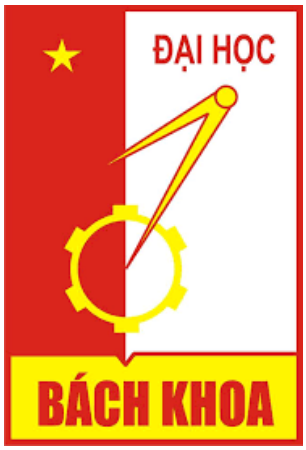
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC

---------- 🕮 ----------



BÁO CÁO MÔN HỌC: HỆ HỖ TRỢ QUYẾT ĐỊNH

Đề tài: Giải quyết vấn đề mất cân bằng dữ liệu cho bài toán phân loại khách hàng

Giảng viên hướng dẫn:	TS. Lê Hải Hà
Sinh viên thực hiện:	Đoàn Lê Tường Vy - 20196010
Lớp:	Hệ thống thông tin, 01 – K64

HÀ NỘI, 8/2022

LỜI CẢM ƠN

Đầu tiên, em xin gửi lời cảm ơn sâu sắc tới TS. Lê Hải Hà, người đã tận tình hướng dẫn, giúp đỡ em trong suốt 15 tuần học tập và thực hiện môn học Hệ hỗ trợ quyết định. Qua môn học này, em đã được trang bị những kiến thức về các hệ hỗ trợ ra quyết định quản lý, các kỹ thuật phân tích và khai phá dữ liệu kinh doanh, máy học và trí tuệ nhân tạo. Từ đó có thể vận dụng các kiến thức vào việc phát triển một số hệ hỗ trợ quyết định trong thực tế.

Do còn nhiều hạn chế về kiến thức nên nội dung trong bài báo cáo khó tránh khỏi những thiếu sót. Vì vậy em rất mong nhận được những nhận xét và ý kiến đóng góp của thầy để bài báo cáo được hoàn thiện hơn.

Em xin chân thành cảm ơn!

MỤC LỤC

MỞ ĐẦU

Trong lớp các mô hình học có giám sát (*supervised learning*) thì có hai bài toán chính là bài toán dự báo (đối với biến mục tiêu liên tục) và bài toán phân loại (đối với biến mục tiêu là rời rạc). Bài toán phân loại là lớp bài toán được ứng dụng phổ biến nhất trong machine learning. Có rất nhiều các tác vụ mà chúng ta có thể kể đến liên quan tới lớp bài toán phân loại:

- **Phân loại nợ xấu trong ngân hàng:** Danh mục *banking book* của ngân hàng luôn tồn tại những rủi ro. Các khoảng vay tiềm ẩn những nguy cơ vỡ nợ nên các ngân hàng cần phải đánh giá rủi ro đối với khách hàng của mình thông qua các mô hình phân loại nợ xấu để ra quyết định có *cho vay hay không* và vay với *hạn mức, kỳ hạn, lãi suất* là bao nhiêu? Sự khác biệt về sức khoẻ tín dụng giữa các ngân hàng được đánh giá trên khả năng kiểm soát nợ xấu. Để tạo ra một hệ thống quản trị rủi ro hiệu quả thì bên cạnh những biện pháp về phòng ngừa và tuân thủ, ngân hàng cần sử dụng các công cụ mô hình để lượng hoá rủi ro khách hàng.
- **Phát hiện đầu cơ và gian lận trong thương mại điện tử (TMĐT)**: Đầu cơ và gian lận là hiện tượng phổ biến trong lĩnh vực TMĐT. Đầu cơ là việc khách hàng mua vượt quá số lượng cho phép và bán ra thị trường với giá cao hơn nhằm hưởng lợi. Một số hành vi gian lận khác gây hậu quả nghiêm trọng cho sàn TMĐT đó là cửa hàng lợi dụng chính sách thưởng doanh số để nhận hoa hoa hồng, khách hàng gian lận để hưởng khuyến mãi,… Cả hai hành vi đầu cơ và gian lận gây đều gây thâm dụng vốn cho các sàn TMĐT. Xây dựng được một hệ thống phát hiện gian lận sẽ giúp sàn TMĐT hoạt động hiệu quả, tối ưu hoá được ngồn lực về vốn và mở rộng tập khách hàng để tạo ra những lợi thế cạnh tranh trên thị trường.

