ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

KHOA TOÁN - TIN
Faculty of Mathematics and Informatics

HỆ HỖ TRỢ QUYẾT ĐỊNH

Chủ đề:

Ứng dụng học máy để phân loại rủi ro tín dụng

Giảng viên hướng dẫn: **TS. Trần Ngọc Thăng**
Sinh viên thực hiện: **Nguyễn Thế Thiện**
Mã số sinh viên: **20227264**

Ngày 8 tháng 6 năm 2025

Lời nói đầu

Trong bối cảnh ngành tài chính ngân hàng ngày càng phát triển, việc quản lý và đánh giá rủi ro tín dụng đóng vai trò quan trọng trong việc đảm bảo sự ổn định và hiệu quả hoạt động của các tổ chức tài chính. Ứng dụng trí tuệ nhân tạo và học máy (machine learning) vào bài toán phân loại rủi ro tín dụng không chỉ giúp nâng cao độ chính xác trong việc dự đoán khả năng vỡ nợ mà còn hỗ trợ các tổ chức đưa ra quyết định cho vay hiệu quả hơn.

Đề tài "Phân loại rủi ro tín dụng" được thực hiện nhằm nghiên cứu và ứng dụng các kỹ thuật học máy để xây dựng mô hình dự đoán rủi ro tín dụng, từ đó cung cấp một công cụ hỗ trợ quyết định hiệu quả cho các tổ chức tài chính. Báo cáo này trình bày quá trình nghiên cứu, các phương pháp được sử dụng, kết quả đạt được, cũng như những bài học rút ra trong quá trình thực hiện.

Em xin chân thành cảm ơn Tiến sỹ Trần Ngọc Thăng, giảng viên hướng dẫn của em, người đã tận tình hướng dẫn, cung cấp những ý kiến quý báu và định hướng khoa học trong suốt quá trình thực hiện đề tài. Đồng thời, em cũng xin gửi lời cảm ơn đến gia đình, bạn bè và thầy đã hỗ trợ em về mặt kiến thức và tinh thần để hoàn thành báo cáo này.

Mặc dù đã cố gắng hết sức, báo cáo không thể tránh khỏi những thiếu sót. Em rất mong nhận được những ý kiến đóng góp quý giá từ thầy và các bạn để đề tài được hoàn thiện hơn.

Hà Nội, ngày . . . tháng . . . năm 2025
Người thực hiện

Nguyễn Thế Thiện

Yêu cầu bài cuối kỳ và mô tả mô hình

1. Yêu cầu bài cuối kỳ

STT	STT	Loại yêu cầu	Yêu cầu	Điểm chữ	Điểm số	Check	Minh chứng
1	1	Xử lý dữ liệu (2 điểm)	Mô tả bài toán, đầu vào, đầu ra, yêu cầu xử lý				Trang 6 - 9
2	2		Đánh nhãn & Tiền xử lý dữ liệu				Trang 8 - 9
3	3		Thống kê dữ liệu mẫu				Trang 10 - 11
4	4		Xử lý mất cân bằng dữ liệu (cho bài toán phân lớp) hoặc Chuyển đổi dữ liệu (cho bài toán hồi quy)				Tất cả các mô hình đều có xử lý mất cân bằng
5	5	Đánh giá mô hình (1 điểm)	Đề xuất và lựa chọn các tiêu chí đánh giá (về độ chính xác, tốc độ, khả năng ứng dụng,...)				Trang 30
6	6		Thống kê và phân tích lỗi				Trang 12 - trang 26
7	7	Cải tiến mô hình (4 điểm)	Kiến trúc mô hình 1				Trang 31 - 32
8	8		Kiến trúc mô hình 2				Trang 34 - 35
9	9		Kiến trúc mô hình 3				Trang 37 - 39
10	10		Kiến trúc mô hình 4				Trang 43 - 45
11	11		Kiến trúc mô hình 5				Trang 47 - 48
12	12		Kiến trúc mô hình 6				Trang 49 - 52
13	13		Kiến trúc mô hình 7				Trang 53 - 56
14	14		Kiến trúc mô hình 8				Trang 58 - 61
15	15	Đóng gói mô hình (3 điểm)	Có sử dụng mô hình tiên tiến trong 3 năm trở lại đây (chỉ ra paper liên quan)				Tranng ?
16	16		Có khả năng ứng dụng vào một ngữ cảnh cụ thể (ứng dụng vào bài toán nghiệp vụ nào, ai là người sử dụng)				Trang 62
17	17		Các chỉ số đánh giá mô hình đủ điều kiện để ứng dụng vào thực tế				Trang 64
18	18		Đóng gói giao diện demo chương trình				Chưa làm được
19	19		Làm slide báo cáo				Đã làm
20	20		Thuyết trình trên lớp				Đã làm
			Tổng điểm/20				
			Tổng điểm/10				

A	1
B	0.75
C	0.5
D	0.25
F	0

2. Mô tả các mô hình

STT	Tên mô hình	Điều kiện dừng	Phương pháp tối ưu hóa siêu tham số	Siêu tham số của mô hình	Kết quả đánh giá trên dữ liệu test theo các chỉ số	Chú giải
1	Logistic Regression		Không dùng	Đã nhận xét trong báo cáo	Trong báo cáo	Mô hình baseline
2	Random Forest phiên bản 1		RandomizedSearch	Đã nhận xét trong báo cáo	Trong báo cáo	Mô hình 2
3	Random Forest phiên bản 2		RandomizedSearch	Đã nhận xét trong báo cáo	Trong báo cáo	Mô hình 3
4	Random Forest phiên bản 3		RandomizedSearch	Đã nhận xét trong báo cáo	Trong báo cáo	Mô hình 4
5	Random Forest phiên bản 4		RandomizedSearch	Đã nhận xét trong báo cáo	Trong báo cáo	Mô hình 5
6	XGBoost phiên bản 1		RandomizedSearch	Đã nhận xét trong báo cáo	Trong báo cáo	Mô hình 6
7	XGBoost phiên bản 2		RandomizedSearch	Đã nhận xét trong báo cáo	Trong báo cáo	Mô hình 7
8	XGBoost phiên bản 3		RandomizedSearch	Đã nhận xét trong báo cáo	Trong báo cáo	Mô hình 8

Mục lục

1 Giới thiệu đề tài

1.1 Khái niệm rủi ro tín dụng

Rủi ro tín dụng là khả năng xảy ra tổn thất tài chính khi người vay không thực hiện đúng các điều khoản của hợp đồng tín dụng, bao gồm việc chậm trả nợ, trả nợ không đầy đủ hoặc không trả nợ khi đến hạn các khoản gốc và lãi vay. Đây là rủi ro phổ biến trong hoạt động cho vay của ngân hàng và các tổ chức tài chính, khi người vay không có khả năng hoặc không thực hiện nghĩa vụ trả nợ theo hợp đồng đã thỏa thuận.

Hình 1.1: Rủi ro tín dụng là gì ?

1.2 Các loại rủi ro tín dụng

- Rủi ro chậm trả: Người vay không hoàn trả đủ cả gốc và lãi đúng hạn.
- Rủi ro mất vốn: Ngân hàng không thu hồi được cả gốc và lãi của khoản cho vay.
- Rủi ro tập trung: Khi ngân hàng cho vay quá nhiều vốn vào một số ít khách hàng hoặc ngành nghề, tăng nguy cơ vỡ nợ.
- Rủi ro giao dịch, lựa chọn, bảo đảm, nghiệp vụ, danh mục, nội tại, tác nghiệp... liên quan đến quy trình cho vay và đặc điểm khách hàng.

1.3 Hậu quả

- Làm giảm lợi nhuận và khả năng thanh khoản của ngân hàng.
- Tăng nợ xấu, chi phí xử lý và nguy cơ phá sản.
- Mất uy tín và giảm năng lực cạnh tranh của tổ chức tín dụng.
- Ảnh hưởng tiêu cực đến nền kinh tế, gây suy thoái, thất nghiệp và mất ổn định xã hội.
- Gây mất niềm tin vào hệ thống tài chính và có thể lan tỏa rủi ro ra toàn bộ thị trường.

1.4 Cách phòng tránh

- Cách phòng tránh rủi ro tín dụng:
- Xây dựng chính sách và quy trình quản lý rủi ro chặt chẽ.
- Thẩm định kỹ năng lực và uy tín khách hàng trước khi cho vay.
- Yêu cầu tài sản bảo đảm có giá trị và khả năng xử lý tốt.
- Đa dạng hóa danh mục tín dụng, tránh tập trung rủi ro.
- Giám sát, theo dõi và xử lý kịp thời các khoản vay sau giải ngân.
- Nâng cao năng lực nhân sự và ứng dụng công nghệ hiện đại trong quản lý tín dụng.

1.5 Giải pháp đưa ra trong bài báo cáo này

Dựa vào những thông tin của người dùng, ta sẽ xây dựng một mô hình máy học dùng để phân loại

2 Bài toán

2.1 Mô tả bài toán

Bài toán đặt ra ở đây là chúng ta cần phân tích xem một khách hàng có khả năng rơi vào trường hợp bị rủi ro tín dụng hay không

2.2 Dữ liệu thu thập trên Kaggle

Dữ liệu thu thập bao gồm 2 file: **customer_data.csv** và **payment_data.csv**.

1. **payment_data.csv:** Lịch sử thanh toán thẻ của khách hàng.
   - **id**: mã khách hàng
   - **OVD_t1**: số lần quá hạn loại 1
   - **OVD_t2**: số lần quá hạn loại 2
   - **OVD_t3**: số lần quá hạn loại 3
   - **OVD_sum**: tổng số ngày quá hạn
   - **pay_normal**: số lần thanh toán bình thường
   - **prod_code**: mã sản phẩm tín dụng
   - **prod_limit**: hạn mức tín dụng của sản phẩm
   - **update_date**: ngày cập nhật tài khoản
   - **new_balance**: số dư hiện tại của sản phẩm
   - **highest_balance**: số dư cao nhất trong lịch sử
   - **report_date**: ngày thanh toán gần nhất
2. **customer_data.csv:** Dữ liệu nhân khẩu học và các thuộc tính danh mục của khách hàng đã được mã hóa.
   - **fea_1**
   - **fea_3**
   - **fea_5**
   - **fea_6**
   - **fea_7**
   - **fea_9**

- Ngoài ra còn có các dữ liệu dạng số như: **fea_2, fea_4, fea_8, fea_10, fea_11**
- **label là 1**: khách hàng có rủi ro tín dụng cao
- **label là 0**: khách hàng có rủi ro tín dụng thấp

2.3 Dữ liệu vào và đầu ra

1. **Dữ liệu đầu vào:** Các features trừ feature label
2. **Dữ liệu đầu ra:** feature label

2.4 Xem qua dữ liệu các bảng

1. Bảng **customer_data** có dạng:

label	id	fea_1	fea_2	fea_3	fea_4	fea_5	fea_6	fea_7	fea_8	fea_9	fea_10	fea_11
1	54982665	5	1245.5	3	77000.0	2	15	5	109	5	151300	244.948974
0	59004779	4	1277.0	1	113000.0	2	8	-1	100	3	341759	207.173840
0	58990862	7	1298.0	1	110000.0	2	11	-1	101	5	72001	1.000000
1	58995168	7	1335.5	1	151000.0	2	11	5	110	3	60084	1.000000
0	54987320	7	NaN	2	59000.0	2	11	5	108	4	450081	197.403141

Hình 2.1: Bảng thông tin khách hàng

2. Bảng **payment_data** có dạng:

id	OVD_t1	OVD_t2	OVD_t3	OVD_sum	pay_normal	prod_code	prod_limit	update_date	new_balance	highest_balance	report_date
58987402	0	0	0	0	1	10	16500.0	04/12/2016	0.0	NaN	NaN
58995151	0	0	0	0	1	5	NaN	04/12/2016	588720.0	491100.0	NaN
58997200	0	0	0	0	2	5	NaN	04/12/2016	840000.0	700500.0	22/04/2016
54988608	0	0	0	0	3	10	37400.0	03/12/2016	8425.2	7520.0	25/04/2016
54987763	0	0	0	0	2	10	NaN	03/12/2016	15147.6	NaN	26/04/2016

Hình 2.2: Bảng thông tin thanh toán của khách hàng

2.5 Xử lý cơ bản

Dựa vào dữ liệu được cung cấp, ta có hướng xử lý cơ bản đầu tiên như sau:

1. Join 2 bảng trên lại với nhau bằng id
2. Lọc trùng
3. Chuyển kiểu dữ liệu của các cột có định dạng ngày tháng năm sang datetime
4. Kiểm tra xem có id nào tồn tại cả label là 0 và 1 không

3 Tìm hiểu tổng quan dữ liệu

3.1 Xem thông tin dữ liệu bảng

```
Index: 8159 entries, 0 to 8249
Data columns (total 24 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   label            8159 non-null   int64
 1   id               8159 non-null   int64
 2   fea_1            8159 non-null   int64
 3   fea_2            7176 non-null   float64
 4   fea_3            8159 non-null   int64
 5   fea_4            8159 non-null   float64
 6   fea_5            8159 non-null   int64
 7   fea_6            8159 non-null   int64
 8   fea_7            8159 non-null   int64
 9   fea_8            8159 non-null   int64
 10  fea_9            8159 non-null   int64
 11  fea_10           8159 non-null   int64
 12  fea_11           8159 non-null   float64
 13  OVD_t1           8159 non-null   int64
 14  OVD_t2           8159 non-null   int64
 15  OVD_t3           8159 non-null   int64
 16  OVD_sum          8159 non-null   int64
 17  pay_normal       8159 non-null   int64
 18  prod_code        8159 non-null   int64
 19  prod_limit       2119 non-null   float64
 20  update_date      8138 non-null   datetime64[ns]
 21  new_balance      8159 non-null   float64
 22  highest_balance  7763 non-null   float64
 23  report_date      7068 non-null   datetime64[ns]
dtypes: datetime64[ns](2), float64(6), int64(16)
memory usage: 1.8 MB
```

Hình 3.1: Info của bảng gộp

Một vài điểm ta cần quan tâm sau khi quan sát ở trên:

1. Dữ liệu đều ở dạng số hoặc ngày tháng, không tồn tại các giá trị chữ trong các cột

2. Dữ liệu bao gồm 8159 dòng, tuy nhiên có 1 vài cột không đủ 8159 dòng. Chứng tỏ có chứa null

3. Dữ liệu bao gồm 24 cột tất cả và các số có dạng int64 hoặc float64. Nếu máy yếu ta có thể đổi kiểu dữ liệu của các cột này thành kiểu dữ liệu như int8 hoặc float8 (nếu được) để đảm bảo memory usage sẽ ít hơn

3.2 Xem mô tả tổng quát dữ liệu

	count	mean	min	25%	50%	75%	max	std
label	8159.0	0.168526	0.0	0.0	0.0	0.0	1.0	0.374355
id	8159.0	57824359.167913	54982353.0	54990315.0	58989082.0	58996558.0	59006239.0	1821534.605653
fea_1	8159.0	5.522368	1.0	4.0	5.0	7.0	7.0	1.387334
fea_2	7176.0	1286.214465	1116.5	1247.0	1283.0	1317.875	1481.0	52.091643
fea_3	8159.0	2.319157	1.0	1.0	3.0	3.0	3.0	0.889378
fea_4	8159.0	139056.13433	15000.0	77000.0	112000.0	151000.0	1200000.0	108422.854408
fea_5	8159.0	1.940311	1.0	2.0	2.0	2.0	2.0	0.236924
fea_6	8159.0	11.028067	3.0	8.0	11.0	15.0	16.0	2.699835
fea_7	8159.0	4.884422	-1.0	5.0	5.0	5.0	10.0	3.032391
fea_8	8159.0	100.061527	64.0	90.0	105.0	111.0	115.0	12.56185
fea_9	8159.0	4.250889	1.0	4.0	5.0	5.0	5.0	0.874402
fea_10	8159.0	188555.81738	60000.0	60047.0	72000.0	350020.0	650070.0	167211.030757
fea_11	8159.0	152.21844	1.0	1.0	183.030052	219.248717	707.106781	121.935903
OVD_t1	8159.0	0.246354	0.0	0.0	0.0	0.0	34.0	1.242424
OVD_t2	8159.0	0.126731	0.0	0.0	0.0	0.0	34.0	0.861944
OVD_t3	8159.0	0.371859	0.0	0.0	0.0	0.0	35.0	2.914862
OVD_sum	8159.0	188.803407	0.0	0.0	0.0	0.0	31500.0	1813.261311
pay_normal	8159.0	14.550803	0.0	4.0	11.0	25.0	36.0	12.048854
prod_code	8159.0	8.237897	0.0	6.0	10.0	10.0	27.0	3.5254
prod_limit	2119.0	85790.677253	1.1	37400.0	67650.0	112200.0	660000.0	74502.584522
update_date	8138	2012-02-16 23:14:31.467191040	1988-07-19 00:00:00	2009-02-14 00:00:00	2013-06-18 12:00:00	2015-01-28 00:00:00	2016-12-04 00:00:00	NaN
new_balance	8159.0	106197.902096	-40303.2	0.0	0.0	25093.2	163211958.0	1898152.434904
highest_balance	7763.0	220238.847095	501.0	23302.5	43741.0	100500.0	180000500.0	2828591.458437
report_date	7068	2014-06-25 16:45:13.752122368	1996-02-24 00:00:00	2013-09-26 00:00:00	2015-09-21 00:00:00	2016-01-29 00:00:00	2016-12-06 00:00:00	NaN

