ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
ĐẠI HỌC KHOA HỌC XÃ HỘI VÀ NHÂN VĂN
KHOA THƯ VIỆN – THÔNG TIN HỌC

NGHIÊN CỨU VÀ ỨNG DỤNG THUẬT TOÁN HỒI QUY LOGISTIC PHÂN LOẠI KHÁCH HÀNG ĐĂNG KÝ TIỀN GỬI

Giảng viên hướng dẫn: ThS. Nguyễn Tấn Công

Sinh viên thực hiện Huỳnh Phương Vi - 215210151

Phạm Hồng Anh – 2156210090

Nguyễn Thị Diễm Thuy – 2156210141

Hoàng Ngọc Thúy Quỳnh – 2156210063

Tp. Hồ Chí Minh, ngày 23 tháng 1 năm 2024

MỤC LỤC

BẢNG MÔ TẢ CÁC THUẬT NGỮ

STT	Thuật ngữ tiếng Anh	Thuật ngữ tiếng Việt
1	Management Information Systems (MIS)	Hệ thống thông tin quản lý
2	Implement linear equation	Thực hiện phương trình tuyến tính
3	Decision boundary	Ranh giới quyết định
4	Cross-entropy	Hàm mất mát
5	Outlier	Giá trị ngoại lai
6	Accuracy	Độ chính xác
7	Precision	Độ chính xác dự đoán tích cực
8	Recall	Độ chính xác nhận dạng tích cực
9	Decision tree	Cây quyết định
10	Gradient descent	Hàm tối ưu

DANH MỤC CÁC HÌNH

DANH MỤC CÁC BIỂU ĐỒ

DANH MỤC CÁC BẢNG

DANH MỤC CÁC TÀI LIỆU THAM KHẢO

[1] Prashant, A. (2023). Logistic regression classifier tutorial. Retrieved January 23, 2024, from https://www.kaggle.com/code/prashant111/logistic-regression-classifier-tutorial

TÓM TẮT TIỂU LUẬN

Trong bối cảnh cạnh tranh ngày càng gay gắt của thị trường ngân hàng, các ngân hàng cần phải tìm ra cách tiếp cận khách hàng hiệu quả hơn để thu hút và giữ chân khách hàng. Một trong những cách tiếp cận hiệu quả đó là sử dụng Logistic Regression để phân tích dữ liệu khách hàng và xác định các nhóm khách hàng tiềm năng.

Logistic Regression là một kỹ thuật thống kê được sử dụng để dự đoán kết quả nhị phân, chẳng hạn như có hoặc không, đúng hoặc sai, sống hoặc chết. Kỹ thuật này có thể được sử dụng để phân tích dữ liệu khách hàng của ngân hàng, chẳng hạn như thông tin nhân khẩu học, thông tin tài chính, và lịch sử giao dịch, để xác định các nhóm khách hàng có khả năng cao sử dụng các sản phẩm và dịch vụ của ngân hàng. Với mục tiêu là

- Thu hút và giữ chân khách hàng: Bằng cách sử dụng Logistic Regression để phân tích dữ liệu khách hàng và xác định những khách hàng có khả năng mua hàng cao nhất, các ngân hàng có thể tập trung các nỗ lực tiếp thị của mình vào những khách hàng này. Điều này có thể giúp các ngân hàng thu hút và giữ chân khách hàng hiệu quả hơn.
- Tăng doanh số bán hàng: Bằng cách sử dụng Logistic Regression để xác định những khách hàng tiềm năng có khả năng chuyển đổi cao nhất, các ngân hàng có thể tập trung các nỗ lực tiếp thị của mình vào những khách hàng này. Điều này có thể giúp các ngân hàng tăng doanh số bán hàng hiệu quả hơn.
- Tối ưu hóa chi phí tiếp thị: Bằng cách sử dụng Logistic Regression để phân tích hiệu quả của các chiến lược tiếp thị hiện tại, các ngân hàng có thể xác định những thay đổi cần thiết để cải thiện hiệu quả. Điều này có thể giúp các ngân hàng tiết kiệm chi phí tiếp thị.