Có thể thấy rằng bài toán phân loại hiện tại đang giải quyết rất nhiều vấn đề mà con người đang phải đối mặt. Đồng thời với vai trò to lớn của mình, nó đã và đang mang lại nhiều thay đổi cho nhân loại. Với sự bùng nổ về thuật toán, năng lực tính toán và sự dồi dào của dữ liệu. Ngày càng có nhiều ứng dụng của mô hình phân loại đạt độ chính xác ở mức con người hay thậm chí là vượt trội và thay thế con người trong nhiều tác vụ khác nhau.

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

1.1. Mất cân bằng dữ liệu (Imbalanced dataset)

Mất cân bằng dữ liệu là một trong những hiện tượng phổ biến của bài toán phân loại nhị phân (binary classification) như spam email, phát hiện gian lận, dự báo vỡ nợ, chuẩn đoán bệnh lý,…. Trong trường hợp tỷ lệ dữ liệu giữa 2 classes là 50:50 thì được coi là cân bằng. Khi có sự khác biệt trong phân phối giữa 2 classes, chẳng hạn 60:40 thì dữ liệu có hiện tượng mất cân bằng.

Hầu hết các bộ dữ liệu đều khó đạt được trạng thái cân bằng mà luôn có sự khác biệt về tỷ lệ giữa 2 classes. Đối với những trường hợp dữ liệu mất cân bằng nhẹ như tỷ lệ 60:40 thì sẽ không ảnh hưởng đáng kể tới khả năng dự báo của mô hình.

Tuy nhiên nếu hiện tượng **mất cân bằng nghiêm trọng** xảy ra, chẳng hạn như tỷ lệ 90:10 sẽ thường dẫn tới ngộ nhận chất lượng mô hình. Khi đó thước đo đánh giá mô hình là *độ chính xác (accuracy)* có thể đạt được rất cao mà không cần tới mô hình. Ví dụ, một dự báo ngẫu nhiên đưa ra tất cả đều là nhóm đa số thì độ chính xác đã đạt được là 90%. Do đó không nên lựa chọn độ chính xác làm chỉ số đánh giá mô hình để tránh lạc quan sai lầm về chất lượng mô hình.

Trong trường hợp mẫu mất cân bằng nghiêm trọng ta cần phải thay đổi chỉ số đánh giá để đưa ra kết quả hợp lý hơn. Ngoài ra, mất cân bằng dữ liệu nghiêm trọng thường dẫn tới dự báo kém chính xác trên nhóm thiểu số. Bởi đa phần kết quả dự báo ra thường thiên về 1 nhóm là nhóm đa số và rất kém trên nhóm thiểu số, trong khi dự báo chính xác nhóm thiểu số quan trọng hơn nhiều so với dự báo nhóm đa số. Để cải thiện kết quả dự báo chúng ta cần những điều chỉnh thích hợp để mô hình đạt được một độ chính xác cao trên nhóm thiểu số.

1.2. Các phương pháp giải quyết dữ liệu mất cân bằng

1.2.1. Thay đổi metric:

Như đã giải thích ở mục đầu tiên, khi hiện tượng mất cân bằng dữ liệu nghiêm trọng xảy ra thì việc sử dụng độ chính xác làm thước đo đánh giá mô hình thường không hiệu quả bởi hầu hết chúng đều đạt độ chính xác rất cao. Một mô hình ngẫu nhiên dự báo toàn bộ là nhãn thuộc nhóm đa số cũng sẽ mang lại kết quả gần bằng 100%. Khi đó ta có thể cân nhắc tới một số metrics thay thế như *precision, recall, f1-score, gini,....* Các chỉ số này sẽ không quá lớn để dẫn tới ngộ nhận độ chính xác, đồng thời chúng tập trung hơn vào việc

đánh giá độ chính xác trên nhóm thiểu số, nhóm mà chúng ta muốn dự báo chính xác hơn so với nhóm đa số.

