Final Project Report

Topic: Financial Transactions Fraud Detection

Students: Nguyen Phu Loc – 22022547

Nguyen Duy Minh Lam – 22022605

Bui Ngoc Khanh – 22022551

I. Introduction, Goals

1. Introduction

- Hiện nay, lừa đảo tài chính thông qua các giao dịch thẻ tín dụng đang ngày càng tăng. Theo số liệu, trong năm 2021, có 389,845 báo cáo về gian lận thẻ tín dụng tại Hoa Kỳ với Ủy ban Thương mại Liên bang báo cáo rằng đó là loại gian lận danh tính phổ biến nhất ảnh hưởng đến những người trong độ tuổi từ 20-39, gây tổn thất hàng tỉ đô la mỗi năm

2. Goals

- Phát triển một hệ thống học máy phát hiện và dự báo rủi do lừa đảo của các giao dịch tài chính dựa trên các thông số nhất định. Qua đó giúp các cơ quan tài chính giảm thiểu rủi ro cũng như tránh mất mát một số tiền lớn

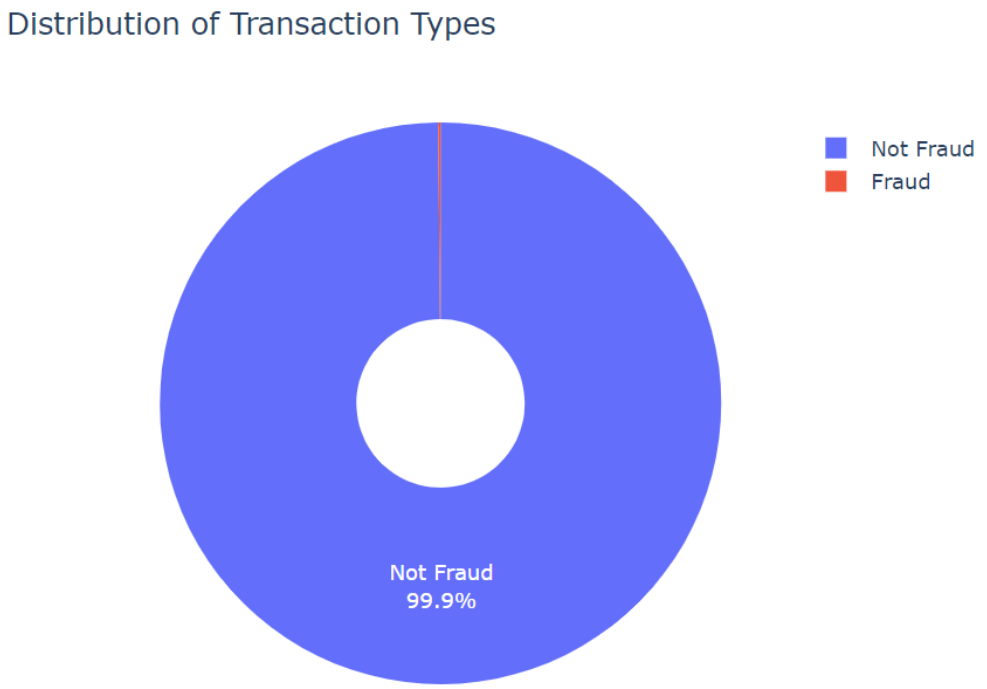
II. Data

- Bộ dữ liệu được sử dụng là Synthetic Financial Dataset for Fraud Detection
- Dữ liệu gồm 6,362,620 điểm dữ liệu
- Các đặc điểm của bộ dữ liệu gồm: Time, Type of Transaction, Amount, Origin, Destination Bank Account, Flagged fraud or not

