

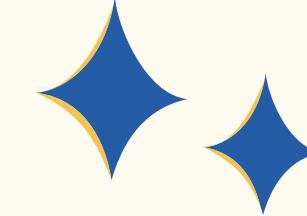
# DỰ ĐOÁN RỦI RO VỠ NỢ CỦA THẺ TÍN DỤNG

Nhóm 5



# NỘI DUNG

- 1** Giới thiệu đề tài và bộ dữ liệu
- 2** Xử lý dữ liệu trước phân tích và Phân tích khám phá dữ liệu (EDA)
- 3** Xây dựng mô hình
- 4** Kết luận



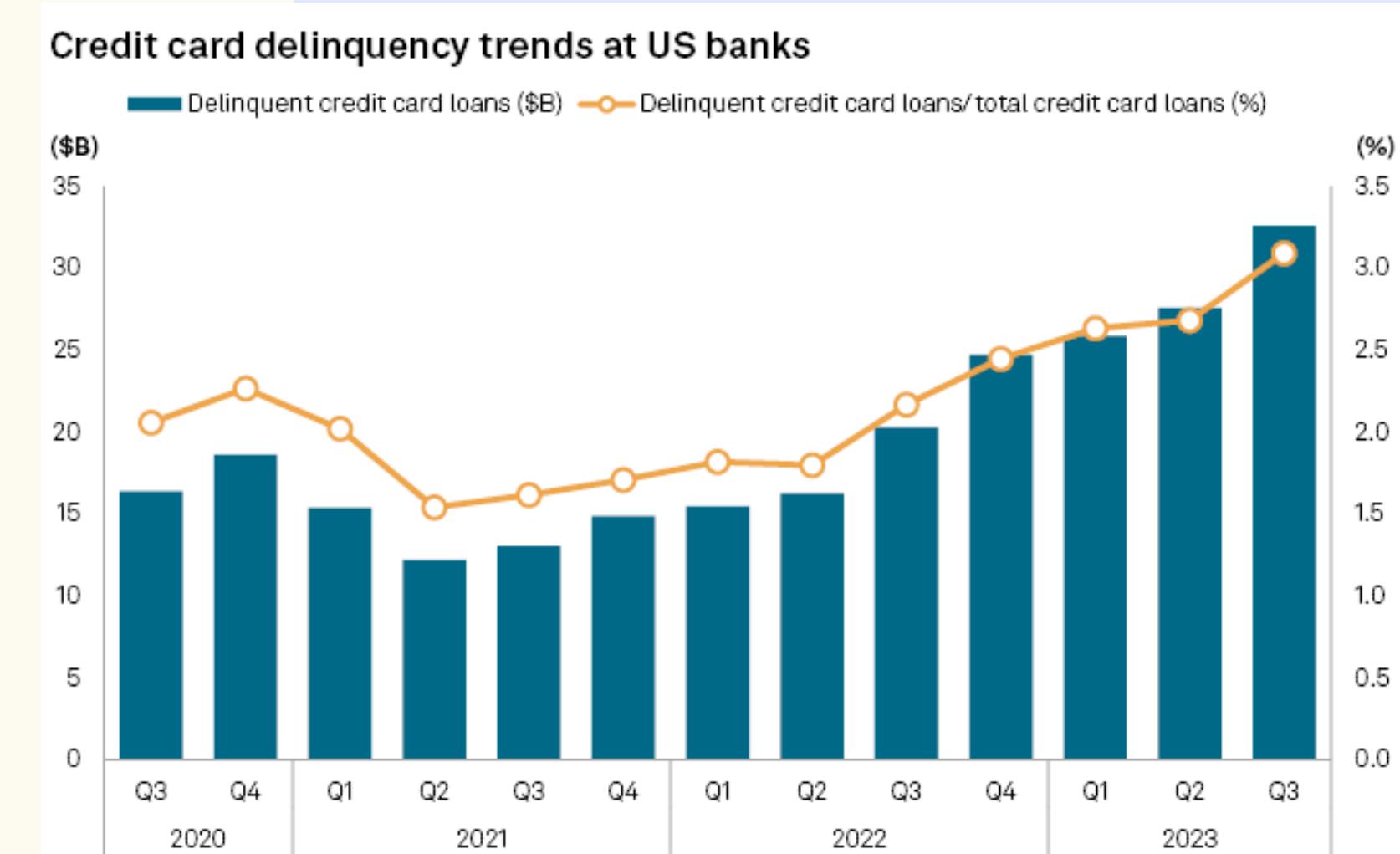
# LÝ DO CHỌN ĐỀ TÀI

## Tính cấp thiết trong lĩnh vực tài chính:

- Vỡ nợ là một trong những vấn đề quan trọng ảnh hưởng trực tiếp đến hệ thống tài chính, các tổ chức tín dụng, và nền kinh tế.

## Giảm thiểu rủi ro cho các tổ chức tài chính:

- Các tổ chức tài chính sử dụng xác suất vỡ nợ (PD) để sàng lọc người vay tiềm năng, đánh giá các điều khoản của các khoản vay mới và quản lý rủi ro xuất phát từ hoạt động cho vay.



# MỤC TIÊU

- Đánh giá hiệu quả của mô hình phát hiện tình trạng vỡ nợ của khách hàng.
- Tập trung vào các **phương pháp tiền xử lý** và đánh giá các mô hình liên quan thông qua việc sử dụng các **thuật toán học máy có giám sát** khác nhau nhằm tăng hiệu năng và tính tin cậy của kết quả mô hình.
- **So sánh** các phương pháp đã thực hiện



# TỔNG QUAN DỮ LIỆU



Bộ dữ liệu này chứa thông tin về khách hàng sử dụng thẻ tín dụng ở Đài Loan từ tháng 4 năm 2005 đến tháng 9 năm 2005.



## Thông tin tổng quan

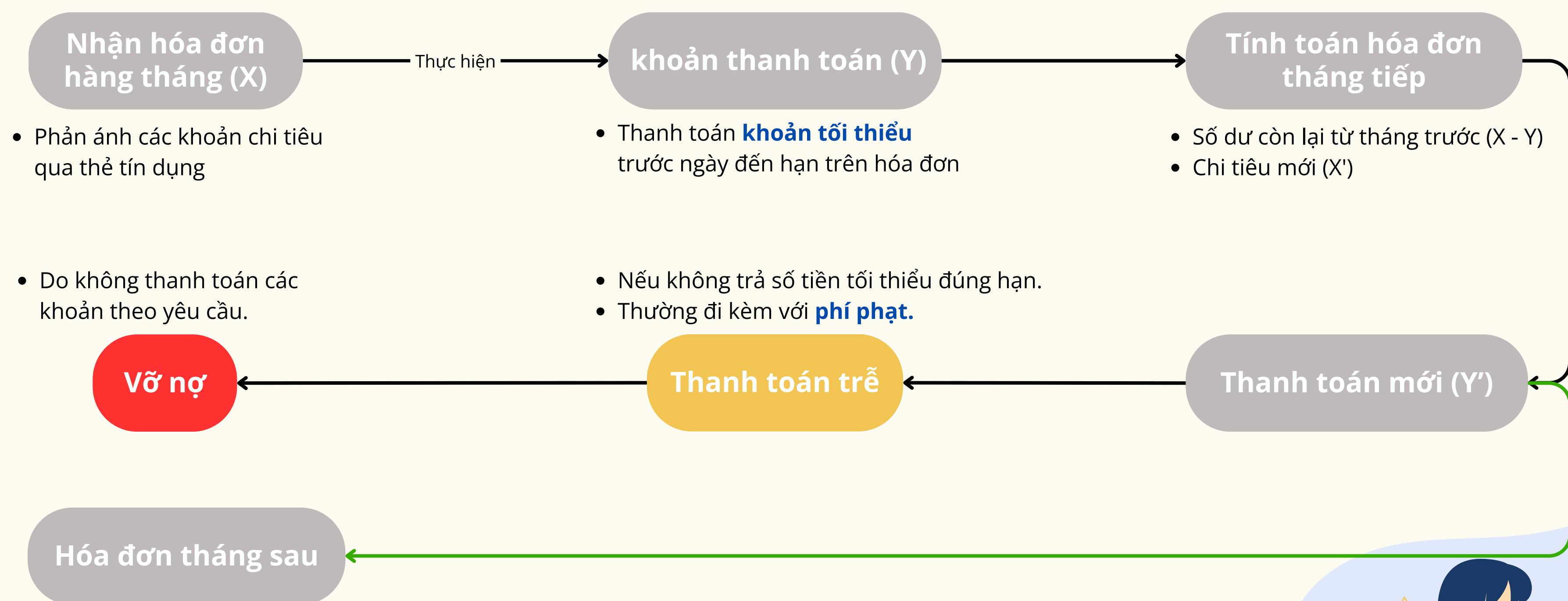
- Dữ liệu gồm 30000 dòng và 25 cột
- Các khoản thanh toán không trả được
- Các yếu tố nhân khẩu học
- Dữ liệu tín dụng
- Lịch sử thanh toán và sao kê hóa đơn của khách hàng



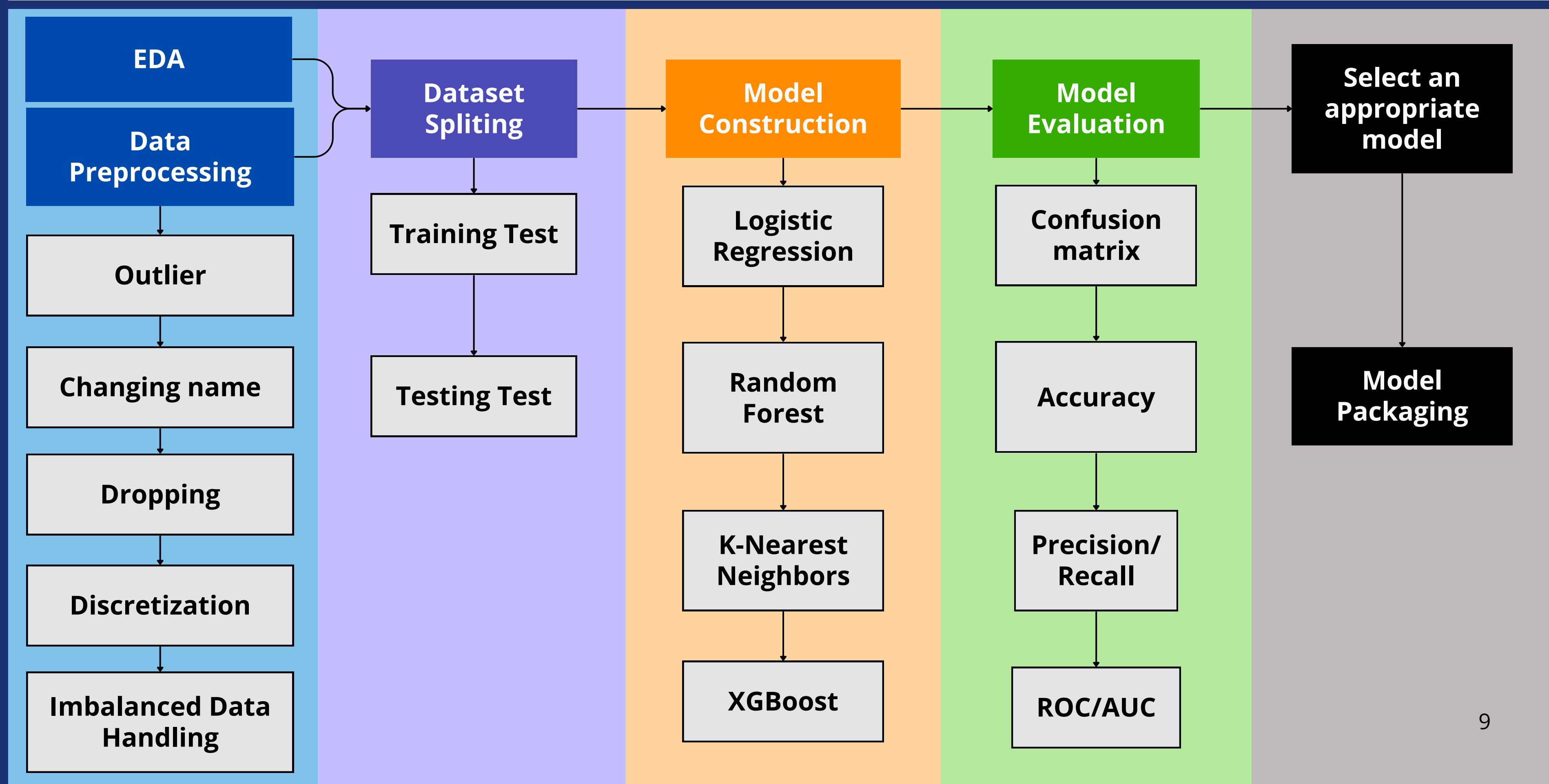
Tên trường	Giá trị	Mô Tả
ID		ID của từng khách hàng
LIMIT_BAL		Số tiền tín dụng được cấp
SEX	1 = male 2 = female	Giới tính
EDUCATION	1 = graduate school 2 = university 3 = high school 4 = others 5 = unknown 6 = unknown	Trình độ học vấn
MARRIAGE	1 = married 2 = single 3=others	Tình trạng hôn nhân

Tên trường	Giá trị	Mô Tả
AGE		Tuổi của khách hàng
PAY_0, 2, 3, 4, 5, 6	<p>-1 = pay duly            1 = payment delay for one month            2 = payment delay for two months            8 = payment delay for eight months            9 = payment delay for nine months and above</p>	Tình trạng thanh toán từ tháng 9 đến tháng 4 năm 2005
BILL_AMT1, 2, 3, 4, 5, 6		Số tiền trên hóa đơn từ tháng 9 đến tháng 4 năm 2005
PAY_AMT1, 2, 3, 4, 5, 6		Số tiền thanh toán trước từ tháng 9 đến tháng 4 năm 2005
default.payment.next.month	<p>1 = yes            0 = no</p>	Dự đoán khả năng thanh toán vào tháng tới

