ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO

PHÁT HIỆN GIAO DỊCH BẤT THƯỜNG VỚI BỘ DỮ LIỆU BITCOIN NETWORK TRANSACTIONAL METADATA

Lớp: Khai thác dữ liệu và ứng dụng - CS313.M21

Giảng viên hướng dẫn: Nguyễn Thị Anh Thư

Nhóm B3		
STT	Họ và tên	MSSV
1	Phan Trọng Hậu	19520077
2	Phan Đại Dương	19520482
3	Hồ Mỹ Hạnh	19521470
4	Huỳnh Văn Hùng	19521564

TP. HỒ CHÍ MINH – 5/2022

PHÁT HIỆN GIAO DỊCH BẤT THƯỜNG VỚI BỘ DỮ LIỆU BITCOIN NETWORK TRANSACTIONAL METADATA

I. Tổng quan:

1. Giới thiệu về tiền ảo – Bitcoin:

Bitcoin là một loại tiền điện tử, được sử dụng và phân phối qua kênh điện tử. Bitcoin cũng là một hệ thống phi tập trung và ngang hàng (P2P). Tức là không có một tổ chức hoặc cá nhân nào nắm quyền kiểm soát. Bitcoin chỉ có một lượng cung nhất định – chỉ có tối đa 21 triệu Bitcoin được tạo ra trong hệ thống.

Một trong những đặc tính quan trọng của Bitcoin là ẩn danh. Ngày nay, các ngân hàng hầu như đều biết mọi thứ về khách hàng của mình: lịch sử tín dụng, địa chỉ, số điện thoại, các thói quen chi tiêu,… Bitcoin thì hoàn toàn ngược lại, vì các ví điện tử không hề liên kết đến một thông tin cá nhân nào. Và thực tế đã chứng minh, loại hình giao dịch này là một trong những phương pháp được các tội phạm ma túy, khủng bố hay rửa tiền lợi dụng

2. Bài toán đặt ra:

Phát hiện bất thường đã là một lĩnh vực được nghiên cứu phổ biến trong một thời gian dài. Các ứng dụng của nó trong lĩnh vực tài chính đã hỗ trợ rất nhiều trong việc xác định các hoạt động đáng ngờ của tin tặc. Mặc dù với những tiến bộ trong lĩnh vực tài chính như blockchain và trí tuệ nhân tạo, nhiều trường hợp gian lận vẫn xuất hiện.

Ở phạm vi đề tài này, nhóm giải quyết bài toán phân lớp nhị phân nhằm xác định xem một giao dịch Bitcoin có phải là một giao dịch bất thường hay không. Nhãn của các mẫu trong bộ dữ liệu trên sẽ thuộc thuộc tính out_and_tx_malicious trong bộ dữ liệu (nếu là giao dịch bất thường thì có giá trị 1 hoặc ngược lại thì có giá trị 0) với suy luận rằng đầu ra của một giao dịch bất thường cũng là một giao dịch bất thường.

- *Input*: Thông tin của một giao dịch Bitcoin bao gồm số lượng giao dịch input, số lượng giao dịch output, số lượng bitcoin nhận được cũng như gửi đi, tổng số lượng bitcoin nhận và gửi, số Bitcoin trung bình nhận được và gửi đi.
- *Output*: 1 nếu đó là giao dịch bất thường, 0 nếu đó là giao dịch không bất thường.

3. Đối tượng, phạm vi và mục tiêu:

Bộ dữ liệu Bitcoin transactional network metadata (2011-2013) sẽ được sử dụng để huấn luyện, đánh giá các phương pháp mà nhóm đề xuất. Các phương pháp giải quyết bài toán mà nhóm sử dụng sẽ là hai kĩ thuật cho việc xử lí mất cân bằng dữ liệu gồm

Resampling, Cost-sensitive learning và ba mô hình để huấn luyện và phân lớp giao dịch gồm DecisionTreeClassifier, SGDClassifier (Linear SVM), MLPClassifier cung cấp bởi thư viện Scikit-learn. Do giới hạn thời gian, phương pháp cho thực nghiệm, đánh giá sẽ là K-fold cross validation với số lượng fold là 5.

II. Mô hình giải bài toán:

1. Bộ dữ liệu Bitcoin network transactional metadata (2011-2013):

a. Giới thiệu:

Bộ dữ liệu Bitcoin transaction network metadata được công bố trong luận văn thạc sĩ "Anomaly detection in blockchain" của Shafiq, Omer. Dữ liệu thô từ bitcoin blockchain được sử dụng để tạo ra bộ dữ liệu này. Tất cả dữ liệu về bitcoin được đồng bộ từ internet sử dụng phần mềm bitcoin client và được chứa trong một sổ cái công khai và biểu diễn bằng đơn vị tiền tệ gọi là Bitcoin (BTC). Dữ liệu của số cái sẽ chứa tất cả các giao dịch bitcoin từ khi mạng lưới vừa được thành lập đến tận bây giờ. Với mỗi giao dịch, có thể có nhiều địa chỉ gửi, địa chỉ nhận. Hơn nữa, một người dùng có thể sở hữu nhiều địa chỉ khác nhau và mỗi người đều là nặc danh vì không có bất kì thông tin nào của người dùng đi cùng với bất kì địa chỉ nào cả.

Bộ dữ liệu được tạo ra sử dụng là dữ liệu bitcoin blockchain từ năm 2011 đến 2013. Lí do để chọn khoảng thời gian này là nhờ vào việc có sẵn dữ liệu của các giao dịch bất thường từ Bitcoin Forum (2014). Dữ liệu từ forum này được rút trích và gán nhãn là "malicious" cho bộ dataset. Bên cạnh đó, các giao dịch con của giao dịch bất thường cũng sẽ được coi như là một giao dịch bất thường.

b. Chi tiết bộ dữ liệu:

Bộ dữ liệu bao gồm 30248134 mẫu và 13 đặc trưng. Chi tiết về các đặc trưng như sau:

Tên đặc trưng	Ý nghĩa	Kiểu dữ liệu	Giá trị lớn nhất	Giá trị nhỏ nhất
tx_hash	Hash của giao dịch bitcoin	Chuỗi		

indegree	Số lượng giao dịch là input của giao dịch tx_hash	Số nguyên	1932	0
outdegree	Số lượng giao dịch là output của giao dịch tx_hash	Số nguyên	1322	0
in_btc	Số lượng bitcoin mà giao dịch tx_hash nhận được	Số thực	550000.0	0.0
out_btc	Số lượng bitcoin mà giao dịch tx_hash gửi đi	Số thực	500020.70037663	0.0
total_btc	Tổng số lượng bitcoin mà giao dịch tx_hash nhận và gửi	Số thực	1050000.0	0.0
mean_in_btc	Trung bình số lượng bitcoin mà giao dịch tx_hash nhận được	Số thực	499259.58754922	0.0
mean_out_btc	Trung bình số lượng bitcoin mà	Số thực	500000.0	0.0

	giao dịch tx_hash gửi đi			
in_malicious	Giá trị 1 nếu tx_hash là input của một giao dịch bất thường	Số nguyên	1	0
out_malicious	Giá trị 1 nếu tx_hash là output của một giao dịch bất thường	Số nguyên	1	0
is_malicious	Giá trị 1 nếu tx_hash là giao dịch bất thường	Số nguyên	1	0
out_and_tx_ malicious	Giá trị 1 nếu tx_hash là giao dịch bất thường hoặc là output của một giao dịch bất thường	Số nguyên	1	0
all_malicious	Giá trị 1 nếu tx_hash là giao dịch bất thường hoặc là input hay output của một	Số nguyên	1	0

	giao dịch bất thường			

2. Tiền xử lý dữ liệu:

a) Loại bỏ các thuộc tính không cần thiết:

Tên đặc trưng được loại bỏ	Ý nghĩa	Lý do loại bỏ
tx_hash	Hash của giao dịch bitcoin	Đặc trưng không cần thiết
in_malicious	Giá trị 1 nếu tx_hash là input của một giao dịch bất thường	Do mục tiêu của bài toán là dự đoán các giao dịch bất thường hay không mà các đặc trưng trên đã xác định được lớp của các giao dịch nên không thể dùng.
out_maclicious	Giá trị 1 nếu tx_hash là output của một giao dịch bất thường	
is_malicious	Giá trị 1 nếu tx_hash là giao dịch bất thường	
all_malicious	Giá trị 1 nếu tx_hash là giao dịch bất thường hoặc là input hay out-put của một giao dịch bất thường	

