0.62	0.86	0.72	98
Avg/total	0.81	0.93	0.86	56962

Bảng 4.6. *Classification trên tập Test – KNN*

Đối với mô hình KNN, ta nhận thấy:

- Mô hình phân loại rất tốt với nhãn "0" với kết quả Precision, Recall và F1 Score đều là 100%.
- Quan sát hiệu suất của nhãn "1" (được xem là nhãn quan trọng trong bài toán), thấy được kết quả Precision (62%), chỉ trên mức trung bình một chút và Recall (86%) tương đối tốt, dẫn đến kết quả F1 Score của nhãn "1" là 72%.

CHƯƠNG V. KẾT LUẬN

STT	Model	Precision	Recall	F1 Score	Accuracy
1	**Logistic Regression**	0.53	0.95	0.55	0.98
2	**Random Forest**	0.94	0.89	0.92	0.99
3	**K-Nearest Neighbors**	0.81	0.93	0.86	0.99

Bảng 5.1. *Bảng so sánh kết quả độ đo của các mô hình*

Trong bài báo cáo này, chúng tôi đã đánh giá nhiều mô hình khác nhau trên tập dữ liệu giao dịch thẻ tín dụng rất mất cân đối. Dữ liệu không cân bằng khiến các mô hình học máy khó phát hiện chính xác lớp thiểu số (trong bài toán này là các giao dịch gian lận), vì các mô hình có xu hướng dự đoán lớp đa số. Trong bài toán này, chúng tôi nhận thấy việc phân loại những trường hợp gian lận thành hợp pháp sẽ gây nên nhiều thiệt hại tài chính hơn việc phân loại những trường hợp hợp pháp thành gian lận.

Để giải quyết sự mất cân bằng dữ liệu, chúng tôi đã lấy mẫu dữ liệu của mình bằng một phương pháp được gọi là Synthetic Minority Oversampling Technique (SMOTE). Chúng tôi nhận thấy rằng SMOTE đã giúp các mô hình của chúng tôi hoạt động tốt hơn khi phân loại các giao dịch là gian lận, vì các mô hình có nhiều dữ liệu hơn trong lớp này.

Khi đánh giá các mô hình trên bộ thử nghiệm, chúng tôi nhận thấy rằng mô hình RandomForest của chúng tôi được điều chỉnh bằng GridSearchCV có hiệu suất tốt nhất theo độ đo F-1 Score (92%), precision trung bình cao (94%) và recall trung bình tương đối (89%). Theo như kết quả trước đó, mô hình này chỉ có thể phân loại chính xác 79% các giao dịch gian lận là gian lận và phân loại chính xác gần 100% các giao dịch bình thường là bình thường. Điều này có nghĩa là mô hình đã bỏ lỡ 21% các giao dịch thực sự là gian lận, làm cho mô hình kém hiệu quả.

Trong khi đó, mô hình KNN có hiệu suất F-1 Score đứng thứ hai (86%), nhưng lại có hiệu suất recall trung bình (93%) cao, với recall của nhãn ‘1’ là 86%, nghĩa là mô hình có thể dự đoán chính xác 86% và dự đoán sai 14% các giao dịch giao gian lận, nhưng cũng đồng thời dự đoán sai khá nhiều giao dịch hợp pháp thành gian lận khi precision chỉ trên mức trung bình (62%).

Mặt khác, mô hình Logistic Regression tuy có hiệu suất recall của nhãn ‘1’ cao nhất trong ba mô hình (92%), nhưng lại có hiệu suất precision rất thấp (6%), dẫn đến F-1 Score trung bình khá thấp (55%). Mô hình có thể dự đoán chính xác 92% giao dịch gian lận là gian lận, nhưng lại dự đoán sai khá nhiều giao dịch hợp pháp thành gian lận.

Trong bài toán này, chúng tôi nhận thấy việc phân loại những trường hợp gian lận thành hợp pháp sẽ gây nên nhiều thiệt hại tài chính hơn việc phân loại những trường hợp hợp pháp thành gian lận, tuy nhiên chúng tôi cũng xem xét mức độ cân bằng khi đánh giá việc phân loại các giao dịch gian lận thành gian lận và hợp pháp thành gian lận. Xét trên tiêu chí đó, chúng tôi đánh giá mô hình mô hình Random Forest là phù hợp nhất cho bài toán này.

