VIỆN NGHIÊN CỨU DỮ LIỆU LỚN VINBIGDATA
Chương trình đào tạo kỹ sư AI Vingroup
Khóa 3

BÁO CÁO DỰ ÁN CUỐI KHÓA
MÔN HỌC MÁY
NỘI DUNG: PHAT HIỆN GIAO DỊCH LỪA ĐẢO

Ngày 20 tháng 11 năm 2023

Nhóm 8

1. Nguyễn Nhật Quang - 3733351
2. Nguyễn Trung Kiên - 3733357
3. Đặng Xuân Lộc - 3733353
4. Nguyễn Bảo Sơn - 3733349
5. Phạm Hoàng Tùng - 3733355

Mục lục

1 Giới thiệu bài toán và phân công công việc

1.1 Bối cảnh

Lừa đảo trong thanh toán điện tử đã và đang là một trong những vấn đề lớn, gây thiệt hại cho các khách hàng và cả những đơn vị hỗ trợ thanh toán. Để đối phó với những chiêu trò lừa đảo này, các ngân hàng đã nghĩ ra rất nhiều các phương án để đối phó. Tuy nhiên, theo thời gian, những thủ đoạn lừa đảo ngày càng tinh vi hơn. Vì thế, những ngân hàng cần phải nảy thêm những ý tưởng mới, phương pháp mới để ngăn chặn những vụ lừa đảo này.

Thời đại của trí tuệ nhân tạo (AI) tạo cơ hội để phát hiện và ngăn chặn những hành vi lừa đảo này, dựa trên sự mạnh mẽ và chính xác của các phương pháp tiên tiến để học được những đặc trưng của những giao dịch lừa đảo.

1.2 Mục tiêu

Những phương pháp phát hiện giao dịch lừa đảo truyền thống thường yêu cầu sự can thiệp trực tiếp của ngân hàng với những người chủ sở hữu thẻ. Tuy nhiên, đây là những phương pháp rất tốn kém và không hiệu quả, vì để bắt được những giao dịch lừa đảo cần tốn rất nhiều thời gian và nguồn lực. Hơn nữa, đa phần những giao dịch vẫn là những giao dịch hợp lệ, và chỉ có một số rất ít mới là giao dịch gian lận và việc phát hiện những thanh toán này còn tốn kém hơn nữa.

Mục tiêu của dự án này của chúng tôi là xây dựng một bộ phân loại những giao dịch chính thống và lừa đảo, sử dụng những phương pháp học máy (Machine Learning) trên một tập dữ liệu e-commerce thực. Đây là một bài toán phân loại nhị phân (binary classification) trong học máy.

1.3 Ràng buộc

- Những dự đoán của mô hình cần phải chính xác: Dự đoán một giao dịch hợp lệ là lừa đảo sẽ dẫn đến trải nghiệm người dùng tiêu cực và khiến ngân hàng mất đi số lượng khách hàng lớn, và dự đoán một giao dịch lừa đảo là hợp lệ sẽ rất nguy hiểm và dẫn đến những tổn thất rất lớn. Tuy nhiên, vì mong muốn phát hiện những giao dịch lừa đảo chính xác, **cái giá (cost) phải trả khi phát hiện nhầm giao dịch lừa đảo là hợp lệ (false negative) sẽ lớn hơn là phát hiện nhầm giao dịch hợp lệ là lừa đảo (false positive)**.
- Giả sử một giao dịch bị phát hiện là lừa đảo, nhưng người dùng phải mấy ngày sau mới biết được điều đó. Ta không muốn điều này xảy ra và vì thế, mô hình phát hiện lừa đảo phải dự đoán được **ngay lập tức** loại giao dịch này.
- **Sự giải thích được (interpretability)** của mô hình cũng cần phải được ưu tiên. Một người bị mô hình dự đoán là đã thực hiện một giao dịch không chính đáng cần phải biết tại sao mô hình đưa ra quyết định như vậy.

1.4 Phân công công việc

Tất cả các thành viên trong nhóm đều đã tham gia chia sẻ ý kiến của mình vào xây dựng và viết các mục dưới đây trong bản báo cáo. Ngoài ra, nhóm chúng tôi đã phân chia dự án này thành những khối công việc chính với những người đảm nhận như sau:

- **Nguyễn Trung Kiên**: Phụ trách EDA và tiền xử lý các cột TransactionDT, M1-M9, D1-D15; tìm hiểu và huấn luyện mô hình LightGBM.

- **Nguyễn Nhật Quang**: Phụ trách EDA và tiền xử lý các cột ProductCD, card1-card6, addr1, addr2, P_emaildomain, R_emaildomain; tìm hiểu và huấn luyện các mô hình AdaBoost và GradientBoosting.
- **Nguyễn Bảo Sơn**: Phụ trách EDA và tiền xử lý các cột TransactionAmt, dist1, dist2, C1-C14; tìm hiểu và huấn luyện các mô hình DecisionTree và RandomForest.
- **Đặng Xuân Lộc**: Phụ trách EDA và tiền xử lý các cột trong bảng định danh (Identity Table); tìm hiểu và huấn luyện các mô hình XGBoost, CatBoost.
- **Phạm Hoàng Tùng**: Phụ trách EDA và tiền xử lý các cột V1 đến V339; tìm hiểu và huấn luyện các mô hình Naive Bayes, Logistic Regression, CatBoost.

Phần code của dự án này có thể được tìm thấy tại link sau:
`https://github.com/DXL64/VinBigData-Fraud-Detection/tree/main`

2 Tập dữ liệu

2.1 Dữ liệu và mô tả các thuộc tính

Dữ liệu được lấy từ cuộc thi **IEEE-CSS Fraud Detection** trên Kaggle, được cung cấp bởi Vesta Corporation - một trong những công ty hàng đầu thế giới về lĩnh vực thanh toán điện tử. Dữ liệu gồm có 2 file: **train_transaction.csv** và **train_identity.csv**, được kết nối (join) với nhau bởi cột (thuộc tính) TransactionID. Khi học, chúng ta sẽ kết hợp (left join) hai dữ liệu bảng này với nhau để được một bảng dữ liệu lớn, gồm 590540 dòng và 434 cột.

Mục tiêu của chúng tôi là từ một giao dịch với các thuộc tính cho sẵn, phải xây dựng một mô hình dự đoán chính xác được nhãn (isFraud) - tượng trưng cho sự hợp lệ hay lừa đảo - của quan sát đó

Lưu ý, một vài thuộc tính trong tập dữ liệu này đã bị chính Vesta - nhà cung cấp dữ liệu này - ẩn đi ý nghĩa để bảo mật thông tin của người dùng theo hợp đồng. Sau đây là những thuộc tính và những ý nghĩa chung của nó theo Vesta.

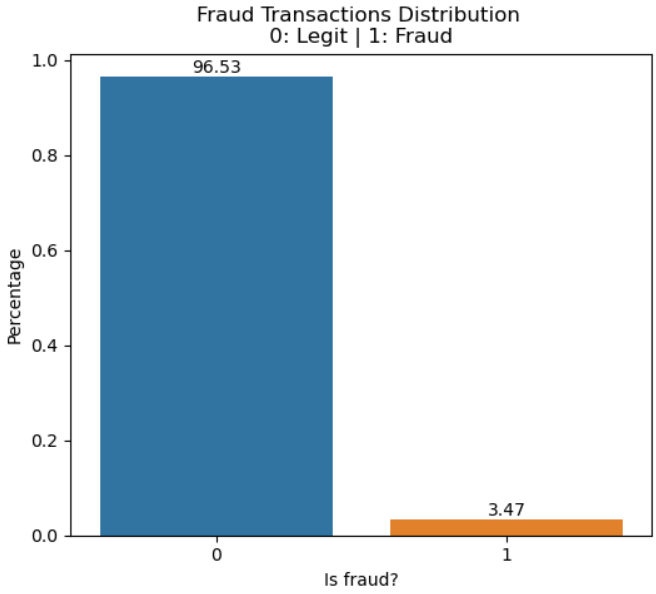
- TransactionID: Số định danh (ID) của giao dịch.
- isFraud: Tính hợp lệ của giao dịch, nhận hai giá trị là 0 (hợp lệ) và 1 (lừa đảo). Đây cũng là nhãn (target) của bài toán và tập dữ liệu.
- TransactionDT: thời gian diễn ra giao dịch, tính bằng khoảng thời gian trôi qua từ một mốc thời gian cố định.
- TransactionAMT: Số tiền được thanh toán trong giao dịch, tính bằng USD.
- ProductCD: Mã sản phẩm.
- card1-card6: Các thông tin liên quan đến thẻ thanh toán như loại thẻ, ngân hàng, quốc gia...
- addr1, addr2: Các cột giữ liệu theo vùng miền và theo quốc gia.
- dist1, dist2: Các cột dữ liệu về khoảng cách.
- P_emaildomain: Đuôi email của người mua.
- R_emaildomain: Đuôi email của người nhận.
- C1-C14: những thông số dữ liệu đếm, ví dụ như đếm bao nhiêu địa chỉ được liên quan đến một thẻ thanh toán... Ý nghĩa thực sự của các cột đã bị ẩn đi.
- D1-D15: Các cột dữ liệu về khoảng thời gian, ví dụ như khoảng thời gian giữa hai giao dịch.
- M1-M9: Các cột dữ liệu về sự tương khớp (True or False), ví dụ như giữa thông tin trên thẻ và địa chỉ của chủ thẻ...
- V1-V339: Các cột dữ liệu không rõ nghĩa được cung cấp bởi Vesta, có thể biểu thị thứ hạng, các thông số đếm liên quan và các thông tin khác.
- id_01-id_38: Các thông tin định danh liên quan đến giao dịch.
- DeviceType: Thể loại thiết bị dùng để thanh toán.
- DeviceInfo: Các thông tin liên quan đến thiết bị dùng để thanh toán.

2.2 Bài toán

Từ tập dữ liệu trên, chúng tối sẽ sử dụng các phương pháp học máy để giải bài toán học có giám sát phân loại nhị phân về việc đánh giá một giao dịch có là lừa đảo hay không. Các giao dịch sẽ bao gồm 433 thuộc tính cùng với 1 nhãn (isFraud). Mục tiêu sẽ là dự đoán chính xác giá trị của nhãn: 0 với giao dịch hợp lệ và 1 với giao dịch lừa đảo. Từ tập dữ liệu cho trước, chúng tối sẽ chia thành hai tập huấn luyện và kiểm tra với tỷ lệ 80:20 sao cho đảm bảo sự phân phối nhãn với tập ban đầu (stratify sampling). Các mô hình sẽ được xây dựng dựa trên tập huấn luyện và được đánh giá, so sánh trên tập kiểm tra.

3 Tiền xử lý và khai phá dữ liệu

3.1 Thông tin về nhãn - isFraud

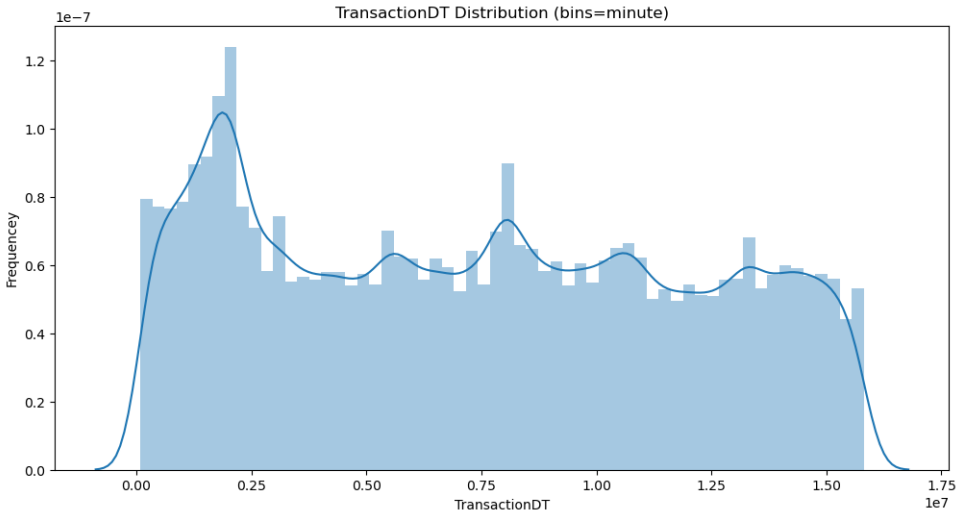


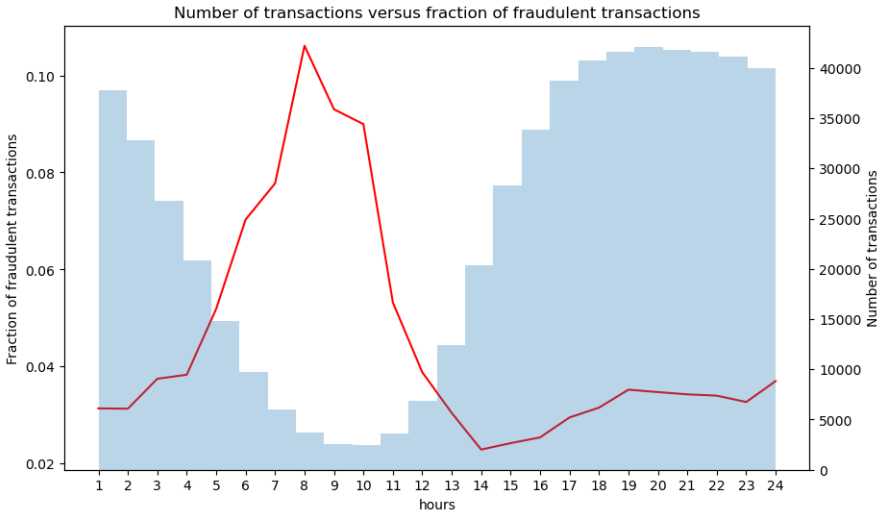
Chúng ta có thể thấy rõ sự chênh lệch lớn giữa số lương giao dịch hợp lệ và lừa đảo trong tập dữ liệu. Chỉ có 3.47% trong tông số 590540 giao dịch trong tập dữ liệu là lừa đảo, điều này phản ánh đúng thực tế khi số lượng giao dịch lừa đảo rất ít. Do tính chất mất cân bằng trong nhãn của tập dữ liệu, các phương pháp được sử dụng cần phải có khả năng xử lý đặc trưng này, hoặc chúng ta cần phải có những bước cân bằng dữ liệu huấn luyện trước khi xây dựng mô hình học.

3.2 Thông tin về các thuộc tính

3.2.1 TransactionDT

Đây là cột chứa thông tin về thời gian giao dịch từ 1 thời điểm xác định trong quá khứ và được tính bằng giây.



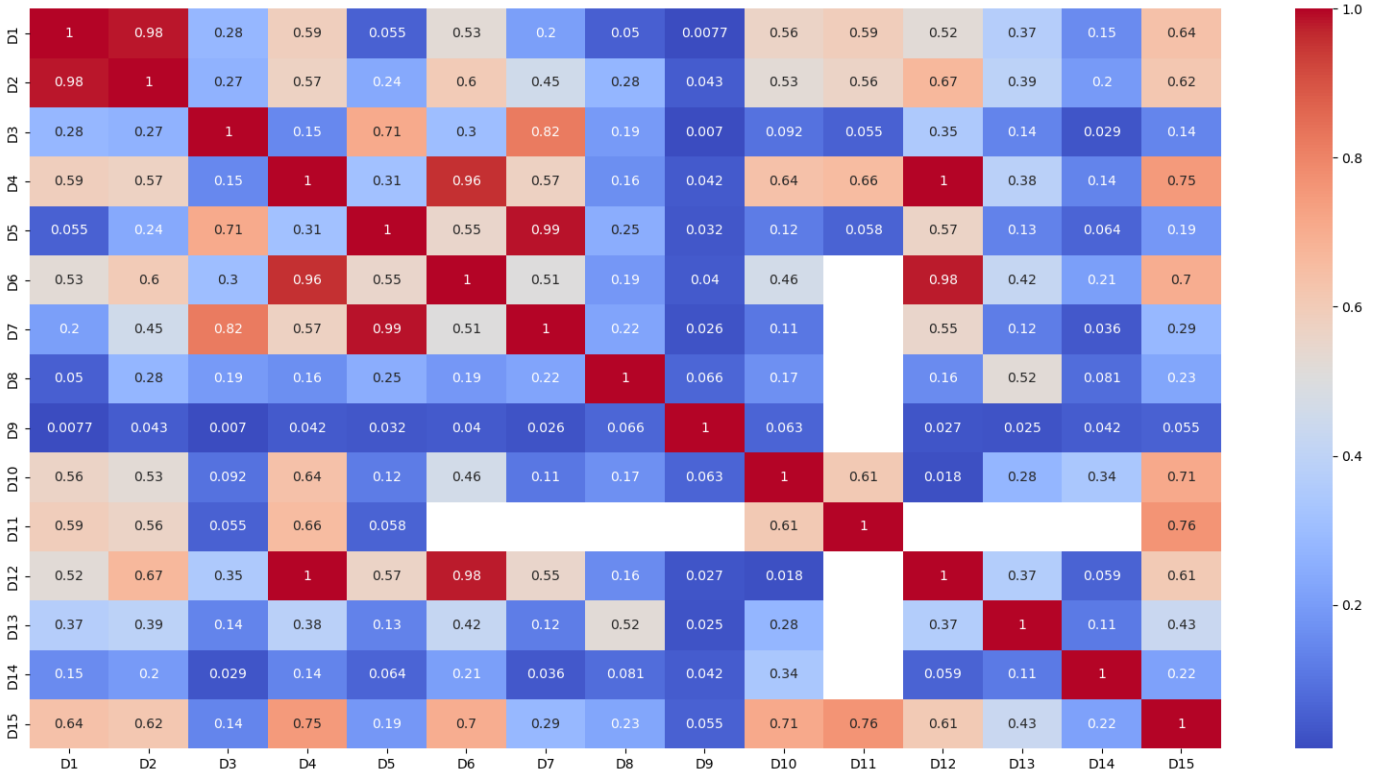


Khi thực hiện phân tích tỷ lệ giao dịch bất thường trong mỗi khung giờ, chúng ta có thể thấy có 1 điểm đáng chú ý là khung giờ từ 7 cho đến 11h số lượng giao dịch rất ít tuy nhiên tỷ lệ bất thường lại đặc biệt cao.Do đó,sẽ cần tạo 1 cột mới là 'hour_warning' để biểu diễn cho phát hiện này

3.2.2 D1 - D15

Cột này chứa thông tin về khoảng thời gian giữa 2 giao dịch liên tiếp và có tỷ lệ missing value lớn cùng với đó là độ correlation cao giữa các feature (Điều này là hoàn toàn hợp lý do chúng ta thường thực hiện giao dịch vào 1 thời điểm nào đó trong ngày)

	Name	dtypes	Missing_Percent	Unique
0	TransactionDT	int64	0.000000	573349
1	D1	float64	0.214888	641
2	D2	float64	47.549192	641
3	D3	float64	44.514851	649
4	D4	float64	28.604667	808
5	D5	float64	52.467403	688
6	D6	float64	87.606767	829
7	D7	float64	93.409930	597
8	D8	float64	87.312290	12353
9	D9	float64	87.312290	24
10	D10	float64	12.873302	818
11	D11	float64	47.293494	676
12	D12	float64	89.041047	635
13	D13	float64	89.509263	577
14	D14	float64	89.469469	802
15	D15	float64	15.090087	859

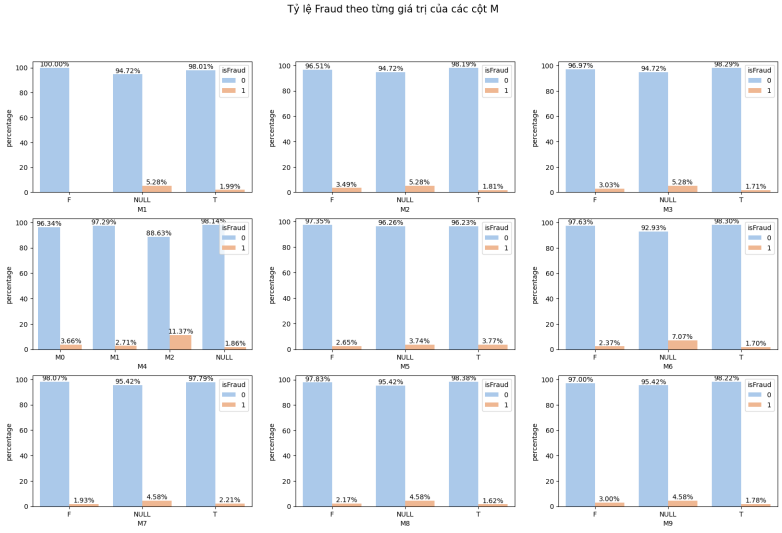


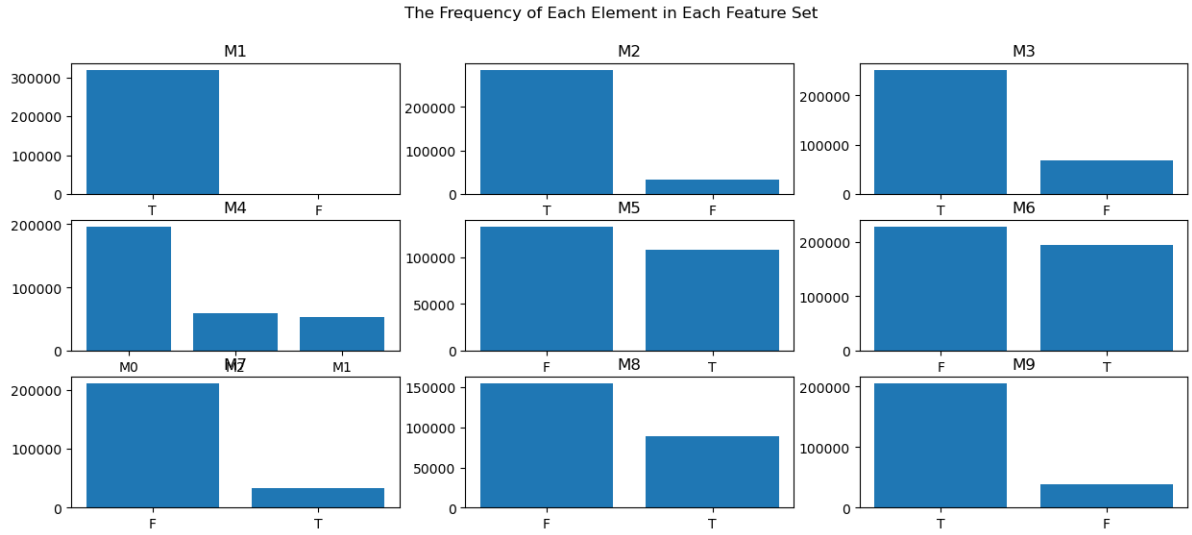
Do tỷ lệ missing percent lớn và độ correlation cao , do đó giữa 2 feature có độ correlation cao ta sẽ loại bỏ đi 1 và giữ lại feature có tỷ lệ missing ít hơn.