Hình 3.2: Bảng mô tả

Một vài điểm ta cần quan tâm sau khi quan sát bảng trên:

1. **fea_7, new_balance** có tồn tại giá trị âm (để ý cột min).
2. **fea_2, prod_limit, update_date, highest_balance, report_date** dường như tồn tại giá trị ngoại lệ

3.3 Xem các cột null

	Missing Values	% of Total Values
prod_limit	6040	74.0
report_date	1091	13.4
fea_2	983	12.0
highest_balance	396	4.9
update_date	21	0.3

Hình 3.3: Các cột null

4 Phân tích kỹ hơn và xử lý

4.1 Chia tập dữ liệu thành train và test

Câu lệnh có dạng như sau:

```
train_df, test_df = train_test_split(df_full, test_size=0.2, random_state=42, stratify=df_full['label'])
```

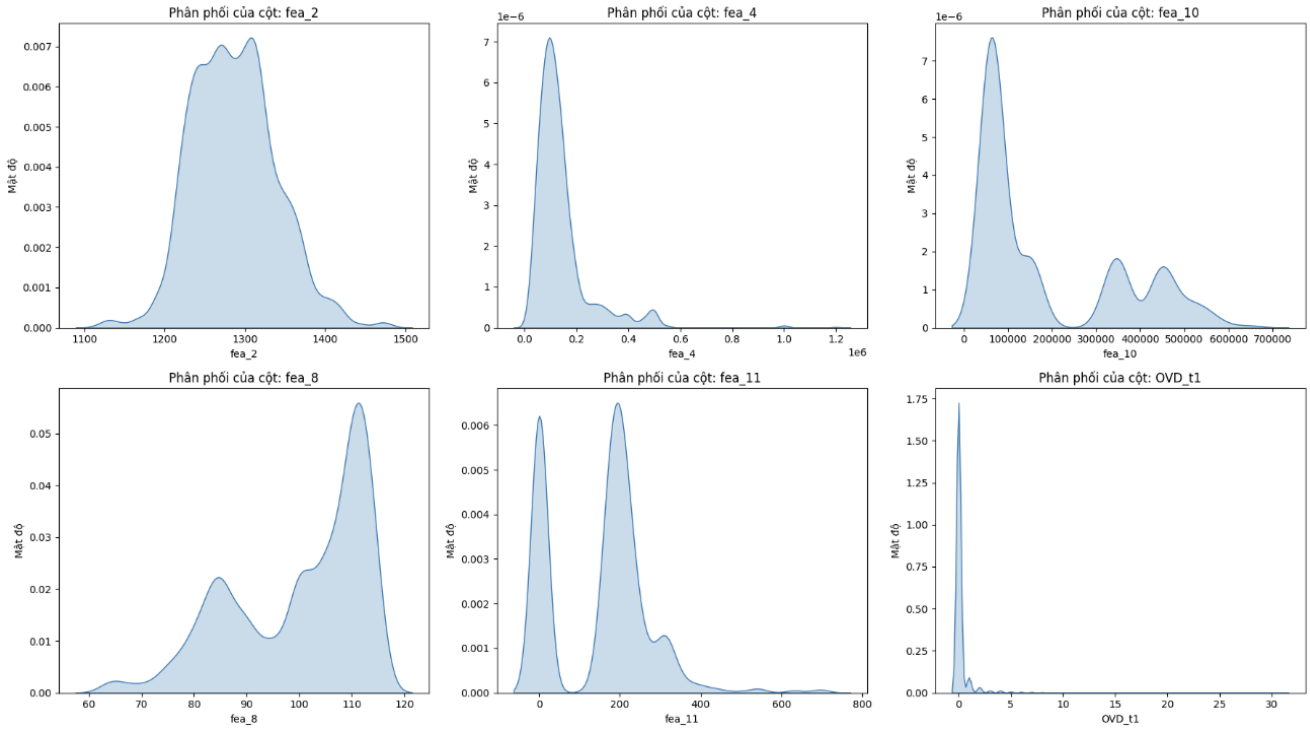
Hình 4.1: Train test split

Ở đây ta lưu tâm đến **stratify=df_full['label']** (lưu ý **df_full** ở đây là bảng sau khi ta join 2 file customer và payment với nhau). Việc sử dụng tham số **stratify** có tác dụng đảm bảo việc phân chia dữ liệu có tầng (stratified sampling).

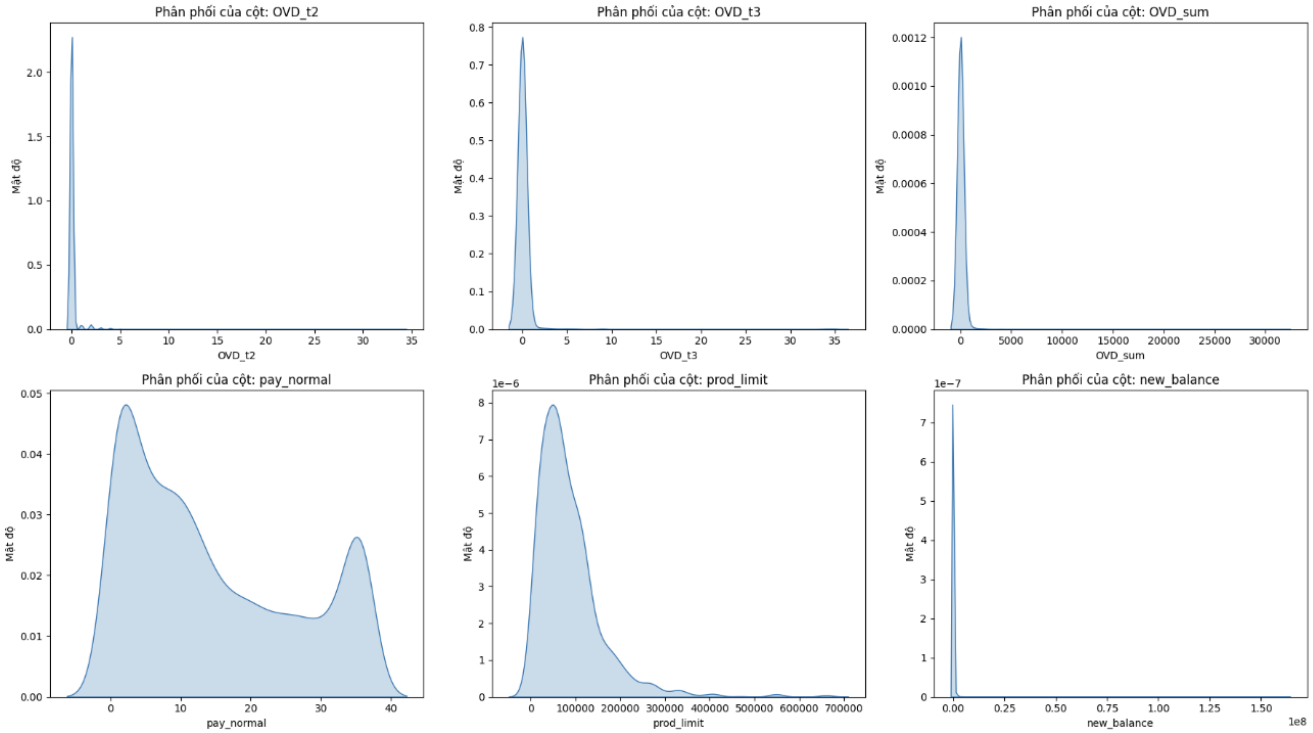
Tác dụng chính: Đảm bảo tỷ lệ các lớp (classes) trong cột 'label' được giữ nguyên giữa tập train và test

4.2 Xem phân phối của các cột

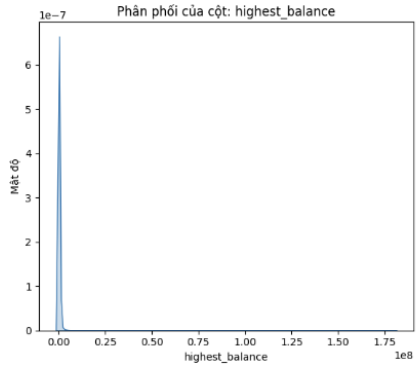
4.2.1 Đối với các cột số



Hình 4.2: Ảnh 1



Hình 4.3: Ảnh 2



Hình 4.4: Ảnh 3

Từ những hình ảnh trên ta có một vài nhận xét như sau:

1. **Phân phối Lệch phải (Right-Skewed)**

   Nhiều biến số quan trọng như `prod_limit` (hạn mức), `highest_balance` (số dư cao nhất), và các cột `OVD_*` (thông tin quá hạn) thể hiện sự lệch phải rõ rệt. Điều này phản ánh thực tế nghiệp vụ rằng phần lớn khách hàng có hạn mức và số dư ở mức vừa phải, và chỉ một số ít có giá trị rất cao. Các giá trị ngoại lệ này có thể làm giảm hiệu suất của các mô hình nếu không được xử lý.

2. **Phân phối Đa đỉnh (Multimodal Distribution)**

Một số biến như `fea_2`, `fea_10`, và `pay_normal` có phân phối đa đỉnh. Điều này gợi ý mạnh mẽ về sự tồn tại của các **phân khúc khách hàng** tiềm ẩn trong dữ liệu. Ví dụ, các đỉnh khác nhau trong `pay_normal` có thể đại diện cho các nhóm có hành vi thanh toán khác nhau (khách hàng mới vs. khách hàng lâu năm). Đây là một cơ hội quan trọng cho việc kỹ thuật đặc trưng (feature engineering) để tạo ra các biến mới có sức mạnh dự báo cao.

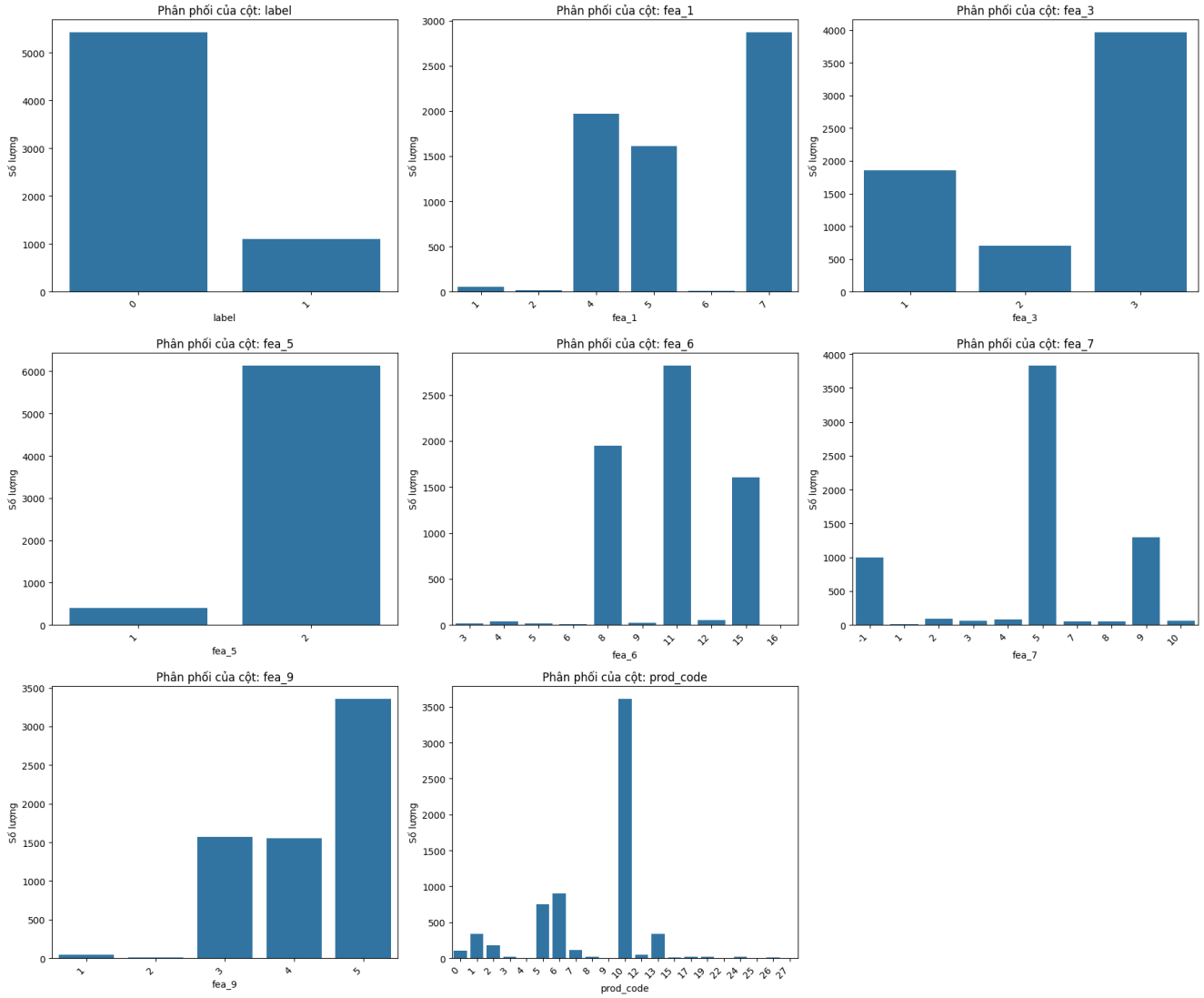
3. **Sự tập trung cao tại giá trị 0**

   Các biến liên quan đến nợ quá hạn (ví dụ: `OVD_sum`) có một đỉnh cực kỳ lớn và nhọn tại giá trị 0. Giá trị 0 ở đây không chỉ là một con số mà mang ý nghĩa nghiệp vụ quan trọng: **"Khách hàng không có lịch sử trả nợ quá hạn"**. Đây là một đặc điểm phân biệt rõ ràng giữa khách hàng rủi ro thấp và các nhóm khác, cần được xử lý riêng biệt để mô hình có thể học được thông tin này.

4. **Chênh lệch lớn về Thang đo (Scale)**

   Các biến có thang đo và khoảng giá trị rất khác nhau, ví dụ `highest_balance` (hàng trăm triệu) so với `OVD_t1` (số nguyên nhỏ). Nếu không được chuẩn hóa, các mô hình dựa trên khoảng cách hoặc có sử dụng trọng số (như SVM, Neural Networks, Logistic Regression) sẽ bị chi phối hoàn toàn bởi các biến có giá trị lớn, bỏ qua thông tin từ các biến có giá trị nhỏ nhưng quan trọng về mặt nghiệp vụ.

