ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

NGÀNH: KHOA HỌC DỮ LIỆU

MÔN HỌC : PYTHON TRONG KHOA HỌC DỮ LIỆU

BÁO CÁO ĐỒ ÁN CUỐI KÌ

ĐỀ TÀI: PHÂN TÍCH DỮ LIỆU KHÁCH HÀNG VÀ XÂY DỰNG MÔ HÌNH HỌC MÁY DỰ ĐOÁN GÓI DỊCH VỤ VIỄN THÔNG

Sinh viên thực hiện: Nhóm 27

- Nguyễn Thị Hoàng Vi - *23280003*
- Nguyễn Hoàng Anh Thư - *23280027*

Tên giảng viên

Hà Minh Tuấn

Table of Contents

1. Giới thiệu bài toán và dữ liệu:

1.1. Giới thiệu bài toán:

- Bối cảnh:

+ Trong bối cảnh số hóa mạnh mẽ, nhu cầu truy cập Internet ổn định và tốc độ cao của du khách quốc tế đến Việt Nam ngày càng tăng. Internet là công cụ thiết yếu để du khách làm việc, giải trí, tìm kiếm thông tin du lịch và duy trì liên lạc trong suốt hành trình. Vì vậy, việc cung cấp các gói cước 4G được cá nhân hóa theo nhu cầu và hành trình của từng du khách đóng vai trò quan trọng đối với các nhà mạng.

+ **Viettel** – nhà cung cấp dịch vụ viễn thông hàng đầu Việt Nam với mạng lưới phủ sóng rộng khắp – đang đẩy mạnh khai thác phân khúc du khách quốc tế bằng các gói cước 4G phù hợp.

- Bài toán đặt ra:

Trong khuôn khổ dự án "Cá nhân hóa gói cước 4G Viettel", bài toán đặt ra là: Thực hiện xây dựng mô hình **phân loại (classification)** nhằm **dự đoán gói dữ liệu (mobile plan) phù hợp cho từng du khách dựa trên thông tin cá nhân và thông tin chuyến đi**. Mục tiêu là hỗ trợ hệ thống gợi ý lựa chọn gói cước tối ưu cho người dùng, đồng thời giúp doanh nghiệp hiểu rõ hơn đặc điểm của từng nhóm du khách.

- Hướng tiếp cận:

Dựa trên cấu trúc dữ liệu và mục tiêu dự báo, nhóm đề xuất **hướng tiếp cận** cho bài toán phân loại: **Phân loại đa lớp (Multi-class Classification).** Xây dựng một mô hình để dự đoán xem mỗi du khách sẽ chọn **một trong năm gói dữ liệu**.
→ Đây là bài toán phân loại nhiều lớp.

- Kết quả:

Dự đoán giúp Viettel cá nhân hóa trải nghiệm, tối ưu chiến dịch marketing và cải thiện tỷ lệ chuyển đổi khi bán gói cước tại sân bay và điểm du lịch.

1.2. Mô tả bộ dữ liệu:

- Bộ dữ liệu được thu thập bằng cách khảo sát các du khách nước ngoài khi họ mua gói cước của Viettel.
- Bộ dữ liệu bao gồm thông tin về chuyến đi của du khách (mục đích,điểm đến, số người đi cùng, ...) và xem rằng họ sẽ chọn gói cước nào phù hợp với bản thân.

Bảng 1: THUỘC TÍNH NGƯỜI DÙNG (user.csv)

Bao gồm: 11572 hàng × 16 cột

No.	Variable	Description
1	**id**	Mã định danh của bài khảo sát
2	**name**	Tên khách hàng
3	**gender**	Giới tính khách hàng
4	**education**	Trình độ học vấn
5	**profession**	Nghề nghiệp hoặc tình trạng việc làm của du khách.
6	**income**	Thu nhập
7	**living_with**	Tình trạng hôn nhân
8	**nation**	Quốc gia
9	**phone**	Số điện thoại

10	**job**	Nghề nghiệp cụ thể
11	**fb_freq**	Tần suất sử dụng Facebook mỗi ngày (giờ)
12	**yt_freq**	Tần suất sử dụng YouTube mỗi ngày (giờ)
13	**tik_freq**	Tần suất sử dụng TikTok mỗi ngày (giờ)
14	**use_less_then_2GB**	Số lần sử dụng 4G ít hơn 2GB trong một tháng
15	**use_2GB_to_4GB**	Số lần sử dụng 4G trong khoảng 2GB–4GB trong một tháng

Bảng 2: THUỘC TÍNH VỀ CHUYẾN ĐI (context.csv)

Bao gồm: 11572 hàng × 12 cột

No.	Variable	Description
1	**id**	Mã định danh của bài khảo sát
2	**purpose**	Mục đích chuyến đi
3	**go_with**	Người đi cùng
4	**weather**	Thời tiết
5	**time**	Thời điểm đáp xuống sân bay
6	**viettel_no_0**	Được giới thiệu về SIM Viettel trong 15 phút đầu tiên sau khi vào sân bay
7	**viettel_no_1**	Được giới thiệu về SIM Viettel trong 30 phút đầu tiên sau khi vào sân bay
8	**viettel_no_2**	Được giới thiệu về SIM Viettel trong 45 phút đầu tiên sau khi vào sân bay
9	**to_hanoi**	Du khách có tới Hà Nội sau khi rời sân bay hay không
10	**to_other**	Du khách có tới các tỉnh/thành khác sau khi rời sân bay hay không

11	**score**	Điểm Viettel++

Bảng 3: ĐỀ XUẤT GÓI DATA CHO NGƯỜI DÙNG (mobile_plan_user.csv)

Bao gồm: 45321 hàng× 3 cột

No.	Variable	Description
1	**id**	Mã định danh của bài khảo sát (liên kết với người dùng và chuyến đi).
2	**mobile_plan**	Gói Data được đề xuất cho du khách dựa trên thông tin chuyến đi và nhu cầu sử dụng.
3	**accept**	Khách hàng có đồng ý đăng ký gói được đề xuất hay không (0 = Không, 1 = Có).

Bảng 4: THÔNG TIN VỀ GÓI DATA (mobile_plan_attr.csv)

Bao gồm: 5 hàng x 4 cột

No.	Variable	Description
1	**mobile_plan**	Tên gói Data.
2	**price**	Giá bán của gói Data.
3	**description**	Mô tả chi tiết về dung lượng, tốc độ, ưu đãi đi kèm,... của gói Data.
4	**duration**	Thời gian hiệu lực của gói (tính theo ngày hoặc theo chu kỳ).

2. Tiền xử lý dữ liệu (Data Preprocessing):

Nhóm quyết định tách hệ thống tiền xử lý thành hai lớp:

(1) **Class DataPreprocessor** chung dùng cho các bước xử lý chung nhằm đảm bảo tính tái sử dụng, giảm lặp code và dễ bảo trì.

(2) **Class Pipeline** dùng cho từng loại file để xử lý các đặc thù riêng của dữ liệu.

→ Cách thiết kế này giúp hệ thống linh hoạt, mở rộng dễ dàng và tiệm cận với kiến trúc Machine Learning trong thực tế.

2.1. Class DataPreprocessor:

- **Class *DataPreprocessor*** là lớp tiện ích chung để tiền xử lý dữ liệu. Mục tiêu của lớp này là chuẩn hóa các bước cơ bản trước khi dữ liệu được sử dụng trong các pipeline riêng cho từng loại file.
- **Các phương thức chính:**

1. __init__(self)

- Khởi tạo object, tạo thuộc tính self.df để lưu trữ dữ liệu.
- Ban đầu self.df = None.

2. load(self, filepath)

- Đọc dữ liệu từ file CSV và lưu vào self.df.
- Trả về DataFrame đã đọc.
- In ra thông báo khi dữ liệu được đọc thành công.

3. inspect(self)

- Kiểm tra thông tin cơ bản của dữ liệu:
  - Kiểu dữ liệu của từng cột ***(info())***
  - Kích thước dữ liệu ***(shape)***
  - 5 dòng đầu tiên ***(head())***
  - Thống kê mô tả ***(describe())***
  - Tỷ lệ giá trị ***NULL*** và ***duplicate***
- Giúp người dùng nắm được tình trạng dữ liệu trước khi xử lý.

4. handle_missing(self, cols=None, strategy="drop")

- Xử lý giá trị thiếu (NULL) theo cột và chiến lược chỉ định:
  - mean, median, mode → điền giá trị trung bình, trung vị, mode.

    - unknown → điền giá trị "Unknown" cho cột dạng string.
    - drop → loại bỏ các dòng có giá trị missing.
- Hỗ trợ xử lý nhiều cột cùng lúc.
- Trả về DataFrame đã xử lý.

5. remove_duplicates(self)

- Xóa các dòng dữ liệu trùng lặp trong DataFrame.
- In ra số dòng bị loại bỏ.
- Giúp đảm bảo dữ liệu duy nhất, không gây sai lệch khi phân tích.

6. export(self, path)

- Xuất DataFrame hiện tại ra file CSV.
- Giữ nguyên dữ liệu sau khi tiền xử lý để sử dụng trong pipeline khác.

2.2. Class DatasetPipeline:

- Class ***DatasetPipeline*** là **pipeline riêng biệt cho từng file dữ liệu**, sử dụng ***class DataPreprocessor*** để thực hiện các bước tiền xử lý chung, đồng thời xử lý các cột đặc thù của từng bảng.
- ***Các phương thức chính:***

2.2.1. __init__(self, preprocessor)

- Nhận object DataPreprocessor (preprocessor) để gọi các phương thức tiền xử lý cơ bản.
- Lưu object này vào self.p.

2.2.2. process_user(self, path_in, path_out)

- Xử lý dữ liệu file user.csv.
- **Các bước:**

1. Load dữ liệu, inspect thông tin cơ bản.
2. Xử lý missing ở cột education bằng giá trị "Unknown".

+ Cột **education** có **4003/11572 (34.6%) giá trị bị thiếu.**

→ **Nhóm** thay thế bằng **"Unknown"** để tránh mất dữ liệu, giữ nguyên số lượng mẫu và giúp mô hình xử lý được nhóm người dùng không xác định trình độ học vấn.

+ Vì tỷ lệ thiếu lên đến hơn ***30%***, nếu xóa hoặc thay bằng ***median/mean*** sẽ làm mất một lượng lớn dữ liệu, gây sai lệch phân phối và ảnh hưởng đến độ chính xác của mô hình.

3. Xóa duplicate.
4. Chuẩn hóa cột income: loại bỏ dấu phẩy, chuyển USD sang VND.

   Cột **income** trong tập dữ liệu ban đầu tồn tại dưới nhiều định dạng khác nhau, bao gồm:

   + Giá trị có dấu phân cách hàng nghìn (,)
   + Giá trị có khoảng trắng dư thừa
   + Gồm hai đơn vị tiền là $ và VND

→ Cột income được làm sạch bằng cách loại bỏ **dấu phẩy và khoảng trắng,** sau đó quy đổi toàn bộ về đơn vị VND **(1 USD = 24,000 VND)**. Cuối cùng, dữ liệu được c**huyển sang dạng số** để phục vụ cho quá trình huấn luyện mô hình.

5. Tách cột living_with thành 2 cột: marriage_status và number_child.

   Cột **"living_with"** trong tập dữ liệu ban đầu tồn tại vấn đề:

   + Khoảng trắng thừa bị chèn ngẫu nhiên vào giữa chữ
   + Lỗi chính tả do nhập liệu
   + Thông tin bị gộp chung 2 ý nghĩa trong cùng một cột, bao gồm:
     1. Tình trạng hôn nhân (Single, Married, Unmarried, Widowed)
     2. Số lượng con (_0, _1, _3, _4, ...)

→ Xóa khoảng trắng thừa và tách cột **Living_with** thành 2 cột **Marriage_status** và **Number_child** để thuận tiện cho việc phân tích.

6. Chuẩn hóa cột nation sang dạng Title Case.
   + Đặc trưng **Nation** được đổi thành dạng chữ ***Title Case*** để đồng nhất với các đặc trưng còn lại.
7. Đồng nhất ngôn ngữ cột job sang tiếng Anh bằng Google Translator.

+ Đặc trưng **Job** có nhiều ngôn ngữ khác nhau, tiến hành chuyển về cùng một ngôn ngữ là tiếng Anh

8. Xuất file đã xử lý.

2.2.3. process_context(self, path_in, path_out)

- Xử lý dữ liệu file context.csv.
- **Các bước:**
    1. Load và inspect dữ liệu.
    2. Xử lý missing và duplicate.
    3. Chuẩn hóa cột go_with và weather: loại bỏ ký tự đặc biệt, chuyển sang Title Case.