# Hệ thống thẻ tín dụng hoạt động như thế nào?



# WorkFlow



# Data Preprocessing & EDA



Outlier  
Changing name

Dropping  
Discretization

Imbalanced Data  
Handling

# DATA PREPROCESSING



- Đổi PAY\_0 thành PAY\_1 để đồng bộ cách đặt tên
- Rút gọn cột “default.payment.next.month” thành “default”

Trước	Sau
PAY_0	PAY_1
default.payment.next.month	default

- Loại bỏ cột ID

```
df = df.drop(columns = 'ID')
```

# DISCRETIZATION

- Phân độ tuổi thành các khoảng:

18 - 34	35 - 44	45 - 64	> 64
---------	---------	---------	------

- Phân giới hạn tín dụng thành các khoảng:

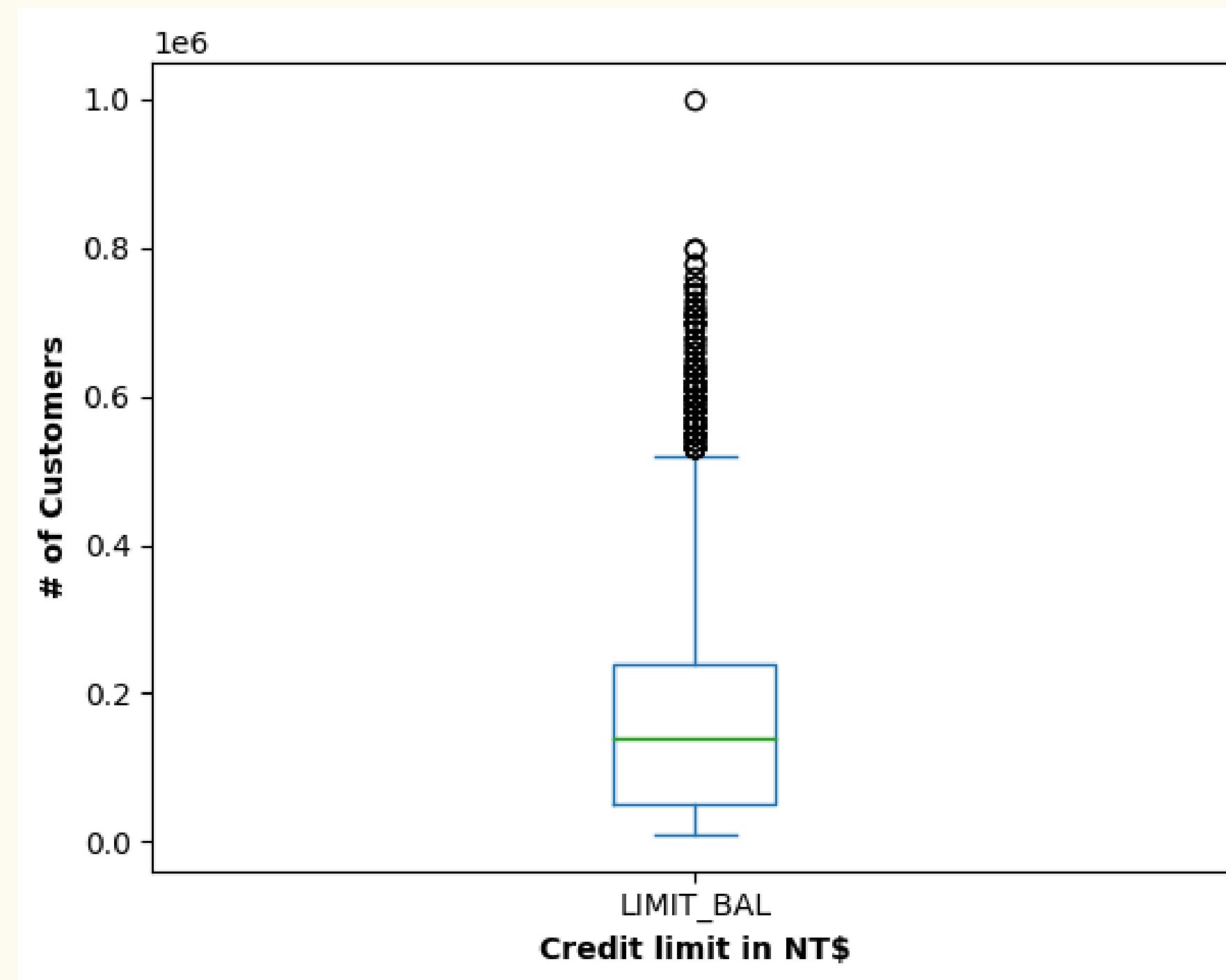
```
['(5000, 50000]' , '(50000, 100000]' , '(100000, 150000]' , '(150000, 200000]' ,  
'(200000, 300000]' , '(300000, 400000]' , '(400000, 500000]' , '(500000, 1100000]'
```

→ Thuận tiện cho việc so sánh giữa các phân khúc khách hàng khác nhau.