Đối với những dữ liệu bị missing , dựa trên thông tin chúng ta biết về cột này . Nó hoàn toàn hợp lý nếu chúng ta điền những giá trị null bằng trung bình của từng cột.

3.2.3 M1 - M9

Cột M chứa thông tin đó là thông tin người dùng điền trên những thẻ tín dụng và thông tin từ căn cước công dân của người đó có trùng nhau hay không.

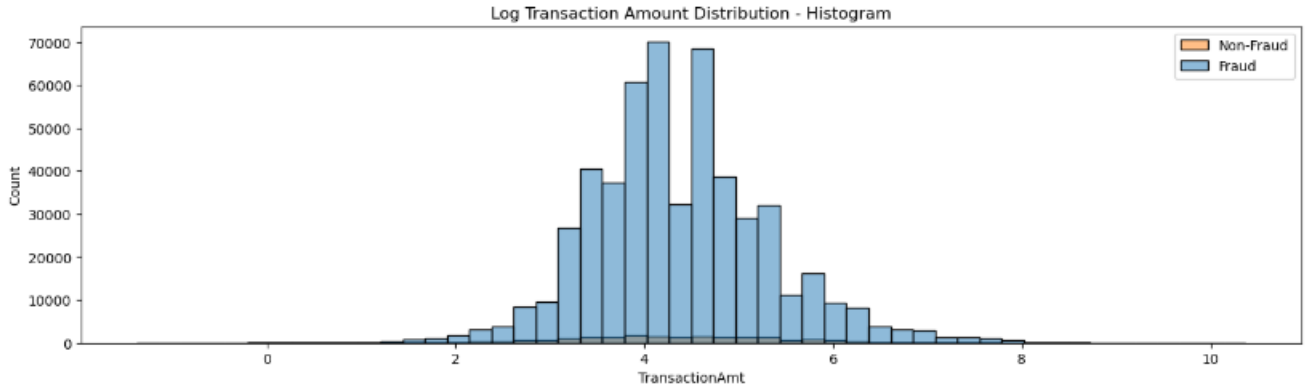


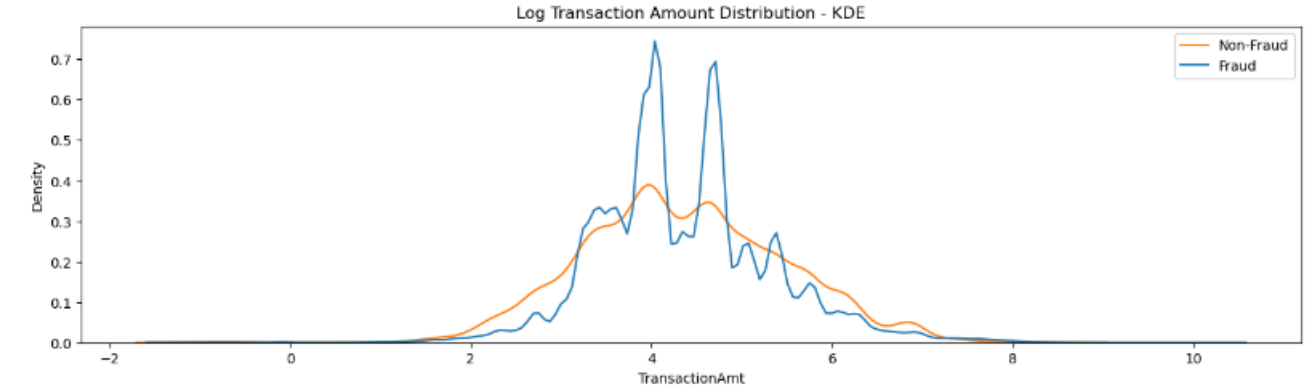


Nhìn vào bảng phân tích phía dưới , tỷ lệ Fraud vẫn rất nhỏ mặc dù những thông tin được người dùng đưa ra không trùng nhau hay là trường này bị thiếu thông tin cộng với việc số lượng giá trị True trong các cột là lớn nhất do đó ta có thể đi đến kết luận rằng điền các giá trị null của các cột này bằng mode(True) là hoàn toàn hợp lý.

3.2.4 TransactionAMT

Những giá trị trong cột TransactionAMT rất lớn, vậy nên chúng tôi quyết định áp dụng phép biến đổi logarit để nâng cao khả năng trực quan hóa của dữ liệu. Bằng cách áp dụng phép biến đổi logarit, chúng tôi đã nhận thấy sự cải thiện đáng kể trong hình dạng của phân phối dữ liệu cho những giao dịch gian lận, thể hiện bằng một đường cong chuông rộng hơn, cho thấy sự tập trung cao hơn của các giao dịch trong các khoảng thời gian cụ thể và sự tiềm ẩn của các hoạt động gian lận. Đặc biệt, chúng tôi đã quan sát rằng các giao dịch có giá trị logarit lớn hơn 5.5 (tương đương 244 đô la) và nhỏ hơn 3.3 (tương đương 27 đô la) thường có tần suất và mật độ xác suất gia tăng liên quan đến gian lận. Ngược lại, các giao dịch nằm trong khoảng từ 3.3 đến 5.5 có khả năng là giao dịch hợp pháp cao hơn.





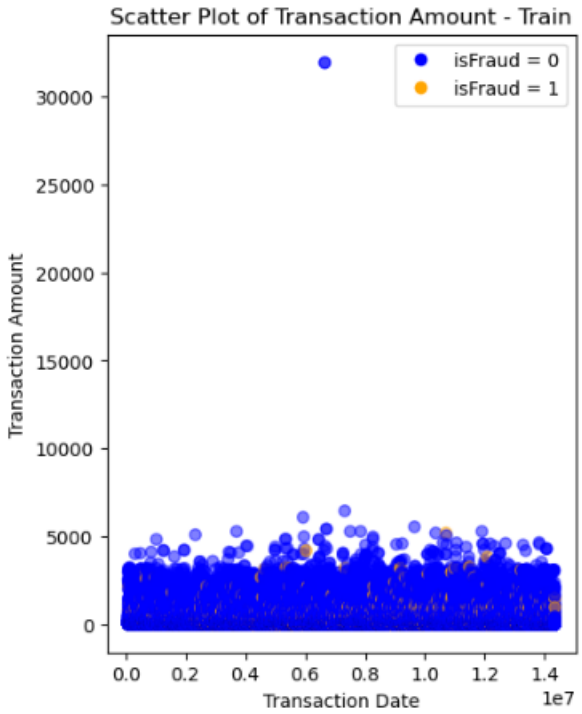
Ngoài ra, thông tin thống kê về số tiền giao dịch gian lận và không gian lận (đã được áp dụng phép biến đổi logarit) cho thấy rằng giá trị trung vị của số tiền giao dịch gian lận cao hơn một chút so với giao dịch không gian lận. Giá trị trung vị của giao dịch gian lận là khoảng 75 USD, trong khi trung vị của giao dịch không gian lận là khoảng 68.5 USD. Điều này có thể cho thấy rằng giao dịch gian lận thường bao gồm số tiền giao dịch ở mức cao hơn.

```
Mean fraud transaction amount: 4.357795773913444 (78.08482797687576 USD)
Median fraud transaction amount: 4.31748811353631 (74.99999999999997 USD)
Mean non-fraud transaction amount: 4.364256583540277 (78.59095241143272 USD)
Median non-fraud transaction amount: 4.22683374526818 (68.5 USD)
```

Trong tập dữ liệu, chúng tôi đã phát hiện một số giá trị ngoại lai cực đoan, mà chúng được phân biệt rõ ràng so với phần còn lại của dữ liệu. Các giá trị này vượt quá mức 10,000 và dường như có sự trùng lặp.

	TransactionID	isFraud	TransactionDT	TransactionAmt	ProductCD	card1	card2	card3	card4	card5	card6	addr1	addr2	dist1	dist2	P_emailc
274336	3261336	0	6652360	31937.391	W	16075	514.0	150.0	mastercard	102.0	credit	205.0	87.0	27.0	NaN	yah
274339	3261339	0	6652391	31937.391	W	16075	514.0	150.0	mastercard	102.0	credit	205.0	87.0	27.0	NaN	yah

2 rows × 435 columns



Những giá trị ngoại lai này có tiềm năng ảnh hưởng đến độ chính xác của các mô hình máy học, đặc biệt là trong trường hợp các thuật toán dựa vào khoảng cách như hồi quy logistic và KNN. Sự lo ngại chính là việc overfitting có thể xảy ra khi mô hình học từ những giá trị ngoại lai này, đồng thời mang vào mô hình nhiễu không liên quan. Ngoài ra, các mô hình dựa trên cây quyết định có thể đặt các giá trị ngoại lai vào các nhánh lá riêng biệt, có thể làm ảnh hưởng đến khả năng tổng quát hóa của mô hình.

Do đó, để tăng tính ổn định và hiệu suất của mô hình trên nhiều thuật toán khác nhau, chúng tôi đã quyết định loại bỏ những giá trị ngoại lai vượt quá 10,000 ra khỏi tập huấn luyện. Điều này giúp giảm nguy cơ overfitting và đảm bảo mô hình hoạt động hiệu quả hơn trên dữ liệu.

Lưu ý rằng chúng tôi không loại bỏ toàn bộ các giá trị ngoại lai, mà chỉ tập trung vào loại bỏ những giá trị cực đoan nhất. Việc này nhằm tránh mất đi thông tin quan trọng trong trường hợp có những giá trị ngoại lai hợp lệ và mang ý nghĩa quan trọng trong việc phát hiện gian lận trong giao dịch.

3.2.5 dist1, dist2

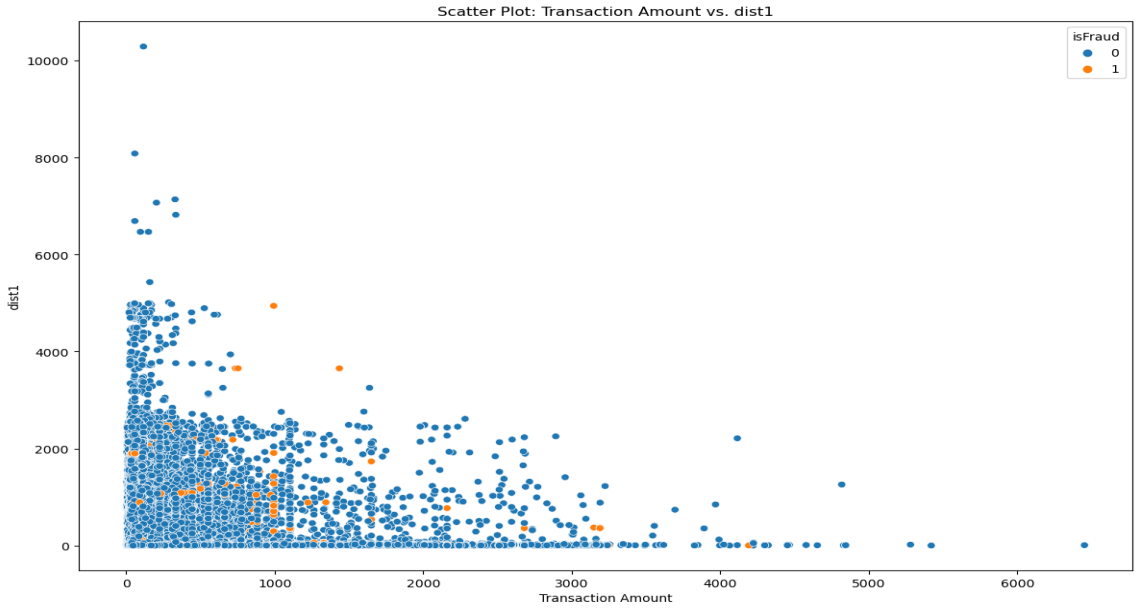
Sau khi kiểm tra dữ liệu thì ta có thể thấy số dữ liệu missing/null của hai cột dist1 và dist2 là rất lớn:

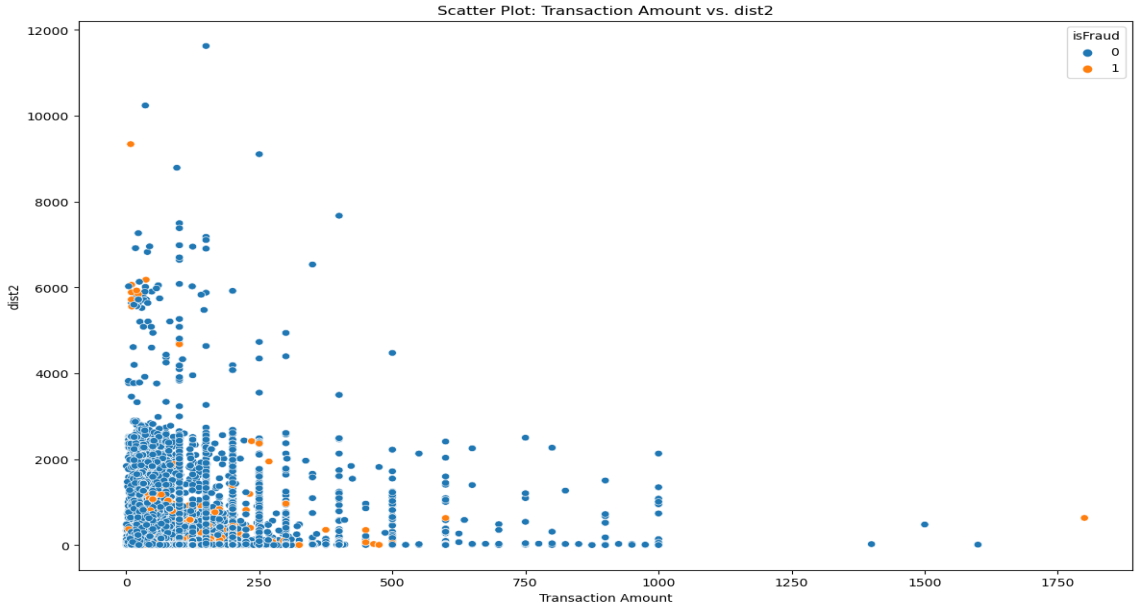
```
Percentage of missing values in dist1 column: 60.05%
Percentage of missing values in dist2 column: 93.49%
```

Trong trường hợp của cột dist1, giá trị trung bình khoảng cách cho các giao dịch gian lận (175.28) lớn hơn khoảng 1.49 lần so với các giao dịch không gian lận (117.73). Sự chênh lệch này có thể được giải thích bằng xu hướng của các tội phạm thực hiện các hoạt động bất hợp pháp từ các vị trí có khoảng cách đáng kể từ địa chỉ của chủ sở hữu thẻ tín dụng.

	dist1	dist1 for Fraud Transactions	dist1 for Non-Fraud Transactions	dist2	dist2 for Fraud Transactions	dist2 for Non-Fraud Transactions
count	217165.000000	4316.000000	212849.000000	35387.000000	3372.000000	32015.000000
mean	118.870426	175.280584	117.726581	233.900641	207.670225	236.663377
std	371.838198	424.987423	370.593983	535.811666	468.406997	542.356436
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	3.000000	3.000000	3.000000	7.000000	7.000000	7.000000
50%	8.000000	11.000000	8.000000	36.000000	49.000000	36.000000
75%	24.000000	80.000000	24.000000	213.000000	249.000000	207.000000
max	10286.000000	4942.000000	10286.000000	11623.000000	9337.000000	11623.000000

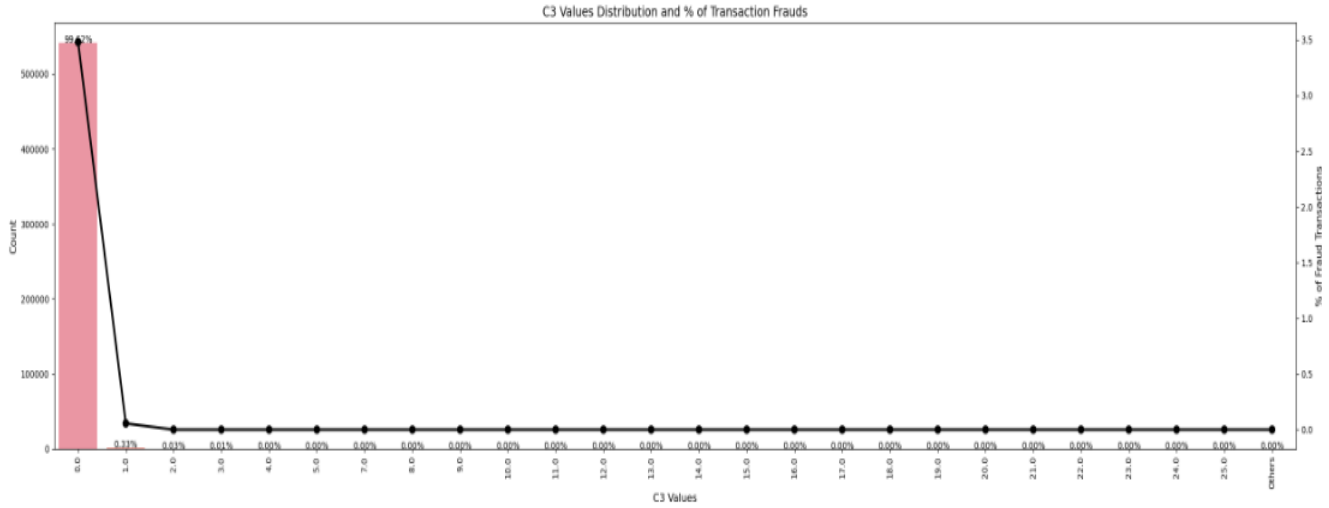
Từ biểu đồ phân tán, ta quan sát được rằng đối với cột dist1, những giá trị lớn hơn hoặc bằng 6000 được coi là các giá trị ngoại lai và có thể được xem xét để loại bỏ. Tương tự, đối với cột dist2, những giá trị lớn hơn hoặc bằng 8000 cũng được xem xét là các giá trị ngoại lai và có thể cần loại bỏ. Tuy nhiên, quan trọng là chúng ta chỉ xem xét loại bỏ các giá trị cực trị, bởi vì có nguy cơ mất thông tin có giá trị nếu những giá trị ngoại lai này là thực sự có liên kết và quan trọng cho quá trình phát hiện gian lận.

