

ĐẠI HỌC BÁCH KHOA HÀ NỘI
KHOA TOÁN - TIN



**BÁO CÁO CUỐI KỲ
HỆ HỒ TRỢ QUYẾT ĐỊNH**

Chủ đề: Bài toán phân loại điểm tín dụng

Giảng viên hướng dẫn: TS. Trần Ngọc Thăng

Sinh viên thực hiện: Tạ Thị Nga

MSSV: 20227026

Lớp: Toán Tin 02 - K67

Mã lớp học: 158243

Hà Nội - 2025

Mục lục

Mở đầu	6
1 Phát biểu bài toán	7
1.1 Mô tả bài toán	7
1.1.1 Mô tả bài toán	7
1.1.2 Đầu vào	8
1.1.3 Đầu ra	9
1.1.4 Yêu cầu xử lý	9
2 Tiền xử lý dữ liệu	10
2.1 Thu thập dữ liệu	10
2.2 Đánh nhãn dữ liệu	11
2.3 Quy trình tiền xử lý dữ liệu	12
2.3.1 Làm sạch dữ liệu nhiễu và định dạng lại giá trị	12
2.3.2 Chuyển đổi kiểu dữ liệu và kỹ thuật đặc trưng (Feature Engineering)	12
2.3.3 Xử lý giá trị thiếu (Missing Value Imputation)	13
2.3.4 Xử lý giá trị ngoại lệ và phi logic (Outlier Handling)	13
2.4 Thống kê dữ liệu mẫu	14
2.4.1 Phân phối nhãn và mất cân bằng dữ liệu	14
2.4.2 Ma trận tương quan	15
2.5 Xử lý mất cân bằng dữ liệu	17
3 Tạo, huấn luyện và đánh giá mô hình	18
3.1 Đề xuất, lựa chọn mô hình và tiêu chí đánh giá	18
3.1.1 Mô hình	18

3.1.2	Tiêu chí đánh giá	19
3.2	Huấn luyện mô hình	19
3.2.1	Chuẩn bị dữ liệu để huấn luyện	19
3.2.2	Quá trình huấn luyện	20
3.2.3	Điều kiện dừng	21
3.2.4	Kết quả huấn luyện và đánh giá mô hình	21
3.3	Thống kê và phân tích lỗi	27
3.4	Phân tích tầm quan trọng của đặc trưng	29
4	Cải tiến mô hình	30
4.1	Kiến trúc mô hình 1 - Logistic Regression	30
4.1.1	LR Saga	30
4.1.2	LR Liblinear	31
4.2	Kiến trúc mô hình 2 - Random Forest	31
4.2.1	RF1	31
4.2.2	RF2	31
4.3	Kiến trúc mô hình 3 - XGBoost	32
4.3.1	XGBoost1	32
4.3.2	XGBoost2	32
4.4	Kiến trúc mô hình 4 - LightGBM	33
4.4.1	LGBM1	33
4.4.2	LGBM2	33
5	Đóng gói mô hình	34
5.1	Bài toán nghiệp vụ	34
5.2	Các chỉ số đánh giá mô hình đủ điều kiện để ứng dụng vào thực tế	34
5.3	Giao diện demo chương trình	35
5.3.1	Xây dựng giao diện	36
5.3.2	Giao diện và luồng hoạt động của ứng dụng	36
5.3.3	Đóng gói giao diện chương trình	38
5.4	Làm slide báo cáo và thuyết trình	38
Kết luận		39

Checklist

YÊU CẦU BÀI TẬP CUỐI KỲ

STT	Loại yêu cầu	Yêu cầu	Điểm chữ	Điểm số	Check	Mình chứng
1	Xử lý dữ liệu (2 điểm)	Mô tả bài toán, đầu vào, đầu ra, yêu cầu xử lý	A	1	X	Trang 7 - 9
2		Đánh nhãn & Tiên xử lý dữ liệu	A	1	X	Trang 11 - 14
3		Thông kê dữ liệu mẫu	A	1	X	Trang 15 - 16
4		Xử lý mất cân bằng dữ liệu (cho bài toán phân lớp) hoặc Chuyển đổi dữ liệu (cho bài toán hồi quy)	A	1	X	Trang 18
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,...)	A	1	X	Trang 19 - 22
6	Cải tiến mô hình (4 điểm)	Thông kê và phân tích lỗi	A	1	X	Trang 28 - 29
7		LR Saga	B	0.75	X	Trang 30
8		LR Liblinear	A	1	X	Trang 31
9		RF1	A	1	X	Trang 31
10		RF2	A	1	X	Trang 31
11		XGBoost1	A	1	X	Trang 32
12		XGBoost2	A	1	X	Trang 32
13		LGBM1	A	1	X	Trang 33
14		LGBM2	A	1	X	Trang 33
15		Có sử dụng mô hình tiên tiến trong 3 năm trở lại đây (chỉ ra paper liên quan)	D	0.25	X	Trang 18
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)	A	1	X	Trang 34
17		Các chỉ số đánh giá mô hình đủ điều kiện để ứng dụng vào thực tế	A	1	X	Trang 34
18		Đóng gói giao diện demo chương trình	A	1	X	Trang 35 - 36
19		Làm slide báo cáo	A	1	X	Trang 36
20		Thuyết trình trên lớp	A	1	X	Trang 37
	Tổng điểm/20			19	0	
	Tổng điểm/10			9.5		

Mô tả chi tiết mô hình

BẢNG MÔ TẢ CHI TIẾT 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	
					Accuracy	Precision	Recall	F1-score		
1	LR	LR	Giá trị hàm mất mát giảm không quá 0.0001 hoặc đạt số vòng lặp tối đa là 10000	Sử dụng GridSearch với 5-fold Cross validation	<code>penalty= l2, C = 1.0, solver = lbfgs, class_weight=balanced</code>	0.67	0.64	0.69	0.66	Mô hình baseline
2		LR Saga	Giá trị hàm mất mát giảm không quá 0.0001 hoặc đạt số vòng lặp tối đa là 10000	Sử dụng GridSearch với 5-fold Cross validation	<code>penalty= l1, C=1.0, solver=saga, class_weight=balanced</code>	0.67	0.64	0.69	0.66	Best param of LR with Recall
3		LR Liblinear	Giá trị hàm mất mát giảm không quá 0.0001 hoặc đạt số vòng lặp tối đa là 10000	Sử dụng GridSearch với 5-fold Cross validation	<code>penalty= l1, C=1.0, solver=liblinear, class_weight=balanced</code>	0.67	0.64	0.69	0.65	Best param of LR with Precision
4	RF	RF	Mô hình huấn luyện đủ 100 cây với độ sâu không giới hạn	Sử dụng GridSearch với 5-fold Cross validation	<code>class_weight': 'balanced', 'criterion': 'gini', 'max_depth': None, 'min_impurity_decrease': 0, 'n_estimators': 100, 'oob_score': False</code>	0.82	0.84	0.8	0.82	
5		RF1	Mô hình huấn luyện đủ 150 cây với độ sâu không giới hạn	Sử dụng GridSearch với 5-fold Cross validation	<code>class_weight': 'balanced', 'criterion': 'gini', 'max_depth': None, 'min_impurity_decrease': 0, 'n_estimators': 150, 'oob_score': False</code>	0.82	0.8	0.85	0.83	Best param for RF
6		RF2	Mô hình huấn luyện đủ 200 cây với độ sâu không giới hạn	Sử dụng GridSearch với 5-fold Cross validation	<code>class_weight': 'balanced', 'criterion': 'gini', 'max_depth': None, 'min_impurity_decrease': 0, 'n_estimators': 200, 'oob_score': False</code>	0.82	0.8	0.84	0.82	
7	XGBoost	XGBoost	Mô hình huấn luyện đủ 100 cây, mỗi cây đạt độ sâu giới hạn là 6	Sử dụng GridSearch với 5-fold Cross validation	'n_estimators':100 , <code>'max_depth':6, 'learning_rate'=0.3</code>	0.76	0.74	0.83	0.78	
8		XGBoost1	Mô hình huấn luyện đủ 150 cây, mỗi cây đạt độ sâu giới hạn là 6	Sử dụng GridSearch với 5-fold Cross validation	'n_estimators':150 , <code>'max_depth':6, 'learning_rate'=0.3</code>	0.78	0.75	0.85	0.8	
9		XGBoost2	Mô hình huấn luyện đủ 200 cây, mỗi cây đạt độ sâu giới hạn là 6	Sử dụng GridSearch với 5-fold Cross validation	'n_estimators':200 , <code>'max_depth':6, 'learning_rate'=0.3</code>	0.79	0.76	0.86	0.81	Best param of XGBoost
10	LGBM	LGBM	<code>n_estimators=1000, early_stopping_rounds=50, random_state=42</code>		<code>'max_depth'=1, 'num_leaves'=31, 'min_child_samples'=20, 'learning_rate'=0.1</code>	0.72	0.76	0.72	0.72	
11		LGBM1	<code>n_iter=20</code>	Sử dụng Randomized Search CV với cv = 3	<code>'colsample_bytree': 0.978, 'learning_rate': 0.0827, 'max_depth': 9, 'n_estimators': 299, 'num_leaves': 28, 'subsample': 0.759</code>	0.75	0.79	0.75	0.75	
12		LGBM2	Dừng khi loại bỏ các đặc trưng có importance < 1% tổng trọng số	Sử dụng feature importance-based stopping	loại bỏ 11 đặc trưng ít quan trọng	0.75	0.78	0.75	0.75	

Mở đầu

Trong bối cảnh ngành tài chính - ngân hàng, việc đánh giá chính xác độ tin cậy của khách hàng là yếu tố then chốt, ảnh hưởng trực tiếp đến an toàn và lợi nhuận của tổ chức. Bài toán đặt ra là xây dựng một hệ thống có khả năng tự động phân loại điểm tín dụng (Credit Score) của khách hàng thành các nhóm: Tốt (Good), Trung bình (Standard) và Kém (Poor) dựa trên dữ liệu lịch sử.

