Project Report
Applied Statistics and Experimental Design
Credit Card Fraud Detection

Teacher: Applied Statistics and Experimental Design Instructor

Student:

Hoang Van Nhan	20235542
Le Nguyen Phuoc Thanh	20235561
Tran Quang Trong	20235565
Nguyen Cong Son	20235619
Nguyen Khac Viet Anh	20235471

Mục lục

1 Giới thiệu

Gian lận thẻ tín dụng là một vấn đề nghiêm trọng trong lĩnh vực tài chính, gây thiệt hại hàng tỷ đô la mỗi năm cho các tổ chức tài chính và người tiêu dùng. Với sự phát triển của thương mại điện tử và thanh toán trực tuyến, các phương thức gian lận ngày càng tinh vi và khó phát hiện hơn. Việc phát triển các hệ thống phát hiện gian lận hiệu quả trở thành một nhu cầu cấp thiết để bảo vệ người dùng và duy trì niềm tin vào hệ thống thanh toán điện tử.

Dự án này tập trung vào việc phát triển một giải pháp phát hiện gian lận thẻ tín dụng dựa trên dữ liệu từ cuộc thi IEEE-CIS Fraud Detection. Mục tiêu chính là xây dựng một mô hình học máy có khả năng phân loại chính xác các giao dịch gian lận và hợp pháp, đồng thời đạt được hiệu suất cao trên tập dữ liệu kiểm tra.

1.1 Tổng quan về cuộc thi IEEE-CIS Fraud Detection

Cuộc thi IEEE-CIS Fraud Detection được tổ chức bởi IEEE Computational Intelligence Society (CIS) và Vesta Corporation trên nền tảng Kaggle. Cuộc thi cung cấp một tập dữ liệu lớn về các giao dịch thẻ tín dụng thực tế, với mục tiêu xây dựng mô hình phát hiện gian lận hiệu quả. Dữ liệu bao gồm các thông tin về giao dịch và định danh, được ẩn danh để bảo vệ thông tin cá nhân của người dùng.

Thách thức chính của cuộc thi là:

- Dữ liệu mất cân bằng nghiêm trọng (chỉ khoảng 3.5% giao dịch là gian lận)
- Nhiều giá trị thiếu và đặc trưng bị ẩn danh
- Yêu cầu độ chính xác cao để giảm thiểu cả false positives và false negatives
- Dữ liệu có cấu trúc phức tạp với nhiều mối quan hệ tiềm ẩn

1.2 Mục tiêu của dự án

Dự án này được thực hiện trong khuôn khổ môn học Applied Statistics and Experimental Design với các mục tiêu cụ thể sau:

- Nghiên cứu và hiểu rõ cơ sở lý thuyết về phát hiện gian lận thẻ tín dụng
- Phân tích và tiền xử lý dữ liệu từ cuộc thi IEEE-CIS Fraud Detection
- Phát triển và đánh giá các mô hình học máy để phát hiện gian lận
- Cải thiện hiệu suất mô hình thông qua kỹ thuật đặc trưng và tối ưu hóa tham số
- Đánh giá và so sánh hiệu suất của các phương pháp khác nhau

Thông qua dự án này, chúng tôi không chỉ áp dụng các kiến thức thống kê và thiết kế thực nghiệm đã học, mà còn phát triển kỹ năng phân tích dữ liệu và xây dựng mô hình học máy trong một bài toán thực tế có ý nghĩa.

2 Cơ sở lý thuyết

2.1 Phát hiện gian lận thẻ tín dụng

Phát hiện gian lận thẻ tín dụng là một bài toán phân loại nhị phân, trong đó mục tiêu là xác định liệu một giao dịch có phải là gian lận hay không. Đây là một bài toán đặc biệt thách thức vì một số lý do sau:

- **Mất cân bằng dữ liệu**: Tỷ lệ giao dịch gian lận thường rất thấp (khoảng 1-3%) so với tổng số giao dịch, dẫn đến vấn đề mất cân bằng nghiêm trọng trong dữ liệu.
- **Chi phí sai sót không đối xứng**: Chi phí của việc bỏ sót một giao dịch gian lận (false negative) thường cao hơn nhiều so với việc phân loại sai một giao dịch hợp pháp (false positive).
- **Phân phối thay đổi theo thời gian**: Các mô hình gian lận thường thay đổi theo thời gian khi kẻ gian lận phát triển các phương pháp mới, dẫn đến hiện tượng "concept drift".
- **Tính thời gian thực**: Các hệ thống phát hiện gian lận cần hoạt động gần như thời gian thực để ngăn chặn giao dịch gian lận trước khi hoàn tất.

2.2 Phương pháp học máy trong phát hiện gian lận

Các phương pháp học máy đã chứng minh hiệu quả trong việc phát hiện gian lận thẻ tín dụng. Trong dự án này, chúng tôi tập trung vào các thuật toán Gradient Boosting, một họ các phương pháp học máy mạnh mẽ đặc biệt phù hợp cho bài toán phân loại:

2.2.1 Gradient Boosting

Gradient Boosting là một kỹ thuật học máy tạo ra một mô hình dự đoán dưới dạng tổng hợp của nhiều mô hình yếu, thường là cây quyết định. Ý tưởng cơ bản là xây dựng các mô hình một cách tuần tự, mỗi mô hình cố gắng sửa chữa lỗi của các mô hình trước đó.

Quá trình huấn luyện của Gradient Boosting có thể được mô tả như sau:

1. Khởi tạo mô hình với một giá trị hằng số
2. Cho mỗi lần lặp $m = 1, 2, ..., M$:
   (a) Tính gradient của hàm mất mát đối với dự đoán của mô hình hiện tại
   (b) Huấn luyện một mô hình yếu (cây quyết định) để dự đoán gradient
   (c) Cập nhật mô hình bằng cách thêm mô hình yếu mới với một trọng số phù hợp

Trong dự án này, chúng tôi sử dụng ba biến thể hiệu quả của Gradient Boosting:

2.2.2 LightGBM

LightGBM (Light Gradient Boosting Machine) là một framework Gradient Boosting được Microsoft phát triển, tập trung vào hiệu suất và khả năng mở rộng. Các đặc điểm chính của LightGBM bao gồm:

- **Histogram-based algorithm**: Chuyển đổi các giá trị đặc trưng liên tục thành bins rời rạc, giảm đáng kể thời gian tính toán và sử dụng bộ nhớ.
- **Leaf-wise tree growth**: Thay vì phát triển cây theo cấp độ (level-wise) như các thuật toán truyền thống, LightGBM phát triển cây theo lá (leaf-wise), chọn lá có tổn thất lớn nhất để phân chia, dẫn đến giảm lỗi nhanh hơn.
- **GOSS (Gradient-based One-Side Sampling)**: Kỹ thuật lấy mẫu giữ lại tất cả các mẫu có gradient lớn (đóng góp nhiều vào thông tin) và lấy mẫu ngẫu nhiên từ các mẫu có gradient nhỏ.
- **EFB (Exclusive Feature Bundling)**: Kỹ thuật gộp các đặc trưng thưa thớt, giảm số lượng đặc trưng mà không mất nhiều thông tin.