III. Data Overview

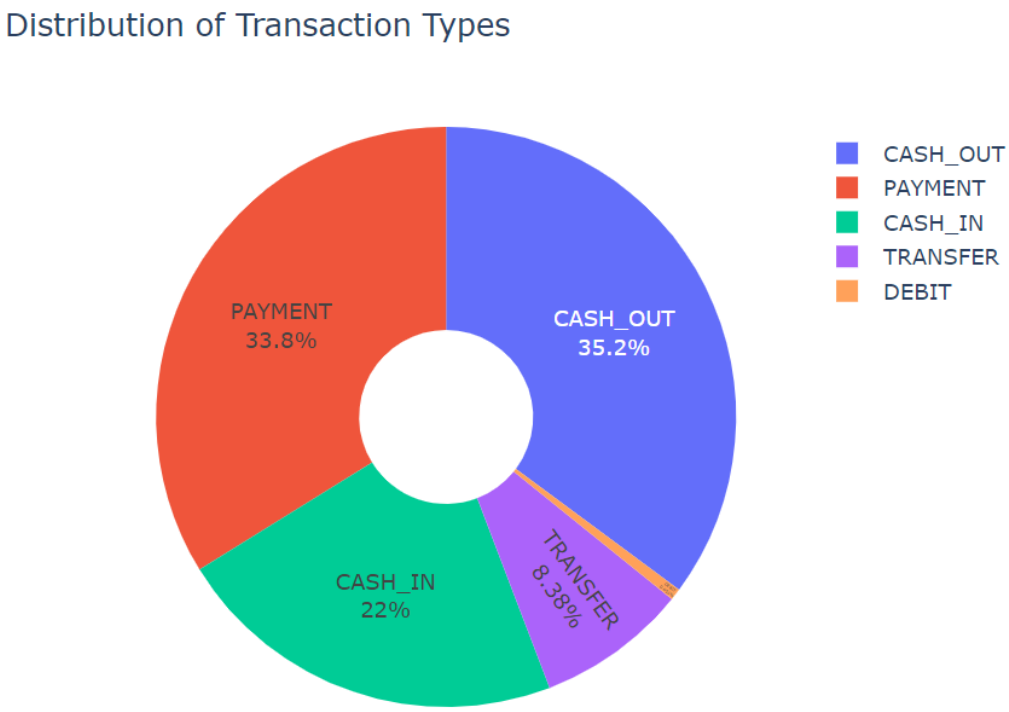
1. Tỉ lệ nhãn Fraud và Not Fraud của Transactions trong bộ dữ liệu

- Dựa vào biểu đồ, chúng ta có thể thấy rõ gần như toàn bộ dữ liệu là Not Fraud Transactions, chiếm tới 99,9%. Đây là một tỉ lệ quá chênh lệch giữa 2 nhãn, có thể đánh giá đây là một bộ dữ liệu không cân bằng (imbalanced)



2. Tỉ lệ các loại giao dịch

- Bộ dữ liệu ghi nhận 5 hình thức giao dịch bao gồm: PAYMENT, CASH_OUT, CASH_IN, TRANSFER VÀ DEBIT