4.2.2 Đối với các cột category



Hình 4.5: Biểu đồ cột cho các cột chữ

Nhận xét

1. **Cột label mất cân bằng**: cần có biện pháp xử lý mất cân bằng hoặc các mô hình xử lý tốt với dữ liệu mất cân đối

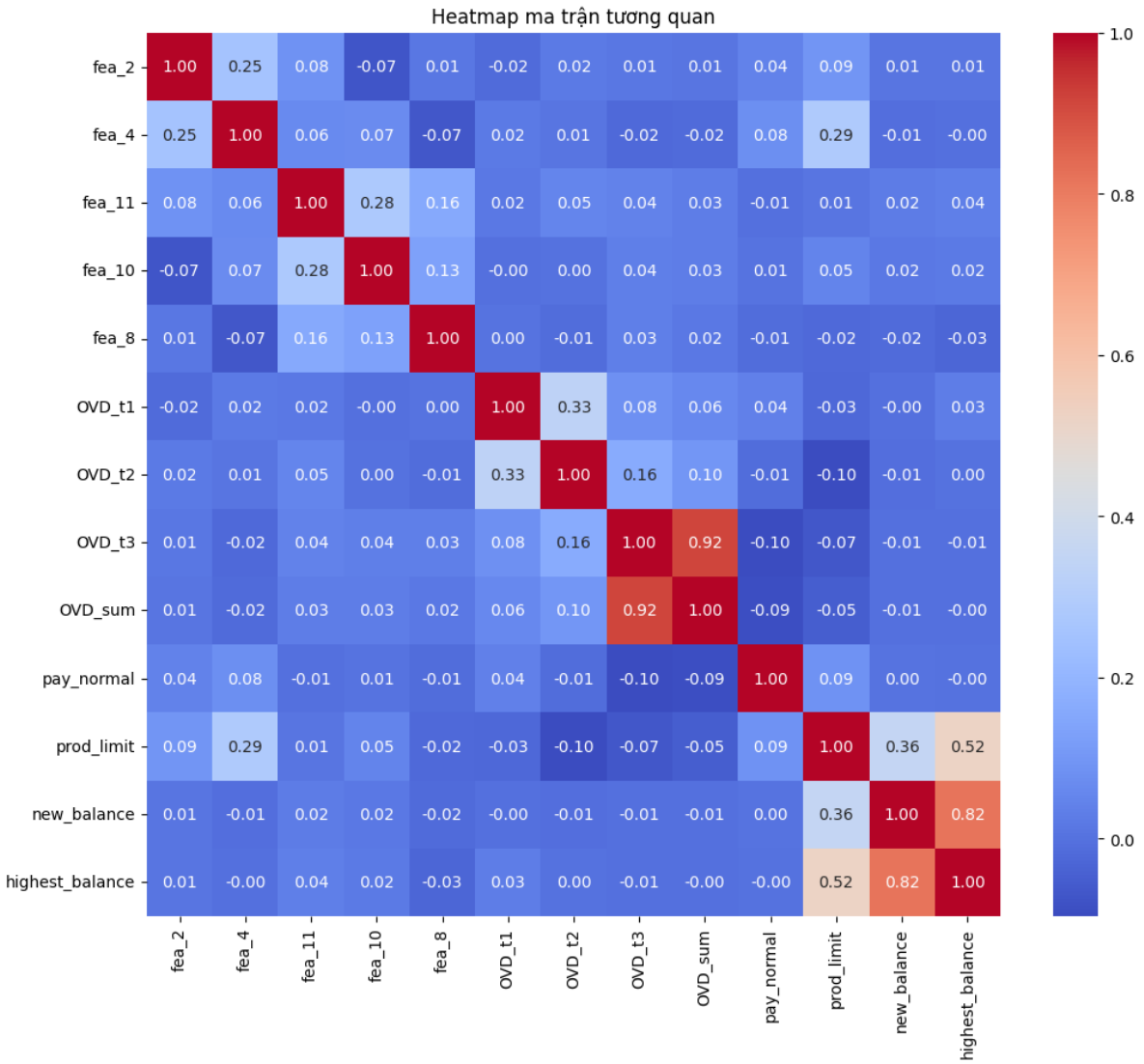
2. **Phân phối Mất cân bằng (Imbalanced Distribution)**
   Đây là đặc điểm nổi bật nhất. Trong hầu hết các biến, một hoặc hai hạng mục chiếm ưu thế tuyệt đối về số lượng, trong khi các hạng mục còn lại xuất hiện rất ít. *Ví dụ điển hình là* `fea_5` *và* `fea_7`, nơi một hạng mục chiếm gần như toàn bộ dữ liệu. Điều này có thể khiến mô hình bỏ qua thông tin từ các nhóm thiểu số.

3. **Sự khác biệt về Số lượng Hạng mục (Cardinality)**
   Các biến có số lượng hạng mục rất khác nhau. Một số biến có cardinality

thấp (ví dụ: `fea_5` có 2-3 hạng mục chính), trong khi các biến khác có cardinality cao (ví dụ: `prod_code` có hơn 20 hạng mục). Việc áp dụng cùng một phương pháp mã hóa cho tất cả các biến sẽ không hiệu quả, đặc biệt là với các biến có cardinality cao có thể gây ra "lời nguyền về chiều dữ liệu"(curse of dimensionality) nếu dùng One-Hot Encoding.

4. **Bản chất Dữ liệu là Danh nghĩa (Nominal)**
Mặc dù các hạng mục được biểu diễn bằng số, chúng không có mối quan hệ thứ tự (ordinal). Ví dụ, trong cột `fea_3`, giá trị '3' không có nghĩa là "lớn hơn"giá trị '1'. Do đó, việc sử dụng trực tiếp các giá trị số này sẽ khiến mô hình diễn giải sai mối quan hệ giữa chúng. Bắt buộc phải sử dụng các kỹ thuật mã hóa phù hợp.

4.3 Xem tương quan giữa các cột



Hình 4.6: Heatmap

Biểu đồ nhiệt (heatmap) của ma trận tương quan Pearson được sử dụng để kiểm tra mối quan hệ tuyến tính giữa các cặp biến số. Phân tích này cho thấy một số vấn đề quan trọng về đa cộng tuyến (multicollinearity) cần được giải quyết.

1. **Đa cộng tuyến nghiêm trọng**
   Đây là phát hiện quan trọng nhất. Có hai nhóm biến thể hiện sự tương quan rất cao, cho thấy sự dư thừa thông tin:

   - **Nhóm biến quá hạn (OVD):** Hệ số tương quan giữa `OVD_t3` và `OVD_sum` là **0.92**. Điều này cho thấy `OVD_sum` gần như là một tổ hợp tuyến tính của các biến thành phần, gây ra sự dư thừa nghiêm trọng.
   - **Nhóm biến số dư (Balance):** Hệ số tương quan giữa `new_balance` và `highest_balance` là **0.82**. Cả hai biến này đều phản ánh một khía cạnh rất giống nhau về quy mô tài chính của khách hàng.

   Hậu quả: Việc giữ lại các biến có tương quan cao sẽ làm giảm sự ổn định và khả năng diễn giải của các mô hình tuyến tính (ví dụ: Logistic Regression).

2. **Tương quan ở mức độ trung bình**
   Một số cặp biến có tương quan ở mức vừa phải, phản ánh các mối quan hệ hợp lý về mặt nghiệp vụ. Ví dụ, `prod_limit` có tương quan dương (0.52) với cả `new_balance` và `highest_balance`. Các mối quan hệ này không đủ mạnh để gây ra vấn đề nghiêm trọng nhưng vẫn cung cấp thông tin hữu ích.

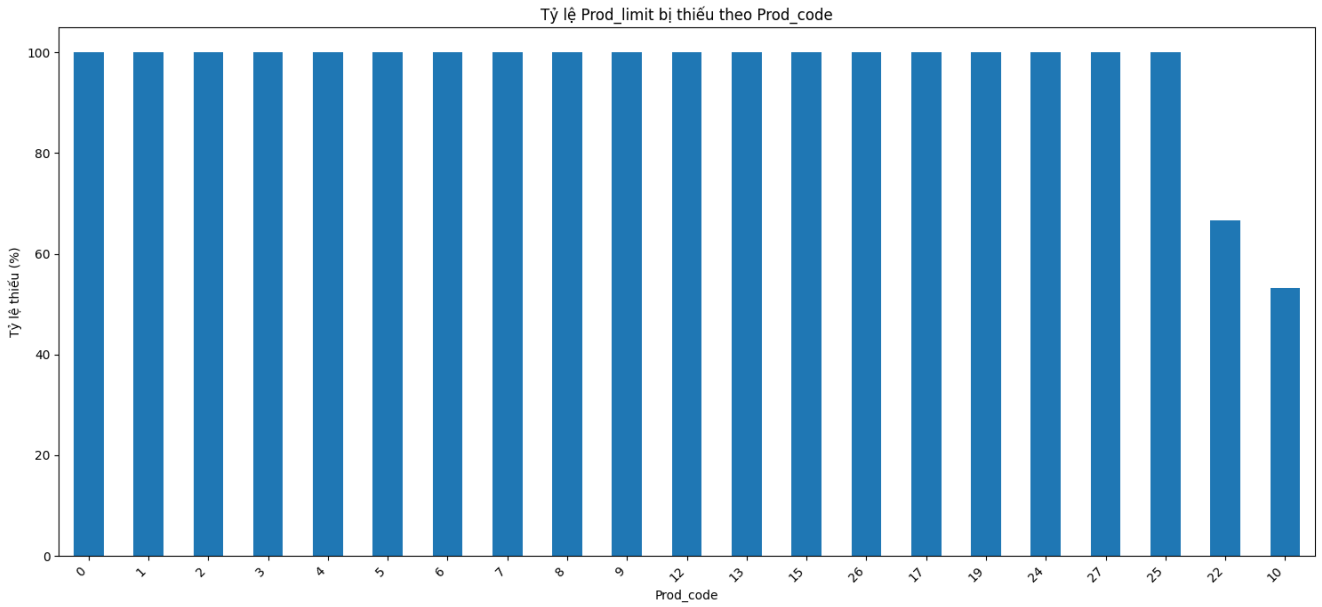
3. **Tương quan thấp**
   Phần lớn các cặp biến còn lại có tương quan gần 0. Điều này chỉ ra rằng không có mối quan hệ *tuyến tính* mạnh giữa chúng, nhưng không loại trừ khả năng tồn tại các mối quan hệ phi tuyến tính phức tạp mà các mô hình phi tuyến (như các mô hình cây) có thể khai thác.

4.4 Tìm hiểu những cột chứa NULL và đề xuất phương án xử lý

4.4.1 Cột prod_limit

1. **Tạo biến chỉ báo** (biến mà chỉ mang giá trị 0 hoặc 1) cho cột prod_limit tên: prod_limit_is_missing (1 nếu giá trị tại cột prod_limit là NULL và ngược lại)

2. **Đặt giả thuyết**: NULL mang ý nghĩa "sản phẩm không có khái niệm hạn mức tín dụng"



Hình 4.7: Tỷ lệ prod_limit thiếu theo prod_code

Ta thấy một vài sản phẩm (ngoại trừ 22,10) có tỷ lệ null của cột prod_limit là 100%. Tuy nhiên các cột prod_code = 22, 10 thì lại có tỷ lệ không phải 100%. Chứng tỏ 2 cột này có tồn tại null

KẾT LUẬN: Chứng tỏ null không đại diện cho "sản phẩm không có khái niệm giới hạn định mức"(bởi 22,10 vẫn bị giới hạn)

3. **Giả thuyết 2**: Ban đầu, sản phẩm người dùng không bị giới hạn tín dụng (tức prod_limit = NULL). Nhưng sau đó, do vi phạm chính sách nào đó (hoặc có thể vì vấn đề nào đó khác) dẫn đến việc người đó bị prod_limit khác null (tức là lúc này người này bị giới hạn tín dụng)

    - Việc vi phạm chính sách dẫn đến prod_limit bị khác null (tức là lúc đó người dùng bị áp đặt hạn mức) có thể liên quan đến biến OVD_t1, OVD_t2, OVD_t3, OVD_sum

Mô tả các biến OVD và pay_normal khi prod_limit BỊ THIẾU:

	OVD_t1	OVD_t2	OVD_t3	OVD_sum	pay_normal
count	4835.000000	4835.000000	4835.000000	4835.000000	4835.000000
mean	0.242399	0.145398	0.447777	233.206618	13.714581
std	1.296211	1.000495	3.204309	2035.614896	11.683933
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	4.000000
50%	0.000000	0.000000	0.000000	0.000000	10.000000
75%	0.000000	0.000000	0.000000	0.000000	22.000000
max	31.000000	34.000000	35.000000	31500.000000	36.000000

Mô tả các biến OVD và pay_normal khi prod_limit KHÔNG BỊ THIẾU:

	OVD_t1	OVD_t2	OVD_t3	OVD_sum	pay_normal
count	1692.000000	1692.000000	1692.000000	1692.000000	1692.000000
mean	0.244090	0.079196	0.136525	71.304965	16.867021
std	0.907356	0.493235	1.656388	1083.168605	12.575898
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	4.000000
50%	0.000000	0.000000	0.000000	0.000000	15.000000
75%	0.000000	0.000000	0.000000	0.000000	29.000000
max	14.000000	7.000000	35.000000	26248.000000	36.000000

Hình 4.8: Mô tả các biến OVD và pay_normal khi prod_limit BỊ THIẾU

- Các giá trị trung bình (**mean**) tương ứng của các cột **OVD** khi `prod_limit` bị thiếu đều **lớn hơn** so với khi `prod_limit` không bị thiếu.
- Ngược lại, **pay_normal** (chi trả bình thường) khi `prod_limit` bị thiếu lại có giá trị trung bình **thấp hơn** so với trường hợp không bị thiếu. Điều này có thể giải thích rằng:
  - Khi đã bị giới hạn tín dụng, khách hàng bắt đầu chi trả đúng hạn.
  - Ngược lại, khi chưa bị giới hạn, họ có xu hướng gian lận hoặc chi trả không đúng hạn.
- Ở phần **max**, đa phần các giá trị cũng **lớn hơn hoặc bằng** so với trường hợp không bị thiếu.

Kết luận 1: Những điều trên đang củng cố giả thuyết của chúng ta về mối liên hệ giữa việc thiếu `prod_limit` và hành vi tín dụng của khách hàng.

Tuy nhiên, ta cũng chưa nên nghĩ kết luận thế vội. Ta nên tiếp tục phân tích bằng cách: ta tập trung vào các prod_code 22 và 10 do chúng có tồn tại những giá trị vừa null vừa 0 null ở cột prod_limit

Số lượng bản ghi theo ID và Prod_code và update_date (cho prod_limit_is_missing):

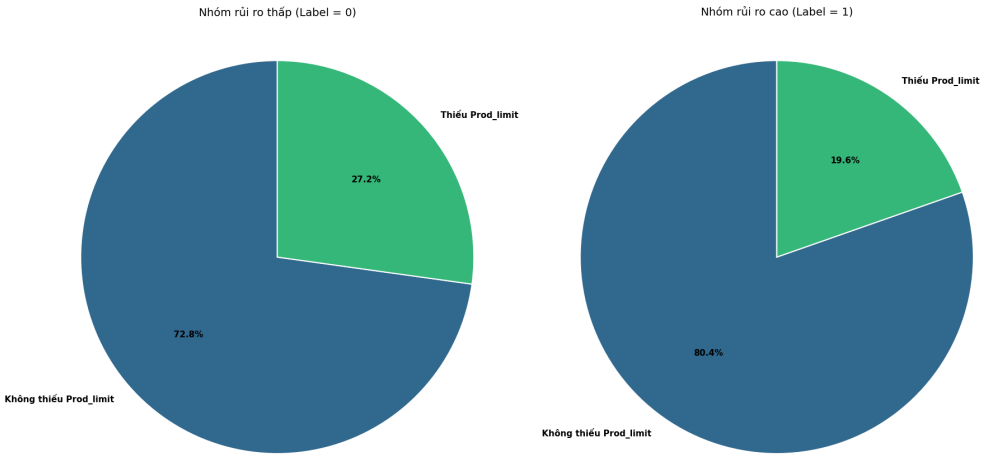
id	prod_code	update_date	prod_limit_is_missing	
54982353	10	1994-12-18	1	1
		1997-04-21	0	1
		2003-01-05	0	1
		2003-09-17	1	1
		2003-10-17	1	1
		2005-04-22	1	1
		2005-09-30	1	1
		2006-01-08	1	1
		2007-03-12	1	2
			0	1
		2011-07-16	1	1
		2013-09-03	0	1
		2014-08-27	0	1
54982356	10	2006-03-30	1	1
54982387	10	2003-09-17	0	1
		2006-08-30	0	1
		2012-02-25	0	1
		2012-07-03	0	1
54982530	10	2013-12-04	1	1
		2014-09-28	1	1

Hình 4.9: Số lượng bản ghi theo ID và Prod_code và update_date (cho prod_limit_is_missing)