Tóm lại, chúng tôi nhận thấy rằng mô hình Random Forest hoạt động tốt nhất, và hoạt động tối ưu hơn khi được đào tạo trên dữ liệu đã được xử lý SMOTE, so sánh với hiệu suất đào tạo theo thang đo F-1 Score trên dữ liệu thô là 89%. Có thể thấy SMOTE là một kỹ thuật rất hữu ích để làm việc với các tập dữ liệu không cân bằng.

TÀI LIỆU THAM KHẢO

[1] "Credit card fraud scenarios", Fraud-detection-handbook.github.io, 2021. [Online]. Available: https://fraud-detection-handbook.github.io/fraud-detection-handbook/Chapter_2_Background/CreditCardFraud.html.

[2] "Credit card transaction fraud continues to climb to new heights", NCR, 2021. [Online]. Available: https://www.ncr.com/blogs/payments/credit-card-fraud-detection.

[3] N. K. Trivedi, S. Simaiya, D. U. Kumar, S. K. Sharma, "An Efficient Credit Card Fraud Detection Model Based on Machine Learning Methods", ResearchGate, 2020. [Online]. Available: https://www.researchgate.net/publication/341932015_An_Efficient_Credit_Card_Fraud_Detection_Model_Based_on_Machine_Learning_Methods.

[4] GeeksforGeeks, "ML | Credit Card Fraud Detection", GeeksforGeeks, 2020. [Online]. Available: https://www.geeksforgeeks.org/ml-credit-card-fraud-detection/.

[5] R. Shakya, "Application of Machine Learning Techniques in Credit Card Fraud Detection", Digital Scholarship@UNLV, 2018. [Online]. Available: https://digitalscholarship.unlv.edu/cgi/viewcontent.cgi?article=4457&context=thesesdissertations.

[6] N. Khare, S. Y. Sait, "Credit Card Fraud Detection Using Machine Learning Models and Collating Machine Learning Models", Acadpubl, 2018. [Online]. Available: https://acadpubl.eu/hub/2018-118-21/articles/21b/90.pdf.

[7] "Tổng quan về Logistic regression (hồi quy Logistic) (Phần 2)", Big Data Uni, 2021. [Online]. Available: https://bigdatauni.com/tin-tuc/tong-quan-ve-logistic-regression-hoi-quy-logistic-phan-2.html.

[8] JavaTpoint, "Machine Learning Random Forest Algorithm", JavaTpoint. [Online]. Available: https://www.javatpoint.com/machine-learning-random-forest-algorithm.

[9] A. Chakure, "Random Forest and Its Implementation", Medium, 2019. [Online]. Available: https://medium.com/swlh/random-forest-and-its-implementation-71824ced454f.

[10] N. T. Hop, "KNN (K-Nearest Neighbors) #1", VIBLO, 2019. [Online]. Available: https://viblo.asia/p/knn-k-nearest-neighbors-1-djeZ14ejKWz.

[11] H. Jaroli, "K-Nearest Neighbors (KNN) with Python", DataScience+, 2019. [Online]. Available: https://datascienceplus.com/k-nearest-neighbors-knn-with-python/.
[12] P. Khánh, “Bài 46 - Đánh giá mô hình phân loại trong ML”, Khanh's blog, 2020. [Online]. Available: https://phamdinhkhanh.github.io/2020/08/13/ModelMetric.html.
[13] thuynt, "Các phương pháp đánh giá một mô hình phân lớp dữ liệu", HocTrucTuyen123, 2018. [Online]. Available: http://hoctructuyen123.net/cac-phuong-phap-danh-gia-mot-mo-hinh-phan-lop-du-lieu/.
[14] V. caihuuthuc, "Precision, Recall và F1-score là gì?", Cái Hữu Thức's notes, 2020. [Online]. Available: https://caihuuthuc.wordpress.com/2020/02/23/precision-recall-va-f1-score-la-gi/.
[15] J. Brownlee, "SMOTE for Imbalanced Classification with Python", Machine Learning Mastery, 2021. [Online]. Available: https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/.
[16] J. Brownlee, "Tune Hyperparameters for Classification Machine Learning Algorithms", Machine Learning Mastery, 2021. [Online]. Available: https://machinelearningmastery.com/hyperparameters-for-classification-machine-learning-algorithms/.
[17] U. Malik, "Cross Validation and Grid Search for Model Selection in Python", Stack Abuse, 2021. [Online]. Available: https://stackabuse.com/cross-validation-and-grid-search-for-model-selection-in-python.