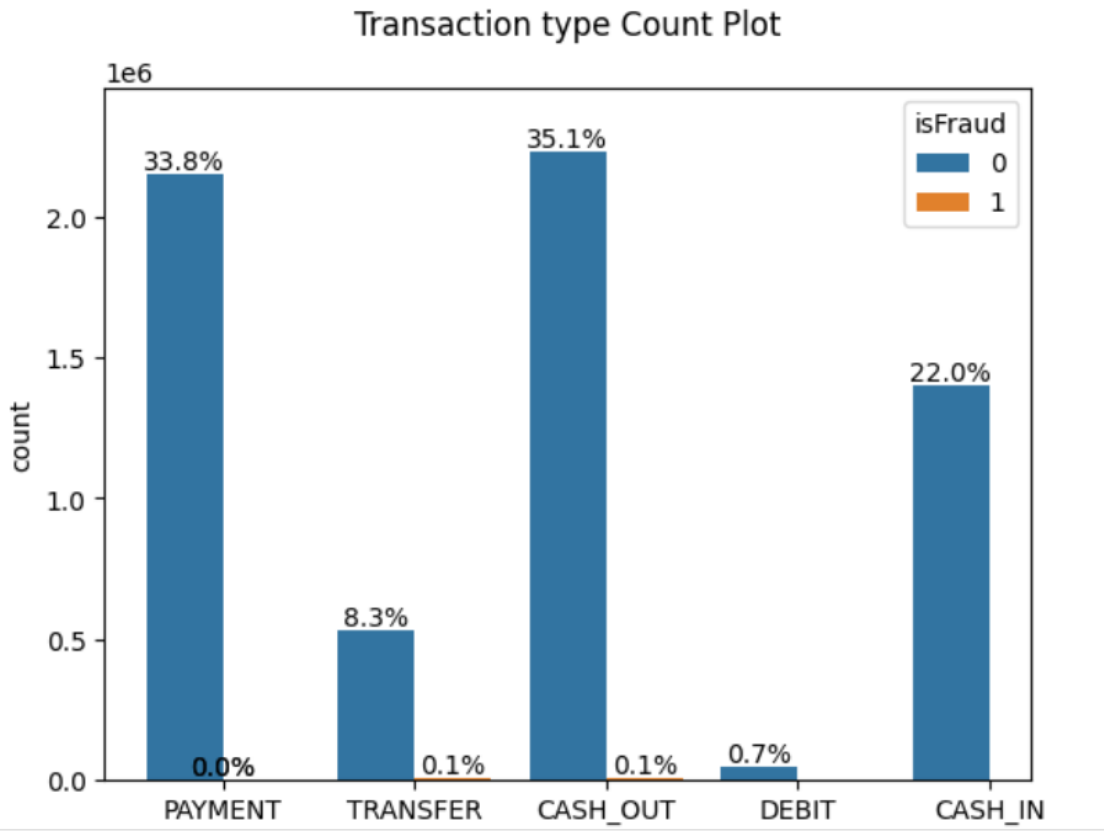


- CASH_OUT và PAYMENT là 2 loại giao dịch chính trong bộ dữ liệu với tỉ lệ lần lượt là 35,2% và 33,8%. Tiếp đến là hình thức CASH_IN với 22% và TRANSFER

với 8.38%. Trong khi đó DEBIT chỉ chiếm một con số không đáng kể trong bộ dữ liệu

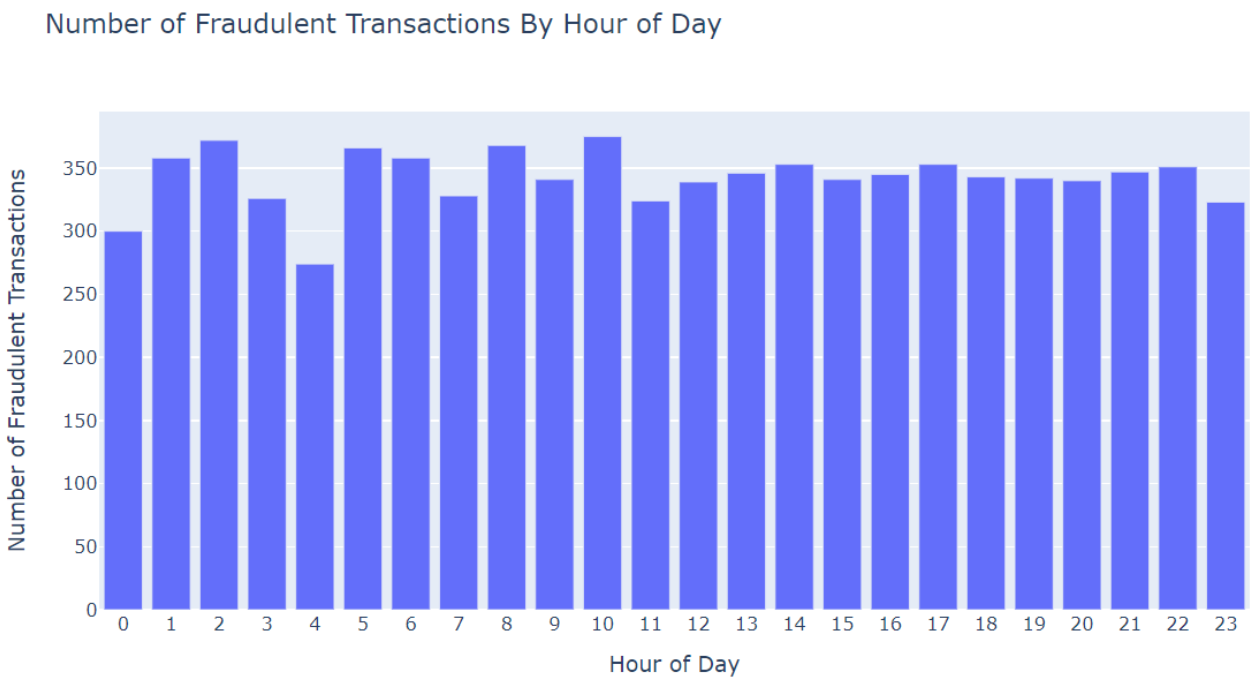
3. Tỉ lệ giao dịch Fraud trên từng hình hình thức