2.2.3 XGBoost

XGBoost (eXtreme Gradient Boosting) là một thuật toán Gradient Boosting được tối ưu hóa, nổi tiếng với hiệu suất cao và khả năng xử lý dữ liệu lớn. Các đặc điểm chính của XGBoost bao gồm:

- **Regularization**: Thêm các thành phần chính quy hóa vào hàm mục tiêu để kiểm soát độ phức tạp của mô hình và ngăn chặn overfitting.
- **Approximate Greedy Algorithm**: Sử dụng thuật toán tham lam xấp xỉ để tìm điểm phân chia tối ưu, cải thiện hiệu suất trên dữ liệu lớn.
- **Sparsity Awareness**: Xử lý hiệu quả dữ liệu thưa thớt thông qua thuật toán phân chia cây được tối ưu hóa.
- **Built-in Cross-Validation**: Hỗ trợ cross-validation tích hợp để dễ dàng tinh chỉnh tham số.

2.2.4 CatBoost

CatBoost là một thuật toán Gradient Boosting được Yandex phát triển, đặc biệt hiệu quả trong việc xử lý dữ liệu phân loại. Các đặc điểm chính của CatBoost bao gồm:

- **Ordered Boosting**: Một thuật toán boosting mới giảm thiểu hiện tượng target leakage trong quá trình huấn luyện.
- **Automatic handling of categorical features**: Tự động xử lý các đặc trưng phân loại thông qua nhiều phương pháp mã hóa tiên tiến.
- **Fast and scalable implementation**: Triển khai hiệu quả cho cả CPU và GPU, cho phép huấn luyện nhanh trên dữ liệu lớn.
- **Robust to overfitting**: Thiết kế để giảm thiểu overfitting, đặc biệt khi làm việc với dữ liệu không cân bằng.

2.3 Kỹ thuật Ensemble

Kỹ thuật Ensemble là phương pháp kết hợp nhiều mô hình để tạo ra một mô hình mạnh hơn. Trong dự án này, chúng tôi sử dụng phương pháp Blending, một dạng đơn giản của Stacking:

- **Blending**: Kết hợp dự đoán từ nhiều mô hình khác nhau bằng cách lấy trung bình có trọng số. Trọng số được xác định thông qua thử nghiệm trên tập validation.
- **Stacking**: Sử dụng dự đoán từ nhiều mô hình cơ sở làm đầu vào cho một mô hình meta-learner. Trong dự án này, chúng tôi sử dụng LogisticRegression làm meta-learner.

2.4 Đánh giá hiệu suất

Trong bài toán phát hiện gian lận thẻ tín dụng, việc lựa chọn thước đo đánh giá phù hợp rất quan trọng do tính chất mất cân bằng của dữ liệu. Chúng tôi sử dụng AUC-ROC (Area Under the Receiver Operating Characteristic Curve) làm thước đo chính:

- **AUC-ROC**: Đo lường khả năng phân biệt của mô hình giữa các lớp dương tính và âm tính. AUC-ROC không bị ảnh hưởng bởi sự mất cân bằng của dữ liệu và ngưỡng phân loại.
- **Validation strategy**: Sử dụng k-fold cross-validation và time-based validation để đánh giá hiệu suất mô hình một cách đáng tin cậy.

3 Phân tích dữ liệu

3.1 Mô tả dữ liệu

Dữ liệu từ cuộc thi IEEE-CIS Fraud Detection bao gồm hai loại thông tin chính:

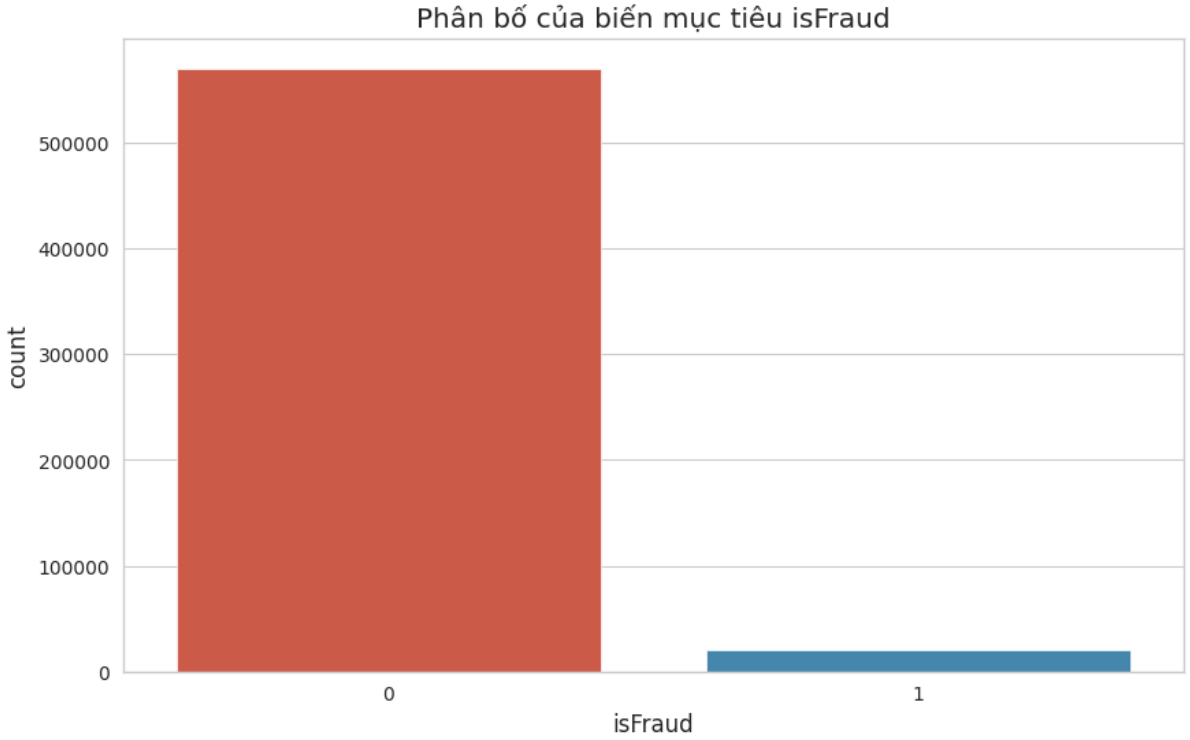
- **Dữ liệu giao dịch (Transaction)**: Chứa thông tin về các giao dịch như số tiền, thời gian, và các đặc trưng khác liên quan đến giao dịch.
- **Dữ liệu định danh (Identity)**: Chứa thông tin về người dùng thực hiện giao dịch, như địa chỉ email, thiết bị, địa chỉ IP, v.v.

Tập dữ liệu huấn luyện bao gồm 590,540 giao dịch, trong đó có 20,663 giao dịch gian lận (khoảng 3.5%). Tập dữ liệu kiểm tra bao gồm 506,691 giao dịch cần được phân loại.