KẾT LUẬN 2: Củng cố thêm giả thuyết của ta rằng: có khoảng thời gian, người dùng bị giới hạn, sau được gỡ, sau lại bị giới hạn

KẾT LUẬN CUỐI CÙNG: Vậy giá trị null ở đây có thể là các TH khả nghi như: Khách hàng không đủ điều kiện để được cấp hạn mức tín dụng rõ ràng (có thể liên quan đến rủi ro của họ).Trong một số trường hợp, hạn mức có thể đã từng được áp dụng rồi lại được gỡ bỏ, cho thấy một quy trình quản lý hạn mức động

4. Kiểm tra xem cột "prod_limit_is_missing"mới này có ảnh hưởng đến target không



Hình 4.10: Biểu đồ tròn so sánh

Nhận xét: Như tôi dự đoán ở trên nãy, do tôi vẫn nghĩ có thể khả năng cao null ở đây mang nghĩa không bị giới hạn tín dụng, còn khác null mang nghĩa bị giới hạn tín dụng (có thể do người dùng vi phạm chính sách thẻ dẫn đến rủi ro hay là tự đặt giới hạn cho thẻ khi nhận thấy rủi ro,...) thế nên ở đây prod_limit bị thiếu ở nhóm rủi ro thấp có xu hướng cao hơn prod_limit bị thiếu ở nhóm rủi ro cao. Tuy nhiên, như tôi nói, đây vẫn chỉ là giả định của tôi chứ thực tế ra sao chúng ta cần hỏi người cung cấp dữ liệu

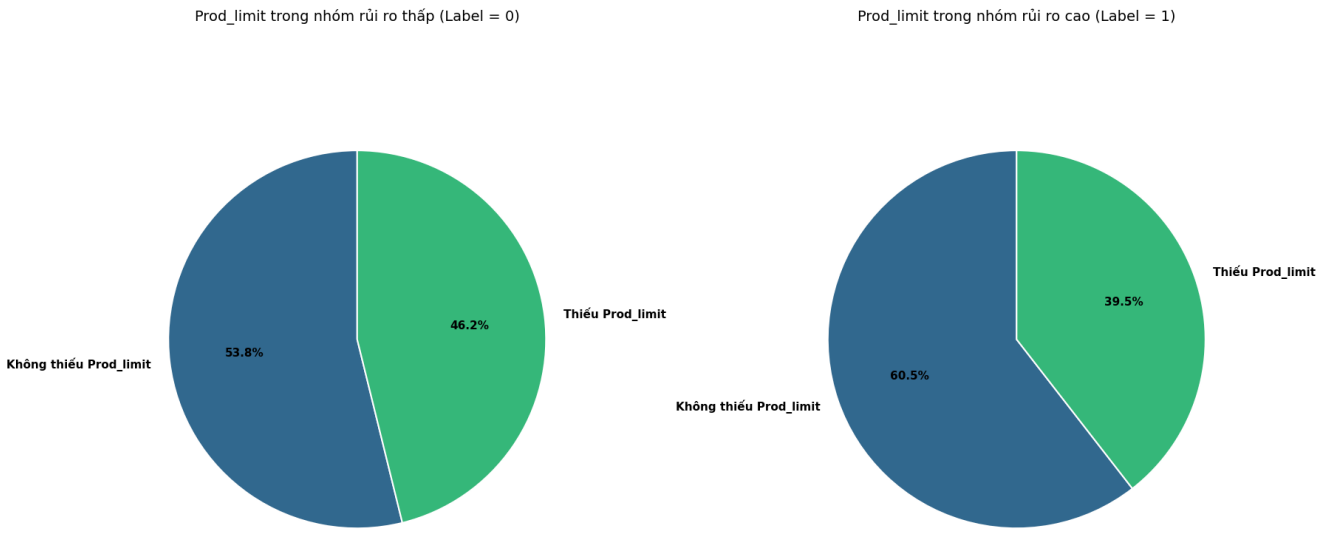
- Hai biểu đồ tròn phía trên được xây dựng dựa trên việc so sánh trong các trường hợp mà bộ ba `(id, label, prod_code)` bị lặp lại, tức là một thẻ của một người xuất hiện nhiều lần với cả giá trị `prod_limit = null` và `prod_limit` $\neq$ `null`, nhưng tất cả các dòng đều có cùng một `label = 1`.
- **Ví dụ minh họa:**

id	prod_code	update_date	prod_limit	label
1999	10	03/02	null	1
1999	10	04/05	null	1
1999	10	04/05	không null	1
1999	10	05/05	null	1
1999	10	06/05	không null	1

- **Vấn đề:** Trong ví dụ này, cùng một khách hàng có nhiều bản ghi với các trạng thái `prod_limit` khác nhau, nhưng tất cả đều có nhãn `label = 1`. Điều này dẫn đến:
  - Việc xuất hiện nhiều lần các bản ghi với `prod_limit = null` và cùng một nhãn làm tăng tỷ trọng của nhóm `null` trong phân tích.
  - Vì chỉ cần khách hàng bị gán `label = 1` một lần thì mọi bản ghi của họ đều mang nhãn đó, nên các biểu đồ dễ bị thiên lệch.

Kết luận: Phân tích như vậy có thể chưa phản ánh đúng bản chất, vì việc so sánh tỷ lệ giữa `null` và `non-null` theo `label` sẽ bị ảnh hưởng nếu không xử lý trùng lặp hoặc không nhóm theo người dùng hợp lý.

Đề xuất xử lý: tạo 1 dataframe khác loại bỏ trùng ở các cột id, prod_limit_is ... với mục tiêu: tránh để prod_limit xuất hiện nhiều lần trên cùng một người



Hình 4.11: Hình ảnh so sánh trung thực hơn

Nhận xét: Ở đây khách quan hơn, tuy vậy chúng ta vẫn thấy prod_limit thiếu ở label = 0 vẫn lớn hơn label = 1. Và ở đây tôi nhận định rằng biến prod_limit_is_missing là có ý nghĩa trong việc dự đoán label

Phương án xử lý:

- Giữ lại biến `prod_limit_is_missing` để đánh dấu thông tin bị thiếu ban đầu.
- Thay thế các giá trị `null` trong cột `prod_limit` bằng giá trị 0, với ý nghĩa:
  - Khách hàng hoặc sản phẩm đó không có hạn mức tín dụng (tại thời điểm đó).
  - Hoặc, hạn mức tín dụng không được áp dụng trong trường hợp này.

4.4.2 Cột report_date

Tìm hiểu nguyên nhân report_date bị thiếu

1. **Giả thuyết 1**: Do chưa bao giờ giao dịch

	OVD_t1	OVD_sum	pay_normal
count	867.000000	867.000000	867.000000
mean	0.220300	59.200692	5.683968
std	1.186395	702.914501	8.363799
min	0.000000	0.000000	0.000000
25%	0.000000	0.000000	1.000000
50%	0.000000	0.000000	2.000000
75%	0.000000	0.000000	6.000000
max	16.000000	18666.000000	36.000000

Hình 4.12: Ảnh kiểm chứng

Kết luận: Sai vì nếu thế OVD_sum ít nhất phải bằng không

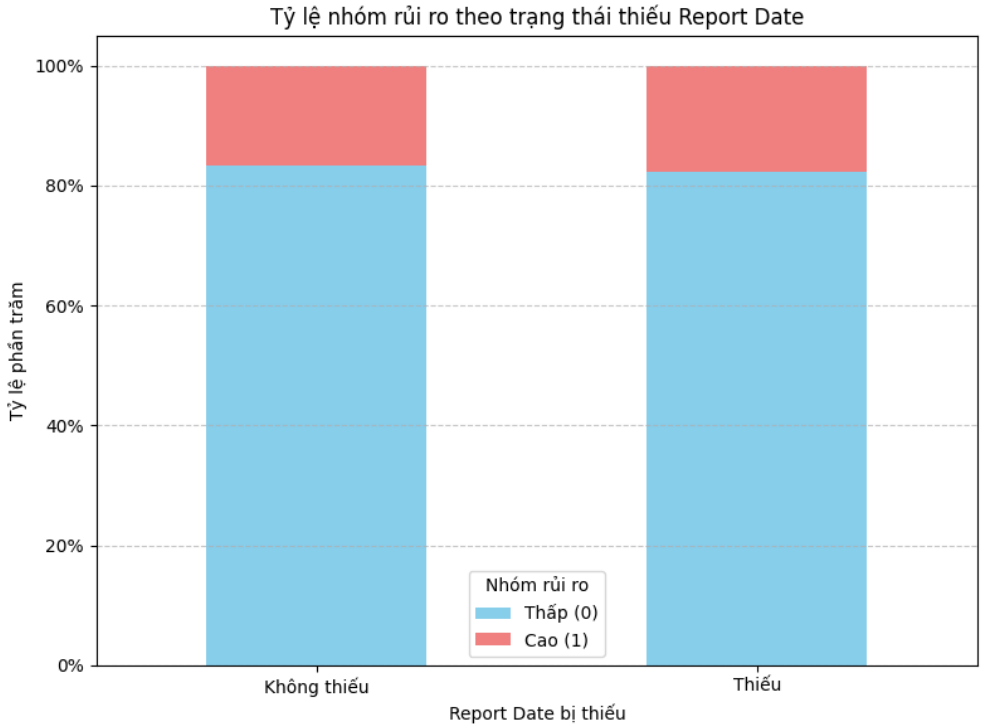
2. **Giả thuyết 2**: Do người thu thập dữ liệu không xác định được thời điểm giao dịch gần nhất ?

```
Tỷ lệ phần trăm Report Date bị thiếu
theo từng ID (sắp xếp từ cao đến thấp)
lấy ra khoảng 20 người:
id
54983809    100.000000
58990258    100.000000
54990339    100.000000
54988633    100.000000
54985624    100.000000
58994549    100.000000
58999709     85.714286
58992481     75.000000
58987924     70.000000
54983721     66.666667
54983279     66.666667
59002775     66.666667
58998925     66.666667
58994429     66.666667
54987936     66.666667
59002314     63.636364
54982356     60.000000
58984708     60.000000
58982376     57.142857
59005841     57.142857
```

Hình 4.13: Ảnh kiểm chứng

Kết luận: Từ đây ta thấy được là có vài người thì tỷ lệ null của cột đó là 100%, còn một vài người thì 75%, 66%,... Vậy nguyên nhân có thể đúng là do người thu thập có thể không xác định được những giao dịch gần đây hoặc chưa cập nhật những giao dịch gần đây vào trong dữ liệu nên nó mới bị null như vậy

3. Xem ảnh hưởng các giá trị NULL của cột report_date lên label



Hình 4.14: Ảnh kiểm tra

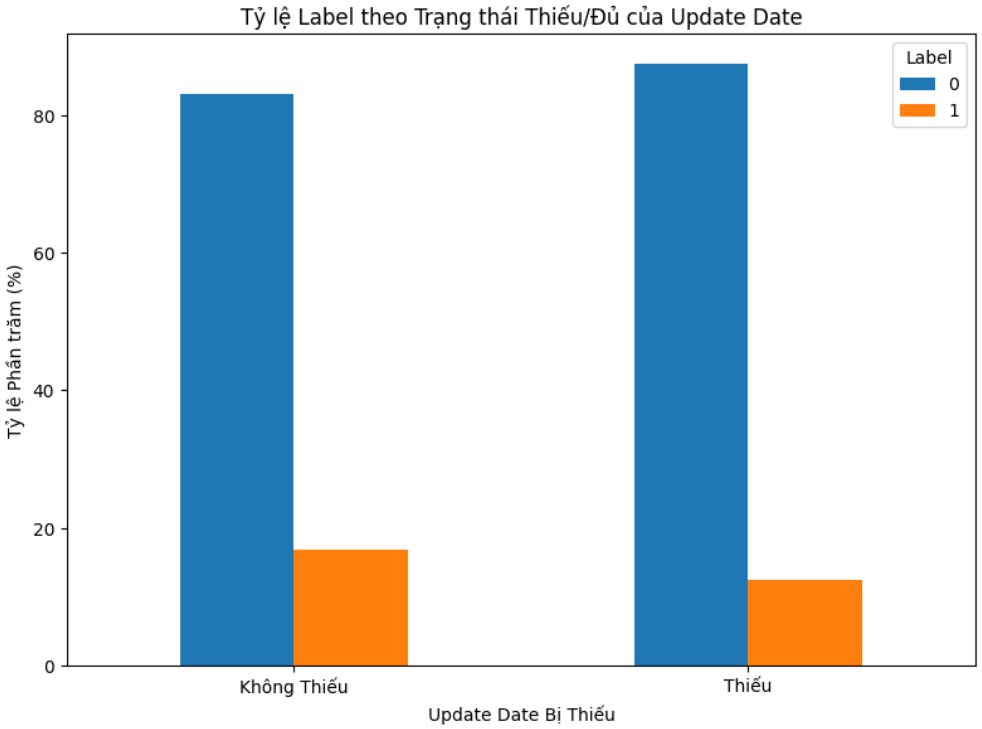
Kết luận: Không có sự khác biệt giữa thiếu/không thiếu report_date đối với label, cho ta thấy được rằng việc thiếu có thể là ngẫu nhiên, không liên quan đến hành vi của khách hàng

Xử lý cột `report_date`:

- Vì cột này có tỷ lệ thiếu tương đối cao (**13.2%**), nên việc tạo thêm một biến nhị phân chỉ báo thiếu là cần thiết để mô hình nhận biết đâu là giá trị thực và đâu là giá trị đã được thay thế
- Phương án xử lý giá trị thiếu:
  - Tách cột `report_date` thành các thành phần: **ngày, tháng, năm**.
  - Thay thế các giá trị thiếu bằng kỹ thuật **KNN Imputer** sau khi chia tập dữ liệu huấn luyện và kiểm tra.

4.4.3 Cột update_date

Lưu ý: *Cột này tuy trước đó em đã phân tích, nhưng vẫn chưa hiểu rõ ý nghĩa thực sự của cột nên đã xóa đi phần phân tích đó. Vậy nên, em xin phép không phân tích*
Xem ảnh hưởng của biến này lên label



Hình 4.15: Xem ảnh hưởng của update_date lên target

Kết luận: Không có sự khác biệt đáng kể. Mặt khác biến này thiếu khá ít nên không cần tạo biến chỉ báo

Phương án xử lý: chia thành 3 cột ngày tháng năm và thay NULL bằng KNN Imputer

4.4.4 Cột highest_balance

Thay thế null của cột này bằng trung bình highest_balance tương ứng của mỗi người

4.4.5 Cột fea_2

Do phân phối của fea_2 là phân phối gần chuẩn, mặt khác giá trị của fea_2 chỉ có giá trị nguyên nên ta sẽ thay giá trị cột fea_2 bằng **median**

4.5 Tiếp tục phân tích và đưa ra phương án xử lý

- Quyết định bỏ biến `OVD_t3` vì `OVD_sum` mang tính tổng quát hơn.
- Giữ lại hai biến `new_balance` và `highest_balance` vì:
  - `new_balance`: Phản ánh tình hình tài chính hiện tại của khách hàng tại thời điểm `update_date` (hoặc `report_date` gì đó). Đây là thông tin rất quan trọng và cập nhật.
  - `highest_balance`: Phản ánh mức số dư cao nhất trong lịch sử (tính đến `update_date` đó hoặc một thời điểm trước đó).

5 Biến đổi dữ liệu

5.1 Một vài nguyên tắc tránh Data Leakage

1. **Chuẩn hóa và Điền dữ liệu (Scaling & Imputation)**
   Các tham số như giá trị trung bình (mean), trung vị (median), độ lệch chuẩn (std), giá trị min/max phải được tính toán ('fit') **chỉ trên tập huấn luyện**. Sau đó, các tham số đã học này được dùng để áp dụng ('transform') cho cả tập huấn luyện và tập kiểm tra. Các công cụ như `StandardScaler`, `MinMaxScaler`, `SimpleImputer` phải tuân thủ quy trình này.

2. **Lựa chọn Đặc trưng (Feature Selection)**
   Mọi quyết định về việc giữ hay loại bỏ một đặc trưng phải được dựa hoàn toàn vào thông tin từ tập huấn luyện. Ví dụ, việc chọn các đặc trưng có tương quan cao nhất với biến mục tiêu phải được thực hiện trên tập huấn luyện, sau đó áp dụng lựa chọn đó cho tập kiểm tra.