- Bộ dữ liệu có tới 5 hình thức giao dịch, tuy nhiên chỉ ghi nhận giao dịch Fraud ở 2 hình thức đó là TRANSFER và CASH_OUT
- Số lượng giao dịch Fraud ở hình thức TRANSFER là 4097 và CASH_OUT là 4116



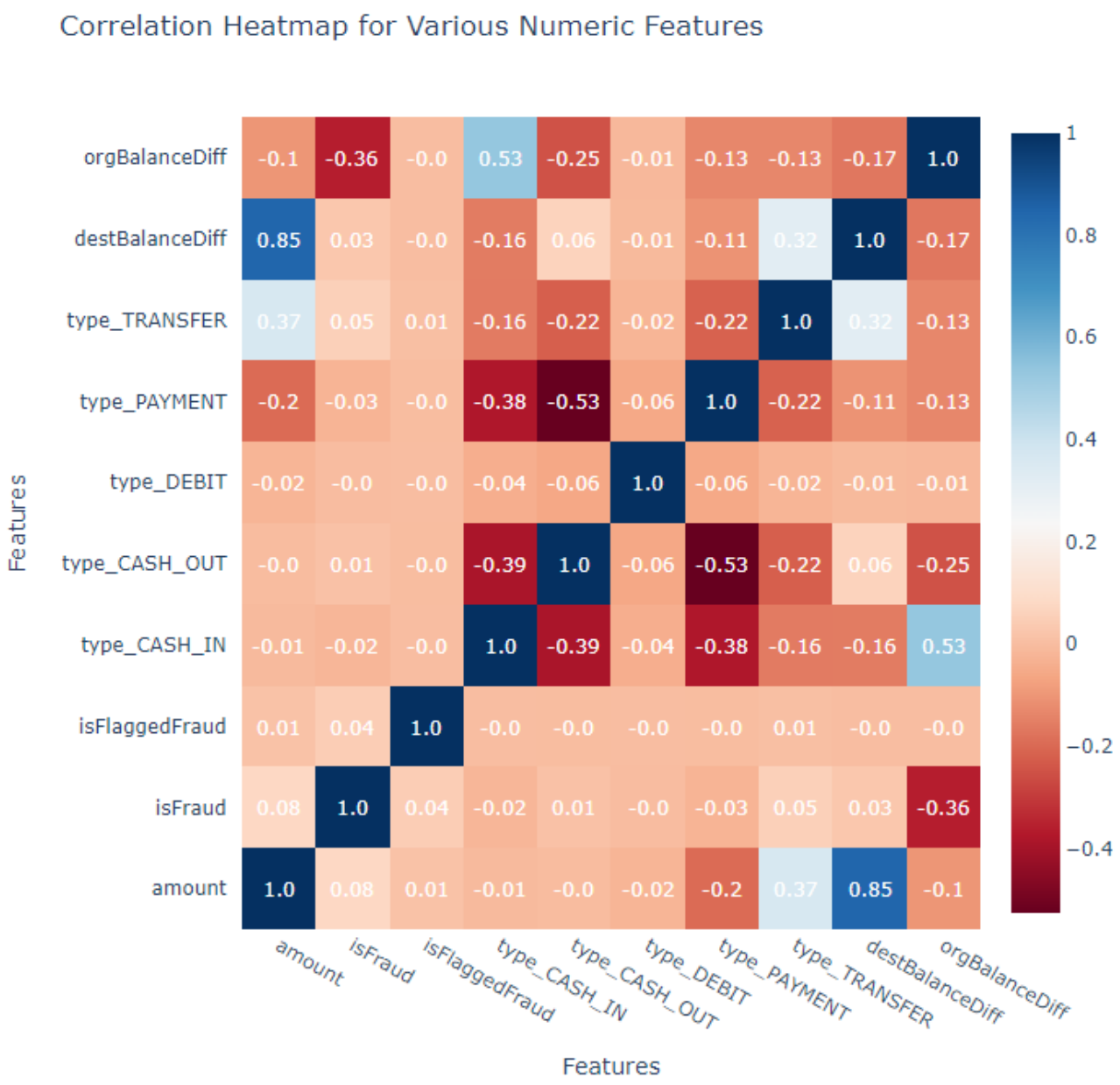
4. Thời gian của các giao dịch Fraud

- Có thể thấy các giao dịch Fraud được thực hiện xuyên suốt trong các khung giờ của một ngày, trung bình mỗi giờ có 342 giao dịch Fraud được thực hiện
- Khung giờ có số lượng giao dịch Fraud thấp nhất là 4 - 5h mới 274, trong khi khung giờ cao nhất là 10-11h với 375 giao dịch



IV. Features Engineering

- Chúng tôi sử dụng one-hot encoding cho Transactions types, loại bỏ một số features có mức độ tương quan lớn bao gồm newBalanceOrig, oldBalanceOrig, newBalanceDest, oldbalanceDest. Ngoài ra, kết hợp số dư tài khoản ban đầu và cuối cùng để tạo ra features mới là origBalanceDiff và destBalanceDiff. Qua một số phương pháp Feature Engineering, các đặc điểm đã giảm đi sự tương quan với nhau đáng kể, đem lại tính khách quan hơn cho bộ dữ liệu.



V. Baseline Performance

