

BỘ NÔNG NGHIỆP VÀ PHÁT TRIỂN NÔNG THÔN

PHÂN HIỆU TRƯỜNG ĐẠI HỌC THỦY LỢI

BỘ MÔN CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN KHAI PHÁ DỮ LIỆU

TÊN ĐỀ TÀI :

DỰ ĐOÁN KHẢ NĂNG KHÁCH HÀNG RỜI BỎ DỊCH VỤ

Giảng viên hướng dẫn: Th.S Vũ Thị Hạnh

Lớp: S25-64CNTT

Sinh viên thực hiện	Mã sinh viên
Phan Minh Nhật	2251068222
Lê Võ Nhựt Nguyên	2251068218

TP Hồ Chí Minh, ngày 20 tháng 10 năm 2025

LỜI MỞ ĐẦU

Trong thời đại công nghệ thông tin và dữ liệu phát triển mạnh mẽ, việc khai thác tri thức từ dữ liệu trở thành yếu tố quan trọng giúp các tổ chức và doanh nghiệp ra quyết định chính xác, nâng cao năng lực cạnh tranh. Khai phá dữ liệu (Data Mining) là một lĩnh vực thuộc khoa học dữ liệu, tập trung vào việc tìm kiếm các mô hình, quy luật tiềm ẩn và mối quan hệ có ý nghĩa trong các tập dữ liệu lớn.

Một trong những bài toán ứng dụng phổ biến của khai phá dữ liệu là dự đoán khả năng khách hàng rời bỏ dịch vụ (Customer Churn Prediction). Đây là vấn đề thực tiễn quan trọng đối với các công ty cung cấp dịch vụ, đặc biệt trong lĩnh vực viễn thông, ngân hàng hay thương mại điện tử. Việc phát hiện sớm nhóm khách hàng có xu hướng ngừng sử dụng dịch vụ giúp doanh nghiệp đưa ra chiến lược chăm sóc phù hợp, giảm tỷ lệ rời bỏ và tối ưu hóa lợi nhuận.

Đề tài “Khai phá dữ liệu dự đoán khả năng khách hàng rời bỏ dịch vụ” được thực hiện với mục tiêu áp dụng các kiến thức đã học trong môn Khai phá dữ liệu, bao gồm: tiền xử lý dữ liệu, phân tích khám phá dữ liệu, xây dựng mô hình học máy và đánh giá hiệu quả mô hình. Trong quá trình thực hiện, các thư viện phổ biến của Python như pandas, numpy, matplotlib, seaborn, và scikit-learn được sử dụng để xử lý và phân tích dữ liệu trên nền tảng Google Colab.

Kết quả của đề tài không chỉ giúp người học hiểu rõ hơn về quy trình khai phá dữ liệu, mà còn thể hiện khả năng ứng dụng lý thuyết vào thực tiễn, góp phần minh họa vai trò của khoa học dữ liệu trong việc hỗ trợ ra quyết định kinh doanh. Báo cáo này trình bày toàn bộ quá trình từ xử lý dữ liệu, xây dựng mô hình, đến đánh giá và phân tích kết quả dự đoán khách hàng rời bỏ dịch vụ.

MỤC LỤC

CHƯƠNG 1. GIỚI THIỆU TỔNG QUAN	5
1.1. Giới thiệu đề tài.....	5
1.2. Mục tiêu nghiên cứu	5
1.3. Phạm vi và giới hạn đề tài	6
1.4. Phương pháp thực hiện	6
1.4.1 Thu thập dữ liệu.....	6
1.4.2 Tiền xử lý dữ liệu (Data Preprocessing):	7
1.4.3 Khai phá dữ liệu – Phân tích mô tả (Data Exploration):	7
1.4.4 Xây dựng mô hình.....	16
1.4.5 Đánh giá mô hình	17
1.5. Ý nghĩa thực tiễn của đề tài	18
CHƯƠNG 2: MỤC TIÊU BÀI TOÁN ĐẶT RA.....	18
2.1 Khái niệm về khai phá dữ liệu	18
2.2. Bài toán dự đoán khách hàng rời bỏ (Customer Churn Prediction)	18
2.3. Tiền xử lý dữ liệu (Data Preprocessing)	19
2.5. Đánh giá mô hình (Model Evaluation).....	19
CHƯƠNG 3: MÔ TẢ DỮ LIỆU VÀ CÁC BƯỚC TIỀN XỬ LÝ.....	19
3.1 Mô tả dữ liệu.....	20
3.2 Các bước tiền xử lý dữ liệu.....	20
CHƯƠNG 4: PHƯƠNG PHÁP KHAI PHÁ DỮ LIỆU / MÔ HÌNH ML	21
4.1 Tiền xử lý dữ liệu:	21
4.2 Phân tích khám phá dữ liệu (EDA):.....	21
4.3 Xây dựng và đánh giá mô hình ML:	22
4.4 Phân cụm khách hàng (Clustering):	23
4.5 Áp dụng mô hình tốt nhất:.....	23

CHƯƠNG 5: KẾT QUẢ VÀ ĐÁNH GIÁ MÔ HÌNH	23
5.1 Đánh giá loại vấn đề	24
5.2 Huấn luyện mô hình	24
5.2.1 Mô hình Logistic Regression	24
5.2.2 Mô hình XGBoost	28
5.2.2.1 Đánh giá mô hình	29
5.3 Cân bằng dữ liệu	30
5.3.1 Cân bằng dữ liệu với phương pháp Nearest	30
5.3.2 Kỹ thuật Oversampling với SMOTE	35
5.4 Tuning các mô hình	38
5.5 Threshhold mô hình	45
5.5.1 Mô hình XGBoost	45
5.5.2 Mô hình Logictic Regissterion	47
5.6 Xác thực chéo (cross-validation)	48
5.7 Phân cụm khách hàng	49
5.8 Giao diện web	51
CHƯƠNG 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	51
6.1 Kết luận	51
6.2 Hướng phát triển	53
CHƯƠNG 7: TÀI LIỆU THAM KHẢO	54

CHƯƠNG 1. GIỚI THIỆU TỔNG QUAN

1.1. Giới thiệu đề tài

Trong thời đại dữ liệu phát triển mạnh mẽ, việc khai thác và phân tích dữ liệu giúp doanh nghiệp hiểu rõ hơn về hành vi, nhu cầu và xu hướng của khách hàng. Một trong những bài toán phổ biến trong lĩnh vực Khai phá dữ liệu (Data Mining) là phân tích và dự đoán khách hàng rời bỏ dịch vụ (Customer Churn Prediction).

Khi một khách hàng ngừng sử dụng dịch vụ, doanh nghiệp không chỉ mất đi nguồn doanh thu mà còn phải tốn thêm chi phí để thu hút khách hàng mới. Do đó, việc dự đoán sớm khách hàng có khả năng rời bỏ giúp doanh nghiệp chủ động đưa ra chính sách giữ chân phù hợp.

Trong đề tài này, người học tiến hành khai phá dữ liệu khách hàng của một công ty viễn thông để xác định các yếu tố ảnh hưởng đến quyết định rời bỏ và xây dựng mô hình dự đoán bằng các kỹ thuật trong môn học Khai phá dữ liệu.

Quá trình thực hiện được triển khai trên Google Colab, sử dụng ngôn ngữ Python cùng các thư viện như pandas, scikit-learn, matplotlib và seaborn.

1.2. Mục tiêu nghiên cứu

Đề tài hướng đến các mục tiêu cụ thể sau:

- Tìm hiểu và làm sạch dữ liệu khách hàng.
- Phân tích, nhận diện các yếu tố ảnh hưởng đến khả năng rời bỏ dịch vụ.
- Áp dụng các mô hình học máy như Logistic Regression, Random Forest, v.v.
- So sánh hiệu quả giữa các mô hình dựa trên các chỉ số: Accuracy, Precision, Recall, F1-score, ROC-AUC.
- Đưa ra mô hình dự đoán tốt nhất giúp doanh nghiệp có thể nhận diện sớm nhóm khách hàng có khả năng rời bỏ.

1.3. Phạm vi và giới hạn đề tài

* Phạm vi:

- Dữ liệu được sử dụng trong đề tài là tập dữ liệu khách hàng của một công ty viễn thông (Telco Customer Churn).
- Quá trình xử lý và mô hình hóa được thực hiện hoàn toàn bằng Python trên Google Colab.

* Giới hạn:

- Đề tài chỉ tập trung vào bài toán phân loại nhị phân (churn hoặc không churn).
- Không triển khai giao diện ứng dụng thực tế.
- Việc đánh giá mô hình chỉ dựa trên dữ liệu có sẵn, không thu thập dữ liệu thực tế.

1.4. Phương pháp thực hiện

Quá trình nghiên cứu và triển khai đề tài được thực hiện qua các bước sau:

1.4.1 Thu thập dữ liệu

Sử dụng tập dữ liệu mẫu từ Kaggle hoặc tệp CSV có sẵn (Telco Customer Churn).

```
# Imports & Load dữ liệu
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from IPython.display import display

# Đường dẫn khả dĩ
paths = ["/content/sample_data/WA_Fn-UseC_-Telco-Customer-Churn.csv",]

data_path = None
for p in paths:
    if os.path.exists(p):
        data_path = p
        break

if data_path is None:
    print("Không tìm thấy file CSV tự động. Vui lòng upload file 'WA_Fn-UseC_-Telco-Customer-Churn.csv' vào Colab (bên trái -> Files -> Upload) hoặc thay đổi đường dẫn.")
else:
    print("Đúng file:", data_path)
    df = pd.read_csv(data_path)
    print("Kích thước ban đầu:", df.shape)
    display(df.head())
```

1.4.2 Tiền xử lý dữ liệu (Data Preprocessing):

- Kiểm tra dữ liệu bị thiếu, sai lệch.
- Loại bỏ các giá trị NaN.
- Mã hóa các biến phân loại bằng OneHotEncoder hoặc LabelEncoder.
- Chia dữ liệu thành tập huấn luyện và kiểm thử.

```
# Xử lý dữ liệu cơ bản
def preprocess_inputs(df):
    data = df.copy()

    # Xóa cột không cần thiết
    if 'customerID' in df.columns:
        data = data.drop(columns=['customerID'])

    # Xử lý Total Charges rỗng và chuyển sang kiểu số
    data['TotalCharges'] = pd.to_numeric(data['TotalCharges'], errors='coerce')

    if df["TotalCharges"].isnull().sum() > 0:
        data = data.dropna(subset=['TotalCharges']).reset_index(drop=True)

    # Chuẩn whitespace cho các cột object
    for c in data.select_dtypes(include='object').columns: # Changed df to data here
        data[c] = data[c].str.strip()

    # Chuyển biến mục tiêu Churn -> 0/1
    data['Churn'] = data['Churn'].astype(str).map({'Yes': 1, 'No': 0})

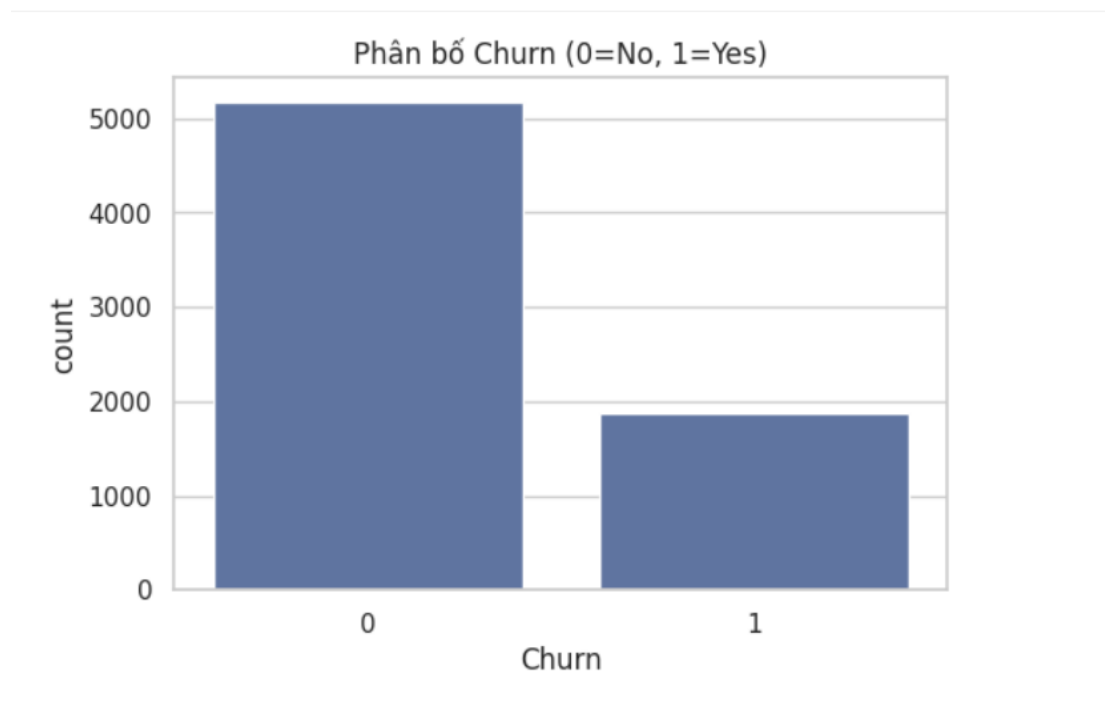
    return data
```

1.4.3 Khai phá dữ liệu – Phân tích mô tả (Data Exploration):

Thống kê mô tả dữ liệu.