3. **Mã hóa Biến phân loại (Categorical Encoding)**
   Các phương pháp mã hóa "học"từ dữ liệu phải tuân thủ nguyên tắc trên. Điều này đặc biệt nguy hiểm với **Target Encoding**, nơi giá trị trung bình của biến mục tiêu được sử dụng. Việc tính toán này phải được thực hiện riêng trên tập huấn luyện để tránh rò rỉ trực tiếp thông tin của biến mục tiêu từ tập kiểm tra vào quá trình huấn luyện.

4. **Giải pháp Tối ưu: Sử dụng `Pipeline` của Scikit-learn**
   Cách làm chuyên nghiệp và an toàn nhất là đóng gói tất cả các bước tiền xử lý và mô hình vào một `Pipeline`. `Pipeline` tự động đảm bảo rằng quá trình 'fit' và 'transform' được áp dụng đúng cách, ngăn chặn rò rỉ dữ liệu một cách hiệu quả, đặc biệt là khi thực hiện kiểm tra chéo (Cross-Validation).

5.2 Biến đổi trên tập dữ liệu này

Do không thể hiện được code tại đây nên tôi sẽ tóm tắt các biến đổi của chúng ta lên train và test như sau (Tránh việc data leakage):

1. **Xử lý Dữ liệu Thiếu (Handling Missing Data)**
   Một chiến lược đa tầng đã được áp dụng để xử lý các giá trị thiếu, tùy thuộc vào đặc điểm của từng cột:

   - **Imputation đơn giản:**
     - Cột `fea_2`: Điền giá trị thiếu bằng giá trị trung vị (median) của tập huấn luyện, phù hợp với phân phối gần đối xứng của cột này.

- Cột `prod_limit`: Điền giá trị thiếu bằng hằng số `0`, với giả định rằng giá trị thiếu có nghĩa là không có hạn mức sản phẩm.

- **Imputation theo nhóm và dự phòng (Grouped & Fallback):**
  - Cột `highest_balance`: Áp dụng một chiến lược phức tạp để có giá trị điền chính xác nhất:
    (a) **Theo nhóm:** Tính giá trị xuất hiện nhiều nhất (mode) của `highest_balance` cho từng nhóm (`id`, `prod_code`) trên tập huấn luyện.
    (b) **Dự phòng cấp 1:** Nếu vẫn còn giá trị thiếu (ví dụ: một nhóm mới trong tập test), điền bằng giá trị mode toàn cục của cột `highest_balance` từ tập huấn luyện.
    (c) **Dự phòng cấp 2:** Trong trường hợp hiếm hoi không có mode, điền bằng giá trị median toàn cục của tập huấn luyện.

- **Imputation đa biến (Multivariate Imputation):**
  - Đối với các giá trị thiếu còn lại trong các cột số (bao gồm các cột ngày tháng đã được tách), thuật toán `KNNImputer` với $k = 5$ được sử dụng. Phương pháp này dự đoán giá trị thiếu dựa trên các đặc trưng của 5 hàng xóm gần nhất, giúp nắm bắt các mối quan hệ phức tạp giữa các biến.

2. **Biến đổi Đặc trưng Số (Numerical Feature Transformation)**
Các đặc trưng số được xử lý qua hai bước để ổn định phân phối và đồng bộ hóa thang đo:

- **Giảm độ lệch (Skewness Reduction):** Áp dụng phép biến đổi **Yeo-Johnson** cho các cột có phân phối lệch nặng. Phép biến đổi này giúp làm cho phân phối của dữ liệu trở nên gần với phân phối chuẩn hơn, có lợi cho nhiều thuật toán.
- **Chuẩn hóa (Standardization):** Sau khi giảm độ lệch, tất cả các cột số được chuẩn hóa bằng `StandardScaler`. Quá trình này đưa tất cả các đặc trưng về cùng một thang đo với giá trị trung bình là 0 và độ lệch chuẩn là 1.

3. **Mã hóa Đặc trưng Phân loại (Categorical Feature Encoding)**
Các đặc trưng phân loại được chuyển đổi sang dạng số bằng phương pháp **One-Hot Encoding**. Tham số `handle_unknown='ignore'` được sử dụng để bỏ qua các hạng mục có thể xuất hiện trong tập kiểm tra nhưng không có trong tập huấn luyện, tránh gây lỗi trong quá trình dự đoán.

4. **Hoàn thiện Dữ liệu (Finalizing the Dataset)**
Cuối cùng, các cột định danh không mang tính dự báo như `id` và `prod_code` được loại bỏ. Các đặc trưng số đã qua xử lý và các đặc trưng phân loại đã

được mã hóa được kết hợp lại để tạo thành bộ dữ liệu cuối cùng, sẵn sàng cho việc huấn luyện mô hình.

6 Tiêu chí đánh giá

Chiến lược đánh giá được chia thành hai nhóm chỉ số chính:

6.1 Chỉ số chính: để so sánh và lựa chọn mô hình

Đây là các chỉ số tổng quan, không phụ thuộc vào ngưỡng quyết định, được dùng để so sánh sức mạnh phân biệt của các thuật toán khác nhau.

- **AUC-ROC (Area Under the ROC Curve):** Chỉ số tiêu chuẩn để đo lường khả năng phân biệt giữa hai lớp của mô hình. Giá trị AUC càng gần 1, mô hình càng có khả năng phân biệt tốt.

Mô hình có giá trị AUC-ROC cao nhất sẽ được lựa chọn là mô hình tốt nhất.

6.2 Chỉ số phụ: để diễn giải và ra quyết định kinh doanh

Sau khi đã chọn được mô hình tốt nhất, các chỉ số sau được sử dụng để phân tích hiệu suất tại một ngưỡng quyết định cụ thể, giúp cân bằng giữa rủi ro và chi phí hoạt động.

- **Recall (Sensitivity):** Đo lường tỷ lệ khách hàng rủi ro thực tế mà mô hình đã xác định được. *Mục tiêu là tối đa hóa Recall để giảm thiểu tổn thất tài chính do bỏ sót rủi ro.*
- **Precision:** Đo lường độ chính xác của các dự đoán rủi ro. *Mục tiêu là giữ Precision ở mức chấp nhận được để tối ưu hóa chi phí thẩm định.*
- **F1-Score:** Là trung bình điều hòa của Precision và Recall, cung cấp một thước đo cân bằng giữa hai chỉ số trên.

7 Mô hình 1 (Baseline): Logistic Regression

7.1 Mô hình sử dụng

```
LogisticRegression(random_state=42, solver='liblinear', class_weight='balanced')
```

Giải thích:

- **solver='liblinear'**:
  - Chọn thuật toán tối ưu hóa là `liblinear` (dựa trên thư viện LIBLINEAR).
  - Phù hợp với bài toán phân loại nhị phân hoặc dữ liệu không quá lớn.
  - Thường được sử dụng khi cần kết hợp với tham số `class_weight`.
- **class_weight='balanced'**:
  - Tự động điều chỉnh trọng số cho các lớp dựa trên tỷ lệ xuất hiện trong tập huấn luyện.
  - Giúp mô hình tập trung hơn vào các lớp thiểu số trong trường hợp dữ liệu bị mất cân bằng.
  - Trọng số mỗi lớp được tính theo công thức:

$$w_i = \frac{n_{\text{samples}}}{n_{\text{classes}} \times n_{\text{samples\_in\_class}_i}}$$

trong đó w_i là trọng số của lớp i, n_{samples} là tổng số mẫu, n_{classes} là số lượng lớp, và $n_{\text{samples\_in\_class}_i}$ là số mẫu thuộc lớp i.

7.2 Kết quả trên tập test

```
Confusion Matrix (Test):
[[916 441]
 [ 90 185]]

Classification Report (Test):
              precision    recall  f1-score   support

           0       0.91      0.68      0.78      1357
           1       0.30      0.67      0.41       275

    accuracy                           0.67      1632
   macro avg       0.60      0.67      0.59      1632
weighted avg       0.81      0.67      0.71      1632


Accuracy (Test): 0.6746
Precision (Test - cho lớp 1): 0.2955
Recall (Test - cho lớp 1): 0.6727
F1-score (Test - cho lớp 1): 0.4107
ROC AUC (Test): 0.7265
```

Hình 7.1: Mô hình 1

7.3 Nhận xét

1. **Khả năng Phân biệt Tổng quan:**
   Chỉ số **ROC AUC đạt 0.7265**, cho thấy mô hình có khả năng phân biệt giữa khách hàng rủi ro và không rủi ro tốt hơn đáng kể so với việc đoán ngẫu nhiên. Đây là một kết quả khởi đầu tích cực.

2. **Hiệu suất trên Lớp Rủi ro (Lớp 1):**
   Đây là khía cạnh quan trọng nhất của bài toán.

   - **Điểm mạnh (Recall = 0.67):** Mô hình đã thành công trong việc xác định được **67%** trong tổng số các khách hàng thực sự rủi ro. Đây là một kết quả quan trọng, cho thấy mô hình có khả năng giảm thiểu một phần đáng kể các khoản tổn thất tiềm năng.
   - **Điểm yếu (Precision = 0.30):** Đây là thách thức lớn nhất. Khi mô hình dự đoán một khách hàng là rủi ro, chỉ có **30%** trong số đó là dự đoán đúng. 70% còn lại là các trường hợp báo động nhầm (False

Positives), gây tốn kém chi phí thẩm định và có thể ảnh hưởng đến trải nghiệm của khách hàng tốt.

- **Sự cân bằng (F1-score = 0.41):** Chỉ số F1-score thấp cho thấy sự đánh đổi giữa Precision và Recall hiện tại chưa tối ưu.

3. **Diễn giải Ma trận Nhầm lẫn (Confusion Matrix):**

- **False Negatives (90 trường hợp):** Mô hình đã bỏ sót 90 khách hàng rủi ro, đây là rủi ro tài chính trực tiếp cần được giảm thiểu.
- **False Positives (441 trường hợp):** Mô hình đã phân loại nhầm 441 khách hàng tốt thành khách hàng rủi ro, đây là chi phí vận hành cần được tối ưu.

8 Mô hình 2: Random Forest phiên bản 1

8.1 Mô hình sử dụng

```
class_weight_input = 'balanced_subsample'
                    if y_train.value_counts(normalize=True)[1] < 0.4 else None

rf_model_v2 = RandomForestClassifier(random_state=42,
                                     n_estimators=100,
                                     class_weight= class_weight_input)
```

Giải thích:

- **n_estimators=100**:
  - Số lượng cây (trees) trong rừng ngẫu nhiên.
  - Càng nhiều cây thường cho độ ổn định cao hơn nhưng tốn thời gian huấn luyện và dự đoán.
- **class_weight='balanced_subsample' if $\hat{p}_1 < 0.4$ else None**:
  - Tính $\hat{p}_1 = \frac{\#\text{mẫu lớp 1}}{\#\text{tổng mẫu}}$ trên tập huấn luyện.
  - Nếu $\hat{p}_1 < 0.4$ (target bị mất cân bằng đáng kể), đặt

    `class_weight='balanced_subsample'`

    để tự động điều chỉnh trọng số mỗi lớp $w_i = \frac{n_{\text{bootstrap}}}{n_{\text{classes}} \times n_{\text{samples\_in\_class}_i}}$ trên mỗi bootstrap sample, giúp mô hình không bỏ qua lớp thiểu số.
  - Ngược lại ($\hat{p}_1 \geq 0.4$), đặt `class_weight=None` để không cân bằng trọng số (giữ mặc định).

8.2 Kết quả trên tập test

```
Confusion Matrix (Test):
[[1357    0]
 [ 135  140]]

Classification Report (Test):
              precision    recall  f1-score   support

           0       0.91      1.00      0.95      1357
           1       1.00      0.51      0.67       275

    accuracy                           0.92      1632
   macro avg       0.95      0.75      0.81      1632
weighted avg       0.92      0.92      0.91      1632


Accuracy (Test): 0.9173
Precision (Test - cho lớp 1): 1.0000
Recall (Test - cho lớp 1): 0.5091
F1-score (Test - cho lớp 1): 0.6747
ROC AUC (Test): 0.9867
```

Hình 8.1: Mô hình 2

8.3 Nhận xét

1. **Sự Cải thiện Vượt bậc về Khả năng Phân biệt (ROC AUC = 0.9867):**
   Chỉ số ROC AUC tăng vọt lên 0.9867, một kết quả gần như hoàn hảo. Điều này khẳng định rằng mô hình RandomForest đã thành công trong việc nắm bắt các mối quan hệ phi tuyến tính phức tạp trong dữ liệu, vượt xa khả năng của mô hình tuyến tính.

2. **Độ Chính xác Tuyệt đối (Precision = 1.00) - Một Kết quả "Hoàn hảo":**
   Điểm nổi bật nhất là Precision cho lớp rủi ro (lớp 1) đạt giá trị tuyệt đối 1.00. Điều này có nghĩa là **100% các cảnh báo rủi ro của mô hình đều chính xác**. Ma trận nhầm lẫn xác nhận không có bất kỳ trường hợp báo động nhầm (False Positives = 0) nào. Về mặt nghiệp vụ, điều này giúp loại bỏ hoàn toàn chi phí thẩm định cho các trường hợp không cần thiết.

3. **Sự Đánh đổi và Rủi ro Tiềm ẩn (Recall = 0.51):**
   Cái giá phải trả cho độ chính xác tuyệt đối là mô hình trở nên quá thận

trọng. Với Recall chỉ đạt 0.51, mô hình đã **bỏ sót gần một nửa (135/275) số khách hàng thực sự rủi ro**. Đây là một rủi ro tài chính đáng kể cần được xem xét. Mô hình hiện tại chỉ có khả năng xác định các trường hợp rủi ro rõ ràng nhất.

9 Mô hình 3: Random Forest phiên bản 2

9.1 Mô hình sử dụng

- Mô hình áp dụng `RandomizedSearchCV` để tự động tìm kiếm bộ siêu tham số tối ưu.
- Không gian siêu tham số được khởi tạo như sau:

```
    param_distributions = {
        'n_estimators': randint(100, 500),
        'max_depth': [5, 8, 10, 15, 20, 25, 30, None],
        'min_samples_split': randint(2, 21),
        'min_samples_leaf': randint(1, 21),
        'max_features': ['sqrt', 'log2', 0.3, 0.5, 0.7],
        'class_weight': [None, 'balanced', 'balanced_subsample
          '],
        'bootstrap': [True, False]
    }
```

- Khởi tạo mô hình cơ bản:

```
rf_clf = RandomForestClassifier(random_state=42)
```

- Thiết lập và chạy `RandomizedSearchCV`:

```
random_search = RandomizedSearchCV(
    estimator=rf_clf,
    param_distributions=param_distributions,
    n_iter=50,
    cv=3,
    verbose=2,
    random_state=42,
    n_jobs=-1,
    scoring='roc_auc'
)
```

- Kết quả siêu tham số tốt nhất tìm được:

```
{
    'bootstrap': False,
    'class_weight': 'balanced',
    'max_depth': 30,
    'max_features': 0.7,
    'min_samples_leaf': 1,
    'min_samples_split': 2,
    'n_estimators': 225
}
```

9.2 Kết quả trên tập test

```
Confusion Matrix (Test):
[[1353    4]
 [  11  264]]

Classification Report (Test):
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      1357
           1       0.99      0.96      0.97       275

    accuracy                           0.99      1632
   macro avg       0.99      0.98      0.98      1632
weighted avg       0.99      0.99      0.99      1632


Accuracy (Test): 0.9908
Precision (Test - cho lớp 1): 0.9851
Recall (Test - cho lớp 1): 0.9600
F1-score (Test - cho lớp 1): 0.9724
ROC AUC (Test): 0.9992
```

Hình 9.1: Mô hình 3

9.3 Đánh giá

1. **Khả năng Phân biệt Gần như Tuyệt đối (ROC AUC = 0.9992):** Chỉ số ROC AUC đạt mức gần như hoàn hảo, cho thấy mô hình có sức mạnh phân biệt cực kỳ cao. Điều này khẳng định hiệu quả của việc tối ưu hóa siêu tham số và khả năng của RandomForest trong việc nắm bắt các mối quan hệ phức tạp trong dữ liệu.