Chúng tôi sử dụng mô hình Logistic Regression (đơn giản, dễ triển khai) làm baseline model cho bài toán Fraud Detection. Đồng thời đưa ra một số metrics bao gồm True Positive Rate, False Negative Rate, False Positive Rate, False Negative Rate, Accuracy và F1 Score để lựa chọn đựa metric đánh giá phù hợp nhất cho mô hình

- Mô hình Logistic Regression

```
from sklearn.linear_model import LogisticRegression

features = df_encoded.drop(columns = ['isFraud', 'isFlaggedFraud'])
labels = df_encoded['isFraud']
amount = df_encoded['amount']
```

```
seed = 42
X_train, X_val, y_train, y_val = train_test_split(
    features, labels, test_size=0.2, random_state=seed
)
```

```
LRclf = LogisticRegression(penalty = None, multi_class='multinomial').fit(X_train, y_train)
y_train_pred = LRclf.predict(X_train)
y_pred = LRclf.predict(X_val)
y_pred
```

```
Train Results:  0.9979959120613835
True Positive Rate: 0.0006172839506172839
False Positive Rate: 0.0007160257580430937
True Negative Rate: 0.9992839742419569
False Negative Rate: 0.9993827160493827
Accuracy : 0.9980126111570391
F1 Score : 0.0007902015013828526
```

- Đánh giá: mô hình cho kết quả với độ chính xác gần như tuyệt đối do dữ liệu giao dịch Non-Fraud có tỉ lệ chênh lệch rất lớn so với Fraud. Vì vậy metrics có vẻ tối ưu nhất đó là F1 Score. F1 Score chỉ đạt con số rất thấp với xấp xỉ 0.0008