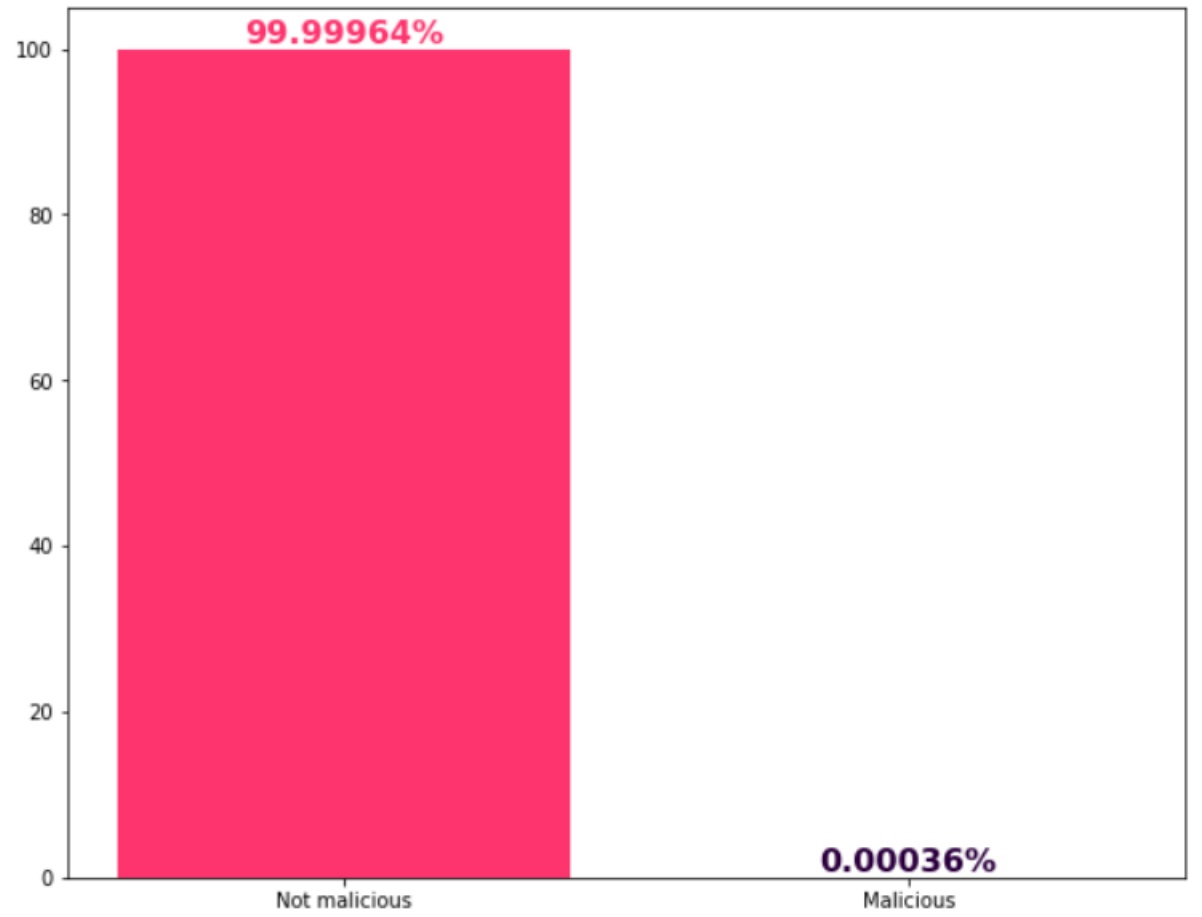
b) Phân tích và chuẩn hóa dữ liệu:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30248134 entries, 0 to 30248133
Data columns (total 8 columns):
 #   Column                Dtype
---  ------                -----
 0   indegree              uint16
 1   outdegree             uint16
 2   in_btc                float32
 3   out_btc               float32
 4   total_btc             float32
 5   mean_in_btc           float32
 6   mean_out_btc          float32
 7   out_and_tx_malicious  uint8
dtypes: float32(5), uint16(2), uint8(1)
memory usage: 721.2 MB
```

Hình 1: Kiểm ta bộ dữ liệu có bị thiếu dữ liệu hay không

Dựa vào hình 1, nhóm nhận thấy bộ dữ liệu ban đầu không bị khuyết dữ liệu.

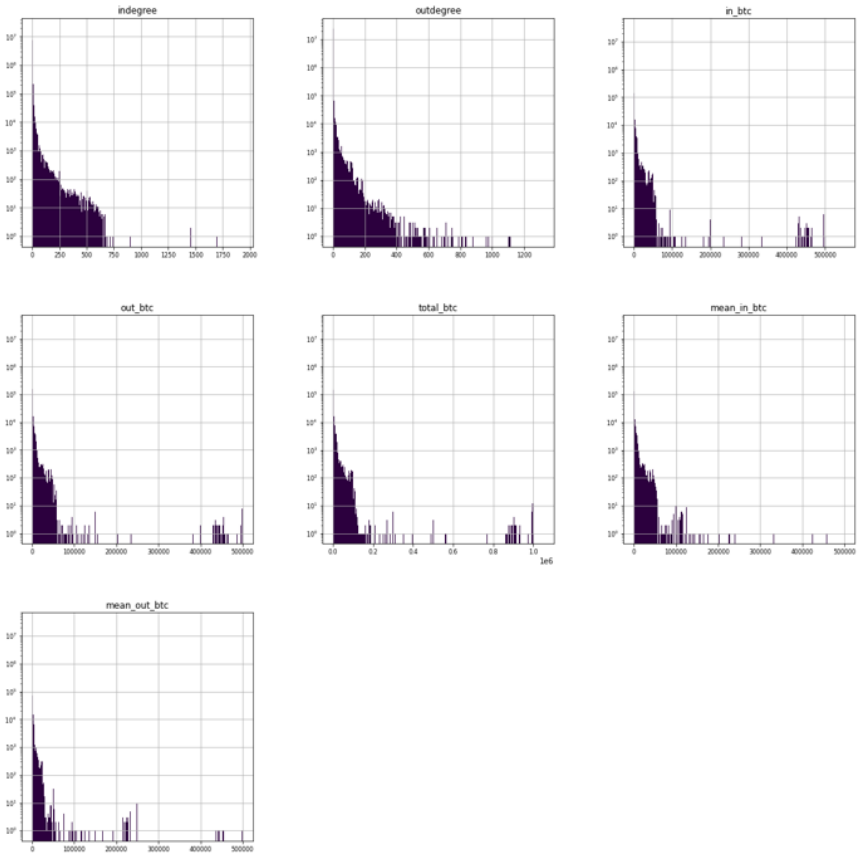


Hình 2: Biểu đồ phân bố dữ liệu giao dịch bitcoin

Dựa vào hình 2, nhóm nhận thấy bộ dữ liệu bị mất cân bằng lớn với các giao dịch bình thường chiếm 99.99964% lớn hơn rất nhiều so với các giao dịch bất thường chỉ chiếm 0.00036%.

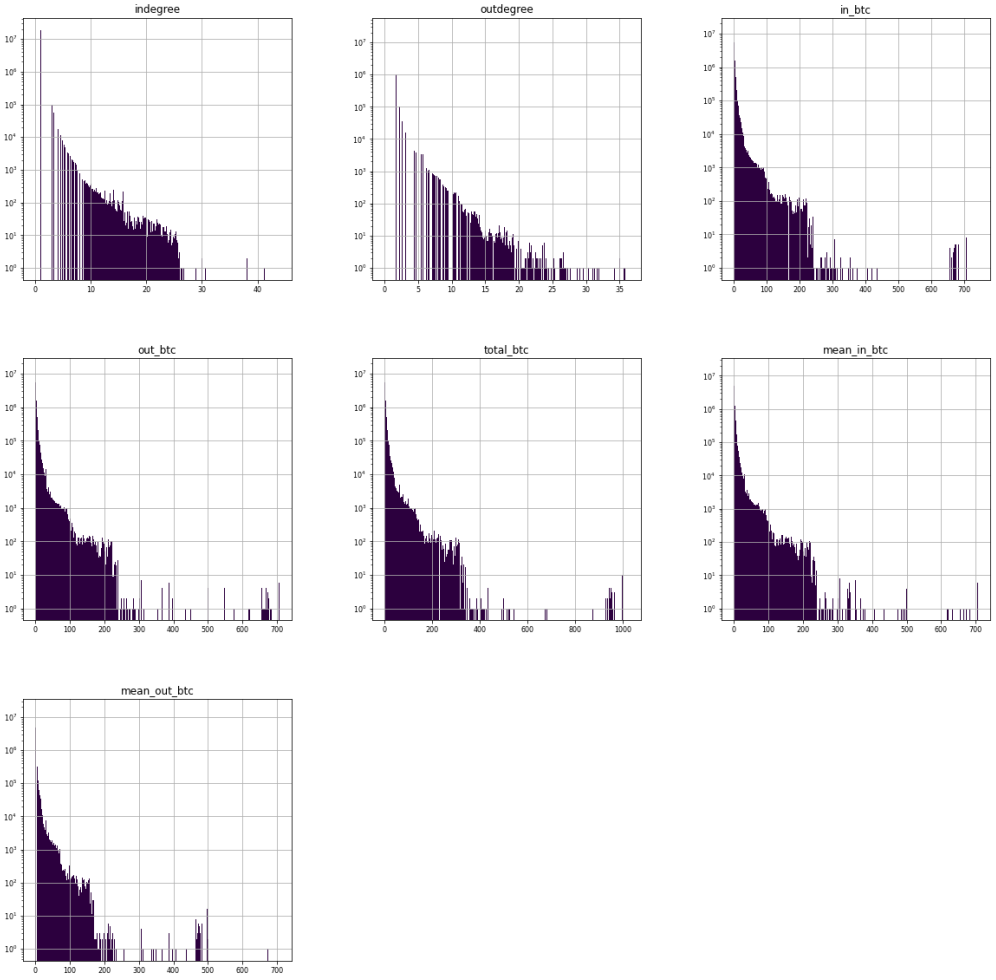
	indegree	outdegree	in_btc	out_btc	total_btc	mean_in_btc	mean_out_btc
count	30248134.00	30248134.00	30248134.00	30248134.00	30248134.00	30248134.00	30248134.00
mean	2.15	2.15	51.59	51.55	103.09	46.72	27.12
std	7.42	4.51	1301.06	1299.25	2598.06	970.91	711.45
min	0.00	0.00	0.00	0.00	0.00	0.00	0.00
25%	1.00	2.00	0.10	0.10	0.21	0.06	0.05
50%	1.00	2.00	0.93	0.96	1.96	0.54	0.49
75%	2.00	2.00	8.59	9.06	18.40	5.61	4.52
max	1932.00	1322.00	550000.00	500020.69	1050000.00	499259.59	500000.00

Bảng 1: Phân tích dữ liệu khi chưa chuẩn hóa

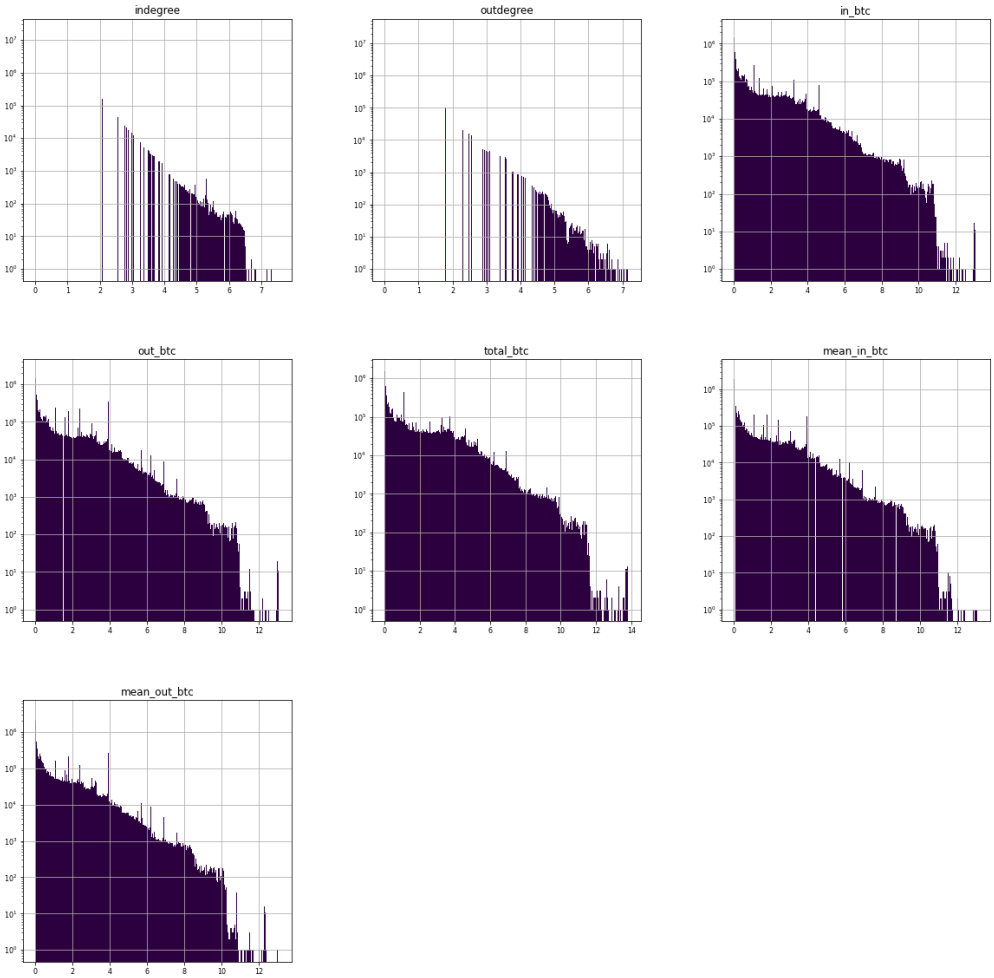


Hình 3: Biểu đồ phân phối tuần suất của dữ liệu khi chưa chuẩn hóa dữ liệu

Dựa vào hình 3, nhóm có nhận xét rằng bộ dữ liệu gốc khi chưa chuẩn hóa dữ liệu được phân bố không đồng đều, các giá trị của các thuộc tính chủ yếu tập trung vào bên trái của biểu đồ. Dựa vào số liệu ở bảng 1, giá trị trung bình (mean) của đa số các thuộc tính lớn hơn nhiều so với giá trị trung vị (median) của chúng. Do đó phân phối tần suất của bộ dữ liệu ban đầu có độ lệch dương, cũng có thể được gọi là phân phối lệch phải (right-skewed distribution). Sự chênh lệch này ảnh hưởng xấu đến quá trình huấn luyện mô hình.



Hình 4: Biểu đồ phân phối tuần suất của dữ liệu khi sử dụng phương pháp giảm độ lệch căn bậc hai



Hình 5: Biểu đồ phân phối tuần suất của dữ liệu khi sử dụng phương pháp làm giảm độ lệch log(x+1)

Để giảm sự chênh lệch này nhóm thử nghiệm với 2 phương pháp làm giảm độ lệch: căn bậc hai và log(x+1) (do bộ dữ liệu trước khi chuyển đổi có các giá trị bằng 0). Dựa vào hình 4 và 5, bộ dữ liệu trên hình 5 phân bố đều hơn so với bộ dữ liệu hình 4. Cho nên nhận thấy rằng phương pháp làm giảm độ lệch log(x+1) đem lại kết quả tốt hơn.

Để việc chuẩn hóa dữ liệu 1 cách tốt hơn, nhóm kết hợp giữa phương pháp biến đổi log(x+1) và phương pháp biến đổi Robust Scaler. Lý do của việc sử dụng phương pháp biến đổi Robust Scaler là bộ dữ liệu có nhiều thuộc tính có các outlier (ngoại lai) - là một điểm dữ liệu cụ thể mà nằm ngoài phạm vi của xác suất cho một tập dữ liệu. Nói cách khác, các outlier là khác biệt với các điểm dữ liệu xung quanh khác trong một cách đặc biệt. Dựa vào bảng 1, có thể thấy rằng các thuộc tính trong bộ dữ liệu có giá trị lớn hơn trung bình (mean) rất nhiều so với giá trị trung vị (median).

Mà các outlier có thể làm lu mờ các điểm dữ liệu khác, điều này sẽ làm ảnh hưởng không tốt đối với phương pháp biến đổi Standard Scaler.

	indegree	outdegree	in_btc	out_btc	total_btc	mean_in_btc	mean_out_btc
count	30248134.00	30248134.00	30248134.00	30248134.00	30248134.00	30248134.00	30248134.00
mean	-0.01	0.01	0.01	0.01	-0.00	-0.00	-0.00
std	0.92	1.00	0.89	0.89	0.88	0.90	0.90
min	-2.04	-4.00	-0.85	-0.85	-0.96	-0.77	-0.76