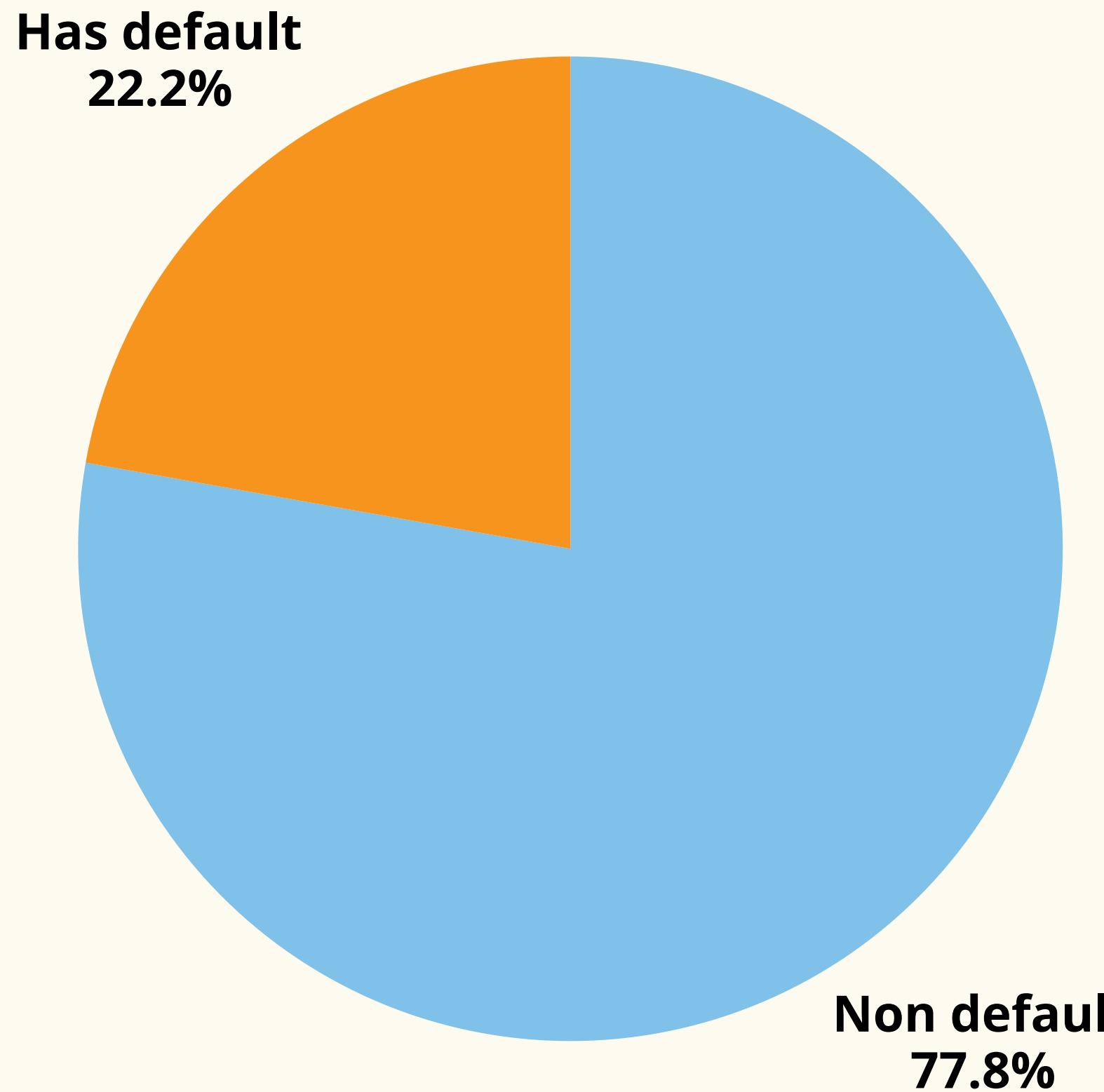


# OUTLIER

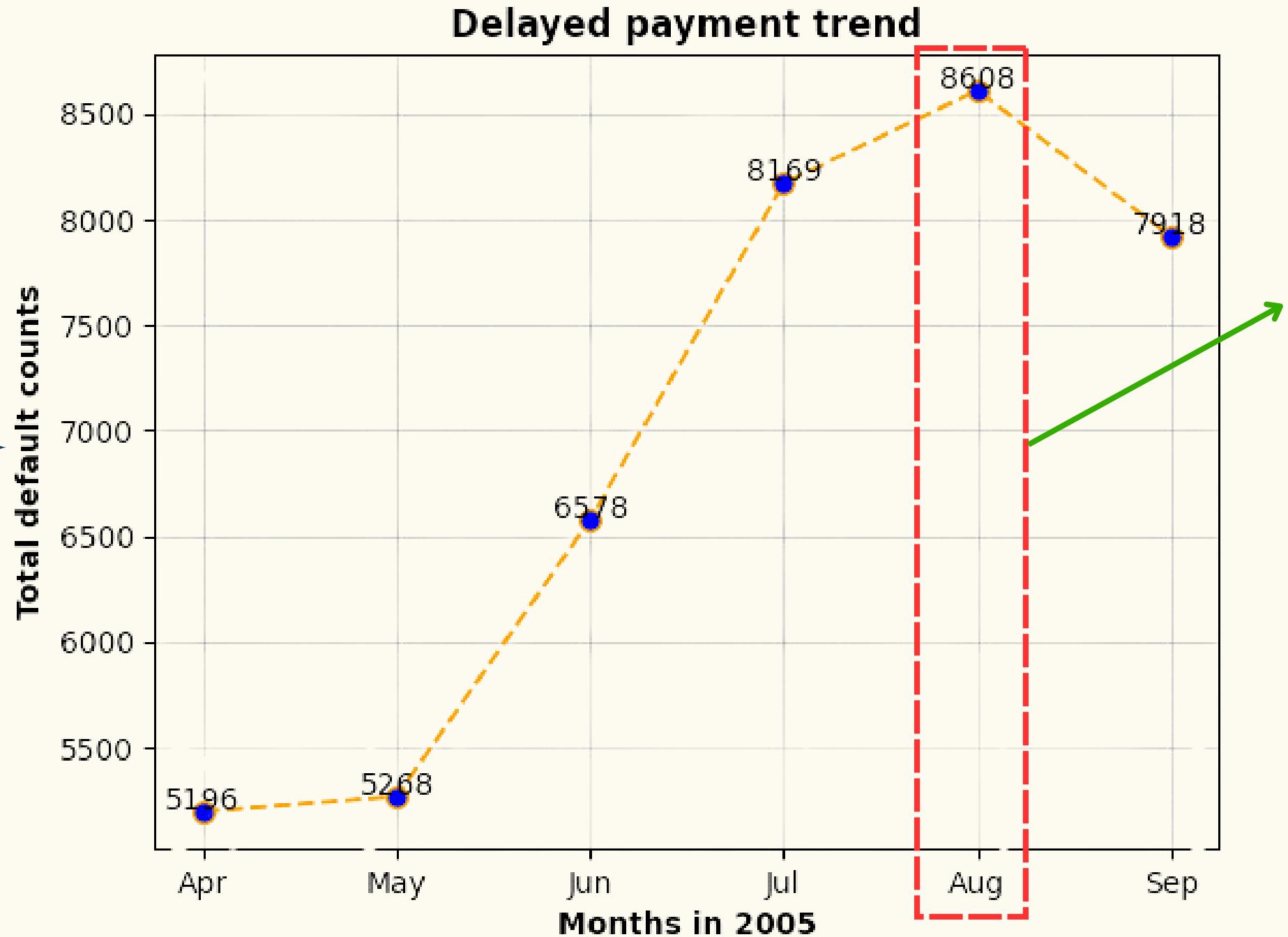
Dữ liệu bị outlier tại **LIMIT\_BAL**



# PHÂN PHỐI TỶ LỆ vỡ NỢ

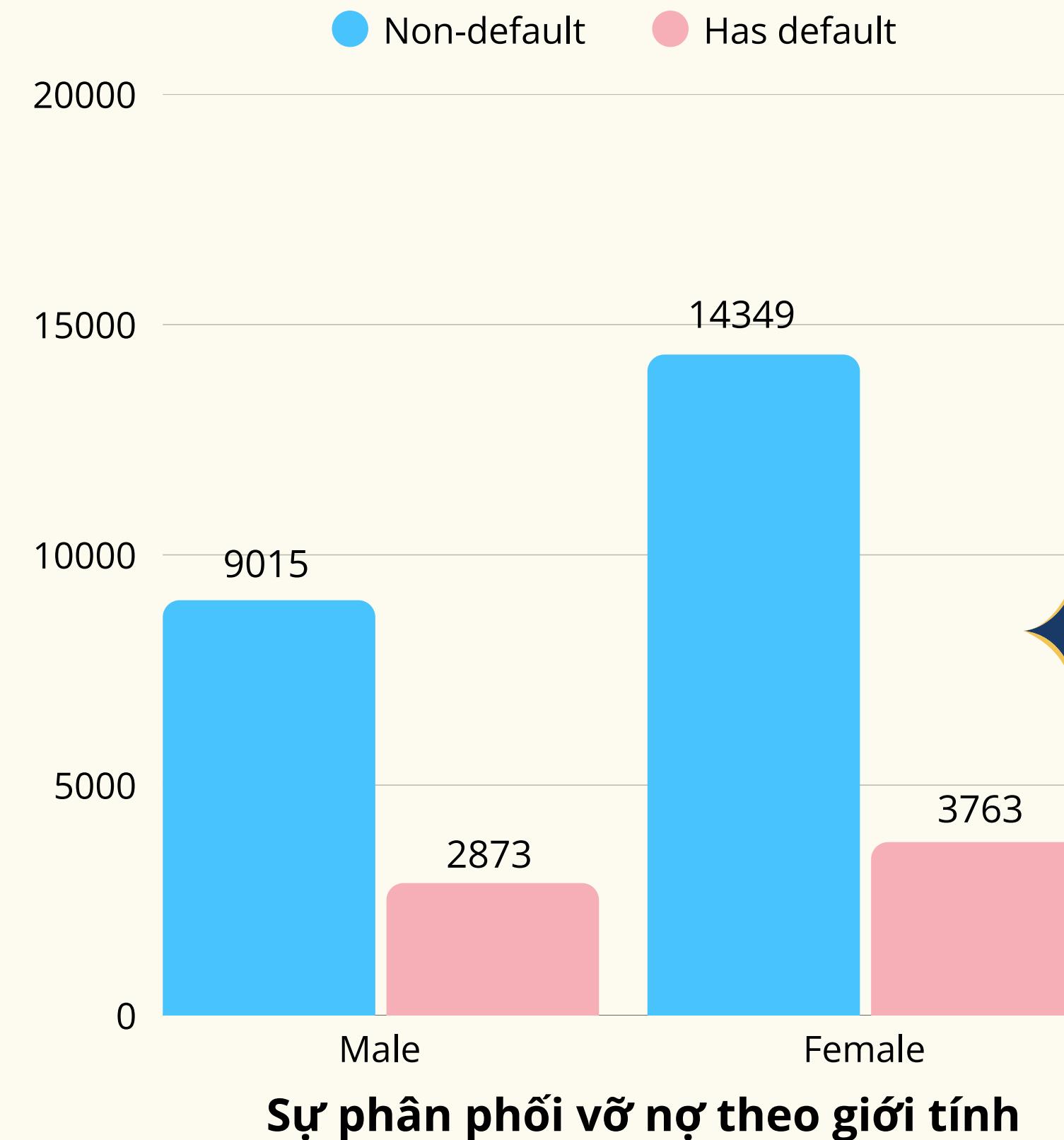
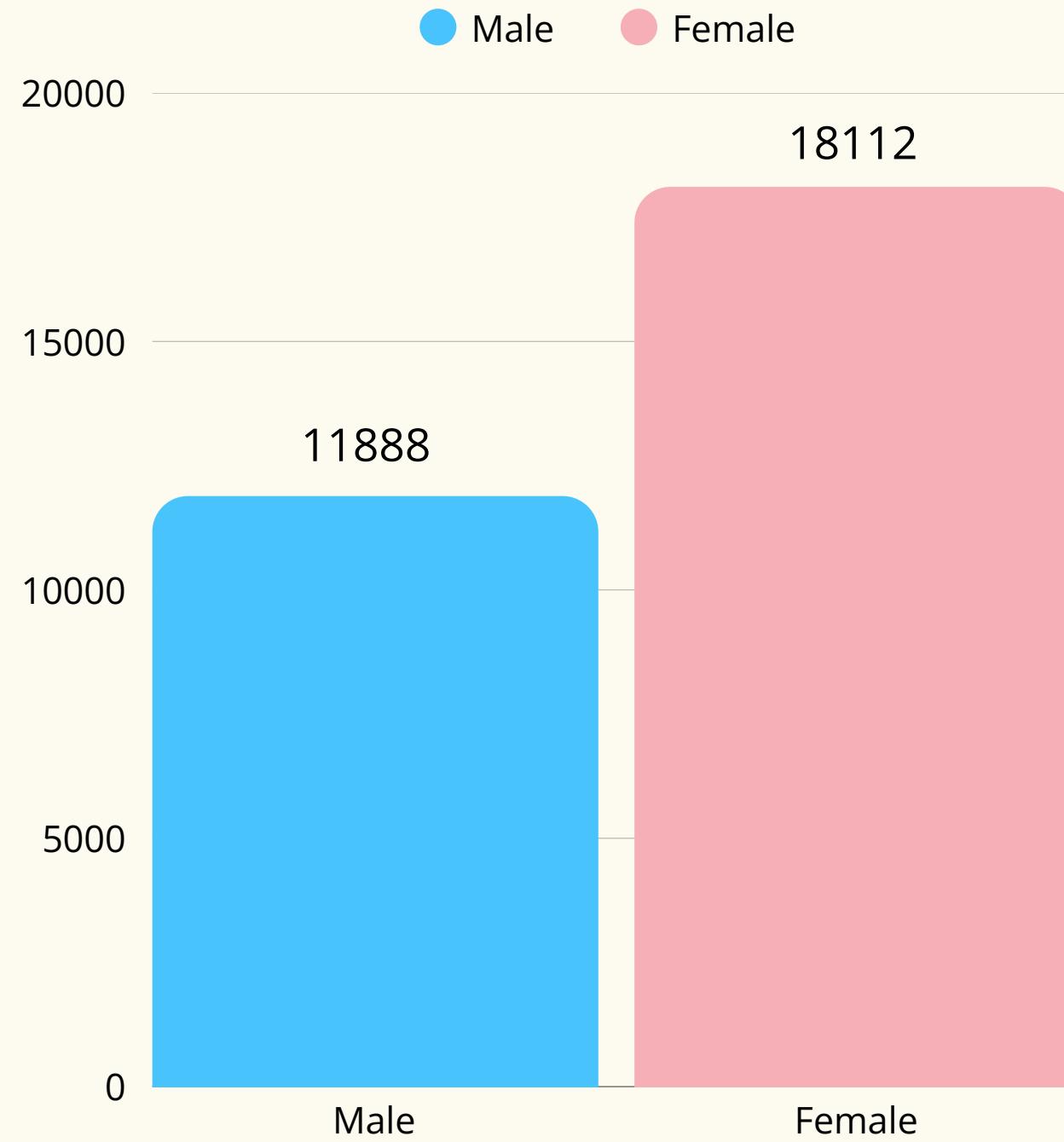


# XU HƯỚNG THANH TOÁN CHẬM

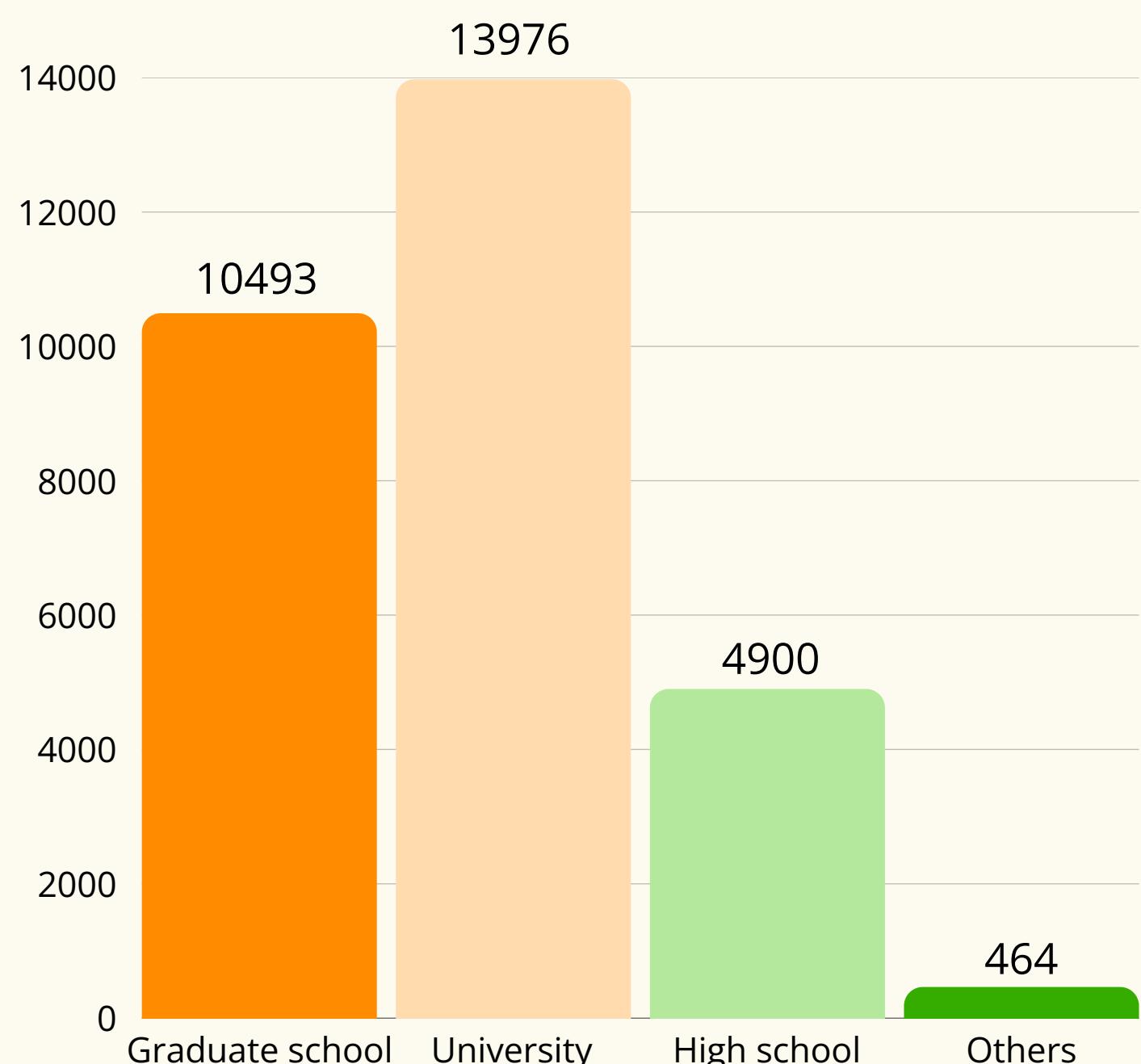


- **Xu hướng tăng:** Từ tháng 4 đến tháng 8, số lần trễ hạn tăng đáng kể từ **5196** trường hợp đến **8608** vào tháng 8.
- Tháng cao điểm (Tháng 8): Tháng có số trường hợp trễ hạn thanh toán cao nhất (8608).
- Giảm dần từ tháng 9: Sau tháng 8, số lượng thanh toán trễ giảm xuống (7,918 trường hợp)