		Actual		
		Positive	**Negative**	
Predicted	**Positive**	True Positive (TP)	False Positive (FP) Type I error	Precision = TP/(TP+FP)
	Negative	False Negative (FN) Type II error	True Negative (TN)	
		Recall = TP/(TP+FN)	False positive rate (FPR) = FP/(FP+TN)	

Hình 1: Bảng cross table mô tả kết quả thống kê chéo giữa nhãn dự báo và ground truth.

Ở đây Positive tương ứng với nhãn 1 (khách hàng xấu) và Negative tương ứng với nhãn 0 (khách hàng tốt).

Từ bảng cross table ta dễ dàng hình dung được ý nghĩa của các chỉ số đó là:

- Precision: Mức độ **dự báo chính xác** trong những trường hợp được dự báo là khách hàng xấu (Positive).

$$precision = \frac{TP}{TP + FP}$$

- Recall: Mức độ **dự báo chính xác** những trường hợp là khách hàng xấu (Positive) trong những trường hợp thực tế là khách hàng xấu (Positive).

$$recall = \frac{TP}{TP + FN}$$

- f1-score: Trung bình kết hợp giữa Precision và Recall. Đây là chỉ số thay thế lý tưởng cho accuracy khi mô hình có tỷ lệ mất cân bằng mẫu cao.

$$f1 - score = \frac{2}{\frac{1}{precision} + \frac{1}{recall}}$$

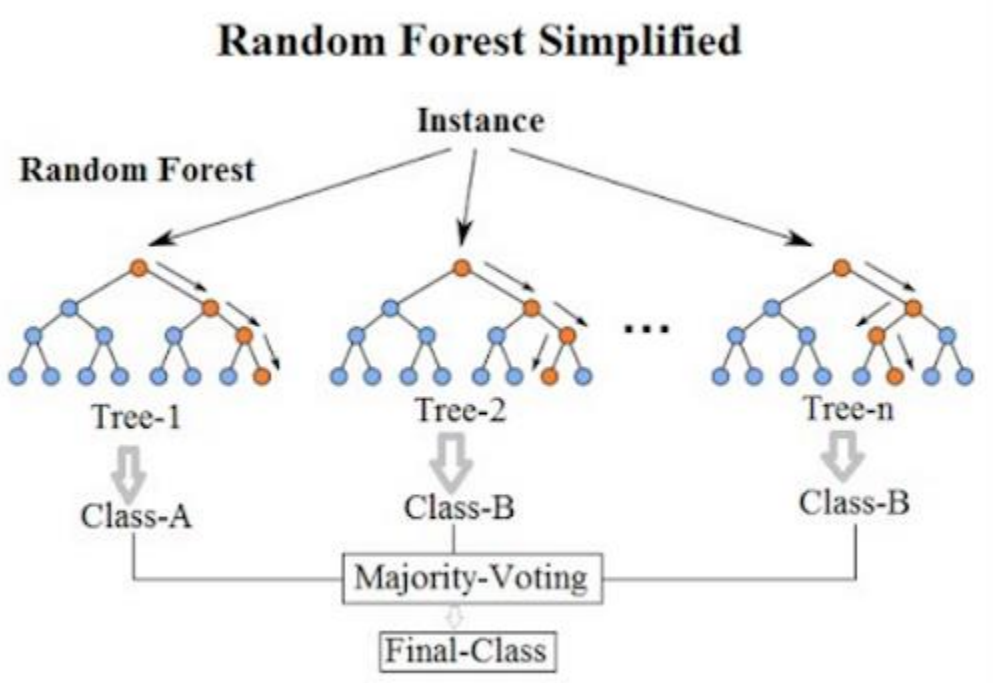
- Kappa-Score: Là chỉ số đo lường mức độ liên kết tin cậy (inter-rater reliability) cho các categories.
- Gini: Đo lường sự bất bình đẳng trong phân phối giữa Positive và Negative được dự báo từ mô hình.
- AUC: Biểu diễn mối quan hệ giữa độ nhạy (sensitivity) và độ đặc hiệu (specificity). Đánh giá khả năng phân loại tốt và xaaus được dự báo từ mô hình.