Các **vấn đề chính** gồm:

- Xuất hiện nhiều **ký tự đặc biệt** như **~, %, &, !, ?**
- Khoảng trắng **thừa** chèn giữa từ
- Cùng một ý nghĩa nhưng bị ghi dưới nhiều **dạng khác nhau**, gây:
    - Sai lệch khi thống kê tần suất
    - Số lượng category bị phình to không cần thiết
    - Khó khăn trong quá trình mã hóa dữ liệu cho mô hình

→ Cột go_with, weather chứa nhiều lỗi ký tự đặc biệt và khoảng trắng gây trùng lặp giả. Nhóm sử dụng **Regex** để loại bỏ toàn bộ ký tự không phải chữ cái và chuẩn hóa chữ hoa, giúp dữ liệu đồng nhất và dễ mã hóa cho mô hình.

4. Chuẩn hóa cột time sang 24h (HH:00) với hàm convert_time.

+ Cột time tồn tại nhiều định dạng khác nhau *("7:00", "7:00 AM", "7PM")*.

→ Nhóm xây dựng hàm convert_time để đưa toàn bộ dữ liệu về định dạng chuẩn **24h HH:00** (*"7:00", "7:00 AM", "7PM" → đều được chuẩn hóa về "07:00" hoặc "19:00" tương ứng)*, giúp dữ liệu đồng nhất và sẵn sàng cho mô hình.

5. Chuẩn hóa 3 cột viettel_no_0, viettel_no_1, viettel_no_2 theo logic one-hot (chỉ một cột = 1).

Ba biến **viettel_no_0, viettel_no_1, viettel_no_2** được sử dụng để mô tả **thời điểm khách hàng được giới thiệu SIM Viettel**, tương ứng với các mốc:

- *viettel_no_0: 15 phút đầu*
- *viettel_no_1: 30 phút đầu*
- *viettel_no_2: 45 phút đầu*

Tuy nhiên, dữ liệu gốc lại được ghi nhận theo **dạng tích lũy (cummulative encoding)** :

- Nếu khách được giới thiệu trong **15 phút đầu** → viettel_no_0 = 1
- Nếu trong **30 phút đầu** → viettel_no_0 = 1, viettel_no_1 = 1
- Nếu trong **45 phút đầu** → cả **3 cột đều bằng 1**

Cách ghi nhận này gây ra các vấn đề nghiêm trọng:

- Chồng lấn thông tin giữa các mức thời gian
- Một khách hàng có thể đồng thời thuộc nhiều nhóm thời gian
- Làm sai lệch ý nghĩa của biến khi đưa vào mô hình
- Vi phạm nguyên tắc one-hot encoding.

Để phản ánh đúng ý nghĩa và đảm bảo mỗi khách hàng chỉ thuộc duy nhất một mốc thời gian, nhóm tiến hành chuẩn hóa lại ba biến này về dạng **one-hot encoding thực sự**.

Nhóm xây dựng hàm **fix_viettel(row)** theo nguyên tắc:

- Nếu viettel_no_2 = 1 → khách thuộc nhóm 45 phút → (0, 0, 1)
- Nếu viettel_no_1 = 1 → khách thuộc nhóm 30 phút → (0, 1, 0)
- Nếu viettel_no_0 = 1 → khách thuộc nhóm 15 phút → (1, 0, 0)
- Nếu cả 3 đều bằng 0 → giữ nguyên (0, 0, 0)

6. Xuất file đã xử lý.

2.2.4. process_mobile_plan_user(self, path_in, path_out, path_out_agg=None)

- Xử lý dữ liệu file mobile_plan_user.csv.
- **Các bước:**
    1. Load, inspect, xử lý missing (mobile_plan và accept) bằng drop.

+ Tỷ lệ null **~6%** là thấp → bỏ đi không ảnh hưởng chất lượng phân tích.

→ Việc này giúp đảm bảo tính chính xác của các biến tổng hợp và giữ dữ liệu ở trạng thái hoàn chỉnh trước khi phân tích.

2. Xóa duplicate.
3. Tạo bảng tổng hợp theo id:
   - Do bảng **mobile_plan_user** có nhiều dòng cho mỗi khách hàng (mỗi dòng tương ứng với một gói cước được đề xuất), việc join trực tiếp sang các bảng khác sẽ gây **nhân dòng và làm sai lệch kết quả phân tích**.
   - **Cách xử lý:** tạo ra các **đặc trưng tổng hợp theo ID** nhằm gom toàn bộ thông tin của khách hàng về một dòng và mô tả đầy đủ hành vi đề xuất – chấp nhận gói cước.
   - **Các đặc trưng mới bao gồm:**
     - **number_of_suggestions**: số lần được đề xuất (tổng số dòng theo ID)
     - **number_of_plans**: số gói khác nhau được đề xuất
     - **number_of_accept**: tổng số gói được khách hàng chấp nhận.
     - **number_of_accept_2**: tổng số gói khác nhau mà khách hàng chấp nhận đăng ký.
     - **mobile_plan**: liệt kê tên các gói unique khách hàng đã đăng ký, nếu không đăng ký gói nào thì nhận giá trị **Unknown**.
4. Xuất file cleaned và file tổng hợp nếu có.

2.2.5. process_mobile_plan_attr(self, path_in, path_out)

- Xử lý dữ liệu file mobile_plan_attr.csv.
- **Các bước:**
  1. Load, inspect.
  2. Xử lý missing và duplicate.
  3. Xuất file đã xử lý.

3. Khám phá dữ liệu (Data Exploratory Analysis):

3.1. Phân tích khẩu nhân học:

3.1.1. Giới tính và Tuổi tác:

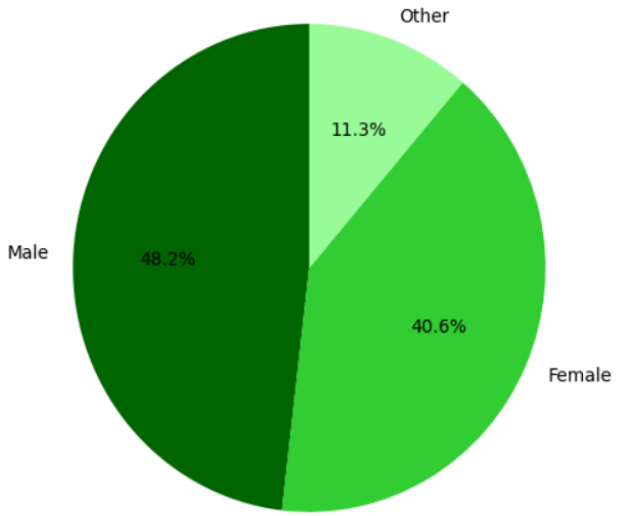

Figure 1. Tỉ lệ phần trăm theo giới tính

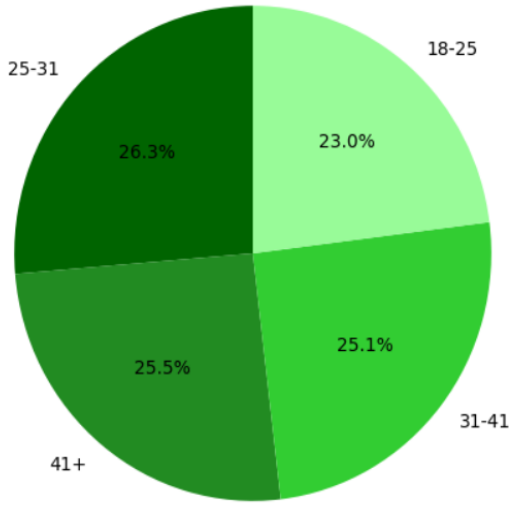

Figure 2. Tỉ lệ phần trăm theo nhóm tuổi

- Nam chiếm tỷ lệ cao nhất **(48.2%)**, tiếp theo là nữ **(40.6%)**. Giới tính khác chiếm tỉ lệ khá nhỏ.
- Các nhóm tuổi phân bố **khá đồng đều**, trong đó nhóm **25–31 cao nhất**, nhóm **18–25 thấp nhất**. Tổng thể tập trung chủ yếu ở độ tuổi lao động 25–41.

Insight: Tập khách hàng chủ lực nằm trong độ tuổi lao động trẻ, có khả năng tiếp cận công nghệ tốt.

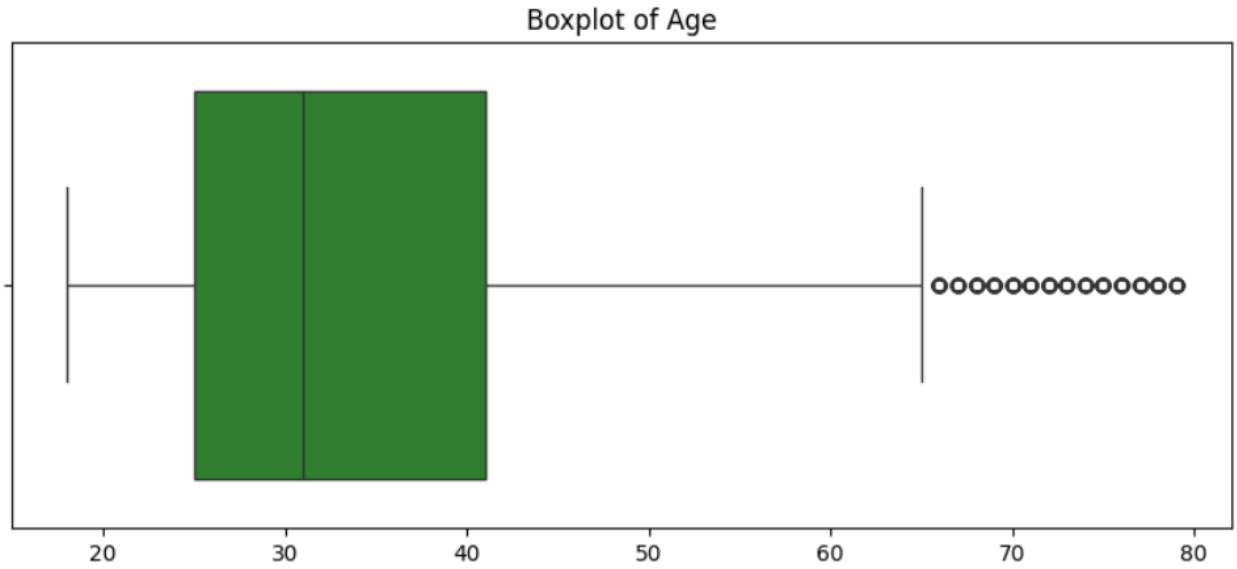

Figure 3. Boxplot của tuổi tác

- Tuổi **trung vị** nằm khoảng **30–35**
- Phần lớn người dùng trong khoảng **25–42** tuổi.
- Phân bố **lệch phải nhẹ**, cho thấy đa số là **người trẻ**, nhưng vẫn xuất hiện một số **outliers trên 65 tuổi**.

3.1.2. Trình độ học vấn (Education Level):

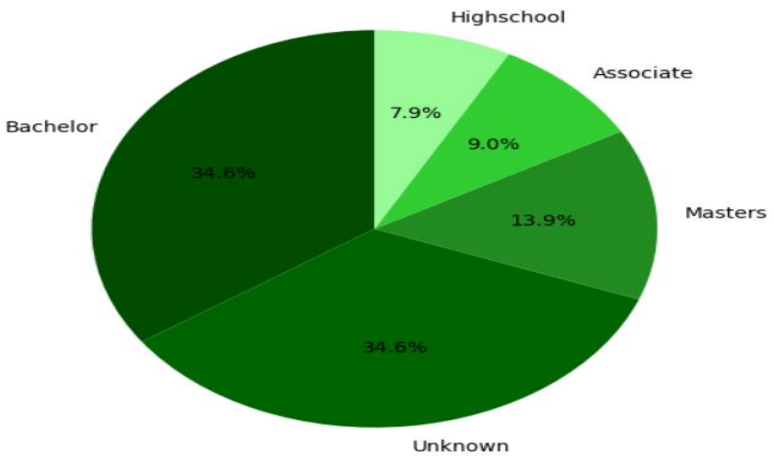

Figure 4. Tỉ lệ phần trăm theo trình độ học vấn

Tỷ lệ trình độ học vấn **Bachelor** và **Unknown (không rõ)** là như nhau, cụ thể **34.6%**. 3 nhóm còn lại là **Masters**, **Associate** và **Highschool** chiếm tỷ lệ dưới **15%** mỗi nhóm.

3.1.3. Tình trạng nghề nghiệp (Profession):

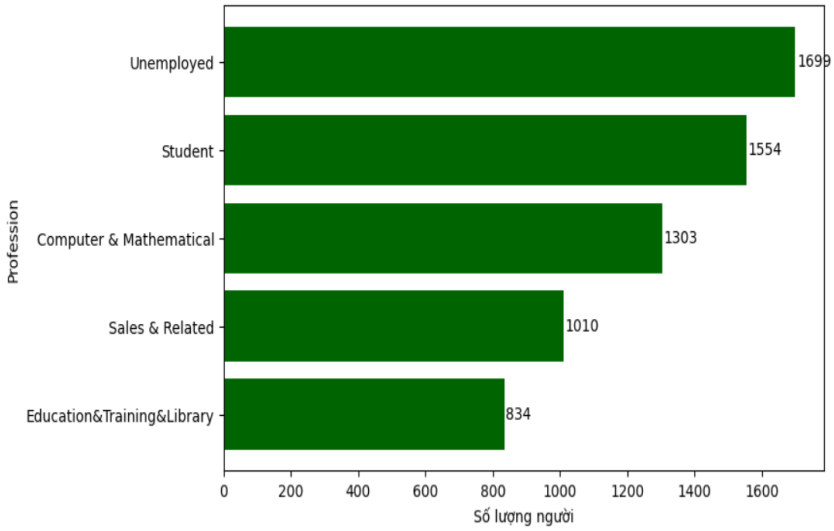

Figure 5. Top 5 nghề có số lượng người nhiều nhất

Nhóm **Unemployed** và **Student** chiếm tỷ lệ **cao nhất**, tiếp theo là **Computer & Mathematical, Sales & Related và Education/Training/Library**.