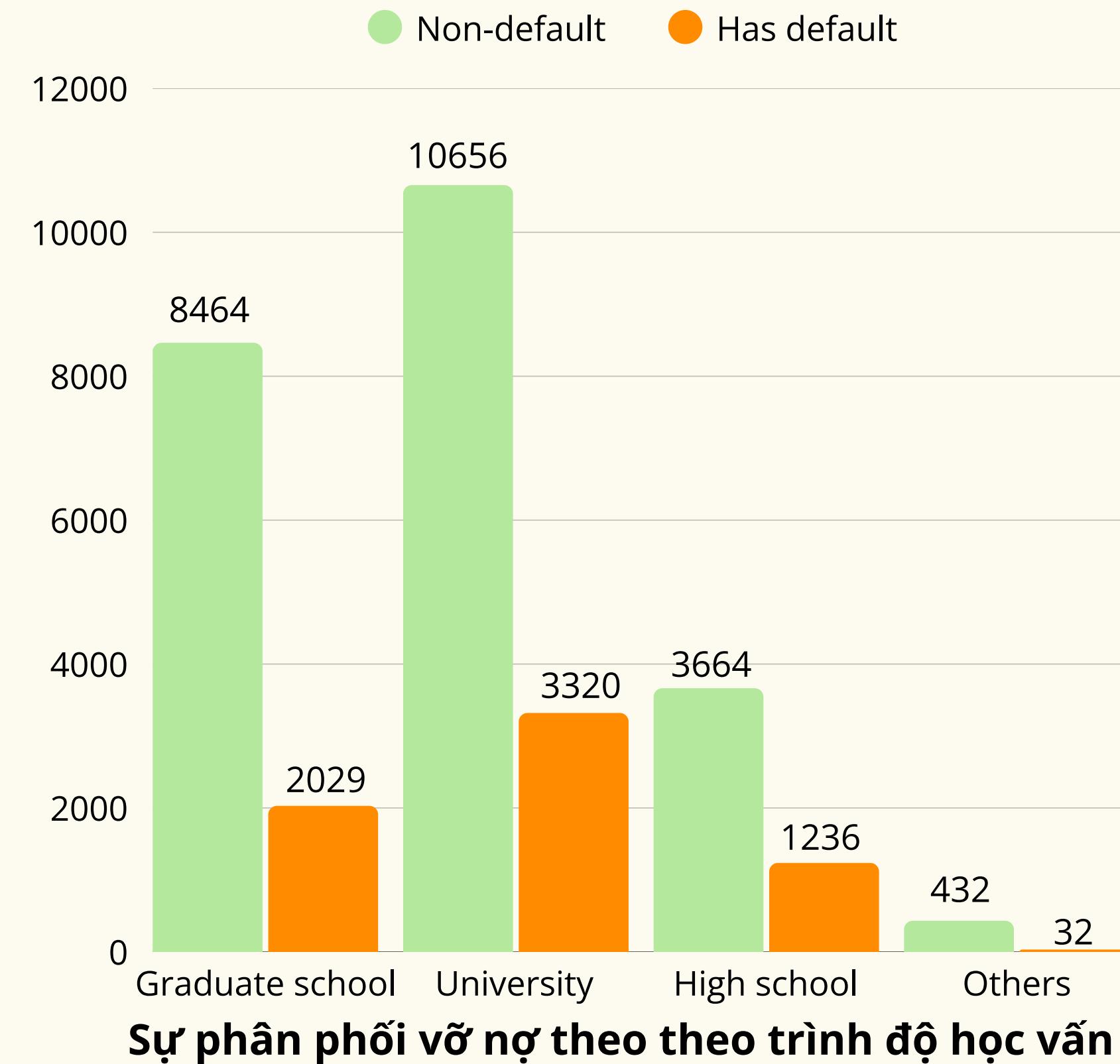
# PHÂN PHỐI THEO BIẾN PHÂN LOẠI



# PHÂN PHỐI THEO BIỂN PHÂN LOẠI

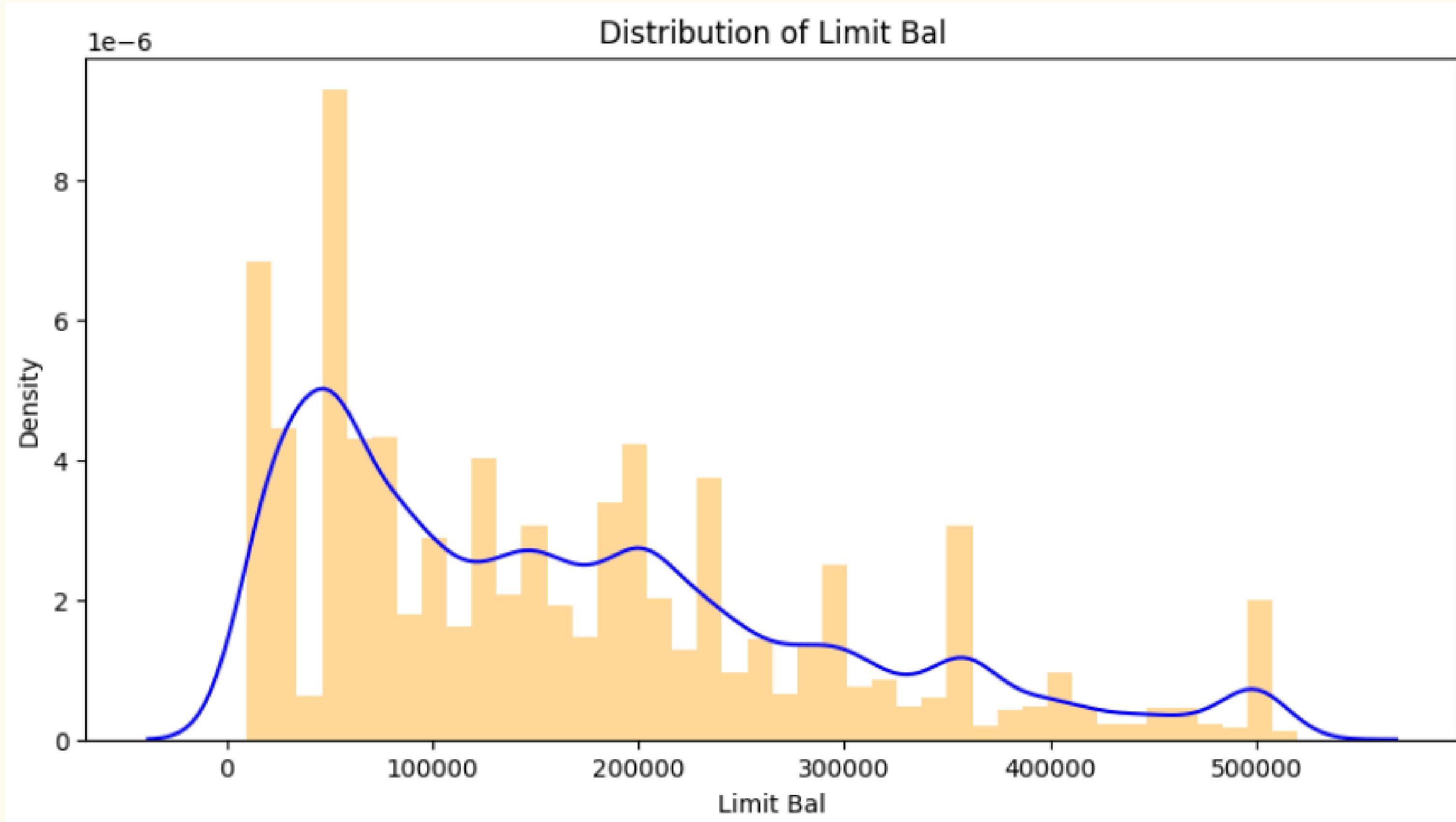


Số lượng khách hàng theo trình độ học vấn

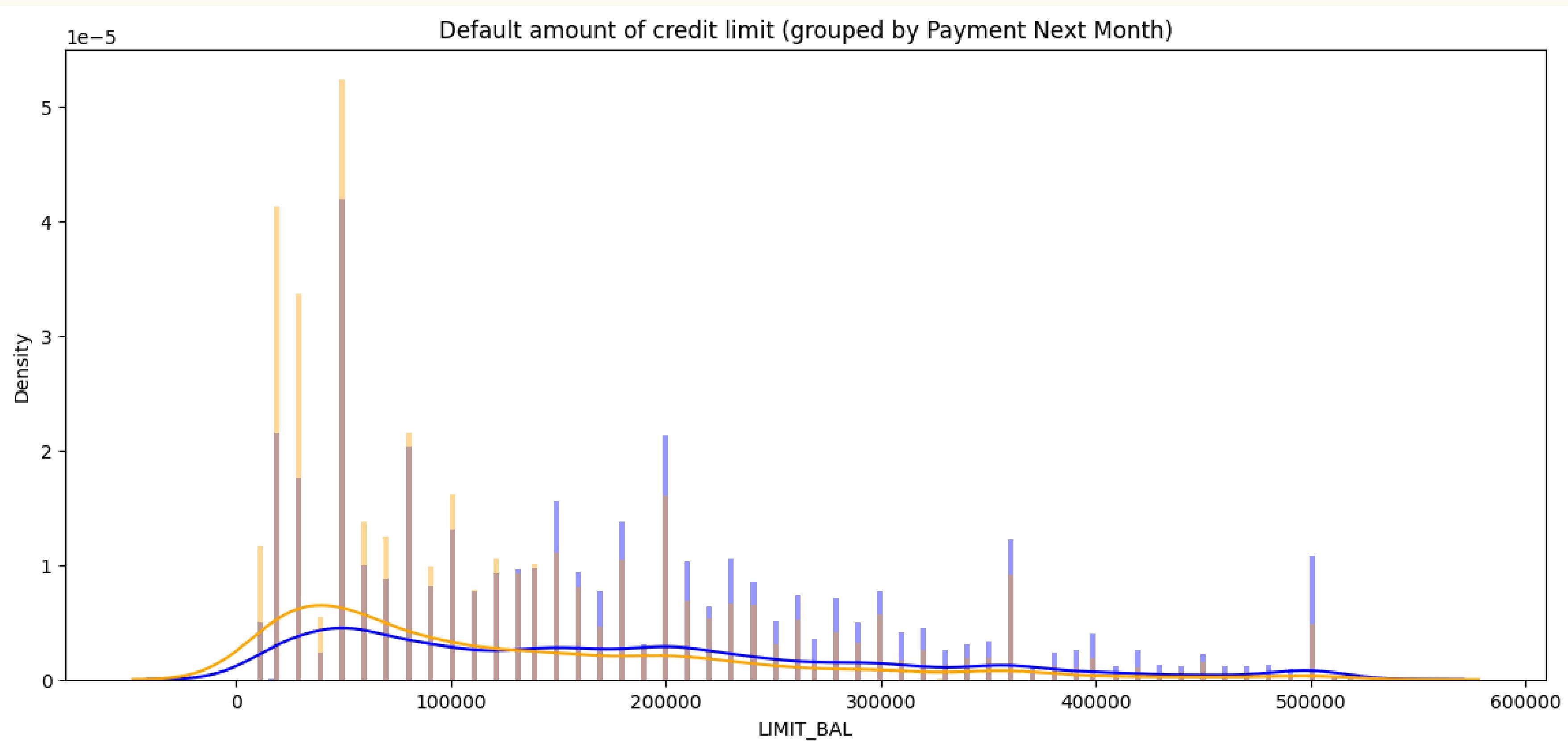


Sự phân phối vỡ nợ theo theo trình độ học vấn

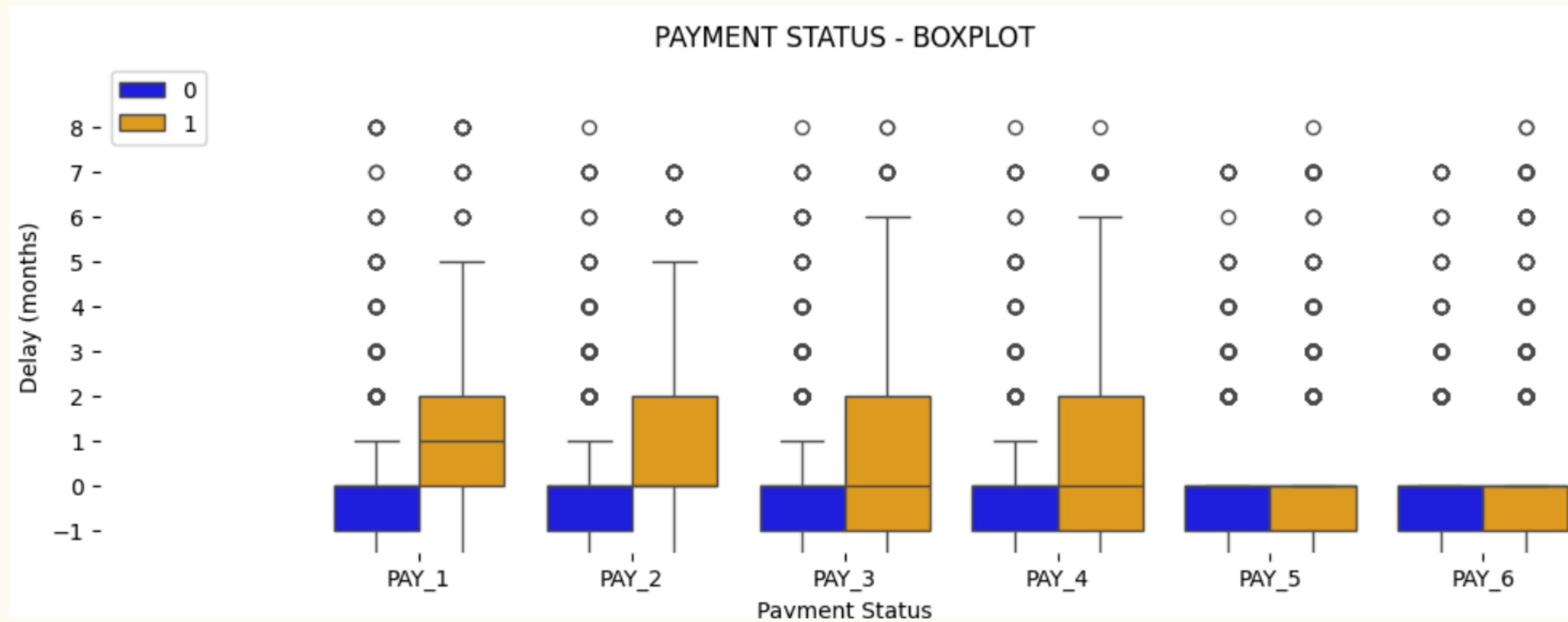
# PHÂN PHỐI HẠN MỨC TÍN DỤNG



# PHÂN PHỐI RỦI RO VỠ NỢ THEO HẠN MỨC TÍN DỤNG

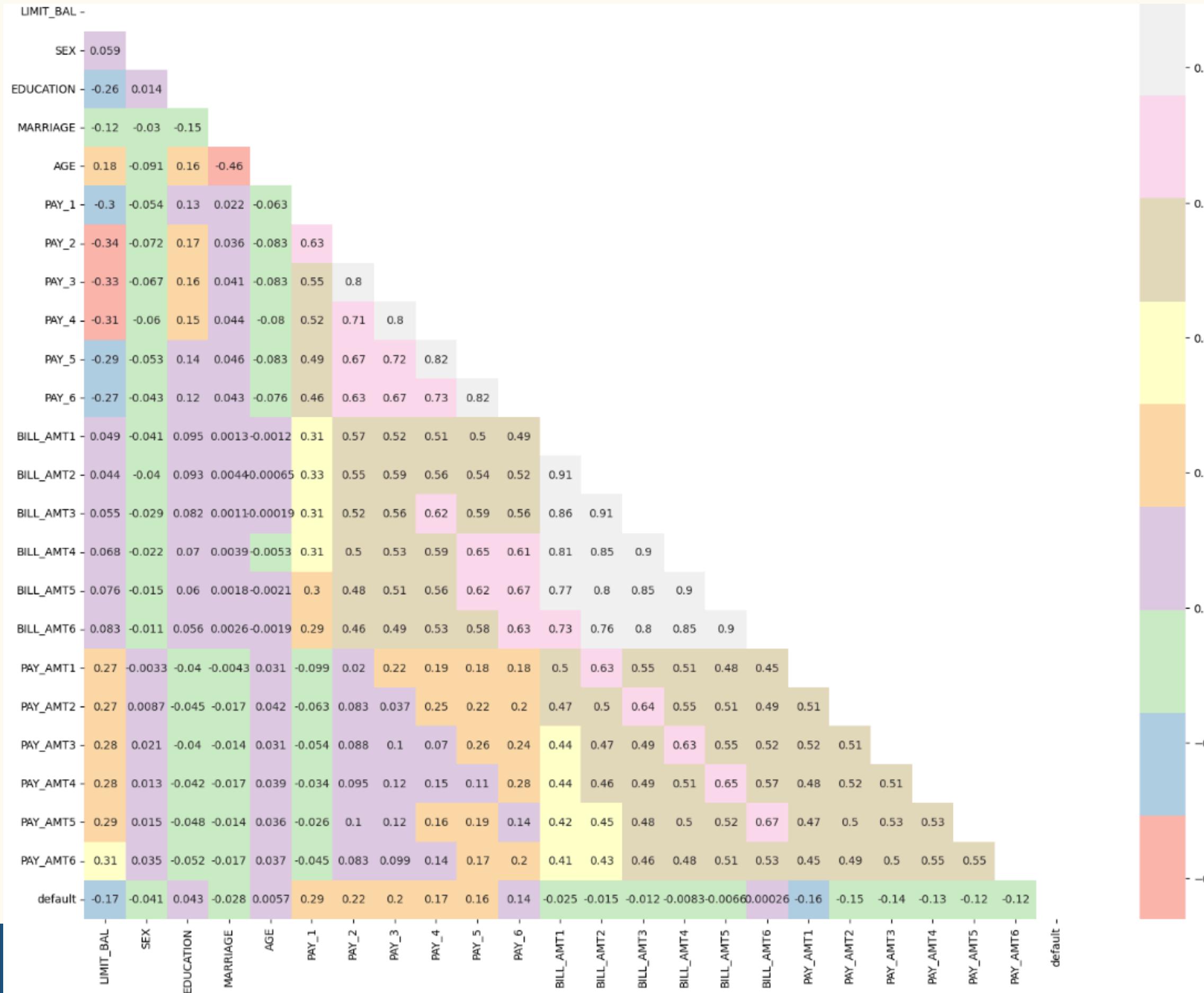


# TRẠNG THÁI THANH TOÁN



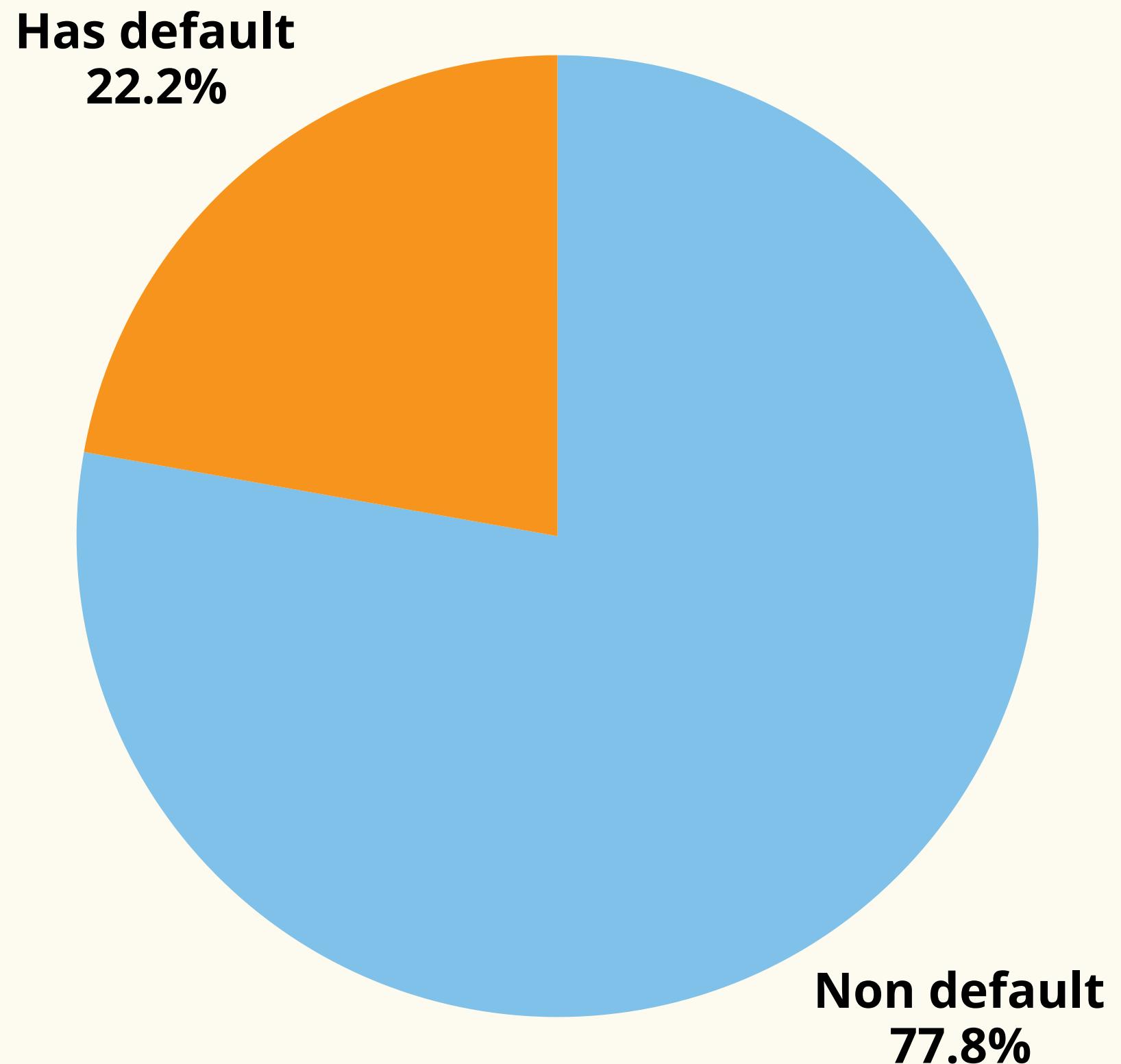
- Có thể thấy rằng những khách hàng chậm thanh toán trong vòng một tháng hoặc ít hơn có ít trường hợp vỡ nợ thẻ tín dụng hơn.
- Đặc biệt, PAY\_1 - trạng thái thanh toán trong tháng 9, có khả năng phân biệt lớn hơn so với trạng thái thanh toán trong các tháng khác

# CORRELATION MATRIX



Mức độ tương quan giữa các đặc trưng và biến mục tiêu yếu  
(dao động từ -0.028 đến 0.22)

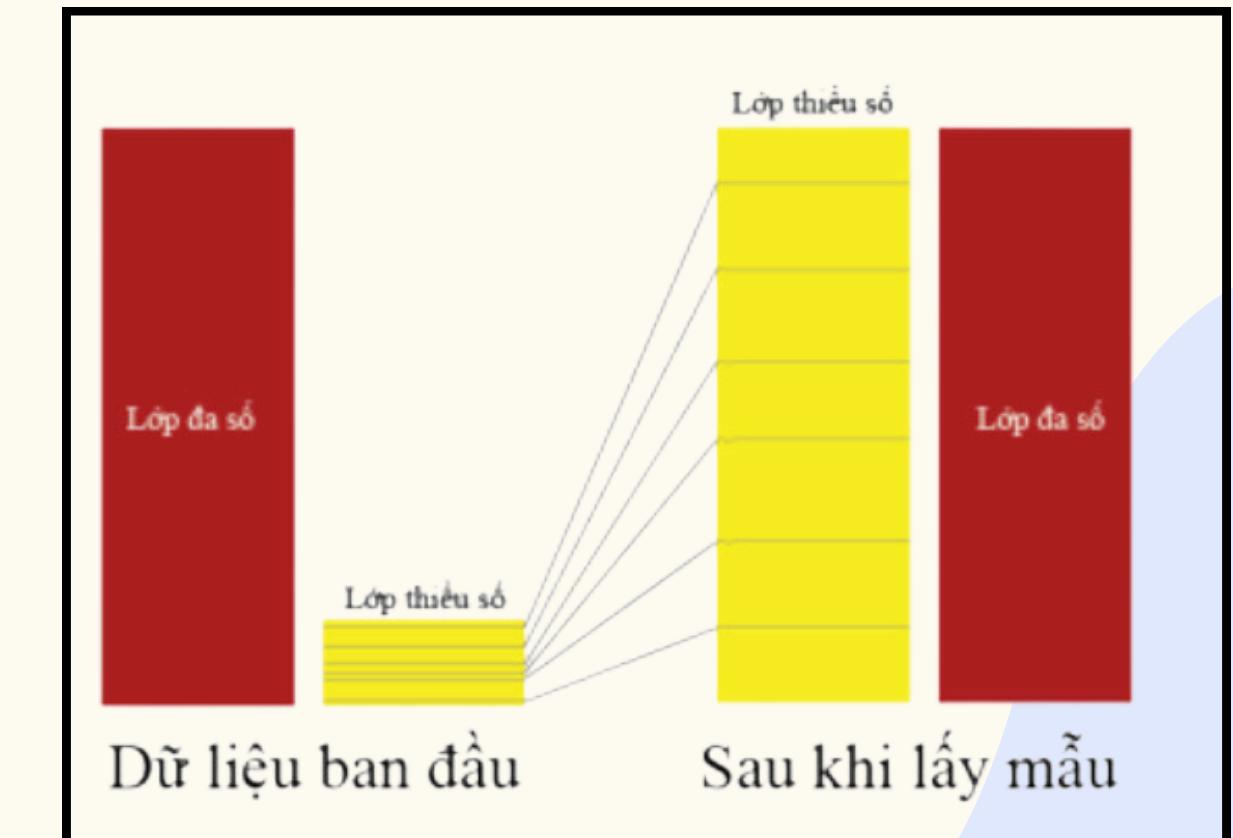
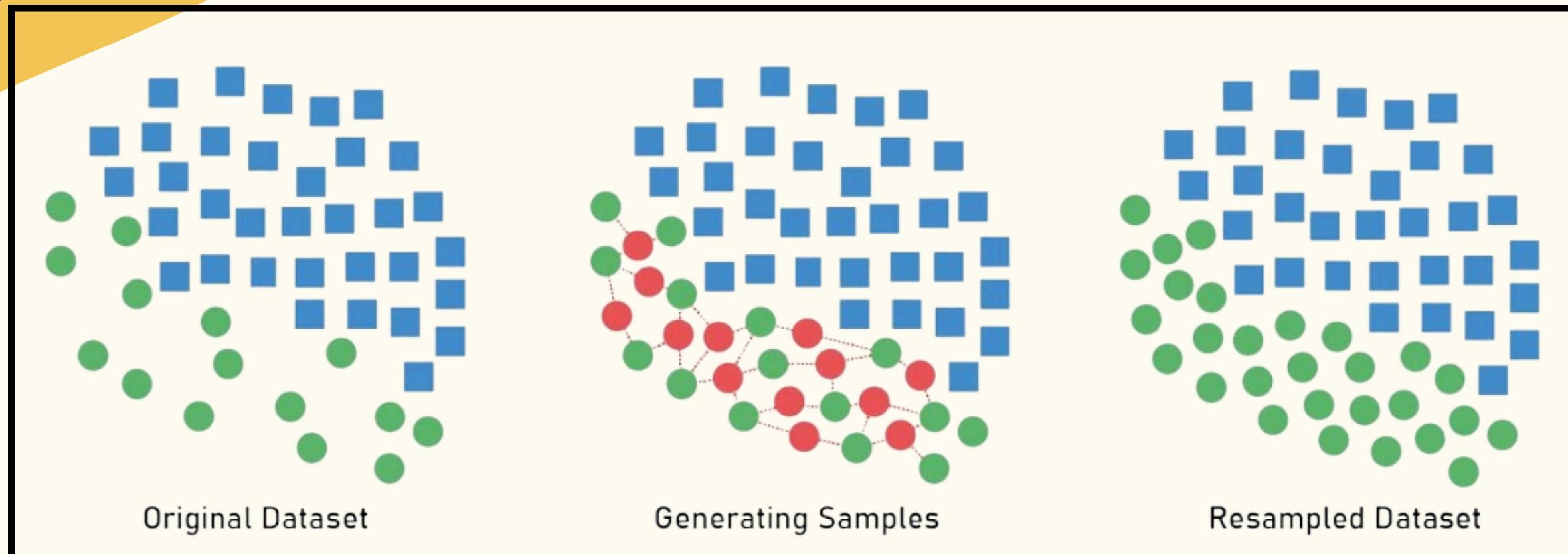