25%	-0.54	0.10	-0.78	-0.79	-0.85	-0.73	-0.73
50%	-0.54	0.10	-0.43	-0.43	-0.36	-0.49	-0.47
75%	0.34	0.10	0.59	0.60	0.69	0.47	0.47
max	14.36	22.81	7.53	7.42	6.74	7.87	8.70
range	16.40	26.81	8.38	8.27	7.70	8.64	9.46

Bảng 2: Mô tả bộ dữ liệu khi sử dụng kết hợp phương pháp giảm độ lệch log(x+1) và phương pháp biến đổi Standard Scaler

	indegree	outdegree	in_btc	out_btc	total_btc	mean_in_btc	mean_out_btc
count	30248134.00	30248134.00	30248134.00	30248134.00	30248134.00	30248134.00	30248134.00
mean	0.60	-0.03	0.30	0.29	0.20	0.47	0.46
std	1.05	0.27	0.65	0.64	0.59	0.71	0.72
min	-1.71	-1.10	-0.30	-0.30	-0.39	-0.23	-0.24
25%	0.00	0.00	-0.26	-0.26	-0.32	-0.20	-0.21
50%	0.00	0.00	0.00	0.00	0.00	0.00	0.00
75%	1.00	0.00	0.74	0.74	0.68	0.80	0.79
max	16.95	6.09	5.80	5.62	4.60	6.95	7.67
range	18.66	7.19	6.10	5.92	4.99	7.18	7.91

Bảng 3: Mô tả bộ dữ liệu khi sử dụng kết hợp phương pháp giảm độ lệch log(x+1) và phương pháp biến đổi Robust Scaler

Phương pháp biến đổi Standard Scaler sử dụng giá trị trung bình (mean) và độ lệch chuẩn. Ngược lại, phương pháp biến đổi Robust Scaler sử dụng giá trị trung vị (median) và phương pháp Interquartile range (IQR). Bởi vì các giá trị Outlier làm cho giá trị trung bình và độ lệch chuẩn tăng cao. Do đó, việc dùng Standard Scaler sẽ làm giảm khoảng cách giữa các giá trị Outlier và các giá trị khác, các giá trị trung bình mean, độ lệch chuẩn bị ảnh hưởng. Vì vậy khi xuất hiện các giá trị Outlier thì phương pháp Standard Scaler sẽ cho việc phân phối dữ liệu một cách sai lệch. Nhưng phương pháp Robust Scaler không gặp phải tình trạng này, khoảng cách giữa các giá trị Outlier và các giá trị khác phần lớn sẽ được còn được nguyên vẹn.

Phương pháp Robust Scaler được thực hiện bằng cách tính các giá trị median (trung vị), giá trị Q_1 (first quartile), giá trị Q_3 (third quartile). Sau đó, giá trị của mỗi biến sẽ được trừ đi giá trị median và được chia theo phương pháp Interquartile range (IQR) tức là khoảng cách giữa giá trị Q_3 và Q_1. Trong đó Q_3 là giá trị trung vị của nửa trên, Q_1 là giá trị trung vị của nửa dưới.

$$\textbf{Value = (Value - Median) / IQR = (Value - Median) / } (Q_3 - Q_1)$$

Ví dụ: cho một dữ liệu gồm: 4, 7, 9, 10, 11, 12, 20 thì ta có: median = 10, $Q_3 = 12$, $Q_1 = 7$.

3. Phương pháp đề xuất:

3.1. Xử lý mất cân bằng dữ liệu:

a) Cost-Sensitive Learning:

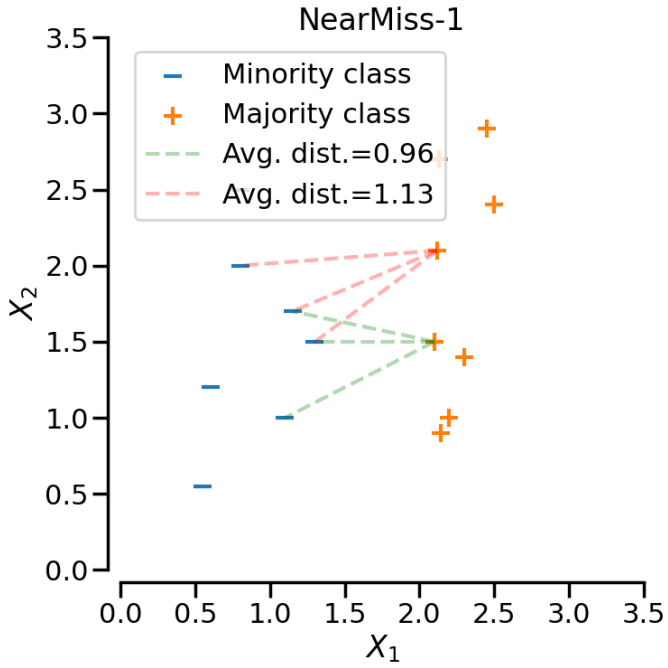
Thông thường, các thuật toán máy học được huấn luyện trên một tập dữ liệu và mục tiêu là giải một bài toán tối ưu hóa để cực tiểu hóa độ lỗi sử dụng một số hàm mất mát. Trong cost-sensitive learning, một hình phạt cho việc dự đoán sai được gọi là "cost" được thêm vào độ lỗi. Mục tiêu của cost-sensitive learning là cực tiểu hóa "cost" khi huấn luyện một mô hình trên một tập dữ liệu, trong đó với tùy theo loại của mẫu dữ đoán sai mà "cost" tương ứng với nó cũng khác nhau.

Thư viện scikit-learn cung cấp một số mô hình hỗ trợ cho cost-sensitive learning thông qua việc tinh chỉnh siêu tham số "class_weights". Ở phạm vi đề tài này, nhóm sử dụng SGDClassifier (Linear SVM) và DecisionTreeClassifier.

b) **Resampling**:

❖ Nearmiss UnderSampling:

Near-miss là một thuật toán có thể giúp cân bằng một tập dữ liệu mất cân bằng. Thuật toán thực hiện điều này bằng cách xem xét sự phân bố của lớp và loại bỏ các mẫu từ lớp lớn hơn. Khi hai điểm thuộc các lớp khác nhau rất gần nhau trong phân phối, thuật toán này loại bỏ điểm dữ liệu của lớp lớn hơn từ đó cố gắng cân bằng phân bố.



Các bước thực hiện thuật toán:

Bước 1: Đầu tiên, thuật toán tính toán khoảng cách giữa tất cả các điểm trong lớp lớn hơn với các điểm trong lớp nhỏ hơn.

Bước 2: Chọn các điểm dữ liệu của lớp lớn hơn có khoảng cách ngắn nhất với một điểm dữ liệu lớp nhỏ hơn. n điểm này cần được lưu trữ để loại bỏ.