Mục đích báo cáo: Áp dụng thuật toán và mô hình phân loại **Logistic Regression** dựa trên việc khai thác dữ liệu từ ngân hàng.

Phạm vi báo cáo:

- **Tập trung giới thiệu thuật tổng quan thuật toán Hồi quy phi tuyến Logistic Regression bao gồm: các khái niệm, ưu nhược điểm thuật toán, ứng dụng thuật toán vào cuộc sống, mô tả bài toán và sử dụng thuật toán giải bài toán giải tay.**
- Ứng dụng cho tập dữ liệu lớn: Tập trung vào khai thác dữ liệu từ khách hàng dựa trên bộ dữ liệu có liên quan đến các chiến dịch tiếp thị trực tiếp của một tổ chức ngân hàng. Mục tiêu là phân loại xác suất khách hàng đăng ký khoản tiền gửi có thời hạn hay không (yes/no) dựa trên việc khai thác bộ dữ liệu này và dùng thuật toán Logistic Regression để phân lớp.
- Phương pháp nghiên cứu: Báo cáo sử dụng phương pháp phân tích dữ liệu định lượng và phương pháp học máy (machine learning):
    - Phân tích mô tả (Khám phá dữ liệu, EDA)
    - Thuật toán phân loại xác suất Logistic Regression

Nội dung tiểu luận bao gồm 2 chương:

Chương 1: Tổng quan về thuật toán Logistic

Chương 2: Ứng dụng thuật toán vào dữ liệu lớn.

CHƯƠNG 1: TỔNG QUAN VỀ LOGISTIC REGRESSION

1.1. Khái niệm thuật toán và những khái niệm liên quan:

Giới thiệu về hồi quy Logistic: Khi các nhà khoa học dữ liệu gặp phải một số vấn đề phân loại mới, thuật toán đầu tiên có thể xuất hiện trong đầu họ là **Logistic Regression**. Nó là một thuật toán phân loại học tập có giám sát được sử dụng để dự đoán các quan sát cho một tập hợp các lớp rời rạc. Trên thực tế, nó được sử dụng để phân loại các quan sát thành các loại khác nhau. Do đó, đầu ra của nó là rời rạc trong tự nhiên. Logistic Regression còn được gọi là Logit Regression. Đây là một trong những thuật toán phân loại đơn giản, dễ hiểu và linh hoạt nhất được sử dụng để giải quyết các vấn đề phân loại.

Logistic Regression intuition: Trong thống kê, mô hình hồi quy Logistic là một mô hình thống kê được sử dụng rộng rãi, chủ yếu được sử dụng cho mục đích phân loại. Điều đó có nghĩa là với một tập hợp các quan sát, thuật toán Logistic Regression giúp chúng ta phân loại các quan sát này thành hai hoặc nhiều lớp rời rạc. Vì vậy, biến đích là rời rạc trong tự nhiên. Thuật toán Logistic Regression là một thuật toán thuộc nhóm các thuật toán phân lớp, được sử dụng trong các bài toán đầu ra có thể được biến đổi về một giá trị xác suất.

Thuật toán hồi quy Logistic hoạt động như sau:

- *Thực hiện phương trình tuyến tính* ***(Implement linear equation)***
  + Thuật toán hồi quy logistic hoạt động bằng cách triển khai phương trình tuyến tính với các biến độc lập hoặc biến giải thích để dự đoán giá trị phản hồi. Ví dụ: chúng ta xem xét ví dụ về số giờ học và xác suất vượt qua kỳ thi. Ở đây, số giờ học là biến giải thích và được ký hiệu là x1. Xác suất vượt qua kỳ thi là biến phản hồi hoặc biến mục tiêu và được ký hiệu là z. Nếu chúng ta có một biến giải thích (x1) và một biến phản hồi (z), thì phương trình tuyến tính sẽ được đưa ra về mặt toán học với phương trình sau

$z = \beta 0 + \beta 1 x1$	Ở đây, các hệ số β0 và β1 là các tham số của mô hình.