Mục tiêu của báo cáo này là tìm hiểu, tiền xử lý dữ liệu, xây dựng và đánh giá các mô hình học máy khác nhau để tìm ra giải pháp tối ưu cho bài toán trên, từ đó đề xuất một hệ thống hỗ trợ quyết định hiệu quả cho việc sàng lọc hồ sơ tín dụng.

Em xin cảm ơn thầy Trần Ngọc Thăng vì những kiến thức thầy đã truyền tải thông qua học phần Hệ hỗ trợ quyết định này nay. Bài báo cáo được thực hiện trong khoảng thời gian hạn chế nên không thể tránh khỏi sai sót, em rất mong nhận được sự đóng góp của thầy và các bạn để bài báo cáo này có thể được hoàn thiện hơn!

Chương 1

Phát biểu bài toán

Chương này sẽ phát biểu bài toán cần giải quyết, đầu vào và đầu ra của bài toán cùng với các yêu cầu xử lý.

1.1 Mô tả bài toán

1.1.1 Mô tả bài toán

Trong lĩnh vực tài chính và ngân hàng, việc đánh giá độ tin cậy của một khách hàng là một nhiệm vụ cốt lõi, quyết định đến sự an toàn và lợi nhuận của tổ chức. Điểm tín dụng (Credit Score) là một chỉ số số học thể hiện khả năng trả nợ đúng hạn của một cá nhân, được tính toán dựa trên lịch sử tài chính của họ. Điểm tín dụng càng cao, độ rủi ro khi cho vay càng thấp và ngược lại.

Bài toán đặt ra là xây dựng một mô hình học máy có khả năng tự động phân loại điểm tín dụng của khách hàng vào các nhóm được định sẵn (Poor (Kém), Standard (Trung bình) và Good (Tốt)) dựa trên các thông tin cá nhân và lịch sử giao dịch của họ. Việc tự động hóa này giúp các tổ chức tài chính đưa ra quyết định cho vay nhanh chóng, khách quan, nhất quán và giảm thiểu rủi ro tín dụng. Đây là một bài toán phân lớp đa lớp (multi-class classification) trong lĩnh vực học máy có giám sát.

- Bài toán: xây dựng một mô hình học máy có khả năng phân loại điểm tín dụng (Credit Score) của khách hàng.
- Điểm tín dụng này được phân thành ba loại: Poor (Kém), Standard (Trung bình)

và Good (Tốt).

- Đây là một bài toán **phân loại đa lớp** (multi-class classification).
- Việc phân loại chính xác điểm tín dụng giúp các tổ chức tài chính đưa ra quyết định cho vay, quản lý rủi ro và cá nhân hóa sản phẩm/dịch vụ.

1.1.2 Đầu vào

Dữ liệu đầu vào là một tập tin CSV (train.csv) chứa thông tin của các khách hàng. Mỗi hàng đại diện cho một khách hàng và mỗi cột đại diện cho một thuộc tính của khách hàng đó. Các thuộc tính chính bao gồm:

- **Thông tin định danh (và các cột cần loại bỏ/xử lý đặc biệt):** ID, Customer_ID, Name, SSN, Month.
- **Thông tin nhân khẩu học:** Age (Tuổi), Occupation (Nghề nghiệp).
- **Thông tin tài chính chung:** Annual_Income (Thu nhập hàng năm), Monthly_Inhand_Salary (Lương thực nhận hàng tháng), Num_Bank_Accounts (Số lượng tài khoản ngân hàng), Outstanding_Debt (Nợ tồn đọng), Total_EMI_per_month (Tổng tiền EMI phải trả hàng tháng), Amount_invested_monthly (Số tiền đầu tư hàng tháng), Monthly_Balance (Số dư hàng tháng).
- **Thông tin về khoản vay và lịch sử tín dụng:** Num_Credit_Card (Số lượng thẻ tín dụng), Interest_Rate (Lãi suất), Num_of_Loan (Số lượng khoản vay), Type_of_Loan (Loại khoản vay), Changed_Credit_Limit (Thay đổi hạn mức tín dụng), Num_Credit_Inquiries (Số lần truy vấn tín dụng), Credit_Mix (Phân loại tín dụng), Credit_Utilization_Ratio (Tỷ lệ sử dụng tín dụng), Credit_History_Age (Tuổi lịch sử tín dụng).
- **Hành vi thanh toán:** Delay_from_due_date (Số ngày trễ hạn thanh toán), Num_of_Delayed_Payment (Số lần thanh toán trễ hạn), Payment_of_Min_Amount (Hành vi thanh toán số tiền tối thiểu), Payment_Behaviour (Hành vi thanh toán tổng quát).

1.1.3 Đầu ra

Đầu ra của mô hình là dự đoán về hạng điểm tín dụng cho mỗi khách hàng. Dựa trên bộ dữ liệu tham khảo, các hạng điểm tín dụng (giá trị của cột Credit_Score) có thể là: Good, Standard hoặc Poor.

Mô hình sẽ trả về một nhãn phân loại tương ứng với một trong các hạng điểm trên cho mỗi mẫu dữ liệu đầu vào.

1.1.4 Yêu cầu xử lý

1. Tiền xử lý dữ liệu:

- Làm sạch: Xử lý giá trị thiếu, ngoại lai.
- Chuyển đổi: Mã hóa biến định tính (categorical), xử lý text (nếu có), chuẩn hóa/quy mô hóa biến số.
- Đặc trưng: Lựa chọn hoặc kỹ thuật đặc trưng (feature engineering/selection).

2. Phân chia dữ liệu: Chia thành tập huấn luyện (train) và tập kiểm tra (test).

3. Xây dựng và huấn luyện mô hình:

- Lựa chọn thuật toán phân loại phù hợp (ví dụ: Logistic Regression, Random Forest, XGBoost).
- Huấn luyện mô hình trên tập huấn luyện.

4. Đánh giá mô hình: Sử dụng các độ đo thích hợp (Accuracy, Precision, Recall, F1-score, Confusion Matrix) trên tập kiểm tra.

5. Cải tiến mô hình:

- Tinh chỉnh siêu tham số để cải thiện hiệu năng.
- Phân tích tầm quan trọng của đặc trưng để hiểu mô hình.

6. Đóng gói mô hình: Sau khi lựa chọn ra mô hình tốt nhất, xây dựng bài toán nghiệp vụ thực tế và demo giao diện.

Chương 2

Tiền xử lý dữ liệu

Đây là giai đoạn quan trọng nhất, quyết định phần lớn đến chất lượng của mô hình. Quy trình được thực hiện một cách có hệ thống để đảm bảo dữ liệu sạch, nhất quán và sẵn sàng cho việc huấn luyện.

2.1 Thu thập dữ liệu

Bộ dữ liệu trên Kaggle:

<https://www.kaggle.com/datasets/parisrohan/credit-score-classification/data>.

- **train.csv**: Chứa 100,000 mẫu và 28 cột với đầy đủ các đặc trưng (features) và biến mục tiêu (Credit_Score). Tệp này được sử dụng để huấn luyện(training) và đánh giá (validation) mô hình với tỷ lệ 80 - 20.
- **test.csv**: Chứa 50,000 mẫu và 27 cột chỉ với các đặc trưng, **không có** biến mục tiêu. Tệp này đại diện cho dữ liệu mới mà mô hình cần dự đoán kết quả.

Để đảm bảo các bước tiền xử lý được áp dụng đồng bộ và nhất quán cho cả tập huấn luyện và tập kiểm thử, hai DataFrame này được hợp nhất thành một DataFrame duy nhất có tên **df** với tổng cộng 150,000 dòng. Thao tác này giúp tránh được hiện tượng rò rỉ dữ liệu và đảm bảo các ánh xạ là như nhau trên cả hai tập.

Một số mẫu dữ liệu ban đầu

ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Interest_Rate	Num_of_Loan	Type_of_Loan	E
0x1602	CUS_0xd40	January	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	1824.843	3	4	3	4	Auto Loan, Credit-Builder Loan, Personal Loan,...	
0x1603	CUS_0xd40	February	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	Nan	3	4	3	4	Auto Loan, Credit-Builder Loan, Personal Loan,...	
0x1604	CUS_0xd40	March	Aaron Maashoh	-500	821-00-0265	Scientist	19114.12	Nan	3	4	3	4	Auto Loan, Credit-Builder Loan, Personal Loan,...	
0x1605	CUS_0xd40	April	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	Nan	3	4	3	4	Auto Loan, Credit-Builder Loan, Personal Loan,...	
0x1606	CUS_0xd40	May	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	1824.843	3	4	3	4	Auto Loan, Credit-Builder Loan, Personal Loan,...	

Hình 2.1: Một số dòng dữ liệu ban đầu của tập train.csv.

	count	mean	std	min	25%	50%	75%	max
Monthly_Inhand_Salary	84998.000	4194.171	3183.686	303.645	1625.568	3093.745	5957.448	15204.633
Num_Bank_Accounts	100000.000	17.091	117.405	-1.000	3.000	6.000	7.000	1798.000
Num_Credit_Card	100000.000	22.474	129.057	0.000	4.000	5.000	7.000	1499.000
Interest_Rate	100000.000	72.466	466.423	1.000	8.000	13.000	20.000	5797.000
Delay_from_due_date	100000.000	21.069	14.860	-5.000	10.000	18.000	28.000	67.000
Num_Credit_Inquiries	98035.000	27.754	193.177	0.000	3.000	6.000	9.000	2597.000
Credit_Utilization_Ratio	100000.000	32.285	5.117	20.000	28.053	32.306	36.497	50.000
Total_EMI_per_month	100000.000	1403.118	8306.041	0.000	30.307	69.249	161.224	82331.000

Hình 2.2: Thống kê mô tả đặc trưng dạng số.

2.2 Đánh nhãn dữ liệu

Biến mục tiêu Credit_Score đã được cung cấp sẵn trong tập dữ liệu, với các nhãn là "Good", "Standard" và "Poor". Không cần thực hiện thêm bước đánh nhãn thủ công và các nhãn này cần được mã hóa sang dạng số như sau:

- Good → 0.
- Standard → 2.