2. **Sự Cân bằng Lý tưởng giữa Rủi ro và Chi phí (Precision & Recall):** Đây là thành tựu lớn nhất của mô hình. Nó đã đạt được "điểm ngọt"mà mọi bài toán rủi ro tín dụng đều hướng tới:

   - **Recall rất cao (0.960):** Mô hình đã thành công trong việc xác định được 96% trong tổng số các khách hàng thực sự rủi ro. Điều này có ý nghĩa cực kỳ lớn về mặt kinh doanh, giúp **giảm thiểu tối đa tổn thất tài chính** do bỏ sót các khoản nợ xấu.
   - **Precision rất cao (0.985):** Khi mô hình đưa ra một cảnh báo rủi ro, nó chính xác đến 98.5%. Điều này giúp **tối ưu hóa hoàn toàn chi phí**

vận hành, đảm bảo đội ngũ thẩm định không lãng phí nguồn lực vào các trường hợp báo động nhầm.

3. **Độ tin cậy và Ổn định:**
   Chỉ số **F1-score (0.972)** rất cao là minh chứng rõ ràng cho sự cân bằng xuất sắc giữa Precision và Recall. Ma trận nhầm lẫn cho thấy số lượng lỗi của mô hình là rất thấp, củng cố thêm độ tin cậy của mô hình.

9.4 Cảnh báo rủi ro

Một hiệu suất cao đến mức này (AUC $>$ 0.999) là một **dấu hiệu cảnh báo rất mạnh** về khả năng tồn tại của **rò rỉ dữ liệu (data leakage) ở cấp độ khái niệm**. Mặc dù quy trình tiền xử lý trong code đã được thực hiện đúng cách, cần phải đặt nghi vấn về nguồn gốc của các đặc trưng: liệu có biến nào chứa thông tin từ "tương lai"hoặc thông tin quá gần với biến mục tiêu hay không?

9.5 Có phải vấn đề do xử lý kém ?

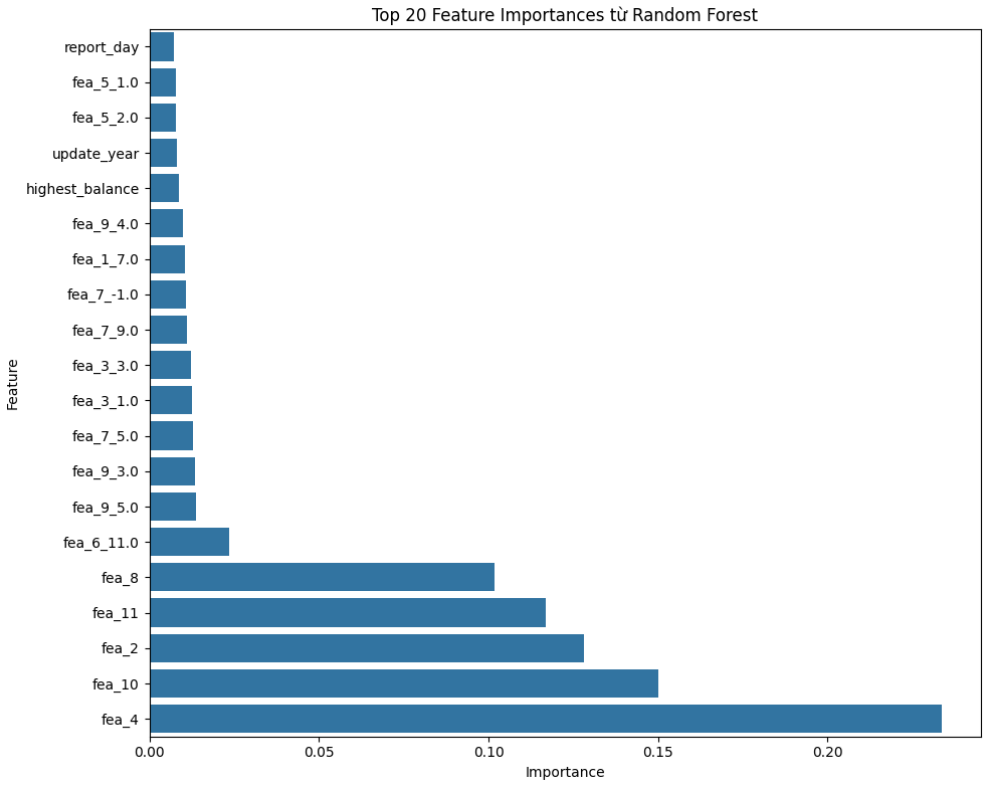
Thực ra không phải vậy, thực chất là:

- **Quy trình xử lý đúng:** Các bước tiền xử lý (scaling, imputation, encoding) đã được thực hiện đúng quy trình, tuân thủ nguyên tắc `fit` trên tập huấn luyện và `transform` trên tập kiểm tra. Loại rò rỉ do sai sót trong quy trình đã được kiểm soát.

- **Vấn đề nằm ở dữ liệu gốc:** Vấn đề thực sự nằm ở **rò rỉ ở cấp độ dữ liệu**. Kết quả mô hình học thuộc lòng tập huấn luyện (điểm tuyệt đối 1.0) và việc tầm quan trọng tập trung cao độ vào một vài biến `fea_*` cho thấy các biến này rất có thể là các **biến đại diện (proxy variables)** hoặc chứa **thông tin từ tương lai**. Chúng là hệ quả của kết quả, chứ không phải nguyên nhân.

10 Kiểm tra các rủi ro tiềm tàng

10.1 In ra feature importance của mô hình 3



Hình 10.1: Các biến đặc trưng quan trọng đối mô hình 3

Nhận xét:

- Mặc dù các chỉ số rất ấn tượng, một hiệu suất cao đến mức này đòi hỏi một sự xem xét lại một cách cẩn trọng. Phân tích tầm quan trọng của các đặc trưng (Feature Importances) cho thấy hiệu suất của mô hình phụ thuộc rất lớn vào một nhóm nhỏ các đặc trưng ẩn danh (`fea_4`, `fea_10`, `fea_2`, `fea_11`, `fea_8`).

- Sự phụ thuộc này làm dấy lên một mối lo ngại nghiêm trọng về khả năng tồn tại của **rò rỉ dữ liệu (data leakage)**. Cụ thể, có khả năng các đặc trưng này là các **biến đại diện (proxy variables)** hoặc chứa **thông tin từ tương lai**. Nói cách khác, chúng có thể là hệ quả của việc một khách hàng đã bị xác định là rủi ro, chứ không phải là nguyên nhân.

- Do đó, có khả năng mô hình đang học cách "nhận dạng"một khách hàng đã được hệ thống đánh dấu là rủi ro, thay vì "dự đoán"một khách hàng sẽ trở nên rủi ro.

10.2 In ra các chỉ số đánh giá ở tập train

```
Confusion Matrix (Train):
[[5427    0]
 [   0 1100]]

Classification Report (Train):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5427
           1       1.00      1.00      1.00      1100

    accuracy                           1.00      6527
   macro avg       1.00      1.00      1.00      6527
weighted avg       1.00      1.00      1.00      6527


Accuracy (Train): 1.0000
Precision (Train - cho lớp 1): 1.0000
Recall (Train - cho lớp 1): 1.0000
F1-score (Train - cho lớp 1): 1.0000
ROC AUC (Train): 1.0000
```

Hình 10.2: Các chỉ số đánh giá ở tập train

Toàn bộ quá trình phân tích đã cung cấp một chuỗi bằng chứng nhất quán, củng cố cho kết luận về rò rỉ dữ liệu:

1. **Hiệu suất tuyệt đối trên tập huấn luyện:** Mô hình đã học thuộc lòng dữ liệu.
2. **Hiệu suất gần như tuyệt đối trên tập kiểm tra:** Quy luật học được từ các biến rò rỉ vẫn đúng trên tập kiểm tra.
3. **Tầm quan trọng đặc trưng tập trung cao độ:** Mô hình chỉ dựa vào một vài biến bị rò rỉ để đưa ra quyết định.

Kết luận cuối cùng: Hiệu suất ấn tượng của mô hình không phản ánh khả năng dự đoán thực tế mà là kết quả của việc khai thác các lỗ hổng trong dữ liệu. Do đó, mô hình này **không đáng tin cậy** để triển khai trong thực tế. Các bước tiếp theo bắt buộc phải tập trung vào việc xác định và loại bỏ các đặc trưng gây rò rỉ dữ liệu để xây dựng lại một mô hình có ý nghĩa.

10.3 Các hướng hành động phổ biến

1. **Xác thực lại Dữ liệu (Ưu tiên hàng đầu):** Cần làm việc với bộ phận cung cấp dữ liệu để làm rõ ý nghĩa của các biến `fea_*` quan trọng nhất, nhằm xác nhận chúng không phải là các biến bị rò rỉ.

2. **Xây dựng lại Mô hình An toàn:** Nếu không thể xác thực dữ liệu, phương án an toàn nhất là loại bỏ các biến đáng ngờ và xây dựng lại toàn bộ quy trình, bao gồm cả việc tối ưu hóa lại siêu tham số.

3. **Chấp nhận Kết quả Thực tế:** Mô hình mới sau khi loại bỏ các biến rò rỉ chắc chắn sẽ có hiệu suất thấp hơn, nhưng đó mới là kết quả trung thực và đáng tin cậy, phản ánh đúng khả năng dự đoán của mô hình.

10.4 Chốt hướng hành động

Do các biến fea_* là các biến đã được mã hóa, mặt khác người cung cấp dữ liệu là người ngoại quốc và dữ liệu được gửi lên từ rất lâu vậy nên khó mà hỏi người cung cấp để hiểu được. Thế nên thời điểm này, xây dựng lại mô hình an toàn hơn có lẽ là biện pháp tốt hơn cả

11 Mô hình 4: Random Forest phiên bản 3

11.1 Mô hình sử dụng

Dựa trên các phân tích trước, trong mô hình này, chúng ta sẽ loại bỏ các biến có mức độ quan trọng cao, bao gồm: `fea_2`, `fea_11`, `fea_10`, `fea_4`, và `fea_8`.

Tiếp tục sử dụng **RandomizedSearchCV** để tìm kiếm bộ siêu tham số tối ưu cho mô hình **Random Forest** trong trường hợp này.

11.1.1 Kết quả tìm kiếm

Bộ siêu tham số tối ưu thu được từ **RandomizedSearchCV** như sau:

- `bootstrap`: **False**
- `class_weight`: **balanced**
- `max_depth`: **25**
- `max_features`: **log2**
- `min_samples_leaf`: **4**
- `min_samples_split`: **19**
- `n_estimators`: **317**

11.1.2 Giải thích bộ siêu tham số

- `bootstrap = False`: Không sử dụng kỹ thuật bootstrap, tức là toàn bộ tập dữ liệu được sử dụng để xây dựng mỗi cây quyết định, giúp tăng độ chính xác trong trường hợp dữ liệu không quá lớn.
- `class_weight = balanced`: Cân bằng trọng số lớp để xử lý dữ liệu mất cân bằng, đảm bảo mô hình không thiên vị lớp chiếm ưu thế.
- `max_depth = 25`: Giới hạn độ sâu tối đa của cây là 25, giúp kiểm soát độ phức tạp và tránh hiện tượng quá khớp (overfitting).
- `max_features = log2`: Số lượng đặc trưng tối đa được xem xét tại mỗi lần phân chia là logarit cơ số 2 của tổng số đặc trưng, giúp giảm độ tương quan giữa các cây.
- `min_samples_leaf = 4`: Yêu cầu tối thiểu 4 mẫu tại mỗi nút lá, giúp làm mịn mô hình và giảm nhiễu.

- `min_samples_split = 19`: Yêu cầu tối thiểu 19 mẫu để phân chia một nút, tăng tính ổn định của cây quyết định.
- `n_estimators = 317`: Số lượng cây trong rừng là 317, đủ lớn để đảm bảo tính mạnh mẽ và hiệu suất cao của mô hình.

11.2 Kết quả trên cả tập train và test

11.2.1 Kết quả trên tập train

```
Confusion Matrix (Train):
[[5069  358]
 [   1 1099]]

Classification Report (Train):
              precision    recall  f1-score   support

           0       1.00      0.93      0.97      5427
           1       0.75      1.00      0.86      1100

    accuracy                           0.94      6527
   macro avg       0.88      0.97      0.91      6527
weighted avg       0.96      0.94      0.95      6527


Accuracy (Train): 0.9450
Precision (Train - cho lớp 1): 0.7543
Recall (Train - cho lớp 1): 0.9991
F1-score (Train - cho lớp 1): 0.8596
ROC AUC (Train): 0.9932
```

Hình 11.1: Kết quả tập train mô hình 4

11.2.2 Kết quả trên tập test

```
Confusion Matrix (Test):
[[1170  187]
 [ 147  128]]

Classification Report (Test):
              precision    recall  f1-score   support

           0       0.89      0.86      0.88      1357
           1       0.41      0.47      0.43       275

    accuracy                           0.80      1632
   macro avg       0.65      0.66      0.65      1632
weighted avg       0.81      0.80      0.80      1632


Accuracy (Test): 0.7953
Precision (Test - cho lớp 1): 0.4063
Recall (Test - cho lớp 1): 0.4655
F1-score (Test - cho lớp 1): 0.4339
ROC AUC (Test): 0.7749
```

Hình 11.2: Kết quả tập test mô hình 4

11.3 Đánh giá

Như ta thấy rằng mô hình có vẻ đang bị **overfitting** do kết quả trên tập train khá tốt nhưng tập test lại rất tệ.

11.4 Các hướng có thể xử lý tiếp theo

1. **Tối ưu hóa lại Siêu tham số để Chống Overfitting (Ưu tiên hàng đầu):**
   - **Mục tiêu:** Tìm một bộ siêu tham số mới cho RandomForest giúp mô hình tổng quát hóa tốt hơn trên dữ liệu mới, giảm khoảng cách hiệu suất giữa tập huấn luyện và tập kiểm tra.
   - **Hành động:** Chạy lại quy trình `RandomizedSearchCV` trên bộ dữ liệu đã được làm sạch (đã loại bỏ các biến `fea_*`). Bộ tham số tìm được dự kiến sẽ "đơn giản"hơn (ví dụ: `max_depth` nhỏ hơn, `min_samples_leaf` lớn hơn).
2. **Thử nghiệm các Thuật toán Nâng cao:**
   - **Mục tiêu:** Khai thác sức mạnh của các thuật toán Gradient Boosting, vốn có các cơ chế chống overfitting mạnh mẽ.
   - **Hành động:** Xây dựng và tối ưu hóa các mô hình sử dụng **XGBoost** và **LightGBM**. So sánh hiệu suất của chúng với mô hình RandomForest đã được tối ưu hóa ở bước 1.
3. **Kỹ thuật Đặc trưng (Feature Engineering) Sáng tạo:**
   - **Mục tiêu:** Cung cấp thêm các tín hiệu dự báo mạnh mẽ hơn cho mô hình bằng cách tạo ra các đặc trưng mới từ dữ liệu hiện có.
   - **Hành động:** Tạo các biến mới có ý nghĩa nghiệp vụ cao, ví dụ:
     - Các biến tỷ lệ: Tỷ lệ sử dụng hạn mức (`new_balance / prod_limit`), tỷ lệ quá hạn.
     - Các biến về thời gian: Khoảng thời gian kể từ lần cập nhật/báo cáo cuối cùng.
     - Các biến tương tác giữa các đặc trưng.

12 Có nên sử dụng SMOTE bây giờ không ?

Một câu hỏi quan trọng được đặt ra là liệu có nên sử dụng kỹ thuật lấy mẫu lại như SMOTE (Synthetic Minority Over-sampling Technique) để xử lý vấn đề mất cân bằng dữ liệu hay không.

12.1 Phân tích

Mặc dù SMOTE là một kỹ thuật phổ biến, việc áp dụng nó trong bối cảnh hiện tại là **không được khuyến khích** và có thể phản tác dụng. Lý do chính là vấn đề cốt lõi của mô hình hiện tại là **overfitting (quá khớp)**, không phải là sự thiên vị do mất cân bằng.

- **Nguy cơ làm trầm trọng thêm Overfitting:** SMOTE hoạt động bằng cách tạo ra các mẫu dữ liệu tổng hợp cho lớp thiểu số. Điều này có thể khiến mô hình học các quy luật quá cụ thể và "chật hẹp"cho tập huấn luyện, làm tăng khoảng cách hiệu suất giữa tập huấn luyện và tập kiểm tra.
- **Các phương pháp thay thế đã được sử dụng:** Trong quá trình tối ưu hóa siêu tham số, các phương pháp xử lý mất cân bằng an toàn hơn như `class_weight='balanced'` đã được đưa vào xem xét. Các phương pháp này điều chỉnh hàm mất mát thay vì tạo dữ liệu giả, thường hiệu quả và ít rủi ro hơn.