Bảng 1: Phương trình tuyến tính với một biến giải thích.

  + Nếu có nhiều biến giải thích thì phương trình trên có thể được mở rộng thành

$z = \beta 0 + \beta 1 x1 + \beta 2 x2 + \ldots\ldots + \beta n xn$	Ở đây các hệ số β0, β1, β2 và βn là các tham số của mô hình. Vì vậy, giá trị phản hồi dự đoán (sigmoid) được cho bởi các phương trình trên và được ký hiệu là z.

Bảng 2: Phương trình tuyến tính với hai biến giải thích

- ***Những khái niệm hàm bên trong thuật toán:***

Hàm sigmoid

Giá trị phản hồi dự đoán này, ký hiệu là z sau đó được chuyển đổi thành giá trị xác suất nằm trong khoảng từ 0 đến 1. Sử dụng hàm sigmoid để ánh xạ các giá trị dự đoán thành các giá trị xác suất. Hàm sigmoid này sau đó ánh xạ bất kỳ giá trị thực nào thành giá trị xác suất trong khoảng từ 0 đến 1.

Trong học máy, hàm sigmoid được sử dụng để ánh xạ các dự đoán theo xác suất. Nó còn được gọi là đường cong sigmoid.
Hàm Sigmoid là trường hợp đặc biệt của hàm Logistic. Nó được đưa ra bởi công thức toán học sau đây. Công thức của hàm sigmoid là:

y_{pred} **= σ(z) = 1 / (1 + e^(-z))**
Trong đó: ● z là giá trị đầu vào của hàm sigmoid ● σ(z) là giá trị đầu ra của hàm sigmoid ● e là hằng số toán học e, bằng khoảng 2,718

Bảng 3: Công thức hàm Sigmoid

Chú ý: Hàm sigmoid có các tính chất sau:

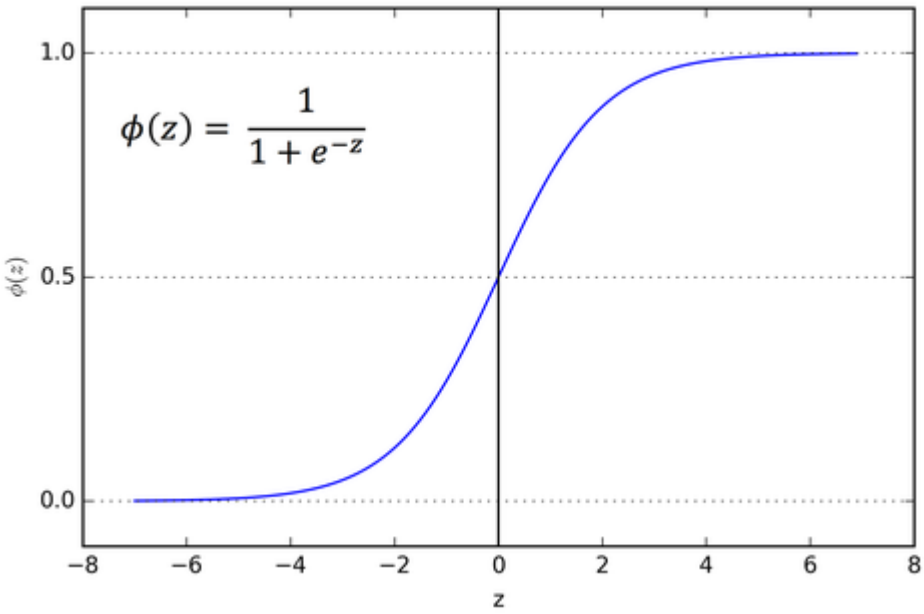
- Hàm sigmoid là một hàm phi tuyến, tức là nó không thể được biểu diễn dưới dạng một hàm tuyến tính.
- Hàm sigmoid có đường cong hình chữ S.
- Hàm sigmoid ánh xạ các giá trị thực bất kỳ thành giá trị xác suất trong khoảng từ 0 đến 1.