Phân tích mối quan hệ giữa các thuộc tính và biến mục tiêu (Churn).

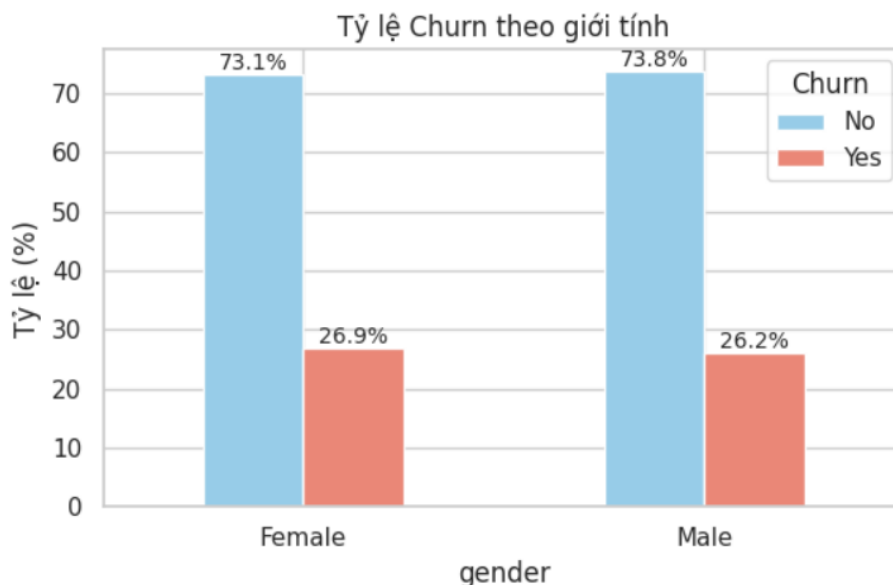
Vẽ biểu đồ minh họa phân bố dữ liệu, tỷ lệ rời bỏ, v.v.



Biểu đồ **Phân bố Churn (0=No, 1=Yes)** cho thấy rõ vấn đề mất cân bằng lớp (Class Imbalance) nghiêm trọng: Khách hàng không rời bỏ (Non-Churn/0) chiếm đa số, khoảng 73.5%, trong khi Khách hàng rời bỏ (Churn/1) chỉ chiếm thiểu số, khoảng 26.5%.

Sự mất cân bằng này có ý nghĩa quan trọng:

- **Độ chính xác (Accuracy) không đáng tin cậy:** Do lớp Churn là thiểu số, ta không thể dùng Accuracy làm chỉ số đánh giá chính.
- **Ưu tiên Đánh giá:** Báo cáo phải tập trung vào các chỉ số cho lớp thiểu số như Recall (Độ nhạy) cho lớp Churn (1) và AUC-ROC để đánh giá khả năng mô hình "bắt" được khách hàng có nguy cơ cao.
- **Chiến lược:** Cần sử dụng kỹ thuật Cân bằng Trọng số Lớp (class_weight='balanced' hoặc scale_pos_weight) trong quá trình huấn luyện mô hình để giải quyết vấn đề mất cân bằng này.

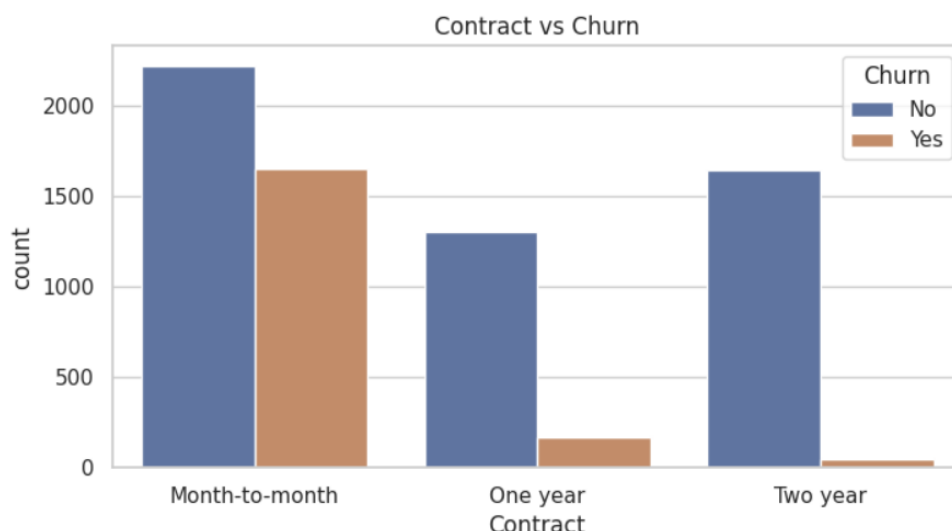


Biểu đồ **Tỷ lệ Churn theo Giới tính** cho thấy sự khác biệt về hành vi rời bỏ dịch vụ giữa nam và nữ là không đáng kể hoặc gần như không tồn tại:

Nhận xét chính: Tỷ lệ Churn (Yes) ở nhóm Female là khoảng 26.9%, trong khi ở nhóm Male là 26.2%. Sự chênh lệch chỉ khoảng 0.7%.

Kết luận: Biến gender (giới tính) dường như không phải là yếu tố dự đoán mạnh cho hành vi rời bỏ khách hàng.

Hành động: Công ty không cần phải xây dựng các chiến lược giữ chân khách hàng (Retention) dựa trên sự phân biệt giới tính. Trong mô hình học máy, biến này được giữ lại nhưng không được kỳ vọng có tầm quan trọng cao.



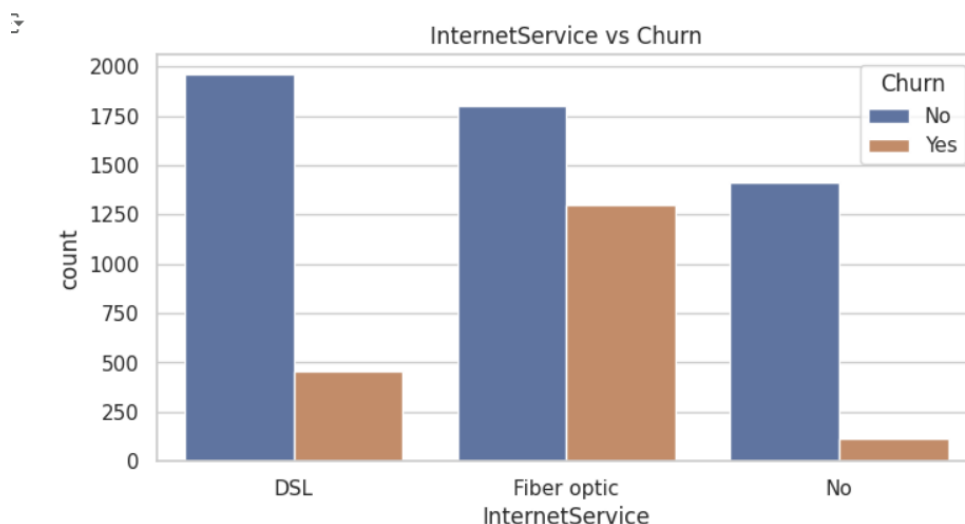
Biểu đồ **Contract vs Churn** (Số lượng rời bỏ theo loại hợp đồng) là một phân tích rất quan trọng và cho thấy mối tương quan mạnh mẽ giữa loại hợp đồng và khả năng Churn.

- **Nhận xét chính:**

Khách hàng có hợp đồng Month-to-month (Tháng-quá-tháng) có số lượng Churn (màu cam) cao nhất tuyệt đối so với hai nhóm hợp đồng còn lại. Đây là nhóm khách hàng rủi ro cao nhất.

Khách hàng có hợp đồng Two year (Hai năm) có số lượng Churn (màu cam) thấp nhất và gần như bị áp đảo hoàn toàn bởi khách hàng không rời bỏ (màu xanh).

- **Kết luận:** Loại hợp đồng là một trong những yếu tố dự đoán mạnh nhất cho hành vi Churn. Khách hàng càng ít cam kết về mặt thời gian (Month-to-month), nguy cơ rời bỏ dịch vụ càng cao.
- **Hành động:** Công ty nên tập trung các ưu đãi giữ chân và khuyến khích để chuyển đổi khách hàng Month-to-month sang các hợp đồng dài hạn (One year hoặc Two year). Đây là chiến lược giữ chân khách hàng hiệu quả nhất.



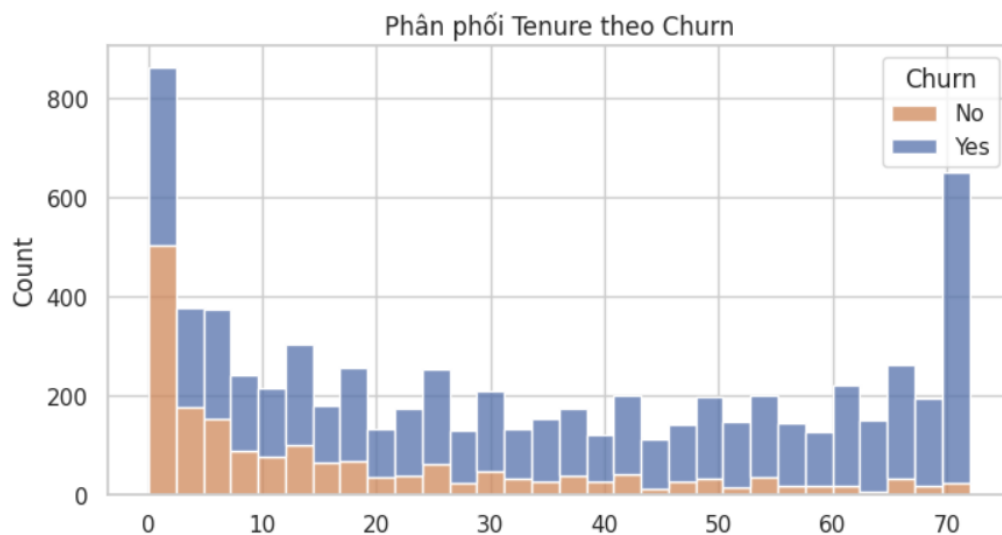
Biểu đồ **Contract vs Churn** (Số lượng rời bỏ theo loại hợp đồng) là một phân tích rất quan trọng và cho thấy mối tương quan mạnh mẽ giữa loại hợp đồng và khả năng Churn.

- **Nhận xét chính:**

Khách hàng có hợp đồng Month-to-month (Tháng-quá-tháng) có số lượng Churn (màu cam) cao nhất tuyệt đối so với hai nhóm hợp đồng còn lại. Đây là nhóm khách hàng rủi ro cao nhất.

Khách hàng có hợp đồng Two year (Hai năm) có số lượng Churn (màu cam) thấp nhất và gần như bị áp đảo hoàn toàn bởi khách hàng không rời bỏ (màu xanh).

- **Kết luận:** Loại hợp đồng là một trong những yếu tố dự đoán mạnh nhất cho hành vi Churn. Khách hàng càng ít cam kết về mặt thời gian (Month-to-month), nguy cơ rời bỏ dịch vụ càng cao.
- **Hành động:** Công ty nên tập trung các ưu đãi giữ chân và khuyến khích để chuyển đổi khách hàng Month-to-month sang các hợp đồng dài hạn (One year hoặc Two year). Đây là chiến lược giữ chân khách hàng hiệu quả nhất.

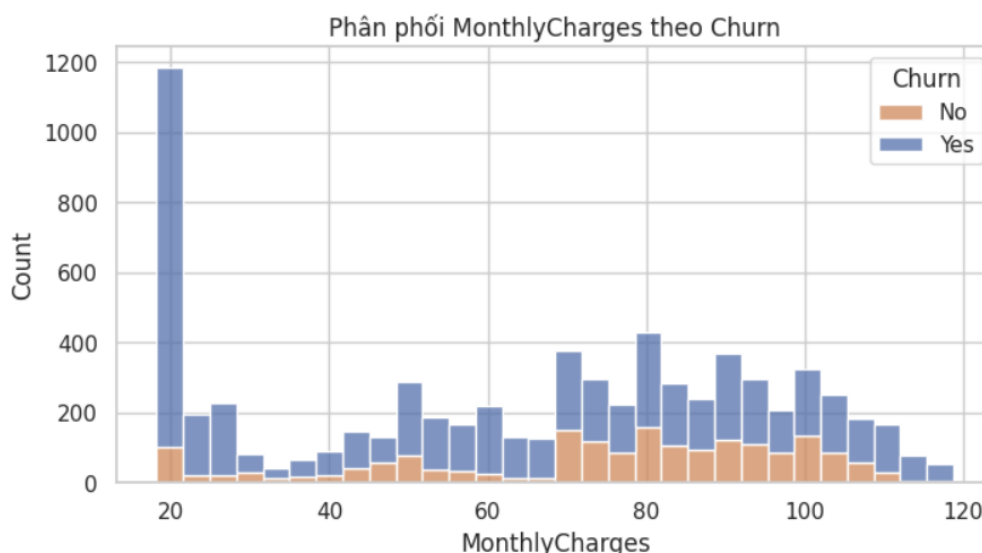


Biểu đồ **Phân phối Tenure theo Churn** (Thời gian gắn bó) là một trong những chỉ số dự đoán mạnh nhất, thể hiện mối quan hệ giữa thời gian sử dụng dịch vụ và khả năng rời bỏ:

Nhận xét chính: Phần lớn số lượng Churn (màu cam) tập trung mạnh ở các cột có **tenure thấp** (khoảng từ 0 đến 10 tháng). Khi tenure tăng (từ 20 tháng trở lên), cột Churn giảm nhanh chóng và cột Non-Churn (màu xanh) chiếm ưu thế áp đảo.