→Tập du khách chủ yếu đến từ **nhóm chưa đi làm chính thức** (sinh viên, thất nghiệp) và **nhóm ngành công nghệ**, những đối tượng có xu hướng sử dụng Internet nhiều.

3.1.4. Khu vực (Nation):

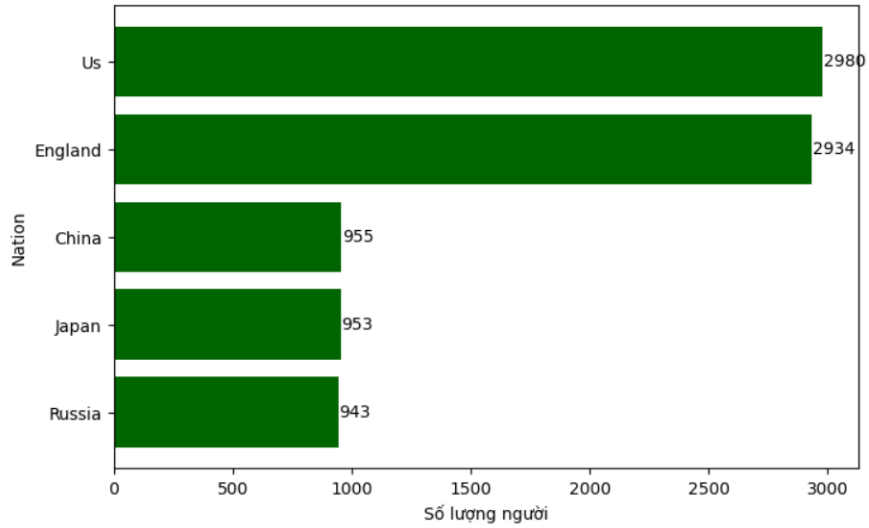

Figure 6. Top 5 quốc tịch có số lượng người cao nhất

Khách hàng chủ yếu đến từ **Mỹ và Anh**, với số lượng **vượt trội** so với các quốc gia còn lại.

Insight: Cần ưu tiên các nội dung tiếp thị bằng tiếng Anh và các gói cước quốc tế phù hợp với thói quen của người phương Tây.

3.1.5. Tình trạng hôn nhân và Số lượng con:

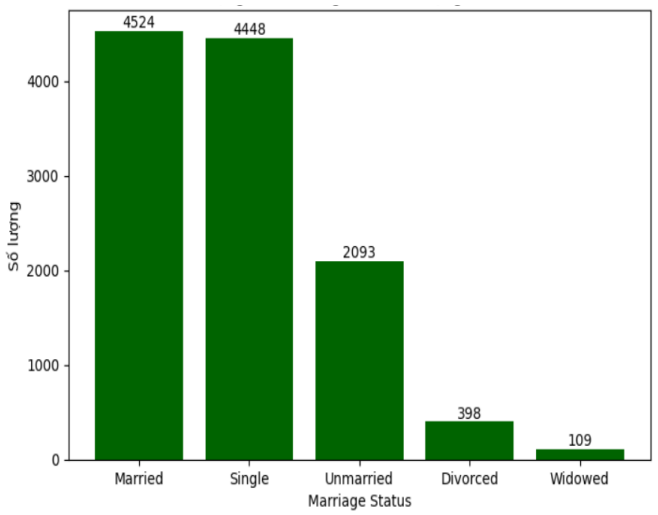

Figure 7. Số lượng khách hàng theo tình trạng hôn nhân

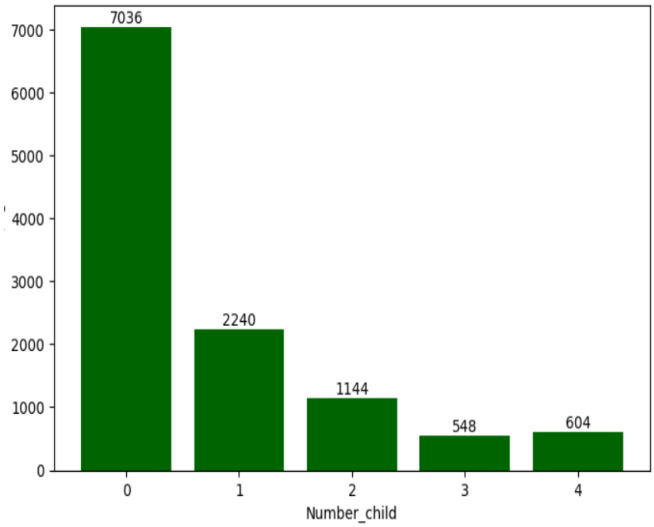

Figure 8. Số lượng khách hàng theo số lượng con cái

Phần lớn khách hàng là **đã kết hôn và độc thân**, trong đó hai nhóm này chiếm **áp đảo** so với các nhóm còn lại.Tuy nhiên, phần lớn chưa có con **(7036 người)**. Số lượng giảm dần theo số con, trong đó nhóm có **1–2 con chiếm ít hơn đáng kể**, còn nhóm **3–4 con rất ít.**

3.1.6. Mức thu nhập:

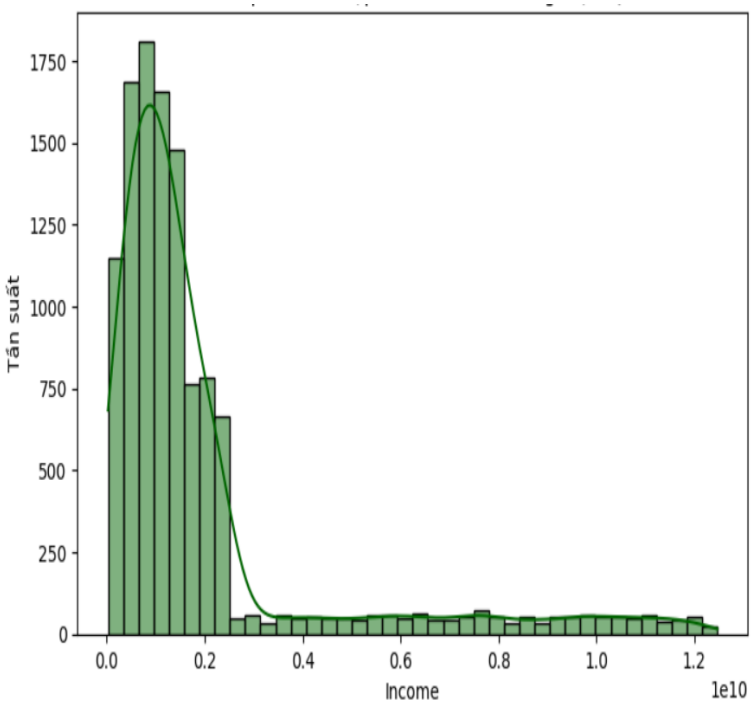

Figure 9. Phân phối thu nhập (Income)

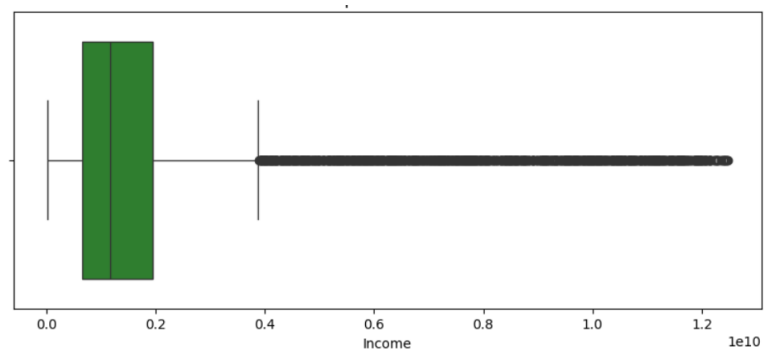

Figure 10. Boxplot của thu nhập (Income)

Thu nhập có phân bố **lệch phải mạnh**: phần lớn khách hàng thuộc nhóm thu nhập **thấp–trung bình**, trong khi chỉ một nhóm rất nhỏ có thu nhập cực cao làm đuôi phân phối kéo dài rõ rệt.

Insight:

- *Việt Nam được xem là một trong 10 điểm đến du lịch rẻ nhất châu Á (Báo Tuổi Trẻ, 2023). Do đó khá dễ hiểu khi những người có thu nhập thấp sẽ ưu tiên du lịch tại Việt Nam để phù hợp với mức thu nhập.*
- *Bên cạnh đó, lý do xuất hiện nhóm thu nhập siêu cao có thể là do mức lương không đồng đều giữa các quốc gia, châu lục, điển hình như Mỹ, Đan Mạch luôn nằm trong top 10 quốc gia có lương trung bình cao nhất thế giới (CEOWORLD, 2022).* ***Vì vậy, đây không được xem là các giá trị ngoại lai.***

3.2. Phân tích hành vi khách hàng:

3.2.1. Mục đích và hình thức chuyến đi:

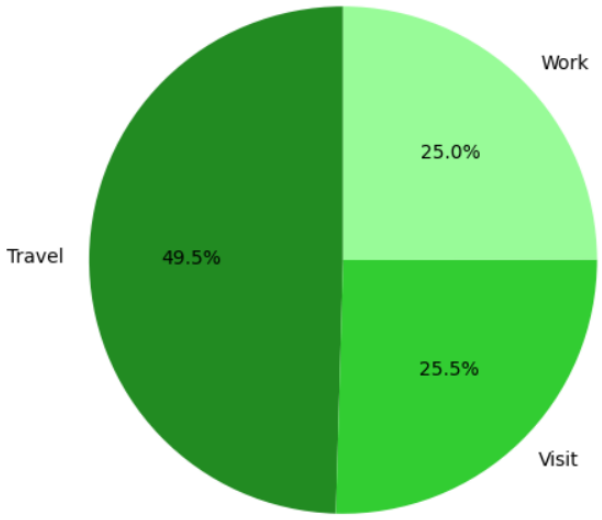

Figure 11. Tỉ lệ khách hàng theo mục đích chuyến đi

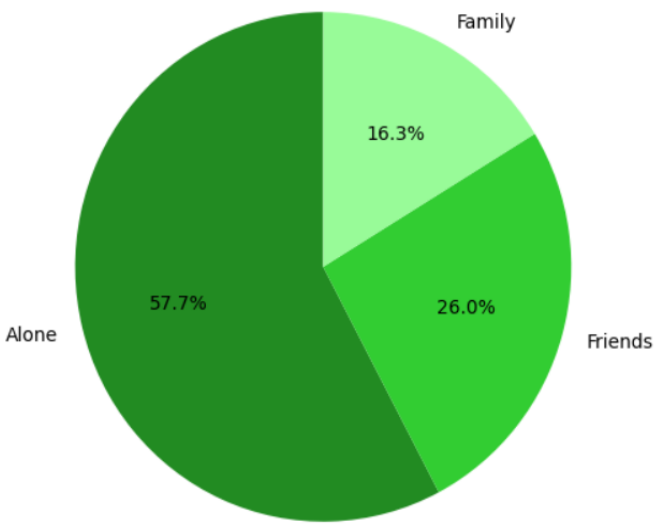

Figure 12. Tỉ lệ khách hàng theo đối tượng đi cùng

Phần lớn du khách đến Việt Nam với mục đích **du lịch** ***(49.5%)***, trong khi mục đích **thăm viếng và làm việc** chiếm tỷ lệ **thấp hơn** nhưng khá tương đương nhau.

Phần lớn du khách **đi một mình (solo traveller)** ***(57.7%)***, tiếp theo là đi cùng **bạn bè**, trong khi nhóm đi cùng **gia đình** chiếm tỷ lệ thấp nhất.