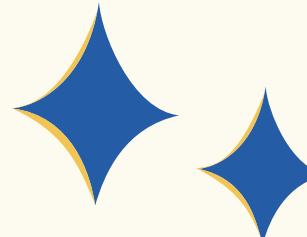
# PHÂN PHỐI TỶ LỆ vỡ NỢ



Xử lý dữ liệu mất cân bằng: **OverSampling**

- Random OverSampling
- SMOTE OverSampling

# OVERSAMPLING



## SMOTE

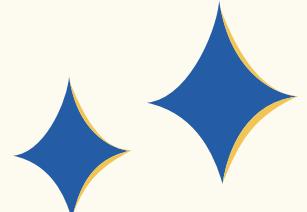
## Random OverSampling



Phát hiện được nhiều giao dịch gian lận hơn và giảm thiểu tỷ lệ dự đoán sai (**False Negative**)  
Tức các **has default** mà mô hình dự đoán nhầm là **non default**.

# DATA SPLITTING

- Chia ngẫu nhiên và không trùng lặp bộ dữ liệu thành các tập **train** và **test**.



```
y = df['Default']
X = df.drop('Default', axis=1)
```

- Đồng thời sẽ tiến hành xử lý mất cân bằng bằng phương pháp **Oversampling, SMOTE** trên tập dữ liệu x,y



## 01 OverSampling

Vì %**has default** > %**Non default**

## 02 Training Test

Biến **input** và **target** của tập train  
→ Huấn luyện mô hình phân loại vỡ nợ.

## 03 Testing Test

Tập **test** sẽ có phân phối giống nhất với dữ liệu thực tế.