Kết luận: Khách hàng **mới** (tenure thấp) là nhóm **rủi ro cao nhất** và dễ bị mất nhất. Khách hàng có tenure trên 24 tháng (hơn 2 năm) có nguy cơ Churn cực kỳ thấp và là nhóm trung thành.

Hành động: Chiến lược giữ chân cần tập trung mạnh vào giai đoạn 0 đến 12 tháng (Onboarding và cam kết) để giúp khách hàng vượt qua giai đoạn rủi ro cao ban đầu.

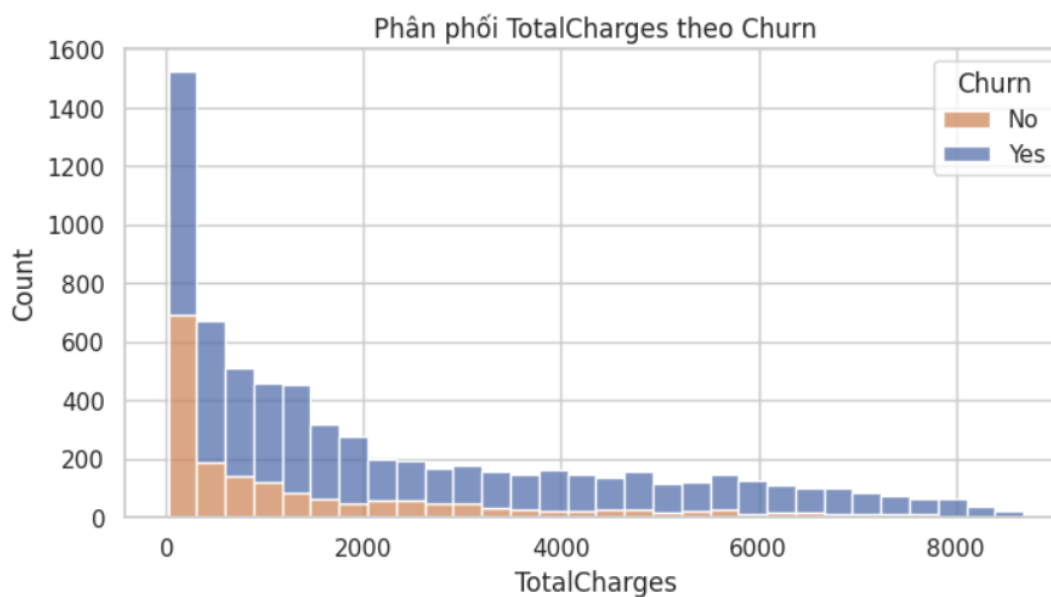


Biểu đồ **Phân phối MonthlyCharges theo Churn** cho thấy mối quan hệ giữa chi phí dịch vụ hàng tháng và khả năng rời bỏ:

Nhận xét chính: Số lượng Churn (màu cam) tập trung đáng kể và chiếm tỷ lệ cao nhất trong các cột có Phí hàng tháng cao (trên 80\$). Ngược lại, ở các cột phí thấp (dưới 30\$), số lượng Non-Churn (màu xanh) chiếm ưu thế áp đảo.

Kết luận: Chi phí hàng tháng cao là một trong những yếu tố dự đoán mạnh mẽ nhất của việc Churn. Điều này thường liên quan đến khách hàng sử dụng gói Fiber Optic (dịch vụ có nguy cơ Churn cao nhất). Khách hàng trả nhiều tiền có kỳ vọng cao hơn về chất lượng dịch vụ và hỗ trợ.

Hành động: Cần tập trung các chương trình giảm giá, ưu đãi cá nhân hóa, hoặc nâng cấp dịch vụ miễn phí cho nhóm khách hàng chi tiêu cao để giảm gánh nặng chi phí/tăng giá trị nhận được.



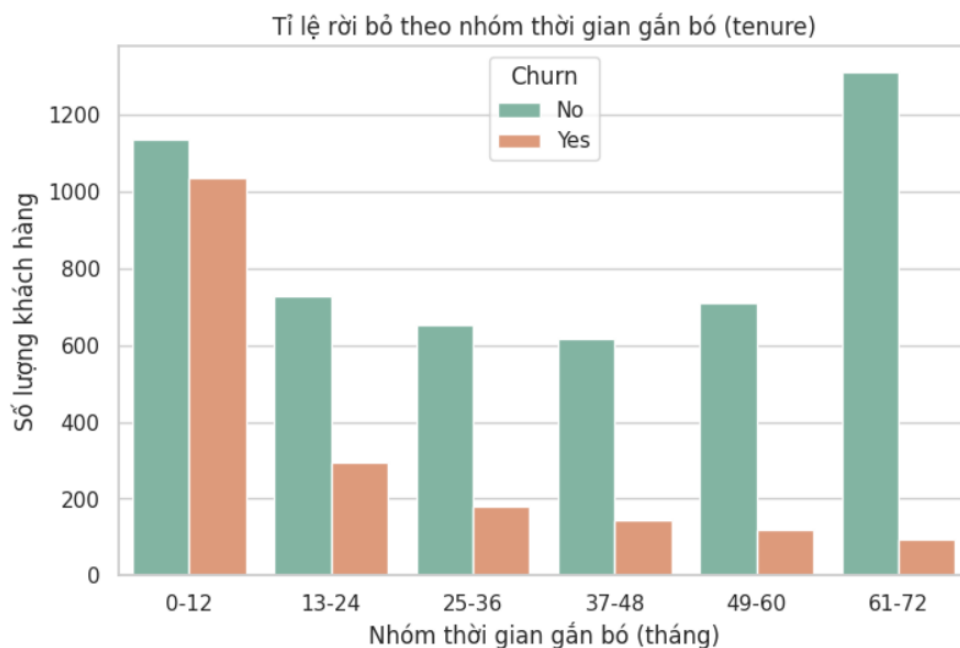
Biểu đồ **Phân phối TotalCharges theo Churn** cho thấy mối quan hệ giữa tổng chi tiêu tích lũy và hành vi rời bỏ:

- **Nhận xét chính:**

Số lượng Churn (màu cam) có mật độ cao nhất ở các cột có Tổng chi phí thấp (gần 0\$).

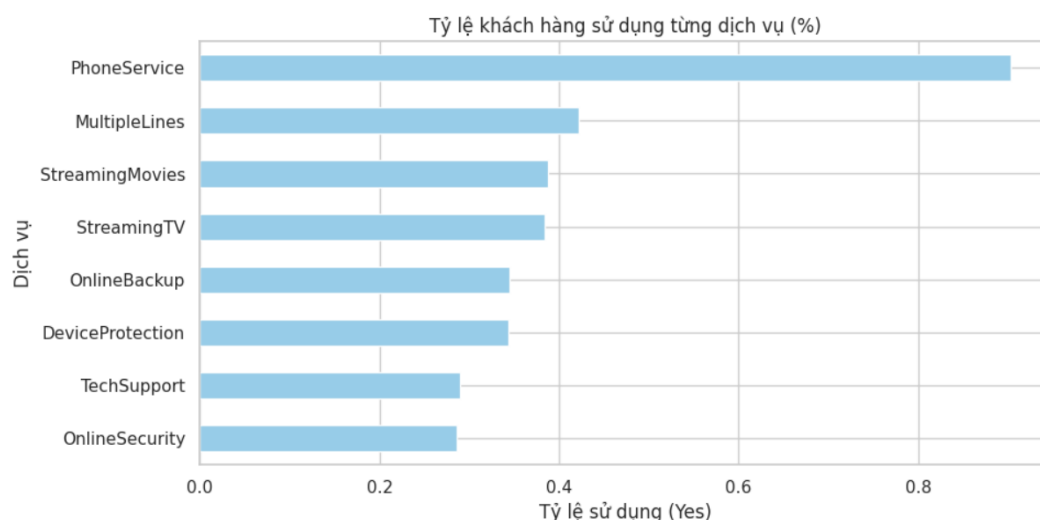
Khi Tổng chi phí tăng lên (đặc biệt trên 2000\$), số lượng Non-Churn (màu xanh) chiếm ưu thế gần như tuyệt đối.

- **Kết luận:** Tổng chi phí thấp là dấu hiệu của khách hàng mới (vì TotalCharges tương quan mạnh với Tenure), và họ là nhóm có nguy cơ rời bỏ cao nhất. Ngược lại, Tổng chi phí cao đồng nghĩa với khách hàng trung thành và ổn định.
- **Hành động:** Kết quả này củng cố các nhận định từ tenure: Chiến lược nên nhắm vào khách hàng mới với chi tiêu tích lũy thấp để chuyển họ thành khách hàng có cam kết dài hạn trước khi họ quyết định rời đi.



Biểu đồ cột này trực quan hóa rõ ràng mối quan hệ giữa thời gian khách hàng gắn bó và hành vi rời bỏ:

- **Nhận xét chính:** Nhóm khách hàng 0-12 tháng có số lượng Churn (màu cam) cao nhất tuyệt đối so với tất cả các nhóm tenure khác.
- **Tầm quan trọng:** Sự rủi ro giảm dần và ổn định khi khách hàng vượt qua mốc 24 tháng. Nhóm 61-72 tháng có tỷ lệ Churn cực kỳ thấp và là nhóm khách hàng trung thành nhất.
- **Kết luận:** Phân tích này là nền tảng để công ty phân bổ nguồn lực giữ chân (Retention) vào từng nhóm: Ưu tiên can thiệp khẩn cấp vào nhóm 0-12 tháng và 13-24 tháng.



Biểu đồ thanh ngang này phân tích mức độ sử dụng của các dịch vụ khác nhau:

- **Nhận xét chính:**

PhoneService là dịch vụ cơ bản nhất, có tỷ lệ sử dụng cao nhất (91%).

Hai dịch vụ **OnlineSecurity** và **TechSupport** có tỷ lệ sử dụng thấp nhất (28%-29%).

- **Mối liên hệ với Churn:** Mặc dù có tỷ lệ sử dụng thấp, nhưng theo phân tích tương quan trước đó, OnlineSecurity và TechSupport lại là các dịch vụ có tương quan âm mạnh với Churn. Điều này có nghĩa là, khách hàng nào sử dụng các dịch vụ này thì nguy cơ rời bỏ thấp hơn đáng kể.
- **Hành động:** Công ty nên coi OnlineSecurity và TechSupport là các "rào cản giữ chân" (stickiness factors) và tập trung khuyến khích khách hàng có nguy cơ cao (nhóm Month-to-month, Fiber optic) sử dụng chúng (ví dụ: tặng miễn phí 3 tháng) để tăng độ ràng buộc.

1.4.4 Xây dựng mô hình

Sau khi tiền xử lý dữ liệu, các mô hình học máy được áp dụng để dự đoán khả năng khách hàng rời bỏ (Churn), bao gồm:

Logistic Regression, K-Nearest Neighbors, và Random Forest.

Các mô hình này được triển khai thông qua Pipeline, giúp kết hợp các bước tiền xử lý và huấn luyện một cách tự động và thống nhất.

Sau khi huấn luyện, mô hình được đánh giá bằng các chỉ số Accuracy, Precision, Recall, F1-score và ROC-AUC để chọn ra mô hình có hiệu quả cao nhất.

```
# Tách dữ liệu Train/Test

from sklearn.model_selection import train_test_split

x = df_fe.drop(columns=['Churn'])
y = df_fe['Churn']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42, stratify=y
)

print("Train:", X_train.shape, "Test:", X_test.shape)
```

Train: (5282, 19) Test: (1761, 19)

```
# Pipeline xử lý dữ liệu (numeric + categorical)

from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

num_pipe = Pipeline([
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

# Với sklearn >= 1.2 dùng sparse_output=False
cat_pipe = Pipeline([
    ('imputer', SimpleImputer(strategy='constant', fill_value='missing')), # handle missing values
    ('onehot', OneHotEncoder(handle_unknown='ignore', sparse_output=False)) # one-hot encode
])

preprocessor = ColumnTransformer([
    ('num', num_pipe, numeric_features),
    ('cat', cat_pipe, categorical_features)
])
```

1.4.5 Đánh giá mô hình

Các mô hình được đánh giá dựa trên các chỉ số Accuracy, Precision, Recall, F1-Score và ROC-AUC nhằm phản ánh độ chính xác và khả năng phân loại của mô hình.

Kết quả thu được từ các mô hình Logistic Regression, K-Nearest Neighbors, và Random Forest. được so sánh để lựa chọn mô hình có hiệu suất dự đoán tốt nhất cho bài toán dự đoán Churn.