3.2.2. Thời gian và địa điểm :

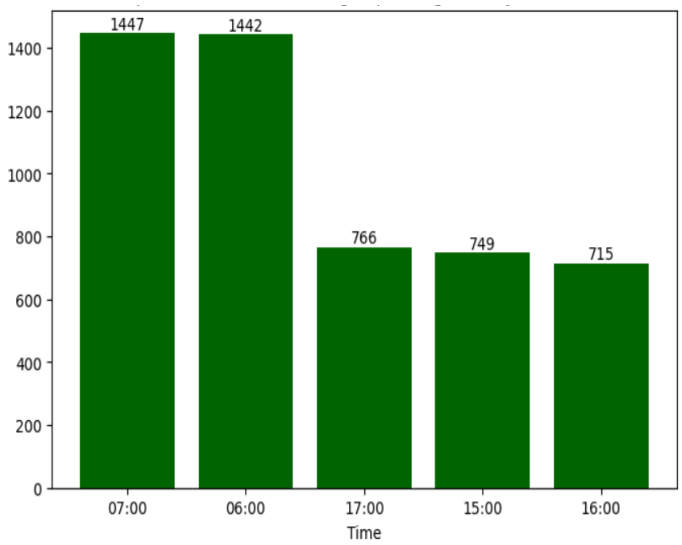

Figure 13. Top 5 thời điểm khách hàng đáp xuống máy bay nhiều nhất

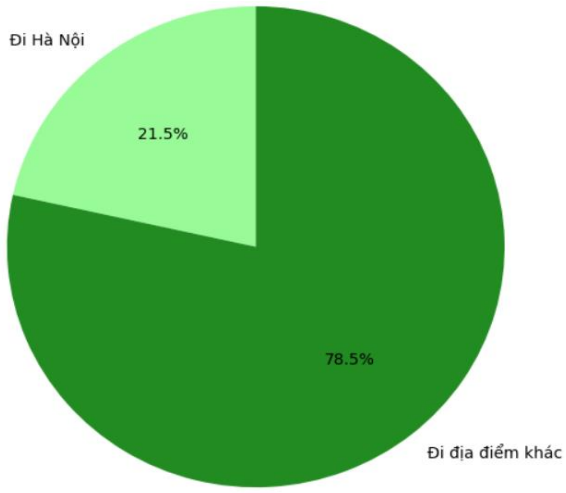

Figure 14. Địa điểm di chuyển sau khi đáp máy bay theo số khách hàng

Thời gian đến: Khách hàng thường đáp máy bay vào **sáng sớm**, sau đó là buổi chiều.
Điểm đến: Sau khi đáp sân bay, **78.5%** khách hàng di chuyển đi các địa điểm khác thay vì tới Hà Nội.

- *Insight:* Cần tiếp cận khách hàng ngay tại sân bay vào khung giờ sáng sớm hoặc cung cấp các gói cước roaming/data hỗ trợ di chuyển liên tỉnh.

3.2.3. Điểm thành viên (Viettel Score) :

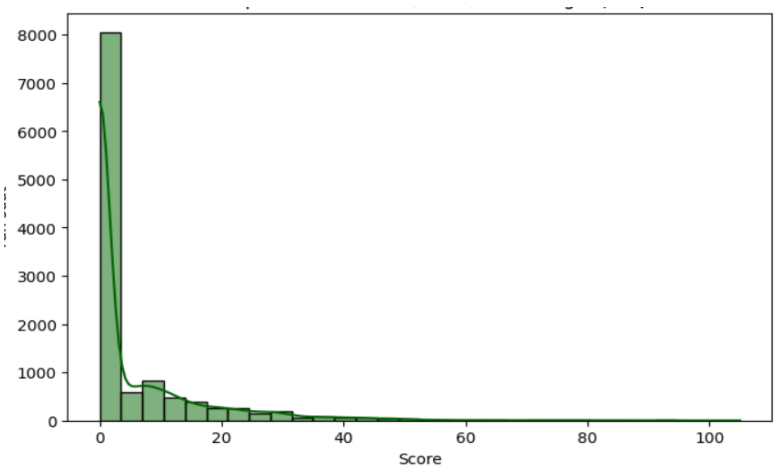

Figure 15. Phân phối Viettel Score

Phân phối **score lệch phải**, đa số khách hàng không có hoặc có rất ít điểm, chỉ một số ít đạt điểm cao, tạo đuôi dài về bên phải.

Insight: Đây chủ yếu là khách hàng mới (***New Users***), chưa có lịch sử gắn bó lâu dài với dịch vụ.

3.2.4. Tỉ lệ chấp nhận (Acceptance Rate)

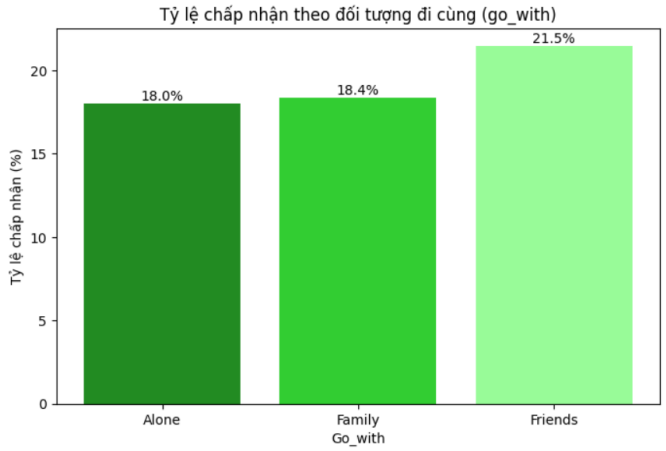

Figure 16. Tỉ lệ chấp nhận (acceptance rate) theo đối tượng đi cùng (go_with)

Biểu đồ cho thấy sự khác biệt rõ rệt về tỷ lệ chấp nhận dịch vụ/gói cước dựa trên người đi cùng:

- **Nhóm đi cùng bạn bè (Friends):** Có tỷ lệ chấp nhận cao nhất (**21.5%**).
- **Nhóm đi một mình:** Tỷ lệ thấp hơn mặc dù số lượng đông nhất.
- *Insight quan trọng:* Hiệu ứng đám đông hoặc nhu cầu kết nối khi đi theo nhóm bạn bè làm tăng khả năng mua hàng.

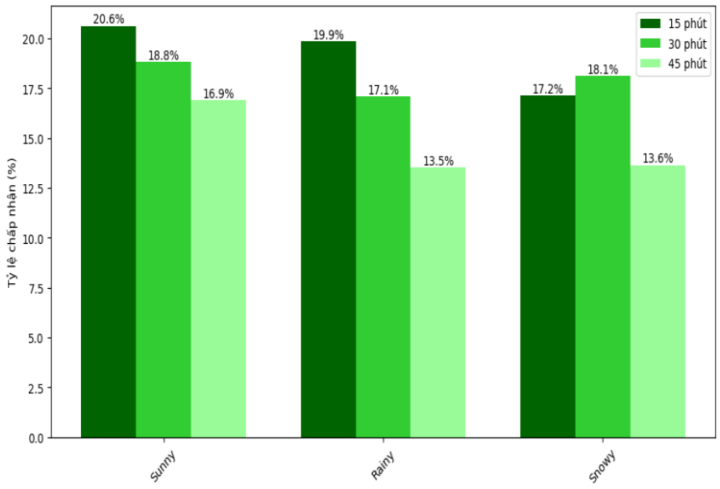

Figure 17. Tỉ lệ chấp nhận (acceptance rate) theo thời điểm được tiếp thị và thời tiết

Tiếp thị sớm trong **15 phút đầu** mang lại tỷ lệ chấp nhận cao nhất, đặc biệt **khi thời tiết nắng ráo**. Hiệu quả **giảm dần khi tiếp thị muộn** hơn (30–45 phút).

*Insight: Càng để lâu, khả năng khách hàng từ chối càng cao. → **Khi trời nắng/đẹp:** Tận dụng tối đa 15 phút đầu, đây là lúc tâm lý khách hàng thoải mái nhất để mua hàng.*

3.3. Phân tích sản phẩm :

3.3.1. Tỉ lệ đăng kí mỗi gói:

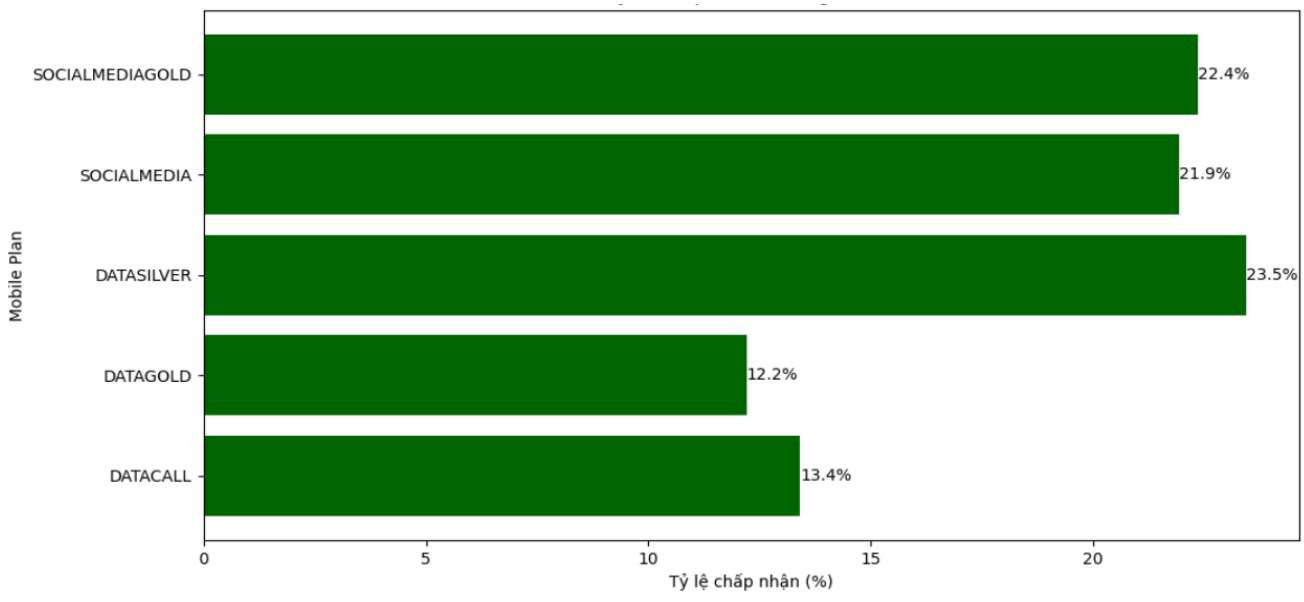

Figure 18. Tỉ lệ đăng kí theo mỗi gói data

- **DATASILVER (23.5%):** Gói cước có tỷ lệ chuyển đổi cao nhất. Lý do có thể là giá cả phải chăng, phù hợp với nhu cầu cơ bản của khách du lịch ngắn ngày hoặc sinh viên.
- **SOCIALMEDIAGOLD (22.4%) & SOCIALMEDIA (21.9%):** Các gói tập trung vào mạng xã hội có hiệu suất rất tốt, phản ánh nhu cầu check-in, liên lạc qua OTT của khách du lịch hiện đại.
- **DATACALL (13.4%):** Khách quốc tế ít có nhu cầu gọi thoại nội địa.
- **DATAGOLD (12.2%):** Tỷ lệ thấp nhất, có thể do mức giá cao hoặc dung lượng quá dư thừa so với nhu cầu thực tế.

Price Sensitivity: Khách hàng ưu tiên gói DATASILVER hơn DATAGOLD, khẳng định lại chân dung khách hàng trẻ/sinh viên tiết kiệm chi phí.

Social First: Nhu cầu Mạng xã hội và Data lấn át hoàn toàn nhu cầu Thoại.

Chiến lược: Tập trung đẩy mạnh DATASILVER và SOCIALMEDIA làm sản phẩm chủ lực. Cân nhắc tái cấu trúc hoặc giảm giá cho DATAGOLD

3.3.2. Số lượng đăng kí theo khách hàng :

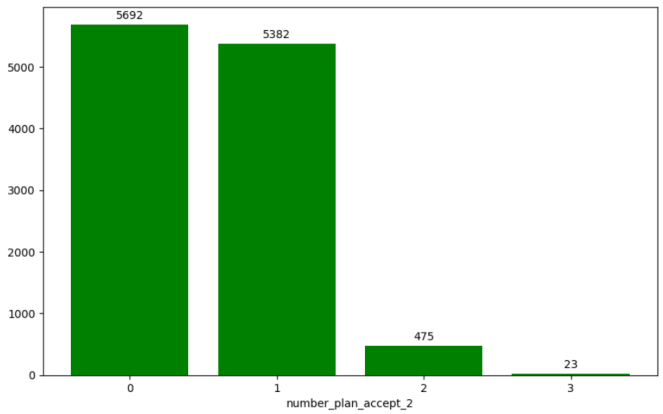

*Figure 19. Số gói đăng kí **khác nhau** theo số lượng khách hàng*

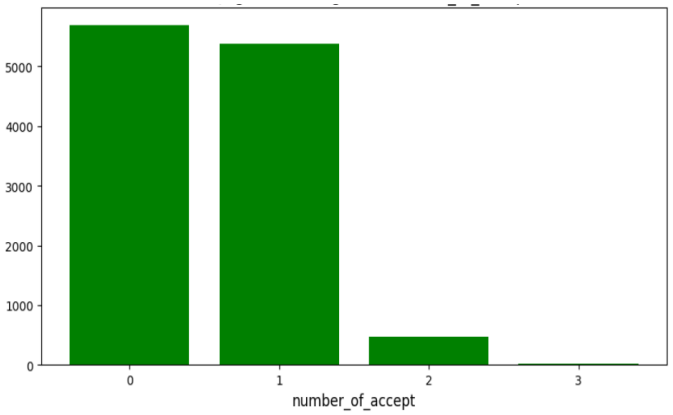

Figure 20. Số gói đăng kí theo số lượng khách hàng

Nhóm phổ biến (0-1 gói): Chiếm tỷ trọng áp đảo với tổng cộng hơn 11.000 khách hàng. Số lượng khách hàng chấp nhận 1 gói (5.382) gần tương đương với số lượng khách hàng chưa đăng ký (5.692), cho thấy khả năng thu hút khách hàng mới (acquisition) ở mức ổn định.