3.2 Phân tích đặc trưng

3.2.1 Phân bố của biến mục tiêu

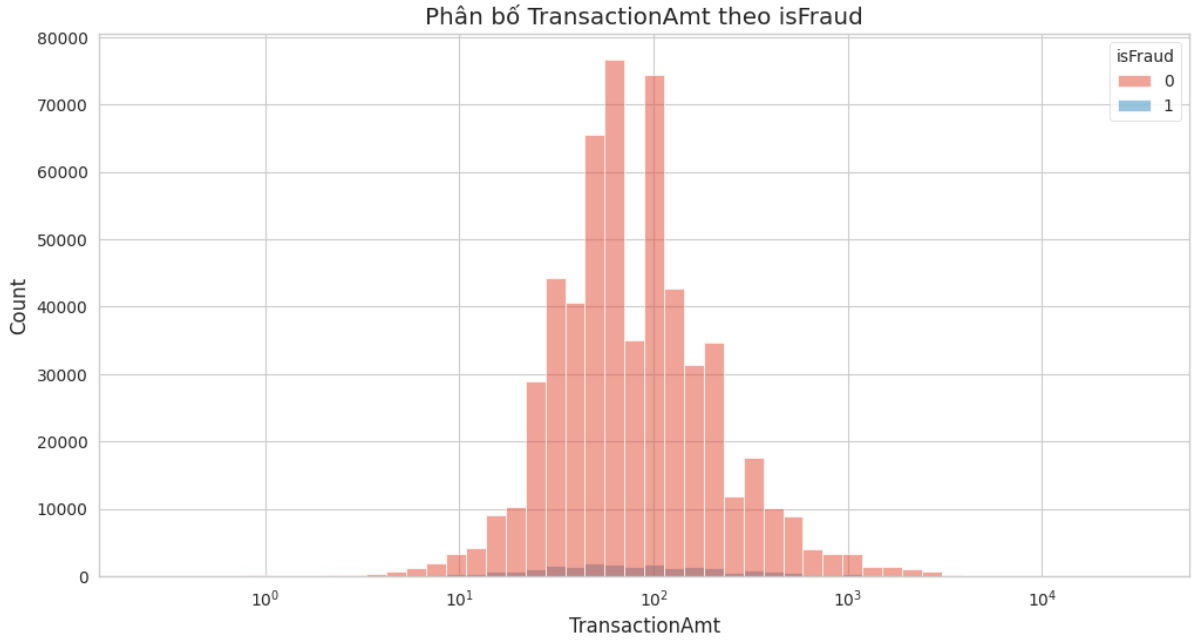
Biến mục tiêu `isFraud` có sự mất cân bằng nghiêm trọng, với chỉ khoảng 3.5% giao dịch được gắn nhãn là gian lận.



Hình 1: Phân bố của biến mục tiêu isFraud

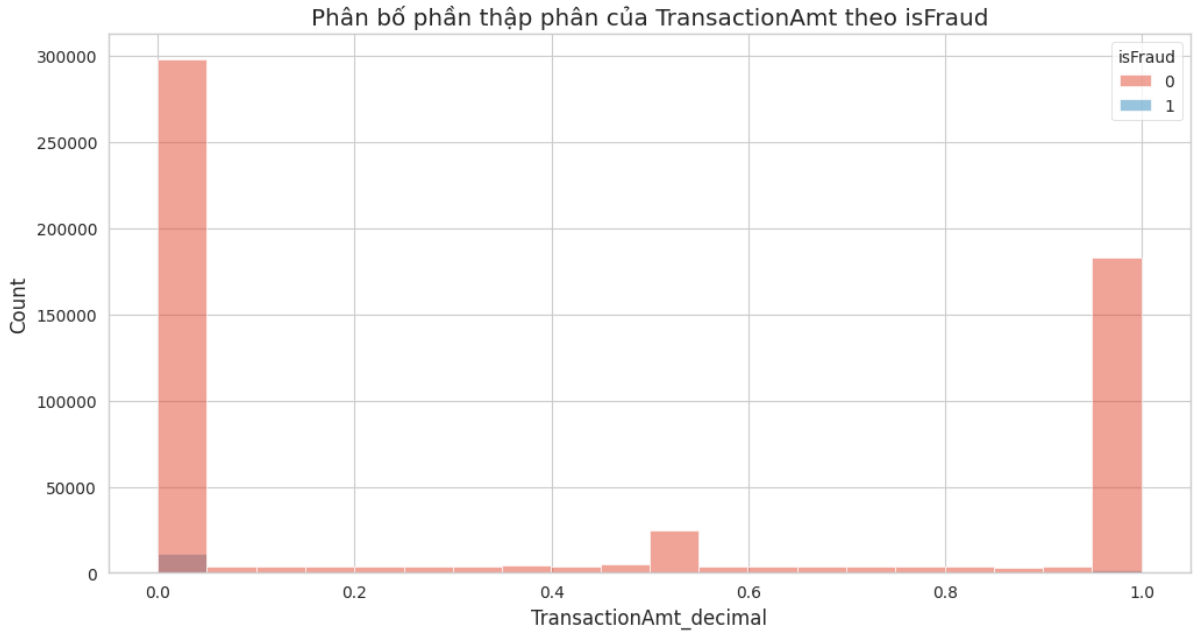
3.2.2 Phân tích số tiền giao dịch

Số tiền giao dịch (`TransactionAmt`) là một đặc trưng quan trọng trong việc phát hiện gian lận. Phân tích cho thấy các giao dịch gian lận thường có phân bố số tiền khác biệt so với các giao dịch hợp pháp.



Hình 2: Phân bố TransactionAmt theo isFraud

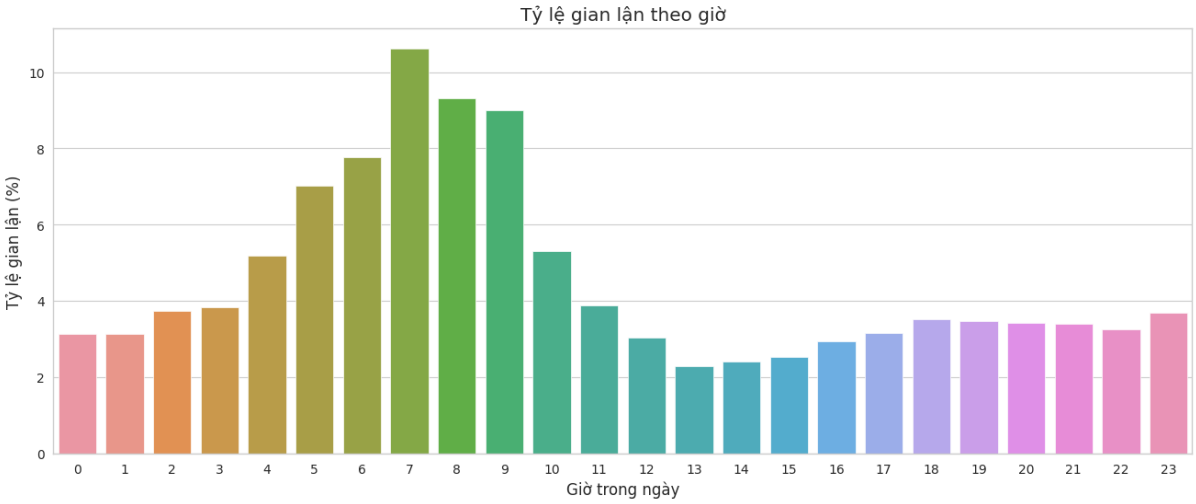
Đặc biệt, phần thập phân của số tiền giao dịch cũng cho thấy sự khác biệt đáng kể giữa giao dịch gian lận và hợp pháp:



Hình 3: Phân bố phần thập phân của TransactionAmt theo isFraud

3.2.3 Phân tích theo thời gian

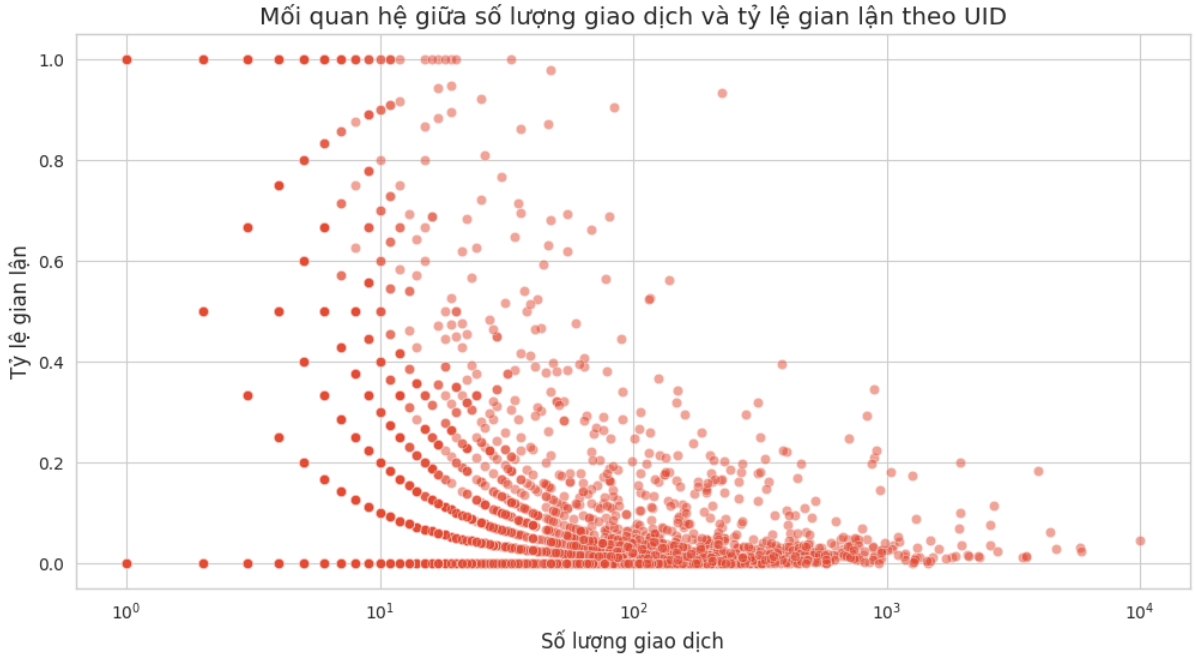
Tỷ lệ gian lận thay đổi theo thời gian trong ngày, với một số khung giờ có tỷ lệ gian lận cao hơn đáng kể:



Hình 4: Tỷ lệ gian lận theo giờ

3.2.4 Phân tích theo định danh khách hàng

Một phát hiện quan trọng là mối quan hệ giữa số lượng giao dịch của một khách hàng (được xác định bằng UID) và tỷ lệ gian lận:



Hình 5: Mối quan hệ giữa số lượng giao dịch và tỷ lệ gian lận theo UID

Biểu đồ cho thấy khách hàng có ít giao dịch thường có tỷ lệ gian lận cao hơn, trong khi khách hàng có nhiều giao dịch thường có tỷ lệ gian lận thấp hơn. Điều này gợi ý rằng

việc phân loại theo khách hàng thay vì từng giao dịch riêng lẻ có thể cải thiện hiệu suất mô hình.

4 Phương pháp tiếp cận

4.1 Tiền xử lý dữ liệu

4.1.1 Xử lý giá trị thiếu

Dữ liệu có nhiều giá trị thiếu, đặc biệt là trong tập dữ liệu định danh. Chúng tôi áp dụng các chiến lược sau để xử lý giá trị thiếu:

- Đối với các đặc trưng số: Thay thế bằng giá trị -999 (một giá trị đặc biệt không xuất hiện trong dữ liệu gốc)
- Đối với các đặc trưng phân loại: Thay thế bằng chuỗi 'missing'
- Tạo các đặc trưng đánh dấu giá trị thiếu để mô hình có thể học từ mẫu của dữ liệu thiếu

4.1.2 Mã hóa đặc trưng

Các đặc trưng phân loại được mã hóa bằng Label Encoding để chuyển đổi thành dạng số mà các thuật toán học máy có thể xử lý:

- Sử dụng `LabelEncoder` từ scikit-learn để chuyển đổi các giá trị phân loại thành các số nguyên
- Áp dụng Frequency Encoding cho các đặc trưng phân loại có nhiều giá trị duy nhất

4.2 Kỹ thuật đặc trưng

4.2.1 Tạo định danh khách hàng (UID)

Một trong những kỹ thuật quan trọng nhất là tạo định danh khách hàng duy nhất (UID) bằng cách kết hợp các đặc trưng liên quan đến thẻ và địa chỉ:

- `uid1 = card1 + addr1`: Kết hợp số thẻ và địa chỉ chính
- `uid2 = card1 + addr1 + addr2`: Kết hợp thêm địa chỉ phụ
- `uid3 = card1 + card2 + card3`: Kết hợp các thông tin thẻ
- `uid4 = card1 + card2 + addr1`: Kết hợp thông tin thẻ và địa chỉ
- `uid5 = P_emaildomain + card1 + addr1`: Kết hợp domain email, thẻ và địa chỉ

4.2.2 Đặc trưng tổng hợp

Dựa trên các UID đã tạo, chúng tôi tính toán các đặc trưng tổng hợp để nắm bắt hành vi của khách hàng:

- `uid_count`: Số lượng giao dịch của mỗi UID
- `uid_amt_mean/std/max/min`: Các thống kê về số tiền giao dịch của mỗi UID
- `uid_hour_nunique`: Số lượng giờ khác nhau mà UID thực hiện giao dịch
- `uid_day_nunique`: Số lượng ngày khác nhau mà UID thực hiện giao dịch

4.2.3 Đặc trưng thời gian

Chúng tôi chuyển đổi `TransactionDT` (thời gian giao dịch tính bằng giây) thành các đặc trưng thời gian có ý nghĩa:

- `DT_hour`: Giờ trong ngày (0-23)
- `DT_day`: Ngày trong tuần (0-6)
- `DT_M`: Tháng (1-12)
- `DT_W`: Tuần trong năm
- `is_weekend`: Đánh dấu giao dịch vào cuối tuần
- `is_night`: Đánh dấu giao dịch vào ban đêm

Ngoài ra, chúng tôi còn tạo các đặc trưng chu kỳ thời gian bằng cách áp dụng hàm sin và cos:

- `hour_sin/cos`: Biểu diễn chu kỳ của giờ trong ngày
- `day_sin/cos`: Biểu diễn chu kỳ của ngày trong tuần
- `month_sin/cos`: Biểu diễn chu kỳ của tháng trong năm

4.2.4 Đặc trưng TransactionAmt

Chúng tôi tạo các đặc trưng bổ sung từ số tiền giao dịch:

- `TransactionAmt_decimal`: Phần thập phân của số tiền
- `TransactionAmt_decimal_len`: Độ dài của phần thập phân
- `TransactionAmt_round/round_10/round_100`: Các giá trị làm tròn
- `TransactionAmt_outlier`: Đánh dấu các giao dịch có số tiền bất thường

4.2.5 Đặc trưng tương tác

Chúng tôi tạo các đặc trưng tương tác giữa số tiền giao dịch và các đặc trưng khác:

- `TransactionAmt_to_mean_card1`: Tỷ lệ giữa số tiền giao dịch và số tiền trung bình của card1
- `TransactionAmt_to_std_card1`: Tỷ lệ giữa số tiền giao dịch và độ lệch chuẩn của card1
- `TransactionAmt_to_mean_addr1`: Tỷ lệ giữa số tiền giao dịch và số tiền trung bình của addr1

4.3 Chiến lược validation

Chúng tôi đã thử nghiệm ba chiến lược validation khác nhau:

1. **Train-Test Split đơn giản**: Chia dữ liệu thành 80% huấn luyện và 20% validation
2. **StratifiedKFold**: Chia dữ liệu thành 5 fold, đảm bảo tỷ lệ gian lận giống nhau trong mỗi fold
3. **Time-Based Cross-Validation**: Chia dữ liệu theo thời gian để mô phỏng tình huống thực tế

Time-Based Cross-Validation cho kết quả tốt nhất vì nó mô phỏng chính xác hơn cách mô hình sẽ được sử dụng trong thực tế, khi dự đoán các giao dịch mới dựa trên dữ liệu lịch sử.