Một mô hình có các chỉ số trên đều cao thì mô hình có chất lượng dự báo càng tốt.

Trong bài báo cáo này, em sẽ sử dụng 2 chỉ số auc và f1-score là 2 thước đo chính đánh giá mô hình.

1.2.2. Xây dựng mô hình

a) Thuật toán Random forest:



Hình 3: Kiến trúc mô hình random forest.

Mô hình là một tập hợp của nhiều cây quyết định. Mỗi một cây quyết định sẽ trả ra một kết quả dự báo. Quyết định cuối cùng về nhãn của quan sát sẽ dựa trên nguyên tắc bầu cử đa số (Majority-Voting) trên toàn bộ các cây quyết định con. Ngoài ra mô hình cũng được chạy trên rất nhiều các sub-sample. Nếu một quan sát xuất hiện tại nhiều sub-

sample thì sẽ thực hiện bầu cử đa số trên tất cả các cây quyết định của toàn bộ các sub-sample.

Random Forest là thuật toán thuộc lớp mô hình kết hợp (ensemble model). Kết quả của thuật toán dựa trên bầu cư đa số từ nhiều cây quyết định. Do đó mô hình có độ tin cậy cao hơn và độ chính xác tốt hơn so với những mô hình phân loại tuyến tính đơn giản như logistic hoặc linear regression.

Tham số của Random Forest:

Để tunning mô hình random forest chúng ta dựa trên một số tham số chính:

1. **n_estimators**: Số lượng các trees trên một cây quyết định.
2. **max_depth**: Độ sâu lớn nhất của một cây quyết định.
3. **min_samples_split**: Số lượng mẫu tối thiểu cần thiết để phân chia một internal node. Nếu kích thước mẫu ở một internal node nhỏ hơn ngưỡng thì ta sẽ không rẽ nhánh internal node.
4. **max_features**: Số lượng các features được xem xét khi tìm kiếm phương án phân chia tốt nhất. Mặc định là toàn bộ các features đầu vào.
5. **class_weight**: Trọng số tương ứng với mỗi class. Mặc định là None, khi đó các class sẽ có mức độ quan trọng như nhau. Nếu lựa chọn balance các class sẽ được đánh trọng số tỷ lệ nghịch với tỷ trọng mẫu của chúng.
6. **min_impurity_split**: Ngưỡng để dừng sớm (early stopping) quá trình phát triển của cây quyết định. Nó sẽ tiếp tục phân chia nếu độ vẩn đục (impurity) lớn hơn ngưỡng threshold, trái lại thì nó là node leaf.

b) Under sampling

Under sampling là việc ta *giảm số lượng các quan sát của nhóm đa số* để nó trở nên cân bằng với số quan sát của nhóm thiểu số. Ưu điểm của under sampling là làm cân bằng mẫu một cách nhanh chóng, dễ dàng tiến hành thực hiện mà không cần đến thuật toán giả lập mẫu.

Tuy nhiên nhược điểm của nó là kích thước mẫu sẽ bị giảm đáng kể. Giả sử nhóm thiểu số có kích thước là 500, như vậy để tạo ra sự cân bằng mẫu giữa nhóm đa số và thiểu số sẽ cần giảm kích thước mẫu của nhóm đa số từ 10000 về 500. Tổng kích thước tập huấn luyện sau under sampling là 1000 và chiếm gần 1/10 kích thước tập huấn luyện ban đầu. Tập huấn luyện mới khá nhỏ, không đại diện cho phân phối của toàn bộ tập dữ liệu và thường dễ dẫn tới hiện tượng overfitting.

c) Over sampling

Over sampling là các phương pháp giúp giải quyết hiện tượng mất cân bằng mẫu bằng cách *gia tăng kích thước mẫu thuộc nhóm thiểu số* bằng các kĩ thuật khác nhau. Trong bài báo cáo này, em sẽ trình bày kỹ thuật SMOTE.