- Poor → 1.

2.3 Quy trình tiền xử lý dữ liệu

Quy trình được chia thành các bước chính, thực hiện tuần tự để giải quyết các vấn đề đã xác định.

2.3.1 Làm sạch dữ liệu nhiễu và định dạng lại giá trị

Bước đầu tiên là loại bỏ các ký tự và chuỗi không hợp lệ, chuyển chúng về dạng giá trị thiếu chuẩn (np.NaN) để dễ dàng xử lý ở các bước sau.

- Một hàm `text_cleaning` được định nghĩa để loại bỏ các ký tự ‘_’, ‘;’, ‘”’ ở đầu và cuối chuỗi.
- Sử dụng phương thức `applymap` để áp dụng hàm này lên toàn bộ DataFrame.
- Các chuỗi rỗng (‘ ’), `nan`, `9#%8`, `#F%$D&8` được thay thế bằng `np.NaN`.

Sau bước này, số lượng giá trị thiếu trong các cột như `Occupation`, `Payment_Behaviour`, `Credit_Mix` tăng lên, phản ánh đúng thực trạng dữ liệu bị nhiễu.

2.3.2 Chuyển đổi kiểu dữ liệu và kỹ thuật đặc trưng (Feature Engineering)

Nhiều cột số học đang ở định dạng chuỗi, cần được chuyển đổi. Đồng thời, một số cột phức tạp được biến đổi để trích xuất thông tin hữu ích.

- `ID`, `Customer_ID`: Chuyển đổi từ chuỗi hexa (ví dụ: '0x1602', 'CUS_0xd40') sang dạng số nguyên để có thể sử dụng trong các thuật toán.
- `Month`: Chuyển đổi từ tên tháng (ví dụ: 'January') sang số thứ tự (1 đến 12) bằng `pd.to_datetime`.
- `SSN`: Loại bỏ các dấu gạch ngang và chuyển sang dạng số.
- **Các cột số khác:** Các cột như `Age`, `Annual_Income`, `Num_of_Loan`, v.v. được ép về kiểu số học (`int` hoặc `float`).

- **Credit_History_Age:** Đây là một bước kỹ thuật đặc trưng quan trọng. Một hàm tùy chỉnh Month_Converter được tạo ra để chuyển đổi định dạng "X Years and Y Months" thành một giá trị số duy nhất biểu diễn tổng số tháng (công thức: tổng tháng = $X * 12 + Y$). Điều này giúp mô hình hiểu được độ lớn của lịch sử tín dụng.
- **Type_of_Loan:** Chuỗi chứa các loại vay được làm sạch (chuyển về chữ thường, loại bỏ "and") để chuẩn bị cho các bước xử lý sau (ví dụ: one-hot encoding).

2.3.3 Xử lý giá trị thiếu (Missing Value Imputation)

Chiến lược xử lý giá trị thiếu được áp dụng một cách thông minh bằng cách gom nhóm theo Customer_ID. Điều này giả định rằng thông tin của một khách hàng qua các tháng là nhất quán.

- **Đối với các cột phân loại (Categorical):**
 - Sử dụng hàm Object_NaN_Values_Reassign_Group_Mode.
 - Các giá trị thiếu trong các cột như Name, Occupation, Payment_Behaviour, Credit_Mix được điền bằng giá trị xuất hiện nhiều nhất (**mode**) của chính khách hàng đó. Ví dụ, nếu một khách hàng có 10 bản ghi là 'Engineer' và 2 bản ghi thiếu, 2 bản ghi này sẽ được điền là 'Engineer'.
- **Đối với các cột số (Numerical):**
 - **Credit_History_Age:** Do đây là một chuỗi thời gian (tăng dần mỗi tháng), phương pháp nội suy tuyến tính (**interpolate**) được sử dụng trong mỗi nhóm khách hàng để điền các giá trị thiếu một cách logic. Các hàm bfill và ffill được dùng để xử lý các giá trị thiếu ở đầu hoặc cuối chuỗi.
 - **Các cột số khác:** Các giá trị thiếu được điền bằng giá trị **mode** của chính khách hàng đó, tương tự như các cột phân loại.

2.3.4 Xử lý giá trị ngoại lệ và phi logic (Outlier Handling)

Sau khi xử lý giá trị thiếu, bước tiếp theo là xác định và sửa các giá trị không hợp lệ.

- **Phương pháp:** Sử dụng hàm Numeric_Wrong_Values_Reassign_Group_Min_Max. Hàm này xác định một khoảng giá trị hợp lệ cho mỗi khách hàng (dựa trên các giá trị mode của họ) và thay thế các giá trị nằm ngoài khoảng này bằng giá trị mode.
- **Các cột được xử lý:** Age (loại bỏ tuổi âm và tuổi quá lớn), Num_Bank_Accounts, Num_Credit_Card, Interest_Rate, Num_of_Loan, Total_EMI_per_month, Annual_Income.
- **Xử lý các trường hợp đặc biệt (post-processing):**
 - Num_Bank_Accounts < 0: Được gán lại bằng 0.
 - Delay_from_due_date < 0: Các giá trị này được xem là lỗi và được gán bằng None, sau đó được điền lại bằng mode của khách hàng.
 - Num_of_Delayed_Payment < 0: Xử lý tương tự như trên.
 - Monthly_Balance < 0: Các giá trị âm rất lớn được xác định là lỗi, gán bằng None và điền lại.
 - Amount_invested_monthly >= 10000: Các giá trị bằng hoặc lớn hơn 10000 được xem là giá trị placeholder/outlier, được gán bằng None và điền lại bằng mode của khách hàng.

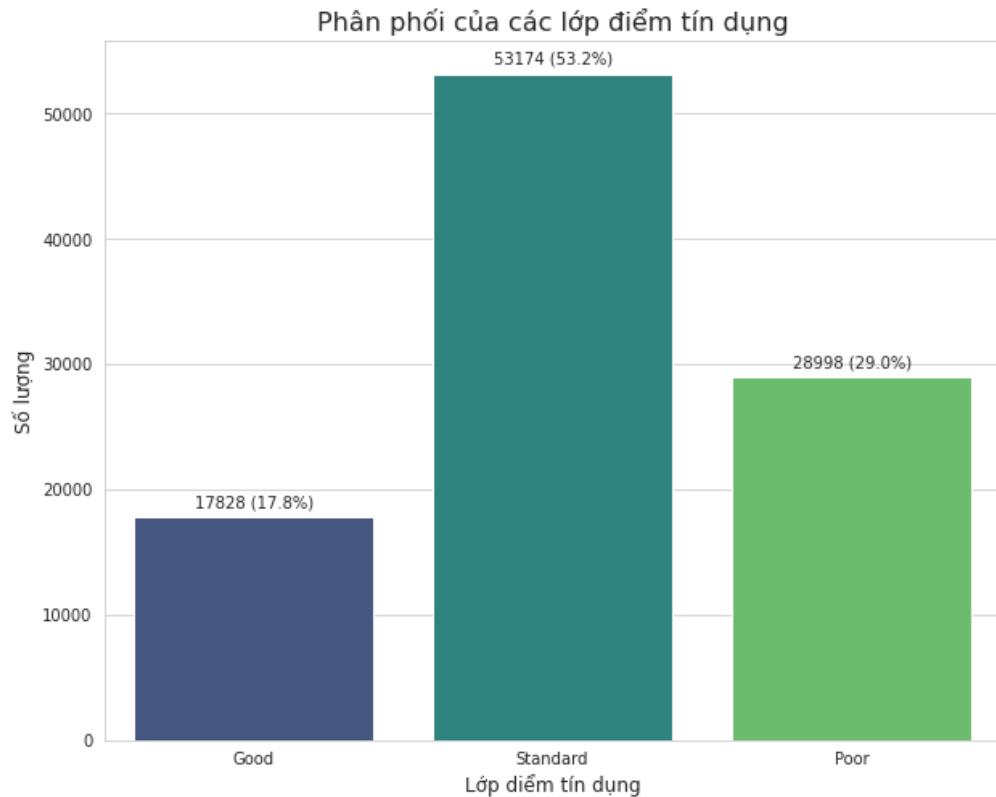
Sau khi hoàn thành tất cả các bước trên, bộ dữ liệu đã sạch, không còn giá trị thiếu hay giá trị phi logic, sẵn sàng cho giai đoạn tiếp theo.

2.4 Thông kê dữ liệu mẫu

Sau quá trình tiền xử lý, việc phân tích thống kê dữ liệu là cần thiết để hiểu rõ hơn về đặc tính của bộ dữ liệu đã làm sạch.

2.4.1 Phân phối nhãn và măt cân bằng dữ liệu

Phân tích cột nhãn Credit_Score trong tập huấn luyện cho thấy sự măt cân bằng giữa các lớp.



Hình 2.3: Các lớp điểm tín dụng.