4.4 Huấn luyện mô hình

Chúng tôi huấn luyện ba mô hình Gradient Boosting: LightGBM, XGBoost và CatBoost với các tham số được tinh chỉnh:

4.4.1 LightGBM

```
lgb_params = {
    "objective": "binary",
    "boosting_type": "gbdt",
    "metric": "auc",
    "n_jobs": -1,
    "learning_rate": 0.01,
    "num_leaves": 256,
    "max_depth": 8,
    "tree_learner": "serial",
    "colsample_bytree": 0.7,
    "subsample": 0.7,
    "min_child_weight": 1,
    "min_child_samples": 20,
    "scale_pos_weight": 1,
    "reg_alpha": 0.1,
```

```
    "reg_lambda": 0.1,
    "verbose": -1,
    "seed": 42
}
```

4.4.2 XGBoost

```
xgb_params = {
    "objective": "binary:logistic",
    "eval_metric": "auc",
    "eta": 0.02,
    "max_depth": 8,
    "min_child_weight": 1,
    "subsample": 0.7,
    "colsample_bytree": 0.7,
    "alpha": 0.1,
    "lambda": 0.1,
    "tree_method": "hist",
    "nthread": -1,
    "scale_pos_weight": 1,
    "seed": 42
}
```

4.4.3 CatBoost

```
cat_params = {
    "iterations": 10000,
    "learning_rate": 0.02,
    "depth": 8,
    "l2_leaf_reg": 10,
    "bootstrap_type": "Bernoulli",
    "subsample": 0.7,
    "scale_pos_weight": 1,
    "eval_metric": "AUC",
    "metric_period": 100,
    "od_type": "Iter",
    "od_wait": 200,
    "random_seed": 42,
    "allow_writing_files": False,
    "task_type": "GPU",
    "verbose": 100
}
```

4.5 Kết hợp mô hình

Chúng tôi đã thử nghiệm hai phương pháp kết hợp mô hình:

1. **Blending**: Kết hợp dự đoán từ ba mô hình với các trọng số khác nhau

2. **Stacking**: Sử dụng LogisticRegression làm meta-learner trên dự đoán OOF (Out-of-Fold) của ba mô hình

Chúng tôi đã thử nghiệm nhiều trọng số khác nhau và chọn trọng số tốt nhất dựa trên AUC trên tập validation.

5 Kết quả thực nghiệm

5.1 Quá trình cải tiến mô hình

Chúng tôi đã thực hiện ba phiên bản mô hình với các cải tiến liên tục:

5.1.1 Phiên bản 1: Kết hợp mô hình đơn giản

Phiên bản đầu tiên sử dụng train-test split đơn giản và kết hợp ba mô hình với trọng số được dò tìm thủ công:

- Trọng số: LightGBM (0.1), XGBoost (0.8), CatBoost (0.1)
- AUC trên Private test: 0.85402
- AUC trên Public test: 0.890363

5.1.2 Phiên bản 2: Sử dụng StratifiedKFold

Phiên bản thứ hai cải thiện chiến lược validation bằng cách sử dụng StratifiedKFold:

- Trọng số: LightGBM (0.1), XGBoost (0.6), CatBoost (0.3)
- AUC trên Private test: 0.890895
- AUC trên Public test: 0.916814

5.1.3 Phiên bản 3: Time-Based Cross-Validation và Stacking

Phiên bản cuối cùng sử dụng Time-Based Cross-Validation và thử nghiệm cả Stacking và Blending:

- Trọng số tốt nhất: LightGBM (0.2), XGBoost (0.6), CatBoost (0.2)
- AUC trên Private test: 0.900404
- AUC trên Public test: 0.925547

5.2 So sánh hiệu suất

Bảng dưới đây tóm tắt hiệu suất của ba phiên bản mô hình:

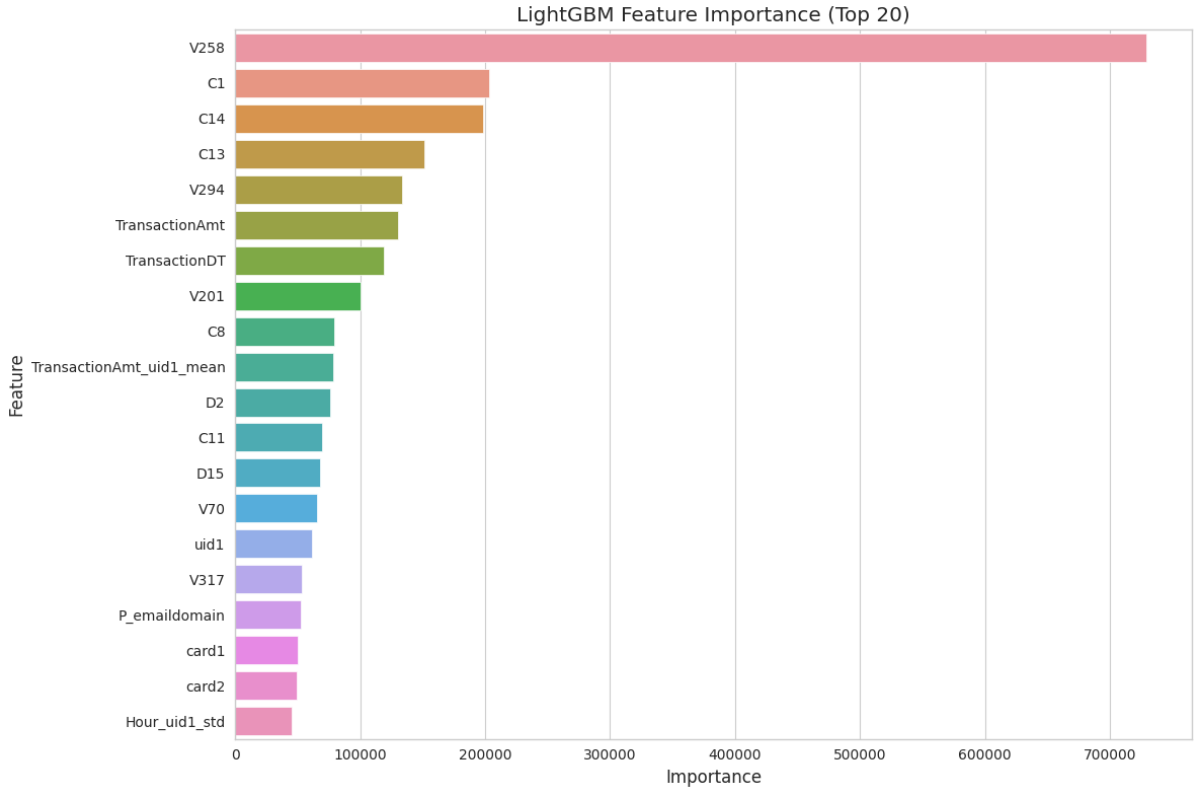
Phiên bản	**Chiến lược validation**	**Private AUC**	**Public AUC**
Phiên bản 1	Train-Test Split	0.85402	0.890363
Phiên bản 2	StratifiedKFold	0.890895	0.916814
Phiên bản 3	Time-Based CV	**0.900404**	**0.925547**