Nhóm cao cấp (2-3 gói): Có sự gãy khúc lớn về số lượng khi chuyển từ 1 gói sang 2 gói (giảm từ 5.382 xuống 475). Số lượng khách hàng sở hữu 3 gói là không đáng kể (23 khách hàng).

Insight: Hiện tượng này chỉ ra rằng mặc dù doanh nghiệp thành công trong việc bán gói đầu tiên, nhưng việc thúc đẩy khách hàng đăng ký thêm các gói bổ sung (cross-sell) đang gặp rào cản lớn.

3.3.3. Tổng khách hàng theo số lượng đăng kí & đề xuất:

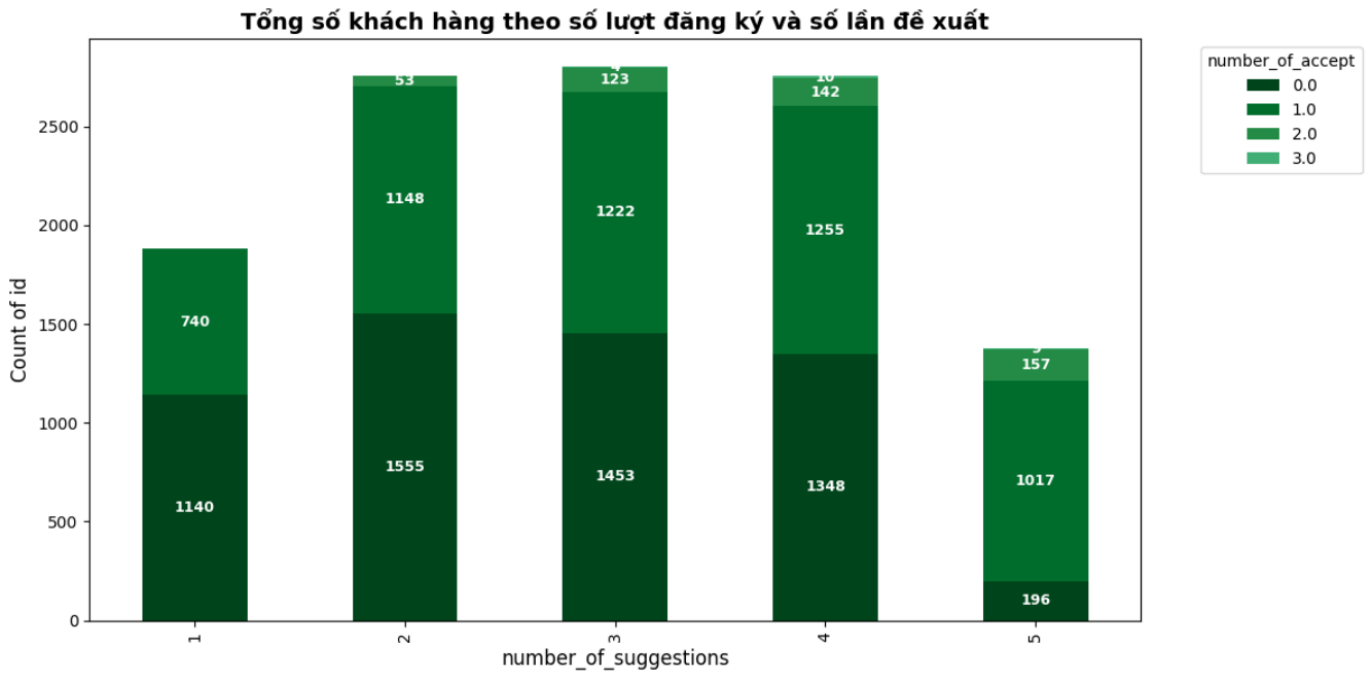

Figure 21 . Số lượng khách hàng theo số lượt đề xuất và đăng kí

- Khi số lần đề xuất là **2 hoặc 3 lần**, tổng số khách hàng tương tác là cao nhất.
- Tuy nhiên, khi đề xuất quá nhiều (**5 lần**), số lượng khách hàng giảm mạnh. Điều này cho thấy việc spam đề xuất không mang lại hiệu quả, thậm chí gây phản tác dụng.

Số lượt chấp nhận (Accept):

- Phần lớn khách hàng chỉ chấp nhận **1 lần** (màu xanh đậm thứ 2 từ dưới lên).
- Rất ít khách hàng chấp nhận 2 hoặc 3 lần (các phần màu xanh nhạt hơn ở trên cùng).

Insight:

- *Chỉ nên đề xuất tối đa 2-3 lần. Đề xuất quá nhiều lần thứ 4, 5 không làm tăng đáng kể lượng khách hàng chấp nhận mà còn làm giảm trải nghiệm.*
- *Các gói data được đề xuất chưa thật sự phù hợp với nhu cầu của khách hàng.*

→ Đề xuất sử dụng các mô hình học máy để nâng cao hiệu quả đồng thời giảm chi phí ở việc tiếp thị.

4. Mô hình (Model):

4.1. Đề xuất phương pháp xây dựng mô hình phân loại: Phân loại Đa lớp (Multiclass Classification)

Mô tả phương pháp: Đây là hướng tiếp cận truyền thống, coi bài toán là việc phân loại một khách hàng vào duy nhất **một trong 5 nhóm** gói cước (ví dụ: chỉ chọn gói DATASILVER hoặc chỉ chọn SOCIALMEDIA).

Xử lý dữ liệu: Chỉ lọc và sử dụng các mẫu dữ liệu từ những khách hàng đang sử dụng đúng **01 gói cước** (number_of_accept == 1).

4.2. Giới thiệu thư viện sử dụng:

4.2.1. Thư viện xử lý dữ liệu và tiện ích chung

Thư viện	Chức năng chính	Vai trò trong dự án
pandas	Quản lý dữ liệu dạng bảng (DataFrame)	Tiền xử lý dữ liệu, gộp dữ liệu, làm sạch dữ liệu
numpy	Xử lý số học nhanh, array, toán tuyến tính	Nền tảng cho ML pipeline, xử lý vector/matrix
joblib	Lưu trữ và tải mô hình nhanh (serialization)	Lưu mô hình đã train để deploy/tái sử dụng
logging	Ghi log quá trình chạy	Theo dõi pipeline và debug
json	Lưu cấu hình, metadata	Lưu các tham số mô hình, kết quả

random, time	Sinh ngẫu nhiên, đo thời gian thực thi	Kiểm soát reproducibility và benchmarking

4.2.2. Trực quan hóa dữ liệu:

Thư viện	Chức năng chính	Vai trò trong dự án
matplotlib	Vẽ biểu đồ cơ bản, kiểm soát chi tiết	Vẽ ROC curve, learning curve, feature importance
seaborn	Trực quan hóa nâng cao	Vẽ heatmap, confusion matrix, phân phối dữ liệu

4.2.3. Thư viện tiền xử lý và pipeline (Scikit-learn)

Thư viện	Chức năng chính	Vai trò trong dự án
train_test_split	Chia dữ liệu train/test	Chuẩn bị dữ liệu cho training
LabelEncoder	Mã hóa nhãn	Chuyển nhãn sang dạng số
OneHotEncoder	Mã hóa categorical dạng one-hot	Xử lý biến phân loại
MultiLabelBinarizer	Mã hóa multi-label	Dùng cho bài toán nhiều nhãn
StandardScaler	Chuẩn hóa dữ liệu	Giúp SVM, LR học tốt hơn
ColumnTransformer	Áp dụng các bước preprocessing theo cột	Tách numerical/categorical để xử lý riêng
Pipeline	Chuỗi xử lý hoàn chỉnh	Đảm bảo quy trình nhất quán trong training/predict

4.2.4. Thư viện đánh giá mô hình

Thư viện	Chức năng chính	Vai trò trong dự án
accuracy_score	Đo độ chính xác	Đánh giá classification
precision_score	Độ chính xác (precision)	Đánh giá mô hình imbalance
recall_score	Tỷ lệ bắt đúng (recall)	Quan trọng trong fraud/medical
f1_score	Trung bình hòa giữa precision & recall	Đánh giá tổng thể
classification_report	Báo cáo đầy đủ các metric	Đánh giá theo từng lớp
confusion_matrix	Ma trận nhầm lẫn	Xác định lỗi mô hình
roc_curve, auc	Vẽ ROC, tính diện tích dưới đường cong	Đánh giá khả năng phân biệt lớp
label_binarize	Chuẩn hóa nhãn phục vụ ROC multi-class	Dùng trong ROC-AUC đa lớp

4.2.5. Thư viện chọn lọc đặc trưng:

Thư viện	**Chức năng chính**	**Vai trò trong dự án**
SelectKBest (f_classif)	Chọn K feature dựa trên thống kê ANOVA	Feature selection nhanh cho dữ liệu tuyến tính
RFE	Loại bỏ đặc trưng theo mô hình	Lựa chọn feature dựa trên quan trọng của mô hình

4.2.6. Thư viện các mô hình máy học:

Mô hình	**Nhóm**	**Đặc điểm chính**	**Lý do sử dụng**

RandomForestClassifier	Ensemble (Bagging)	Mạnh mẽ, ít overfitting, interpretability cao	Phù hợp dữ liệu tabular
LogisticRegression	Linear Model	Đơn giản, dễ diễn giải	Baseline tốt
SVC	Support Vector Machine	Mạnh trên dữ liệu nhỏ, phi tuyến tốt	Tạo benchmark
XGBClassifier	Boosting	Mạnh, chính xác cao	Tối ưu cho dữ liệu lớn
LGBMClassifier	Boosting	Nhanh nhất, hiệu quả với high-dimensional data	Tăng tốc huấn luyện
CatBoostClassifier	Boosting	Xử lý categorical nội bộ	Tối thiểu preprocessing

4.2.6. Thư viện tối ưu siêu tham số:

Thư viện	Chức năng chính	Ưu điểm
GridSearchCV	Tìm kiếm toàn bộ không gian tham số	Chính xác nhất nhưng chậm
RandomizedSearchCV	Lấy mẫu ngẫu nhiên	Nhanh, hiệu quả với không gian lớn
PredefinedSplit	Dùng validation set tùy chỉnh	Kiểm soát strict train/val split (industrial use cases)

4.3. Tổng quan Class ModelTrainer:

- Class **ModelTrainer** được thiết kế để **quản lý toàn bộ pipeline học máy**, từ tiền xử lý dữ liệu, feature engineering, lựa chọn đặc trưng, huấn luyện, tối ưu tham số, đánh giá, đến lưu/truy xuất mô hình.
- Tổng quan kiến trúc lớp ModelTrainer:

1. Load dataset
2. Feature generation (static)

3. Preprocessing (numerical, categorical, encoder)
4. Feature selection (KBest hoặc RFE)
5. Split train/val
6. Downsample
7. Build pipeline
8. Train
9. Hyperparameter tuning
10. Evaluate
11. Run model
12. Save model
13. Load mode

- **Các phương thức chính và công dụng:**

4.3.1. Hàm khởi tạo __init__ :

- Phương thức **__init__()** đóng vai trò là **constructor** của lớp *ModelTrainer*, chịu trách nhiệm thiết lập toàn bộ cấu hình ban đầu cần thiết cho quy trình huấn luyện mô hình.
- **Các Thành Phần và Chức Năng Chính:**

Thành phần	Công dụng
self.model_type, self.params	Lưu cấu hình mô hình
self.random_state + seed	Đảm bảo tái lập kết quả
self.label_encoder, self.pipeline, self.selector	Khởi tạo các thành phần xử lý dữ liệu và mô hình
self.logger + handlers	Theo dõi quá trình huấn luyện, ghi log ra file và console

4.3.2. load_data(self, df, target_col):

Mục đích : Chuẩn bị dữ liệu đầu vào cho các bước tiền xử lý và huấn luyện tiếp theo, đồng thời ghi log thông tin cơ bản.

Mục tiêu:

- Nạp dữ liệu vào class từ DataFrame df.
- Xác định cột mục tiêu (target) target_col.
- Lưu dữ liệu và tên cột mục tiêu vào thuộc tính của class.
- Ghi log thông tin về dữ liệu (số dòng, số cột).