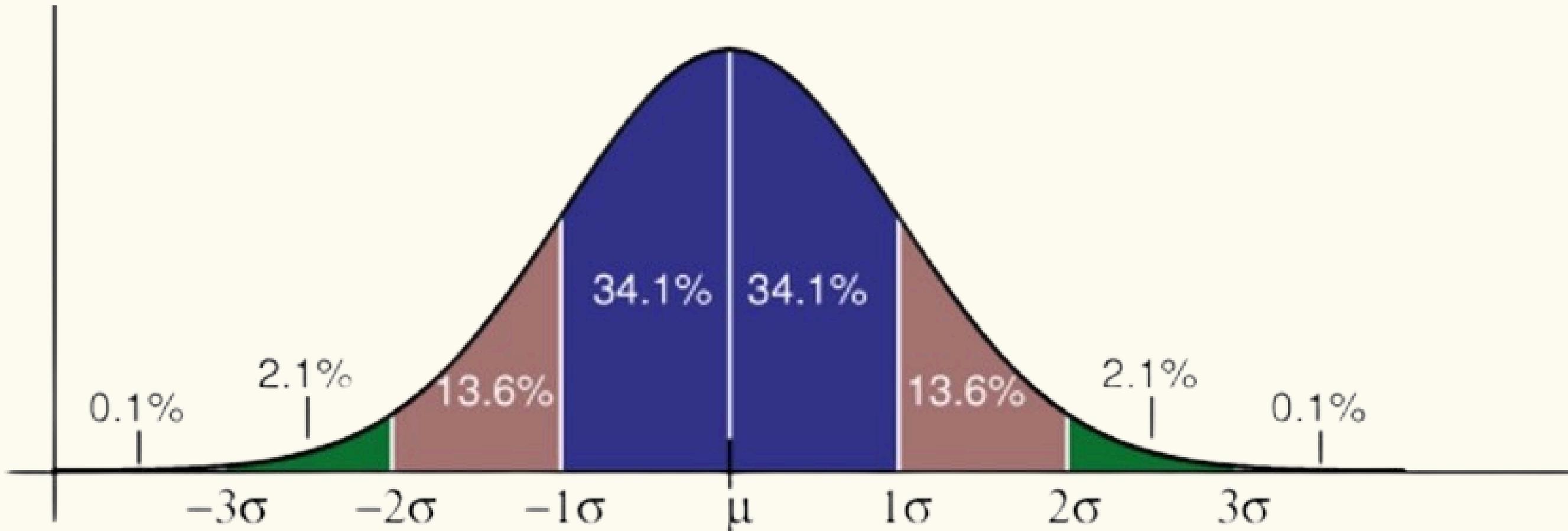
→ đánh giá khả năng áp dụng mô hình vào thực tiễn

# DATA SCALING

- Ứng dụng phân phối **GAUSS** để chuẩn hoá theo phân phối chuẩn các tập dữ liệu X train và test sau khi **resampling**.
- Đảm bảo rằng dữ liệu có phân phối chuẩn

Giá trị trung bình = 0  
Độ lệch chuẩn = 1

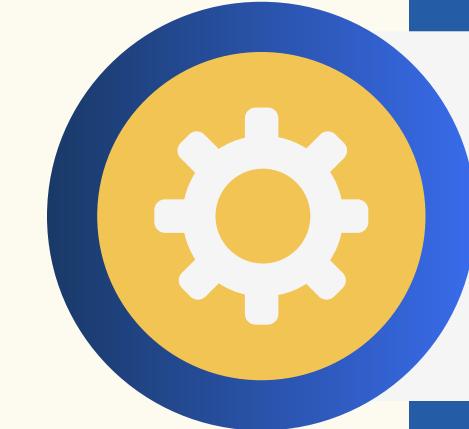
→ Giúp mô hình phân loại hội tụ nhanh hơn và ổn định hơn trên tập dữ liệu



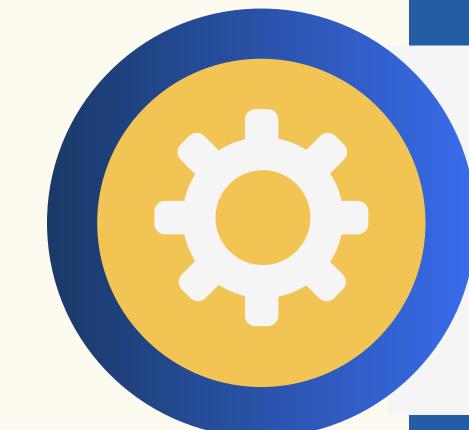
# MODEL CONSTRUCTION



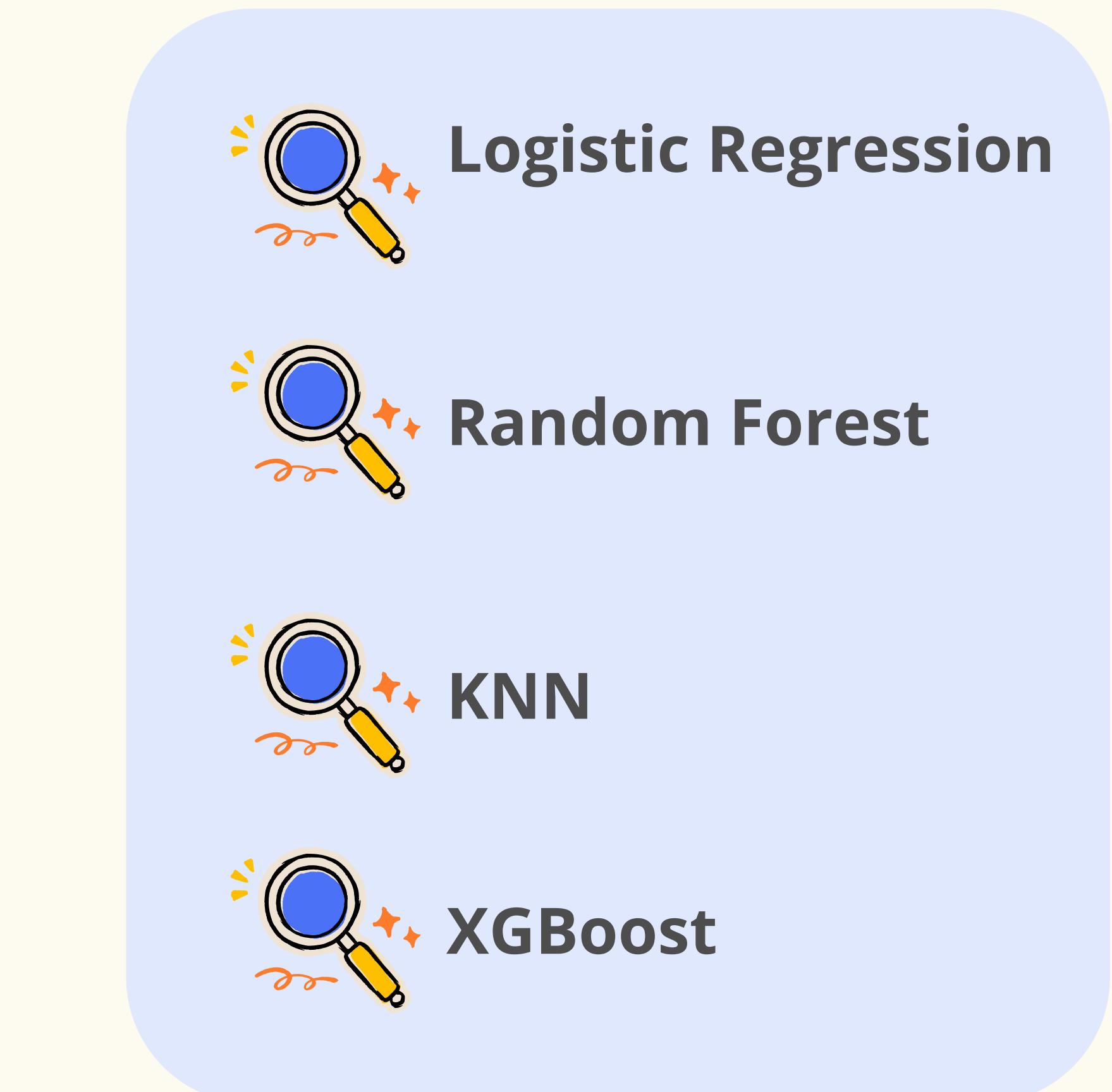
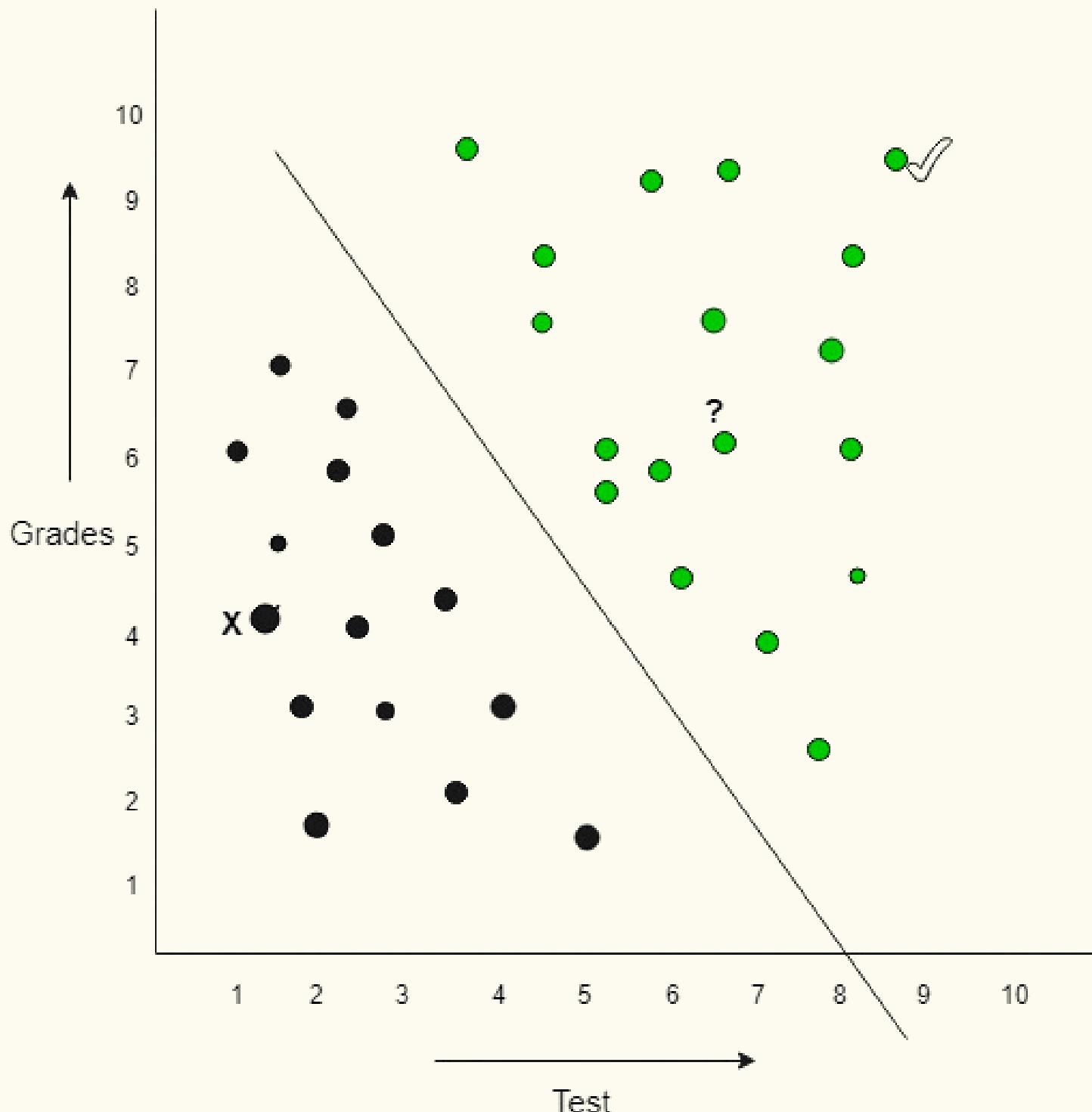
**Random  
OverSampling**



**SMOTE**



# TRAIN MODELS



Chọn ra mô hình hiệu quả và chạy tốt nhất trong các mô hình phân loại  
Để dễ dàng sử dụng cho trong việc dự đoán rủi ro tín dụng