1.5. Ý nghĩa thực tiễn của đề tài

Đề tài có ý nghĩa thực tiễn sâu sắc trong việc giúp doanh nghiệp viễn thông dự đoán và ngăn chặn tình trạng khách hàng rời bỏ. Thông qua việc ứng dụng các mô hình học máy, doanh nghiệp có thể nhận diện sớm nhóm khách hàng có nguy cơ rời bỏ cao, từ đó đưa ra các chính sách giữ chân phù hợp, tiết kiệm chi phí marketing và nâng cao chất lượng dịch vụ.

CHƯƠNG 2: MỤC TIÊU BÀI TOÁN ĐẶT RA

2.1 Khái niệm về khai phá dữ liệu

Khai phá dữ liệu là quá trình phát hiện ra các mẫu, quy luật, mối quan hệ tiềm ẩn trong một tập dữ liệu lớn bằng các phương pháp thống kê, học máy và trí tuệ nhân tạo.

Mục tiêu chính là chuyển đổi dữ liệu thô thành tri thức hữu ích, hỗ trợ doanh nghiệp ra quyết định chính xác hơn.

2.2. Bài toán dự đoán khách hàng rời bỏ (Customer Churn Prediction)

Churn là hiện tượng khách hàng ngừng sử dụng sản phẩm hoặc dịch vụ của doanh nghiệp.

Bài toán dự đoán churn nhằm xác định khả năng một khách hàng sẽ rời bỏ dựa trên các đặc điểm trong dữ liệu (thời gian hợp đồng, loại dịch vụ, phương thức thanh toán, v.v.).

Việc dự đoán churn giúp doanh nghiệp:

- Nhận diện sớm khách hàng có nguy cơ rời bỏ.
- Đưa ra chính sách ưu đãi, chăm sóc phù hợp.
- Giảm chi phí marketing tìm khách hàng mới.

2.3. Tiền xử lý dữ liệu (Data Preprocessing)

Dữ liệu thực tế thường chứa giá trị rỗng, trùng lặp hoặc không đồng nhất, nên cần tiền xử lý để đảm bảo mô hình học máy hoạt động hiệu quả.

Các bước chính:

- Xử lý giá trị thiếu (Missing Values).
- Mã hóa dữ liệu phân loại (Categorical Encoding) bằng OneHotEncoder.
- Chuẩn hóa dữ liệu (Normalization) để các thuộc tính có cùng thang đo.
- Chia dữ liệu thành tập huấn luyện (train) và kiểm thử (test).

2.5. Đánh giá mô hình (Model Evaluation)

Các chỉ số được sử dụng để đánh giá độ chính xác của mô hình gồm:

- Accuracy: Tỷ lệ dự đoán đúng trên tổng số mẫu.
- Precision: Tỷ lệ dự đoán đúng trong các mẫu được dự đoán là rời bỏ.
- Recall: Tỷ lệ khách hàng rời bỏ thực tế được mô hình phát hiện đúng.
- F1-Score: Trung bình điều hòa giữa Precision và Recall.
- ROC-AUC: Khả năng phân biệt giữa hai lớp khách hàng.

CHƯƠNG 3: MÔ TẢ DỮ LIỆU VÀ CÁC BƯỚC TIỀN XỬ LÝ

3.1 Mô tả dữ liệu

Dữ liệu được sử dụng trong phân tích này là tập dữ liệu Telco Customer Churn, chứa thông tin về khách hàng của một công ty viễn thông, bao gồm các đặc điểm về dịch vụ, thanh toán, và trạng thái rời bỏ dịch vụ (Churn). Mục tiêu là dự đoán hành vi của khách hàng để xác định những người có khả năng rời bỏ dịch vụ trong tương lai.

Dữ liệu ban đầu có 7043 dòng và 21 cột. Ý nghĩa của từng cột đã được mô tả chi tiết ở phần trước.

3.2 Các bước tiền xử lý dữ liệu

Các bước tiền xử lý đã thực hiện bao gồm:

1. Kiểm tra và xử lý giá trị khuyết (NaN): Dữ liệu ban đầu được kiểm tra và cột TotalCharges có một số giá trị rỗng. Những dòng này đã được loại bỏ.
2. Xử lý cột TotalCharges: Cột này ban đầu có kiểu dữ liệu object và đã được chuyển đổi sang kiểu số (numeric). Các giá trị không hợp lệ trong quá trình chuyển đổi được xử lý bằng cách thay thế bằng NaN và sau đó loại bỏ các dòng chứa NaN.
3. Chuẩn hóa khoảng trống: Các giá trị dạng chuỗi (object) trong dữ liệu được loại bỏ khoảng trống ở đầu và cuối để đảm bảo tính nhất quán.
4. Chuyển đổi biến mục tiêu Churn: Cột mục tiêu Churn (Yes/No) được chuyển đổi thành dạng số (1/0) để phù hợp với các mô hình học máy.
5. Chuẩn hóa giá trị đặc biệt: Các giá trị như "No internet service" và "No phone service" trong các cột dịch vụ đã được chuẩn hóa thành "No" để đơn giản hóa và tăng tính đồng nhất cho các cột phân loại.

Sau các bước tiền xử lý, tập dữ liệu còn lại 7032 dòng và 20 cột, sẵn sàng cho các phân tích tiếp theo và xây dựng mô hình.

CHƯƠNG 4: PHƯƠNG PHÁP KHAI PHÁ DỮ LIỆU / MÔ HÌNH ML

4.1 Tiền xử lý dữ liệu:

Làm sạch dữ liệu:

- Xóa cột customerID.
- Chuyển đổi cột TotalCharges sang dạng số và xử lý các giá trị rỗng bằng cách xóa các dòng tương ứng.
- Chuẩn hóa khoảng trắng cho các cột kiểu object.
- Chuyển đổi biến mục tiêu Churn từ 'Yes'/'No' sang 1/0.
- Chuẩn hóa các giá trị đặc biệt ("No internet service", "No phone service") trong các cột dịch vụ về "No".

Kiểm tra dữ liệu:

- Kiểm tra các giá trị thiếu (isna().sum()).
- Kiểm tra giá trị ngoại lệ bằng biểu đồ boxplot cho các cột numeric (tenure, MonthlyCharges, TotalCharges).

4.2 Phân tích khám phá dữ liệu (EDA):

- Trực quan hóa phân bố của biến mục tiêu Churn.
- Phân tích tỷ lệ Churn theo giới tính.
- Khám phá mối quan hệ giữa Contract, InternetService, tenure, MonthlyCharges, TotalCharges và Churn thông qua các biểu đồ countplot và histplot.
- Phân tích tỷ lệ Churn theo các nhóm tenure.
- Trực quan hóa ma trận tương quan giữa các biến numeric.
- Xác định tỷ lệ sử dụng của các dịch vụ khác nhau và chỉ ra top 3 dịch vụ được sử dụng nhiều nhất và ít nhất.

- Xem xét các thống kê cơ bản về tenure, MonthlyCharges, TotalCharges.

4.3 Xây dựng và đánh giá mô hình ML:

Chuẩn bị dữ liệu cho mô hình:

- Tách các đặc trưng (X) và biến mục tiêu (y).
- Thực hiện One-hot encoding cho các biến phân loại.
- Chia dữ liệu thành tập huấn luyện và tập kiểm tra.

Xử lý dữ liệu mất cân bằng:

Sử dụng NearMiss (undersampling) và SMOTE (oversampling) trên tập huấn luyện để cân bằng lại tỷ lệ các lớp.

Mô hình đã sử dụng:

- Logistic Regression: Huấn luyện trên dữ liệu gốc, dữ liệu sau NearMiss, và dữ liệu sau SMOTE.
- XGBoost: Huấn luyện trên dữ liệu gốc, dữ liệu sau NearMiss, và dữ liệu sau SMOTE.

Tuning Hyperparameters:

Sử dụng GridSearchCV với 5-fold cross-validation để tìm ra bộ tham số tốt nhất cho Logistic Regression và XGBoost dựa trên dữ liệu sau SMOTE.

Điều chỉnh Threshold:

Tìm threshold tối ưu cho mô hình Logistic Regression và XGBoost dựa trên F1-score trên tập kiểm tra để cải thiện hiệu suất dự đoán lớp thiểu số (Churn=1).

Đánh giá mô hình:

- Sử dụng classification_report để đánh giá các mô hình (precision, recall, f1-score, support, accuracy, macro avg, weighted avg) trên tập kiểm tra.

- Sử dụng cross-validation với 5 folds và scoring là f1 để đánh giá độ ổn định của mô hình Logistic Regression và XGBoost trên toàn bộ dữ liệu.

4.4 Phân cụm khách hàng (Clustering):

Sử dụng các đặc trưng numeric (tenure, MonthlyCharges, TotalCharges, SeniorCitizen) để phân cụm.

Sử dụng phương pháp Elbow Method và Silhouette Score để xác định số cụm tối ưu (k=3 được chọn).

Thực hiện K-Means clustering với k=3.

Phân tích profile trung bình của từng cụm dựa trên các đặc trưng được sử dụng và tỷ lệ churn trong mỗi cụm.

Gán nhãn cho các cụm dựa trên phân tích profile (Loyal, At-Risk, Churners).

Trực quan hóa các cụm bằng biểu đồ scatterplot sau khi giảm chiều dữ liệu bằng PCA.

Đánh giá chất lượng phân cụm bằng Silhouette Score.

Trực quan hóa phân bố churn thực tế trong từng cụm.

4.5 Áp dụng mô hình tốt nhất:

Lưu mô hình tốt nhất (Logistic Regression sau tuning threshold) bằng joblib.

Xây dựng một ứng dụng web đơn giản bằng Streamlit để dự đoán churn dựa trên đầu vào của người dùng.

Sử dụng ngrok để tạo public URL cho ứng dụng Streamlit.

CHƯƠNG 5: KẾT QUẢ VÀ ĐÁNH GIÁ MÔ HÌNH

5.1 Đánh giá loại vấn đề

Vấn đề cần giải quyết trong tập dữ liệu “Telco Customer Churn” là loại supervised learning (học có giám sát) - thuộc dạng classification (phân loại). Mục tiêu là dự đoán hành vi rời bỏ (churn) để giữ chân khách hàng — phân tích dữ liệu khách hàng và xây dựng chương trình giữ chân tập trung.

So sánh điểm mạnh của thuật toán với vấn đề:

Mô hình	Điểm mạnh	Phù hợp với
Logistic Regression	<p>Dễ triển khai, huấn luyện nhanh, cho biết biến ảnh hưởng mạnh/yếu</p> <p>Cho xác suất rõ ràng, ít bị overfitting nếu regularization</p>	Tập dữ liệu vừa, có cả biến số và biến phân loại, mối quan hệ gần tuyến tính
XGBoost	<p>Mô hình mạnh mẽ, hiệu quả cao trên tập dữ liệu lớn; xử lý tốt dữ liệu nhiều chiều, cả dạng số lẫn phân loại</p> <p>Có khả năng khắc phục phần nào mất cân bằng lớp; ít bị overfitting; đạt độ chính xác cao, linh hoạt, dễ mở rộng và hỗ trợ xử lý song song.</p>	Tập dữ liệu lớn, nhiều đặc trưng, có quan hệ phi tuyến.

5.2 Huấn luyện mô hình

5.2.1 Mô hình Logistic Regression

Do biến mục tiêu Churn bị mất cân bằng, nên trước tiên ta sẽ huấn luyện mô hình trên tập dữ liệu gốc để quan sát hiệu suất ban đầu. Sau đó, ta sẽ áp dụng các kỹ

thuật cân bằng dữ liệu như undersampling và oversampling, đồng thời tiến hành tuning tham số, điều chỉnh ngưỡng dự đoán (threshold) và cross-validation để tối ưu kết quả.

Mô hình đầu tiên được sử dụng là Logistic Regression – một mô hình tuyến tính. Cơ chế hoạt động của nó là tìm ra một siêu phẳng (đường phân cách) nhằm chia các điểm dữ liệu thành hai phía tương ứng với hai lớp.

Trước khi huấn luyện, ta cần mã hóa các biến dạng object bằng phương pháp One-Hot Encoding, vì nhiều mô hình (bao gồm Logistic Regression) không thể xử lý trực tiếp dữ liệu dạng ký tự, mà chỉ làm việc với biến số học (numeric features).

```
# Tạo ra onehot
dataset_dummy = pd.get_dummies(df_processed, drop_first=True)
display(dataset_dummy)
```

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges	Churn	gender_Male	Partner_Yes	Dependents_Yes	PhoneService_Yes	MultipleLines_Yes	...	DeviceProtection_Yes	TechSupport_
0	0	1	29.85	29.85	0	False	True	False	False	False	...	False	Fi
1	0	34	56.95	1889.50	0	True	False	False	True	False	...	True	Fi
2	0	2	53.85	108.15	1	True	False	False	True	True	...	False	Fi
3	0	45	42.30	1840.75	0	True	False	False	False	False	...	True	1
4	0	2	70.70	151.65	1	False	False	False	True	False	...	False	Fi
...