Tóm tắt hoạt động

- self.df = df.copy() # sao chép DataFrame để tránh thay đổi dữ liệu gốc
- self.target_col = target_col # lưu tên cột mục tiêu
- self.logger.info(f"Data loaded with shape {self.df.shape}")
- # in thông tin số dòng và cột của dữ liệu

4.3.3. feature_generation(df):

Chức năng: Tạo các đặc trưng (feature) mới từ dữ liệu gốc nhằm nâng cao hiệu quả mô hình.

Loại phương thức: @staticmethod - không cần truy cập thuộc tính của class, chỉ dựa vào dữ liệu truyền vào.

1. **Tổng tần suất sử dụng mạng xã hội:**
    - Cộng các cột fb_freq, tik_freq, yt_freq thành cột freq.
2. **Tỷ lệ chấp nhận đề xuất (accept_rate):**
    - Tính bằng number_of_accept / number_of_suggestions.
    - Xử lý giá trị thiếu (NaN) bằng 0.
3. **Phân loại thu nhập (income bins):**
    - Chia thu nhập thành các nhóm: <6M, 6–12M, 12–20M, >20M.
4. **Nhóm tuổi (age_group):**
    - Gom các giá trị tuổi thành 4 nhóm: 18–25, 25–31, 31–41, 41+.
5. **Khoảng thời gian trong ngày (time buckets):**
    - Chuyển cột time thành giờ, sau đó tạo các cột nhị phân: time_morning, time_afternoon, time_evening, time_night.

Kết quả: DataFrame mới chứa các **feature có ý nghĩa**, sẵn sàng cho các bước tiền xử lý, chọn đặc trưng, và huấn luyện mô hình.

4.3.4. preprocess(self, cat_cols=None):

Mục đích: Tiền xử lý dữ liệu trước khi huấn luyện mô hình, bao gồm: mã hóa nhãn mục tiêu, xử lý biến categorical, chuẩn hóa biến numerical.Phương thức này tạo điều kiện cho pipeline huấn luyện hoạt động trơn tru, đồng thời cung cấp log chi tiết để theo dõi các bước tiền xử lý.

Loại phương thức: Instance method

Chức năng chính:

1. **Encode target**: Chuyển cột mục tiêu thành dạng số (LabelEncoder).
2. **Xác định biến categorical**: Lấy các cột object/category, loại bỏ target.
3. **Xác định biến numerical**: Lấy các cột số, loại bỏ target.
4. **Scaler numerical**:
    - StandardScaler cho LR, SVM.
    - Giữ nguyên (passthrough) cho RF, XGB, LGBM, CatBoost.
5. **Tạo ColumnTransformer**:
    - Numerical → Scaler hoặc passthrough.
    - Categorical → OneHotEncoder (bỏ qua giá trị chưa xuất hiện).
6. **Ghi log**: Thông tin categorical, numerical và trạng thái scaler.

Kết quả:

- self.preprocessor sẵn sàng dùng trong pipeline.
- Dữ liệu được chuẩn hóa, target và feature đã được xử lý phù hợp với mô hình.

4.3.5.feature_selection(self, method="kbest", k_best=30, estimator=None):

Loại phương thức: Instance method
Mục đích: Chọn ra các feature quan trọng nhất để cải thiện hiệu quả mô hình và giảm chiều dữ liệu.
Chức năng chính:

1. SelectKBest (method="kbest")
    - Chọn k_best feature dựa trên f_classif.
    - Ghi log phương pháp, số feature và score function.
2. RFE (method="rfe")
    - Chọn k_best feature bằng Recursive Feature Elimination.

- Dùng estimator (mặc định RandomForestClassifier nếu không truyền).
- Ghi log phương pháp, số feature và tên estimator.

3. Kiểm tra tham số
   - Nếu method không phải "kbest" hoặc "rfe" → raise ValueError.
4. Ghi log chung
   - Thông báo selector đã được tạo, sẵn sàng fit trong pipeline.

Kết quả: self.selector được khởi tạo và sẵn sàng dùng để chọn feature trong pipeline huấn luyện.

4.3.6.split_train_val(val_size=0.2)

Loại phương thức: Instance method

Mục đích: Chia dữ liệu đã tiền xử lý thành tập huấn luyện và tập validation, phục vụ huấn luyện và đánh giá mô hình.

Chức năng chính:

1. **Tách dữ liệu**
   - X_train, X_val: các feature cho train và validation.
   - y_train, y_val: nhãn target cho train và validation.
2. **Stratify**
   Giữ tỷ lệ nhãn target tương tự trong cả train và validation.
3. **Sử dụng random_state**
   - Đảm bảo kết quả tách dữ liệu có thể tái lập.
4. **Ghi log**
   - In ra kích thước của train và validation set.

Kết quả: Dữ liệu được chia sẵn sàng cho việc huấn luyện và đánh giá mô hình.

4.3.7.downsample(self):

Loại phương thức: Instance method

Mục đích: Cân bằng dữ liệu huấn luyện bằng cách giảm số lượng mẫu của các lớp đông đảo, giúp mô hình không bị bias.

Chức năng chính:

1. **Tạo DataFrame huấn luyện kết hợp X và y**
2. **Xác định số lượng mẫu nhỏ nhất (min_size)** theo các lớp target.
3. **Downsampling**
    - Lấy ngẫu nhiên min_size mẫu từ mỗi lớp target.
    - Dùng random_state để đảm bảo reproducibility.
4. **Cập nhật X_train và y_train**
5. **Ghi log:** In ra số lượng mẫu trước và sau khi downsample.

4.3.8.build_pipeline(self):

Mục đích: Phương thức build_pipeline() có nhiệm vụ khởi tạo mô hình học máy dựa trên tham số model_type, sau đó kết hợp *mô hình* với toàn bộ các bước *tiền xử lý* và *chọn đặc trưng* (nếu có) để tạo thành một pipeline hoàn chỉnh phục vụ huấn luyện, đánh giá và dự đoán

Chức năng chính:

- Khởi tạo classifier theo loại mô hình được chỉ định ***(lr, svm, rf, lgbm, xgb, cat).***
- Tự động gắn các siêu tham số từ self.params.
- Xây dựng pipeline với các bước: **preprocessor → selector → classifier**
    1. **preprocessor** — xử lý dữ liệu (encode, scale, impute,...).
    2. **selector** (tuỳ chọn) — chọn đặc trưng nếu đã cấu hình.
    3. **classifier** — mô hình huấn luyện chính.
- Lưu pipeline vào self.pipeline để sử dụng cho train/tuning/predict.
- Log lại thông tin pipeline để thuận tiện debug và theo dõi.

Kết quả: Pipeline đầy đủ, sẵn sàng cho huấn luyện và dự đoán.

4.3.9.train(self):

Mục đích: Phương thức **train()** thực hiện quá trình huấn luyện mô hình bằng dữ liệu đã được chia train/val và đã xây dựng pipeline trước đó. Đây là bước cốt lõi nơi pipeline học từ dữ liệu.

Chức năng chính:

- Ghi log số lượng mẫu dùng để train và loại model.

- Đo thời gian huấn luyện.
- Gọi pipeline.fit() để train toàn bộ pipeline: preprocess → feature selection (nếu có) → classifier.
- Lưu log thời gian hoàn thành.

4.3.10.optimize_params(self, param_grid, search="grid", n_iter=20, scoring="f1_macro"):

Mục đích:

Phương thức optimize_params() dùng để tìm bộ siêu tham số tốt nhất cho mô hình thông qua **Grid Search** hoặc **Randomized Search.** Phương thức được thiết kế để:

- Tìm kiếm siêu tham số trực tiếp trên **pipeline**, đảm bảo tối ưu cả khi có preprocessing và feature selection.
- Sử dụng **PredefinedSplit** để giữ nguyên tập validation (không xáo trộn bởi cross-validation thông thường).
- Cập nhật pipeline với best_estimator_ sau khi tối ưu.

Chức năng chính:

1. Kiểm tra điều kiện trước khi chạy

Phương thức yêu cầu pipeline phải được xây dựng trước (build_pipeline()). Nếu không, hệ thống ném lỗi nhằm tránh tối ưu sai cấu trúc.

2. Thiết lập chiến lược tách tập bằng PredefinedSplit

- Gộp X_train và X_val thành một tập duy nhất.
- Tạo vector test_fold với quy ước:
  - -1 = sample thuộc train
  - 0 = sample thuộc val
- Giúp toàn bộ quá trình tối ưu sử dụng chính xác cặp train/val, không cross-validation ngẫu nhiên.

3. Chuẩn hóa lưới siêu tham số

- Toàn bộ khóa của param grid được đổi thành dạng: classifier__<param_name>
- Phù hợp với cách sklearn tham chiếu tham số trong pipeline nhiều bước.

4. Chọn phương thức tối ưu

- **GridSearchCV**
  - Duyệt toàn bộ tổ hợp param.
  - Dùng khi số lượng param nhỏ – cần độ chính xác cao.
- **RandomizedSearchCV**
  - Lấy mẫu ngẫu nhiên n_iter tổ hợp param.
  - Dùng khi không gian tham số lớn – cần tốc độ.

Cả hai đều chạy đa luồng n_jobs=-1 và dùng metric đánh giá tùy chỉnh, mặc định f1_macro.

5. Huấn luyện bộ tìm kiếm

- Searcher chạy fit trên toàn bộ X_combined, y_combined theo định nghĩa của PredefinedSplit.
- Kết quả trả về best_estimator_ và best_params_.

6. Cập nhật pipeline

- Pipeline gốc được thay thế bằng phiên bản tối ưu nhất.
- Ghi log đầy đủ để theo dõi quá trình.

Kết quả:

- Mô hình trong pipeline đạt cấu hình tối ưu.
- Toàn bộ bước tiền xử lý, chọn feature, và mô hình đều được học và kiểm chứng nhất quán trên tập validation.
- Sẵn sàng chạy đánh giá final hoặc xuất mô hình.

4.3.11.evaluate(self, X=None, y=None, plot_cm=True, plot_roc=True):

Mục đích: Đánh giá hiệu năng mô hình trên tập kiểm tra, sinh báo cáo phân loại, vẽ Confusion Matrix và ROC multi-class.Phương thức này cung cấp góc nhìn đầy đủ về hiệu suất mô hình, giúp định hướng cho các bước cải thiện tiếp theo hoặc đưa vào sử dụng thực tế.

Chức năng chính:

1. Xác định dữ liệu đánh giá

- Nếu không truyền X, y, phương thức tự dùng self.X_test, self.y_test.
- Giúp linh hoạt đánh giá trên tập bất kỳ.

2. Dự đoán và tính các metric chính

- Tính Accuracy, F1-macro, Precision-macro, Recall-macro.
- Ghi log và in kết quả dưới định dạng chuẩn.
- Phù hợp cho multi-class, đảm bảo tính công bằng giữa các lớp.

3. Sinh Classification Report

- In báo cáo theo từng lớp: precision, recall, f1-score.
- Dùng self.label_encoder.classes_ để hiển thị tên lớp chính xác.

4. Vẽ Confusion Matrix (nếu plot_cm=True)

- Tính confusion matrix giữa y_test và y_pred.
- Vẽ heatmap trực quan bằng Seaborn.
- Giúp phân tích chi tiết các lỗi nhầm lẫn giữa các lớp.

5. Vẽ ROC Curve (multi-class) (nếu plot_roc=True)

- One-hot encode nhãn thật (label_binarize).
- Lấy xác suất dự đoán (predict_proba hoặc decision_function).
- Vẽ đường ROC cho từng lớp cùng AUC tương ứng.
- Hữu ích để đánh giá hiệu năng trong bối cảnh multi-class.

Kết quả trả về:

```
{
  "accuracy": ...,
  "f1_macro": ...,
  "precision": ...,
  "recall": ...
}
```

4.3.12.run_models(self, df_train, df_test, target_col, models, k_best=40, param_grids=None, search="grid", n_iter=20, save_path=None, top_n_features=20):

Mục đích:

- Phương thức run_models() tự động hóa toàn bộ quy trình chạy và so sánh nhiều mô hình học máy trong một pipeline thống nhất.

- Nó bao gồm các bước từ feature engineering, preprocessing, feature selection, train/val split, downsampling, xây dựng pipeline, huấn luyện, tối ưu tham số, retraining, đến đánh giá cuối cùng trên tập test.
- Kết quả trả về được cấu trúc rõ ràng để phân tích và có thể lưu thành file CSV/JSON.
- Điều này đảm bảo đánh giá các mô hình một cách khách quan, lặp lại được, và tiện lợi trong báo cáo.

Chức năng chính:

1. Lặp qua danh sách mô hình cần chạy

- Với mỗi model_name trong models, phương thức khởi tạo lại cấu hình mô hình và reset tham số.
- Đảm bảo từng mô hình được xử lý độc lập và thống nhất.

2. Sinh feature cho tập train và test

- Gọi feature_generation() cho cả df_train và df_test.
- Chuẩn bị dữ liệu đầu vào phù hợp cho toàn bộ pipeline.

3. Load dữ liệu huấn luyện

- Gọi load_data() một lần cho toàn bộ mô hình.
- Thiết lập df, target_col, logging.