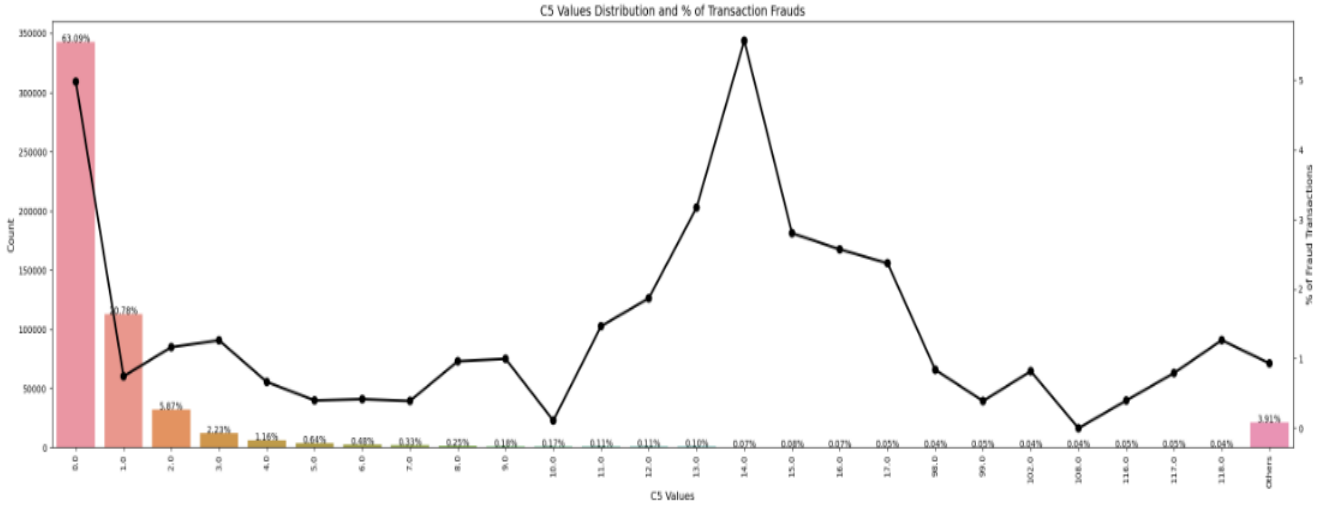


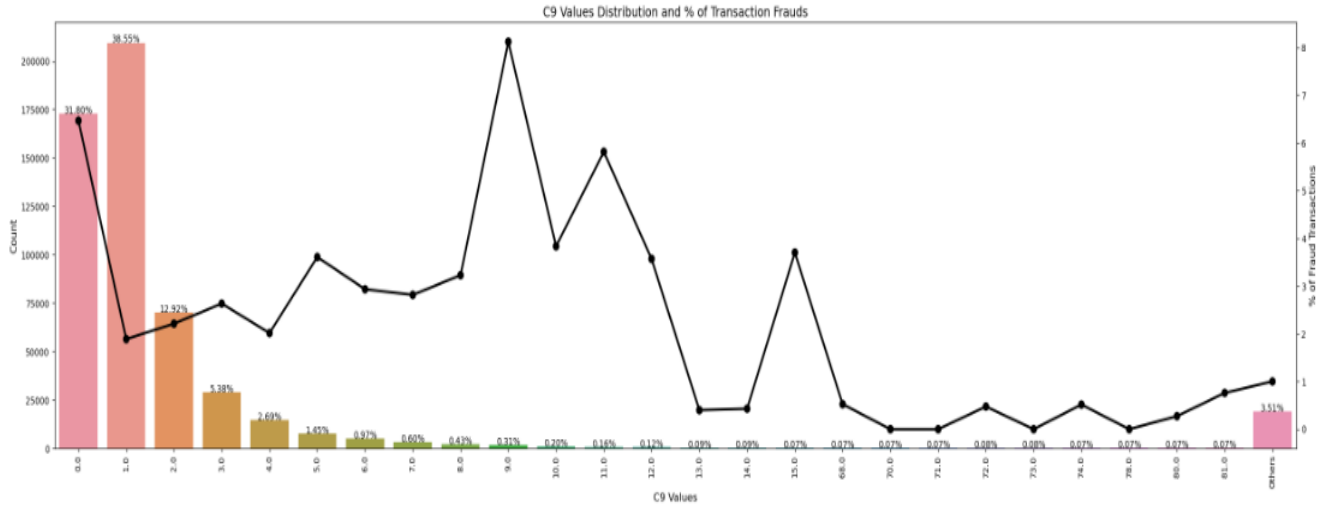


3.2.6 C1 - C14

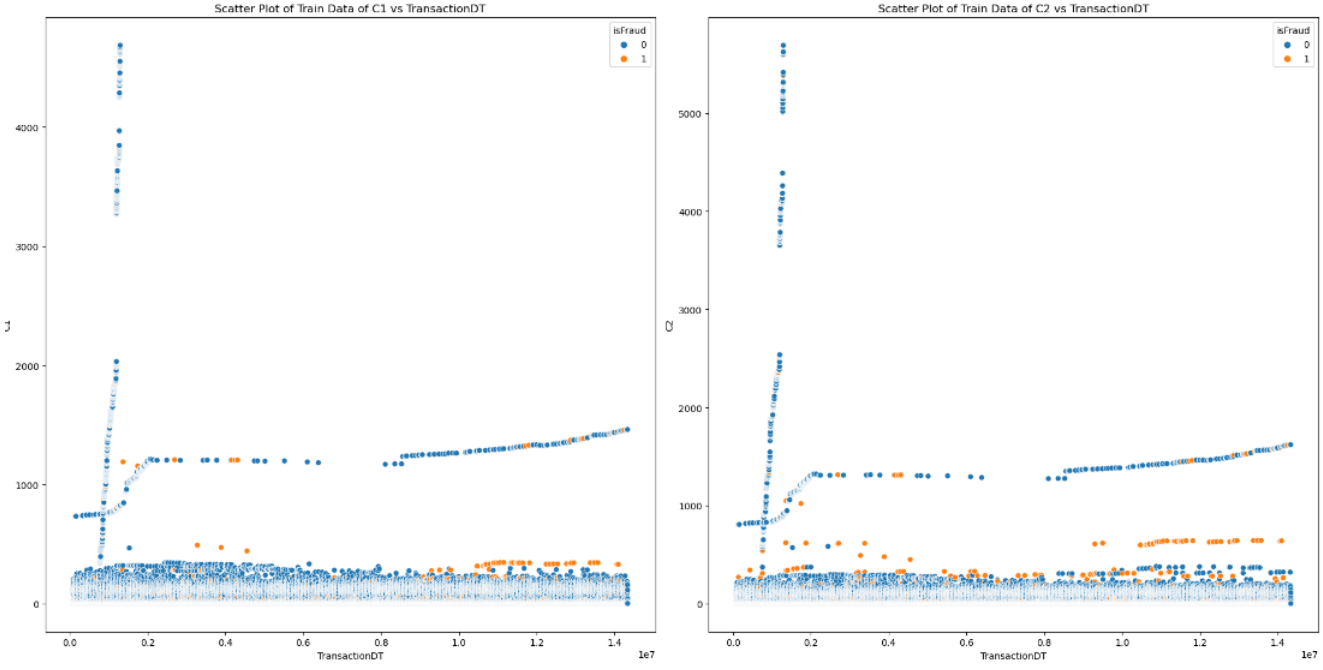
Chúng tôi đã tiến hành phân tích dữ liệu bằng cách vẽ biểu đồ phân phối của các giá trị trong cột C, và kết quả cho thấy có một số điểm đáng chú ý. Cụ thể, chúng tôi quan sát rằng các cột C3, C5 và C9 là những cột duy nhất có một hoặc nhiều giá trị có số lần xuất hiện cao, và tỷ lệ giao dịch lừa đảo trong số các dữ liệu này cũng rất cao. Điều này cho thấy rằng các cột này có thể chứa thông tin quan trọng cho việc phân loại giao dịch lừa đảo. Các cột khác thường không có mối liên hệ nào đáng chú ý với tỷ lệ giao dịch lừa đảo.

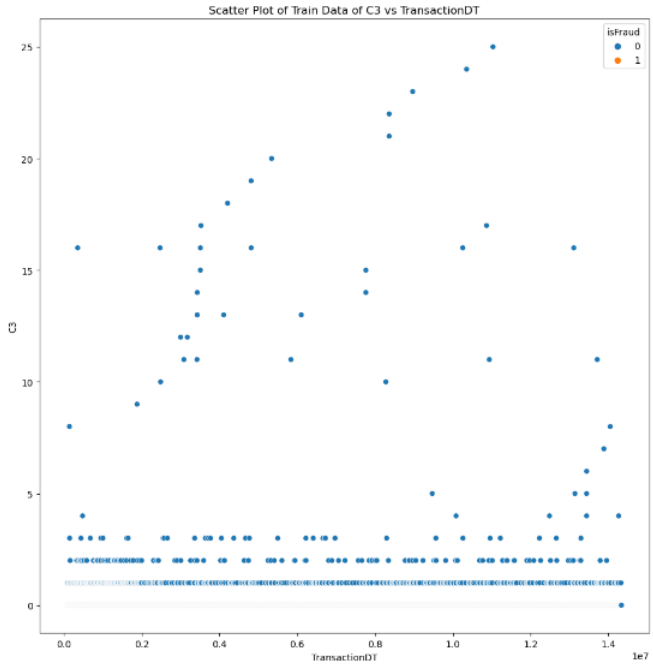


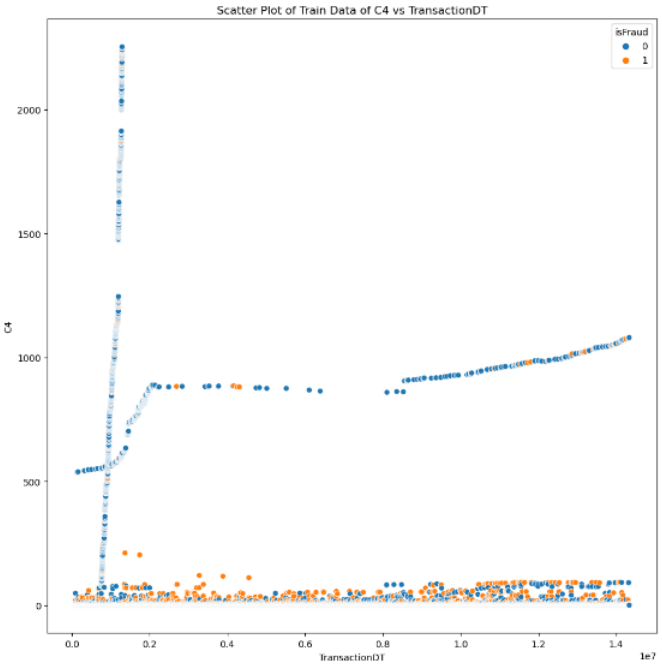




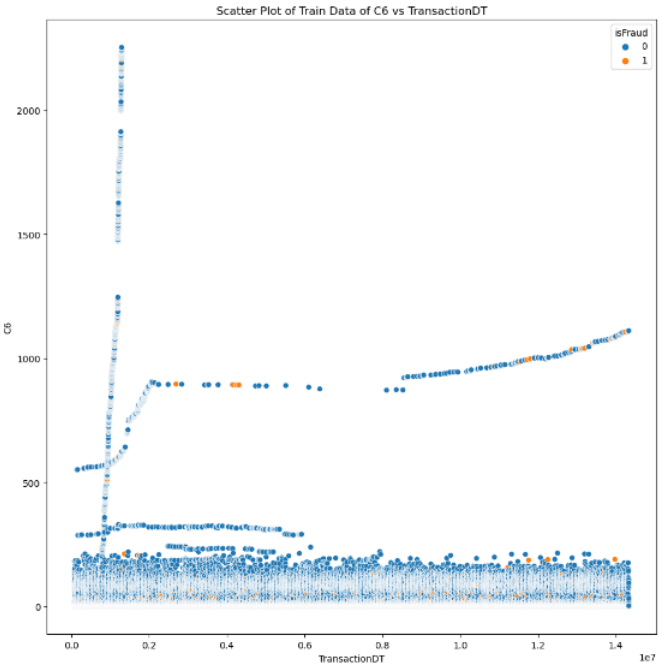
Sau khi nghiên cứu qua các biểu đồ phân tán giữa giá trị của cột C tương ứng với thời gian giao dịch, chúng tôi nhận thấy các cột C3, C5 và C9 có vẻ không chứa các giá trị ngoại lai cực đại. Trong khi đó, chúng tôi đã xác định rằng một số đặc trưng C khác có sự hiện diện của các giá trị ngoại lai cực độ, chúng thường có các giá trị TransactionDT nằm trong khoảng từ 1028000 đến 1199904 (tương ứng với các giao dịch từ ngày 4 đến ngày 6). Đáng chú ý, trong khoảng thời gian cụ thể này, chúng tôi đã xác định một số lượng đáng kể địa chỉ, thông tin, etc. được kết nối với các giao dịch thanh toán có giá trị ngoại lai. Điều này có thể đề xuất sự liên quan đến mô hình gian lận trong khoảng thời gian cụ thể đó. Chúng tôi đã thực hiện việc loại bỏ các giao dịch có những giữ liệu C nằm trong khoảng 900-3000 tùy vào giá trị C tương ứng.

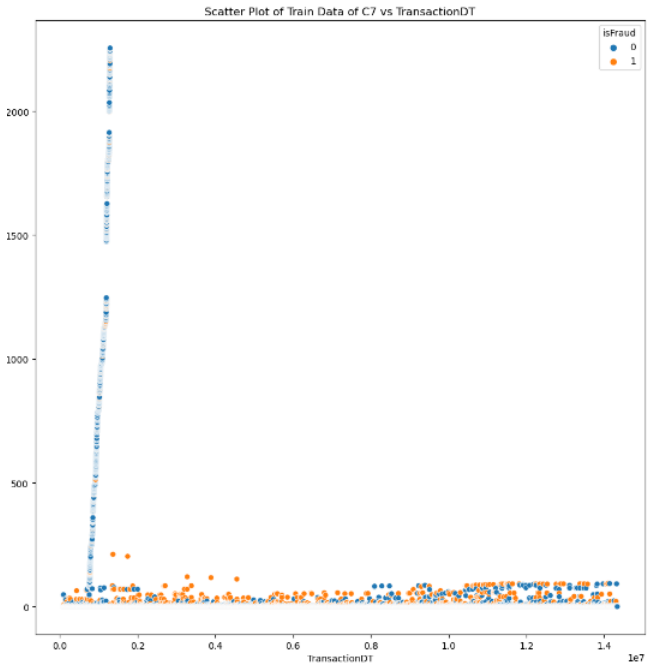


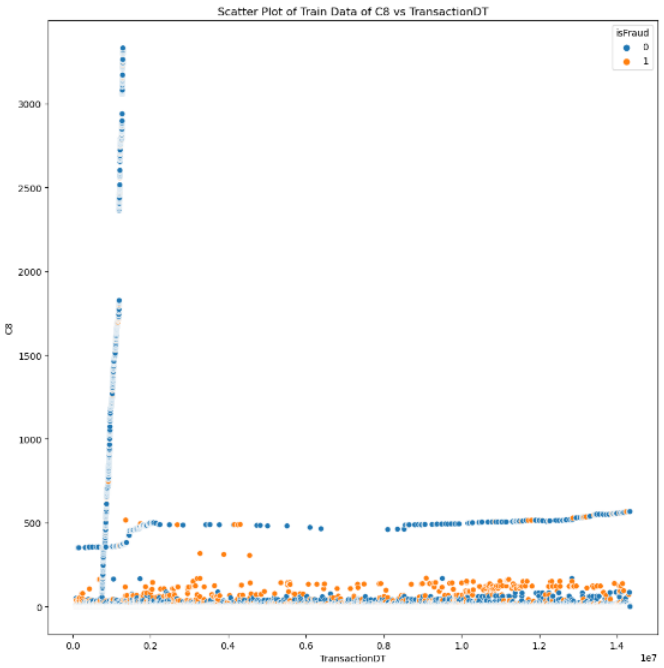

Scatter Plot of Train Data of C3 vs TransactionDT
isFraud
0
1
C3
TransactionDT
1e7

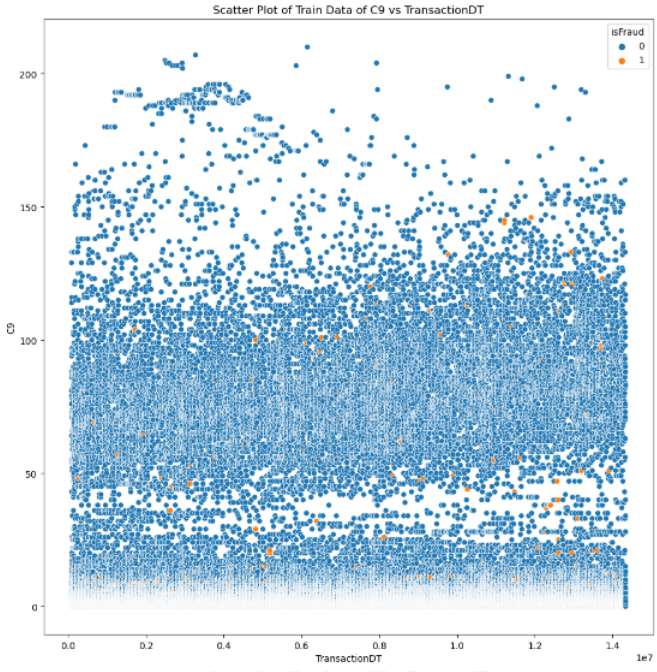

Scatter Plot of Train Data of C4 vs TransactionDT
isFraud
0
1
C4
TransactionDT
1e7

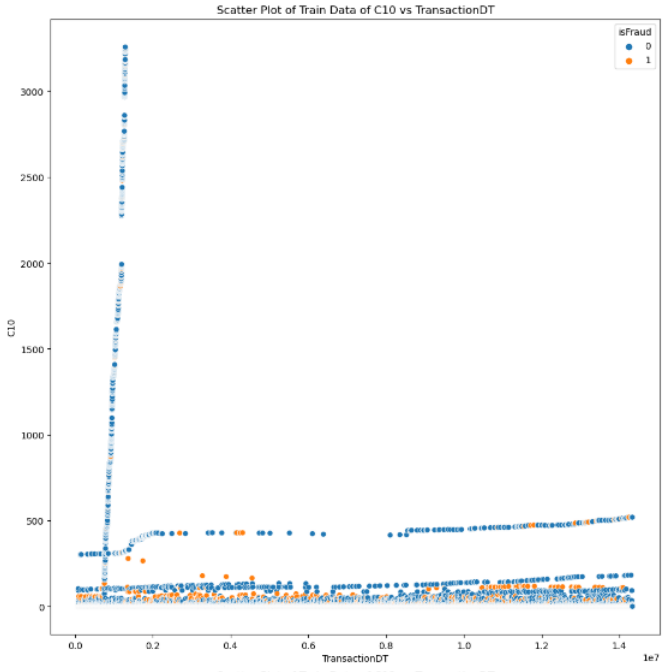

Scatter Plot of Train Data of C5 vs TransactionDT
isFraud
0
1
TransactionDT
1e7

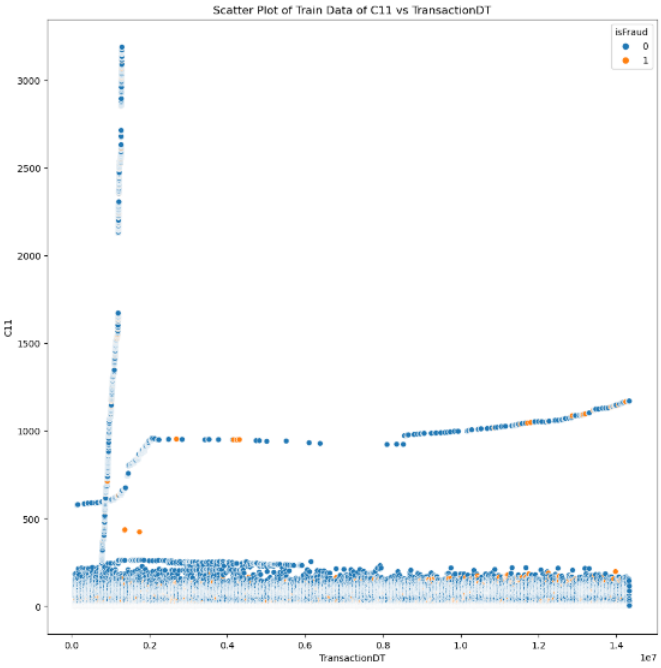

Scatter Plot of Train Data of C6 vs TransactionDT
isFraud
0
1
C6
TransactionDT
1e7