VI. Handling inbalanced data

Sử dụng phương pháp RandomOverSampler để sinh ra các giao dịch Fraud cho đến khi số lượng giao dịch Fraud bằng ½ số lượng giao dịch Non Fraud

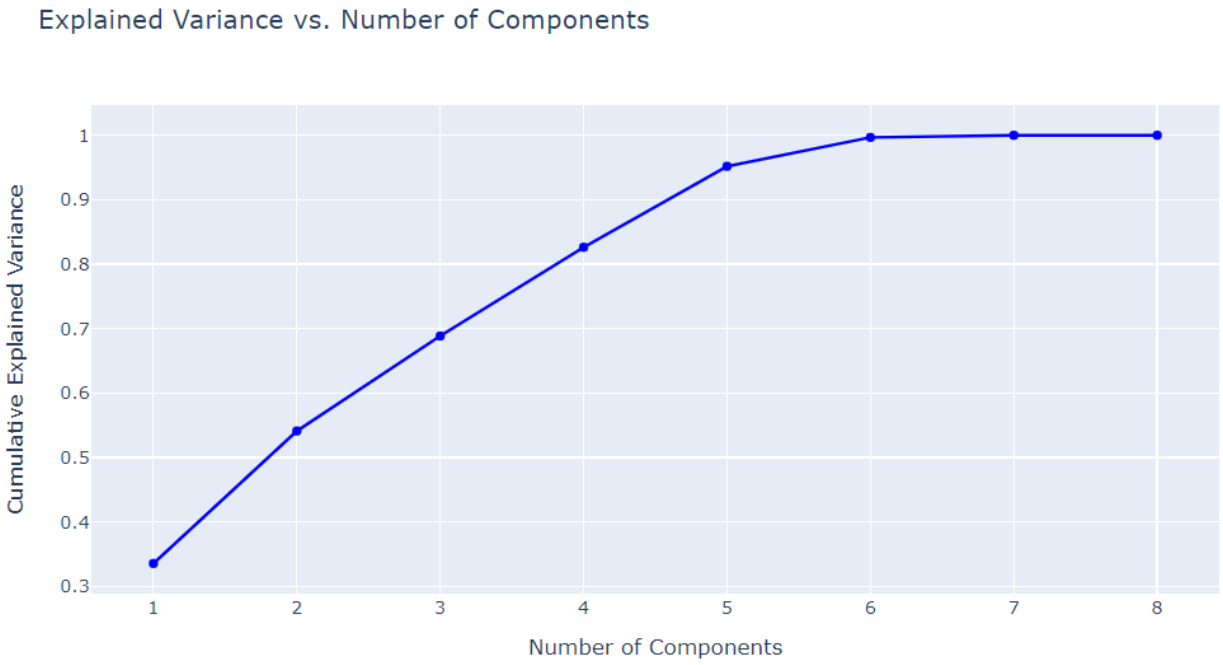
```
# Oversample
ros = RandomOverSampler(sampling_strategy=0.5, random_state=1)
X_oversample, y_oversample = ros.fit_resample(X, y)
```

Sử dụng phương pháp RandomUnderSampler để loại bỏ bớt các giao dịch Non Fraud để cân bằng số lượng giao dịch mỗi loại

```
# Undersample
rus = RandomUnderSampler(sampling_strategy=1.0, random_state=1)
X_balanced, y_balanced = rus.fit_resample(X_oversample, y_oversample)
```

VII. Principle Components Analysis

Chúng tôi chuẩn hóa dữ liệu bằng Standard Scaler và sau đó xác định variance dựa vào số lượng components. Và dựa vào biểu đồ, số components cần phải giữ để đạt được threshold 95% variance là 5.