Bước 3: Nếu có m điểm dữ liệu của lớp nhỏ hơn thì thuật toán sẽ loại bỏ m * n điểm của lớp lớn hơn.

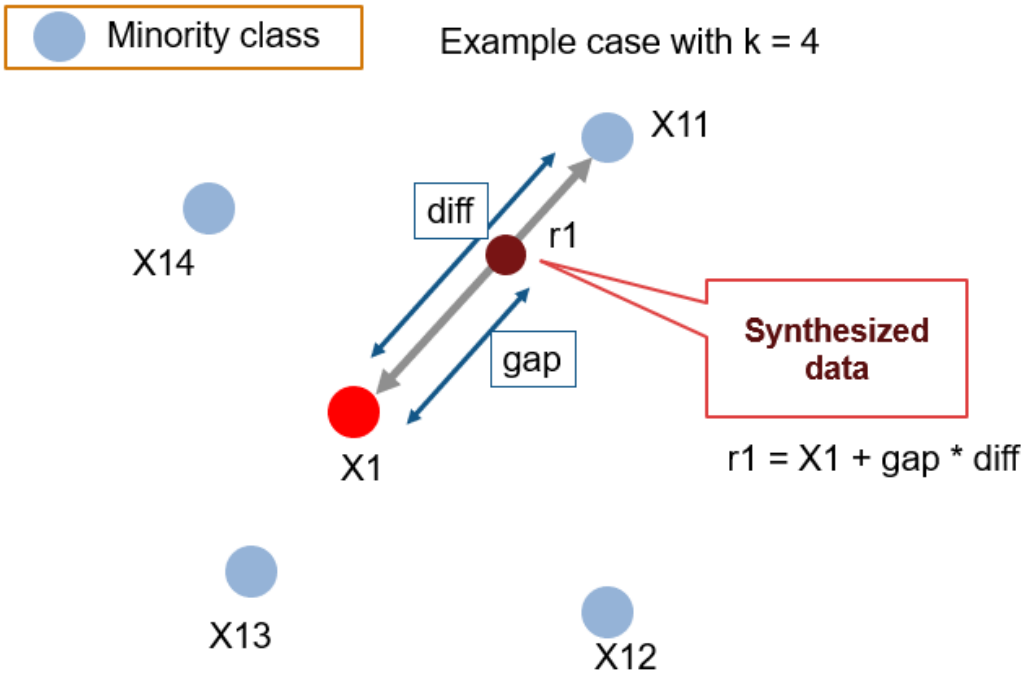
❖ SMOTEENN:

Ý tưởng chung của SMOTE là tạo ra dữ liệu tổng hợp giữa mỗi mẫu của lớp thiểu số và "k" hàng xóm gần nhất của nó. Có nghĩa là, đối với mỗi mẫu của lớp thiểu số, "k" các láng giềng gần nhất của nó được chọn. Sau đó giữa các cặp điểm được tạo bởi mẫu và từng láng giềng của nó thì ta sẽ có được data mới. Phương pháp để lựa chọn ra các láng giềng của một quan sát có thể dựa trên thuật toán *kNN* hoặc *SVM*.

SMOTE hoạt động như sau:

- Chọn ngẫu nhiên một điểm dữ liệu từ lớp thiểu số
- Xác định k vùng lân cận gần nhất từ điểm dữ liệu đó.
- Xác định vector giữa điểm đó tới một trong những điểm lân cận. Vector này sẽ được nhân với 1 số ngẫu nhiên từ 0 đến 1, sau đó cộng với điểm dữ liệu đang xét để tạo thành điểm dữ liệu mới.

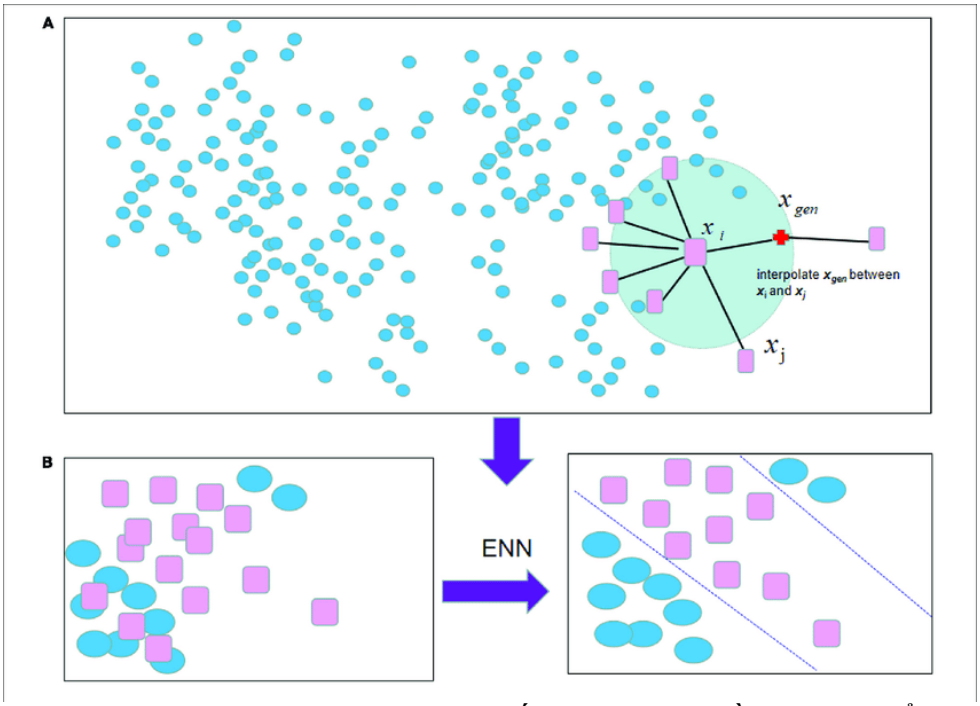
Quá trình này sẽ đảm bảo điểm dữ liệu mới được tạo ra không là bản sao của điểm dữ liệu hiện có, đồng thời cũng không quá khác biệt với các điểm trong lớp thiểu số.



Kỹ thuật ENN sẽ loại bỏ điểm của lớp đa số có dự đoán không phải là lớp đa số khi được thực hiện bằng phương pháp KNN. Do đó, nếu một điểm $x_i \in N$ có nhiều lân cận của một lớp khác, x_i sẽ bị loại bỏ

ENN hoạt động như sau:

- Lấy k vùng lân cận gần nhất của x_i, $x_i \in N$
- x_i sẽ bị loại bỏ nếu số lượng hàng xóm từ một lớp khác chiếm ưu thế
- Quá trình được lặp lại cho mỗi điểm của lớp đa số.



Thuật toán SMOTE cũng có những hạn chế của nó, bao gồm cả tính tổng quát và phương sai. Trong nhiều trường hợp, ranh giới lớp không được xác định rõ ràng vì một số trường hợp lớp thiểu số tổng hợp có thể vượt qua để xuất hiện trong không gian lớp đa số, đặc biệt là đối với dữ liệu phi tuyến có không gian đặc trưng lớn. Do đó, một số mẫu tổng hợp mới trong lớp thiểu số có thể bị gắn nhãn sai và việc cố gắng học hỏi từ các bộ dữ liệu đó thường dẫn đến tỷ lệ dự đoán sai cao hơn. Để loại bỏ các mẫu bị gắn nhãn sai được tạo bởi kỹ thuật SMOTE, nhóm đã áp dụng SMOTEENN, sự kết hợp giữa SMOTE và thuật toán Edited Nearest Neigh-bor (ENN), để làm sạch các mẫu dữ liệu tổng hợp.

3.2. Thuật toán:

a) Decision Tree:

❖ Ý tưởng:

Cây quyết định là một cây phân cấp có cấu trúc được dùng để phân lớp các đối tượng dựa vào dãy các luật. Các thuộc tính của đối tượng có thể thuộc các kiểu dữ liệu khác nhau như Nhị phân (Binary), Định danh (Nominal), Thứ tự (Ordinal), Số lượng (Quantitative) trong khi đó thuộc tính phân lớp phải có kiểu dữ liệu là Binary hoặc Ordinal.

Nói chung, cho dữ liệu về các đối tượng gồm các thuộc tính cùng với lớp (classes) của nó, cây quyết định sẽ sinh ra các luật để dự đoán lớp của các dữ liệu chưa biết.

❖ Ưu điểm:

- Mô hình dễ hiểu và dễ giải thích.
- Cần ít dữ liệu để huấn luyện.
- Có thể xử lý tốt với dữ liệu dạng số (rời rạc và liên tục) và dữ liệu hạng mục.
- Mô hình dạng white box rõ ràng.
- Xây dựng nhanh.
- Phân lớp nhanh.

❖ Nhược điểm:

- Không đảm bảo xây dựng được cây tối ưu.
- Có thể overfitting (tạo ra những cây quá khớp với dữ liệu huấn luyện hay quá phức tạp).
- Thường ưu tiên thuộc tính có nhiều giá trị (khắc phục bằng các sử dụng Gain Ratio).

b) SGD Classifier:

❖ Ý tưởng:

Stochastic Gradient Descent (SGD) là một thuật toán được sử dụng để tìm các giá trị của các tham số hay hệ số của các hàm, giúp tối thiểu hóa hàm chi phí. Thuật toán này là một biến thể của Gradient Descent. Thay vì tính toán hệ số trên toàn bộ dữ liệu qua mỗi vòng lặp thì nó sẽ cập nhật các hệ số cho từng điểm dữ liệu huấn luyện. Điều này sẽ phù hợp với những bộ dữ liệu có quy mô lớn và được cập nhật liên tục.

❖ Tham số:

loss *− str, default = 'hinge'*

Nó đại diện cho hàm mất mát. Nhóm để giá trị mặc định là "hinge", thuật toán trở thành linear SVM.

❖ Ưu điểm:

- Stochastic Gradient Descent (SGD) rất hiệu quả.
- Rất dễ thực hiện.

❖ Nhược điểm:

- Stochastic Gradient Descent (SGD) yêu cầu một vài siêu tham số như các tham số cho kĩ thuật regularization.

c) MLP Classifier:

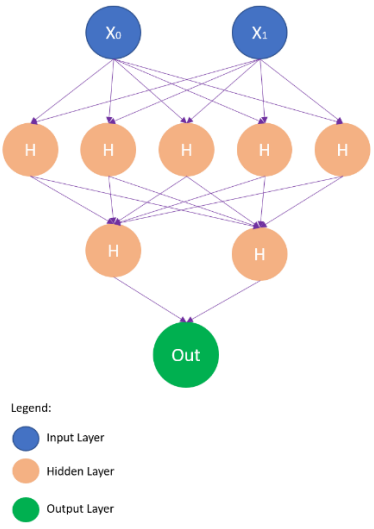
❖ Ý tưởng:

– Thuật toán Multi-layer Perceptron (MLP):

Multi-layer Perceptron (MLP) là một thuật toán máy học có giám sát dựa trên mô hình mạng nơ ron (Neural Network). Một mạng MLP tổng quát có n (n≥2) lớp: trong đó gồm một lớp đầu ra (tầng thứ n) và (n-1) lớp ẩn.

Kiến trúc của một mạng MLP có thể được mô tả như sau:

- Đầu vào là các vector $(x_1, x_2, \dots, x_p)$ trong không gian p chiều, đầu ra là các vector $(y_1, y_2, \dots, y_q)$ trong không gian q chiều. Trong đó p là kích thước mẫu đầu vào và q là số lớp cần phân loại.
- Giữa lớp đầu vào và lớp đầu ra có thể có một hoặc nhiều lớp ẩn phi tuyến (non-linear hidden layers).
- Mỗi nơron thuộc lớp sau sẽ liên kết đầy đủ với các nơron của lớp liền trước.
- Đầu ra của nơron lớp trước sẽ là đầu vào của nơron thuộc lớp liền sau nó.



– MLPClassifier:

Mô hình MLPClassifier triển khai thuật toán MLP được huấn luyện theo cơ chế lan truyền ngược (Backpropagation).

❖ Ưu điểm:

- Có khả năng học các mô hình phi tuyến tính.
- Phù hợp với các bộ dữ liệu lớn.

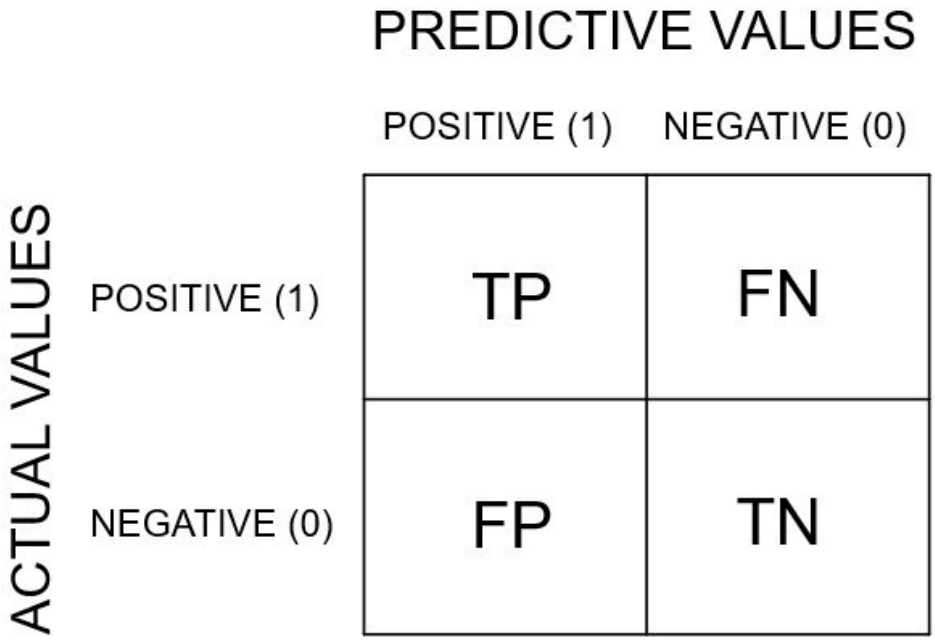
❖ Nhược điểm:

- Yêu cầu tinh chỉnh các siêu tham số như số lượng nơ ron ẩn, số lượng lớp ẩn, số lần lặp lại (iterations).
- Nhạy cảm với các dữ liệu được chuẩn hóa.

4. Độ đo đánh giá:

4.1. Confusion matrix:

Ma trận nhầm lẫn (hay còn gọi là ma trận lỗi) là một bảng đặc biệt để minh họa hiệu suất của thuật toán. Đây là một trong những kỹ thuật đo lường hiệu suất phổ biến nhất và được sử dụng rộng rãi cho các mô hình phân loại.



Trong đó:

- TP (True Positive): Số lượng các giá trị thực sự là Positive và được dự đoán là Positive.

- TN (True Negative): Số lượng các giá trị thực sự là Negative và được dự đoán là Negative.
- FP (False Positive): Số lượng các giá trị thực sự là Negative nhưng được dự đoán là Positive.
- FN (False Negative): Số lượng các giá trị thực sự là Positive nhưng được dự đoán là Negative.

4.2. Balanced accuracy:

Độ chính xác cân bằng được dùng để đánh giá hiệu suất của mô hình phân lớp khi bộ dữ liệu bị mất cân bằng, tức là khi một trong các lớp xuất hiện nhiều hơn lớp còn lại. Nó được tính bằng giá trị trung bình của độ nhạy và độ đặc hiệu.

$$\text{Balanced Accuracy} = \frac{\text{sensitivity} + \text{specificity}}{2}$$

Trong đó:

- Sensitivity (độ nhạy): Tỷ lệ các trường hợp là positive được dự đoán chính xác trong tổng số các trường hợp là positive.

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

- Specificity (độ đặc hiệu): Tỷ lệ các trường hợp là negative được dự đoán chính xác trong tổng số các trường hợp là negative.

$$\text{Specificity} = \frac{TN}{TN+FP}$$

Độ chính xác cân bằng càng gần 1, mô hình càng phân loại chính xác.

4.3. Macro average metric:

Macro average sẽ tính toán chỉ số độc lập cho từng lớp và lấy giá trị trung bình, do đó nó đối xử bình đẳng với tất cả các lớp. Trong khi đó micro average sẽ tổng hợp số liệu các mẫu của tất cả các lớp để tính giá trị trung bình, mang đến độ chính xác tổng thể. Mục tiêu của bài toán mà nhóm hướng đến là phát hiện bất thường, nên lựa chọn macro trong trường hợp này là tốt hơn hết.

- Macro precision:

Precision là tỷ lệ các dự đoán chính xác là positive trên tổng số các dự đoán là positive, được tính bởi công thức $\frac{TP}{TP+FP}$. Precision cao đồng nghĩa với xác suất dự đoán một mẫu positive chính xác cao.

Macro precision cho C lớp được tính như sau:

$$\text{macro precision} = \frac{1}{C}\sum_{j=1}^{C}\frac{TP_j}{TP_j + FP_j}$$

Với TP_j và FP_j lần lượt là true positive và false positive của mỗi lớp j

- Macro recall:

Recall là tỷ lệ các các dự đoán chính xác là positive trên tổng số các trường hợp là positive, được tính bởi công thức $\frac{TP}{TP+FN}$. Recall cao đồng nghĩa với việc True Positive Rate cao, tức tỉ lệ bỏ sót các điểm thực sự *positive* là thấp.

Macro recall score cho C lớp được tính như sau:

$$\text{macro recall} = \frac{1}{C}\sum_{j=1}^{C}\frac{TP_j}{TP_j + FN_j}$$

Với TP_j và FN_j lần lượt là true positive và false negative của mỗi lớp j

- Macro F1

F1-score là trung bình điều hòa của precision và recall, được tính bởi công thức $\frac{2\times(Precision\times Recall)}{Precision+Recall}$. F1-score có giá trị nằm trong khoảng (0; 1], khi F1 càng cao thì bộ phân lớp càng tốt.

Khi đó, macro-f1 score là trung bình điều hòa giữa precision P_j và recall R_j

$$\text{macro } F_1 = \frac{1}{C}\sum_{j=1}^{C}\frac{2\times P_j \times R_j}{P_j + R_j}$$

5. Phương pháp đánh giá:

Cross validation là quy trình lấy mẫu được sử dụng để đánh giá các mô hình máy học khi bộ dữ liệu bị hạn chế.

Quy trình:

- Xáo trộn dữ liệu ngẫu nhiên
- Chia tập dữ liệu thành k tập con có cùng kích thước
- Sử dụng một tập con là tập thử nghiệm và các tập con còn lại là tập huấn luyện. Quá trình này được lặp lại cho đến khi mỗi tập con đều được sử dụng làm tập thử nghiệm.
- Tại mỗi lần lặp, huấn luyện mô hình trên tập huấn luyện và đánh giá mô hình trên bộ thử nghiệm. Kết quả đánh giá mô hình cuối cùng sẽ là trung bình cộng kết quả đánh giá của k lần huấn luyện.

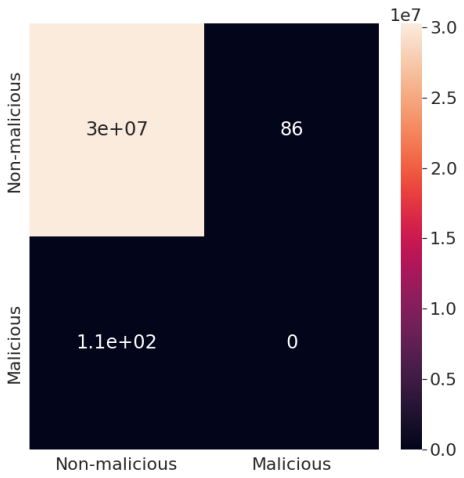
III. Thực nghiệm:

Việc đánh giá hiệu suất của các thuật toán được thực hiện sử dụng phương pháp K-fold cross validation. Ở phạm vi đề tài này, số lượng fold mà nhóm đặt ra là 5. Điều đó có nghĩa 80% bộ dữ liệu là tập train, 20% bộ dữ liệu là tập test với mỗi fold. Nhóm cũng đảm bảo tỉ lệ giữa các lớp trong tập test bằng với tỉ lệ giữa các lớp trong toàn bộ dữ liệu.

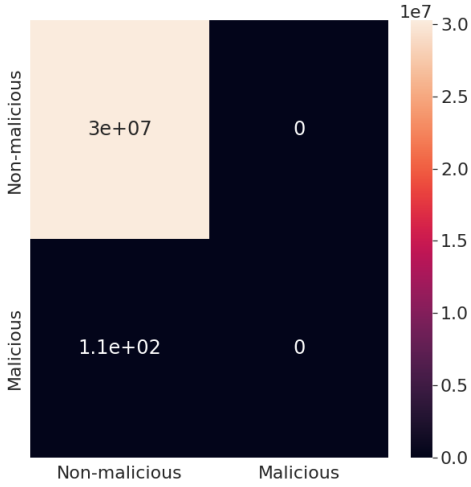
Nhóm tiến hành đánh giá các thuật toán dựa trên ba cách huấn luyện khác nhau: huấn luyện mà không có xử lí mất cân bằng dữ liệu, huấn luyện với cost-sensitive learning và cuối cùng là huấn luyện với bộ dữ liệu đã được cân bằng bởi một số thuật toán resampling. Lưu ý là dữ liệu khi huấn luyện đều đã được xử lý dùng log transform và RobustScaler. Ba thuật toán được sử dụng cho việc phân lớp là: Decision Tree, Linear SVM (sử dụng Stochastic gradient descent), Multi-layer Perceptron.

1. Huấn luyện mô hình mà không có xử lí mất cân bằng dữ liệu:

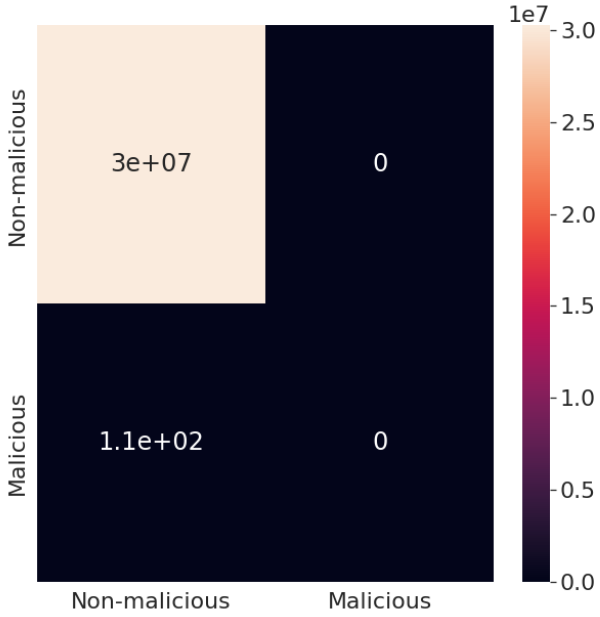
Dưới đây là Confusion matrix của ba mô hình phân lớp khi huấn luyện mà không sử dụng bất cứ kĩ thuật cân bằng dữ liệu nào:



Confusion matrix của Decision tree



Confusion matrix của SVM



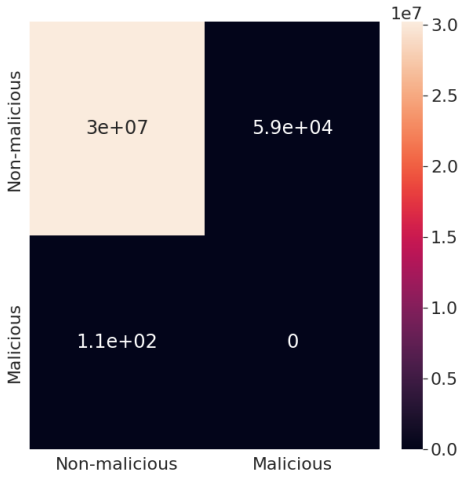
Confusion matrix của MLP

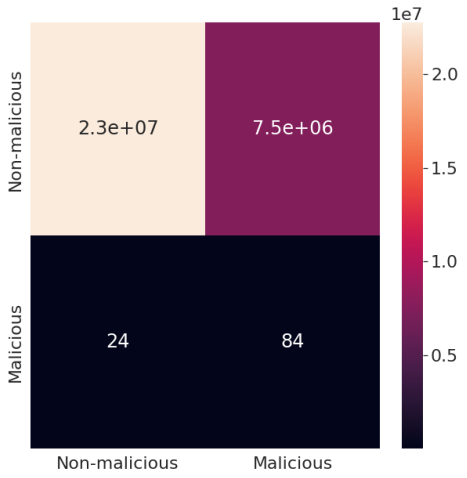
Có thể thấy rằng với bộ dữ liệu mất cân bằng lớn như vậy, cả ba mô hình đều không thể dự đoán được lớp bất thường (nhãn 1) do số lượng lớp bình thường (nhãn 0) lớn hơn rất nhiều. Do đó, mô hình sẽ có xu hướng dự đoán mọi giao dịch là bình thường để tăng độ

chính xác khiến không có một giao dịch bất thường nào được phát hiện trong quá trình đánh giá.

2. Sử dụng cost-sensitive learning để giải quyết mất cân bằng dữ liệu:

Ở phạm vi đề tài này, cost sensitive learning được sử dụng với hai mô hình là Decision Tree và Linear SVM bằng cách đặt giá trị siêu tham số "class_weights" của hai classifiers trong Scikit-learn là "balanced". Dưới đây là confusion matrix có được khi thực hiện cost sensitive learning với hai mô hình trên:





Confusion matrix của Decision tree *Confusion matrix của SVM*

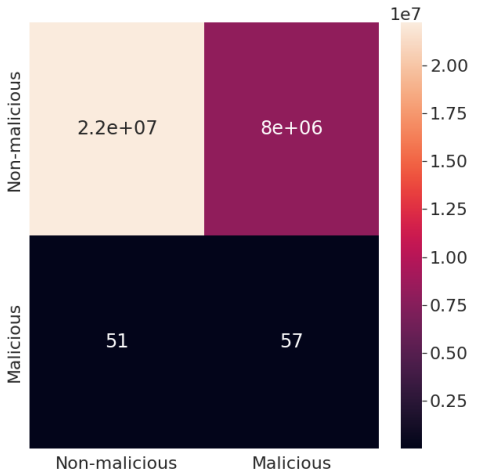