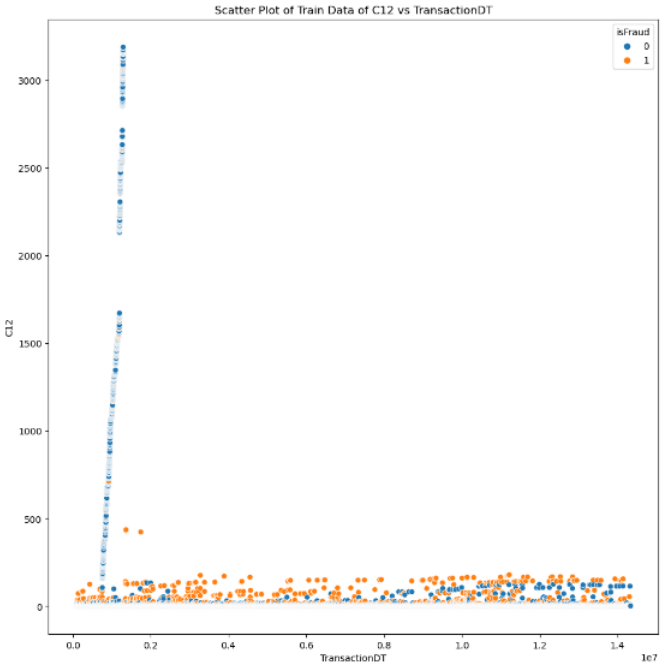

Scatter Plot of Train Data of C7 vs TransactionDT
isFraud
0
1
2000
1500
1000
500
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
TransactionDT
1e7

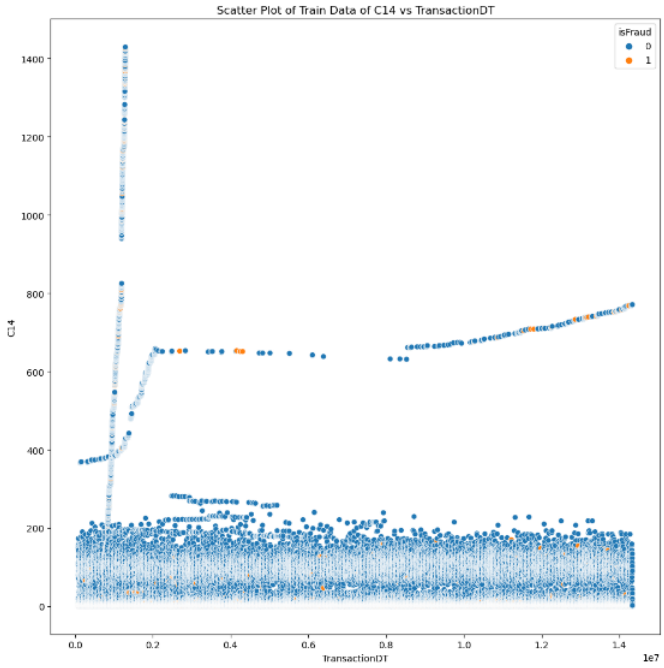

Scatter Plot of Train Data of C8 vs TransactionDT
isFraud
0
1
C8
3000
2500
2000
1500
1000
500
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
TransactionDT
1e7


Scatter Plot of Train Data of C9 vs TransactionDT
isFraud
0
1
C9
200
150
100
50
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
TransactionDT
1e7


Scatter Plot of Train Data of C10 vs TransactionDT
isFraud
0
1
C10
3000
2500
2000
1500
1000
500
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
TransactionDT
1e7

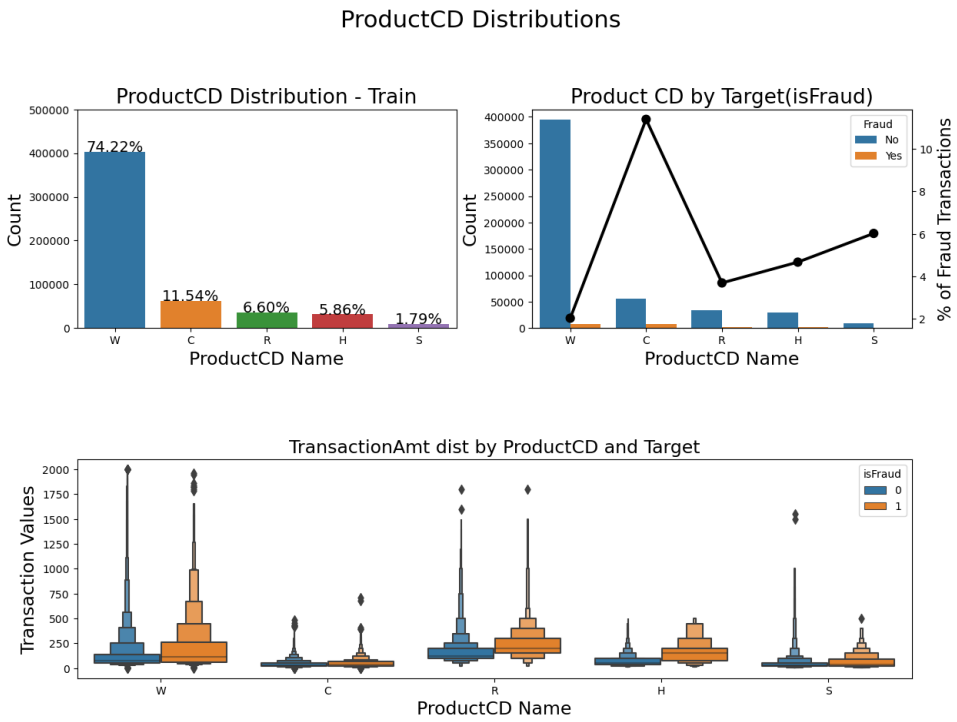






3.2.7 ProductCD

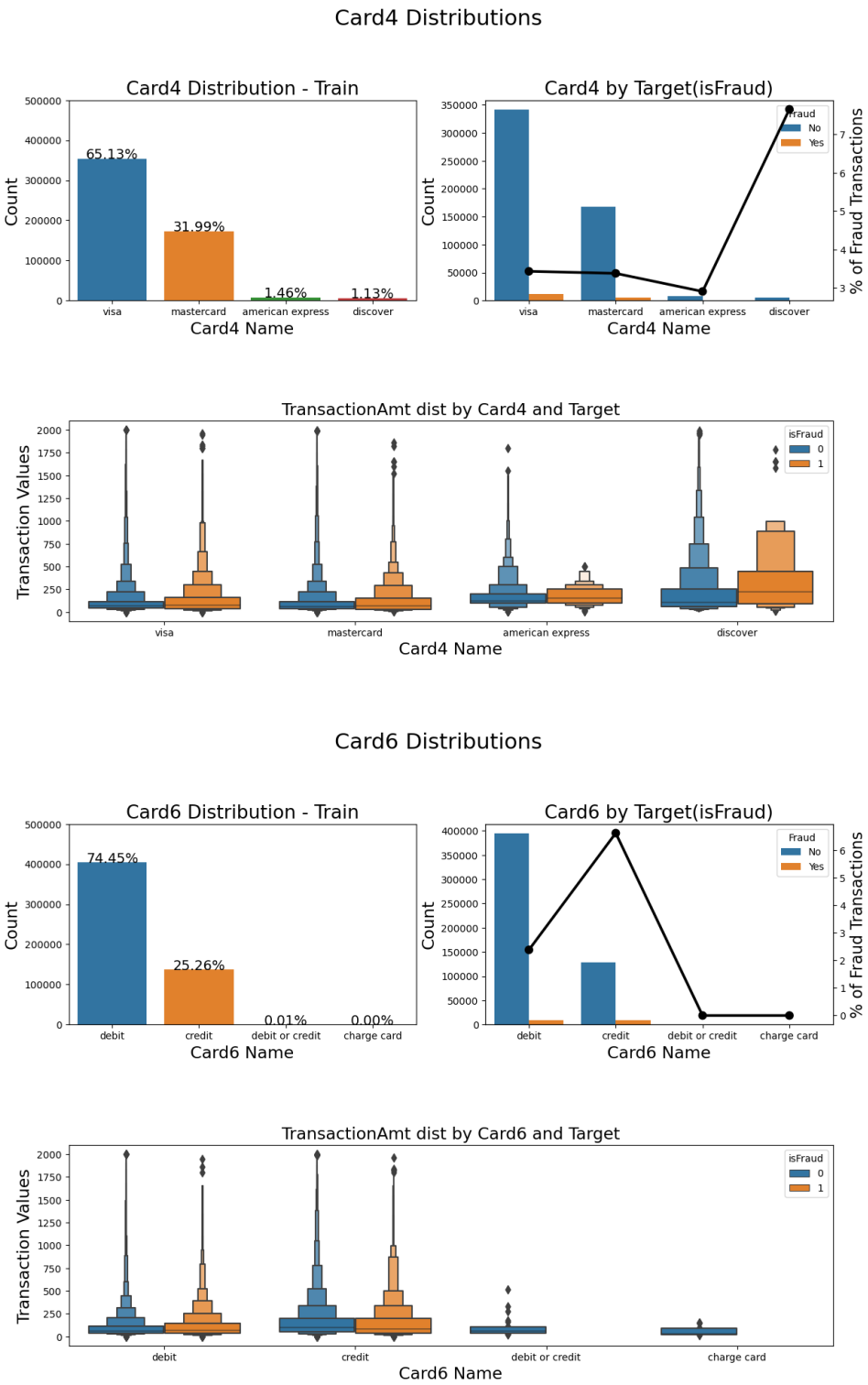
Thuộc tính ProductCD cho biết thông tin về mã của các sản phẩm được thanh toán trong giao dịch. Cột không có dữ liệu bỏ trống (missing data), bao gồm 5 giá trị: W,C,R,H,S. Chúng tôi đã khảo sát phân bố của các giá trị, cùng với khám phá những mối quan hệ của mã sản phẩm với nhãn và các thuộc tính khác.



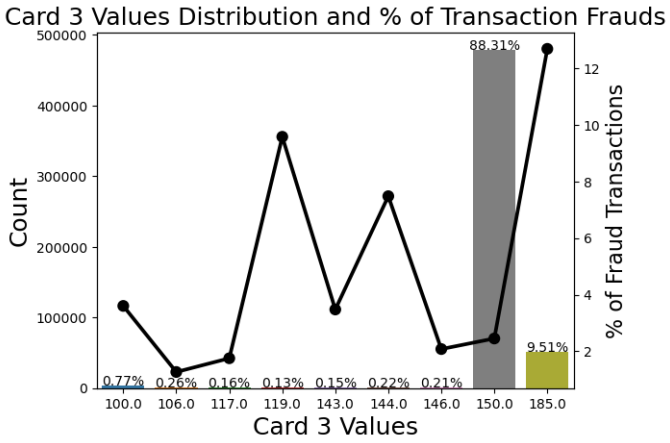
Trong 5 mã sản phẩm, mã W chiếm tỷ lệ lớn nhất, hơn 74%, các mã khác có tỷ lệ khá nhỏ. Quan sát xác suất lừa đảo của các mã, ta thấy được ở mã C tỷ lệ này cao đột biến (hơn 10%) trong khi mã W có tỷ lệ thấp hơn trung bình tập dữ liệu, chỉ vào khoảng 2%. Theo dõi các mã sản phẩm cùng với nhãn và cả số tiền thanh toán trong giao dịch, ta thấy ở các mã W, R, H có phân phối số tiền thanh toán cao hơn ở các giao dịch lừa đảo so với các giao dịch bình thường.

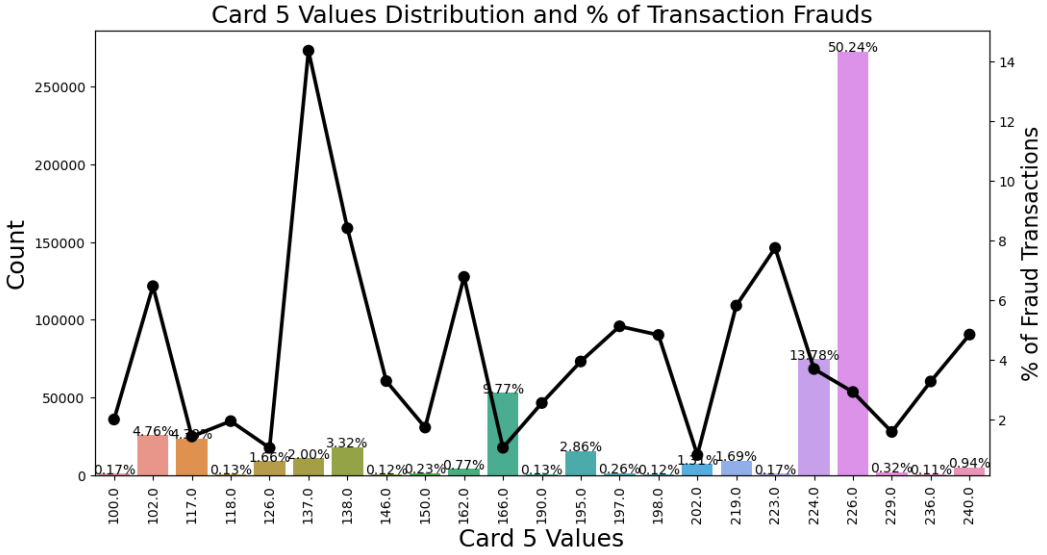
3.2.8 card1 - card6

Đây là các cột thông tin về thẻ thanh toán đã được che giấu ý nghĩa chính xác bởi nhà cung cấp. Các thông tin được suy đoán là các thông tin định doanh phân loại với các cột card1-3,5 do có nhiều giá trị nên đã được mã hóa bởi các số nguyên. Cột card4 và card6 chứa thông tin về loại thẻ giao dịch có số lượng giá trị ít nên ta cũng sẽ tiến hành khai phá như cột ProductCD. Về số lượng giá trị null, các cột có tỷ lệ dữ liệu thiếu thấp, cao nhất hơn 1% ở cột card2, các cột card3,4,6 có số lượng dữ liệu null khá tương đồng, trong khi cột card1 không có dữ liệu còn thiếu. Chúng tôi nhận ra khi cột card3 có giá trị null, các cột card2,4,5,6 cũng sẽ tương tự. Có tất cả 1565 bản ghi như vậy (chiếm khoảng 0,26%).



Có thể thấy phần lớn các giao dịch được thanh toán bằng thẻ visa và thẻ ghi nợ (debit), tỷ lệ thẻ mastercard và thẻ thanh toán (credit) cũng khá cao. Những các loại thẻ còn lại chiếm thiểu số. Có sự chênh lệch nhất định trong tỷ lệ lừa đảo của các loại thẻ, như thẻ discover và thẻ thanh toán (credit) có tỷ lệ cao hơn bình quân trên tập dữ liệu khá nhiều.

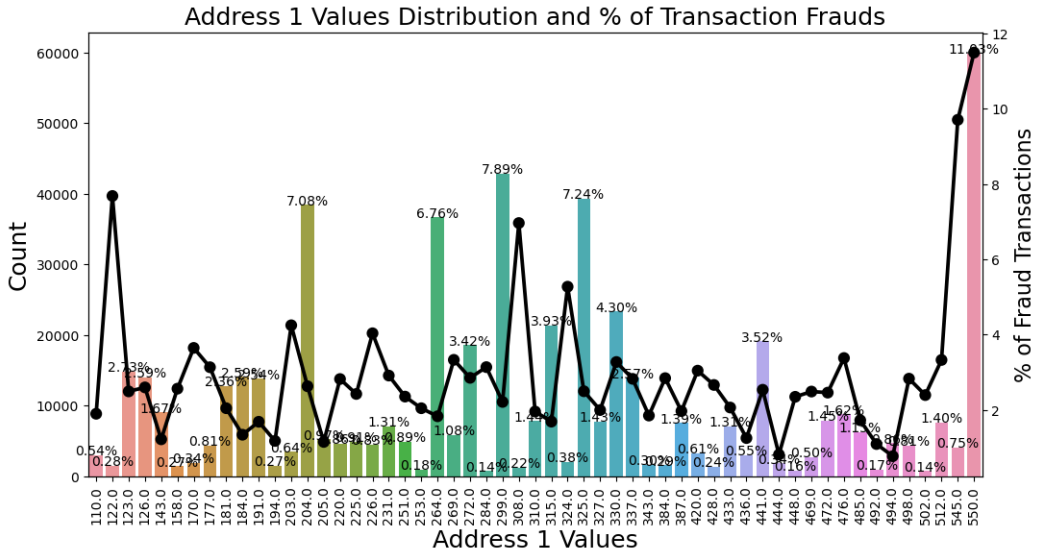


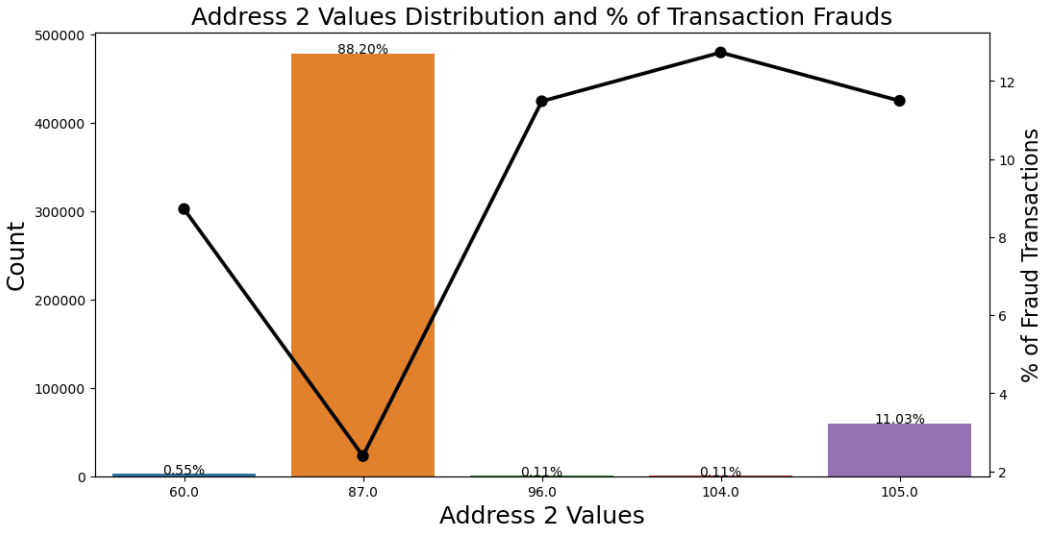


Đối với các cột thông tin thẻ có dữ liệu đã được chuyển về dạng số, chúng tôi cũng tiến hành khảo sát phân bố cũng như tỷ lệ lừa đảo trên những giá trị phổ biến. Chẳng hạn, cột card3 có hơn 88% giá trị là 150 trong khi hơn một nửa giá trị ơt cột card5 là 226. Cũng có sự chênh lệch khá lớn trong xác suất lừa đảo trên những giá trị khác nhau của cùng một thuộc tính.