Bảng 1: So sánh hiệu suất của các phiên bản mô hình

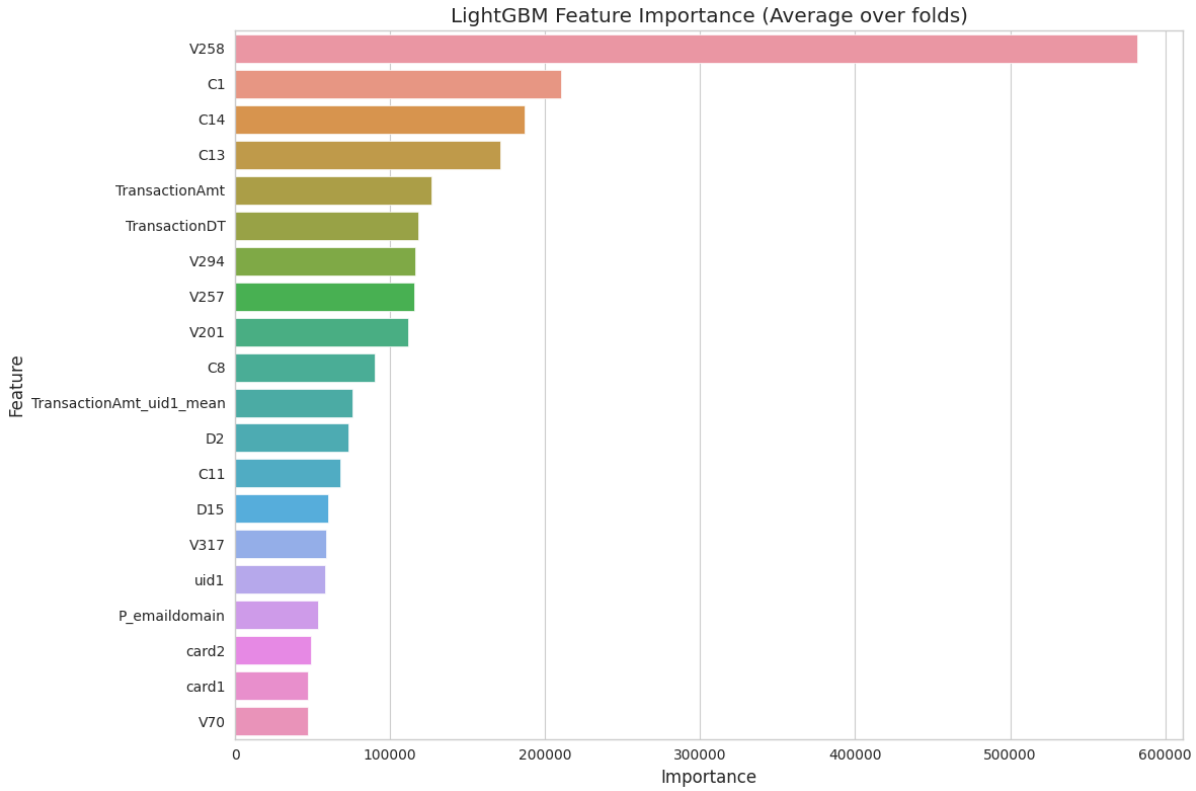
5.3 Phân tích đặc trưng quan trọng

Chúng tôi phân tích đặc trưng quan trọng từ cả ba mô hình để hiểu rõ hơn về các yếu tố ảnh hưởng đến việc phát hiện gian lận:

5.3.1 LightGBM Feature Importance

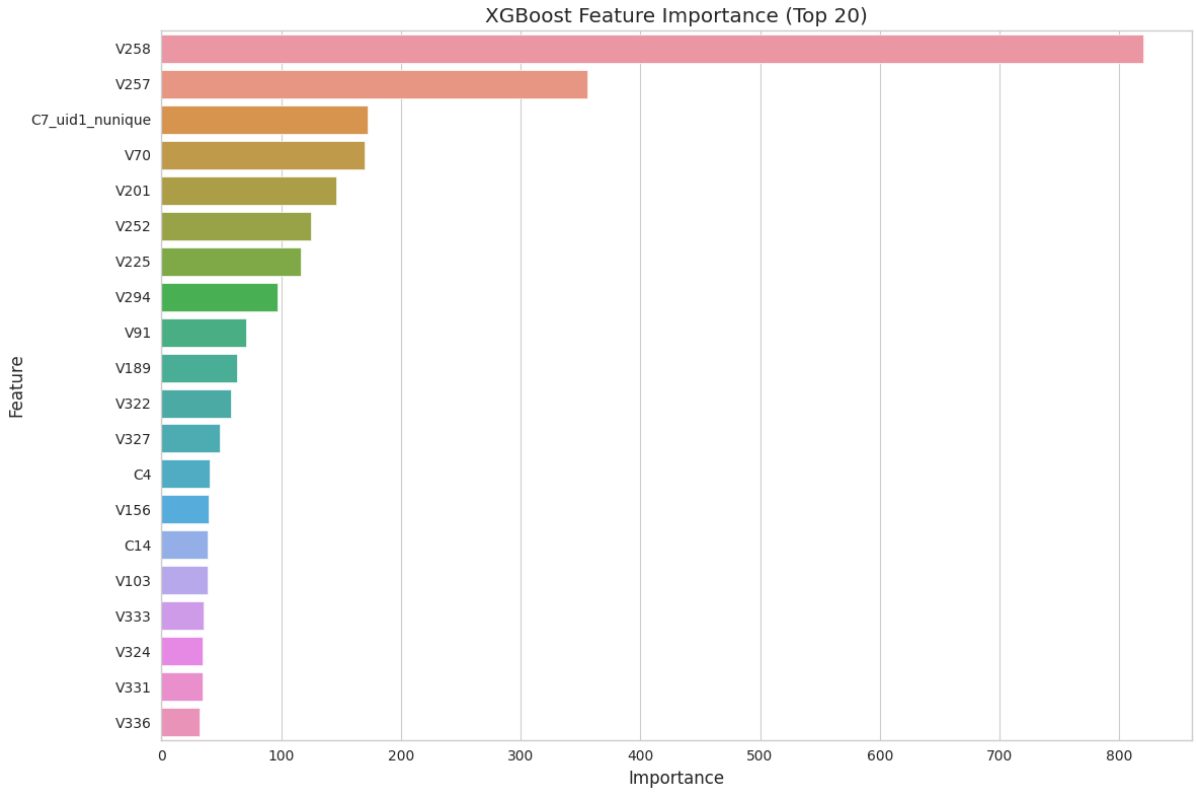


Hình 6: LightGBM Feature Importance (Top 20)