SMOTE (Synthetic Minority Over-sampling) là phương pháp sinh mẫu nhằm gia tăng kích thước mẫu của nhóm thiểu số trong trường hợp xảy ra mất cân bằng mẫu. Để gia tăng kích thước mẫu, với mỗi một mẫu thuộc nhóm thiểu số ta sẽ lựa chọn ra k mẫu láng giềng gần nhất với nó và sau đó thực hiện tổ hợp tuyến tính để tạo ra mẫu giả lập. Phương pháp để lựa chọn ra các láng giềng của một quan sát có thể dựa trên thuật toán kNN hoặc SVM.

CHƯƠNG 2: ỨNG DỤNG GIẢI QUYẾT VẤN ĐỀ MẤT CÂN BẰNG DỮ LIỆU CHO BÀI TOÁN PHÂN LOẠI KHÁCH HÀNG

2.1. Mô tả tập dữ liệu

Tiến hành xây dựng một mô hình trên bộ dữ liệu khách hàng vay tín dụng "xấu" (những khách hàng có khả năng cao nợ quá hạn) và thực hiện các đánh giá đo lường mức độ cải thiện mô hình trước và sau khi áp dụng các điều chỉnh.

2.1.1. Thông tin về bộ dữ liệu:

Bộ dữ liệu bao gồm thông tin của 1723 khách hàng theo 13 yếu tố ảnh hưởng tới khả năng trả nợ như: độ tuổi, giới tính, tình trạng hôn nhân, trình độ học vấn, thu nhập, số sản phẩm khách hàng nắm giữ, nơi ở, số tiền vay tín dụng, thời hạn vay…

2.2.2. Thông tin trường:

- month: tháng vay tín dụng
- credit_amount: số tiền yêu cầu vay
- credit_term: thời hạn cho vay (tính theo tháng)
- age: độ tuổi
- sex: giới tính
- education: trình độ học vấn
- product_type: loại sản phẩm
- having_children_flg: đã có con hay chưa (0 = chưa có con, 1 = đã có con)
- region: khu vực nơi ở khách hàng (khu vực 0-1-2)
- income: thu nhập
- family_status: tình trạng hôn nhân
- phone_operator: đang sử dung nhà cung cấp dịch vụ di động nào (nhà cung cấp số 1-2-3-4)
- is_client: có phải khách hàng không
- bad_client_target: khách hàng tốt hay xấu (1 = xấu, 0 = tốt)

2.2. Tiền xử lý dữ liệu

Import một vài thư viện cần thiết

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from collections import Counter
from imblearn import under_sampling, over_sampling
from imblearn.over_sampling import SMOTE
```

Quan sát dữ liệu

```
df.describe()
```

	month	credit_amount	credit_term	age	having_children_flg	region	income	phone_operator	is_client	bad_client_target
count	1723.000000	1723.000000	1723.000000	1723.000000	1723.000000	1723.000000	1723.000000	1723.000000	1723.000000	1723.000000
mean	6.708067	29264.654672	11.546721	35.911782	0.428323	1.681370	32652.350551	1.125363	0.604759	0.113755
std	3.538420	27926.778301	6.548354	13.120203	0.494979	0.704256	20913.193158	1.015822	0.489044	0.317606
min	1.000000	5000.000000	3.000000	18.000000	0.000000	0.000000	1000.000000	0.000000	0.000000	0.000000
25%	3.000000	13000.000000	6.000000	26.000000	0.000000	2.000000	21000.000000	0.000000	0.000000	0.000000
50%	7.000000	21500.000000	12.000000	32.000000	0.000000	2.000000	27000.000000	1.000000	1.000000	0.000000
75%	10.000000	34000.000000	12.000000	44.000000	1.000000	2.000000	38000.000000	2.000000	1.000000	0.000000
max	12.000000	301000.000000	36.000000	90.000000	1.000000	2.000000	401000.000000	4.000000	1.000000	1.000000