3.2.9 addr1, addr2

Đây là hai thuộc tính cung cấp thông tin về địa chỉ trong đó addr1 là địa chỉ vùng và addr2 là địa chỉ quốc gia. Có hơn 11% bản ghi không có dữ liệu cho cả hai cột, vì vậy chúng tôi mã hóa chúng thành các giá trị mới: 550 ở cột addr1 và 105 ở cột addr2

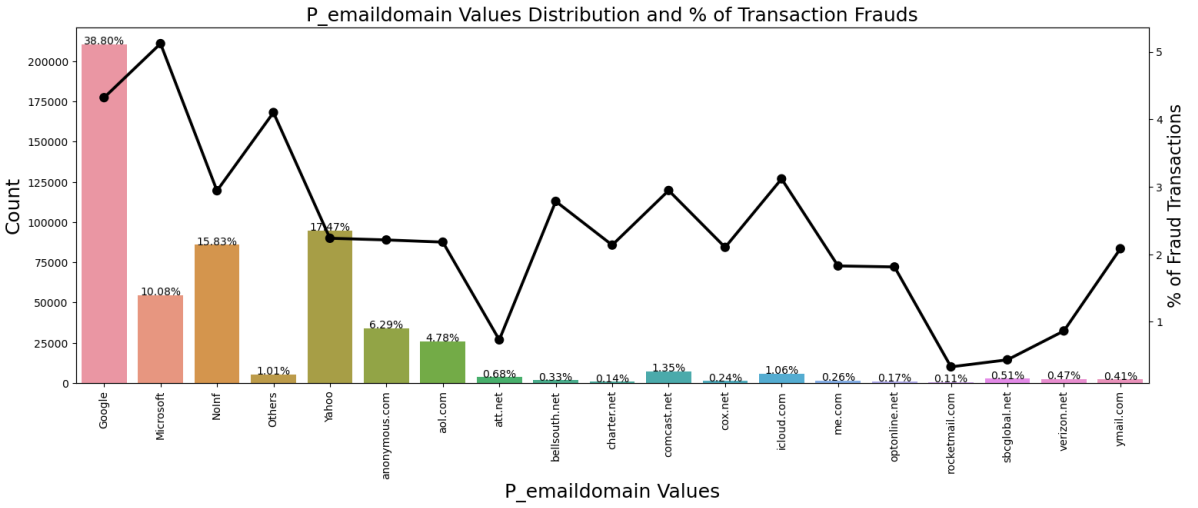


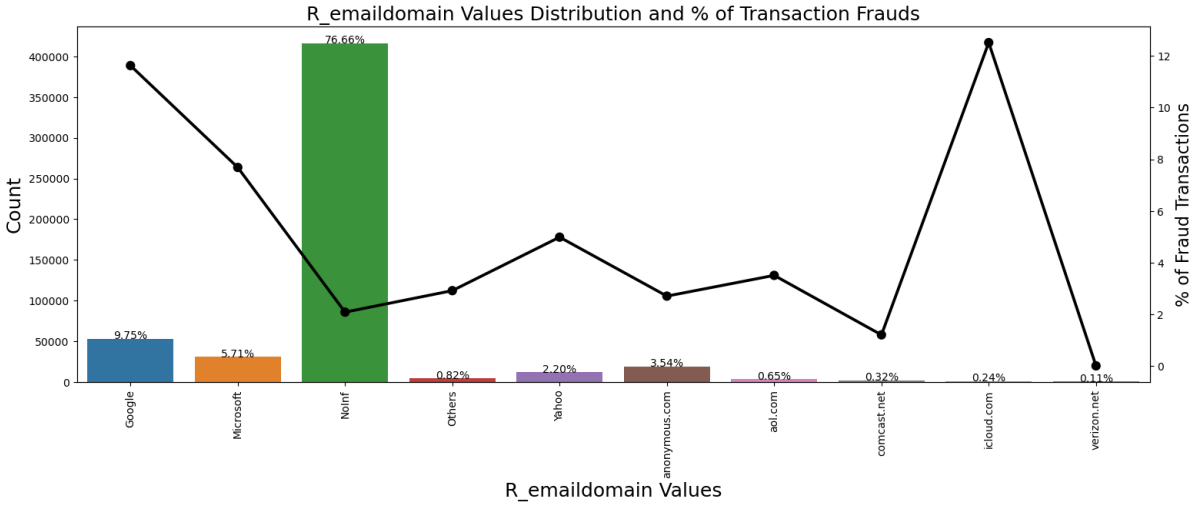


Ta thấy ở cột addr1, các giá trị phân bố khá dàn trải, tỷ lệ giao dịch lừa đảo cũng có sự chênh lệch nhưng nổi bật là ở giá trị null (gần 12%). Ở cột addr2, ta thấy hầu hết địa chỉ đều tập trung ở một quốc gia (87). Cắc giá trị khác (bao gồm cả null) có tỷ lệ giao dịch lừa đảo cao. Điều này chứng tỏ các giao dịch lừa đảo có thể dễ xảy ra ở nước ngoài hơn.

3.2.10 P_emaildomain, R_emaildomain

Các cột thông tin về đuôi email của người mua và người nhận hàng có số giá trị khá là lớn, vì vậy chúng ta cần nhóm những email này thành những phân loại chúng như Google, Microsoft. Đồng thời chúng tôi cũng tạo riêng giá trị NoInf cho các giá trị bị thiếu, chiếm lần lượt hơn 17% và 76% ở hai cột





Quan sát ta có thấy được phân bố không đồng đều của các đuôi email đồng thời là những thông tin khá rõ về tỷ lệ giao dịch lừa đảo có thể giúp ích cho việc học các mô hình phân loại sau này.

3.2.11 V1 - V339

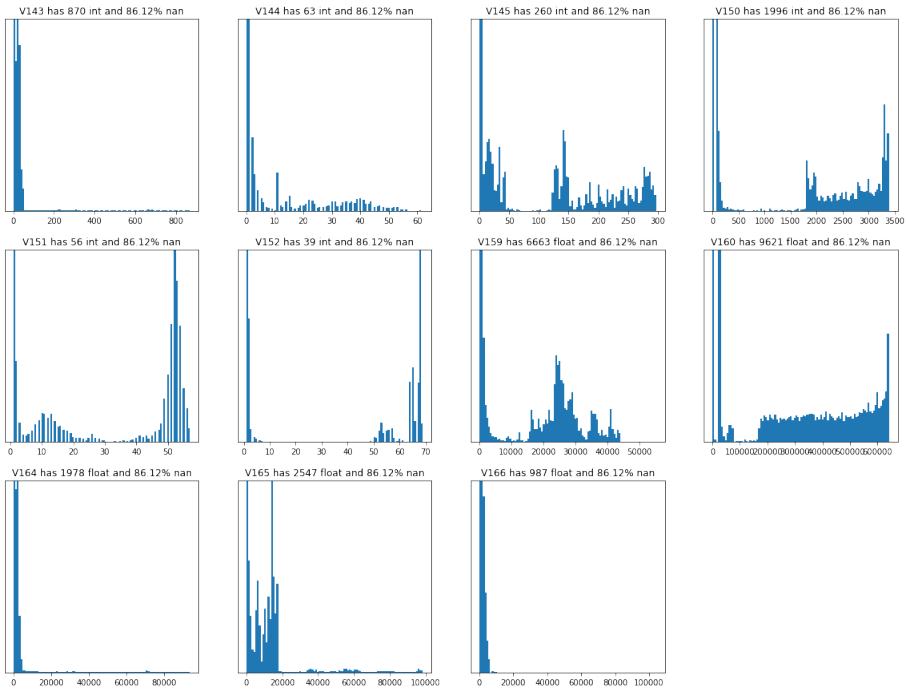
Đây là các cột chứa nhiều thông tin liên quan được Vesta corporation tự tạo ra và thêm vào, nhưng ý nghĩa thực sự của chúng đã bị ẩn đi.

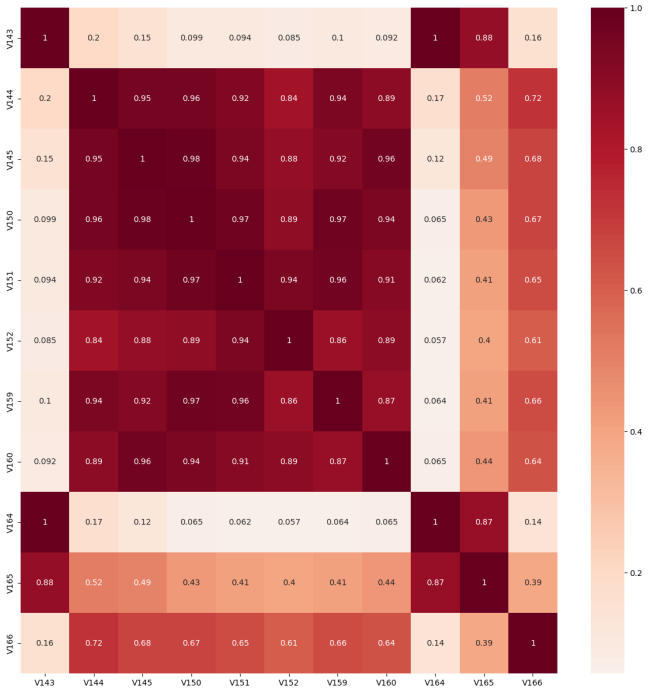
Đầu tiên, khi chúng tôi khảo sát số lượng dữ liệu bị mất đi, chúng tôi thấy được có rất nhiều cột với số lượng dữ liệu NaNs giống nhau

	dtypes	missing (count)	missing (%)	unique
V1	float64	279287	47.293494	2
V2	float64	279287	47.293494	9
V3	float64	279287	47.293494	10
V4	float64	279287	47.293494	7
V5	float64	279287	47.293494	7
...	...	...	...	...
V335	float64	508189	86.054967	672
V336	float64	508189	86.054967	356
V337	float64	508189	86.054967	254
V338	float64	508189	86.054967	380
V339	float64	508189	86.054967	334

339 rows × 4 columns

Điều này dẫn đến phỏng đoán của chúng tôi là có rất nhiều cột dữ liệu có nhiều tương quan với nhau, thậm chí là có thể có nhiều cột bị lặp lại dẫn đến việc dư thừa quá nhiều cột không cần thiết. Ta đã biết có nhiều cột tương quan là một điều không tốt cho việc dự đoán, và ta cần phải giảm đi số cột V này và chỉ giữ lại những cột V đặc trưng. Ví dụ như những cột V dưới đây (được biểu thị qua biểu đồ nhiệt của ma trận hệ số tương quan ở dưới) là những cột có cùng số lượng dữ liệu mất đi là 508589





Nhìn vào bức ảnh, ta có thể thấy là rất nhiều cột V đôi một tương quan với nhau, biểu thi bằng các vùng đỏ đậm. Trong các cột V có cùng số lượng dữ liệu missing như thế này, ta có thể nhóm chúng tiếp thành các nhóm trong đó các cột đều đôi một tương quan đáng kể với nhau (chỉ số correlation > 0.75). Sau đó, ta sẽ tiến hành chọn một cột đặc trưng duy nhất trong các nhóm nhỏ đó, và cột này sẽ được chọn với tiêu chí là chứa nhiều giá trị riêng (unique values) nhất để giữ lại nhiều thông tin nhất.

Từ 339 cột V, sau đó chúng tôi đã giảm được xuống còn 128 cột. Trong 128 cột này, chúng tôi tiếp tục tiến hành bỏ đi các cột với số lượng dữ liệu mất đi lớn hơn 80%, giữ lại các cột V có nhiều đặc trưng nhất.

Vì các cột V ở đây có dạng numeric, chúng tôi sẽ thay các giá trị NaNs của mỗi cột bằng giá trị nhỏ nhất của cột đó trừ 2 (min - 2). Việc này nhằm đảm bảo đánh dấu các giá trị NaNs là các giá trị đặc biệt nàm ngoài khoảng dữ liệu của cột.

3.2.12 Bảng định danh (Identity Table)

Dữ liệu của bảng này là dữ liệu để định danh thông tin cho các giao dịch (Ví dụ như thông tin về máy sử dụng để giao dịch, các giao thức mạng, hệ điều hành của máy, ...)

Dữ liệu gồm khoảng 40 cột và mang khá nhiều thông tin ý nghĩa, tuy nhiên thì các cột ở bảng lại có số lượng missing rất lớn. Các cột mang dữ liệu missing ít nhất là 70% và nhiều nhất lên tới 99%. Vì vậy có thể nó sẽ gây ra nhiễu chứ chúng ta không thể sử dụng như là 1 cột thông tin có ích

Chúng ta cần phải xem xét dữ liệu dưới đây để có các phương pháp xử lý dữ liệu hợp lý ví dụ như nên bỏ các cột để loại bỏ nhiễu hay các cột đó có mang thông tin phân loại tốt để có thể giữ lại và sử dụng các phương pháp xử lý missing để điền dữ liệu bị thiếu

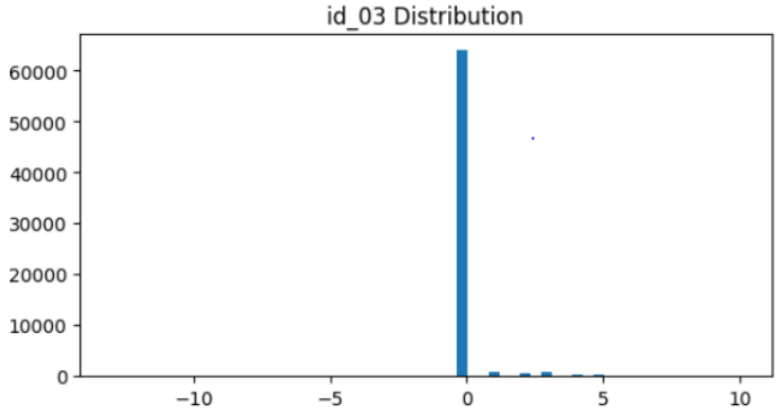
	Name	dtypes	Missing(%)	Uniques
0	id_01	float64	75.576083	77
1	id_02	float64	76.145223	115655
2	id_03	float64	88.768923	24
3	id_04	float64	88.768923	15
4	id_05	float64	76.823755	93
5	id_06	float64	76.823755	101
6	id_07	float64	99.127070	84
7	id_08	float64	99.127070	94
8	id_09	float64	87.312290	46
9	id_10	float64	87.312290	62
10	id_11	float64	76.127273	365
11	id_12	object	75.576083	2
12	id_13	float64	78.440072	54
13	id_14	float64	86.445626	25
14	id_15	object	76.126088	3
15	id_16	object	78.098012	2
16	id_17	float64	76.399736	104

Phía trên đây là % dữ liệu bị missing của 1 số cột trong bảng định danh. Trước hết chúng ta sẽ loại bỏ các cột mà có số lượng missing $> 90\%$ hay những cột có 1 phần tử mà tần suất xuất hiện $> 90\%$ vì cột đó sẽ tập trung dữ liệu vào 1 giá trị mà không mang nhiều giá trị phân loại.

Tiếp theo, ở trong bảng định danh có cả 2 loại dữ liệu là dữ liệu dạng phân loại (categorical) và dữ liệu dạng số (numeric)

- Dữ liệu dạng số (numeric): id01 - id11
- Dữ liệu dạng phân loại (categorical): id12 - id38 + DeviceType + DeviceInfo

Dữ liệu dạng số hầu như đều là các cột có phân bố tập trung rất lớn vào 1 điểm dữ liệu, xem ví dụ như id03 dưới đây:



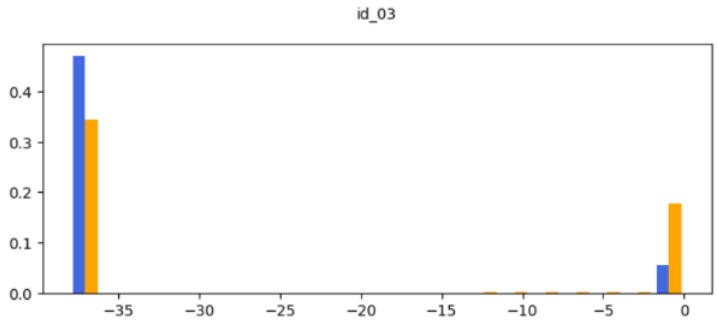

```
NaN    0.887689233582822
0.0    0.108211128797372
1.0    0.001461374335354
3.0    0.001131168083449
2.0    0.000712906831036
Name: id_03, dtype: float64
```

Có thể thấy là 88% dữ liệu là giá trị bị missing và 10% là giá trị 0.0. Như vậy là có tới 98% dữ liệu của cột chỉ tập trung vào 2 giá trị 0 và NaN

Nhìn qua thì có thể nghĩ là nên bỏ vì giá trị missing cũng khá lớn: 88%

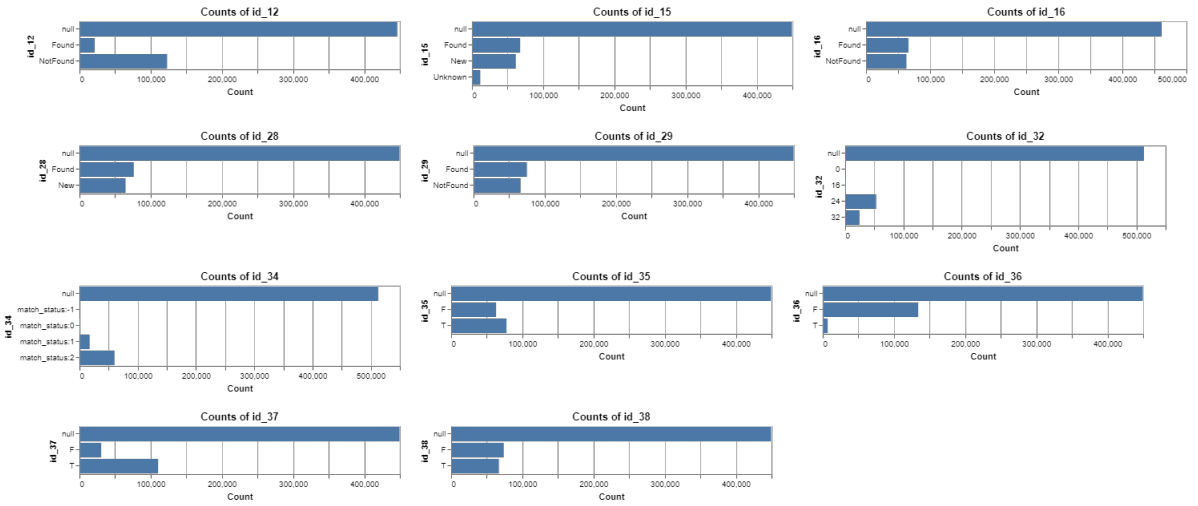
Tuy nhiên nếu coi giá trị missing là 1 nhãn mới thì nó lại có khả năng phân loại khá tốt vì cả cột thuộc tính id03 này chỉ tập chung chính vào 2 giá trị là 0 và NaN

Ảnh dưới đây thể hiện % của các nhãn so với tổng số nhãn, có thể thấy sự chênh lệch % giữa các nhãn khá rõ ràng nên nó có thể mang thông tin phân loại khá tốt



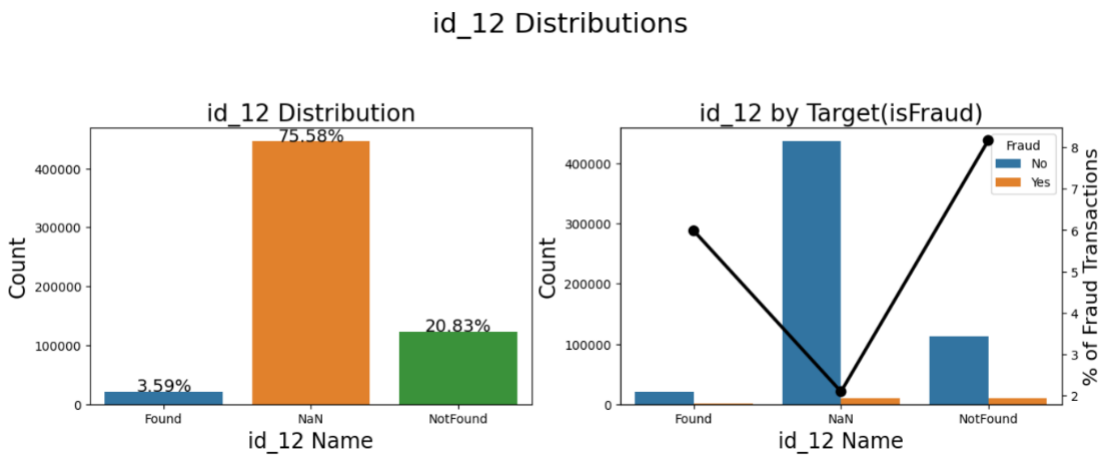
Với các cột giá trị liên tục, bằng cách sử dụng điền giá trị bị thiếu với 1 giá trị nhỏ hơn giá trị nhỏ nhất của cột hoặc lớn hơn giá trị lớn nhất của cột thì chúng ta có thể tạo ra 1 nhãn mới không trùng với các giá trị trong cột cũng như thể hiện cho giá trị missing và bắt cho mô hình phải học giá trị missing đó như 1 giá trị thực vì nó có tính phân loại khá tốt

Dữ liệu rời rạc (categorical) cũng đều có phân bố mất cân bằng vì hầu như đều tập trung vào 1 điểm dữ liệu, ở đây là tập trung vào giá trị NULL