Nhận xét:

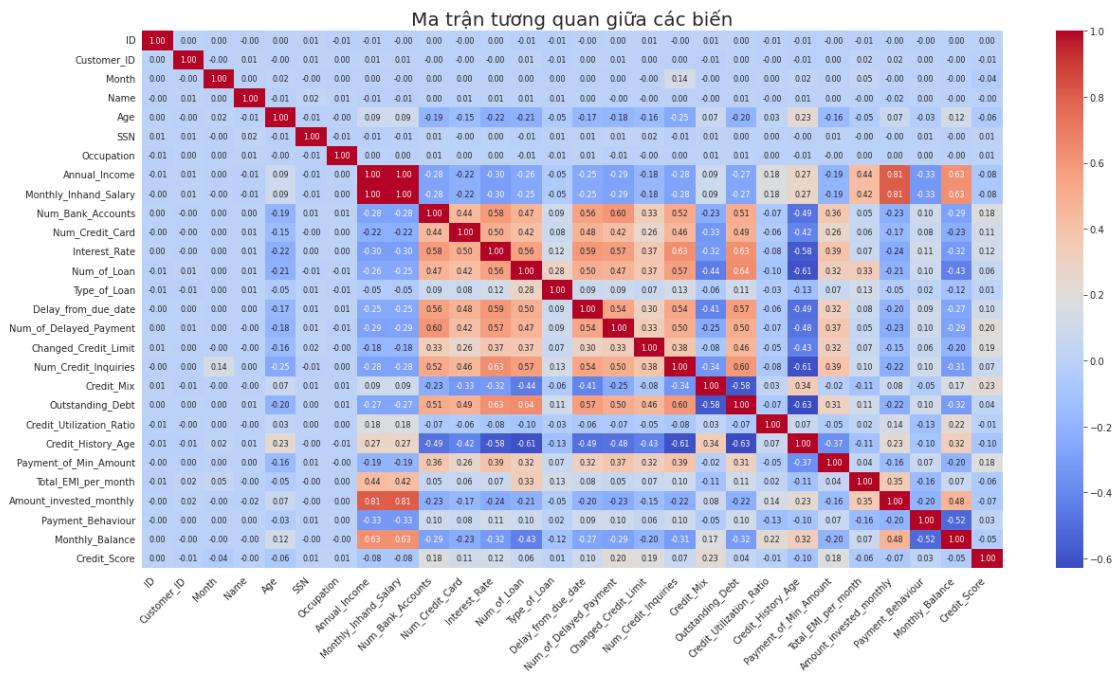
- Dữ liệu bị mất cân bằng (Imbalanced Data):
 - Biểu đồ cho thấy rõ sự chênh lệch lớn giữa các lớp.
 - Lớp Standard chiếm đa số với 53.2%.
 - Lớp Poor chiếm 29.0%.
 - Lớp Good là lớp thiểu số, chỉ chiếm 17.8%.
- Nếu không xử lý, mô hình học máy có xu hướng dự đoán tốt cho lớp Standard nhưng lại kém hiệu quả trên hai lớp còn lại, đặc biệt là lớp Good. Điều này rất nguy hiểm trong thực tế vì mô hình có thể bỏ lỡ các khách hàng tốt.

2.4.2 Ma trận tương quan

Sau khi toàn bộ dữ liệu đã được chuyển đổi sang dạng số, việc tính toán ma trận tương quan giữa các biến là cần thiết để khám phá mối quan hệ tuyến tính giữa chúng. Ma trận

này giúp:

- Xác định các thuộc tính có tương quan cao với biến mục tiêu (`Credit_Score`).
- Phát hiện hiện tượng đa cộng tuyến (multicollinearity), tức là các biến đầu vào có tương quan mạnh với nhau. Điều này có thể gây nhiễu cho một số mô hình học máy.



Hình 2.4: Ma trận tương quan giữa các biến.

Nhận xét:

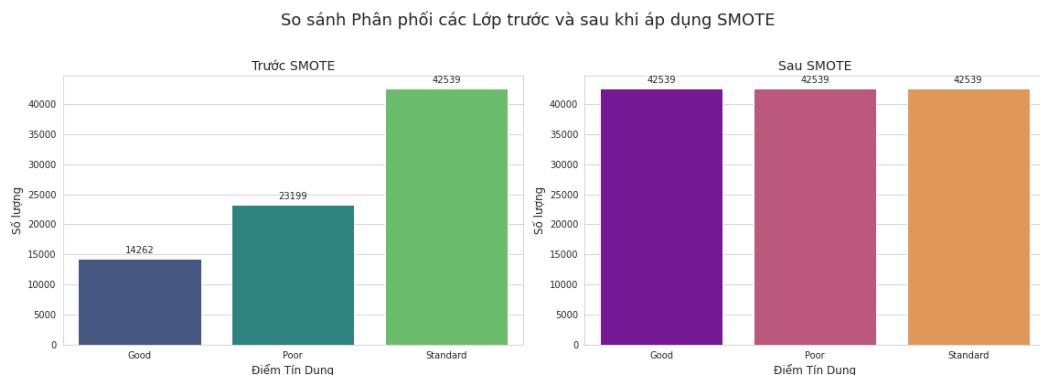
- `Annual_Income` và `Monthly_Inhand_Salary` có tương quan dương rất mạnh (gần như bằng 1), điều này là hợp lý. Chúng ta có thể xem xét loại bỏ một trong hai biến để tránh đa cộng tuyến.
- `Num_of_Loan` và `Total_EM_per_month` cũng có mối tương quan dương đáng kể.
- Biến mục tiêu `Credit_Score` có vẻ tương quan với các biến như `Interest_Rate`, `Delay_from_due_date`, `Num_of_Delayed_Payment` và `Credit_Mix`. Cụ thể, lãi suất cao `Interest_Rate` và số lần trả trễ `Num_of_Delayed_Payment` có xu hướng liên quan đến điểm tín dụng thấp hơn.

2.5 Xử lý mất cân bằng dữ liệu

Để giải quyết vấn đề mất cân bằng dữ liệu, các kỹ thuật sau đây được đề xuất để áp dụng trước khi huấn luyện mô hình:

- **SMOTE (Synthetic Minority Over-sampling Technique):** Đây là một kỹ thuật over-sampling phổ biến, tạo ra các mẫu dữ liệu tổng hợp mới cho các lớp thiểu số ("Good" và "Poor") bằng cách nội suy giữa các mẫu gần nhau. Điều này giúp cân bằng phân phối lớp mà không cần phải xóa dữ liệu.
- **Sử dụng trọng số lớp (Class Weighting):** Trong quá trình huấn luyện mô hình, có thể gán trọng số cao hơn cho các lớp thiểu số. Điều này khiến cho việc phân loại sai một mẫu thuộc lớp thiểu số sẽ bị "phạt" nặng hơn, buộc mô hình phải chú ý hơn đến các lớp này.

Việc áp dụng một trong các phương pháp trên sẽ giúp cải thiện đáng kể hiệu suất của mô hình phân loại.



Chương 3

Tạo, huấn luyện và đánh giá mô hình

3.1 Đề xuất, lựa chọn mô hình và tiêu chí đánh giá

3.1.1 Mô hình

Các mô hình em lựa chọn để chạy dữ liệu là:

1. **Logistic Regression:** Đây là một mô hình đơn giản, nhanh và dễ diễn giải. Nó đóng vai trò là một mô hình baseline tốt để so sánh hiệu suất với các mô hình phức tạp hơn.
2. **Random Forest:** Là một mô hình ensemble learning dựa trên cây quyết định. Nó có khả năng xử lý các mối quan hệ phi tuyến tính phức tạp, ít bị overfitting hơn so với một cây quyết định đơn lẻ và thường cho kết quả tốt trên dữ liệu dạng bảng.
3. **XGBoost:** Là một thuật toán gradient boosting tiên tiến, thường xuyên đạt hiệu suất hàng đầu trong các cuộc thi về dữ liệu dạng bảng. Nó mạnh mẽ, hiệu quả và có nhiều cơ chế chống overfitting tích hợp sẵn. Đây được coi là một mô hình state-of-the-art¹.
4. **LightGBM:** Là một phiên bản tối ưu khác của Gradient Boosting, được thiết kế để có tốc độ huấn luyện nhanh hơn và sử dụng ít bộ nhớ hơn so với XGBoost, đặc biệt hiệu quả trên các bộ dữ liệu lớn. Việc đưa LightGBM vào so sánh giúp đánh giá sự cân bằng giữa hiệu suất và hiệu quả tính toán.

¹<https://dl.acm.org/doi/pdf/10.1145/2939672.2939785>

3.1.2 Tiêu chí đánh giá

- **Confusion Matrix (Ma trận nhầm lẫn):** Cho thấy số lượng dự đoán đúng và sai cho từng lớp.
- **Classification Report (Báo cáo phân loại):** Bao gồm:
 - **Precision (Độ chính xác):** Tỷ lệ các mẫu được dự đoán là thuộc một lớp mà thực sự thuộc lớp đó.
 - **Recall (Độ phủ, Độ nhạy):** Tỷ lệ các mẫu thực sự thuộc một lớp được mô hình dự đoán đúng.
 - **F1-score:** Trung bình điều hòa của Precision và Recall, cung cấp một cái nhìn cân bằng về hiệu suất của mô hình.
 - **Accuracy (Độ chính xác tổng thể):** Tỷ lệ tổng số mẫu được dự đoán đúng. (Đơn giản nhưng có thể gây hiểu nhầm nếu mất cân bằng)
- Các chỉ số này được tính cho từng lớp và giá trị trung bình (macro avg, weighted avg).

Ngoài ra, đường cong Precision-Recall và ROC-AUC cũng được xem xét để so sánh hiệu năng giữa các mô hình, đặc biệt là trong bối cảnh dữ liệu mất cân bằng.

Trong bài toán này, **Recall cho lớp "Poor"** là cực kỳ quan trọng, vì việc bỏ sót một khách hàng có điểm tín dụng kém (dự đoán sai thành "Standard" hoặc "Good") sẽ dẫn đến rủi ro tài chính lớn cho tổ chức. Do đó, trong quá trình hiệu chỉnh siêu tham số (GridSearchCV), "recall" cho lớp "Poor" (được mã hóa là lớp 1) được chọn làm tiêu chí chính.

3.2 Huấn luyện mô hình

3.2.1 Chuẩn bị dữ liệu để huấn luyện

1. **Đọc dữ liệu đã làm sạch:** Tải lại tệp `train.csv` và `test.csv` đã được làm sạch từ bước trước.

2. **Loại bỏ các cột không cần thiết:** Các cột định danh như ID, Customer_ID, Month, Name, SSN được loại bỏ khỏi cả tập train và test vì chúng không mang thông tin dự đoán.
3. **Phân chia Train/Validation:** Dữ liệu huấn luyện (df_train) được chia thành tập huấn luyện (X_train, y_train) và tập validation (X_val, y_val) với tỷ lệ 80:20. Tập X_test giữ nguyên từ df_test.
4. **Chuẩn hóa dữ liệu:** MinMaxScaler được áp dụng cho dữ liệu sau khi đã OneHotEncode (cho Logistic Regression). Đối với các mô hình cây, việc chuẩn hóa không bắt buộc nhưng vẫn được thực hiện trong một số thử nghiệm.