Kết quả thu được là tốt hơn so với việc huấn luyện mà không có xử lí mất cân bằng dữ liệu. Các mô hình không còn có xu hướng dự đoán toàn bộ các giao dịch là bình thường như trước nữa. Thay vào đó, với Decision Tree, mô hình dự đoán khoảng 5900 giao dịch bất thường tuy nhiên toàn bộ đều là false positive. Với SVM, mô hình dự đoán khoảng 7,500,000 giao dịch bất thường và trong đó dự đoán đúng được 84/108 giao dịch bất thường trong bộ dữ liệu. Do bài toán đặt ra là phân lớp giao dịch bất thường nên có thể coi đây là kết quả tốt hơn trước mặc dù số lượng dự đoán false positive tăng lên khá nhiều.

3. Sử dụng các phương pháp resampling để cân bằng dữ liệu:

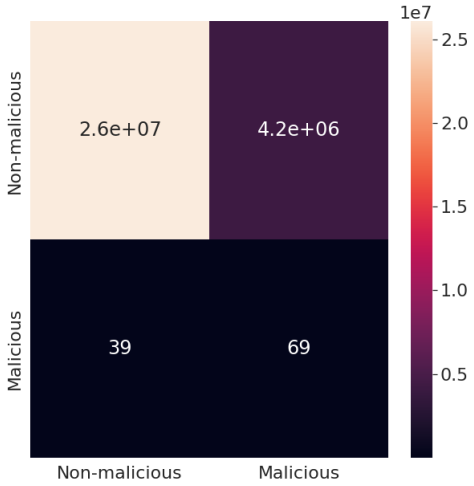
Phương pháp resampling này sẽ chỉ được áp dụng vào tập train. Để cân bằng dữ liệu, nhóm sử dụng Nearmiss undersampling để giảm số lượng lớp bình thường trong bộ dữ liệu sao cho tỉ lệ giữa lớp bất thường và lớp bình thường là 0.00001. Tiếp theo đó kĩ

thuật SMOTEENN được sử dụng để tăng số lượng mẫu của lớp bất thường cho bằng với số lượng mẫu của lớp bình thường và loại bỏ một số mẫu giúp phân bố của hai lớp này được phân chia rõ ràng với nhau hơn.

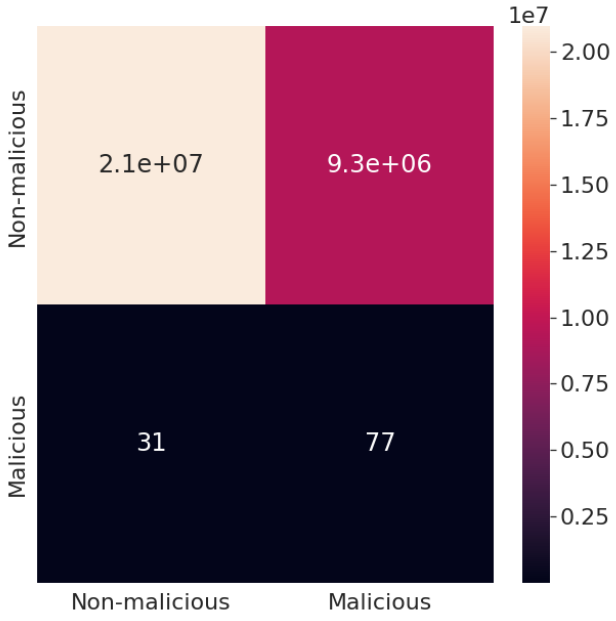
Dưới đây là confusion matrix của các mô hình khi sử dụng các kĩ thuật Resampling:



Confusion matrix của Decision tree



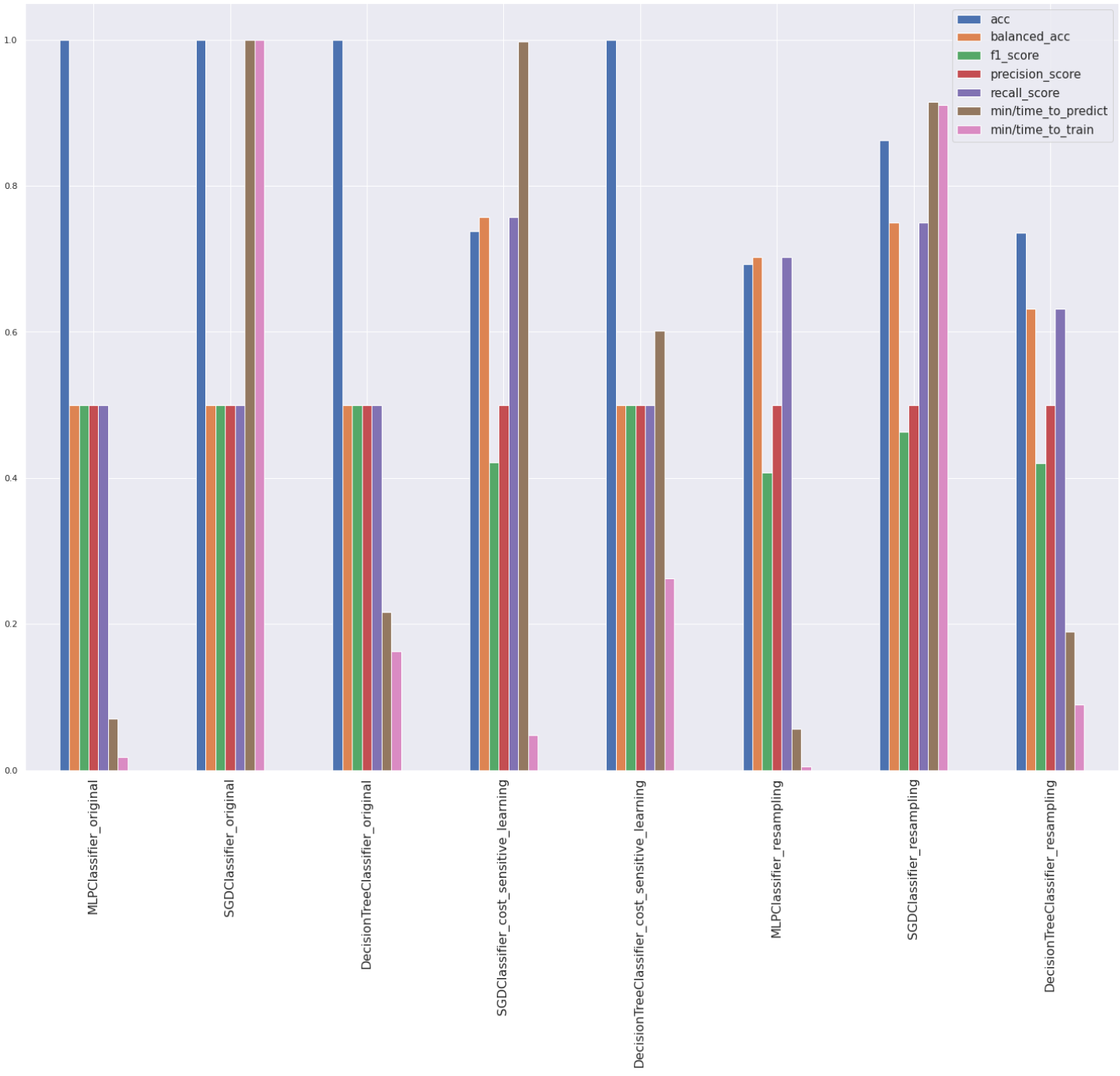
Confusion matrix của SVM



Confusion matrix của MLP

Tương tự cost sensitive learning, việc áp dụng resampling cũng giúp cho mô hình tránh việc dự đoán tất cả các giao dịch là bình thường. Tuy vậy, đánh đổi với nó là số lượng false positive của cả ba mô hình khá lớn, tăng lên đến hàng triệu chỉ để dự đoán được vài chục giao dịch bất thường nhưng đây vẫn là một điểm tốt so với việc huấn luyện mà không có kỹ thuật xử lý mất cân bằng dữ liệu nào.

4. So sánh các mô hình khi sử dụng các phương pháp khác nhau:



Trên đây là biểu đồ so sánh hiệu suất của ba thuật toán với các phương pháp xử lí mất cân bằng dữ liệu khác nhau. Thời gian huấn luyện cũng như thời gian để dự đoán đã được nghịch đảo trước khi đưa vẽ biểu đồ giúp việc đánh giá trở nên dễ dàng hơn. Mô hình nào có các cột càng cao, tức các độ đo tương ứng như accuracy, balanced accuracy, macro-precision, macro-recall, macro-f1 càng cao và thời gian huấn luyện, thời gian dự đoán càng ngắn.

Từ biểu đồ có thể thấy rằng cả ba mô hình khi không có xử lí mất cân bằng đều đạt accuracy rất cao, tuy nhiên các độ đo khác đều thấp do không thể dự đoán được các giao dịch bất thường. Khi có áp dụng phương pháp xử lí mất cân bằng dữ liệu, ta có thể dễ dàng nhận thấy rằng so với MLPClassifier và DecisionTreeClassifier, SGDClassifier (tức Linear SVM) tốt hơn rõ rệt cả khi dùng cost sensitive learning hay resampling với các độ đo đều cao hơn các mô hình khác (ngoại trừ DecisionTreeClassifier khi dùng cost sensitive learning có accuracy, f1 score cao nhưng lại rơi vào trường hợp không thể dự đoán được một giao dịch bất thường nào). So sánh SGDClassifier giữa cost sensitive learning và resampling, phương pháp resampling cho accuracy và f1 score cao hơn, các độ đo khác tương tự nhau và thời gian train ngắn hơn rất nhiều. Do đó nhóm kết luận:

Với bộ dữ liệu như trên, giữa ba mô hình DecisionTreeClassifier, SGDClassifier (Linear SVM), MLPClassifier kết hợp với cost sensitive learning và resampling thì SGDClassifier (Linear SVM) với resampling cho hiệu suất tốt nhất.

IV. Kết luận và hướng phát triển:

Ở đề tài này, nhóm đã tiến hành tìm hiểu, giải quyết một bài toán phân lớp nhị phân nhằm dự đoán xem một giao dịch Bitcoin có phải là giao dịch bất thường hay không. Bộ dữ liệu Bitcoin network transactional metadata được sử dụng cho việc huấn luyện, đánh giá. Đây là một bộ dữ liệu có sự mất cân bằng rất lớn giữa lớp không bất thường và bất thường. Do đó, kết hợp với ba mô hình máy học là DecisionTreeClassifier, SGDClassifier (Linear SVM), MLPClassifier cung cấp bởi Scikit-learn, hai kĩ thuật được sử dụng để giải quyết vấn đề mất cân bằng của dữ liệu là cost-sensitive learning và resampling cũng được