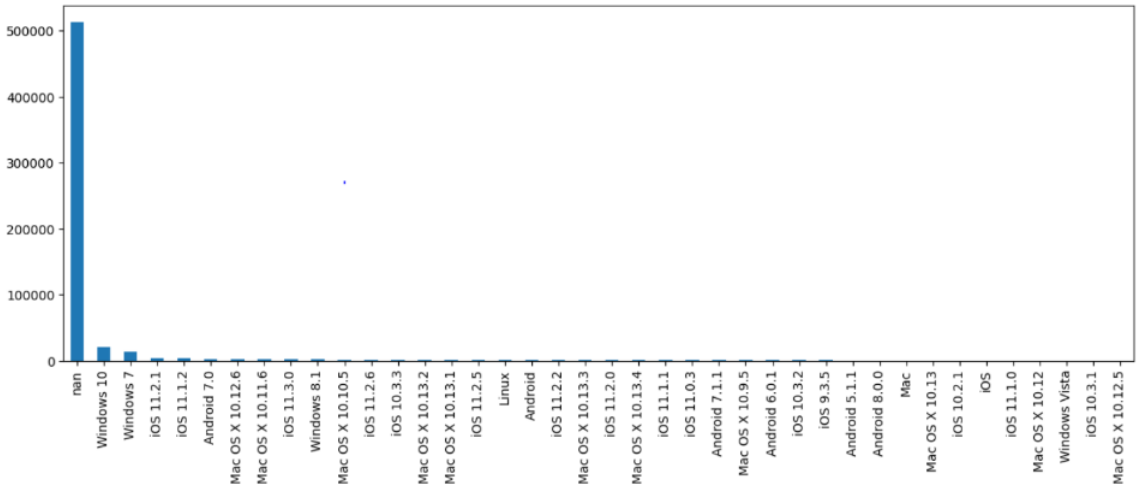
Nếu chúng ta giảm threshold như đã nhắc tới ở trên là 90% dữ liệu bị missing sẽ loại bỏ cột xuống còn 70% hay 80% thì hầu như tất cả các cột ở bảng định danh sẽ bị loại bỏ và không còn giữ lại được những thông tin có ích, tuy nhiên thì khi visualize dữ liệu thì chúng ta có thể thấy rằng giá trị bị missing ở các cột cũng có phần trăm số lượng missing khác biệt so với các giá trị khác trong cột

Vì vậy chúng ta cũng có thể sử dụng cách tạo nhãn mới để biểu diễn cho giá trị missing vì nó vẫn mang tính phân loại. Ví dụ như phân phối của id12

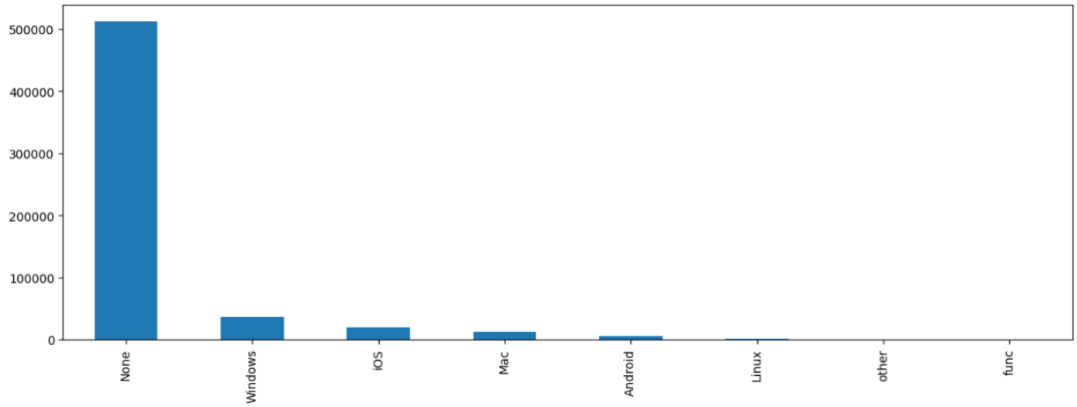


Phần trăm giá trị bị missing ở các cột có sự khác nhau khá rõ, chúng ta có thể sử dụng để phân loại khá tốt

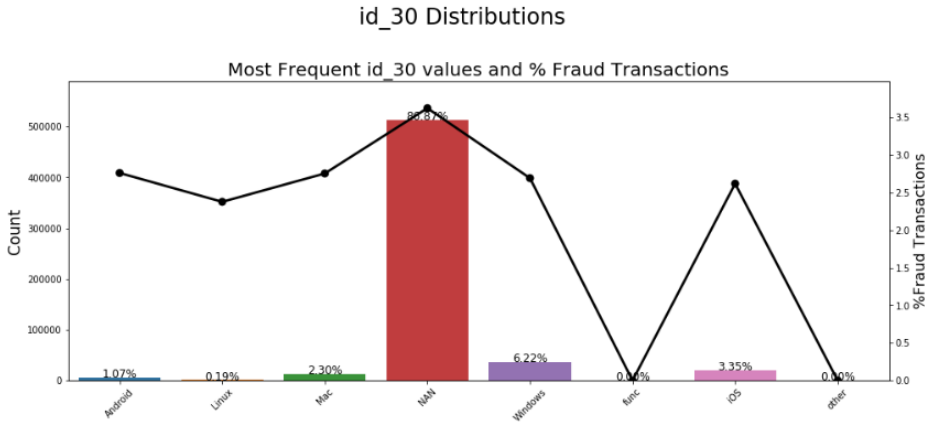
Tiếp theo 1 số cột mặc dù là giá trị phân loại (categorical) nhưng lại có số lượng giá trị riêng khá lớn, ví dụ như cột id30



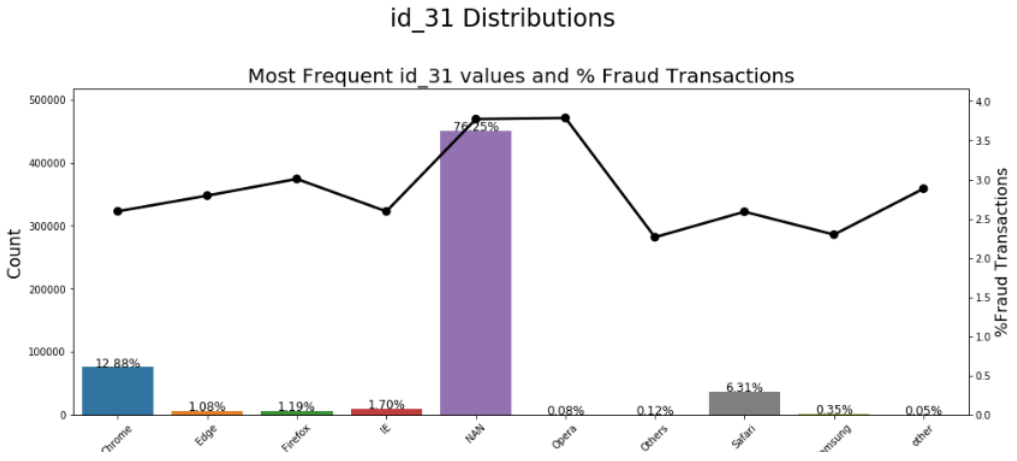
20 giá trị có tần suất xuất hiện nhiều nhất, và sau đó còn khoảng 1000 giá trị với tần suất xuất hiện $< 0{,}1\%$. Vì vậy có chúng ta có thể gộp các giá trị có chung thuộc tính lại để cô đọng thông tin hơn, cũng như phù hợp cho việc Encoding sau này. Tuy nhiên thì vẫn phải giữ được tính phân loại tốt sau khi gộp thuộc tính, vì nếu biến đổi mà mang lại kết quả xấu hơn trước đó thì việc đó là dư thừa Rất may thì sau khi biến đổi thì phần trăm của nhãn có sự khác biệt lớn và tương tự thì nó sẽ mang tính phân loại tốt



Các giá trị với các version được gộp lại và các giá trị quá ít riêng biệt thì tạo chung thành giá trị "Other"



Chúng ta có thể thấy phần trăm của các nhãn có tính phân loại. Tương tự với id31



3.3 Các phương pháp xử lý

3.3.1 Xử lý dữ liệu bị thiếu

- Bỏ những cột có trên 80% giá trị NaNs
- Điền bằng mean hoặc mode, hoặc điền theo phân bố xác suất khi tỉ lệ dữ liệu bị thiếu trong cột đó rất nhỏ và phân bố không có gì quá nổi bật

- Tạo một nhãn mới cho các giá trị NaNs ở cột categorical khi tỉ lệ giá trị bị thiếu ở mức vừa phải

- Điền theo phương pháp hồi quy (ví dụ như Random Forest Regressor) từ những cột không bị thiếu dữ liệu. Phương pháp này được sử dụng cho cột dist1. Mô hình được lựa chọn là Random Forest Regressor, có khả năng mô hình hóa các mối quan hệ phi tuyến tính trong dữ liệu. Trước tiên, chúng tôi đã xác định những cột số không có giá trị bị thiếu và những cột dạng phân loại không có giá trị thiếu. Chúng tôi đã tiến hành mã hóa các cột phân loại sử dụng kỹ thuật One-Hot Encoding và kết hợp chúng với các cột số. Sau đó, dữ liệu đã được chia thành tập huấn luyện và tập kiểm tra dựa trên giá trị của cột dist1. Mô hình Random Forest Regressor đã được huấn luyện trên tập huấn luyện để dự đoán giá trị bị thiếu trong tập kiểm tra. Để kiểm tra kết quả của việc dự đoán, chúng tôi đã tạo một biểu đồ phân tán (scatter plot) so sánh giá trị dự đoán với giá trị gốc trong cả tập dữ liệu gốc và tập dữ liệu sau khi thay thế.

3.3.2 Xử lý các cột categorical

Với những cột categorical, chúng tôi có thể xử lý bằng các cách sau

- Các cột categorical nhiều giá trị: Gộp các giá trị có ít bản ghi hoặc tạo 1 miền giá trị mới và ánh xạ tập giá trị cũ vào

- Label Encode những cột có 2 hoặc nhiều giá trị khác biệt (unique values)

- One Hot Encode những cột có ít giá trị khác biệt (3 - 5 giá trị)

3.3.3 Xử lý dữ liệu mất cân bằng

- **Synthetic Minority Oversampling Technique (SMOTE)**: Đây là phương pháp tạo thêm các dữ liệu của nhãn hiếm bằng cách sử dụng K-nearest neighbors. Nói một cách nôm na, với mỗi điểm dữ liệu trong tập dữ liệu của nhãn thiểu số, ta sẽ tìm ra k láng giềng gần nhất và nối điểm đó với các láng giềng lại với nhau. Sau đó, ta sẽ ngẫu nhiên tạo ra các điểm trên các đoạn thẳng đó.

- **Random Undersampling**: Đây là phương pháp lấy mẫu có hoặc không cần sự thay thế, lấy mẫu cho đến khi số lượng mẫu của nhãn đa số được giảm bằng số lượng mẫu của nhãn thiểu số. Phương pháp này có một nhược điểm là có thể không tổng quát hóa tốt do những hàng dữ liệu không được chọn có thể cung cấp nhiều thông tin quý giá cho việc dự đoán, nhưng với số lượng quan sát rất lớn (hơn 590 000 mẫu), chúng tôi tin rằng việc dự đoán sẽ không bị ảnh hưởng đáng kể.

- **Học có trọng số**: Một số những mô hình cho phép việc kết hợp học có trọng số các ví dụ huấn luyện điều này giúp giảm ảnh hưởng của việc mất cân bằng dữ liệu.

4 Các mô hình huấn luyện

Vì đây là bài toán phân loại, chúng tôi sẽ sử dụng những phương pháp học máy thuần cho việc phân loại bao gồm: Naive Bayes, Logistic Regression, các phương pháp kết hợp cây quyết định (tree-based methods) sử dụng kỹ thuật bagging (Random Forest) và boosting (AdaBoost, Gradient Boosting, XGBoost, CatBoost và LightGBM).

Chúng tôi sẽ không sử dụng thuật toán Support Vector Machine vì thời gian training sẽ rất lớn $\mathcal{O}(nd^2)$ với n là số quan sát và d là số thuộc tính [1].

Với những thuật toán đã được thảo luận ở trên lớp, chúng tối sẽ chỉ giới thiệu khái quát về ý nghĩa và cách hoạt động của các thuật toán đó. Còn với các thuật toán gradient boosting, chúng tôi sẽ còn đi sâu thêm vào giải thích cả phần lý thuyết toán học.

4.1 Naive Bayes

Thuật toán phân loại Naive Bayes là một trong họ hàng của những bộ phân loại đơn giản sử dụng xác suất dựa trên định lý Bayes, cùng giả định (naive) độc lập giữa các thuộc tính. Nhận một quan sát là véc-tơ đầu vào gồm các thuộc tính, Naive Bayes sẽ giả sử rằng các thuộc tính đó độc lập với nhau, và trả về xác suất của quan sát đó thuộc lớp C_i nào đó.

4.2 Logistic Regression

Thuật toán Logistic Regression là một biến thể của Linear Regression để áp dụng được vào các bài toán phân loại. Ở đây có một mối quan hệ tuyến tính, nhưng mối quan hệ tuyến tính này không phải là giữa y (target) và các x (features). Sự tuyến tính ở đây được thể hiện qua hàm **logit**, hay còn được biết đến là **log-odds** (logarithm of odds):

$$ln(x) = log(\frac{x}{1-x})$$

Thuật toán logistic regression cho rằng mối quan hệ giữa logit của đầu ra y và các đầu vào x là tuyến tính. Nói cách khác:

$$log(\frac{y}{1-y}) = \beta_0 + \beta_1 x + \cdots + \beta_n x_n$$

Sau một vài biến đổi đại số, ta có thể viết lại rằng

$$y = \frac{1}{1 + exp(-(\beta_0 + \beta_1 x + \cdots + \beta_n x_n))}$$

Ta có thể hiểu đơn giản hơn là logistic regression sử thêm một lớp hàm sigmoid tác động vào linear regression, để cho ra một kết quả từ 0 đến 1. Đây sẽ là xác suất để y thuộc lớp 1, $P(y = 1)$

4.3 Decision Tree & Random Forest

Decision tree (cây quyết định) là một thuật toán được dùng cho cả phân loại và hồi quy. Cây quyết định được xây dựng từ việc chia tập dữ liệu thành các miền không trùng nhau, sao cho mỗi miền chứa ít sự hỗn tạp (impurity), hay chứa nhiều sự đồng nhất (homogeneity) của tập dữ liệu nhất. Việc chia dữ liệu được tính bởi công thức entropy và information gain.

Random Forest (rừng ngẫu nhiên) là một thuật toán ensemble, kết hợp nhiều các cây quyết định với nhau để đưa ra một mô hình mới rất mạnh để làm giảm sự học quá khớp của cây quyết

định. Ngoài ra, các cây trong random forest sẽ chỉ được học trên tập con của các thuộc tính được lấy ngẫu nhiên, để giảm sự tương quan giữa các cây, tránh để một hoặc vài thuộc tính luôn luôn được dùng để chia trước (theo công thức information gain) dẫn đến các cây quyết định có cấu trúc giống nhau.

4.4 Các thuật toán Boosting

Trong các phương pháp bagging (như Random Forest), một mô hình học được xây dựng tách biệt nhau, điều này khiến chúng có những ưu nhược điểm riêng và không thể khắc phục, rút kinh nghiệm từ nhau. Chúng tôi tiếp tục thí nghiệm với những phương pháp boosting, với đặc điểm chung là các mô hình được xây dựng tuần tự để có thể cải thiện lẫn nhau theo thời gian.

4.4.1 AdaBoost (Adaptive Boost)

Chúng tôi sử dụng lớp AdaBoostClassifier của thư viện Scikit-learn cho bài toán này. Lớp này áp dụng thuật toán AdaBoost SAMME [7] (Stagewise Additive Modeling using a Multiclass Exponential loss function) và biến thể của nó SAMME.R (real) giúp quá trình hội tụ nhanh hơn[7]. Cũng giống như đặc điểm chung của phương pháp AdaBoost, thuật toán cũng sử dụng những mô hình phân loại yếu để học lặp đi lặp lại. Những mô hình học sau được rút kinh nghiệm từ những sai lầm của mô hình học trước nhờ việc cập nhật trọng số của các ví dụ học, tập trung hơn vào những ví du bị phân loại sai ở lần học trước. Giả sử tập dữ liệu bao gồm $K = 2$ lớp $0 - 1$ với n ví dụ $x_1, x_2, ..., x_n$ và các nhãn tương ứng $c_1, c_2, ..., c_n$. Mục tiêu của chúng ta là học được một mô hình phân loại $C(x)$ tìm được chính xác nhãn của các ví dụ. Các bước của thuật toán AdaBoost SAMME được trình bày như sau:

- Khởi tạo trọng số ban đầu cho các ví dụ học: $w_i = 1/n$ với $i = 1, 2, ..., n$
- Từ $m = 1$ cho đến M:
  - Học mô hình huấn luyện yếu $T^m(x)$ dựa trên các ví dụ và bộ trọng số w_i hiện tại.
  - Tính hàm lỗi:
$$err^m = \frac{\sum_{i=1}^{n} w_i.Equal(c_i, T^m(x_i))}{\sum_{i=1}^{n} w_i} \tag{1}$$
  - Tính trọng số của mô hình phân loại $T^m(x)$:
$$\alpha^m = \log \frac{1 - err^m}{err^m} + \log K - 1 \tag{2}$$
    Với $K = 2$:
$$\alpha^m = \log \frac{1 - err^m}{err^m} \tag{3}$$
  - Cập nhật lại bộ trọng số:
$$w_i = w_i.exp(\alpha^m.Equal(c_i, T^m(x_i))) \tag{4}$$
    với $i = 1, 2, ..., n$
  - Chuẩn hóa lại bộ trọng số (về tổng bằng 1)
- Kết quả trả về:
$$C(x) = argmax_k \sum_{m=1}^{M} \alpha^m.NotEqual(k, T^m(x)) \tag{5}$$

Với biến thể AdaBoost SAMME.R, các mô hình học yếu có trọng số bằng nhau và thay vì chúng ta dùng kết quả phân loại để đánh giá cập nhật trọng số các ví dụ, ta sẽ xét đến phân bố xác suất của các nhãn.

- Khởi tạo trọng số ban đầu cho các ví dụ học: $w_i = 1/n$ với $i = 1, 2, ..., n$
- Từ $m = 1$ cho đến M:
  - Học mô hình huấn luyện yếu $T^m(x)$ dựa trên các ví dụ và bộ trọng số w_i hiện tại.
  - Ta thu được phân bố xác suất các nhãn:

$$p_k^m(x) = Prob_w(c = k|x) \tag{6}$$

    với $k = 0, 1, ..., K - 1$
  - Tính:

$$h_k^m(x) = (K - 1)(\log p_k^m(x) - \frac{1}{K}\sum_{k'} \log p_{k'}^m(x)) \tag{7}$$

    với $k = 0, 1, ..., K - 1$. Với $K = 2$:

$$h_k^m(x) = \log p_k^m(x) - \frac{1}{2}\sum_{k'} \log p_{k'}^m(x) \tag{8}$$

    với $k = 0, 1$.
  - Cập nhật lại bộ trọng số:

$$w_i = w_i.exp(-\frac{K-1}{K} y_i^T \log p_k^m(x_i)) \tag{9}$$

    với $i = 1, 2, ..., n$, y_i là one-hot vector nhãn của x_i. Với K =2:

$$w_i = w_i.exp(-\frac{1}{2} y_i^T \log p_k^m(x_i)) \tag{10}$$

  - Chuẩn hóa lại bộ trọng số (về tổng bằng 1)
- Kết quả trả về:

$$C(x) = argmax_k \sum_{m=1}^{M} h_k^m(x) \tag{11}$$

Các thuật toán AdaBoost có thể làm việc với nhiều loại mô hình học cơ bản khác nhau. Chúng tôi sử dụng cây phân loại DecisionTreeClassfier là những mô hình học yếu để áp dụng AdaBoost.

4.4.2 GradientBoosting

Gradient Boosting[3] là tên gọi chung cho các thuật toán boosting làm việc với những hàm mất mát khả vi. Chúng tôi sử dụng lớp GradientBoostingClassification của thư viện scikit-learn. Thuật toán có một số đặc điểm nổi bật sau:

- Các mô hình học yếu (cơ bản): là những cây quyết định (decision trees) có độ dài bị giới hạn.
- Hàm mất mát: Phải có tính khả vi

- Mô hình additive: Các mô hình học (cây) mới sẽ được thêm vào từng bước mà không thay đổi những mô hình (cây) trước đó, để tạo thành một mô hình cuối cùng chính xác nhất có thể.

$$C_M(x_i) = \sum_{m=1}^{M} T^m(x_i) \tag{12}$$

$$C_m(x_i) = C_{m-1}(x_i) + T^m(x_i) \tag{13}$$

- Chiến lược Gradient Descent được học để học một cây mới, với mục tiêu cực tiểu hóa hàm lỗi khi thêm vào mô hình mới.

$$T^m = argmin_T L_m = argmin_T \sum_{i=1}^{n} l(c_i, C_{m-1}(x_i) + T^m(x_i)) \tag{14}$$

Sử dụng ước lượng Taylor bậc 1

$$l(z) = l(a) + (z-a).l(a)'_a \tag{15}$$

Ta thu được:

$$l(c_i, C_{m-1}(x_i) + T^m(x_i)) = l(c_i, C_{m-1}(x_i)) + T^m(x_i).l(c_i, C_{m-1}(x_i))'_{C_{m-1}(x_i)} \tag{16}$$

Đại lượng $l(c_i, C_{m-1}(x_i))'_{C_{m-1}(x_i)}$chính là đạo hàm của hàm mất mát dựa trên tham số thứ hai là $C_{m-1}(x_i)$ và có thể dễ dàng tính được. Ta ký hiệu nó là g_i. Khi đó, lược bỏ các đại lượng hằng số:

$$T^m = argmin_T \sum_{i=1}^{n} T^m(x_i) g_i \tag{17}$$

Hàm này được cực tiểu hóa khi T^m được học để dự đoán đại lượng tỷ lệ với gradient âm $-g_i$. Đó chính là nguyên lý của thuật toán Gradient Boosting.