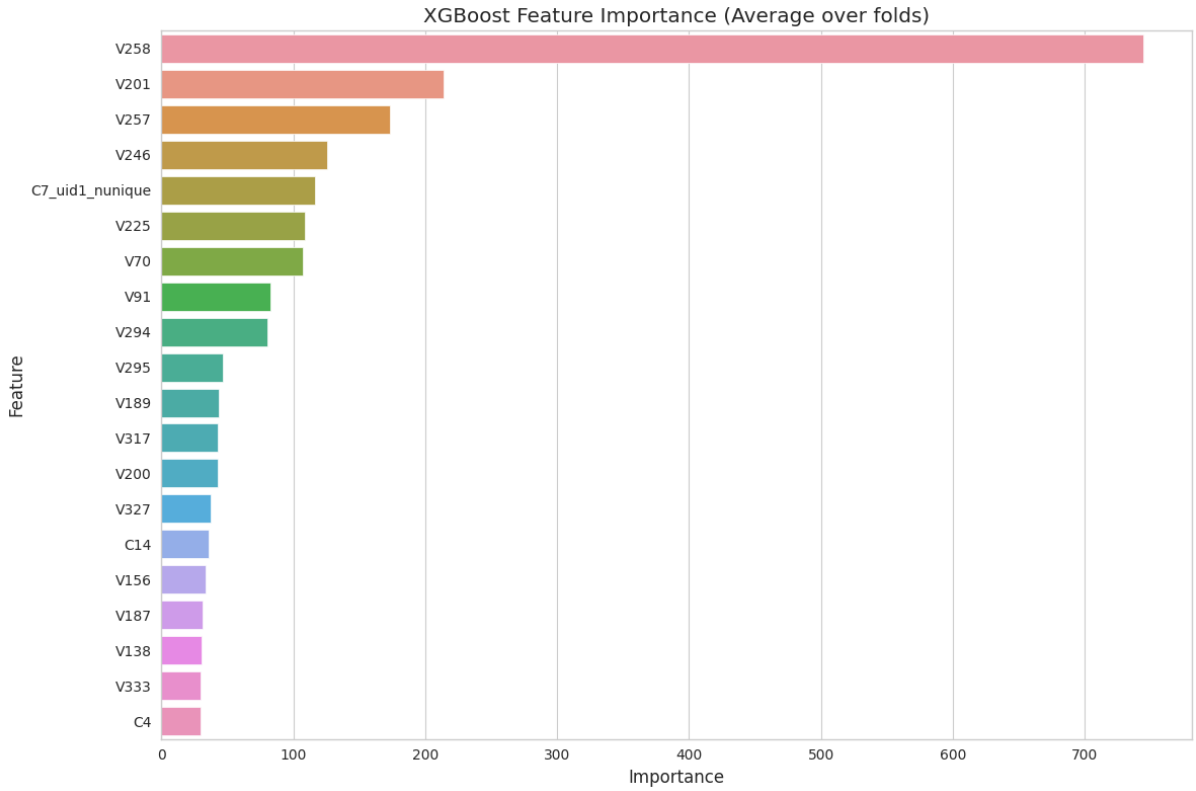


Hình 7: LightGBM Feature Importance (Average over folds)

5.3.2 XGBoost Feature Importance

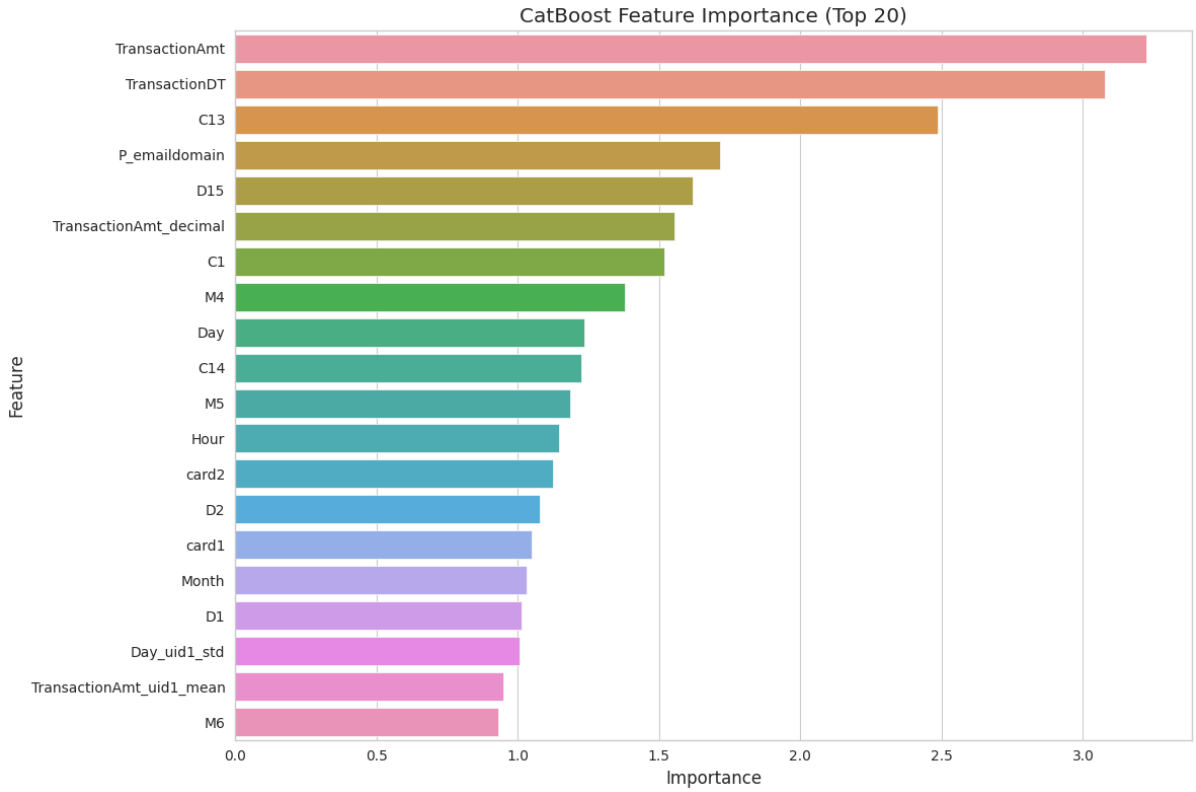


Hình 8: XGBoost Feature Importance (Top 20)