3.2.2 Quá trình huấn luyện

- Logistic Regression: Mô hình học một tập hợp các trọng số (coefficients) cho mỗi đặc trưng. Quá trình này được thực hiện bằng cách tối ưu hóa một hàm mất mát (log-loss) thông qua các thuật toán như saga hay liblinear.
- Random Forest: Mô hình xây dựng một "khu rừng" gồm nhiều cây quyết định. Mỗi cây được huấn luyện trên một tập mẫu con ngẫu nhiên của dữ liệu (bootstrap sampling). Kết quả cuối cùng được tổng hợp từ dự đoán của tất cả các cây (thường bằng cách bỏ phiếu).
- XGBoost: Mô hình xây dựng các cây quyết định một cách tuần tự. Mỗi cây mới được tạo ra để sửa lỗi của các cây trước đó. Quá trình này được gọi là "boosting".
- LightGBM: Tương tự XGBoost, đây cũng là một mô hình gradient boosting. Tuy nhiên, nó sử dụng một chiến lược tăng trưởng cây khác biệt gọi là leaf-wise thay vì level-wise. Thay vì phát triển cây theo từng tầng, LightGBM chọn lá cây có khả năng giảm mất mát nhiều nhất để phân tách tiếp. Cách tiếp cận này giúp mô hình hội tụ nhanh hơn và thường hiệu quả hơn về mặt tính toán, đặc biệt trên các bộ dữ liệu lớn.

3.2.3 Điều kiện dừng

- Logistic Regression: Dừng khi đạt đến số vòng lặp tối đa (max_iter) hoặc khi sự cải thiện của hàm mất mát nhỏ hơn một ngưỡng (tol).
- Random Forest: Dừng khi đã xây dựng đủ số cây (n_estimators). Các điều kiện khác như max_depth (độ sâu tối đa của cây) cũng giới hạn sự phát triển của từng cây.
- XGBoost: Tương tự Random Forest, dừng khi đủ n_estimators. Ngoài ra, nó hỗ trợ Early Stopping, tức là dừng quá trình huấn luyện sớm nếu hiệu suất trên tập validation không cải thiện sau một số vòng lặp nhất định.
- LightGBM: Dừng lại khi đã xây dựng đủ số lượng cây (n_estimators) trong mỗi lần thử của RandomizedSearchCV.

3.2.4 Kết quả huấn luyện và đánh giá mô hình

Logistic Regression (LR)

GRID LOG MODEL BALANCED					
TEST					
[[2940 96 491] [887 4041 946] [2066 2184 6349]]					
	precision	recall	f1-score	support	
0	0.50	0.83	0.62	3527	
1	0.64	0.69	0.66	5874	
2	0.82	0.60	0.69	10599	
accuracy				0.67	20000
macro avg		0.65	0.71	0.66	20000
weighted avg		0.71	0.67	0.67	20000

TRAIN					
[[11989 378 1934] [3693 15795 3636] [8491 8581 25503]]					
	precision	recall	f1-score	support	
0	0.50	0.84	0.62	14301	
1	0.64	0.68	0.66	23124	
2	0.82	0.60	0.69	42575	
accuracy				0.67	80000
macro avg		0.65	0.71	0.66	80000
weighted avg		0.71	0.67	0.67	80000

Hình 3.1: Kết quả đánh giá theo các chỉ số của mô hình LR.

Nhận xét:

- Recall trung bình cho lớp "Poor" (Recall lớp 1 = 0.69): Mô hình chỉ phát hiện được 69% trong tổng số các khách hàng thực sự có tín dụng kém. Điều này có nghĩa là 31% khách hàng rủi ro đã bị bỏ sót và bị phân loại nhầm vào nhóm "Good" hoặc "Standard". Đây là một lỗ hổng rủi ro đáng kể.
- Recall thấp cho lớp "Standard" (Recall lớp 2 = 0.60): Mô hình chỉ xác định được 60% khách hàng thuộc nhóm "Standard". Số còn lại bị phân loại nhầm sang hai nhóm kia. Điều này có thể dẫn đến việc đánh giá sai và bỏ lỡ các cơ hội kinh doanh với nhóm khách hàng này.

```

LOGISTIC REGRESSION MODEL (trên dữ liệu SMOTE)
TEST
[[2861 98 568]
 [ 882 4218 774]
 [3553 2563 4483]]
      precision    recall   f1-score  support
          0       0.39     0.81     0.53      3527
          1       0.61     0.72     0.66      5874
          2       0.77     0.42     0.55     10599

      accuracy           0.58      20000
      macro avg       0.59     0.65     0.58      20000
      weighted avg    0.66     0.58     0.58      20000
-----  

TRAIN
[[35648 1103 5824]
 [ 6612 30393 5570]
 [14210 10029 18336]]
      precision    recall   f1-score  support
          0       0.63     0.84     0.72      42575
          1       0.73     0.71     0.72      42575
          2       0.62     0.43     0.51      42575

      accuracy           0.66      127725
      macro avg       0.66     0.66     0.65      127725
      weighted avg    0.66     0.66     0.65      127725

```

Hình 3.2: Kết quả trên dữ liệu sử dụng SMOTE của mô hình LR

Kết quả này thấp hơn kết quả sử dụng tham số ở trên, em cũng đã thử cho cả 4 mô hình đều có kết quả thấp hơn nên em sẽ chỉ sử dụng tham số để cân bằng dữ liệu cho các mô hình.

Random Forest(RF)

GRID RF MODEL BALANCED				
TEST				
[[3007 98 422] [1019 4448 407] [3073 2628 4898]]				
precision	recall	f1-score	support	
0	0.42	0.85	0.57	3527
1	0.62	0.76	0.68	5874
2	0.86	0.46	0.60	10599
accuracy				0.62
macro avg				0.62
weighted avg				0.62

TRAIN				
[[12256 407 1638] [4166 17461 1497] [12313 10318 19952]]				
precision	recall	f1-score	support	
0	0.43	0.86	0.57	14301
1	0.62	0.76	0.68	23124
2	0.86	0.47	0.61	42575
accuracy				0.62
macro avg				0.62
weighted avg				0.62

Hình 3.3: Kết quả đánh giá theo các chỉ số của mô hình RF.

Nhận xét:

- Điểm mạnh:
 - Recall cao cho lớp "Good" (Recall lớp 0 = 0.85): Mô hình có khả năng tìm ra đến 85% các khách hàng thực sự có tín dụng tốt. Đây là một cải thiện so với mô hình Logistic.
 - Recall khá cho lớp "Poor" (Recall lớp 1 = 0.76): Mô hình phát hiện được 76% khách hàng rủi ro, cao hơn mức 69% của mô hình Logistic. Về mặt quản lý rủi ro.
- Điểm yếu:
 - Precision cực thấp cho lớp "Good" (Precision lớp 0 = 0.42): Đây là một vấn đề rất nghiêm trọng. Khi mô hình dự đoán một khách hàng là "Good", xác suất đúng chỉ là 42%.

- Recall cực thấp cho lớp "Standard" (Recall lớp 2 = 0.46): Đây là điểm yếu chí mạng của mô hình. Nó chỉ nhận diện đúng được chưa tới một nửa số khách hàng có tín dụng "Standard".
- F1-score thấp: F1-score của lớp "Good" (0.57) và "Standard" (0.60) đều rất thấp, cho thấy sự mất cân bằng nghiêm trọng giữa Precision và Recall ở các lớp này.

XGBoost(XGB)

```

GRID XGB MODEL BALANCED
TEST
[[3064 37 426]
 [ 367 4880 627]
 [1655 1724 7220]]
      precision   recall   f1-score   support
          0       0.60     0.87     0.71      3527
          1       0.73     0.83     0.78      5874
          2       0.87     0.68     0.77     10599

      accuracy
macro avg       0.74     0.79     0.75     20000
weighted avg     0.78     0.76     0.76     20000
-----TRAIN
[[13192    77 1032]
 [ 995 20368 1761]
 [ 6094 6183 30298]]
      precision   recall   f1-score   support
          0       0.65     0.92     0.76     14301
          1       0.76     0.88     0.82     23124
          2       0.92     0.71     0.80     42575

      accuracy
macro avg       0.78     0.84     0.79     80000
weighted avg     0.82     0.80     0.80     80000

```

Hình 3.4: Kết quả đánh giá theo các chỉ số của mô hình XGB.

Nhận xét:

- Hiệu suất cao: Với accuracy 76% và F1-score 76% (weighted) trên tập TEST, mô hình này rõ ràng mạnh mẽ hơn đáng kể so với các mô hình trước đó (67% và 62%).
- Cân bằng tốt giữa các lớp: Macro F1-score (0.75) và Weighted F1-score (0.76) khá gần nhau, cho thấy mô hình hoạt động tương đối đồng đều trên các lớp, không bị "thiên vị" quá nhiều cho lớp đa số.
- Điểm mạnh:

- Phát hiện rất tốt khách hàng "Good" (Recall lớp 0 = 0.87): Mô hình có khả năng nhận diện chính xác 87% khách hàng có tín dụng tốt, một con số rất ấn tượng.
- Phát hiện rất tốt khách hàng "Poor" (Recall lớp 1 = 0.83): Đây là cải thiện quan trọng nhất về mặt nghiệp vụ. Mô hình phát hiện đúng 83% khách hàng có tín dụng kém. Tỷ lệ bỏ sót khách hàng rủi ro đã giảm xuống chỉ còn 17%, thấp hơn đáng kể so với các mô hình trước.
- Precision cao cho lớp "Poor" (Precision lớp 1 = 0.73): Khi mô hình dự đoán một khách hàng là "Poor", nó đúng trong 73% trường hợp. Điều này giúp giảm số lượng khách hàng tiềm năng bị từ chối nhầm.
- Precision rất cao cho lớp "Standard" (Precision lớp 2 = 0.87): Khi mô hình dự đoán một khách hàng là "Standard", nó có độ tin cậy rất cao.