tìm hiểu. Kết quả thực nghiệm cho thấy rằng SGDClassifier (Linear SVM) kết hợp với resampling là mô hình có hiệu suất tốt nhất trong số các mô hình trên.

Mặc dù sử dụng các kĩ thuật cho dữ liệu mất cân bằng khác nhau và đã dự đoán được một số giao dịch bất thường khi đánh giá thì thách thức của bài toán này vẫn là số lượng giao dịch bình thường quá lớn so với giao dịch bất thường. Việc thiếu sót dữ liệu giao dịch bất thường khiến mô hình khó khăn trong việc dự đoán các giao dịch thuộc lớp này. Do đó, các nguồn dữ liệu về giao dịch Bitcoin bất thường sau năm 2013 có thể được xem xét, kết hợp với bộ dữ liệu để tăng số lượng mẫu lớp bất thường. Bên cạnh đó, ta có thể thực nghiệm với bộ dữ liệu không chỉ với các thuật toán có giám sát mà còn với các thuật toán bán giám sát hay không giám sát để chọn ra thuật toán tối ưu cho bài toán.

V. Phân công công việc:

Họ tên	MSSV	Nhiệm vụ phụ trách	Ghi chú
Phan Trọng Hậu	19520077	– Tiền xử lí dữ liệu và tìm hiểu thuật toán MLP Classifier.	Các thành viên hoàn thành tốt công việc được giao.
Phan Đại Dương	19520482	– Tổng quan về đề tài – Thực nghiệm: tiền xử lí dữ liệu và cài đặt, huấn luyện các mô hình máy học; so sánh, đánh giá các phương pháp. – Trực quan hóa dữ liệu. – Kết luận và hướng phát triển của đề tài.	
Hồ Mỹ Hạnh	19521470	– Các phương pháp xử lý dữ liệu mất cân bằng – Một số độ đo đánh giá – Tổng hợp nội dung, thiết kế slide Powerpoint để thuyết trình	
Huỳnh Văn Hùng	19521564	– Các phương pháp xử lý dữ liệu mất cân bằng. – Tìm hiểu thuật toán decision tree và SGDClassifier. – Thiết kế nội dung slide	

VI. Tài liệu tham khảo:

[1] J. Brownlee, "Cost-Sensitive Learning for Imbalanced Classification," 7 February 2020. [Online]. Available: https://machinelearningmastery.com/cost-sensitive-learning-for-imbalanced-classification/. [Accessed 5 May 2022].

[2] Shafiq, Omer, "Anomaly Detection In Blockchain," Tampere University, Finland, 2019.

[3] D. Radečić, "Top 3 Methods for Handling Skewed Data," 4 January 2020. [Online]. Available: https://towardsdatascience.com/top-3-methods-for-handling-skewed-data-1334e0debf45. [Accessed 8 May 2022].

[4] S. Kumar, "Stop using SMOTE to handle all your Imbalanced Data," 2 May 2021. [Online]. Available: https://towardsdatascience.com/stop-using-smote-to-handle-all-your-imbalanced-data-34403399d3be. [Accessed 5 May 2022].

[5] V. Lendave, "How can SMOTE technique improve the performance of weak learners?," 27 January 2022. [Online]. Available: https://analyticsindiamag.com/how-can-smote-technique-improve-the-performance-of-weak-learners/.

[6] B. Madhukar, "Using Near-Miss Algorithm For Imbalanced Datasets," 29 October 2020. [Online]. Available: https://analyticsindiamag.com/using-near-miss-algorithm-for-imbalanced-datasets/.

[7] G. Pilotti, "Oversampling and Undersampling: ADASYN vs ENN," 17 February 2020. [Online]. Available: https://medium.com/quantyca/oversampling-and-undersampling-adasyn-vs-enn-60828a58db39.

[8] J. Brownlee, "A Gentle Introduction to k-fold Cross-Validation," 23 May 2018. [Online]. Available: https://machinelearningmastery.com/k-fold-cross-validation/.

[9] michael-fuchs, "NN - Multi-layer Perceptron Classifier (MLPClassifier)," 2 March 2021. [Online]. Available: https://michael-fuchs-python.netlify.app/2021/02/03/nn-multi-layer-perceptron-classifier-mlpclassifier/. [Accessed 13 May 2022].

[10] K. N. Chính, "Ứng dụng mạng nơ ron nhân tạo vào bài toán dự đoán," Vietnam Academy of Science and Technology, 2017.

[11] Y. Singh, "Handle Outliers With Robust Scaling," 22 March 2022. [Online]. Available: https://proclusacademy.com/blog/robust-scaler-outliers/. [Accessed 13 May 2022].