4. Thực hiện toàn bộ giai đoạn tiền xử lý và huấn luyện

Bao gồm:

- preprocess() – mã hóa target, xử lý categorical/numerical.
- feature_selection() – chọn đặc trưng theo KBest hoặc RFE.
- split_train_val() – chia dữ liệu train/validation.
- downsample() – cân bằng lớp nếu cần.
- build_pipeline() – xây dựng pipeline theo mô hình.
- train() – huấn luyện model ban đầu.

5. Tối ưu siêu tham số (tùy chọn)

- Nếu param_grids được cung cấp, phương thức tự quyết định loại tối ưu: *GridSearch* hoặc *RandomSearch.*
- Có thể nhận param chung hoặc param theo từng mô hình.
- Sau tối ưu, pipeline được cập nhật với mô hình tốt nhất.

6. Huấn luyện lại sau tối ưu

- Nếu có tối ưu, mô hình được train lại trên toàn bộ tập train+val.
- Tăng độ ổn định và khai thác tối đa dữ liệu.

7. Chuẩn bị dữ liệu test

- Không dùng lại load_data() để tránh reset pipeline.
- Gọi lại preprocess() để sinh transformer phù hợp.
- Tách X_test, y_test.

8. Đánh giá mô hình

- Gọi evaluate() để tính Accuracy, Precision, Recall, F1-macro.
- Lưu kết quả vào dict results.

9. Lưu kết quả (tùy chọn)

- Xuất results ra CSV hoặc JSON nếu có save_path.

10. Trả về bảng tổng hợp kết quả

- In toàn bộ kết quả tất cả mô hình.
- Trả ra dict theo dạng {model_name: metrics}.

4.3.13.save_model(path="model.pkl")

Mục đích : Lưu toàn bộ pipeline (gồm preprocessing, feature selection, classifier và tham số đã tối ưu) thành một file duy nhất.

Chức năng chính:

- Đảm bảo mô hình có thể được sử dụng lại mà không cần huấn luyện lại.
- Giữ nguyên trạng toàn bộ cấu trúc pipeline, bao gồm encoder và transformer.
- Ghi log xác nhận mô hình đã được lưu thành công.

4.3.14.load_model(self, path="model.pkl")

Mục đích : Nạp lại pipeline đã lưu từ file.

Chức năng chính:

- Khôi phục mô hình cho inference hoặc đánh giá mà không cần retrain.
- Tái sử dụng toàn bộ preprocessing, encoding, feature selection như lúc train.
- Ghi log xác nhận mô hình đã được nạp thành công.

5. Phân tích kết quả :

Nhóm đã tiến hành chạy **6** mô hình để giải quyết bài toán **Multi-class Classification (5 lớp)**:

1. Logistic Regression (LR)
2. Support Vector Machine (SVM)
3. Random Forest (RF)
4. LightGBM (LGBM)
5. XGBoost (XGB)
6. CatBoost (CB)

Các kĩ thuật nhóm đã sử dụng để tối ưu mô hình:

1. Feature Generation
2. Preprocessing Techniques: Label Encoding, One-Hot Encoding, StandardScaler()
3. Feature Selection: SelectKBest (không chạy RFE vì thời gian chạy quá lâu)
4. Train / Validation Split: Stratified Split (dùng tập val để đánh giá và optimize tham số cho mô hình)
5. Sampling Technique: Downsampling để cân bằng dữ liệu
6. Hyperparameter Optimization: Grid Search & Randomized Search

Các chỉ số nhóm đã sử dụng để đánh giá mô hình: Accuracy, Precision, Recall, F1-score, Confusion Matrix, ROC - AUC.

BỘ THAM SỐ THỬ CHO MỖI MÔ HÌNH

```
lr_param_grid = {
  'C': [0.01, 0.1, 1.0, 10.0],
  'solver': ['lbfgs'], # 'lbfgs' is default, good for multiclass
  'max_iter': [500]
}
svm_param_grid = {
  'C': [0.1, 1.0, 10.0],
```

```
    'gamma': ['scale', 0.01, 0.1],
    'kernel': ['rbf'] # radial basis function kernel is common
}
rf_param_grid = {
    'n_estimators': [100, 200, 500],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 5]
}
lgbm_param_grid = {
    'n_estimators': [100, 300, 500],
    'max_depth': [-1, 10, 20],
    'num_leaves': [20, 31, 50],
    'learning_rate': [0.01, 0.05, 0.1]
}
xgb_param_grid = {
    'n_estimators': [100, 300, 500],
    'max_depth': [3, 6, 10],
    'learning_rate': [0.01, 0.05, 0.1],
    'subsample': [0.7, 0.9],
    'colsample_bytree': [0.7, 0.9]
}
cat_param_grid = {
    'iterations': [100, 300, 500],
    'depth': [4, 6, 8],
    'learning_rate': [0.01, 0.05, 0.1],
    'l2_leaf_reg': [1, 3, 5]
```

}

5.1. Kết quả các chỉ số Accuracy, F1-macro, Precision, Recall:

Model	Accuracy	F1-macro	Precision	Recall	Best Parameters
Logistic Regression	0.429	0.390	0.415	0.386	'classifier__solver':'lbfgs', 'classifier__max_iter': 500, 'classifier__C': 0.1
SVM	0.576	0.538	0.536	0.544	'classifier__C': 10.0, 'classifier__gamma': 'scale', 'classifier__kernel': 'rbf'
Random Forest	0.613	0.602	0.601	0.608	'classifier__max_depth': 20, 'classifier__min_samples_leaf': 1, 'classifier__min_samples_split': 10, 'classifier__n_estimators': 200
LightGBM	0.599	0.579	0.585	0.579	'classifier__learning_rate': 0.05, 'classifier__max_depth': -1, 'classifier__n_estimators': 300, 'classifier__num_leaves': 20
XGBoost	0.606	0.593	0.593	0.605	'classifier__colsample_bytree': 0.7, 'classifier__learning_rate': 0.01, 'classifier__max_depth': 6, 'classifier__n_estimators':10, 'classifier__subsample': 0.9
CatBoost	0.546	0.554	0.562	0.583	'classifier__depth': 8, 'classifier__iterations': 100, 'classifier__l2_leaf_reg': 5, 'classifier__learning_rate': 0.01

Nhận xét:

- Kết quả cho thấy các mô hình dựa trên **cây quyết định** cho hiệu suất tốt nhất. **Random Forest** đạt kết quả cao nhất với Accuracy **0.613** và F1-macro **0.602**, theo sát là **XGBoost** (Accuracy 0.606, F1-

macro 0.593) và **LightGBM** (Accuracy 0.599, F1-macro 0.579). Điều này chứng tỏ các mô hình **phi tuyến** tính phù hợp hơn với cấu trúc dữ liệu.

- Ngược lại, **Logistic Regression** cho kết quả thấp nhất (Accuracy 0.429, F1-macro 0.390), cho thấy mô hình tuyến tính không đủ khả năng nắm bắt quan hệ phức tạp giữa các biến.
- Các mô hình có **Precision** và **Recall khá cân bằng**, thể hiện qua chỉ số **F1-macro ổn định**, đặc biệt ở các mô hình cây.

> *→ Nên triển khai **Random Forest** hoặc **XGBoost**. Tuy nhiên hiệu suất **~61% vẫn còn thấp**, cần cân nhắc các kĩ thuật khác: Feature Engineering nâng cao, cân bằng dữ liệu nâng cao hơn (SMOTE, Upsampling) và thử nghiệm mô hình Deep Learning...*

5.2. Kết quả ma trận nhầm lẫn & đường ROC-AUC:

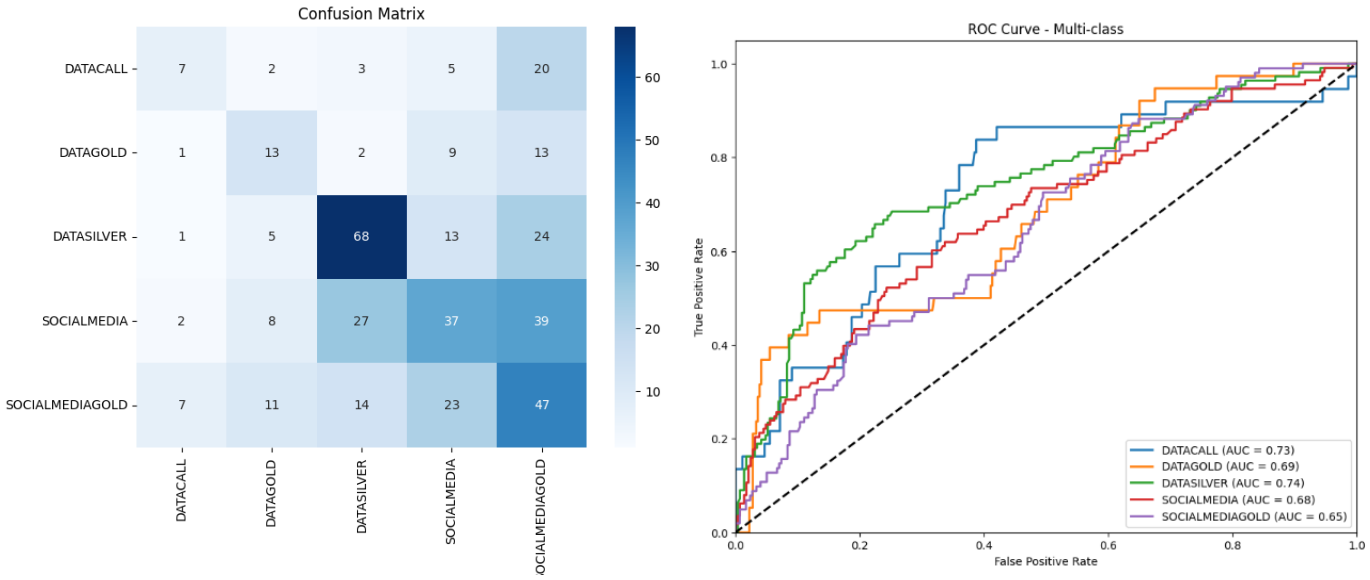

*Figure 22. Confusion Matrix & ROC-AUC Curve của mô hình **Logistic Regression***

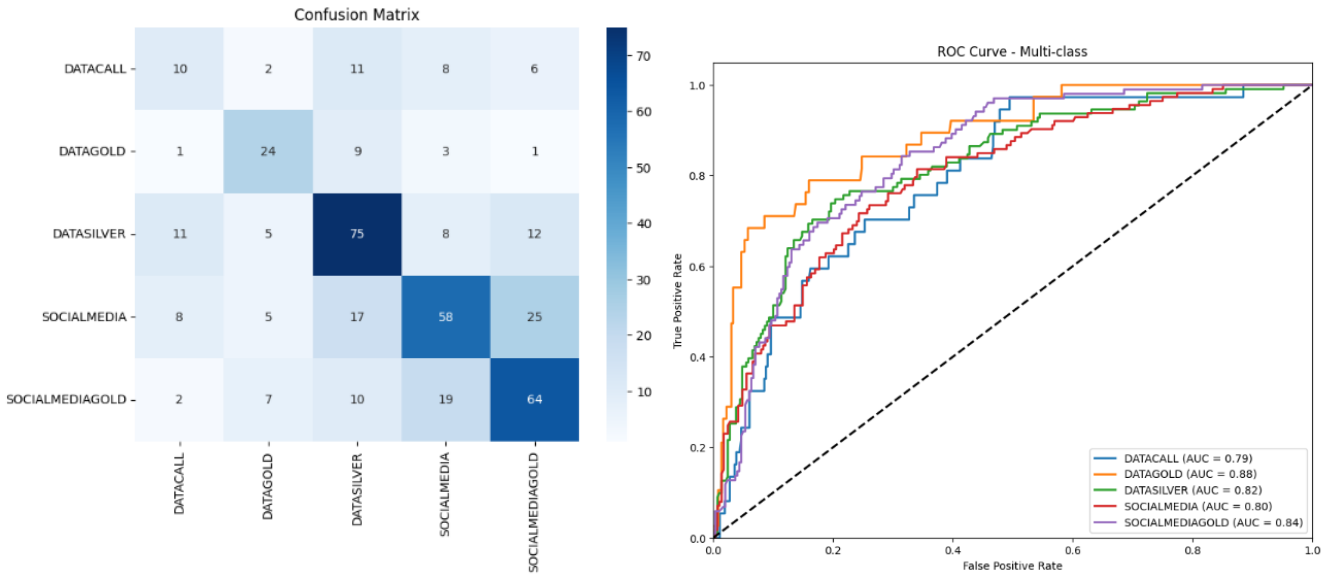

Figure 23. Confusion Matrix & ROC-AUC Curve của mô hình ***Support Vector Machine***