```
df.bad_client_target.value_counts()
```

```
0    1527
1     196
Name: bad_client_target, dtype: int64
```

Nhận xét: Tập dữ liệu bị mất cân bằng. Số lượng class ‘bad_client_target’ = 0 chiếm đa số. Trong những trường hợp phân phối lớp bị lệch như vậy, số liệu về độ chính xác sẽ bị sai lệch.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1723 entries, 0 to 1722
Data columns (total 14 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   month                1723 non-null   int64 
 1   credit_amount        1723 non-null   int64 
 2   credit_term          1723 non-null   int64 
 3   age                  1723 non-null   int64 
 4   sex                  1723 non-null   object
 5   education            1723 non-null   object
 6   product_type         1723 non-null   object
 7   having_children_flg  1723 non-null   int64 
 8   region               1723 non-null   int64 
 9   income               1723 non-null   int64 
 10  family_status        1723 non-null   object
 11  phone_operator       1723 non-null   int64 
 12  is_client            1723 non-null   int64 
 13  bad_client_target    1723 non-null   int64 
dtypes: int64(10), object(4)
memory usage: 188.6+ KB
```

Nhìn vào tổng quan thông tin của các trường dữ liệu, ta thấy cần chuyển đổi dạng dữ liệu do thuật toán không thể xử lý dữ liệu object. Ta tiến hành chuyển đổi ánh xạ dữ liệu về dạng int.

Chuyển đổi sex, family_status về dạng số:

```
df['sex'] = df['sex'].replace({'female':0, 'male':1})
df['family_status'] = df['family_status'].replace({'Another':0, 'Unmarried':1, 'Married': 2})
```

Chuyển đổi product_type về dạng số:

```
gle = LabelEncoder()
product_type_labels = gle.fit_transform(df['product_type'])
product_type_mappings = {index: label for index, label in
                  enumerate(gle.classes_)}
product_type_mappings
```

```
{0: 'Audio & Video',
 1: 'Auto',
 2: 'Boats',
 3: 'Cell phones',
 4: "Childen's goods",
 5: 'Clothing',
 6: 'Computers',
 7: 'Construction Materials',
 8: 'Cosmetics and beauty services',
 9: 'Fishing and hunting supplies',
 10: 'Fitness',
 11: 'Furniture',
 12: 'Garden equipment',
 13: 'Household appliances',
 14: 'Jewelry',
 15: 'Medical services',
 16: 'Music',
 17: 'Repair Services',
 18: 'Sporting goods',
 19: 'Tourism',
 20: 'Training',
 21: 'Windows & Doors'}
```

Chuyển đổi education về dạng số:

```
education_labels = gle.fit_transform(df['education'])
education_mappings = {index: label for index, label in
                      enumerate(gle.classes_)}
education_mappings
```

```
{0: 'Higher education',
 1: 'Incomplete higher education',
 2: 'Incomplete secondary education',
 3: 'PhD degree',
 4: 'Secondary education',
 5: 'Secondary special education'}
```

```
df['education'] = education_labels
df['product_type'] = product_type_labels
df.head()
```

nonth	credit_amount	credit_term	age	sex	education	product_type	having_children_flg	region	income	family_status	phone_operator	is_client	bad_client_tar
1	7000	12	39	1	5	3	0	2	21000	0	0	0	
1	19000	6	20	1	5	13	1	2	17000	0	3	1	
1	29000	12	23	0	5	13	0	2	31000	0	2	0	
1	10000	12	30	1	5	3	1	2	31000	1	3	1	
1	14500	12	25	0	0	3	0	2	26000	2	0	1	

2.3. Xây dựng mô hình

2.3.1. Random Forest với bộ dữ liệu ban đầu

Ta chia mẫu quan sát thành tập huấn luyện (training set) và tập kiểm thử (testing set) với tỷ lệ 70:30:

```
# Split the data into train & test units

x = df.drop(columns=['bad_client_target'])
y = df['bad_client_target']

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.30,random_state=0)
print('x_train.shape')
print(x_train.shape)
print('y_train.shape')
print(x_test.shape)
```

```
x_train.shape
(1206, 13)
y_train.shape
(517, 13)
```