Bước tiếp theo, ta chuẩn hóa (scale) các biến dạng số (numeric features) về khoảng [0, 1]. Việc này giúp giảm chênh lệch về độ lớn giữa các đặc trưng, giúp mô hình học ổn định và nhanh hội tụ hơn.

```
Numeric features: ['SeniorCitizen', 'tenure', 'MonthlyCharges', 'TotalCharges']
Categorical features: []
```

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges
0	-0.440327	-1.280248	-1.161694	-0.994194
1	-0.440327	0.064303	-0.260878	-0.173740
2	-0.440327	-1.239504	-0.363923	-0.959649
3	-0.440327	0.512486	-0.747850	-0.195248
4	-0.440327	-1.239504	0.196178	-0.940457

Sau đó, ta chia tập dữ liệu đã được xử lý thành hai phần: tập huấn luyện (train) và tập kiểm tra (test).

Biến X sẽ chứa tất cả các đặc trưng đầu vào, không bao gồm biến mục tiêu Churn.

Biến y sẽ chỉ chứa cột mục tiêu Churn, dùng để đánh giá hiệu suất của mô hình trong quá trình huấn luyện và kiểm thử.

```
❏ X = dataset_dummy.drop(columns=['Churn'])  
   y = dataset_dummy['Churn']  
  
   X_train, X_test, y_train, y_test = train_test_split(  
       X, y, test_size=0.2, random_state=42, stratify=y  
   )  
  
   print("Train:", X_train.shape, "Test:", X_test.shape)  
  
   # Đánh giá hiệu suất  
   def print_scores(y_true, y_pred):  
       print(classification_report(y_true, y_pred))
```

```
🔗 Train: (5625, 23) Test: (1407, 23)
```

Bước tiếp theo, ta sẽ huấn luyện mô hình Logistic Regression.

Trước tiên, cần import thư viện LogisticRegression từ sklearn.linear_model. Sau đó, khởi tạo và huấn luyện mô hình trên tập dữ liệu huấn luyện gồm X_train và y_train.

Khi mô hình đã được huấn luyện xong, ta đánh giá hiệu suất bằng cách dự đoán trên tập dữ liệu kiểm tra (X_test) và so sánh kết quả dự đoán với giá trị thực tế (y_test) để xem mức độ chính xác của mô hình.

5.2.1.1 Đánh giá mô hình Logistic Regression

Mô hình Logistic Regression:

Logistic Regression ROC AUC: 0.8461				
	precision	recall	f1-score	support
0	0.9092	0.7349	0.8128	1294
1	0.5203	0.7966	0.6294	467
accuracy			0.7513	1761
macro avg	0.7147	0.7658	0.7211	1761
weighted avg	0.8060	0.7513	0.7642	1761

ROC AUC = 0.8461 → Mô hình có khả năng phân biệt giữa khách hàng rời và không rời ở mức tốt, khá đáng tin cậy.

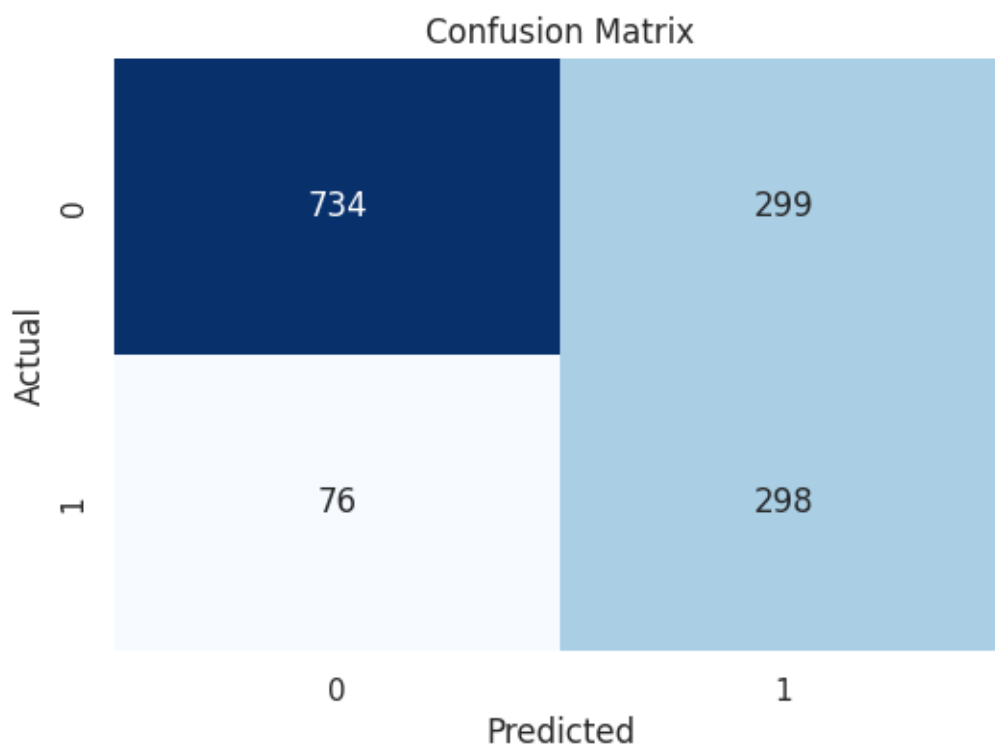
Precision (lớp 1 – khách hàng rời): 0.52 → Trong các khách hàng được dự đoán rời, chỉ khoảng 52% là đúng thực sự, nghĩa là mô hình còn dự đoán nhầm một số khách hàng không rời.

Recall (lớp 1): 0.80 → Mô hình bắt được 80% khách hàng thực sự rời, tức khả năng phát hiện khách hàng có nguy cơ cao rất tốt — điều này quan trọng trong bài toán giữ chân.

Accuracy: 75% → Mức chính xác tổng thể khá ổn, nhưng vì dữ liệu có mất cân bằng nên không phải chỉ số quan trọng nhất.

Đánh giá tổng thể:

Mô hình Logistic Regression hoạt động tốt, đặc biệt trong việc phát hiện khách hàng sắp rời (recall cao), tuy nhiên cần cải thiện precision để giảm số khách hàng bị dự đoán nhầm là rời đi.



5.2.2 Mô hình XGBoost

Với mô hình XGBoost, quy trình ban đầu tương tự như Logistic Regression. Trước hết, ta thực hiện One-Hot Encoding trên các biến phân loại trong tập dữ liệu gốc, sau đó chia dữ liệu thành hai phần: tập huấn luyện (train) và tập kiểm tra (test).

Khác với Logistic Regression, XGBoost không yêu cầu chuẩn hóa (scaling) dữ liệu, vì mô hình có khả năng tự điều chỉnh và tối ưu hóa các đặc trưng trong quá trình huấn luyện. Do đó, ta có thể bỏ qua bước scaler và chuyển sang khởi tạo mô hình.

Tiếp theo, ta tạo một mô hình phân loại XGBoost bằng lớp `XGBClassifier` với các tham số cơ bản:

- **random_state=42**: đảm bảo tính tái lập kết quả (reproducibility).
- **n_estimators=200**: xác định số lượng cây (trees) trong mô hình là 200.

Sau khi khởi tạo, ta huấn luyện mô hình trên tập train và thực hiện dự đoán trên tập test để đánh giá hiệu quả phân loại của XGBoost.

5.2.2.1 Đánh giá mô hình

	precision	recall	f1-score	support
0	0.83	0.87	0.85	1033
1	0.58	0.51	0.54	374
accuracy			0.77	1407
macro avg	0.71	0.69	0.69	1407
weighted avg	0.76	0.77	0.77	1407

Mô hình XGBoost đạt độ chính xác tổng thể (accuracy) là 0.77, tức dự đoán đúng khoảng 77% dữ liệu kiểm thử.

Lớp 0 (khách hàng không rời bỏ) có precision = 0.83, recall = 0.87, F1-score = 0.85, cho thấy mô hình nhận diện tốt nhóm này.

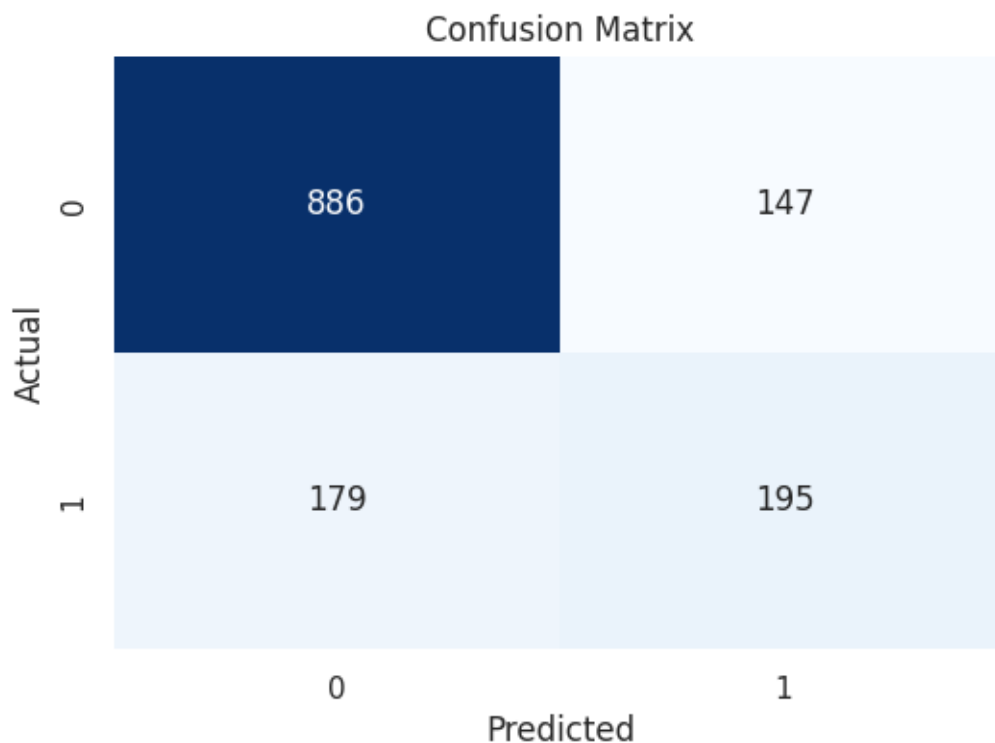
Lớp 1 (khách hàng rời bỏ) có precision = 0.58, recall = 0.51, F1-score = 0.54, phản ánh khả năng dự đoán nhóm rời bỏ còn hạn chế.

Mô hình có xu hướng thiên về lớp 0, tức dễ dự đoán khách hàng ở lại hơn là khách hàng rời bỏ.

Macro average F1-score = 0.69 cho thấy độ cân bằng giữa hai lớp chưa cao.

Weighted average F1-score = 0.77 cao hơn do lớp 0 chiếm đa số và được mô hình dự đoán tốt.

Nhìn chung, mô hình có hiệu quả tương đối tốt nhưng khả năng nhận diện khách hàng có nguy cơ rời bỏ vẫn còn thấp.



5.3 Cân bằng dữ liệu

5.3.1 Cân bằng dữ liệu với phương pháp Nearest

Tiếp theo, ta tiến hành cân bằng dữ liệu bằng phương pháp Undersampling, tức là giảm số lượng mẫu thuộc lớp chiếm ưu thế để cân đối với lớp thiểu số. Phương pháp này được thực hiện thông qua thư viện NearestNeighbors, giúp lựa chọn các mẫu đại diện gần nhất để giữ lại.

Trước khi thực hiện, nếu dữ liệu chưa được One-Hot Encoding và chưa được tách thành hai tập train – test, ta cần thực hiện mã hóa One-Hot trước để đảm bảo mô hình có thể xử lý đúng dạng dữ liệu.

```
# --- 1. One-hot encoding ---
dataset_dummy = pd.get_dummies(df_processed, drop_first=True)

# --- 2. Tách feature và target ---
X = dataset_dummy.drop(["Churn"], axis=1)
y = dataset_dummy["Churn"]

# --- 3. Chia dữ liệu train/test ---
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

Bước tiếp theo là cân bằng dữ liệu trên tập huấn luyện (train set).

Điều này nghĩa là ta chỉ điều chỉnh tỷ lệ mẫu trong tập train, sao cho số lượng của hai lớp trở nên cân bằng hơn. Không nên cân bằng toàn bộ dữ liệu trước khi tách train – test, vì điều đó có thể gây ra rò rỉ dữ liệu (data leakage) — mô hình sẽ “vô tình học” được đặc trưng của toàn bộ tập dữ liệu, làm kết quả đánh giá mất tính khách quan.

```
print("Before resampling:")
print("Train:", X_train.shape, "Test:", X_test.shape)
print("Tỷ lệ trước khi cân bằng:", y_train.value_counts(normalize=True))

# --- 4. Cân bằng dữ liệu (undersampling) chỉ trên tập train ---
nm = NearMiss()
X_train_res, y_train_res = nm.fit_resample(X_train, y_train)

print("\nAfter resampling (NearMiss):")
print("Train_res:", X_train_res.shape)
print("Tỷ lệ sau khi cân bằng:", y_train_res.value_counts(normalize=True))
```

Sau khi cân bằng xong, ta chuẩn hóa (scale) các biến dạng số (numeric features) về khoảng [0, 1] để đảm bảo mô hình học ổn định và nhanh hơn.