4.4.3 XGBoost

XGBoost, hay còn được gọi là Extreme Gradient Boosting, là một thuật toán học máy mạnh mẽ được sử dụng rộng rãi cho các nhiệm vụ có dữ liệu có cấu trúc như dự đoán, phân loại, và hồi quy. XGBoost thuộc loại thuật toán Ensemble Learning, nó kết hợp nhiều cây quyết định yếu (weak decision trees) để tạo thành một mô hình mạnh hơn và nó là 1 biến thể của Gradient Boosting

Trong thuật toán này, các cây quyết định được tạo 1 cách tuần tự, trọng số đóng 1 vai trò quan trọng trong việc tạo cây. Tại mỗi thời điểm, trọng số được gán cho toàn bộ các biến độc lập, sau đó được đưa vào cây quyết định để dự đoán kết quả. Trọng số của các biến bị dự đoán sai bởi cây quyết định sẽ được tăng lên và sẽ được đưa vào cây quyết định thứ 2, sau đó thì các cây quyết định riêng lẻ này sẽ được kết hợp lại để đưa ra 1 mô hình mạnh mẽ và chính xác hơn [2].

Trước khi bắt đầu giải thích về phần toán của XGBoost, chúng ta nhắc lại 1 chút về cây quyết định CART

Những dự đoán của các cây quyết định riêng lẻ được viết theo công thức sau:

$$\hat{y_i} = \sum_{k=1}^{K} f_k(x_i), f_k \in \mathcal{F} \tag{18}$$

Với K là số cây, f là hàm của tập F, F là 1 tập các cây quyết định CART. Hàm mục tiêu của mô hình được viết theo công thức sau:

$$obj(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{19}$$

Ở đây, thành phần đầu tiên là hàm mất mát (loss function) và thành phần thứ hai là tham số điều chỉnh (regularization parameter). Khi tham số điều chỉnh được đặt bằng không,hàm mục tiêu quay trở lại phương pháp truyền thống của việc tạo cây quyết định dựa trên gradient boosting [2].

Thay vì học cây quyết định một lần và điều này làm cho quá trình tối ưu hóa phức tạp hơn, chúng ta áp dụng chiến lược cộng dồn, tối thiểu hóa mất mát của những gì chúng ta đã học và thêm vào một cây quyết định mới, điều này có thể được thực hiện như sau [2]:

$$\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\
&.... \\
\hat{y}_i^{(t)} &= \textstyle\sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned}$$

Hàm mục tiêu được viết lại như sau:

$$\begin{aligned}
obj^{(t)} &= \textstyle\sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) \\
&= \textstyle\sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + constant
\end{aligned}$$

$$\begin{aligned}
obj^{(t)} &= \textstyle\sum_{i=1}^{n} (y_i - (\hat{y}_i^{(t-1)} + f_t(x_i)))^2 + \sum_{i=1}^{t} \Omega(f_i) \\
&= \textstyle\sum_{i=1}^{n} [2(\hat{y}_i^{(t-1)} - y_i) f_t(x_i) + f_t(x_i)^2] + \Omega(f_t) + constant
\end{aligned}$$

Chúng ta có thể sử dụng đạo hàm bậc 2 để xấp xỉ hàm mục tiêu

$$\mathcal{L} = \sum_{i=1}^{n} [l(y_i, \hat{y}^{(t-1)} + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \tag{20}$$

Trong đó g_i và h_i lần lượt là đạo hàm bậc 1 và đạo hàm bậc 2 của hàm lỗi. Sau khi tối giản và loại bỏ hằng số:

$$\mathcal{L} = \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \tag{21}$$

Tối ưu của XGBoost [2]:

- Regularization: Vì cây quyết định có thể phức tạp, XGBoost sử dụng regularization để thưởng phạt với mô hình quá phức tạp
- Parallelization and Cache block: Trong XGBoost, chúng ta không thể huấn luyện nhiều cây một cách song song, nhưng nó có thể tạo các nút khác nhau của cây một cách song song. Để làm điều này, dữ liệu cần phải được sắp xếp theo thứ tự. Để giảm chi phí của việc sắp xếp, nó lưu trữ dữ liệu theo dạng khối.

- Cắt tỉa cây (Tree Pruning): XGBoost sử dụng tham số max-depth để xác định tiêu chí dừng cho việc chia nhánh và bắt đầu cắt tỉa cây ngược.
- Cache-Awareness and Out-of-core Computation: Thuật toán này được thiết kế để tận dụng hiệu quả tài nguyên phần cứng. Điều này được thực hiện thông qua nhận thức về bộ nhớ đệm bằng cách cấp phát các bộ đệm nội tại trong từng luồng để lưu trữ thống kê gradient.

4.4.4 LightGBM

Các model thuộc nhóm Boosting đã chứng minh dược độ hiệu quả khi đối mặt với dữ liệu dạng bảng (GradientBoosting, XGBoost,...) nhưng nó gặp vấn đề khi phải làm việc với những dữ liệu lớn về cả mặt số lượng samples và số lượng feature vì nó thường phải duyệt qua toàn bộ dữ liệu đê có thể thực hiện thuật toán.

LightGBM được sinh ra để giải quyết vấn đề được nêu trên với 2 ý tưởng chính là GOSS(Gradient-based One-Side Sampling) và EFB (Exclusive Feature Bundling) [4]. GOSS được đề xuất để có thể giảm số lượng mẫu dữ liệu sau mỗi epoch của thuật toán Boosting trong khi EFB đươc đề xuất để giảm số lượng feature của dữ liệu.

Gradient-based One-Side Sampling

GOSS được đề xuất để loại bỏ đi những mẫu dữ liệu mang ít thông tin sau mỗi epoch của thuật toán Boosting. Nó dựa trên Gradient cho mỗi mẫu dữ liệu của thuật toán GBDT (Gradient Boosting Decision Tree) với ý tưởng là mẫu dữ liệu nào có gradient nhỏ tức là nó đã được học tốt. Và ở đây , họ sẽ thực hiện loại bỏ những mẫu dữ liệu như vậy và tập trung vào những dữ liệu khó học trong khi vẫn dữ được phân bố ban đầu của tập dữ liệu [4].

Algorithm 1: Gradient-based One-Side Sampling (GOSS)

Input: I: training data, d: iterations
Input: a: sampling ratio of large gradient data
Input: b: sampling ratio of small gradient data
Input: *loss*: loss function, L: weakLearner

$model \leftarrow \{\}, fact \leftarrow \frac{1-a}{b}$
$topN \leftarrow a \times \text{len}(I), randN \leftarrow b \times \text{len}(I)$
for $i \leftarrow 1$ **to** d **do**
  $preds \leftarrow$ models.predict(I)
  $g \leftarrow \text{loss}(I, preds)$, $w \leftarrow \{1, 1, \ldots\}$
  $sorted \leftarrow$ getSortedByIndices(abs(g))
  $topSet \leftarrow sorted[1 : \text{topN}]$
  $randSet \leftarrow \text{randomSample}(sorted[\text{topN} :], \text{randN})$
  $usedSet \leftarrow \text{topSet} + \text{randSet}$
  $w[\text{randSet}] \leftarrow w[\text{randSet}] \times \text{fact}$
  $newModel \leftarrow L(I[\text{usedSet}], -g[\text{usedSet}], w[\text{usedSet}])$
  $model.append(newModel)$
end

Alg.1 mô tả cách hoạt động của thuật toán , cách để bảo toàn phân bố dữ liệu được thể hiện trong dòng 4 và dòng 5 của đoạn mã giả.

Exclusive Feature Bundling (EFB)

- Exclusive Feature: 1 cặp feature mà giá trị của chúng không bao giờ nhận cùng 1 giá trị khác 0

- A Bundle: Một thùng chứa các Exclusive Feature

Dữ liệu nhiều chiều thường rất thưa và thường chứa những Exclusive Feature. Ở đây, EFD được thiết kế để tạo ra những Bundle đồng thời merge những Feature bên trong 1 Bundle thành 1. Qua đó giảm kích thước dữ liệu từ (data × features) xuống thành (data × bundles) với bundles $\ll$ features [4].

1. **Thuật toán 1:** Xây dựng các Bundle

   - Ở đây xây dựng 1 đồ thị G = <V,E> là 1 dạng fully connected graph với các node là các feature và trọng số của các cạnh là độ conflict giữa 2 features (Tổng số cặp giá trị bằng nhau giữa 2 feature)
   - Ở đây với mỗi Bundle, thay vì ràng buộc tất cả feature bên trong nó đều là Exclusive Feature ,họ nới lỏng ràng buộc bằng cách thêm 1 tham số K (số lượng conflict tối đa bên trong 1 bundle)
   - Duyệt qua từng node(feature) theo thứ tự trọng số từ cao đến thấp (trọng số được hiểu là tổng weight tương ứng của mỗi node và bằng cách duyệt như vậy,phần nào đó sẽ giúp thuật toán hội tụ nhanh hơn) sau đó kiểm tra điều kiện và đưa ra quyết định xem node(feature) đó sẽ được đưa vào 1 bundle có sẵn hay tạo ra 1 bunble mới .

   Algorithm 2: Create Bundles

   Input: F: features, K: Max Conflict Count
   Construct Graph G
   $searchOrder \leftarrow G.sortByDegree()$
   $bundles \leftarrow \{\}, bundlesConflict \leftarrow \{\}$
   for i ***in*** $searchOder$ **do**
   $needNew \leftarrow False$
   for $j = 1$ **to** $len(bundles)$ **do**
   $cnt \leftarrow ConflictCount(bundles[j], F[i])$
   if *cnt + bundlesConflict[j]* $< K$ **then**
   $needNew \leftarrow False$
   bundles[j].append(F[i])
   end
   end
   if $needNew$ **then**
   Add F[i] as a new bundle to bundles
   end
   end

2. **Thuật toán 2** : Gộp feature bên trong 1 bundle
   Sau khi chia được các feature vào trong các bundles , chúng ta cần 1 phương pháp để gộp các feature này lại làm 1 sao cho từ feature mới được tạo ra chúng ta có thể suy luận ngược lại các feature ban đầu.

   - Tính toán offset cần được cộng vào cho mỗi feature
   - Lặp qua toàn bộ dữ liệu và feature
   - Khởi tạo 1 vector thuộc tính mới cho từng mẫu dữ liệu
   - Cập nhật vector thuộc tính bằng cách công giá trị offset vào giá trị ban đầu

Feature1	Feature2	FeatureBundle
1	0	4
2	0	5
0	1	1
0	2	2
0	3	3

Bảng 1: Example Of Feature Bundling

Algorithm 3: Gộp Feature bên trong 1 bundle

Input: *numdata*:Số lượng dữ liệu
Input: F :1 Bundle chứa các Exclusive Feature
binRanges $\leftarrow \{0\}, totalBin \leftarrow 0$
for f ***in*** F **do**
  totalBin += f.numBin
  binRanges.append(totalBin)
end
newBin ←new Bin(*numdata*)
for $i \leftarrow 1$ **to** *numdata* **do**
  newBin[i] $\leftarrow 0$
  for $j \leftarrow 1$ **to** *len(F)* **do**
    if *F[j].bin[i]* $\neq 0$ **then**
      newBin[i] = F[j].bin[i] + binRanges[j]
    end
  end
end

4.4.5 CatBoost

CatBoost là một framework sử dụng gradient boosting để đưa ra dự đoán, nhưng nhỉnh hơn các thuật toán gradient boosting thông thường ở thời gian chạy nhanh hơn.

Điểm khác biệt lớn nhất giữa CatBoost và Gradient Boosting thông thường là CatBoost mã hóa các cột categorical theo phương pháp Ordered Target Encoding thay vì sử dụng One Hot hoặc Label Encoding. Điều này để phòng tránh hiện tượng data leakage, đó là khi dữ liệu học chứa các thông tin về kết quả đang hướng đến, nhưng loại dữ liệu như vậy thường sẽ không có sẵn khi mô hình được áp dụng để dự đoán trong thực tế [5].

Ordered Target Encoding

Ordered Target Encoding sẽ giả sử các hàng của dữ liệu được xếp theo thứ tự (sequentially), và dùng công thức sau để mã hóa các cột categorical:

$$\text{Cat Boost Encoding} = \frac{\text{OptionCount} + \text{prior}}{n+1}$$

với n là số hàng dữ liệu ở trước hàng đang xét có giá trị của cột thuộc tính đang xét giống với hiện tại, và Option Count sẽ là số hàng ở đằng trước có giá trị của cột thuộc tính đang xét bằng giá trị hiện tại và nhãn đầu ra bằng 1 [5].

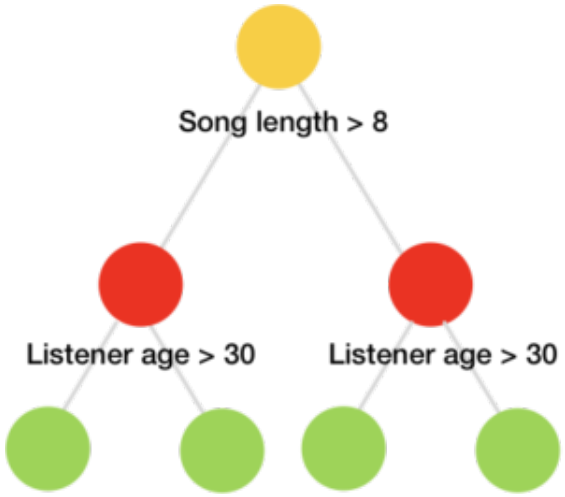
Favorite Color	Height (m)	Loves Troll 2
0.05	1.77	1
0.05	1.32	0
0.05	1.81	1
Blue	1.56	0
Green	1.64	1
Green	1.61	0
Blue	1.73	0

$$\text{CatBoost Encoding} = \frac{\text{OptionCount} + 0.05}{n + 1}$$

n = Number of rows that have already been seen that have the same value for Favorite Color

Ví dụ như ở bức ảnh trên, ở hàng thứ 4, $n = 1$ vì hàng đầu tiên có chứa thuộc tính Blue. OptionCount = 1 vì chỉ quan sát đầu tiên có thuộc tính Blue và nhãn bằng 1 (thích Troll 2). Vì vậy, encoding của hàng này sẽ là $(1 + 0.05)/(1+1) = 0.325$ (prior ở đây đã được chọn làm 0.05)

Symmetrical trees



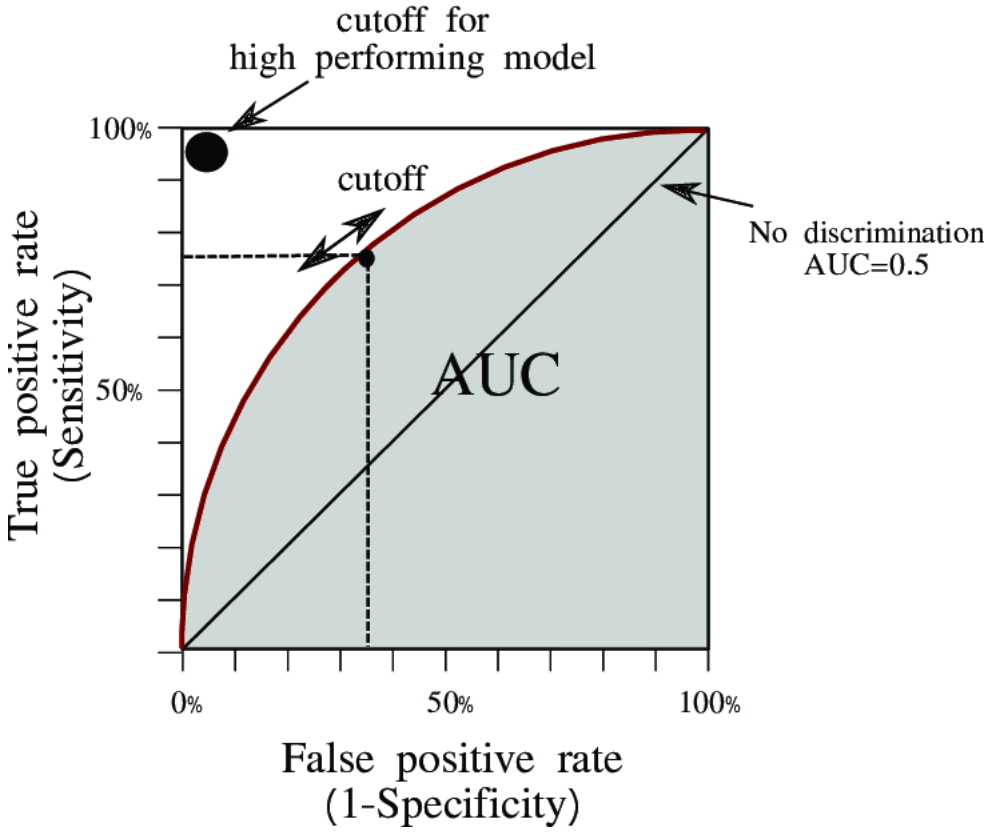
Điểm khác biệt thứ hai giữa CatBoost và các thuật toán boosting thông thường là CatBoost sử dụng cấu trúc cây đối xứng (symmetrical trees) để xây dựng các cây quyết định. Điều này có nghĩa là ở mỗi tầng của cây quyết định, ở bất kỳ nhánh nào (trái hay phải), thuộc tính và giá trị dùng để chia dữ liệu sẽ giống nhau [6]. Điều này sẽ làm tối ưu hơn thời gian và tài nguyên dùng để chạy mô hình, vì mô hình sẽ hỏi cùng một câu hỏi để chia dữ liệu ở bất kỳ nhánh nào. Vì vậy, ta sẽ không cần phải lưu trữ những câu hỏi khác nhau ở các nhánh khác nhau.

Ngoài ra, CatBoost còn có các tính năng có sẵn tự động xử lý các dữ liệu bị thiếu bằng phương pháp Symmetric Weighted Quantile Sketch và tự động scale các giá trị về cùng một khoảng, tiết kiệm thời gian tiền xử lý cho chúng ta [6].

5 Kết quả và đánh giá

5.1 Độ đo và phương pháp đánh giá

Đối với sự mất cân bằng dữ liệu theo nhãn, ta sẽ sử dụng tiêu chí ROC-AUC để đánh giá các mô hình học máy phân loại hai lớp. Như đã thảo luận trong khóa học, việc sử dụng độ chính xác (accuracy) cho các dữ liệu mất cân bằng là không được khuyến khích, tại vì một mô hình chỉ cần trả ra kết quả toàn là các nhãn ở lớp phổ biến cũng có thể đạt được chỉ số accuracy đáng kể.



Đường cong ROC (Receiver Operating Curve) được vẽ với FPR (False Positive Rate) ở trục hoành và TPR (True Positive Rate) ở trục tung ứng với các ranh giới khác nhau. AUC là phần diện tích phía dưới của ROC, thể hiện cho sự phân loại của hai lớp. AUC càng cao (càng gần tới 1), mô hình học máy càng thể hiện tốt (dự đoán những nhãn 0 chĩnh xác là 0 và 1 chính xác là 1), và AUC càng thấp thì mô hình sẽ dự đoán sai hoàn toàn (những nhãn 0 sẽ bị dự đoán là 1 và ngược lại). AUC = 0.5 thể hiện rằng mô hình này không có khả năng phân loại nhãn tốt.

Ngoài ra, ta cũng quan tâm đến độ chính xác (precision) và và độ phủ (recall) trong việc phát hiện giao dịch lừa đảo (nhãn 1). Với tính chất của bài toán này, việc dự đoán một giao dịch lừa đảo là hợp lệ sẽ có hậu quả nguy hiểm hơn là dự đoán giao dịch hợp lệ là lừa đảo. Vì muốn mô hình của ta dự đoán được nhiều giao dịch lừa đảo trên thực tế hơn, precision và recall của nhãn 1 cũng sẽ được đưa vào để xem xét và đánh giá các mô hình.