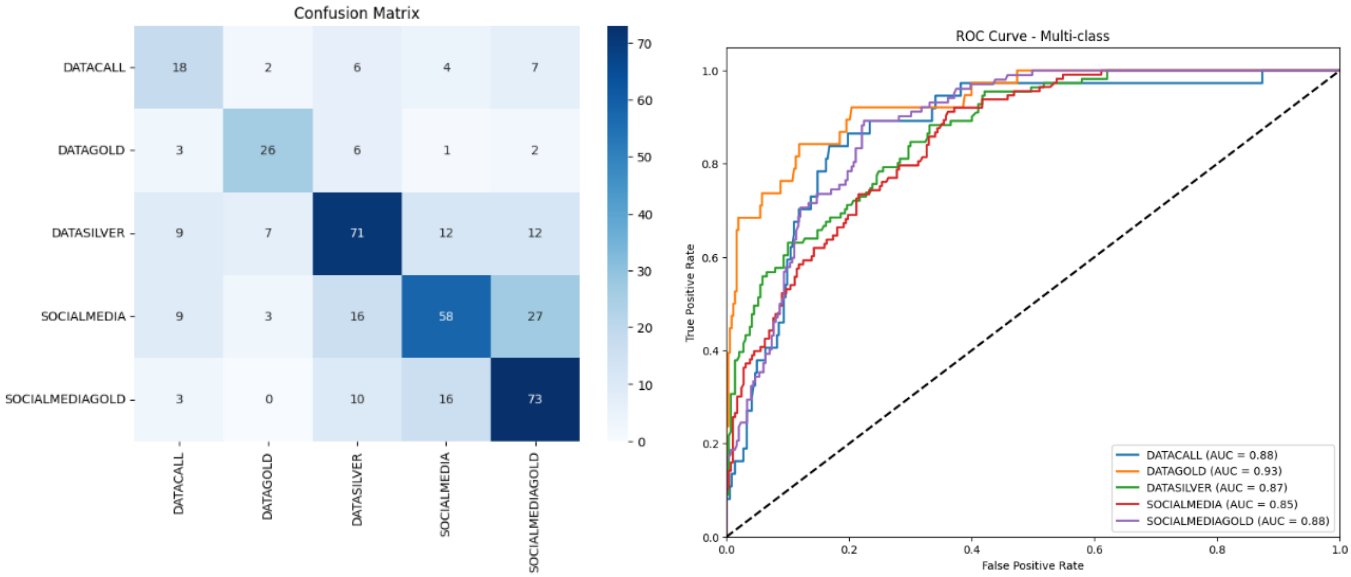

Figure 24. Confusion Matrix & ROC-AUC Curve của mô hình ***Random Forest***

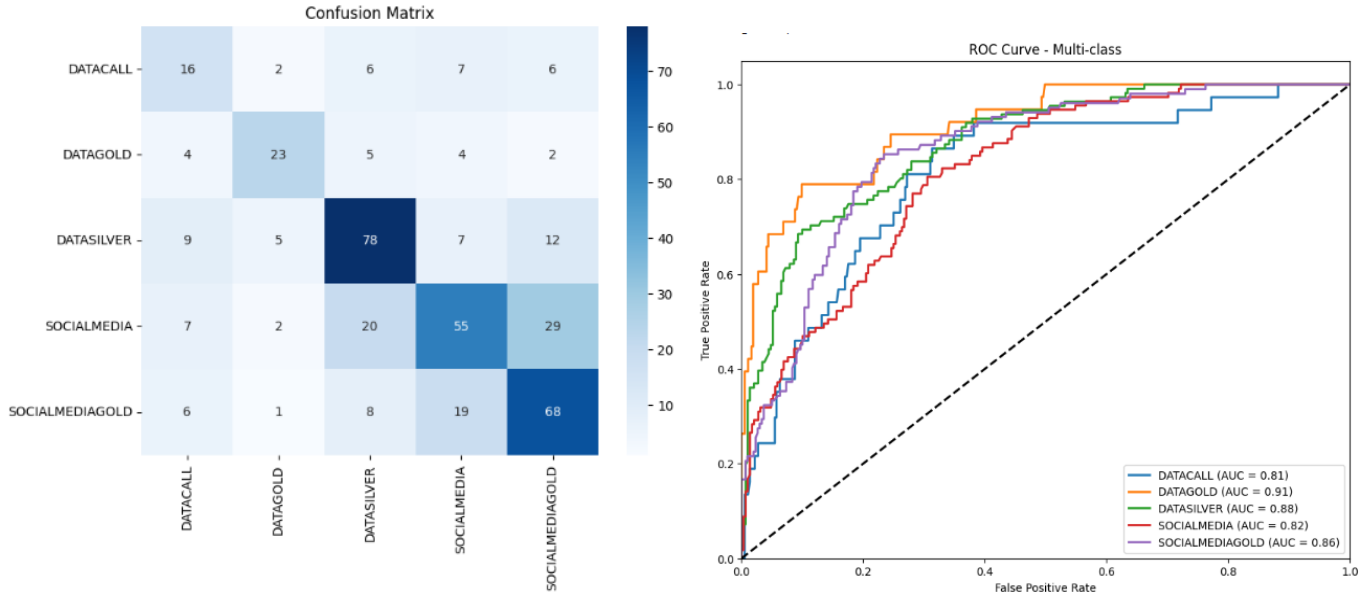

*Figure 25. Confusion Matrix & ROC-AUC Curve của mô hình **LightGBM***

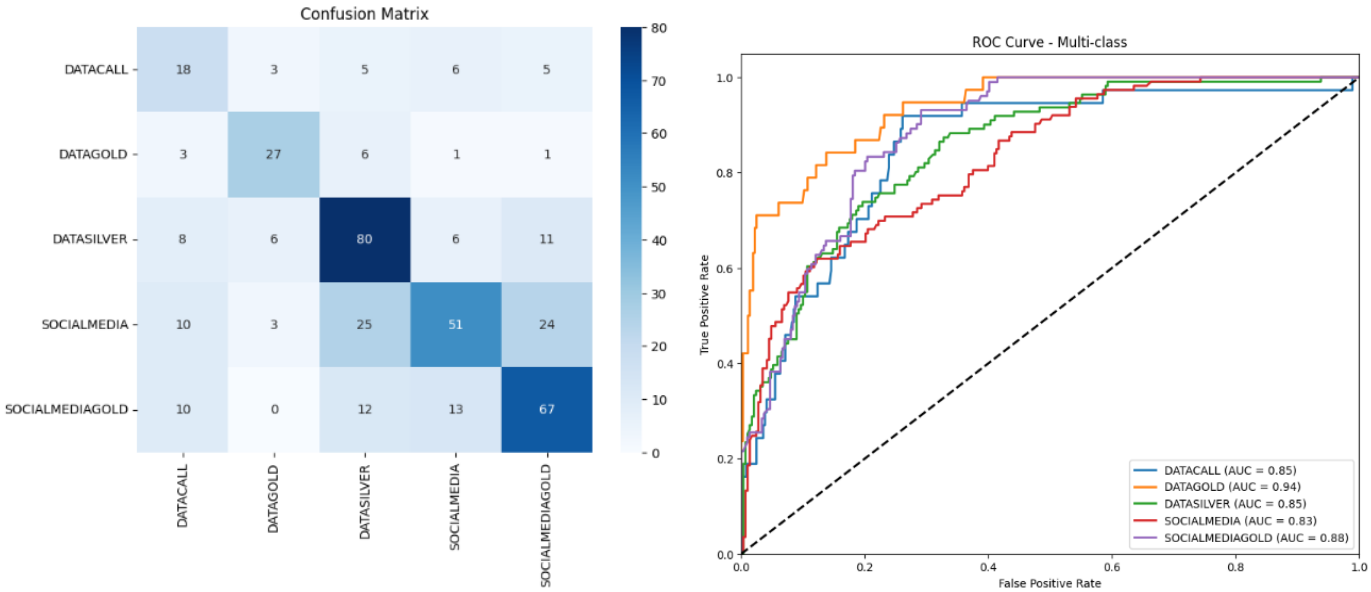

*Figure 26. Confusion Matrix & ROC-AUC Curve của mô hình **XGBoost***

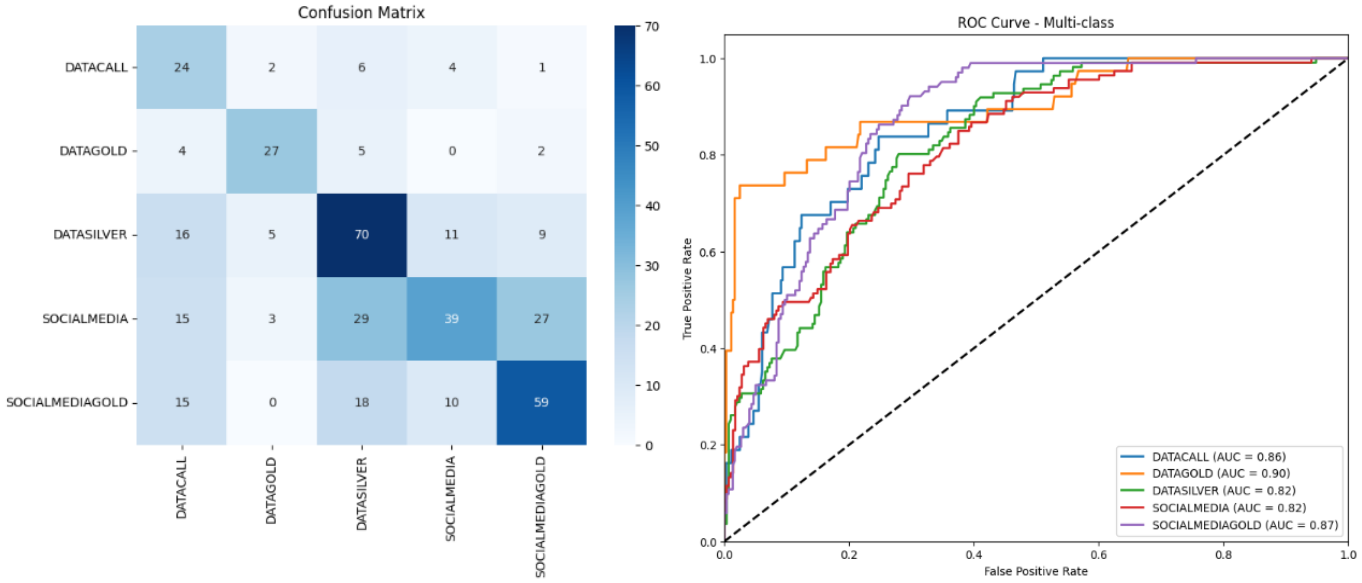

*Figure 27. Confusion Matrix & ROC-AUC Curve của mô hình **CatBoost***

Nhận xét:

- Logistic Regression:

• Ma trận nhầm lẫn cho thấy mô hình gặp khó khăn trong phân loại, nhiều lớp bị nhầm lẫn, đặc biệt **DATACALL** thường bị dự đoán thành **SOCIALMEDIAGOLD.**

• ROC-AUC dao động **0.65–0.74** → khả năng phân biệt lớp **kém**.

- SVM:

• Các giá trị trên đường chéo chính cải thiện so với Logistic Regression, ví dụ DATASILVER dự đoán đúng 75 mẫu.

• Vẫn còn nhầm lẫn chéo giữa các lớp, DATASILVER bị nhầm 12 mẫu sang SOCIALMEDIAGOLD.

• ROC-AUC dao động **0.79–0.88** → phân biệt lớp tốt hơn.

- Random Forest:

• Ma trận nhầm lẫn **"sạch"**, giá trị trên đường chéo chính cao, DATASILVER dự đoán đúng 71 mẫu, SOCIALMEDIAGOLD 73 mẫu.

• ROC-AUC **≥ 0.85**, DATAGOLD đạt **0.93** → khả năng phân biệt lớp **tốt nhất**

- XGBoost:

• Ma trận nhầm lẫn **tốt**, DATASILVER dự đoán đúng 80 mẫu.

• ROC-AUC **cao**, DATAGOLD đạt **0.94** → khả năng phân biệt **mạnh mẽ**.

• Hiệu suất gần **tương đương** Random Forest.

- LightGBM:

• Ma trận nhầm lẫn **tương đối sạch**, DATASILVER dự đoán đúng 78 mẫu.

• ROC-AUC ổn định **0.81–0.91** → phân biệt lớp **đồng đều và tin cậy**.

- CatBoost:

• ROC-AUC **cao**, DATAGOLD đạt 0.90.

• Ma trận nhầm lẫn **kém "sạch"** hơn, lớp DATASILVER nhận nhầm nhiều từ SOCIALMEDIA (29) và DATACALL (16).

→ ***Random Forest*** *và* ***XGBoost*** *là hai mô hình* ***nổi bật nhất*** *và được* ***đề xuất triển khai*** *không chỉ vì đạt Accuracy và F1-macro cao nhất mà còn do khả năng* ***phân biệt lớp rất mạnh mẽ*** *thể hiện qua* ***ROC-AUC*** *và* ***ma trận nhầm lẫn sạch***. ***LightGBM*** *tuy hiệu suất thấp hơn đôi chút nhưng vẫn rất ổn định và là lựa chọn* ***thay thế tin cậy*** *khi cần mô hình* ***nhẹ*** *và dự đoán* ***nhanh***.