```
# --- 6. Chuẩn hóa dữ liệu numeric ---
scaler = StandardScaler()
scaler.fit(X_train_res[numeric_features])

X_train_res[numeric_features] = scaler.transform(X_train_res[numeric_features])
X_test[numeric_features] = scaler.transform(X_test[numeric_features])
```


Sau khi xong các bước trên thì tiến hành huấn luyện mô hình ta sẽ huấn luyện với mô hình Logictics Regression

```
# --- 7. Huấn luyện mô hình Logistic Regression ---
model = LogisticRegression(max_iter=1000)
model.fit(X_train_res, y_train_res)

# --- 8. Dự đoán và đánh giá ---
y_pred = model.predict(X_test)

print("\n--- Kết quả đánh giá ---")
print(classification_report(y_test, y_pred))
```

5.3.1.1 Đánh giá mô hình Logistic Regression

Cho ra kết quả đánh giá như sau:

```
Before resampling:
Train: (5625, 23) Test: (1407, 23)
Tỷ lệ trước khi cân bằng: Churn
0    0.734222
1    0.265778
Name: proportion, dtype: float64

After resampling (NearMiss):
Train_res: (2990, 23)
Tỷ lệ sau khi cân bằng: Churn
0    0.5
1    0.5
Name: proportion, dtype: float64

Numeric features: ['SeniorCitizen', 'tenure', 'MonthlyCharges', 'TotalCharges']

--- Kết quả đánh giá ---
              precision    recall  f1-score   support

0               0.87         0.72         0.79         1033
1               0.48         0.72         0.57          374

 accuracy               0.72         1407
 macro avg              0.68         0.72         0.68         1407
 weighted avg           0.77         0.72         0.73         1407
```

Mô hình Logistic Regression đạt độ chính xác (accuracy) là 0.72, tức là dự đoán đúng khoảng 72% tổng số mẫu kiểm thử.

Lớp 0 (khách hàng không rời bỏ) có precision = 0.87, recall = 0.72, F1-score = 0.79, thể hiện khả năng nhận diện khá tốt nhóm khách hàng ở lại.

Lớp 1 (khách hàng rời bỏ) có precision = 0.48, recall = 0.72, F1-score = 0.57, cho thấy mô hình đã cải thiện đáng kể khả năng phát hiện khách hàng rời bỏ so với trước, nhờ kỹ thuật downsampling.

Recall của lớp 1 đạt 0.72, tức mô hình phát hiện được phần lớn các trường hợp rời bỏ, điều này rất quan trọng trong bài toán churn.

Tuy nhiên, precision của lớp 1 còn thấp (0.48), nghĩa là vẫn có nhiều dự đoán sai (dự đoán rời bỏ nhưng thực tế không rời).

Macro average F1 = 0.68 và weighted average F1 = 0.73 phản ánh mô hình cân bằng hơn giữa hai lớp sau khi xử lý dữ liệu.

Mô hình có xu hướng cải thiện khả năng nhận diện khách hàng rời bỏ (class 1) nhưng đánh đổi bằng việc giảm độ chính xác ở lớp 0.

Nhìn chung, kỹ thuật downsampling bằng NearMiss đã giúp mô hình bớt thiên lệch và nâng cao recall cho lớp rời bỏ, phù hợp khi mục tiêu là phát hiện khách hàng có nguy cơ rời bỏ.

➔ Nhận xét: cho thấy rằng sau khi sample dữ liệu thì khả năng nhận diện lớp 1 tức lớp rời đi kém hơn so với ban đầu

5.3.1.2 Đánh giá mô hình XGBoost

Với XGBoost cũng làm tương tự cho ra kết quả như sau:

```

Before resampling:
Train: (5625, 23) Test: (1407, 23)
Tỷ lệ trước khi cân bằng: Churn
0    0.734222
1    0.265778
Name: proportion, dtype: float64

After resampling (NearMiss):
Train_res: (2990, 23)
Tỷ lệ sau khi cân bằng: Churn
0    0.5
1    0.5
Name: proportion, dtype: float64

Numeric features: ['SeniorCitizen', 'tenure', 'MonthlyCharges', 'TotalCharges']

```

	precision	recall	f1-score	support
0	0.83	0.84	0.84	1033
1	0.55	0.53	0.54	374
accuracy			0.76	1407
macro avg	0.69	0.69	0.69	1407
weighted avg	0.76	0.76	0.76	1407

Mô hình XGBoost đạt độ chính xác (accuracy) là 0.76, tức là dự đoán đúng khoảng 76% tổng số mẫu kiểm thử.

Lớp 0 (khách hàng không rời bỏ) có precision = 0.83, recall = 0.84, F1-score = 0.84, cho thấy mô hình nhận diện rất tốt nhóm khách hàng ở lại.

Lớp 1 (khách hàng rời bỏ) có precision = 0.55, recall = 0.53, F1-score = 0.54, thể hiện mô hình có khả năng phát hiện khách hàng rời bỏ ở mức trung bình.

So với mô hình Logistic Regression dùng NearMiss, precision và recall của lớp 1 thấp hơn một chút, nhưng độ chính xác tổng thể cao hơn.

Macro average F1 = 0.69 cho thấy sự cân bằng giữa hai lớp vẫn còn hạn chế, nhưng đã có cải thiện nhẹ so với khi chưa xử lý dữ liệu.

Weighted average F1 = 0.76 phản ánh hiệu suất tổng thể khá ổn, chủ yếu nhờ mô hình dự đoán tốt lớp chiếm đa số (class 0).

Mô hình vẫn có xu hướng thiên về lớp 0, nghĩa là vẫn dễ nhận diện khách hàng không rời bỏ hơn.

Nhìn chung, XGBoost sau khi resample bằng NearMiss giúp cải thiện độ cân bằng giữa hai lớp nhưng chưa đủ mạnh để tăng recall cho nhóm rời bỏ, do đó

có thể xem xét tinh chỉnh tham số hoặc thử kỹ thuật resampling khác (SMOTE, ADASYN) để tăng khả năng phát hiện churn.

5.3.2 Kỹ thuật Oversampling với SMOTE

Kỹ thuật SMOTE (Synthetic Minority Over-sampling Technique) là một phương pháp tăng mẫu cho lớp thiểu số bằng cách tạo ra các điểm dữ liệu mới (synthetic samples) dựa trên nội suy giữa các mẫu hiện có. Nói một cách đơn giản, SMOTE giúp tăng số lượng mẫu của lớp thiểu số sao cho cân bằng với lớp chiếm ưu thế.

Quy trình thực hiện Oversampling bằng SMOTE tương tự như phương pháp Undersampling, gồm các bước:

1. Chuẩn bị dữ liệu.
2. Thực hiện One-Hot Encoding cho các biến phân loại (nếu có).
3. Chia dữ liệu thành hai phần: train và test.
4. Chuẩn hóa (scale) các biến dạng số về khoảng $[0, 1]$.
5. Áp dụng SMOTE để tăng mẫu của lớp thiểu số trong tập huấn luyện.

6. Huấn luyện mô hình (ví dụ: Logistic Regression) trên tập train đã được cân bằng.

Sau khi hoàn tất huấn luyện, ta đánh giá hiệu suất của mô hình trên tập kiểm tra (test set) để so sánh kết quả với các phương pháp cân bằng khác.

5.3.2.1 Đánh giá mô hình Logistic Regression

```
Tập train/test ban đầu:
Train: (5625, 30) Test: (1407, 30)
Sau SMOTE: (8260, 30) (8260,)
Numeric features: ['SeniorCitizen', 'tenure', 'MonthlyCharges', 'TotalCharges']
```

	precision	recall	f1-score	support
0	0.86	0.80	0.83	1033
1	0.54	0.65	0.59	374
accuracy			0.76	1407
macro avg	0.70	0.72	0.71	1407
weighted avg	0.78	0.76	0.77	1407

Mô hình Logistic Regression đạt độ chính xác (accuracy) là 0.76, tức dự đoán đúng khoảng 76% tổng số mẫu kiểm thử.

Lớp 0 (khách hàng không rời bỏ) có precision = 0.86, recall = 0.80, F1-score = 0.83, cho thấy mô hình vẫn nhận diện tốt nhóm khách hàng ở lại.

Lớp 1 (khách hàng rời bỏ) có precision = 0.54, recall = 0.65, F1-score = 0.59, thể hiện hiệu suất phát hiện khách hàng rời bỏ đã được cải thiện rõ rệt so với các mô hình trước.

Recall của lớp 1 đạt 0.65, tức mô hình đã phát hiện được khoảng 65% các trường hợp rời bỏ – đây là mức khá tốt cho bài toán churn.

So với mô hình downsample bằng NearMiss, SMOTE giúp tăng recall và F1-score của lớp 1, cho thấy hiệu quả trong việc khắc phục mất cân bằng dữ liệu.

Macro average F1 = 0.71 và weighted average F1 = 0.77, phản ánh sự cân bằng giữa hai lớp tốt hơn và tổng thể mô hình hoạt động ổn định.

Mô hình có sự cân bằng tốt hơn giữa hai lớp, không còn thiên lệch quá nhiều về phía khách hàng không rời bỏ.

Nhìn chung, Logistic Regression sau khi áp dụng SMOTE đã cải thiện rõ rệt khả năng phát hiện khách hàng rời bỏ, giúp mô hình phù hợp hơn cho mục tiêu cảnh báo churn sớm.

5.3.2.1 Đánh giá mô hình XG Boost

XG Boost

Tập train/test ban đầu:					
Train: (5625, 30) Test: (1407, 30)					
Sau SMOTE: (8260, 30) (8260,)					
Numeric features: ['SeniorCitizen', 'tenure', 'MonthlyCharges', 'TotalCharges']					
	precision	recall	f1-score	support	
0	0.83	0.86	0.84	1033	
1	0.57	0.52	0.54	374	
accuracy			0.77	1407	
macro avg	0.70	0.69	0.69	1407	
weighted avg	0.76	0.77	0.76	1407	

Mô hình XGBoost đạt độ chính xác (accuracy) là 0.77, tức dự đoán đúng khoảng 77% tổng số mẫu kiểm thử.

Lớp 0 (khách hàng không rời bỏ) có precision = 0.83, recall = 0.86, F1-score = 0.84, cho thấy mô hình nhận diện rất tốt nhóm khách hàng ở lại.

Lớp 1 (khách hàng rời bỏ) có precision = 0.57, recall = 0.52, F1-score = 0.54, thể hiện khả năng dự đoán khách hàng rời bỏ ở mức trung bình.

Recall của lớp 1 đạt 0.52, thấp hơn mong đợi — cho thấy mô hình vẫn bỏ sót một phần đáng kể các khách hàng rời bỏ dù đã áp dụng SMOTE.

So với XGBoost chưa oversample, chỉ số F1 của lớp 1 gần như không thay đổi đáng kể, cho thấy SMOTE không cải thiện nhiều cho mô hình XGBoost.

Macro average $F1 = 0.69$ và weighted average $F1 = 0.76$, phản ánh hiệu suất tổng thể ổn định nhưng vẫn có sự chênh lệch giữa hai lớp.

Mô hình vẫn thiên lệch về lớp 0, có xu hướng dự đoán khách hàng sẽ ở lại hơn là rời bỏ.

Nhìn chung, XGBoost sau SMOTE duy trì được độ chính xác cao, nhưng chưa cải thiện đáng kể khả năng phát hiện khách hàng rời bỏ, gợi ý rằng cần tinh chỉnh tham số (hyperparameter tuning) hoặc thử các kỹ thuật cân bằng khác như SMOTEENN hoặc ADASYN để tăng recall cho lớp 1.

5.4 Tuning các mô hình

Đầu tiên thực hiện cân bằng dữ liệu bằng smote (oversample)

Sau đó áp dụng grid search cv để tìm tham số tối ưu cho từng mô hình

Đầu tiên:

```
models = {  
    "Logistic Regression": LogisticRegression(random_state=42),  
    "XGBoost": XGBClassifier(random_state=42)  
}
```

Tạo một **dictionary** chứa 2 mô hình cần huấn luyện.

random_state=42: đảm bảo kết quả có thể tái lập được (cố định random seed).

LogisticRegression: mô hình tuyến tính, thích hợp cho bài toán nhị phân.

XGBClassifier: mô hình boosting mạnh mẽ, hoạt động tốt với dữ liệu phi tuyến.

Tập hợp các siêu tham số (hyperparameters)

```
hyperparameters = {
```

```
"Logistic Regression": {  
    "penalty": ["l1", "l2", "elasticnet", None],  
    "C": [0.01, 0.1, 1, 10],  
    "solver": ["liblinear", "saga", "lbfgs"]  
},  
"XGBoost": {  
    "learning_rate": [0.01, 0.1, 0.2],  
    "max_depth": [3, 5, 7]  
}  
}
```

Giải thích siêu tham số:

→ **Logistic Regression**

- **penalty**: loại regularization (chuẩn bị chống overfitting)
 - **l1**: Lasso, loại bỏ bớt biến không quan trọng.
 - **l2**: Ridge, giảm hệ số nhưng không triệt tiêu.
 - **elasticnet**: kết hợp cả L1 và L2.
 - **None**: không regularization.
- **C**: nghịch đảo của mức độ regularization.
 - Giá trị nhỏ → regularization mạnh hơn (đơn giản mô hình).
- **solver**: thuật toán tối ưu dùng để huấn luyện.
 - **liblinear** (tốt cho dữ liệu nhỏ),

- saga (hỗ trợ elasticnet),
- lbfgs (cho l2 hoặc không regularization).

→ XGBoost

- learning_rate: tốc độ học, giá trị nhỏ giúp mô hình học chậm nhưng ổn định.
- max_depth: độ sâu tối đa của mỗi cây (càng sâu → càng phức tạp, dễ overfit).

Vòng lặp GridSearchCV