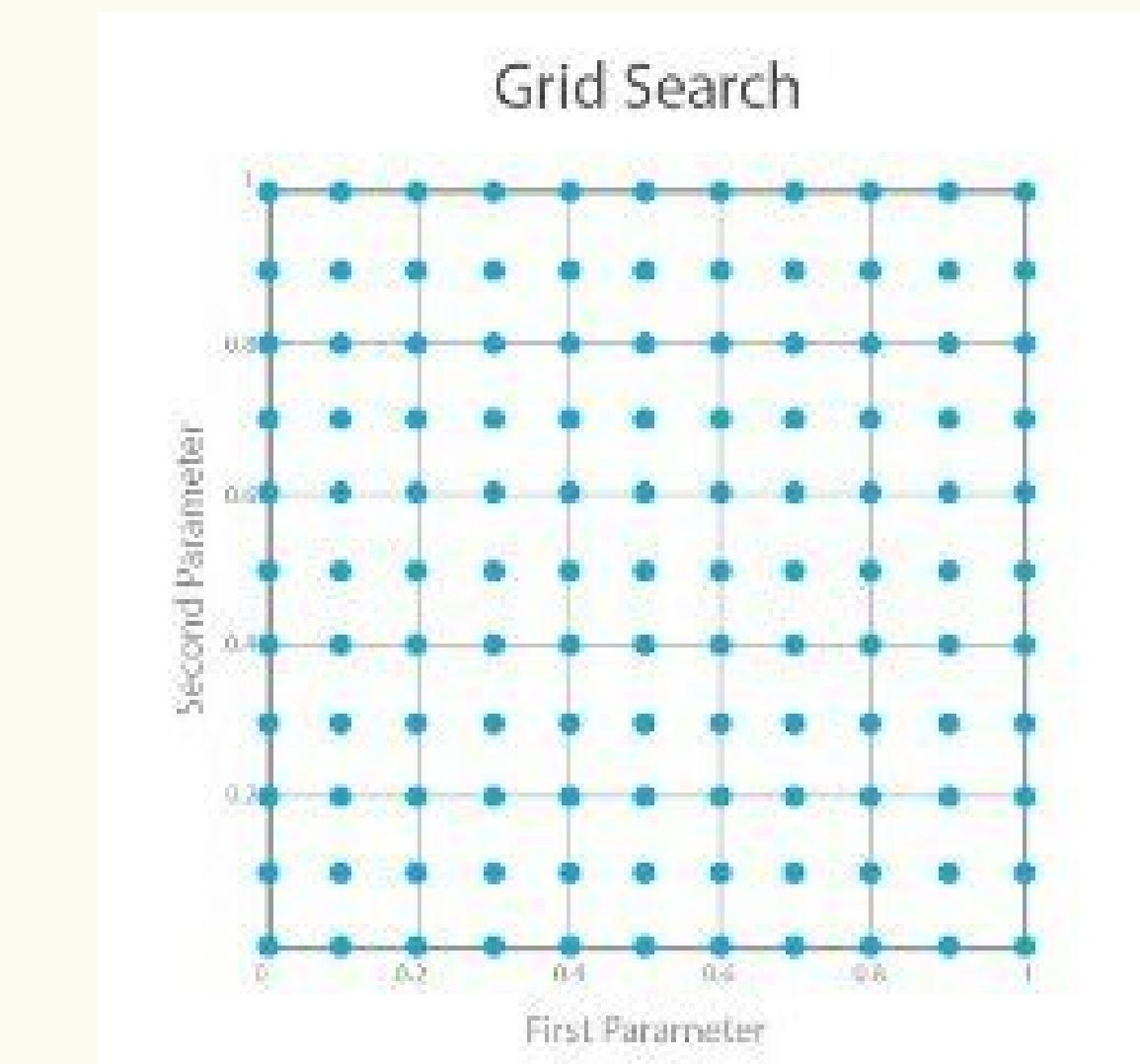
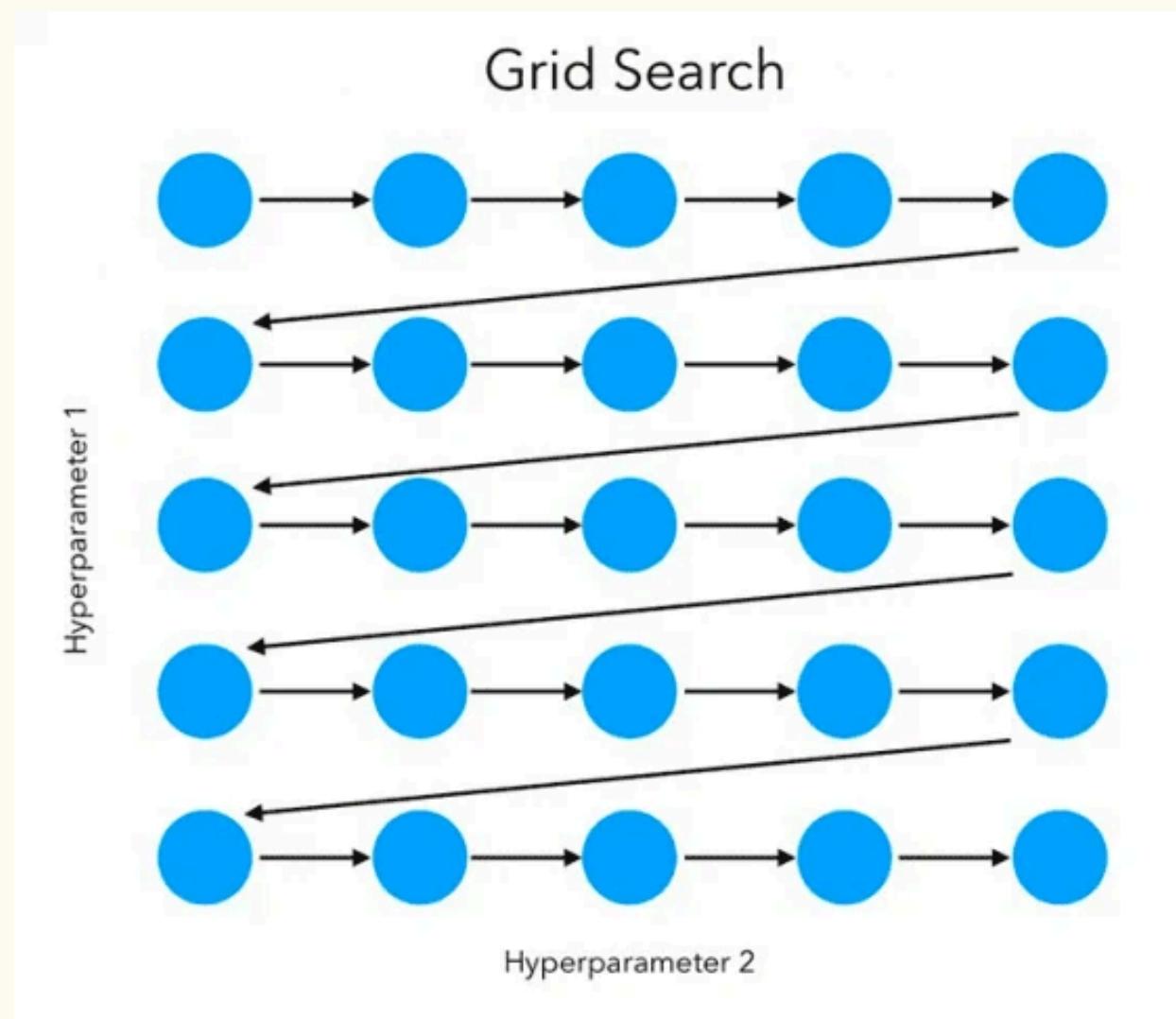
# GRIDSEARCH

**Mục đích:** Tìm ra các siêu tham số tối ưu cho mô hình phân loại



Cải thiện hiệu suất của mô hình hơn so với việc huấn luyện thông thường.

Giúp xác định các giá trị tốt nhất cho các siêu tham số của mô hình



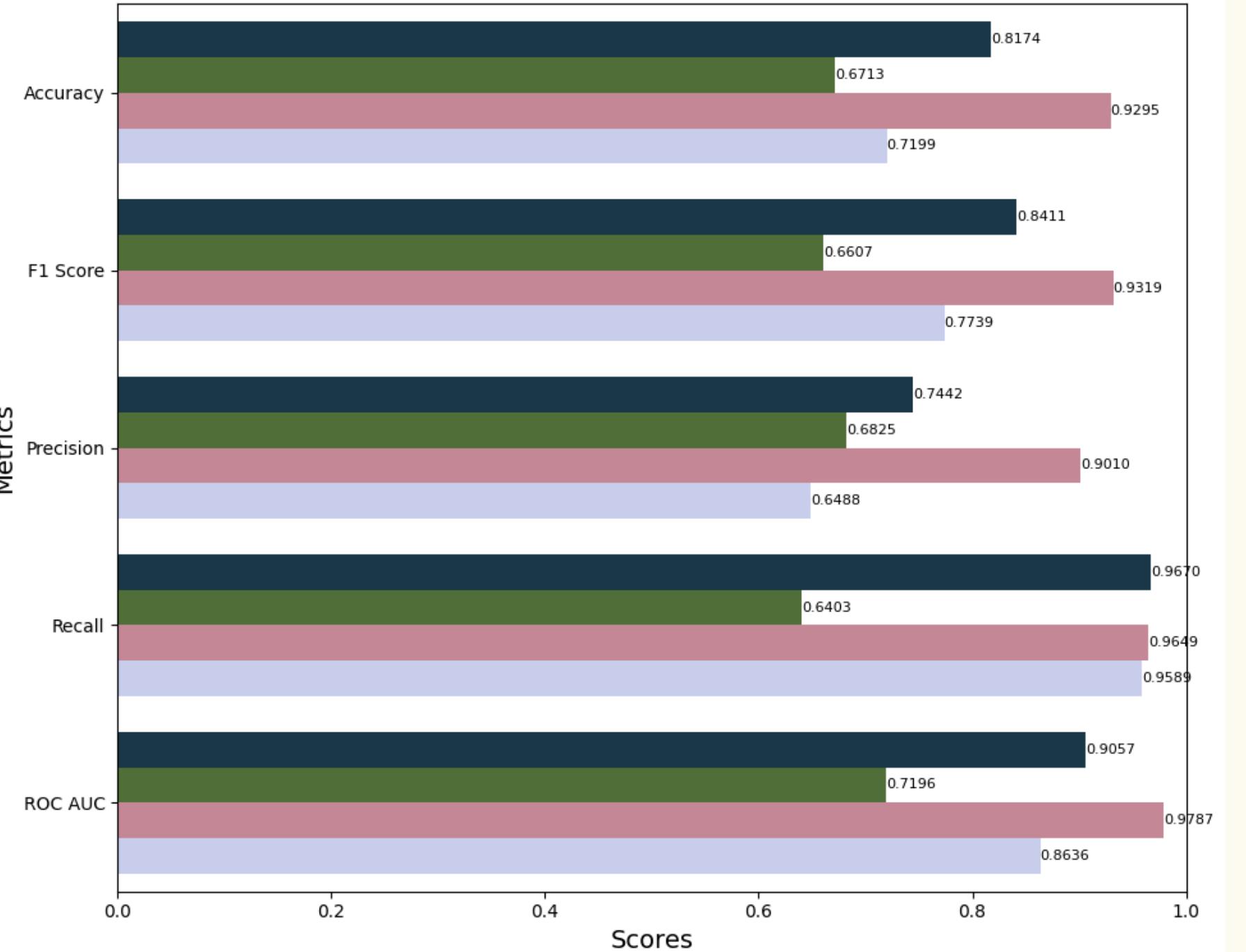
# GRIDSEARCH

```
Oversampling:  
Best parameters: {'C': 1, 'class_weight': 'balanced', 'penalty': 'l1', 'solver': 'liblinear'}  
Best CV score: 0.6615  
SMOTE:  
Best parameters: {'C': 1, 'class_weight': None, 'penalty': 'l1', 'solver': 'liblinear'}  
Best CV score: 0.7223  
  
Random Forest:  
Oversampling:  
Best parameters: {'class_weight': None, 'max_depth': None, 'min_samples_split': 2, 'n_estimators': 100}  
Best CV score: 0.9187  
SMOTE:  
Best parameters: {'class_weight': 'balanced', 'max_depth': None, 'min_samples_split': 2, 'n_estimators': 100}  
Best CV score: 0.8320  
  
KNN:  
Oversampling:  
Best parameters: {'metric': 'euclidean', 'n_neighbors': 3, 'weights': 'distance'}  
Best CV score: 0.8215  
SMOTE:  
Best parameters: {'metric': 'manhattan', 'n_neighbors': 3, 'weights': 'distance'}  
Best CV score: 0.7998  
  
XGBoost:  
Oversampling:  
Best parameters: {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 1, 'n_estimators': 100, 'scale_pos_weight': 3}  
Best CV score: 0.7719  
SMOTE:  
Best parameters: {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 1, 'n_estimators': 100, 'scale_pos_weight': 1}  
Best CV score: 0.8062
```

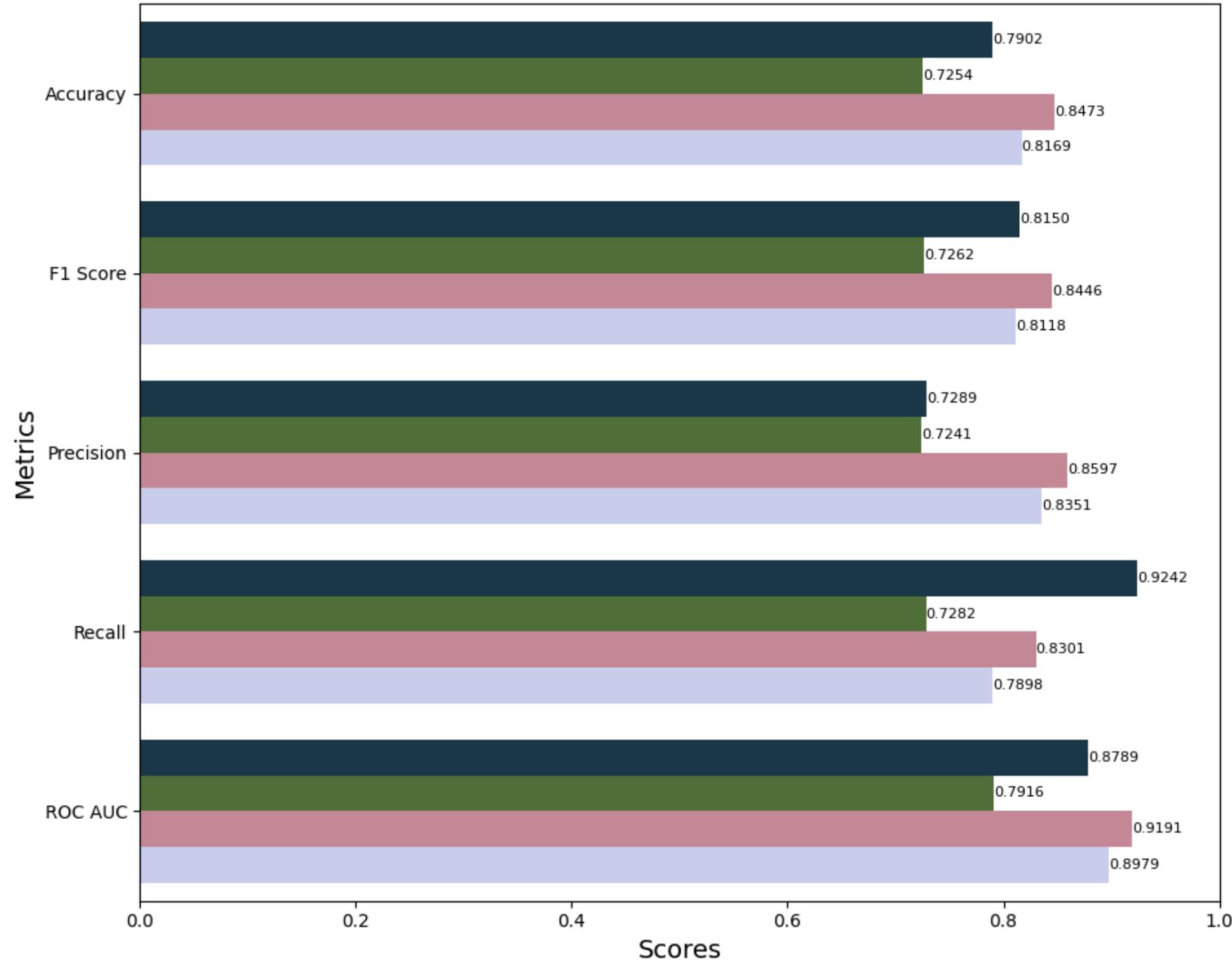
# RANDOM OVERSAMPLING

# SMOTE OVERSAMPLING

Metrics for Oversampling Sampling



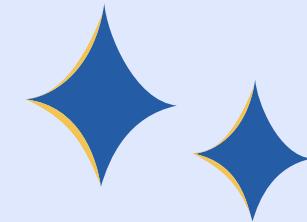
Metrics for SMOTE Sampling



Models

- KNN
- Logistic Regression
- Random Forest
- XGBoost

# ĐÁNH GIÁ CÁC CHỈ SỐ



- Dựa vào bộ chỉ số bao gồm **ROC-AUC, Accuracy, Recall, Precision, F1-Score** của 4 mô hình trong 2 phương pháp **SMOTE Oversampling, Random Oversampling**