Hàm sigmoid thường được sử dụng trong các mô hình học máy để ánh xạ các dự đoán theo xác suất. Ví dụ, trong bài toán phân loại nhị phân, hàm sigmoid có thể được sử dụng để ánh xạ các giá trị dự đoán thành xác suất của một đối tượng thuộc lớp 1 hoặc lớp 2.

Ví dụ, nếu giá trị đầu vào của hàm sigmoid là z = 1, thì giá trị đầu ra của hàm sẽ là σ(1) = 1 / (1 + e^(-1)) = 0,731. Giá trị này có thể được hiểu là xác suất của một đối tượng thuộc lớp 1 là 73,1%.

Hàm sigmoid cũng có thể được sử dụng trong các bài toán hồi quy. Ví dụ, trong bài toán dự đoán giá nhà, hàm sigmoid có thể được sử dụng để ánh xạ các giá trị dự đoán thành xác suất của một căn nhà có giá trị trên một ngưỡng nhất định.

- Về mặt đồ họa, chúng ta có thể biểu diễn hàm sigmoid bằng biểu đồ sau:



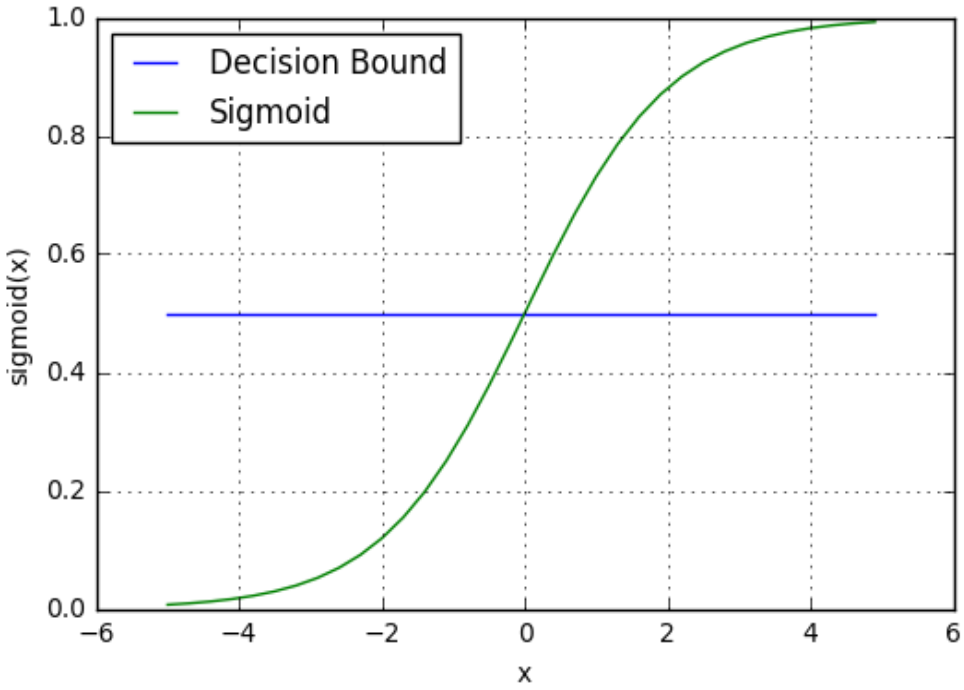
Biểu đồ 1:Biểu diễn hàm Sigmoid

Ranh giới quyết định (Decision boundary):

Hàm sigmoid trả về giá trị xác suất trong khoảng từ 0 đến 1. Giá trị xác suất này sau đó được ánh xạ tới một lớp rời rạc là "0" hoặc "1". Để ánh xạ giá trị xác suất này vào một lớp riêng biệt (đạt/không đạt, có/không, đúng/sai), chúng tôi chọn một giá trị ngưỡng. Giá trị ngưỡng này được gọi là ranh giới Quyết định. Trên giá trị ngưỡng này, chúng tôi sẽ ánh xạ các giá trị xác suất vào lớp 1 và dưới giá trị này chúng tôi sẽ ánh xạ các giá trị vào lớp 0.
Về mặt toán học, nó có thể được biểu diễn như sau:

$p \geq 0,5 \Rightarrow$ lớp = 1
$p < 0,5 \Rightarrow$ lớp = 0

Nói chung, ranh giới quyết định được đặt thành 0,5. Vì vậy, nếu giá trị xác suất là 0,8 (> 0,5), chúng tôi sẽ ánh xạ quan sát này vào lớp 1. Tương tự, nếu giá trị xác suất là 0,2 (< 0,5), chúng tôi sẽ ánh xạ quan sát này vào lớp 0. Điều này được biểu thị trong biểu đồ dưới:



Biểu đồ 2: Biểu diễn hàm Sigmoid qua ranh giới quyết định

Gradient Descent

Gradient Descent là một thuật toán tối ưu được sử dụng để tìm ra tham số tối ưu của mô hình. Trong Logistic Regression, Gradient Descent được sử dụng để tìm ra các trọng số của mô hình sao cho **hàm chi phí** đạt giá trị nhỏ nhất.

Trong thuật toán hồi quy logistic, hàm chi phí được sử dụng để hướng dẫn quá trình học tập của mô hình. Mô hình sẽ được cập nhật theo hướng làm giảm hàm chi phí. Quá trình này được thực hiện bởi thuật toán gradient descent.

Gradient descent là một thuật toán tìm kiếm theo hướng, nghĩa là nó bắt đầu từ một điểm khởi đầu ngẫu nhiên và sau đó thực hiện các bước nhỏ theo hướng giảm hàm chi phí.

Hàm chi phí

Hàm chi phí là hàm dùng để đo lường mức độ sai lệch giữa kết quả dự đoán của mô hình và giá trị thực tế. Trong Logistic Regression, hàm chi phí thường được sử dụng là hàm mất mát cross-entropy. Hàm chi phí càng thấp thì mô hình càng tốt.

Công thức của hàm mất mát cross- entropy trong thuật toán Logistic Regression là:

J = (-1/m) * (y * log(y_pred) + (1-y) * log(1-y_pred))
Trong đó:

- J là hàm chi phí. - y là lớp thực tế của điểm dữ liệu. - y_{pred} là hàm dự đoán của mô hình. - m là số lượng điểm dữ liệu trong tập dữ liệu đào tạo

Bảng 4: Công thức hàm mất mát

Như vậy, hàm chi phí có thể được hiểu là hàm đánh giá mức độ phù hợp của hàm dự đoán y_pred với cả hai lớp của điểm dữ liệu. Hàm chi phí càng thấp thì mô hình càng tốt.

Một số lưu ý khi sử dụng hàm chi phí

- Hàm chi phí này chỉ phù hợp với bài toán phân loại nhị phân.
- Giá trị của hàm chi phí có thể rất lớn, đặc biệt khi số lượng điểm dữ liệu trong tập dữ liệu đào tạo lớn. Do đó, thường người ta sử dụng hàm chi phí được chuẩn hóa sau khi tính toán.

Trong mỗi bước, gradient descent sẽ tính đạo hàm của hàm chi phí theo các trọng số của mô hình. Đạo hàm cho biết hướng mà hàm chi phí giảm nhanh nhất tại điểm hiện tại.

Tham số của mô hình được cập nhật theo hướng giảm hàm chi phí. Input: Hàm chi phí (J), learning rate (α), wi là một trọng số của mô hình, đạo hàm riêng của hàm chi phí theo từng trọng số.

$\mathbf{w_i := w_i - \alpha \frac{\partial J}{\partial wi}}$ **(với α là learning rate)**
$\mathbf{b := b - \alpha \frac{\partial J}{\partial b}}$

Bảng 5: Đạo hàm riêng của hàm chi phí theo từng trọng số

Sau đó, gradient descent sẽ cập nhật các trọng số của mô hình theo hướng ngược với đạo hàm. Điều này sẽ đảm bảo rằng hàm chi phí sẽ giảm trong bước tiếp theo.