- Điểm yếu:

- Recall của lớp "Standard" vẫn còn ở mức vừa phải (Recall lớp 2 = 0.68): Mặc dù đã cải thiện so với Random Forest (46%), mô hình vẫn chỉ nhận diện được 68% khách hàng "Standard". Điều này cho thấy lớp "Standard" có các đặc điểm khá giống với hai lớp còn lại và khó phân biệt.

LightGBM(LBGM)

```
==== Kết quả đánh giá LightGBM ====
Accuracy: 0.7199
Precision: 0.7590
Recall: 0.7199
F1-score: 0.7224

Báo cáo phân loại:
      precision    recall   f1-score   support
  Good        0.55      0.86      0.67      3527
  Poor        0.70      0.80      0.75      5874
Standard     0.86      0.63      0.73     10599

accuracy          0.72      20000
macro avg       0.70      0.76      0.71      20000
weighted avg    0.76      0.72      0.72      20000
```

Hình 3.5: Kết quả đánh giá theo các chỉ số của mô hình LGBM.

Nhận xét:

- Hiệu suất tổng thể: Với accuracy 72% và F1-score 72% (weighted), mô hình này có hiệu suất cao hơn đáng kể so với Logistic Regression (67%) và Random Forest (62%), nhưng thấp hơn một chút so với XGBoost đã tinh chỉnh (76%).
- Cân bằng giữa các lớp: Macro F1-score (0.71) và Weighted F1-score (0.72) khá gần nhau. Điều này cho thấy mô hình không quá thiên vị lớp đa số và hoạt động tương đối ổn định trên cả ba lớp.

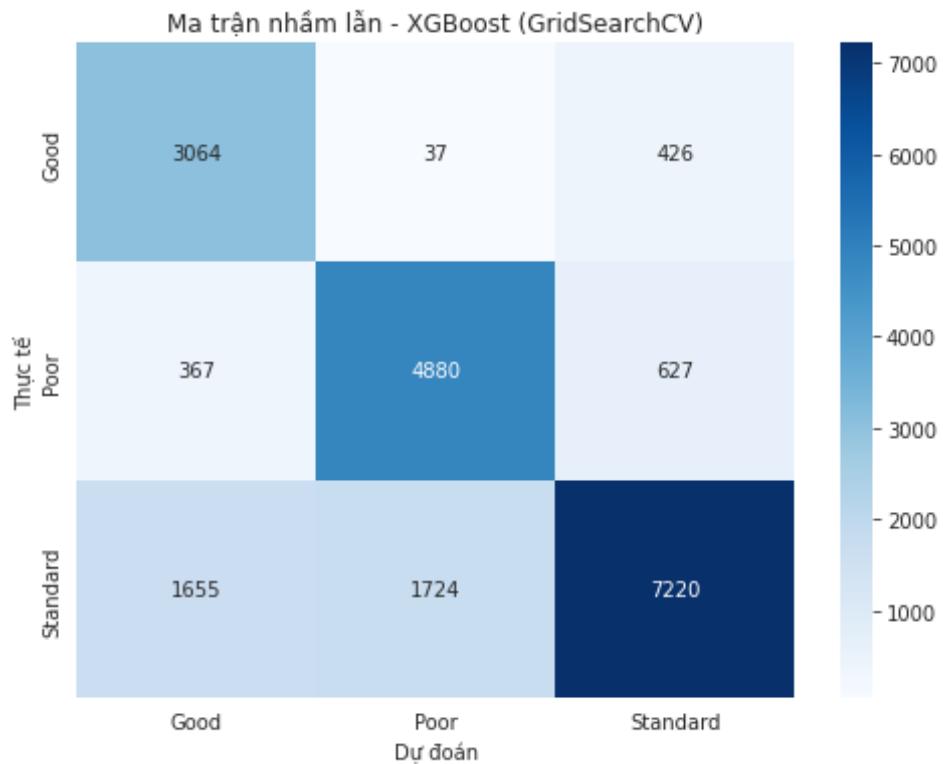
Đánh giá các mô hình

Bảng 3.1: So sánh tổng quan hiệu suất các mô hình trên tập kiểm tra

Mô hình	Accuracy	F1-score (weighted)	Recall (lớp "Poor")
Logistic Regression	67%	0.67	69%
Random Forest	62%	0.62	76%
XGBoost	76%	0.76	83%
LightGBM	75%	0.75	83%

Mô hình XGBoost cho kết quả tốt hơn các mô hình còn lại. Dựa trên các kết quả đánh giá trong quá trình thử nghiệm (bao gồm Logistic Regression, Random Forest, XGBoost và LightGBM), mô hình **XGBoost** được lựa chọn làm mô hình cuối cùng để triển khai.

3.3 Thống kê và phân tích lỗi



Hình 3.6: Ma trận nhầm lẫn của mô hình XGB

Thống kê tổng quan

1. Các dự đoán đúng (Đường chéo chính):

- Lớp **Good**: 3,064 mẫu
- Lớp **Poor**: 4,880 mẫu
- Lớp **Standard**: 7,220 mẫu

Tổng số dự đoán đúng: $3064 + 4880 + 7220 = 15,164$.

2. Tổng số mẫu trong tập dữ liệu:

- Tổng hàng "Good": $3064 + 37 + 426 = 3,527$
- Tổng hàng "Poor": $367 + 4880 + 627 = 5,874$
- Tổng hàng "Standard": $1655 + 1724 + 7220 = 10,599$

Tổng số mẫu: $3527 + 5874 + 10599 = 19,990$.

3. Độ chính xác tổng thể (Overall Accuracy):

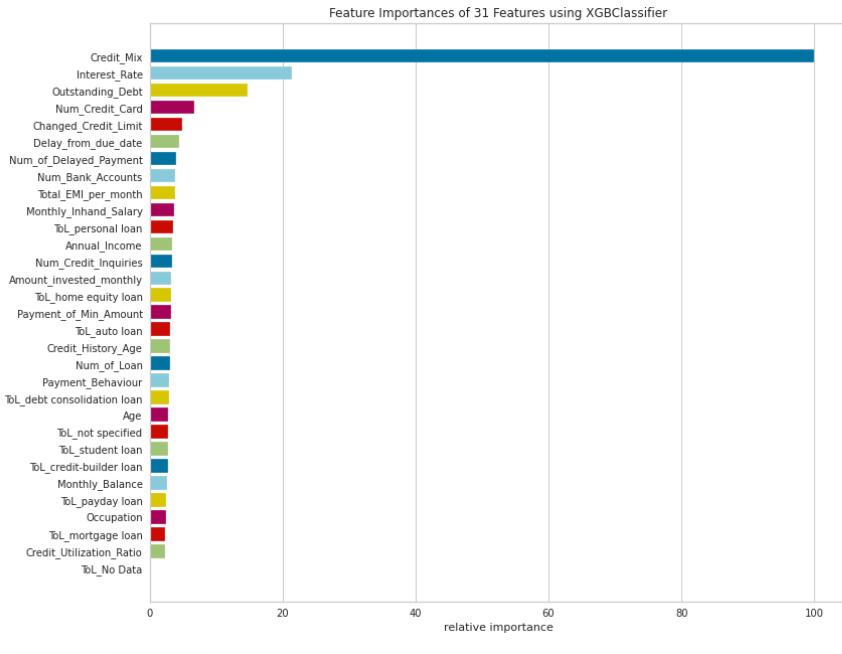
$$\text{Accuracy} = \frac{\text{Tổng số dự đoán đúng}}{\text{Tổng số mẫu}} = \frac{15,164}{19,990} \approx 75.86\%$$

Phân tích các lỗi nghiêm trọng nhất

Các lỗi lớn nhất của mô hình xảy ra khi nó phân loại nhầm lớp "Standard".

1. **Nhầm lẫn Standard → Poor (1,724 lần):** Đây là lỗi phổ biến nhất. Ranh giới giữa hai lớp này có vẻ không rõ ràng đối với mô hình.
2. **Nhầm lẫn Standard → Good (1,655 lần):** Lỗi nghiêm trọng thứ hai, cho thấy lớp "Standard" có các đặc trưng chồng chéo lên cả hai lớp còn lại.
3. **Nhầm lẫn Poor → Standard (627 lần):** Mô hình có xu hướng "an toàn" khi phân loại một số trường hợp "Poor" thành "Standard".
4. **Lỗi ít gặp nhất (Good → Poor, chỉ 37 lần):** Một điểm rất tích cực. Mô hình phân biệt rất tốt giữa hai thái cực "Good" và "Poor".

3.4 Phân tích tầm quan trọng của đặc trưng



Hình 3.7: Tầm quan trọng của các đặc trưng theo mô hình XGBoost.

Nhận xét:

- **Credit_Mix** là đặc trưng có ảnh hưởng lớn nhất, cho thấy sự đa dạng và loại hình tín dụng của khách hàng là yếu tố dự báo mạnh mẽ nhất.
- Các yếu tố phản ánh rủi ro và hành vi trực tiếp như **Interest_Rate** (lãi suất hiện tại), **Outstanding_Debt** (nợ tồn đọng), và **Delay_from_due_date** (số ngày trễ hạn) đều nằm trong nhóm các đặc trưng quan trọng nhất.
- Một phát hiện đáng chú ý là các đặc trưng về thu nhập (**Annual_Income**) lại có tầm quan trọng thấp hơn nhiều so với các đặc trưng về hành vi tín dụng. Điều này khẳng định rằng, trong bài toán này, cách một cá nhân quản lý các khoản nợ của mình là yếu tố quyết định, hơn cả năng lực tài chính của họ.