5.2 Tập dữ liệu

Từ tập dữ liệu đã được xử lý, chúng tôi chia thành nhỏ thành hai tập huấn luyện (train) và kiểm tra (test) theo tỷ lệ 80-20 sao cho đảm bảo được phân bố nhãn của tập ban đầu vào hai tập con. Các mô hình áp dụng các thuật toán và các kỹ thuật khác nhau sau khi được học trên

tập huấn luyện sẽ được kiểm tra và đánh giá kết quả trên tập test.

5.3 Kết quả và đánh giá các mô hình

5.3.1 Áp dụng Random Undersampling

Mô hình	Fraud Precision	Fraud Recall	AUC
Multinomial Naive Bayes	0.08	0.55	0.7116
Gaussian Naive Bayes	0.09	0.53	0.7656
Logistic Regression	0.13	0.74	0.8551
Decision Tree	0.14	0.76	0.8632
Random Forest	0.38	0.74	0.9370
AdaBoost	0.42	0.77	0.9458
GradientBoosting	0.43	0.81	0.9598
XGBoost	0.43	0.81	0.9586
LightGBM	0.41	0.84	0.9649
CatBoost	0.43	0.82	0.9608

Bảng 2: Kết quả của các mô hình sử dụng phương pháp Random Undersampling

Bảng 2 mô tả những kết quả thí nghiệm của chúng tôi với các mô hình học áp dụng phương pháp Random Undersampling để cân bằng dữ liệu. Ở một số mô hình như DecisionTree, Random Forest, Gradient Boosting,... chúng tôi kết hợp với việc học các ví dụ có trọng số với tham số class_weight = balance. Ưu điểm của phương pháp Random Undersampling nằm ở việc thu nhỏ đáng kể tập dữ liệu khiến tốc độ học được cải thiện rất nhiều, từ đó tạo điều kiện cho chúng tôi tunning các siêu tham số (hyperparameter). Bảng 2 là tập hợp những kết quả tốt nhất của chúng tôi với các mô hình học sau quá trình tunning.

- **Naive Bayes**: Naive Bayes thường hay được sử dụng làm mô hình đầu tiên để chạy trong các bài toán phân loại do giả sử độc lập của các thuộc tính dẫn đến thời gian chạy khá nhanh. Ở đây, Multinomial Naive Bayes đã cho recall của nhãn 1 (Fraud) là 0.55 với AUC vào khoảng 0.71. Gaussian Naive Bayes đã có sự thể hiện tốt hơn khi có AUC cao hơn một chút so với Multinomial Naive Bayes.

- **Logistic Regression**: Khá là ngạc nhiên khi sử dụng Logistic Regression đã cho ra một kết quả recall và AUC khá ấn tượng, lần lượt là 0.74 và 0.85, thể hiện một sự tiến bộ rõ rệt hơn với Naive Bayes. Sự cải thiện rõ rệt này có thể bởi vì Logistic Regression đã xem xét sự cần thiết của các thuộc tính (bằng cách tính các hệ số (coefficients) của các cột), làm cho mô hình có thể học được đâu là những thuộc tính quan trọng và đâu là những thuộc tính ít cần thiết hơn. Naive Bayes đã không tận dụng điều này và chỉ đếm số lần xuất hiện của các sự kiện, cùng với giả thuyết độc lập naive, trong khi dữ liệu thực tế có thể có sự tương quan giữa các cột với nhau.

- **Decision Tree**: Decision Tree cũng cho thấy độ chính xác (precision) thấp (0.14) như các mô hình trước, điều này có nghĩa rằng mô hình này còn tồn tại nhiều sai sót khi dự đoán các trường hợp là gian lận. Tuy nhiên, recall (0.76) khá cao, cho thấy mô hình có khả năng phát hiện nhiều trường hợp gian lận, nhưng có thể có nhiều dự đoán sai về trường hợp bình thường. AUC-ROC (0.863) khá tốt và gần 1, cho thấy mô hình có khả năng phân loại tốt giữa các lớp.

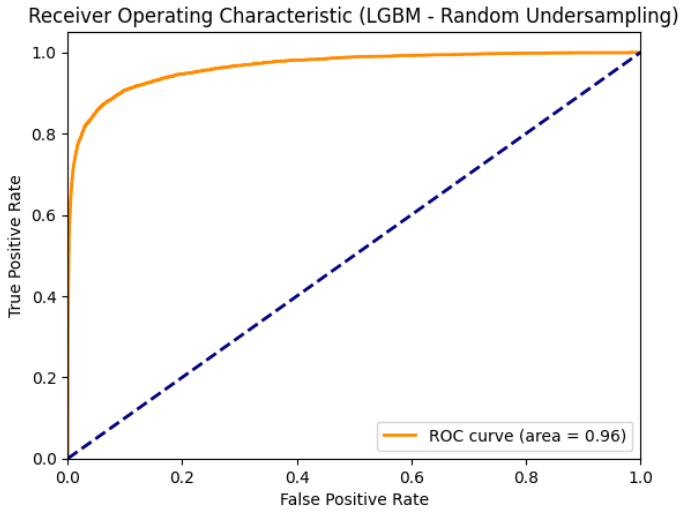
- **Random Forest**: Random Forest thể hiện độ chính xác (precision) cao rất nhiều hơn so với Decision Tree (0.38), đồng nghĩa với việc mô hình này có khả năng giảm thiểu số lượng dự đoán sai lớp gian lận. Recall (0.74) vẫn khá cao, cho thấy khả năng phát hiện các trường hợp gian lận vẫn tốt. AUC-ROC (0.937) cũng cao hơn so với Decision Tree, đây là một điểm mạnh cho mô hình Random Forest. Tổng quan, Random Forest có hiệu suất tốt hơn so với Decision Tree trong trường hợp này. Tuy Decision Tree có recall nhỉnh hơn, nhưng Random Forest cung cấp độ chính xác tốt hơn và AUC-ROC score cao hơn, cho thấy khả năng phân loại và phát hiện gian lận của nó mạnh hơn.

- **AdaBoost**: AdaBoost tiếp tục cải thiện độ chính xác trong việc phát hiện giao dịch lừa đảo, lên đến 0.42 đồng thời AUC 0.94 cũng cao hơn tất cả các phương pháp trước đó

- **Grafient Boosting và XGBoost** : Kết quả của hai mô hình khá tương đồng. Độ chính xác của nhãn Fraud vẫn khá thấp dù tiếp tục được cải thiện (0.43). Lý do có thể việc undersample quá mức khiến cho tỷ lệ hai nhãn không hợp lý và mô hình có xu hướng trả lại nhiều kết quả lừa đảo hơn.

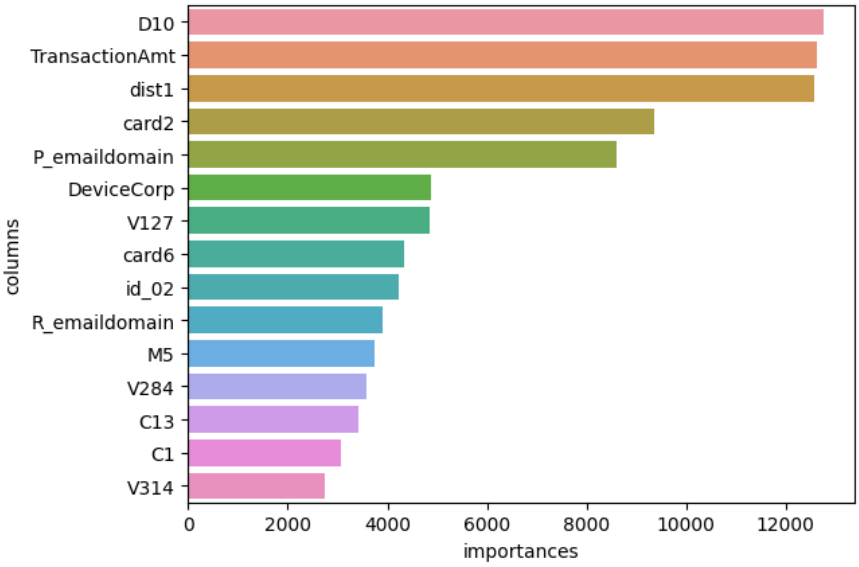
- **LightGBM** : Mô hình gây ấn tượng khi có độ phủ dự đoán giao dịch lừa đảo cao 0.84 mặc dù có sự thua kém độ chính xác một chút so với những mô hình Boosting trước đó. Nhìn chung, đây là một mô hình tốt khi có AUC lên đến 0.96.

- **CatBoost**: Cũng là một mô hình tốt với AUC 0.96. CatBoost có độ phủ hơi thấp hơn LightGBM (0.82) nhưng lại có độ chính xác cao nhất trong số các phương pháp Boosting (0.43).



Hình 1: ROC curve of LightGBM model

Nói tóm lại, đặc điểm chung của các mô hình được áp dụng cùng với phương pháp Random Undersampling là có độ phủ dự đoán các giao dịch lừa đảo khá cao đổi lại độ chính xác còn thấp. Các mô hình Boosting cho kết quả ấn tượng nhất, tiêu biểu là mô hình LighGBM với độ phủ cao nhất. Xét trong bài toán phát hiện giao dịch lừa đảo thì chúng ta có thể chấp nhận được kết quả này để áp dụng trong thực tế khi mô hình phát hiện được các giao dịch lừa đảo vởi xác suất cao. Việc khảo sát độ quan trọng của các thuộc tính cũng cho chúng tôi những dự đoán về ý nghĩa thực sự của các cột này.



Hình 2: Các thuộc tính quan trọng của mô hình LightGBM

5.3.2 Áp dụng SMOTE

Mô hình	Fraud Precision	Fraud Recall	AUC
Multinomial Naive Bayes	0.08	0.55	0.7124
Gaussian Naive Bayes	0.08	0.58	0.7286
Logistic Regression	0.12	0.67	0.8321
Decision Tree	0.43	0.52	0.7469
Random Forest	0.92	0.52	0.9299
AdaBoost	0.84	0.58	0.8976
GradientBoosting	0.85	0.51	0.9305
XGBoost	0.92	0.58	0.9569
LightGBM	0.89	0.60	0.9587
CatBoost	0.95	0.62	0.9664

Bảng 3: Kết quả của các mô hình khi sử dụng phương pháp SMOTE

Bảng 3 mô tả các kết quả làm việc của chúng tôi áp dụng phương pháp SMOTE để cân bằng dữ liệu huấn luyện với các mô hình học khác nhau. Vì SMOTE khiến cho tập dữ liệu huấn luyện vốn đã rất lớn nay càng lớn hơn, chúng tôi quyết định không thực hiện quá trình tunning ở bước này và tiến hành tham khảo cách đặt siêu tham số từ kết quả tunning của Random Undersampling.

- **Naive Bayes**: Các kết quả của Multinomial Naive Bayes tương đồng với khi sử dụng phương pháp Random Undersampling. Tuy nhiên, AUC của Gaussian Naive Bayes lại giảm từ 0.76 xuống 0.72, dù cho có sự cải thiện ở độ phủ khi phát hiện các giao dịch lừa đảo. Những sự thay đổi này là không quá đáng kể và có thể là do sự biến thiên và ngẫu nhiên của dữ liệu khi dùng SMOTE.
- **Logistic Regression**: Logistic Regression đã có một sự giảm nhẹ trong chỉ số AUC từ 0.85 xuống còn 0.83, độ phủ và độ chính xác cũng có sự thụt lùi so với khi sử dụng Random Undersampling.

- **Decision Tree**: Decision Tree khi áp dụng SMOTE đã cải thiện rất nhiều độ chính xác (precision) so với phiên bản sử dụng Random Undersampling, tăng từ 0.14 lên 0.43. Trái lại, độ phủ lại giảm mạnh xuống 0.52, mô hình mất đi khả năng tìm ra được các giao dịch lừa đảo.
- **Random Forest**: Random Forest khi áp dụng SMOTE thể hiện độ chính xác (precision) rất cao (0.92), nhưng độ phủ vẫn chỉ ở mức trung bình 0.52. AUC-ROC (0.93) gần tiến tới 1, thể hiện khả năng phân loại của mô hình rất tốt.
- **Các phương pháp Boosting**: Do việc oversampling bằng SMOTE đã làm tăng số nhãn của hẫn thiểu số và giúp cho mô hình có thêm dữ liệu để học, điều này khiến cho các mô hình Boosting cũng có các tính chất của mô hình Random Forest, với độ chính xác cao nhưng độ phủ trung bình trong bài toán phát hiện các giao dịch lừa đảo. Catboost là mô hình nổi trội nhất khi sử dụng SMOTE, đứng đầu ở tất cả các độ đo AUC, precision và recall.

Nói tóm lại, ngoại trừ các phương pháp Naive Bayes và Logistic Regression không cho thấy nhiều sự thay đổi trong kết quả, các mô hình khác khi áp dụng SMOTE cho thấy kết quả tằng đáng kể ở độ chính xác và giảm mạnh độ phủ về mức trung bình ở nhãn 1. Có thể nói, SMOTE giúp tăng độ chính xác nhưng có thể làm giảm khả năng phát hiện gian lận so với Random Undersampling. Xét trong bài toán phát hiển giao dịch lừa đảo, chúng tôi ưu tiên chọn phương pháp Random Undersampling vì việc để sót nhiều sẽ gây ra những hậu quả lớn đến nhiều bên trong thực tiễn.

6 Kết luận và Đề xuất

6.1 Kết luận

Tổng kết lại, với cách phân tích, tiền xử lý và sử dụng những mô hình mà chúng tôi đề xuất, kết quả phát hiện được những giao dịch lừa đảo là tương đối ổn. Các phương pháp Ensemble đã cho ra kết quả vượt trội đáng kể so với những phương pháp thông thường như Naive Bayes và Logistic Regression như những gì chúng tôi đã được học và dự đoán, bởi sự kết hợp của rất nhiều learners đã cho một mô hình tổng thể rất mạnh. Một lần nữa, các phương pháp Boosting thể hiện rất tốt, nhỉnh hơn một chút so với Random Forest bởi các cây trong Boosting được học và rút kinh nghiệm từ nhau.

Áp dụng kỹ thuật Random Undersampling, chúng tôi thu được các mô hình ở đây đều có độ phủ của nhãn Fraud (1) khá cao, với sự hi sinh của độ chính xác. Điều này nằm trong dự liệu của chúng tôi, bởi vốn dĩ dữ liệu đã rất mất cân bằng, nên chúng tôi chấp nhận hi sinh độ chính xác ở một vài mô hình (đoán giao dịch là Fraud nhiều hơn, dù có sai hay đúng) để tăng độ phủ, tăng khả năng phát hiện các giao dịch lừa đảo tránh gây thiệt hại lớn. Mô hình được chúng tôi đánh giá là phù hợp nhất với bài toán là LightGBM khi sử dụng Random Undersampling.

6.2 Đề xuất

Do thời gian và nguồn lực có hạn, chúng tôi đã chưa kịp thử hết một vài phương pháp và cách xử lý dữ liệu tối ưu hơn. Nếu có thời gian, chúng tôi sẽ tiến hành thử thêm những ý tưởng sau:

6.2.1 Khám phá thêm dữ liệu

- **Fraudulent Clients**: Lời giải đạt được top 1 của cuộc thi này trên Kaggle đã phát hiện ra được một điều thú vị sau khi khám phá dữ liệu: Thực ra chúng ta không đang cố gắng đoán một giao dịch là lừa đảo, mà ta đang đoán xem ai mới là người lừa đảo. Theo nhà cung cấp thông tin Vespa Corp, một khi thẻ thanh toán đã bị phát hiện là lừa đảo, mọi thông tin về tài khoản của thanh toán đó đều có isFraud = 1. Sử dụng các cột như card1, add1 và D1, họ đã có khả năng phân cụm được những giao dịch nào thuộc về người nào.

6.2.2 Các kỹ thuật feature engineering khác

- **PCA/Sparse PCA**: Ở dự án này chúng tôi sử dụng correlation analysis để chọn ra số cột V đặc trưng. Tuy nhiên, ta có thể áp dụng PCA để chọn ra ít cột V đặc trưng hơn nữa mà vẫn giữ được số lượng thông tin đáng kể. Vì bản chất của các cột V là không rõ nghĩa, việc mất đi ý nghĩa của các cột bằng PCA cũng không quá quan trọng.
- **Tạo ra các UID (Unique Identification)**: Dây là ý tưởng chính của lời giải top 1 trên Kaggle, sử dụng thông tin về dự đoán các cá nhân lừa đảo. Họ sẽ tạo ra một cột UID để phân cụm các giao dịch thành các người khác nhau, rồi sau đó tận dụng triệt để cột UID này để dự đoán nhãn isFraud. Dưới đây là bức ảnh được lấy từ lời giải đạt top 1 của cuộc thi này để minh họa cách mà họ phát hiện được ra các transaction khác nhau của cùng một người có nhãn isFraud=1, và đã được mã hóa định danh là 2898694

	TransactionID	isFraud	TransactionAmt	card1	addr1	D1n	day	D3n	dist1	P_emaildomain	UID
1694	2988694	1	240.0	15775	251.0	-81.0	1.0	0.0	NaN	yahoo.com	2988694.0
10046	2997046	1	260.0	15775	251.0	-81.0	3.0	1.0	NaN	yahoo.com	2988694.0
34029	3021029	1	250.0	15775	251.0	-81.0	9.0	3.0	NaN	yahoo.com	2988694.0
36812	3023812	1	315.0	15775	251.0	-81.0	10.0	9.0	NaN	yahoo.com	2988694.0
40459	3027459	1	390.0	15775	251.0	-81.0	11.0	10.0	NaN	yahoo.com	2988694.0
43926	3030926	1	475.0	15775	251.0	-81.0	12.0	11.0	NaN	yahoo.com	2988694.0
43941	3030941	1	445.0	15775	251.0	-81.0	12.0	12.0	NaN	yahoo.com	2988694.0
44717	3031717	1	445.0	15775	251.0	-81.0	12.0	12.0	NaN	yahoo.com	2988694.0
44727	3031727	1	445.0	15775	251.0	-81.0	12.0	12.0	12.0	NaN	2988694.0
58485	3045485	1	295.0	15775	251.0	-81.0	15.0	12.0	NaN	yahoo.com	2988694.0

6.2.3 Xử lý mất cân bằng dữ liệu

Từ tính chất của Random Undersampling và Oversampling với SMOTE, chúng tôi có dự định kết hợp cả hai phương pháp để có thể kết hợp những ưu điểm từ chúng. Mặc dù vậy, việc kết hợp cần có một chiến lược phù hợp với quá trình thí nghiệm kỹ lưỡng, vì vậy chúng tôi chưa thể đưa vào trong nội dung dự án này.

6.2.4 Các mô hình khác

- **Sử dụng Mô hình Deep Learning:** Mô hình Deep Learning, như Mạng Nơ-ron Học Sâu (Deep Neural Networks) và mạng nơ-ron học sâu tạo cơ hội để học được những đặc trưng phức tạp hơn từ dữ liệu. Có thể thử nghiệm các mô hình như Convolutional Neural Networks (CNNs) hoặc Recurrent Neural Networks (RNNs) để cải thiện khả năng phát hiện gian lận.
- **Mô hình Học Sâu Kết hợp:** Kết hợp sử dụng các mô hình học sâu (Deep Learning) với các mô hình học máy truyền thống như Random Forest hoặc Gradient Boosting để tận dụng lợi ích của cả hai loại mô hình. Điều này có thể cải thiện khả năng tổng quát hóa của mô hình.
- **Sử dụng Phương pháp Tiên tiến Hơn:** Nghiên cứu và thử nghiệm các thuật toán và phương pháp mới, chẳng hạn như mô hình học tăng cường, mô hình học dạng mạng (Graph Neural Networks), hoặc học tập không giám sát (Unsupervised Learning) để xem chúng có thể nâng cao hiệu suất phát hiện gian lận hay không.

Tài liệu tham khảo

[1] Olivier Chapelle. Training a support vector machine in the primal. *Neural computation*, 19(5):1155–1178, 2007.

[2] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[3] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[4] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[5] StatQuest with Josh Starmer. CatBoost Part 1: Ordered Target Encoding, 2023.

[6] StatQuest with Josh Starmer. CatBoost Part 2: Building and Using Trees, 2023.

[7] Ji Zhu, Saharon Rosset, Hui Zou, and Trevor Hastie. Multi-class adaboost. *Statistics and its interface*, 2, 02 2006.