Sử dụng thuật toán Random Forest cho tập huấn luyện và kiểm tra độ chính xác của mô hình trên tập kiểm thử:

```
# Random Forest Algorithm

rf_model = RandomForestClassifier()
rf_model.fit(x_train,y_train)
```

```
▾ RandomForestClassifier

RandomForestClassifier()
```

```
y_pred_rf = rf_model.predict(x_test)
print('confusion matrix')
print(metrics.confusion_matrix(y_test, y_pred_rf))
print('classification report')
print(metrics.classification_report(y_test,y_pred_rf))
print('accuracy: %f' %(metrics.accuracy_score(y_test, y_pred_rf)))
print('area under the curve: %f' % (metrics.roc_auc_score(y_test, y_pred_rf)))
```

```
confusion matrix
[[458   5]
 [ 51   3]]
classification report
              precision    recall  f1-score   support

           0       0.90      0.99      0.94       463
           1       0.38      0.06      0.10        54

    accuracy                           0.89       517
   macro avg       0.64      0.52      0.52       517
weighted avg       0.84      0.89      0.85       517

accuracy: 0.891683
area under the curve: 0.522378
```

Nhận xét: Có thể thấy, độ chính xác của mô hình được tính toán khá cao (89%). Tuy nhiên như đã trình bày, khi xảy ra hiện tượng mất cân bằng dữ liệu thì việc sử dụng độ chính xác làm thước đo đánh giá mô hình thường không hiệu quả. Ta quan sát 2 chỉ thay thế: f1-score và AUC (area under the curve) đều ở mức thấp. Vậy nên, tập dữ liệu này chưa thực sự tốt.

Ta sẽ sử dụng phương pháp **Random Under Sampling** và **SMOTE** để ngăn chặn mất cân bằng dữ liệu, đồng thời quan sát và đánh giá xem phương pháp nào là phù hợp nhất với tập dữ liệu để mô hình có độ chính xác lớn nhất.

2.3.2. Random Under Sampling

Thực hiện Under Sampling bằng cách giảm số mẫu good client về bằng với số lượng mẫu bad client. Sau đó lại trộn bad client và good client thành bộ dữ liệu test:

```
df_good = df[df.bad_client_target == 0]
df_bad = df[df.bad_client_target == 1]

#random sampling
ignore_me, sample = train_test_split(df_good, test_size = len(df_bad))
```

```
import warnings
warnings.filterwarnings("ignore")

sample = pd.concat([sample, df_bad])

sample
sample.bad_client_target.value_counts()
```

```
0    196
1    196
Name: bad_client_target, dtype: int64
```

```
#Split into train and test units

x1 = sample.drop(columns=['bad_client_target'])
y1 = sample['bad_client_target']

x1_train,x1_test,y1_train,y1_test = train_test_split(x1,y1,test_size=0.30,random_state=0)
print('x1_train.shape')
print(x1_train.shape)
print('y1_train.shape')
print(x1_test.shape)
```

```
x1_train.shape
(274, 13)
y1_train.shape
(118, 13)
```

Tương tự, ta sử dụng thuật toán Random Forest để kiểm tra độ chính xác của mô hình sau khi đã Under Sampling:

```
#Random Forest Algorithm on resampled dataset

rf_model1 = RandomForestClassifier()
rf_model1.fit(x1_train,y1_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
y1_pred_rf = rf_model1.predict(x1_test)
print('confusion matrix')
print(metrics.confusion_matrix(y1_test, y1_pred_rf))
print('classification report')
print(metrics.classification_report(y1_test,y1_pred_rf))
print('accuracy: %f' %(metrics.accuracy_score(y1_test, y1_pred_rf)))
print('area under the curve: %f' % (metrics.roc_auc_score(y1_test, y1_pred_rf)))
```

```
confusion matrix
[[41 17]
 [15 45]]
classification report
              precision    recall  f1-score   support

           0       0.73      0.71      0.72        58
           1       0.73      0.75      0.74        60

    accuracy                           0.73       118
   macro avg       0.73      0.73      0.73       118
weighted avg       0.73      0.73      0.73       118

accuracy: 0.728814
area under the curve: 0.728448
```

Nhận xét: So sánh kết quả cho ta thấy khi thực hiện Under Sampling đã giúp cải thiện được nhiều kết quả dự báo trên tập test ở cả 2 chỉ số AUC và f1-score. Tuy nhiên, các chỉ số vẫn ở mức trung bình và chưa được cao, nguyên nhân có thể là do việc thực hiện over sampling đã dẫn tới overfitting.