VIII. Models

- Bộ dữ liệu được phân chia train/test set với tỉ lệ 0.85/0.15 do số lượng sample lớn nên test set không cần phải có quá nhiều dữ liệu
- Bên cạnh đó, dữ liệu cũng được chuẩn hóa, áp dụng feature engineering, pca, xử lí imbalance nên đủ điều kiện để có thể đưa vào huấn luyện cho mô hình
- Chúng tôi đề xuất 3 mô hình học máy bao gồm Logistic Regression, Random Forest và XGBoost và sử dụng các kĩ thuật để tối ưu hyperparmeters

1. Logistic Regression

- Sử dụng GridSearch để tìm kiếm Regularization types tối ưu nhất cho mô hình. Đối với bộ dữ liệu này, Logistic Regression không sử dụng Regularization types (penalty = None) là trường hợp cho kết quả tốt nhất

```
param_grid = {
  'penalty': ['l1', 'l2', 'none'],  # Regularization types
}

grid_search = GridSearchCV(estimator=lg_balanced, param_grid=param_grid, cv=5, n_jobs=-1, verbose=2)
```

2. **Random Forest**

- Tương tự, sử dụng GridSearch để tìm kiếm số estimators và max_depth tối ưu cho mô hình RandomForest. Trong trường hợp này, bộ siêu tham số tối ưu là n_estimator = 30 và max_depth = None

```
param_grid = {
    'n_estimators': [10, 20, 30],  # Number of trees in the forest
    'max_depth': [None, 5, 10],    # Maximum depth of the tree
}

# Instantiate the GridSearchCV object
grid_search = GridSearchCV(estimator=rf_classifier, param_grid=param_grid, cv=5, n_jobs=-1, verbose=2, scoring='f1')
```

3. **XGBoost**

- Sử dụng GridSearch cho XGBoost trên một param_grid nhỏ cho kết quả bộ siêu tham số tối ưu dựa trên F1 Score. Bộ siêu tham số tối ưu là max_depth = 7, learning_rate = 0.3, n_estimators = 70.

```
param_grid = {
    'max_depth': [3, 5, 7],
    'learning_rate': [0.05, 0.1, 0.2, 0.3],
    'n_estimators': [50, 70],
}
grid_search = GridSearchCV(estimator=xgb_clf, param_grid=param_grid, cv=2, scoring='f1', verbose=2)
```

- Đặc biệt đối với XGBoost, mô hình cho thấy kết quả với ROC AUC = 0.979

Bảng so sánh các thông số giữa các mô hình

Model	Logistic Regression	Random Forest	XGBoost
TPR	0.89	0.733	0.983
FPR	0.06	0.0007	0.04
TNR	0.93	0.999	0.96
FNR	0.101	0.267	0.01
Accuracy	0.935	0.9999	0.968
F1-Score	0.033	0.63	0.06

Nhận xét: Dựa vào kết quả của quá trình huấn luyện và kiểm thử trên các mô hình đã được đề xuất, Random Forest cho kết quả tốt nhất trên tập dữ liệu sau khi đã xử lí với các

thông số khá vượt trội so với mô hình còn lại. Tuy nhiên vẫn cần tối ưu hơn nữa để tăng True Positive Rate và F1 Score, đồng thời giảm False Negative Rate

IX. Interpretation

- Do tài nguyên tính toán còn hạn chế, lượng dữ liệu khá lớn, nên khi sử dụng GridSearch chưa tìm kiếm được các siêu tham số trong phạm vi giá trị rộng hơn nữa
- Trong quá trình cố gắng cân bằng lại bộ dữ liệu, phương pháp RandomOverSampler đã loại bỏ đi một phần lớn dữ liệu thực tế. Điều này giúp quá trình huấn luyện diễn ra tốt hơn, tuy nhiên lại đạt hiệu quả không được cao trên bộ dữ liệu thực tế
- Trên thực tế, phát hiện giả mạo đối với giao dịch có trị giá lớn sẽ quan trọng hơn nhiều so với nhiều giao dịch có giá trị nhỏ, nên cần xem xét thêm việc đặt trọng số.