Quá trình này sẽ được lặp lại cho đến khi hàm chi phí đạt giá trị nhỏ nhất hoặc cho đến khi đạt đến một ngưỡng nhất định.

Đưa ra dự đoán

Bây giờ, chúng ta đã biết về hàm sigmoid và ranh giới quyết định trong hồi quy logistic. Chúng ta có thể sử dụng kiến thức về hàm sigmoid và ranh giới quyết định để viết hàm dự đoán. Hàm dự đoán trong hồi quy logistic trả về xác suất quan sát là dương, Có hoặc Đúng. Chúng ta gọi đây là lớp 1 và được ký hiệu là P(class = 1). Nếu xác suất tiến gần đến 1 thì chúng ta sẽ tự tin hơn về mô hình của mình rằng quan sát thuộc lớp 1, nếu không thì nó thuộc lớp 0.

1.2. Giả định của thuật toán:

Mô hình hồi quy logistic yêu cầu một số giả định chính. Chúng như sau:

- Mô hình hồi quy logistic yêu cầu biến phụ thuộc phải có bản chất nhị phân, đa thức hoặc thứ tự.
- Nó đòi hỏi các quan sát phải độc lập với nhau. Vì vậy, các quan sát không nên đến từ các phép đo lặp đi lặp lại.
- Thuật toán hồi quy logistic yêu cầu ít hoặc không có hiện tượng đa cộng tuyến giữa các biến độc lập. Điều đó có nghĩa là các biến độc lập không nên có mối tương quan quá cao với nhau.
- Mô hình hồi quy logistic giả định tính tuyến tính của các biến độc lập và tỷ lệ logarit.
- Sự thành công của mô hình hồi quy logistic phụ thuộc vào kích thước mẫu. Thông thường, nó đòi hỏi cỡ mẫu lớn để đạt được độ chính xác cao.

1.3. Ứng dụng của thuật toán:

Thuật toán học máy được sử dụng rộng rãi cho các bài toán phân loại. Ở dạng cơ bản, nó được sử dụng cho bài toán phân loại nhị phân chỉ có hai lớp để dự đoán, nơi mục tiêu là dự đoán xác suất rơi vào một trong hai lớp. Tuy nhiên, thuật toán này cũng có thể được mở rộng để áp dụng cho bài toán phân loại đa lớp.

Các ứng dụng của thuật toán hồi quy logistic có thể được tìm thấy trong nhiều lĩnh vực khác nhau, bao gồm:

- Phân loại thư rác: Thuật toán hồi quy logistic có thể được sử dụng để phân loại email thành thư rác hoặc không phải thư rác.
- Phân loại khách hàng: Thuật toán hồi quy logistic có thể được sử dụng để phân loại khách hàng thành các nhóm khác nhau, chẳng hạn như khách hàng tiềm năng, khách hàng hiện tại hoặc khách hàng đã mất.
- Phân tích tín dụng: Thuật toán hồi quy logistic có thể được sử dụng để xác định khả năng rủi ro của một khách hàng vay tiền.
- Dự đoán biến cố: Thuật toán hồi quy logistic có thể được sử dụng để dự đoán các biến cố trong tương lai, chẳng hạn như khả năng mắc bệnh, khả năng mua hàng hoặc khả năng bỏ học. Phân loại bệnh nhân ung thư/không ung thư. (Chẩn đoán xem một người có nguy cơ bị đột quỵ hay không dựa trên kết quả khám tổng quát của họ).
- Phân loại ảnh người/ảnh vật.

Dưới đây là một số ví dụ cụ thể về ứng dụng của thuật toán hồi quy logistic:

- Google sử dụng thuật toán hồi quy logistic để lọc thư rác.
- Amazon sử dụng thuật toán hồi quy logistic để đề xuất sản phẩm cho khách hàng.
- Netflix sử dụng thuật toán hồi quy logistic để đề xuất phim và chương trình truyền hình cho người dùng.
- Các ngân hàng sử dụng thuật toán hồi quy logistic để xác định khả năng rủi ro của khách hàng vay tiền.