Chương 4

Cải tiến mô hình

4.1 Kiến trúc mô hình 1 - Logistic Regression

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	
					Accuracy	Precision	Recall	F1-score		
1	LR	LR	Giá trị hàm mất mát giảm không quá 0.0001 hoặc đạt số vòng lặp tối đa là 10000	Sử dụng GridSearch với 5-fold Cross validation	<code>penalty = 'l2', C = 1.0, solver = 'lbfgs', class_weight='balanced'</code>	0.67	0.64	0.69	0.66	Mô hình baseline
2		LR Saga	Giá trị hàm mất mát giảm không quá 0.0001 hoặc đạt số vòng lặp tối đa là 10000	Sử dụng GridSearch với 5-fold Cross validation	<code>penalty='l1', C=1.0, solver='saga', class_weight='balanced'</code>	0.67	0.64	0.69	0.66	Best param of LR with Recall
3		LR Liblinear	Giá trị hàm mất mát giảm không quá 0.0001 hoặc đạt số vòng lặp tối đa là 10000	Sử dụng GridSearch với 5-fold Cross validation	<code>penalty='l1', C=1.0, solver='liblinear', class_weight='balanced'</code>	0.67	0.64	0.69	0.65	Best param of LR with Precision

Hình 4.1: Mô tả chi tiết các mô hình cải tiến từ mô hình LR.

4.1.1 LR Saga

Mô hình này được cải tiến từ mô hình LR baseline bằng cách thay `penalty = 'l1'`, `solver = 'saga'`, ta được kết quả như trong bảng mô tả chi tiết mô hình 4.1. Ta thấy kết quả không thay đổi.

- Mục tiêu:** Tìm ra bộ tham số tốt nhất để tối đa hóa chỉ số **Recall**.
- Siêu tham số:** Sử dụng `solver='saga'` và `penalty='l1'`.
- Kết quả:** $\text{Recall} = 0.69$, $\text{F1-score} = 0.66$.

4.1.2 LR Liblinear

Mô hình này được cải tiến từ mô hình LR baseline bằng cách thay `penalty = 11`, `solver = liblinear`, ta được kết quả như trong bảng mô tả chi tiết mô hình 4.1.

- Mục tiêu:** Tìm ra bộ tham số tốt nhất để tối đa hóa chỉ số **Precision**.
- Siêu tham số:** Sử dụng `solver='liblinear'` và `penalty='11'`.
- Kết quả:** Precision = 0.64, F1-score = 0.65.

4.2 Kiến trúc mô hình 2 - Random Forest

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
						Accuracy	Precision	Recall	F1-score	
4	RF		Mô hình huấn luyện đủ 100 cây với độ sâu không giới hạn	Sử dụng GridSearch với 5-fold Cross validation	<code>class_weight': 'balanced', 'criterion': 'gini', 'max_depth': None, 'min_impurity_decrease': 0, 'n_estimators': 100, 'oob_score': False</code>	0.82	0.84	0.8	0.82	
5	RF	RF1	Mô hình huấn luyện đủ 150 cây với độ sâu không giới hạn	Sử dụng GridSearch với 5-fold Cross validation	<code>class_weight': 'balanced', 'criterion': 'gini', 'max_depth': None, 'min_impurity_decrease': 0, 'n_estimators': 150, 'oob_score': False</code>	0.82	0.8	0.85	0.83	Best param for RF
6		RF2	Mô hình huấn luyện đủ 200 cây với độ sâu không giới hạn	Sử dụng GridSearch với 5-fold Cross validation	<code>class_weight': 'balanced', 'criterion': 'gini', 'max_depth': None, 'min_impurity_decrease': 0, 'n_estimators': 200, 'oob_score': False</code>	0.82	0.8	0.84	0.82	

Hình 4.2: Mô tả chi tiết các mô hình cải tiến từ mô hình RF.

4.2.1 RF1

- Siêu tham số:** `n_estimators = 150`.
- Nhận xét:** Khi tăng số cây, Precision giảm nhẹ nhưng **Recall tăng mạnh mẽ** từ 80% lên 85%, giúp **F1-score đạt mức cao** ở 83%. Đây là cấu hình cân bằng và hiệu quả.

4.2.2 RF2

- Siêu tham số:** `n_estimators = 200`.

- Nhận xét:** Việc tăng thêm số cây **không mang lại cải thiện**. Hiệu suất giảm nhẹ so với mốc 150 cây, cho thấy hiện tượng **hiệu suất giảm dần**.

4.3 Kiến trúc mô hình 3 - XGBoost

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	
					Accuracy	Precision	Recall	F1-score		
7	XGBoost	Mô hình huấn luyện đủ 100 cây, mỗi cây đạt độ sâu giới hạn là 6	Sử dụng GridSearch với 5-fold Cross validation	<code>n_estimators=100, max_depth=6, learning_rate=0.3</code>	0.76	0.74	0.83	0.78		
8		XGBoost1	Mô hình huấn luyện đủ 150 cây, mỗi cây đạt độ sâu giới hạn là 6	Sử dụng GridSearch với 5-fold Cross validation	<code>n_estimators=150, max_depth=6, learning_rate=0.3</code>	0.78	0.75	0.85	0.8	Best param of XGBoost
9		XGBoost2	Mô hình huấn luyện đủ 200 cây, mỗi cây đạt độ sâu giới hạn là 6	Sử dụng GridSearch với 5-fold Cross validation	<code>n_estimators=200, max_depth=6, learning_rate=0.3</code>	0.79	0.76	0.86	0.81	

Hình 4.3: Mô tả chi tiết các mô hình cải tiến từ mô hình XGB.

4.3.1 XGBoost1

- Siêu tham số:** `n_estimators = 150`.
- Nhận xét:** Tất cả các chỉ số đều được cải thiện so với mô hình cơ sở.

4.3.2 XGBoost2

- Siêu tham số:** `n_estimators = 200`.
- Nhận xét:** Mô hình này tiếp tục cải thiện hiệu suất trên mọi mặt trận, đạt điểm F1-score, Recall, Precision và Accuracy cao nhất trong ba thử nghiệm.

4.4 Kiến trúc mô hình 4 - LightGBM

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
					Accuracy	Precision	Recall	F1-score	
10	LGBM	n_estimators=1000, early_stopping_rounds=50, random_state=42		max_depth=-1, num_leaves=31, min_child_samples=20, learning_rate=0.1	0.72	0.76	0.72	0.72	
11	LGBM1	n_iter=20	Sử dụng Randomized Search CV với cv = 3	colsample_bytree': 0.978, 'learning_rate': 0.0827, 'max_depth': 9, 'n_estimators': 299, 'num_leaves': 28, 'subsample': 0.759	0.75	0.79	0.75	0.75	
12	LGBM2	Dừng khi loại bỏ các đặc trưng có importance < 1% tổng trọng số	Sử dụng feature importance-based stopping	loại bỏ 11 đặc trưng ít quan trọng	0.75	0.78	0.75	0.75	

Hình 4.4: Mô tả chi tiết các mô hình cải tiến từ mô hình LGBM.

4.4.1 LGBM1

- Thiết lập:** Kết quả từ RandomizedSearchCV.
- Nhận xét:** Có một bước nhảy vọt về hiệu suất. Điều này chứng tỏ tầm quan trọng của việc tinh chỉnh siêu tham số.

4.4.2 LGBM2

- Thiết lập:** Dựa trên mô hình LGBM1, loại bỏ 11 đặc trưng có tầm quan trọng thấp.
- Nhận xét:** Một kết quả rất tích cực. Mô hình trở nên **đơn giản và nhanh hơn** mà không làm giảm hiệu suất tổng thể. Đây là một sự đánh đổi rất tốt.

Chương 5

Đóng gói mô hình

5.1 Bài toán nghiệp vụ

Mô hình này được xây dựng để giải quyết bài toán **tự động hóa đánh giá sơ bộ điểm tín dụng cho khách hàng mới** khi họ đăng ký các sản phẩm tài chính như vay tiêu dùng, thẻ tín dụng, hoặc các khoản vay cá nhân khác. Mục tiêu là cung cấp một kết quả phân loại nhanh chóng (Good, Poor, Standard) để hỗ trợ chuyên viên tín dụng, từ đó:

- Giảm thời gian xử lý hồ sơ.
- Đảm bảo tính nhất quán trong quyết định.
- Phân luồng hồ sơ hiệu quả: các hồ sơ "Good" có thể được ưu tiên xử lý nhanh, trong khi các hồ sơ "Poor" cần xem xét để tránh rủi ro.

5.2 Các chỉ số đánh giá mô hình đủ điều kiện để ứng dụng vào thực tế

- **Recall (Độ nhạy):** Đây là điểm mạnh nhất của mô hình và là yếu tố quan trọng nhất cho bài toán sàng lọc.
 - **Recall lớp Poor là 84%:** Điều này có nghĩa là mô hình có khả năng **phát hiện đúng 84%** các khách hàng thực sự có rủi ro tín dụng cao. Trong thực tế, việc bỏ sót một khách hàng rủi ro có thể dẫn đến tổn thất tài chính lớn. Tỷ lệ phát hiện cao này giúp giảm thiểu đáng kể rủi ro cho tổ chức.

- **Recall lớp Good là 88%**: Mô hình xác định được 88% các khách hàng thực sự có điểm tín dụng tốt. Điều này giúp bộ phận kinh doanh không bỏ lỡ các khách hàng tiềm năng và có thể nhanh chóng phê duyệt hồ sơ cho họ, nâng cao trải nghiệm khách hàng và tăng tính cạnh tranh.
- **Precision (Độ chính xác dự đoán):**
 - **Precision lớp Good là 59%**: Đây là một chỉ số cần lưu ý. Nó có nghĩa là trong số các khách hàng được mô hình dự đoán là "Tốt", chỉ có 59% thực sự là "Tốt". 41% còn lại là "Trung bình" hoặc "Kém". Điều này khẳng định vai trò của mô hình là một **công cụ hỗ trợ sàng lọc sơ bộ**, chứ không phải là công cụ ra quyết định cuối cùng. Các hồ sơ được gắn nhãn Good sẽ được ưu tiên xem xét trước, nhưng vẫn cần sự thẩm định của chuyên viên tín dụng.
 - **F1-score và Accuracy tổng thể**: Điểm F1-score trung bình (weighted) là 0.76, cho thấy sự cân bằng tốt giữa Precision và Recall. Độ chính xác tổng thể 76% là một mức hiệu suất rất khả quan đối với một hệ thống hỗ trợ ra quyết định.