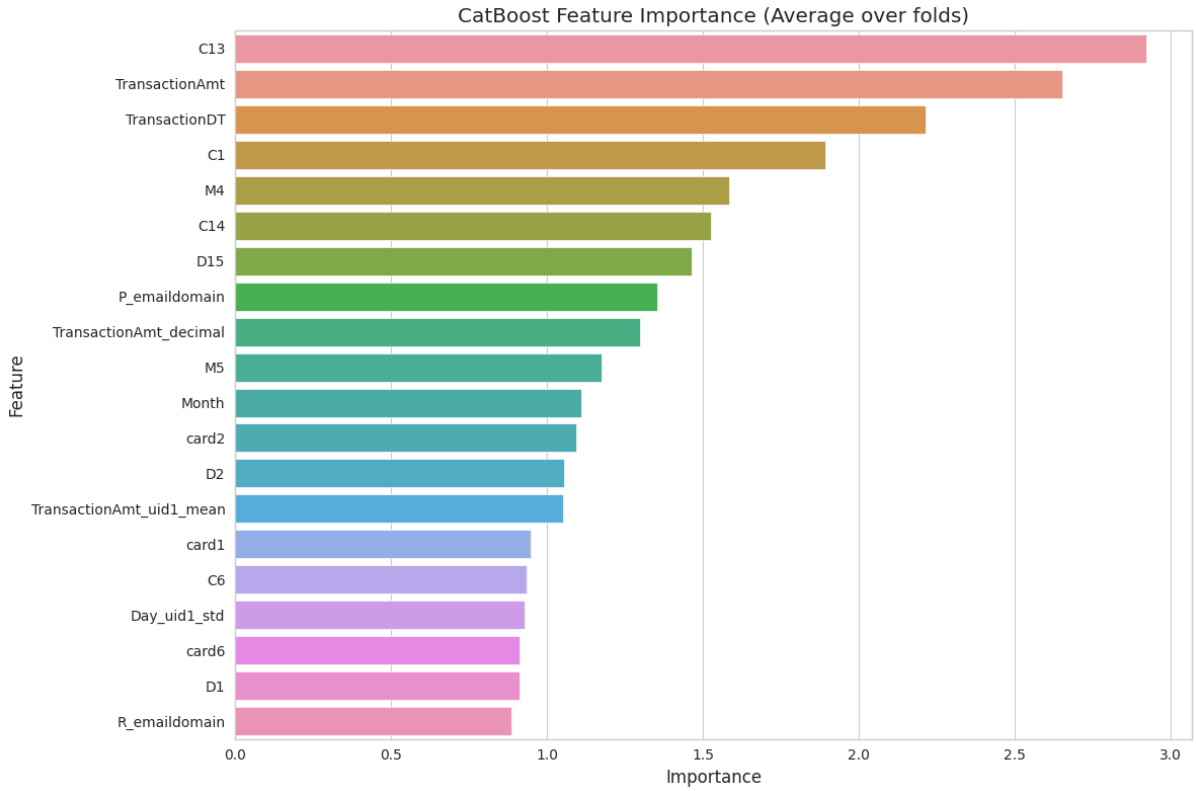


Hình 9: XGBoost Feature Importance (Average over folds)

5.3.3 CatBoost Feature Importance



Hình 10: CatBoost Feature Importance (Top 20)



Hình 11: CatBoost Feature Importance (Average over folds)

Từ các biểu đồ trên, chúng tôi có thể thấy một số đặc trưng quan trọng nhất trong việc phát hiện gian lận:

- **V258**: Một đặc trưng ẩn danh, nhưng có tầm quan trọng cao nhất trong cả LightGBM và XGBoost
- **TransactionAmt và TransactionDT**: Số tiền và thời gian giao dịch là những đặc trưng quan trọng trong CatBoost
- **C1, C13, C14**: Các đặc trưng liên quan đến thẻ có tầm quan trọng cao trong nhiều mô hình
- **uid1_nunique**: Số lượng giá trị duy nhất của uid1, phản ánh tầm quan trọng của việc phân loại theo khách hàng
- **TransactionAmt_decimal**: Phần thập phân của số tiền giao dịch, một đặc trưng quan trọng trong CatBoost

6 Thảo luận

6.1 Phân tích kết quả

Kết quả thực nghiệm cho thấy một số điểm quan trọng:

- **Chiến lược validation**: Time-Based Cross-Validation cho kết quả tốt nhất, phản ánh tầm quan trọng của việc mô phỏng chính xác cách mô hình sẽ được sử dụng trong thực tế.

- **Kết hợp mô hình**: Kết hợp ba mô hình Gradient Boosting với trọng số phù hợp cải thiện đáng kể hiệu suất so với từng mô hình riêng lẻ.
- **Kỹ thuật đặc trưng**: Các đặc trưng tổng hợp theo UID và đặc trưng thời gian đóng vai trò quan trọng trong việc cải thiện hiệu suất mô hình.
- **Cải tiến liên tục**: Quá trình cải tiến từ Phiên bản 1 đến Phiên bản 3 cho thấy tầm quan trọng của việc thử nghiệm và tinh chỉnh liên tục.

6.2 So sánh với các giải pháp hàng đầu

So với giải pháp đạt hạng nhất trong cuộc thi, mô hình của chúng tôi có một số điểm tương đồng và khác biệt:

- **Tương đồng**: Cả hai đều sử dụng kết hợp nhiều mô hình Gradient Boosting và tập trung vào việc tạo đặc trưng theo khách hàng.
- **Khác biệt**: Giải pháp hạng nhất sử dụng hậu xử lý phức tạp hơn, thay thế dự đoán của từng giao dịch bằng dự đoán trung bình của khách hàng. Trong khi đó, chúng tôi tập trung vào việc cải thiện chiến lược validation và tạo đặc trưng thời gian nâng cao.

6.3 Hạn chế và hướng cải thiện

Mặc dù đạt được kết quả tốt, mô hình của chúng tôi vẫn có một số hạn chế:

- **Tối ưu hóa tham số**: Chúng tôi chưa sử dụng các công cụ tối ưu hóa tham số tự động như Optuna để tìm tham số tối ưu cho từng mô hình.
- **Hậu xử lý**: Chúng tôi chưa áp dụng đầy đủ các kỹ thuật hậu xử lý như trong giải pháp hạng nhất.
- **Đặc trưng nâng cao**: Có thể tạo thêm các đặc trưng phức tạp hơn, như đặc trưng dựa trên đồ thị giao dịch hoặc phân tích chuỗi thời gian.

Các hướng cải thiện trong tương lai bao gồm:

- Sử dụng Optuna để tối ưu hóa tham số tự động
- Áp dụng các kỹ thuật hậu xử lý nâng cao
- Thử nghiệm các mô hình học sâu như Neural Networks
- Tạo thêm các đặc trưng dựa trên phân tích chuỗi thời gian và đồ thị giao dịch

7 Kết luận

Trong dự án này, chúng tôi đã phát triển một giải pháp phát hiện gian lận thẻ tín dụng dựa trên dữ liệu từ cuộc thi IEEE-CIS Fraud Detection. Chúng tôi đã áp dụng các kỹ thuật tiên tiến trong xử lý dữ liệu, tạo đặc trưng, và kết hợp mô hình để đạt được hiệu suất cao.

Quá trình cải tiến liên tục từ Phiên bản 1 đến Phiên bản 3 đã cải thiện đáng kể hiệu suất mô hình, với AUC trên Private test tăng từ 0.85402 lên 0.900404. Điều này cho thấy tầm quan trọng của việc lựa chọn chiến lược validation phù hợp, tạo đặc trưng hiệu quả, và kết hợp nhiều mô hình.

Các kết quả và phân tích trong dự án này không chỉ có giá trị trong việc giải quyết bài toán phát hiện gian lận thẻ tín dụng, mà còn cung cấp những hiểu biết sâu sắc về việc áp dụng các phương pháp thống kê và học máy trong các bài toán phân loại không cân bằng trong thực tế.

8 Tài liệu tham khảo

1. Giải pháp hạng nhất trong cuộc thi IEEE-CIS Fraud Detection. https://www.kaggle.com/competitions/ieee-fraud-detection/discussion/111308

2. Giải pháp của thí sinh đạt điểm số ấn tượng.

   https://github.com/shejz/IEEE-CIS-Fraud-Detection/tree/master

3. Hướng dẫn chi tiết về IEEE-CIS Fraud Detection. https://towardsdatascience.com/ieee-cis-fraud-detection-top-5-solution-5488fc66e95f

4. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T. Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. Advances in neural information processing systems, 30.

5. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794).

6. Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: unbiased boosting with categorical features. Advances in neural information processing systems, 31.

7. Dal Pozzolo, A., Caelen, O., Johnson, R. A., & Bontempi, G. (2015). Calibrating probability with undersampling for unbalanced classification. In 2015 IEEE Symposium Series on Computational Intelligence (pp. 159-166).

8. Breiman, L. (1996). Stacked regressions. Machine learning, 24(1), 49-64.

You also can see my code and my submission file of the competition on Github