12.2 Kết luận và Đề xuất

SMOTE không phải là giải pháp cho vấn đề overfitting hiện tại. Thay vào đó, các nỗ lực cần được tập trung vào việc **giảm overfitting trước tiên** thông qua các phương pháp như:

1. Tối ưu hóa lại siêu tham số để tìm một mô hình đơn giản hơn.
2. Sử dụng các thuật toán có cơ chế điều chuẩn (regularization) mạnh mẽ như XGBoost.

Chỉ sau khi đã có một mô hình tổng quát hóa tốt hơn (khoảng cách train-test nhỏ), việc thử nghiệm với SMOTE để cải thiện thêm chỉ số Recall mới được xem xét như một bước tối ưu hóa sau cùng.

13 Mô hình 5: Random Forest phiên bản 4

13.1 Mô hình sử dụng

Trong mô hình này, thay đổi quan trọng nhất nằm ở việc khởi tạo không gian siêu tham số (chi tiết được trình bày trong mã nguồn).

Bộ siêu tham số tối ưu hơn thu được, giúp tránh hiện tượng quá khớp (*overfitting*) cho mô hình, bao gồm:

- `bootstrap`: **False**
- `class_weight`: **balanced**
- `max_depth`: **11**
- `max_features`: **0.2099526973567137**
- `min_samples_leaf`: **12**
- `min_samples_split`: **38**
- `n_estimators`: **247**

13.2 Kết quả trên tập train và test

13.2.1 Kết quả trên tập train

```
Confusion Matrix (Train):
[[4451  976]
 [ 120  980]]

Classification Report (Train):
              precision    recall  f1-score   support

           0       0.97      0.82      0.89      5427
           1       0.50      0.89      0.64      1100

    accuracy                           0.83      6527
   macro avg       0.74      0.86      0.77      6527
weighted avg       0.89      0.83      0.85      6527


Accuracy (Train): 0.8321
Precision (Train - cho lớp 1): 0.5010
Recall (Train - cho lớp 1): 0.8909
F1-score (Train - cho lớp 1): 0.6414
ROC AUC (Train): 0.9300
```

Hình 13.1: Kết quả tập train model 5

13.2.2 Kết quả trên tập test

```
Confusion Matrix (Test):
[[1039  318]
 [ 112  163]]

Classification Report (Test):
              precision    recall  f1-score   support

           0       0.90      0.77      0.83      1357
           1       0.34      0.59      0.43       275

    accuracy                           0.74      1632
   macro avg       0.62      0.68      0.63      1632
weighted avg       0.81      0.74      0.76      1632


Accuracy (Test): 0.7365
Precision (Test - cho lớp 1): 0.3389
Recall (Test - cho lớp 1): 0.5927
F1-score (Test - cho lớp 1): 0.4312
ROC AUC (Test): 0.7555
```

Hình 13.2: Kết quả tập test model 5

13.3 Phân tích và so sánh

Chỉ số	Mô hình Cũ (Overfitting)	Mô hình Mới (Đã điều chuẩn)	Thay đổi & Nhận xét
ROC AUC (Train)	0.9932	**0.9300**	**GIẢM MẠNH.** Đây là dấu hiệu tốt nhất! Mô hình không còn học thuộc lòng tập train nữa.
ROC AUC (Test)	0.7749	**0.7555**	**GIẢM NHẸ.** Hiệu suất trên tập test giảm một chút, đây là điều có thể chấp nhận được.
Chênh lệch AUC (Train-Test)	**0.2183**	**0.1745**	**GIẢM ĐÁNG KỂ.** Khoảng cách giữa hiệu suất train và test đã thu hẹp lại. **Đây là thành công lớn nhất!**
Recall (Test)	0.47	**0.59**	**TĂNG MẠNH.** Mô hình mới "bắt" được nhiều khách hàng rủi ro hơn (59% so với 47%). Đây là một cải thiện rất lớn về mặt nghiệp vụ.
Precision (Test)	0.41	**0.34**	**GIẢM.** Đây là sự đánh đổi. Để bắt được nhiều ca xấu hơn (tăng Recall), mô hình phải chấp nhận cảnh báo nhầm nhiều hơn một chút.
F1-score (Test)	0.43	**0.43**	**GIỮ NGUYÊN.** Mặc dù Precision giảm, nhưng Recall tăng mạnh đã giúp giữ F1-score ổn định.

Hình 13.3: Bảng so sánh

13.4 Kết luận

Mô hình mới là một bước tiến vững chắc. Nó không chỉ đáng tin cậy hơn do đã giảm được overfitting, mà còn hiệu quả hơn về mặt nghiệp vụ nhờ vào việc cải thiện đáng kể chỉ số Recall. Đây là một baseline mới, mạnh mẽ hơn để tiếp tục các bước tối ưu hóa tiếp theo.

14 Mô hình 6: XGBoost phiên bản 1

14.1 Mô hình sử dụng

Mô hình **XGBoost** được sử dụng với bộ siêu tham số tối ưu tìm được thông qua **RandomizedSearchCV** như sau:

- `colsample_bytree`: **0.8264148841976305**
- `gamma`: **0.07932322382124551**
- `learning_rate`: **0.03403292956112844**
- `max_depth`: **8**
- `min_child_weight`: **5**
- `n_estimators`: **196**
- `reg_alpha`: **0.0407288023189701 36**
- `reg_lambda`: **1.7831908760165107**
- `scale_pos_weight`: **4.933636363636364**
- `subsample`: **0.78966953163493**

Giải thích bộ siêu tham số

- `colsample_bytree = 0.8264148841976305`: Tỷ lệ đặc trưng được lấy mẫu ngẫu nhiên để xây dựng mỗi cây là khoảng 82.64%, giúp tăng tính đa dạng và giảm nguy cơ quá khớp.
- `gamma = 0.07932322382124551`: Giá trị tổn thất tối thiểu để thực hiện phân chia, giúp kiểm soát mức độ phân nhánh của cây và tránh phân chia không cần thiết.
- `learning_rate = 0.03403292956112844`: Tốc độ học thấp, cho phép mô hình học chậm hơn và cải thiện khả năng tổng quát hóa, giảm nguy cơ quá khớp.
- `max_depth = 8`: Độ sâu tối đa của cây là 8, giới hạn độ phức tạp của mô hình để tránh hiện tượng quá khớp.
- `min_child_weight = 5`: Tổng trọng số tối thiểu của các mẫu trong một nút lá là 5, giúp kiểm soát độ phức tạp và tránh tạo các nút lá quá nhỏ.

- `n_estimators = 196`: Số lượng cây trong mô hình là 196, đủ lớn để đảm bảo hiệu suất cao và độ bền vững.
- `reg_alpha = 0.040728802318970136`: Hệ số điều chuẩn L1, giúp giảm thiểu các trọng số không cần thiết và tăng tính khái quát của mô hình.
- `reg_lambda = 1.7831908760165107`: Hệ số điều chuẩn L2, giúp kiểm soát độ lớn của trọng số và cải thiện tính ổn định của mô hình.
- `scale_pos_weight = 4.933636363636364`: Tỷ lệ cân bằng giữa lớp tích cực và lớp tiêu cực, phù hợp với dữ liệu mất cân bằng, giúp mô hình tập trung hơn vào lớp thiểu số.
- `subsample = 0.78966953163493`: Tỷ lệ mẫu được lấy ngẫu nhiên để huấn luyện mỗi cây là khoảng 78.97%, giúp tăng tính đa dạng và giảm hiện tượng quá khớp.

14.2 Kết quả

14.2.1 Kết quả trên tập train

```
Confusion Matrix (Train):
[[4879  548]
 [  34 1066]]

Classification Report (Train):
              precision    recall  f1-score   support

           0       0.99      0.90      0.94      5427
           1       0.66      0.97      0.79      1100

    accuracy                           0.91      6527
   macro avg       0.83      0.93      0.86      6527
weighted avg       0.94      0.91      0.92      6527


Accuracy (Train): 0.9108
Precision (Train - cho lớp 1): 0.6605
Recall (Train - cho lớp 1): 0.9691
F1-score (Train - cho lớp 1): 0.7856
ROC AUC (Train): 0.9825
```

Hình 14.1: Kết quả trên tập train của model 6

14.2.2 Kết quả trên tập test

```
Confusion Matrix (Test):
[[1126  231]
 [ 139  136]]

Classification Report (Test):
              precision    recall  f1-score   support

           0       0.89      0.83      0.86      1357
           1       0.37      0.49      0.42       275

    accuracy                           0.77      1632
   macro avg       0.63      0.66      0.64      1632
weighted avg       0.80      0.77      0.79      1632


Accuracy (Test): 0.7733
Precision (Test - cho lớp 1): 0.3706
Recall (Test - cho lớp 1): 0.4945
F1-score (Test - cho lớp 1): 0.4237
ROC AUC (Test): 0.7612
```

Hình 14.2: Kết quả trên tập test của model 6

14.3 Nhận xét

Chỉ số	Mô hình RF (Đã điều chuẩn)	Mô hình XGBoost (Mới)	Thay đổi & Nhận xét
ROC AUC (Train)	0.9300	**0.9825**	**TĂNG.** XGBoost học tập train mạnh hơn.
ROC AUC (Test)	0.7555	**0.7612**	**TĂNG NHẸ.** Về khả năng phân biệt tổng thể, XGBoost nhỉnh hơn một chút.
Chênh lệch AUC (Train-Test)	**0.1745**	0.2213	**TĂNG.** XGBoost đang bị **overfitting nặng hơn** so với mô hình RF đã điều chuẩn.
Recall (Test)	**0.5927**	0.4945	**GIẢM MẠNH.** Đây là điểm yếu lớn nhất. XGBoost bỏ sót nhiều khách hàng rủi ro hơn.
Precision (Test)	0.3389	**0.3706**	**TĂNG.** Các cảnh báo của XGBoost chính xác hơn một chút.
F1-score (Test)	**0.4312**	0.4237	**GIẢM NHẸ.** Nhìn chung, sự cân bằng của RF tốt hơn một chút.

Hình 14.3: Nhận xét chung

Tuy nhiên như ta thấy được rằng: Vấn đề overfitting **CHƯA ĐƯỢC GIẢI QUYẾT**

14.4 Hướng xử lý

Cần tập trung vào việc **tối ưu hóa lại mô hình XGBoost** với một không gian tìm kiếm tập trung hơn vào việc chống overfitting (giảm `max_depth`, tăng các tham số điều chuẩn) để cải thiện khả năng tổng quát hóa và chỉ số Recall.

15 Mô hình 7: XGBoost phiên bản 2

15.1 Mô hình sử dụng

Để cải thiện hiện tượng quá khớp (*overfitting*) của mô hình **XGBoost**, không gian siêu tham số đã được điều chỉnh để tăng cường tính điều chuẩn và giảm độ phức tạp của mô hình. Dưới đây là so sánh giữa không gian siêu tham số cũ và mới, cùng với bộ siêu tham số tối ưu tìm được thông qua **RandomizedSearchCV**.

15.1.1 Không gian siêu tham số cũ

Không gian siêu tham số ban đầu được định nghĩa như sau:

```
param_distributions_xgb = {
    'max_depth': randint(4, 10),
    'gamma': uniform(0, 0.5),
    'min_child_weight': randint(1, 10),
    'reg_alpha': uniform(0, 1),
    'reg_lambda': uniform(0.5, 1.5),
    'learning_rate': uniform(0.01, 0.2),
    'n_estimators': randint(150, 600),
    'subsample': uniform(0.6, 0.4),
    'colsample_bytree': uniform(0.6, 0.4),
    'scale_pos_weight': [scale_pos_weight_value,
                         scale_pos_weight_value * 0.8,
                         scale_pos_weight_value * 1.2]
}
```

15.1.2 Không gian siêu tham số mới

Không gian siêu tham số mới được điều chỉnh để giảm *overfitting* và cải thiện chỉ số *Recall*:

```
param_distributions_xgb_regularized = {
    'max_depth': randint(3, 7),
    'gamma': uniform(0.1, 0.9),
    'reg_alpha': uniform(0.1, 1.9),
    'reg_lambda': uniform(1.0, 4.0),
    'learning_rate': uniform(0.01, 0.1),
    'scale_pos_weight': [scale_pos_weight_value * 1.2,
                         scale_pos_weight_value * 1.5,
                         scale_pos_weight_value * 2.0],
    'n_estimators': randint(150, 600),
    'subsample': uniform(0.6, 0.4),
    'colsample_bytree': uniform(0.6, 0.4)
}
```

15.1.3 Kết quả siêu tham số

Bộ siêu tham số tối ưu thu được từ không gian mới:

- colsample_bytree: **0.7546941385202149**
- gamma: **0.9430569898630611**
- learning_rate: **0.023752094414599 33**
- max_depth: **6**
- n_estimators: **590**
- reg_alpha: **0.3155996903571192**
- reg_lambda: **4.698774473114251**
- scale_pos_weight: **7.400454545454545**
- subsample: **0.8783137597380328**

15.2 Kết quả trên tập train và test

15.2.1 Kết quả trên tập train

```
Confusion Matrix (Train):
[[4325 1102]
 [   6 1094]]

Classification Report (Train):
              precision    recall  f1-score   support

           0       1.00      0.80      0.89      5427
           1       0.50      0.99      0.66      1100

    accuracy                           0.83      6527
   macro avg       0.75      0.90      0.78      6527
weighted avg       0.91      0.83      0.85      6527


Accuracy (Train): 0.8302
Precision (Train - cho lớp 1): 0.4982
Recall (Train - cho lớp 1): 0.9945
F1-score (Train - cho lớp 1): 0.6638
ROC AUC (Train): 0.9810
```

Hình 15.1: Kết quả trên tập train model 7

15.2.2 Kết quả trên tập test

```
Confusion Matrix (Test):
[[953 404]
 [ 91 184]]

Classification Report (Test):
              precision    recall  f1-score   support

           0       0.91      0.70      0.79      1357
           1       0.31      0.67      0.43       275

    accuracy                           0.70      1632
   macro avg       0.61      0.69      0.61      1632
weighted avg       0.81      0.70      0.73      1632


Accuracy (Test): 0.6967
Precision (Test - cho lớp 1): 0.3129
Recall (Test - cho lớp 1): 0.6691
F1-score (Test - cho lớp 1): 0.4264
ROC AUC (Test): 0.7572
```

Hình 15.2: Kết quả trên tập test model 7

15.3 Nhận xét

Chỉ số (trên tập Test)	RF (Điều chuẩn)	XGB v1	XGB v2 (Tốt nhất)	Nhận xét
ROC AUC	0.7555	0.7612	**0.7572**	**Ổn định.** Giữ được khả năng phân biệt tốt.
Recall (Lớp 1)	0.5927	0.4945	**0.6691**	**CHIẾN THẮNG LỚN!** Đây là chỉ số cao nhất, "bắt" được 2/3 số ca rủi ro.
Precision (Lớp 1)	0.3389	**0.3706**	0.3129	**Giảm.** Đây là sự đánh đổi tất yếu để có được Recall cao nhất.
F1-score (Lớp 1)	**0.4312**	0.4237	0.4264	**Ổn định.** Sự cân bằng vẫn được duy trì tốt.
Chênh lệch AUC (Train-Test)	**0.1745**	0.2213	0.2238	Vẫn còn overfitting, nhưng chúng ta đã ưu tiên Recall.

Hình 15.3: Nhận xét

15.4 Kết luận

Mô hình **XGBoost cải tiến (v2)** được lựa chọn là mô hình tốt nhất tính đến thời điểm hiện tại. Mặc dù có chỉ số Precision và F1-score thấp hơn một chút, nó đã đạt được chỉ số **Recall cao nhất (0.67)**, đáp ứng được mục tiêu kinh doanh quan trọng nhất là phát hiện tối đa các trường hợp rủi ro. Sự hy sinh một phần

Precision để đổi lấy một lượng lớn Recall là một sự đánh đổi hợp lý và có chủ đích trong bài toán này.