```
for model_name, model in models.items():
    print(f"Tuning {model_name}.....")
    grid = GridSearchCV(estimator=model, param_grid=hyperparameters[model_name], cv=5, scoring = "accuracy")
    grid.fit(X_train_smote, y_train_smote)
    print(f"Best parameters for {model_name} : {grid.best_params_}")
    print(f"Best Accuracy for {model_name} : {grid.best_score_: .2f}\n")
```

for model_name, model in models.items()

→ duyệt qua từng mô hình trong dictionary.

GridSearchCV(...)

- estimator: mô hình cần tuning.
- param_grid: tập hợp các bộ tham số cần thử.
- cv=5: chia dữ liệu thành 5 phần để cross-validation.
- scoring="accuracy": tiêu chí đánh giá là độ chính xác.

grid.fit(X_train_smote, y_train_smote)

→ huấn luyện mô hình với dữ liệu đã oversample bằng SMOTE.

grid.best_params_

→ in ra tổ hợp tham số tốt nhất tìm được.

grid.best_score_

→ in ra độ chính xác cao nhất tương ứng với bộ tham số tốt nhất.

5.4.1 Kết quả thu được cho 2 mô hình

Tuning Logistic Regression.....

Best parameters for Logistic Regression : {'C': 1, 'penalty': 'l1', 'solver': 'liblinear'}
Best Accuracy for Logistic Regression : 0.83

Tuning XGBoost.....

Best parameters for XGBoost : {'learning_rate': 0.2, 'max_depth': 7}
Best Accuracy for XGBoost : 0.84

Tiến hành huấn luyện trên các siêu tham số được tối ưu ở trên

```
from sklearn.linear_model import LogisticRegression

# Khởi tạo mô hình với tham số tốt nhất
log_model = LogisticRegression(C=1, penalty='l1', solver='liblinear')

# Huấn luyện mô hình
log_model.fit(X_train, y_train)

# Dự đoán
y_pred_log = log_model.predict(X_test)

print(f"Classification Report : \n", classification_report(y_test, y_pred_log))
```

```
Classification Report :
              precision    recall  f1-score   support

     0       0.85         0.89         0.87         1033
     1       0.65         0.57         0.61          374

 accuracy          0.80
 macro avg         0.75
 weighted avg      0.80
```

Nhận xét:

Mô hình Logistic Regression sau khi tối ưu siêu tham số đạt độ chính xác (accuracy) là 0.80, tức là dự đoán đúng khoảng 80% tổng số mẫu kiểm thử.

Lớp 0 (khách hàng không rời bỏ) có precision = 0.85, recall = 0.89, F1-score = 0.87, cho thấy mô hình nhận diện rất tốt nhóm khách hàng ở lại.

Lớp 1 (khách hàng rời bỏ) có precision = 0.65, recall = 0.57, F1-score = 0.61, thể hiện khả năng dự đoán nhóm rời bỏ được cải thiện rõ rệt so với các mô hình Logistic trước đó.

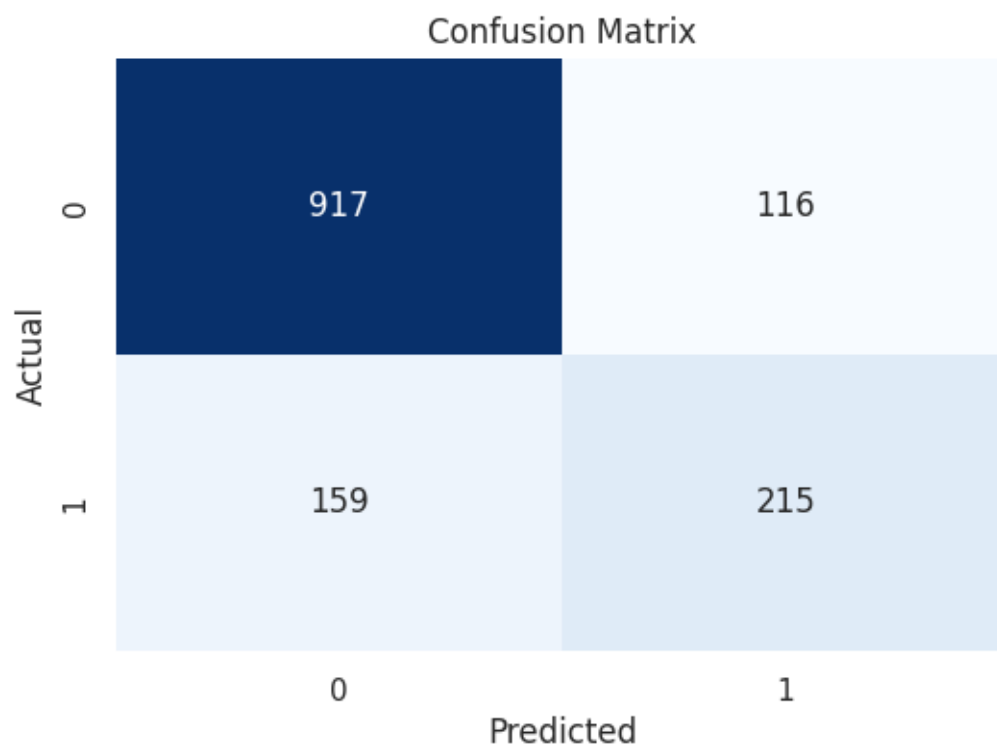
Recall của lớp 1 đạt 0.57, nghĩa là mô hình phát hiện được hơn một nửa số khách hàng rời bỏ, cho thấy khả năng cảnh báo churn đã tốt hơn.

Macro average $F1 = 0.74$ và weighted average $F1 = 0.80$, chứng tỏ mô hình hoạt động cân bằng và ổn định trên cả hai lớp.

So với Logistic Regression gốc, việc tối ưu siêu tham số giúp tăng cả độ chính xác và F1-score, đặc biệt là cải thiện precision cho lớp 1.

Mô hình vẫn có độ lệch nhẹ về phía lớp 0, nhưng không còn chênh lệch quá lớn như trước.

Nhìn chung, việc tinh chỉnh siêu tham số đã nâng cao hiệu suất tổng thể của mô hình Logistic Regression, giúp mô hình dự đoán cân bằng hơn và phát hiện tốt hơn nhóm khách hàng có nguy cơ rời bỏ.



Mô hình XGBoost

```

from xgboost import XGBClassifier

# Khởi tạo mô hình với tham số tốt nhất
xgb_model = XGBClassifier(learning_rate=0.2, max_depth=7, use_label_encoder=False, eval_metric='logloss')

# Huấn luyện mô hình
xgb_model.fit(X_train, y_train)

# Dự đoán
y_pred_xgb = xgb_model.predict(X_test)

print(f"Classification Report : \n", classification_report(y_test, y_pred_xgb))

```

	precision	recall	f1-score	support
0	0.84	0.86	0.85	1033
1	0.59	0.54	0.56	374
accuracy			0.78	1407
macro avg	0.71	0.70	0.71	1407
weighted avg	0.77	0.78	0.77	1407

*Nhận xét:

Mô hình XGBoost đạt độ chính xác (accuracy) là 0.78, tức là dự đoán đúng khoảng 78% tổng số mẫu kiểm thử.

Lớp 0 (khách hàng không rời bỏ) có precision = 0.84, recall = 0.86, F1-score = 0.85, thể hiện mô hình nhận diện rất tốt nhóm khách hàng ở lại.

Lớp 1 (khách hàng rời bỏ) có precision = 0.59, recall = 0.54, F1-score = 0.56, cho thấy khả năng phát hiện khách hàng rời bỏ ở mức trung bình.

Recall của lớp 1 đạt 0.54, nghĩa là mô hình phát hiện được khoảng một nửa số khách hàng rời bỏ, vẫn còn bỏ sót khá nhiều trường hợp.

Macro average F1 = 0.71 cho thấy sự cân bằng giữa hai lớp chưa cao nhưng vẫn ổn định.

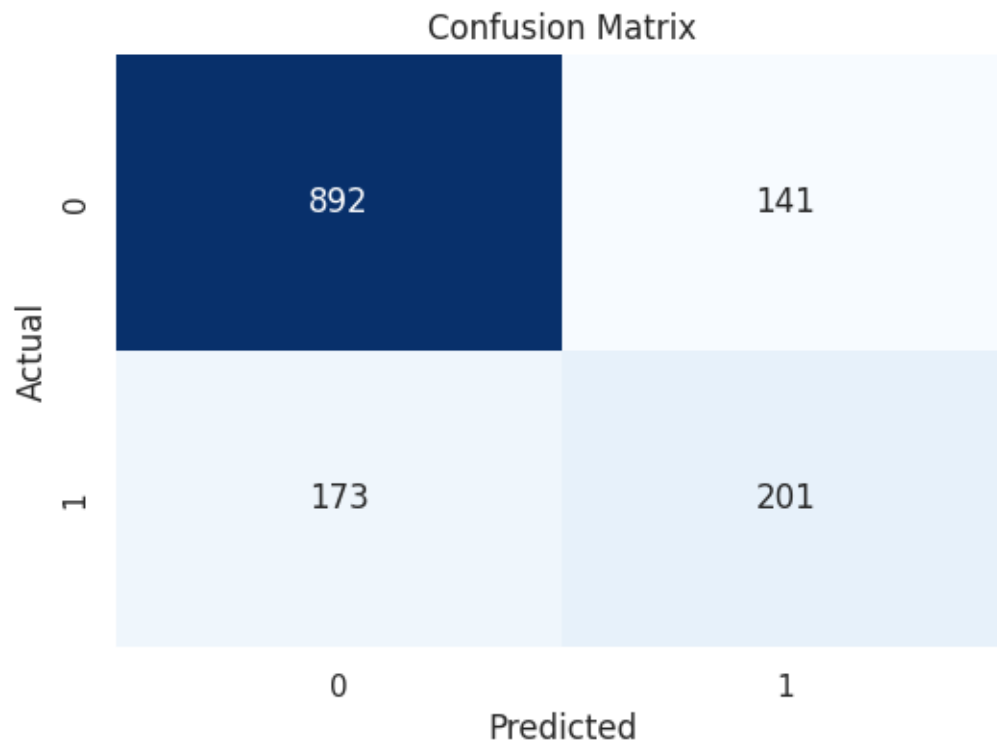
Weighted average F1 = 0.77 phản ánh hiệu suất tổng thể khá tốt, chủ yếu nhờ mô hình dự đoán chính xác nhóm chiếm đa số (lớp 0).

Mô hình vẫn có xu hướng nghiêng về lớp 0, dễ dự đoán khách hàng ở lại hơn là rời bỏ.

Nhìn chung, XGBoost mang lại hiệu quả tổng thể khá cao, nhưng để cải thiện khả năng nhận diện khách hàng rời bỏ (class 1), có thể cần tinh chỉnh thêm siêu


tham số (learning_rate, max_depth, scale_pos_weight) hoặc kết hợp kỹ thuật xử lý mất cân bằng dữ liệu (SMOTE, class weighting).

- ➔ Nhân xét chung: có thể thấy được mô hình cho ra kết quả cũng không khác nhiều so với lúc chưa tuning



5.5 Threshhold mô hình

5.5.1 Mô hình XGBoost

 ✔ Best threshold: 0.201 Best F1-score: 0.601					
Classification Report (with best threshold):					
	precision	recall	f1-score	support	
0	0.88	0.75	0.81	1033	
1	0.51	0.72	0.60	374	
accuracy			0.74	1407	
macro avg	0.70	0.74	0.71	1407	
weighted avg	0.78	0.74	0.76	1407	
Classification Report (default threshold = 0.5):					
	precision	recall	f1-score	support	
0	0.83	0.85	0.84	1033	
1	0.54	0.51	0.52	374	
accuracy			0.76	1407	
macro avg	0.68	0.68	0.68	1407	
weighted avg	0.75	0.76	0.75	1407	

Mô hình XGBoost ban đầu sử dụng ngưỡng mặc định 0.5, sau khi tinh chỉnh ngưỡng tối ưu threshold = 0.201, đạt F1-score tốt nhất là 0.601.

Ở ngưỡng mặc định (0.5), mô hình đạt accuracy = 0.76, F1-score của lớp 1 = 0.52, cho thấy khả năng phát hiện khách hàng rời bỏ còn hạn chế.

Sau khi giảm ngưỡng xuống 0.201, mô hình đạt accuracy = 0.74, tức giảm nhẹ về độ chính xác tổng thể, nhưng đổi lại F1-score của lớp 1 tăng từ 0.52 lên 0.60.

Recall của lớp 1 tăng mạnh từ 0.51 → 0.72, nghĩa là mô hình phát hiện được nhiều hơn các trường hợp khách hàng rời bỏ.

Tuy nhiên, precision của lớp 1 giảm nhẹ từ 0.54 \rightarrow 0.51, phản ánh việc tăng số dự đoán dương tính (nhiều dự đoán “rời bỏ” hơn), kéo theo một số sai lệch.


Macro average F1 tăng từ 0.68 \rightarrow 0.71, và weighted average F1 tăng từ 0.75 \rightarrow 0.76, cho thấy hiệu suất tổng thể của mô hình được cải thiện sau khi thay đổi threshold.

Lớp 0 (khách hàng không rời bỏ) bị giảm recall từ 0.85 xuống 0.75, nhưng precision tăng lên 0.88 — mô hình trở nên thận trọng hơn khi dự đoán “không rời bỏ”.

Nhìn chung, việc tối ưu ngưỡng phân loại giúp mô hình XGBoost cân bằng tốt hơn giữa precision và recall, đặc biệt tăng khả năng phát hiện khách hàng rời bỏ (class 1), rất hữu ích cho mục tiêu cảnh báo sớm churn.

Kết luận: điều chỉnh threshold về 0.201 là hợp lý — giúp tăng khả năng nhận diện khách hàng rời bỏ mà không làm giảm đáng kể hiệu suất tổng thể của mô hình.

5.5.2 Mô hình Logistic Regression

 Best threshold: 0.409 Best F1-score: 0.630					
Classification Report (with best threshold):					
	precision	recall	f1-score	support	
0	0.88	0.83	0.85	1033	
1	0.59	0.68	0.63	374	
accuracy			0.79	1407	
macro avg	0.73	0.75	0.74	1407	
weighted avg	0.80	0.79	0.79	1407	
Classification Report (default threshold = 0.5):					
	precision	recall	f1-score	support	
0	0.85	0.89	0.87	1033	
1	0.65	0.57	0.61	374	
accuracy			0.80	1407	
macro avg	0.75	0.73	0.74	1407	
weighted avg	0.80	0.80	0.80	1407	

Mô hình Logistic Regression sử dụng ngưỡng mặc định 0.5 và ngưỡng tối ưu 0.409, trong đó ngưỡng tối ưu cho F1-score cao nhất = 0.630.

Ở ngưỡng mặc định (0.5), mô hình đạt accuracy = 0.80, F1-score lớp 1 = 0.61, cho thấy mô hình hoạt động tốt tổng thể nhưng khả năng nhận diện khách hàng rời bỏ vẫn còn hạn chế.

Sau khi giảm ngưỡng xuống 0.409, accuracy giảm nhẹ còn 0.79, nhưng F1-score của lớp 1 tăng lên 0.63, thể hiện mô hình cân bằng hơn giữa precision và recall.

Recall của lớp 1 tăng mạnh từ 0.57 → 0.68, nghĩa là mô hình phát hiện được thêm nhiều khách hàng có nguy cơ rời bỏ — rất có ý nghĩa trong bài toán churn prediction.

Precision của lớp 1 giảm nhẹ từ $0.65 \rightarrow 0.59$, điều này là bình thường khi recall tăng (mô hình dự đoán “rời bỏ” nhiều hơn, chấp nhận thêm sai số để không bỏ sót).

Lớp 0 (khách hàng không rời bỏ) vẫn duy trì F1-score cao (0.85), chứng tỏ mô hình không bị ảnh hưởng nhiều ở nhóm chiếm đa số.

Macro average F1 tăng từ $0.74 \rightarrow 0.74$ (ổn định) và weighted average F1 giảm rất nhẹ từ $0.80 \rightarrow 0.79$, cho thấy mô hình vẫn giữ được hiệu suất tổng thể tốt.

Nhìn chung, việc tối ưu threshold giúp mô hình Logistic Regression phát hiện tốt hơn nhóm khách hàng rời bỏ, đồng thời duy trì hiệu suất cao và ổn định cho nhóm không rời bỏ.

Kết luận: Ngưỡng 0.409 là lựa chọn hợp lý, giúp tăng khả năng cảnh báo sớm khách hàng rời bỏ (class 1) mà không làm giảm đáng kể độ chính xác tổng thể của mô hình.

5.6 Xác thực chéo (cross-validation)