2.3.3. SMOTE

Với phương pháp SMOTE, trái ngược với Under Sampling, ta thực hiện tăng kích thước của dữ liệu bad client sao cho bằng với kích thước của dữ liệu good client và tạo thành bộ dữ liệu test.

```
x2 = df.drop(columns=['bad_client_target'])
y2 = df['bad_client_target']
```

```
#Increase the bad samples from 196 to 600

sm = SMOTE()
x_res,y_res = sm.fit_resample(x2,y2)
print('Resample dataset shape {}'.format(Counter(y_res)))
```

```
Resample dataset shape Counter({0: 1527, 1: 1527})
```

```
#Split into train and test units

x2_train,x2_test,y2_train,y2_test = train_test_split(x_res,y_res,test_size=0.30,random_state=0)
print('x2_train.shape')
print(x2_train.shape)
print('y2_train.shape')
print(x2_test.shape)
```

```
x2_train.shape
(2137, 13)
y2_train.shape
(917, 13)
```

Tiếp tục sử dụng Random Forest để kiểm tra các chỉ số đã điều chỉnh mô hình.

```
#Random Forest Classifier on resampled data

rf_model2 = RandomForestClassifier()
rf_model2.fit(x2_train,y2_train)
```

```
▾ RandomForestClassifier

RandomForestClassifier()
```

```
y2_pred_rf = rf_model2.predict(x2_test)
print('confusion matrix')
print(metrics.confusion_matrix(y2_test, y2_pred_rf))
print('classification report')
print(metrics.classification_report(y2_test,y2_pred_rf))
print('accuracy: %f' %(metrics.accuracy_score(y2_test, y2_pred_rf)))
print('area under the curve: %f' % (metrics.roc_auc_score(y2_test, y2_pred_rf)))
```

```
confusion matrix
[[444  45]
 [ 34 394]]
classification report
              precision    recall  f1-score   support

           0       0.93      0.91      0.92       489
           1       0.90      0.92      0.91       428

    accuracy                           0.91       917
   macro avg       0.91      0.91      0.91       917
weighted avg       0.91      0.91      0.91       917

accuracy: 0.913850
area under the curve: 0.914268
```

Nhận xét: Ta thấy phương pháp SMOTE đã giúp tăng mạnh chỉ số f1-score (91%) và AUC (91%) so với mô hình ban đầu. Như vậy, có thể kết luận rằng SMOTE là phương pháp cải thiện tốt nhất đối với bộ dữ liệu này.

KẾT LUẬN

Trong quá trình xây dựng mô hình, đặc biệt là các mô hình phân loại nhị phân (2 classes) chúng ta sẽ thường xuyên gặp hiện tượng mất cân bằng dữ liệu. Hiện tượng này sẽ dẫn tới mô hình dự báo kém chính xác và đa phần kết quả dự báo bị thiên về nhãn đa số. Bài báo cáo này đã trình bày một vài phương pháp hữu hiệu thường được sử dụng để đối phó với các tình huống mất cân bằng dữ liệu. Tùy vào từng bài toán và từng bộ dữ liệu mà người lập trình có thể lựa chọn một hoặc kết hợp một vài phương pháp để cải thiện hiệu năng mô hình.

TÀI LIỆU THAM KHẢO

Tiếng Việt

[1] https://phamdinhkhanh.github.io/

[2] https://viblo.asia/

Tiếng Anh

[2] https://machinelearningmastery.com/

[3] https://towardsdatascience.com/