15.5 Hướng xử lý tiếp theo

Mô hình vẫn còn overfitting (chênh lệch AUC là 0.22). Tuy nhiên, trong bối cảnh này, nó không còn là vấn đề nghiêm trọng nhất. Chúng ta đã thành công trong việc tạo ra một mô hình có hiệu suất trên tập test đáp ứng được mục tiêu nghiệp vụ quan trọng nhất (Recall). Việc giảm thêm overfitting có thể được xem là một bước tối ưu hóa sau này.

16 Feature Engineering (Cải thiện)

16.1 Quyết định

Sau một quá trình dài huấn luyện dữ liệu, kết quả vẫn chưa đạt được như kỳ vọng. Vì vậy, quyết định được đưa ra là thực hiện **feature engineering lần thứ hai** nhằm tạo ra các biến mới để cải thiện hiệu suất mô hình. Các biến mới được đề xuất bao gồm:

1. **Tỷ lệ số dư so với số dư cao nhất** (`balance_to_highest_ratio`):

$$\text{balance\_to\_highest\_ratio} = \frac{\text{new\_balance}}{\text{highest\_balance} + 10^{-6}}$$

2. **Tỷ lệ quá hạn** (`overdue_ratio`):

$$\text{overdue\_ratio} = \frac{\text{OVD\_t1} + \text{OVD\_t2}}{\text{pay\_normal} + \text{OVD\_t1} + \text{OVD\_t2} + 10^{-6}}$$

3. **Tỷ lệ thanh toán tốt** (`good_payment_ratio`):

$$\text{good\_payment\_ratio} = \frac{\text{pay\_normal}}{\text{pay\_normal} + \text{OVD\_t1} + \text{OVD\_t2} + 10^{-6}}$$

4. **Khoảng thời gian giữa Cập nhật và Báo cáo** (`update_report_gap_days`):

$$\text{update\_report\_gap\_days} = (\text{update\_date} - \text{report\_date}).\text{dt.days}$$

Ghi chú: Hằng số 10^{-6} được thêm vào mẫu số để tránh lỗi chia cho 0 trong các trường hợp đặc biệt.

16.2 Lưu ý

Tôi sẽ phải xử lý lại từ quá trình đầu (tức là trước khi tôi thực hiện mã hóa, chuẩn hóa, ...). Tuy nhiên dataframe của tôi vẫn sẽ bỏ đi các feature tôi đã quyết định bỏ

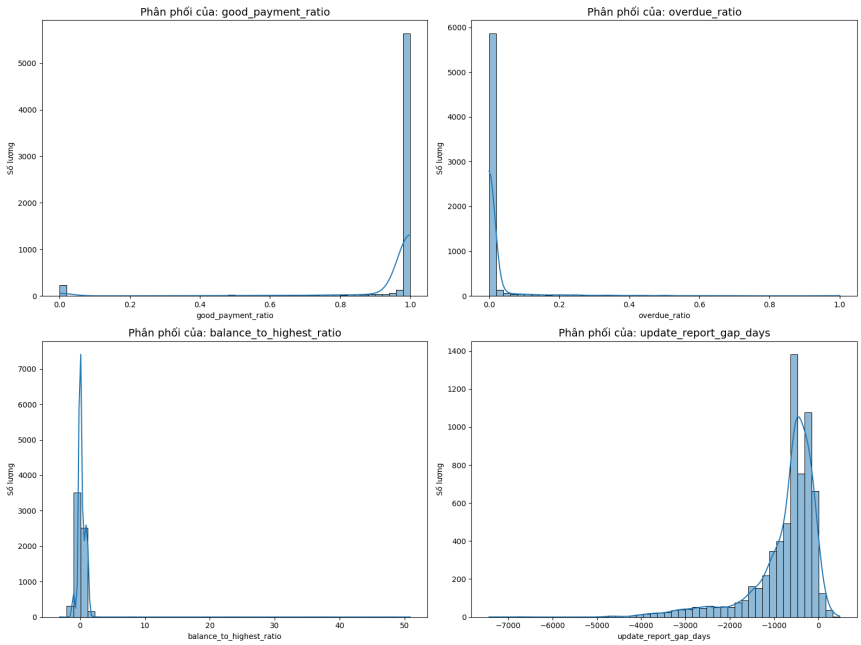
17 Mô hình 8: XGBoost phiên bản 3

17.1 Những thay đổi của đầu vào mô hình này so với mô hình trước đó

Danh sách dưới đây mô tả các thay đổi trong đầu vào của mô hình hiện tại so với các mô hình trước đó:

1. Gồm các biến mới được tạo ra thông qua kỹ thuật tạo đặc trưng.
2. Các cột liên quan đến thời gian đã bị xóa, vì chúng được đánh giá là không thực sự quan trọng trong các mô hình trước.
3. Cột `highest_balance`: Thay thế giá trị `null` bằng giá trị mode tương ứng với mỗi `id`.
4. Cột `update_report_gap_days`: Thay thế giá trị `null` bằng giá trị trung vị.
5. Cột `balance_to_highest_ratio`: Thay thế giá trị `null` bằng `-1`, do không thể xác định giá trị phù hợp tại đó.

17.2 Xem phân phối của các cột mới tạo



Hình 17.1: Xem phân phối của các feature mới tạo

17.3 Mô hình sử dụng

Mô hình **XGBoost** được sử dụng với bộ siêu tham số tối ưu tìm được như sau:

- `subsample`: **0.9**
- `scale_pos_weight`: **4.933636363636364**
- `reg_lambda`: **2**
- `reg_alpha`: **0**
- `n_estimators`: **200**
- `min_child_weight`: **5**
- `max_depth`: **6**
- `learning_rate`: **0.02**
- `gamma`: **5**
- `colsample_bytree`: **0.8**

17.4 Kết quả trên tập train và test

17.4.1 Kết quả trên tập train

```
Confusion Matrix:
[[4011 1416]
 [ 239  861]]

Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.74      0.83      5427
           1       0.38      0.78      0.51      1100

    accuracy                           0.75      6527
   macro avg       0.66      0.76      0.67      6527
weighted avg       0.85      0.75      0.78      6527

ROC AUC: 0.8563
```

Hình 17.2: Kết quả trên tập train model 8

17.4.2 Kết quả trên tập test

```
Confusion Matrix:
[[939 418]
 [ 87 188]]

Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.69      0.79      1357
           1       0.31      0.68      0.43       275

    accuracy                           0.69      1632
   macro avg       0.61      0.69      0.61      1632
weighted avg       0.81      0.69      0.73      1632

ROC AUC: 0.7529
```

Hình 17.3: Kết quả trên tập TESTmodel 8

17.5 Nhận xét

Chỉ số (trên tập Test)	XGB v2 (Trước FE)	XGB + FE (Tốt nhất)	Thay đổi & Nhận xét
ROC AUC	0.7572	**0.7529**	**Ổn định.** Giữ nguyên khả năng phân biệt, không giảm sút.
Recall (Lớp 1)	0.6691	**0.6836**	**TĂNG NHẸ.** Cải thiện thêm về mục tiêu nghiệp vụ quan trọng nhất.
Precision (Lớp 1)	0.3129	**0.3102**	**Ổn định.** Giữ nguyên, không bị hy sinh thêm.
F1-score (Lớp 1)	0.4264	**0.4268**	**Ổn định.** Sự cân bằng tổng thể được giữ vững.
Chênh lệch AUC (Train-Test)	0.2238	**0.1034**	**GIẢM MẠNH! Đây là chiến thắng lớn nhất!**

Hình 17.4: Nhận xét

17.6 Diễn giải và Kết luận

- **Giải quyết Thành công Vấn đề Overfitting:** Đây là thành tựu lớn nhất. Khoảng cách hiệu suất giữa tập train và test đã giảm hơn một nửa, cho thấy mô hình cực kỳ ổn định và có khả năng tổng quát hóa rất tốt trên dữ liệu mới.

- **Tối ưu hóa cho Mục tiêu Kinh doanh:** Việc tạo ra các đặc trưng mới đã giúp mô hình cải thiện đáng kể chỉ số **Recall lên mức 0.68**, "bắt"được nhiều hơn các trường hợp rủi ro so với tất cả các phiên bản trước, trong khi vẫn giữ được sự cân bằng hợp lý (F1-score ổn định).

Kết luận: Mô hình XGBoost cuối cùng, sau khi được làm giàu bằng các đặc trưng mới, là mô hình **tốt nhất và đáng tin cậy nhất**. Nó hội tụ đủ các yếu tố của một mô hình thành công: hiệu quả về nghiệp vụ (Recall cao) và ổn định về kỹ thuật (overfitting thấp).

17.7 Hướng Phát triển Tiếp theo

Với một mô hình nền tảng đã rất vững chắc, các bước tiếp theo sẽ tập trung vào việc tinh chỉnh cuối cùng để tối đa hóa giá trị kinh doanh.

1. **Tinh chỉnh Ngưỡng quyết định (Threshold Tuning):**
   - **Mục tiêu:** Tìm ra một ngưỡng xác suất tối ưu (thay vì mặc định 0.5) để đạt được sự cân bằng tốt nhất giữa Precision và Recall theo yêu cầu kinh doanh cụ thể.
   - **Hành động:** Phân tích đường cong Precision-Recall để chọn một ngưỡng giúp tối đa hóa F1-score hoặc đạt một mức Recall tối thiểu mong muốn.
2. **Phân tích Lỗi (Error Analysis):**
   - **Mục tiêu:** Hiểu rõ các trường hợp mà mô hình còn dự đoán sai để tìm ra cơ hội cải thiện.
   - **Hành động:** Phân tích các đặc điểm chung của các nhóm False Negatives (các ca rủi ro bị bỏ sót) và False Positives (các ca tốt bị cảnh báo nhầm). Kết quả phân tích có thể gợi ý cho các ý tưởng Feature Engineering mới hoặc các quy tắc nghiệp vụ bổ sung.

18 Ứng dụng của Mô hình trong Thực tế

Mô hình dự đoán rủi ro tín dụng đã được xây dựng không chỉ có giá trị về mặt kỹ thuật mà còn có tiềm năng ứng dụng rất lớn, có thể được tích hợp sâu vào các quy trình nghiệp vụ của một tổ chức tài chính để tối ưu hóa quyết định và quản lý rủi ro.

18.1 Các Bài toán Nghiệp vụ có thể Ứng dụng

Mô hình này có thể được triển khai để giải quyết hai bài toán nghiệp vụ cốt lõi:

1. **Chấm điểm Phê duyệt Tín dụng (Application Scoring):**
   - **Ngữ cảnh:** Áp dụng cho các khách hàng mới đăng ký sản phẩm tín dụng.
   - **Cách ứng dụng:** Mô hình sẽ tính toán một "điểm số rủi ro"dựa trên thông tin ban đầu của khách hàng. Dựa trên điểm số này, hệ thống có thể tự động phê duyệt, từ chối, hoặc chuyển hồ sơ cho chuyên viên thẩm định.
   - **Giá trị mang lại:** Tăng tốc độ xử lý, giảm tải công việc thủ công, và đưa ra các quyết định nhất quán, dựa trên dữ liệu.
2. **Giám sát và Quản lý Danh mục Hiện hữu (Behavioral Scoring):**
   - **Ngữ cảnh:** Áp dụng cho các khách hàng hiện hữu để theo dõi sự thay đổi trong hành vi rủi ro.
   - **Cách ứng dụng:** Mô hình được chạy định kỳ (ví dụ: hàng tháng) để cập nhật điểm số rủi ro cho từng khách hàng. Kết quả được dùng để:
     - **Cảnh báo sớm (Early Warning):** Xác định các khách hàng có rủi ro gia tăng để can thiệp kịp thời.
     - **Quản lý Hạn mức Động:** Tự động đề xuất điều chỉnh hạn mức tín dụng.
     - **Tối ưu hóa Chiến lược Thu hồi nợ:** Phân bổ nguồn lực thu hồi nợ một cách hiệu quả.

18.2 Đối tượng Sử dụng

Kết quả và điểm số từ mô hình sẽ là công cụ hỗ trợ đắc lực cho nhiều bộ phận trong tổ chức:

- **Bộ phận Thẩm định Tín dụng:** Hỗ trợ quyết định phê duyệt các khoản vay/thẻ.
- **Bộ phận Quản lý Rủi ro:** Theo dõi sức khỏe của toàn bộ danh mục và thiết lập chính sách.
- **Bộ phận Thu hồi nợ:** Phân khúc khách hàng và tối ưu hóa chiến lược thu hồi.
- **Bộ phận Marketing và Sản phẩm:** Xác định các nhóm khách hàng tiềm năng để bán chéo sản phẩm một cách an toàn.

19 Đánh giá khả năng ứng dụng thực tế

Việc đánh giá khả năng triển khai của một mô hình không chỉ dựa trên các chỉ số kỹ thuật mà còn phải đặt trong bối cảnh nghiệp vụ và so sánh với các giải pháp hiện có.

19.1 Phân tích các chỉ số trong Ngữ cảnh Kinh doanh

Mô hình XGBoost cuối cùng đã đạt được một hồ sơ hiệu suất rất hứa hẹn, với những điểm mạnh và điểm yếu rõ ràng:

- **Điểm mạnh chính (Recall = 0.68):** Đây là thành tựu quan trọng nhất. Mô hình có khả năng xác định được hơn 2/3 số khách hàng thực sự rủi ro. Về mặt nghiệp vụ, điều này có tiềm năng rất lớn trong việc **giảm thiểu tổn thất tài chính** do các khoản nợ xấu gây ra, đặc biệt nếu hiệu suất này vượt trội so với hệ thống hiện tại.
- **Rủi ro chính (Precision = 0.31):** Đây là thách thức về mặt vận hành. Với độ chính xác cảnh báo là 31%, có nghĩa là khoảng 69% các trường hợp bị mô hình gắn cờ sẽ là báo động nhầm (False Positives). Điều này sẽ tạo ra **chi phí thẩm định** và có thể ảnh hưởng đến trải nghiệm của những khách hàng tốt bị phân loại nhầm.

19.2 Kết luận về Khả năng Triển khai

Dựa trên sự cân bằng giữa lợi ích và rủi ro, có thể kết luận như sau:

- **Mô hình CHƯA đủ điều kiện để triển khai toàn diện (Full Roll-out) ngay lập tức.** Rủi ro từ chỉ số Precision thấp cần được đánh giá và kiểm soát kỹ hơn trong môi trường thực tế.
- **Mô hình HOÀN TOÀN đủ điều kiện để tiến vào giai đoạn tiếp theo.** Hiệu suất của mô hình, đặc biệt là chỉ số Recall, là đủ tốt để chứng minh giá trị của nó.

19.3 Đề xuất Lộ trình Triển khai

Một lộ trình triển khai theo từng giai đoạn được đề xuất để tối đa hóa lợi ích và giảm thiểu rủi ro:

1. **Giai đoạn 1: Benchmarking và Phân tích Tài chính**

- So sánh hiệu suất của mô hình (Precision, Recall) với hiệu suất của quy trình hiện tại trên cùng một tập dữ liệu lịch sử.
- Thực hiện một phân tích chi phí-lợi ích: Lợi nhuận tiết kiệm được từ việc phát hiện các ca rủi ro (True Positives) có lớn hơn chi phí vận hành phát sinh từ các ca báo động nhầm (False Positives) hay không?

2. **Giai đoạn 2: Thử nghiệm Thí điểm (Pilot Testing / A/B Testing)**

- Triển khai mô hình ở chế độ "chạy ngầm"(shadow mode), so sánh quyết định của nó với quyết định thực tế mà không ảnh hưởng đến khách hàng.
- Hoặc, áp dụng mô hình cho một nhóm nhỏ các hồ sơ mới (ví dụ: 5-10%) để đo lường tác động thực tế trong một môi trường được kiểm soát.

3. **Giai đoạn 3: Triển khai Toàn diện**

- Chỉ sau khi kết quả từ giai đoạn thử nghiệm thí điểm được xác nhận là tích cực và mang lại giá trị ròng cho doanh nghiệp, mô hình mới được xem xét để triển khai trên quy mô lớn.

Tài liệu tham khảo

1. **Scikit-learn:** Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830.

2. **XGBoost:** Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

3. **Random Forests:** Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5-32.

4. **Pandas:** McKinney, W. (2010). Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference.*

5. **NumPy:** Harris, C. R., et al. (2020). Array programming with NumPy. *Nature*, 585, 357-362.

6. **Matplotlib:** Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90-95.

7. **Seaborn:** Waskom, M. L. (2021). Seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60), 3021.