Chỉ số cho Phương Pháp Lấy Mẫu Oversampling:

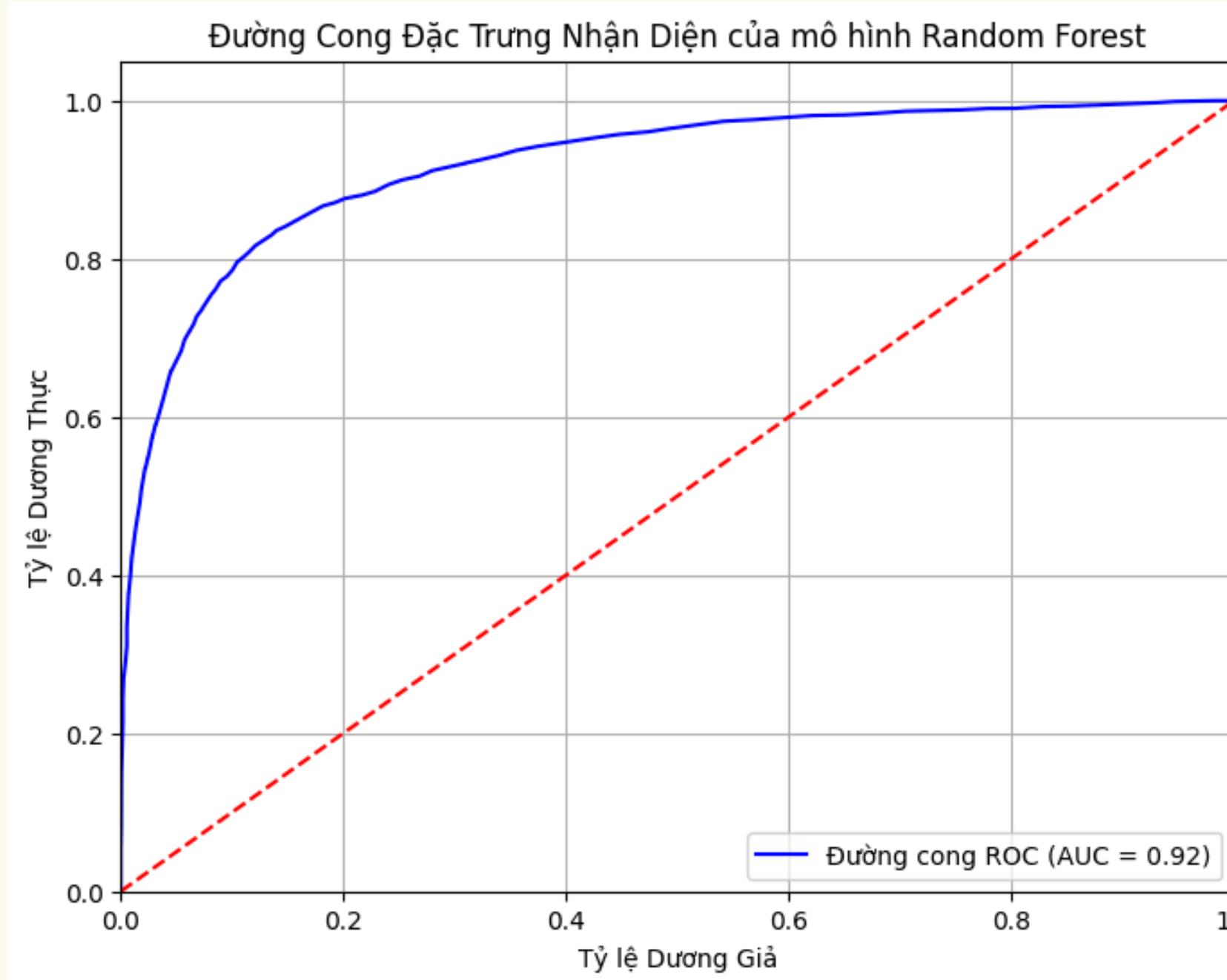
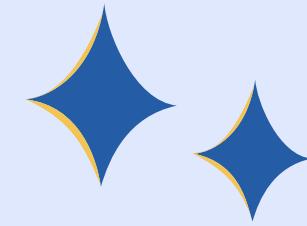
	Model	Accuracy	F1 Score	Precision	Recall	ROC AUC
0	Logistic Regression	0.671261	0.660740	0.682507	0.640319	0.719572
1	Random Forest	0.929471	0.931877	0.901046	0.964893	0.978747
2	KNN	0.817379	0.841139	0.744240	0.967047	0.905702
3	XGBoost	0.719931	0.773924	0.648790	0.958863	0.863557

Chỉ số cho Phương Pháp Lấy Mẫu SMOTE:

	Model	Accuracy	F1 Score	Precision	Recall	ROC AUC
0	Logistic Regression	0.725423	0.726160	0.724138	0.728193	0.791567
1	Random Forest	0.847313	0.844620	0.859692	0.830067	0.919060
2	KNN	0.790244	0.815005	0.728894	0.924187	0.878923
3	XGBoost	0.816948	0.811822	0.835117	0.789791	0.897867

→ **RANDOM FOREST là mô hình tốt nhất** có trung bình các chỉ số cao nhất trong 4 mô hình

# ĐƯỜNG CONG ROC CỦA MÔ HÌNH RANDOM FOREST



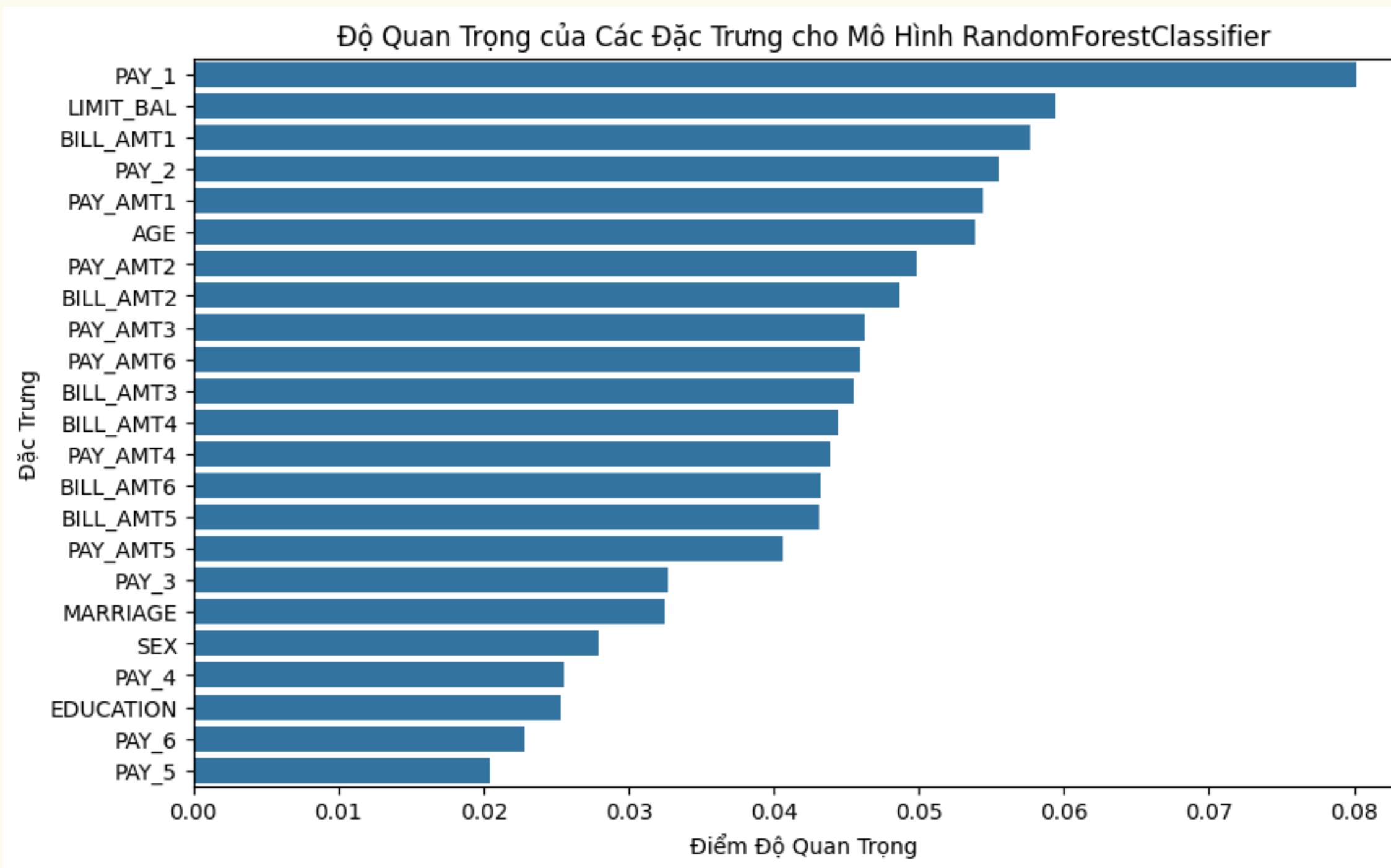
Mức độ ROC ( $AUC = 0.92$ ) được đánh giá là **xuất sắc**.

## Đường cong ROC:

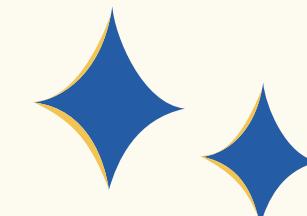
- **AUC = 0.92** cao hơn đường chéo ( $AUC = 0.5$ ) cho thấy mô hình dự đoán tốt hơn so với dự đoán ngẫu nhiên.
- Diện tích dưới đường cong càng lớn, khả năng phân biệt giữa hai nhóm khách hàng vỡ nợ và ko vỡ nợ càng cao

→ Mô hình dự đoán: có **độ hiệu quả cực kỳ cao** với **tỷ lệ sai rất thấp**.

# ĐỘ QUAN TRỌNG CÁC ĐẶC TRƯNG CHO MÔ HÌNH RANDOM FOREST



- **PAY\_1:** Yếu tố quan trọng nhất
- **LIMIT\_BAL:** Ảnh hưởng lớn
- Yếu tố xã hội và nhân khẩu học không ảnh hưởng nhiều đến mô hình



→ Có **tác động lớn nhất** đến khả năng dự đoán chính xác của mô hình.

# ĐÓNG GÓI MÔ HÌNH

- Sử dụng thư viện **JOBLIB**, rất hiệu quả cho việc lưu lại các mô hình phân loại phức tạp như **Random Forest**

→ Đóng gói lưu dưới dạng file có đuôi .Joblib vào thư mục local trong máy theo đường dẫn.

```
from joblib import dump  
dump(final_model, 'BestModel_RF.joblib')
```

Hàm đóng gói mô hình **dump**

- Sau khi đóng gói có thể dùng lại mô hình nếu cần

```
from joblib import load  
Bestmodel_RF = load('BestModel_RF.joblib')
```

Hàm sử dụng lại mô hình **load**

# KẾT LUẬN



## Về phương pháp xử lý mất cân bằng:



Đối với bộ dữ liệu bị mất cân bằng thì nhóm đã tiến hành so sánh hai phương pháp liên quan đến **Oversampling** là **SMOTE** và **Random Oversampling** (ros).



Trong đó phương pháp ros đã được chọn vì nó đáp ứng yêu cầu đề ra là mang lại hiệu suất cao cho mô hình dự đoán.



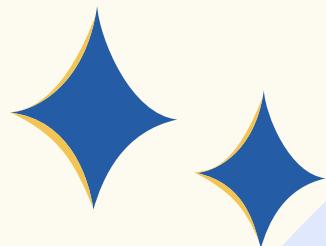
## Về mô hình thuật toán:



- Trong bốn mô hình mô hình đem lại hiệu năng tốt nhất là **Random Forest** khi áp dụng phương pháp **ros** cùng với việc áp dụng thêm **GridSearchCV** để tìm ra bộ siêu tham số tối ưu cho mô hình hoạt động mang lại hiệu suất tốt nhất.

- bài toán đặt ra là dự đoán khả năng vỡ nợ của khách hàng nên chỉ số **recall** là rất quan trọng

- Mô hình **Random Forest** cho chỉ số **recall** cao nhất cũng như là mọi chỉ số đều cao, nghĩa là việc dự đoán cả hai lớp đều mang lại độ chính xác cao



# THANK YOU

Nhóm 5