```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression

model = LogisticRegression(C=1, penalty='l1', solver='liblinear')
scores = cross_val_score(model, X, y, cv=5, scoring='f1')
# cv = 5 là 4 phần train - 1 phần test

# In từng giá trị F1 của mỗi fold
print("F1-score từng fold:", scores)

# In trung bình F1
print("F1-score trung bình:", scores.mean())
```

```
F1-score từng fold: [0.59710145 0.62393162 0.56891496 0.6167147 0.59084195]
F1-score trung bình: 0.5995009352806983
```

Mô hình Logistic Regression đạt F1 trung bình ≈ 0.60 , cho thấy hiệu suất dự đoán mức trung bình, độ ổn định giữa các fold tương đối đồng đều.

```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression

model = XGBClassifier(learning_rate=0.2, max_depth=7, use_label_encoder=False, eval_metric='logloss')
scores = cross_val_score(model, X, y, cv=5, scoring='f1')

# In từng giá trị F1 của mỗi fold
print("F1-score từng fold:", scores)

# In trung bình F1
print("F1-score trung bình:", scores.mean())
```

F1-score từng fold: [0.54571843 0.57641921 0.54189944 0.56676558 0.59334298]
F1-score trung bình: 0.5648291295294334

Mô hình XGBoost đạt F1 trung bình ≈ 0.56 , hiệu suất dự đoán ở mức trung bình khá, các fold có độ ổn định tương đối đồng đều.

5.7 Phân cụm khách hàng

	Cluster	tenure	MonthlyCharges	TotalCharges	SeniorCitizen	Churn
0	0	57.242667	88.629227	5056.322213	0.0	14.45
1	1	20.581818	49.396351	838.359477	0.0	27.95
2	2	33.295972	79.820359	2810.465193	1.0	41.68

Nhân xét các cụm:

Cụm 0 – Nhóm khách hàng trung thành, giá cao, churn thấp

Nhận xét:

Là nhóm khách hàng gắn bó lâu dài và chi tiêu cao hàng tháng.

Dù trả nhiều tiền, tỷ lệ rời bỏ thấp, cho thấy mức độ hài lòng cao với dịch vụ.

Có thể là khách hàng VIP / trung thành, cần duy trì & chăm sóc kỹ:

- Ưu đãi đặc biệt
- Chương trình tri ân

- Dịch vụ hỗ trợ riêng

Cụm 1 – Khách hàng mới, phí thấp, churn trung bình

Nhận xét:

Là nhóm khách hàng mới, thời gian sử dụng ngắn.

Chi tiêu thấp → Có thể đang dùng gói cơ bản hoặc ưu đãi khuyến mãi.

Tỷ lệ churn trung bình → Có nguy cơ rời bỏ sau khi hết khuyến mãi.

Đề xuất:

- Chính sách giữ chân nhẹ (giảm giá, tặng thêm dung lượng, nâng cấp gói).
- Tăng tương tác, nhắc gia hạn sớm.

Cụm 2 – Người lớn tuổi, chi phí cao, churn cao

Nhận xét:

Là nhóm khách hàng lớn tuổi, dùng dịch vụ lâu và chi trả cao hàng tháng.

Tuy nhiên, tỷ lệ rời bỏ cao nhất (~41.7%), nguyên nhân có thể do:

- Khó thích ứng với công nghệ mới.
- Cảm thấy phí cao so với nhu cầu.

Giải pháp đề xuất:

- Tăng cường hỗ trợ kỹ thuật / tư vấn cá nhân.
- Cung cấp gói dịch vụ phù hợp hơn hoặc chính sách giảm giá.
- Đây là nhóm nguy cơ churn cao, cần ưu tiên can thiệp sớm.

5.8 Giao diện web

Hình ảnh giao diện web

Dự đoán Khách hàng rời bỏ dịch vụ

gender (Giới tính):
Male

SeniorCitizen (Khách hàng cao tuổi):
0

Partner (Có bạn đời/đối tác):
Yes

Dependents (Có người phụ thuộc):
Yes

tenure (Thời gian gắn bó - tháng):
0

PhoneService (Dịch vụ điện thoại):
Yes

MultipleLines (Nhiều đường dây):
Yes

InternetService (Loại Internet):
DSL

CHƯƠNG 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1 Kết luận

Để áp dụng mô hình tốt nhất và thực tiễn thì ta tiến hành tạo một trang web localhost áp dụng mô hình vào. Các bước thực hiện cơ bản:

1. **Cài đặt thư viện cần thiết:** Sử dụng pip để cài đặt streamlit, pyngrok và các thư viện khác để tạo một giao diện web đơn giản cho mô hình.
2. **Lưu mô hình:** Sử dụng joblib để lưu mô hình đã được huấn luyện (best_model) vào một file có tên churn_model.pkl. Điều này cho phép chúng ta sử dụng lại mô hình mà không cần huấn luyện lại.

3. **Tạo ứng dụng Streamlit:** Viết code giao diện trong colab sử dụng Streamlit để tạo một giao diện người dùng. Giao diện này cho phép người dùng nhập thông tin của một khách hàng mới và sử dụng mô hình đã lưu để dự đoán khả năng rời bỏ của khách hàng đó.
4. **Thiết lập ngrok:** Sử dụng pyngrok để tạo một tunnel công khai đến ứng dụng Streamlit đang chạy cục bộ trên Colab. Điều này giúp bạn có thể truy cập ứng dụng từ bất kỳ đâu thông qua một URL công khai.
5. **Chạy ứng dụng Streamlit**

Dựa trên quá trình phân tích và xây dựng mô hình, đã hoàn tất xây dựng một mô hình dự đoán khách hàng rời bỏ (churn) với độ chính xác và khả năng phát hiện churn khá tốt (được thể hiện qua chỉ số ROC AUC).

Các yếu tố chính có ảnh hưởng mạnh mẽ đến quyết định rời bỏ của khách hàng bao gồm:

- **Thời gian gắn bó (Tenure):** Khách hàng mới (tenure thấp) có xu hướng churn cao hơn.
- **Loại hợp đồng (Contract):** Khách hàng sử dụng hợp đồng tháng-tới-tháng (Month-to-month) có tỷ lệ churn cao hơn đáng kể so với các loại hợp đồng dài hạn hơn.
- **Loại dịch vụ Internet (InternetService):** Khách hàng sử dụng Fiber optic có tỷ lệ churn cao hơn so với DSL.
- **Tổng chi phí (TotalCharges) và Chi phí hàng tháng (MonthlyCharges):** Có mối tương quan nhất định với churn.
- **Các dịch vụ khác (OnlineSecurity, TechSupport, v.v.):** Việc không sử dụng các dịch vụ bảo mật hoặc hỗ trợ kỹ thuật cũng làm tăng nguy cơ churn.

Mô hình Logistic Regression được chọn làm mô hình tốt nhất dựa trên chỉ số ROC AUC, cho thấy khả năng phân biệt giữa nhóm churn và không churn hiệu quả.

6.2 Hướng phát triển

Để nâng cao hơn nữa hiệu quả của mô hình và ứng dụng thực tế, chúng ta có thể xem xét các hướng phát triển sau:

- **Thu thập thêm dữ liệu:** Bổ sung dữ liệu về hành vi sử dụng dịch vụ chi tiết hơn (ví dụ: thời gian sử dụng, lượng data tiêu thụ), lịch sử tương tác với bộ phận chăm sóc khách hàng, phản hồi khảo sát, thông tin nhân khẩu học chi tiết hơn.
- **Kỹ thuật Feature Engineering nâng cao:** Khám phá và tạo ra các đặc trưng mới từ dữ liệu hiện có, ví dụ: tỷ lệ TotalCharges trên Tenure, số lượng dịch vụ bổ sung mà khách hàng đăng ký.
- **Thử nghiệm các mô hình phức tạp hơn:** Áp dụng các thuật toán học máy nâng cao hơn như Gradient Boosting (XGBoost, LightGBM), mạng nơ-ron (Neural Networks) để xem liệu có cải thiện hiệu suất dự đoán hay không.
- **Tối ưu hóa siêu tham số (Hyperparameter Tuning):** Sử dụng các kỹ thuật như Grid Search hoặc Random Search để tìm ra bộ siêu tham số tốt nhất cho mô hình đã chọn.
- **Xử lý dữ liệu mất cân bằng (Imbalanced Data):** Mặc dù chúng ta đã sử dụng `class_weight='balanced'`, có thể thử nghiệm các kỹ thuật khác như SMOTE để tạo ra dữ liệu tổng hợp cho lớp thiểu số.
- **Giải thích mô hình (Model Interpretability):** Sử dụng các công cụ như LIME hoặc SHAP để hiểu rõ hơn lý do tại sao mô hình đưa ra dự đoán

như vậy cho từng khách hàng cụ thể, giúp bộ phận marketing và chăm sóc khách hàng có hành động phù hợp.

- **Xây dựng Dashboard theo dõi:** Phát triển một dashboard tương tác để theo dõi tỷ lệ churn theo thời gian, phân tích các yếu tố gây churn theo từng phân khúc khách hàng và đánh giá hiệu quả của các chiến dịch giữ chân khách hàng.
- **Triển khai thực tế:** Tích hợp mô hình vào hệ thống CRM (Customer Relationship Management) của công ty để tự động xác định khách hàng có nguy cơ churn cao và kích hoạt các quy trình giữ chân phù hợp.

CHƯƠNG 7: TÀI LIỆU THAM KHẢO

Thuật toán XGBoost 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). ACM.

Khai phá Dữ liệu Larose, D. T. (2014). *Discovering Knowledge in Data: An Introduction to Data Mining*. Wiley.