Kết luận về tính thực tiễn

Dựa trên các phân tích trên, mô hình XGBoost cuối cùng hoàn toàn **đủ điều kiện để đưa vào ứng dụng thực tế** với vai trò là một hệ thống sàng lọc và phân luồng hồ sơ tín dụng sơ bộ. Mô hình vượt trội hơn hẳn so với cách tiếp cận thông thường và đặc biệt mạnh mẽ trong việc nhận diện các nhóm khách hàng quan trọng (Good, Poor), giúp tối ưu hóa quy trình nghiệp vụ và quản lý rủi ro hiệu quả.

5.3 Giao diện demo chương trình

Ứng dụng cho phép người dùng nhập trực tiếp thông tin của khách hàng và nhận lại kết quả phân loại điểm tín dụng một cách nhanh chóng, giúp giải quyết các bài toán:

- **Tự động hóa sơ bộ**: Giảm thiểu các bước nhập liệu và tính toán thủ công, tiết kiệm thời gian cho chuyên viên.
- **Hỗ trợ ra quyết định**: Cung cấp một đánh giá khách quan, dựa trên dữ liệu để tham khảo bên cạnh các quy tắc nghiệp vụ truyền thống.

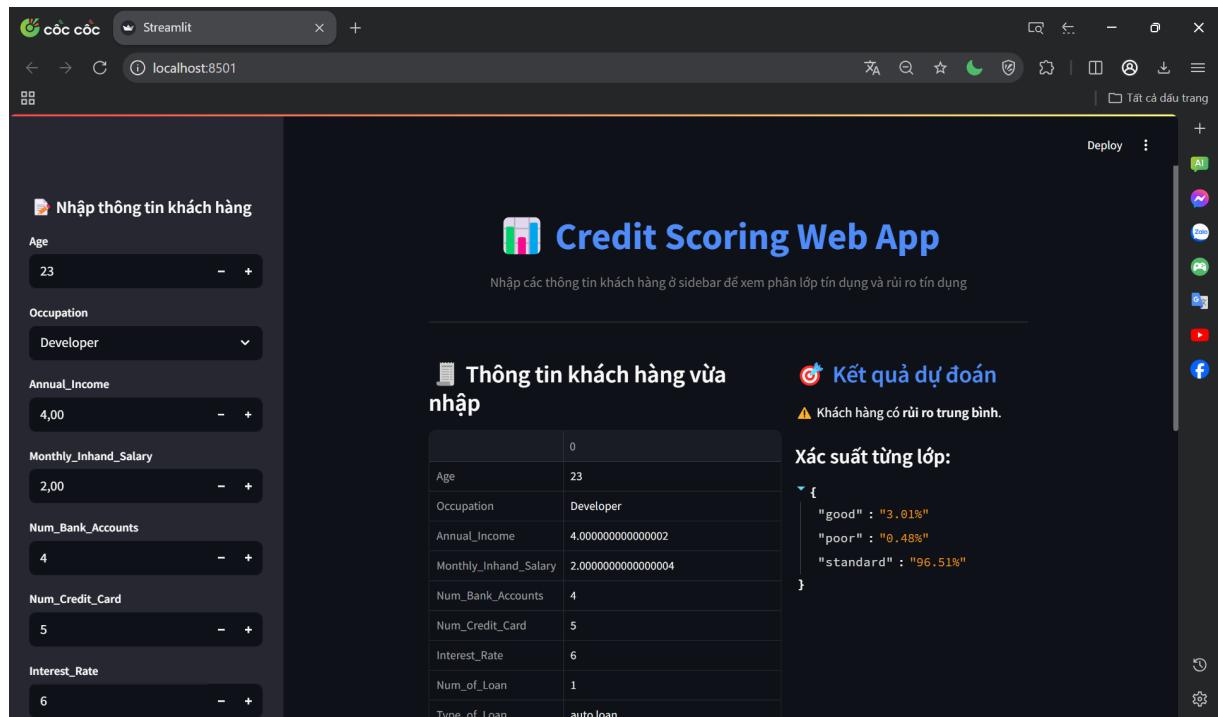
- **Phân luồng hồ sơ:** Giúp nhanh chóng xác định các khách hàng tiềm năng (Good) hoặc rủi ro cao (Poor) để có hướng xử lý phù hợp.

5.3.1 Xây dựng giao diện

- Ngôn ngữ lập trình: Python.
- Thư viện xây dựng giao diện web: Streamlit - một framework cho phép xây dựng nhanh các ứng dụng web tương tác trực tiếp từ các kịch bản dữ liệu.
- Thư viện máy học: XGBoost để thực hiện dự đoán.
- Thư viện xử lý dữ liệu: Pandas và NumPy.

5.3.2 Giao diện và luồng hoạt động của ứng dụng

Giao diện ứng dụng được thiết kế tối giản và tập trung vào người dùng cuối, bao gồm hai khu vực chính như trong Hình 5.1.



Hình 5.1: Giao diện chính.

Người dùng (chuyên viên tín dụng) thực hiện theo các bước sau:

1. **Nhập liệu:** Tại thanh bên (sidebar) bên trái, người dùng nhập các thông tin đầu vào của khách hàng. Các trường thông tin này tương ứng với các đặc trưng quan trọng mà mô hình sử dụng để dự đoán, ví dụ:

- Tuổi (Age)
- Nghề nghiệp (Occupation)
- Thu nhập hàng năm (Annual_Income)
- Lãi suất (Interest_Rate)
- ... và các thông tin khác.

2. **Xác nhận và dự đoán:**

- **Khu vực hiển thị thông tin:** Ở giữa màn hình, ứng dụng hiển thị lại một bảng tóm tắt các thông tin vừa được nhập để người dùng kiểm tra lại.
- **Khu vực kết quả:** Ngay sau khi thông tin được nhập đầy đủ, mô hình sẽ tự động xử lý và trả về kết quả dự đoán ở cột bên phải.

3. **Nhận kết quả:** Kết quả được trình bày một cách trực quan và dễ hiểu:

- **Phân loại rủi ro:** Một kết luận rõ ràng được đưa ra. Ví dụ trong Hình 5.1, kết quả là "*Khách hàng có rủi ro trung bình*".
- **Xác suất từng lớp:** Để cung cấp thông tin chi tiết hơn cho chuyên viên, ứng dụng hiển thị xác suất mà mô hình tính toán cho từng lớp. Ví dụ:
 - Good: 3.01%
 - Poor: 0.48%
 - Standard: 96.51%

Thông tin này rất hữu ích, vì nó cho thấy độ chắc chắn của mô hình trong dự đoán. Một khách hàng được dự đoán là "Standard" với xác suất 96.51% sẽ đáng tin cậy hơn nhiều so với một khách hàng khác cũng được dự đoán là "Standard" nhưng với xác suất chỉ 55%.

5.3.3 Đóng gói giao diện chương trình

Giao diện demo của chương trình đã được đóng gói thành `run_app.exe`.

Name	Status	Date modified	Type	Size
app	⟳	2025-06-08 12:56 PM	Python Source File	6 KB
pipeline.pkl	⟳	2025-06-08 12:56 PM	PKL File	4,250 KB
requirements	⟳	2025-06-08 12:56 PM	Text Document	1 KB
run_app	⟳	2025-06-08 1:56 PM	Application	7,073 KB
xgb_model.pkl	⟳	2025-06-08 12:56 PM	PKL File	2,505 KB

Hình 5.2: Đóng gói giao diện.

5.4 Làm slide báo cáo và thuyết trình

Em đã làm slide báo cáo và thuyết trình trên lớp.

Kết luận

Báo cáo đã trình bày toàn bộ quá trình giải quyết bài toán phân loại điểm tín dụng, từ khâu thu thập, tiền xử lý dữ liệu một cách chi tiết, lựa chọn và huấn luyện mô hình cho đến đánh giá và cải tiến hiệu suất.

- **Tiền xử lý dữ liệu toàn diện:** Đã thực hiện thành công việc làm sạch, xử lý các giá trị thiếu, chuyển đổi và chuẩn hóa dữ liệu. Đặc biệt, đã phân tích và giải quyết hiệu quả vấn đề mất cân bằng dữ liệu thông qua kỹ thuật gán trọng số lớp (class weight) và kỹ thuật SMOTE.
- **Xây dựng và đánh giá mô hình:** Đã xây dựng, huấn luyện và so sánh một cách có hệ thống 4 thuật toán khác nhau (Logistic Regression, Random Forest, XGBoost, LightGBM) để tìm ra giải pháp tối ưu.
- **Tinh chỉnh và lựa chọn mô hình tốt nhất:** Dựa trên các tiêu chí nghiệp vụ, mô hình **XGBoost** đã được lựa chọn và tinh chỉnh để đạt hiệu suất cao nhất, với độ chính xác tổng thể **76%**, Recall cho lớp "Poor" là **83%** và lớp "Good" là **88%**.
- **Đóng gói và xây dựng giao diện demo:** Không chỉ dừng lại ở mô hình, một giao diện người dùng trực quan đã được xây dựng để minh họa cách sản phẩm hoạt động trong thực tế, cho phép người dùng nhập thông tin và nhận kết quả phân loại ngay lập tức.

Với mô hình XGBoost mạnh mẽ và giao diện demo hoàn chỉnh, sản phẩm đã chứng minh được tính khả thi và sẵn sàng để triển khai. Đây là một công cụ hỗ trợ quyết định giá trị, giúp các tổ chức tài chính tối ưu hóa quy trình thẩm định tín dụng, giảm thiểu rủi ro và nâng cao hiệu quả kinh doanh.