Thuật toán hồi quy logistic là một công cụ mạnh mẽ có thể được sử dụng để giải quyết nhiều loại vấn đề phân loại.

1.4. Ưu – nhược điểm – đặc trưng thuật toán:

Ưu Điểm	Nhược Điểm
- Không đòi hỏi nhiều tài nguyên tính toán. - Đơn giản và dễ hiểu: Thuật toán hồi quy logistic là một thuật toán đơn giản và dễ hiểu, ngay cả đối với những người mới bắt đầu học học máy. - Hiệu quả: Thuật toán hồi quy logistic có thể đạt được hiệu quả cao trong nhiều bài toán phân loại đặc biệt là nhị phân. - Dễ dàng triển khai: Thuật toán hồi quy logistic có thể được triển khai một cách dễ dàng trên nhiều nền tảng khác nhau.	- Hạn chế trong việc xử lý dữ liệu ngoại lai và nhiễu. - Chỉ áp dụng cho bài toán phân loại nhị phân. - Không mạnh mẽ như các mô hình phức tạp hơn. - **Cần dữ liệu cân bằng**: Thuật toán hồi quy logistic hoạt động tốt nhất khi dữ liệu được cân bằng, nghĩa là số lượng dữ liệu cho mỗi lớp là như nhau. - Có thể bị quá khớp: Thuật toán hồi quy logistic có thể bị quá khớp với dữ liệu đào tạo, dẫn đến việc dự đoán kém chính xác trên dữ liệu mới. - Không thể giải thích: Thuật toán hồi quy logistic không thể giải thích được cách thức hoạt động của nó, khiến việc hiểu lý do tại sao nó đưa ra các dự đoán nhất định trở nên khó khăn.
Đặc trưng thuật toán: - Đầu ra của mô hình là một giá trị xác suất. - Sử dụng hàm Sigmoid để chuyển đổi đầu ra của mô hình về một giá trị xác suất. - Sử dụng hàm mất mát cross-entropy để đo lường mức độ sai lệch giữa kết quả dự đoán của mô hình và giá trị thực tế. - Sử dụng thuật toán Gradient Descent để tìm ra tham số tối ưu của mô hình.	

Bảng 6: Ưu nhược điểm và đặc trưng của thuật toán Logistic Regression

1.5. Mô tả thuật toán:

Mô tả thuật toán Logistic Regression có thể được chia thành các bước sau:

Cho trước m bộ dữ liệu để đào tạo thuật toán. Bộ dữ liệu thứ i chứa input x(i) và output y(i) thuộc một trong hai giá trị 0 và 1. Nhiệm vụ của ta là tìm ra vector hệ số $w = (w_1, w_2,..., w_n)$ của ranh giới quyết định Decision Boundary.

B1. Thu thập dữ liệu: Dữ liệu huấn luyện cho Logistic Regression phải có hai cột: cột đầu tiên chứa các giá trị đầu vào, cột thứ hai chứa các giá trị phân loại.

B2. Khởi tạo tham số:

· Input: Dữ liệu huấn luyện gồm các đặc trưng (x_1, x_2,...,x_n) và nhãn (y).

· Output: Các trọng số (w_1, w_2,..., w_n) và bias (b).

⇒Tham số của Logistic Regression là các trọng số của mô hình. Tham số có thể được khởi tạo ngẫu nhiên hoặc sử dụng các kỹ thuật khởi tạo khác. w và b là các trọng số cần được khởi tạo.

B3. Tính toán đầu ra dự đoán: Sử dụng hàm sigmoid, ta tính toán giá trị dự đoán y_{pred} cho từng mẫu dữ liệu

$$y_{pred} = \text{sigmoid}(z) = 1/(1+(e^{-z}))$$

B4. Tính toán hàm chi phí: Hàm chi phí được tính toán dựa trên kết quả dự đoán của mô hình và giá trị thực tế. Sử dụng hàm chi phí Cross-Entropy Loss để đo lường chênh lệch giữa dự đoán và thực tế.

$$J = (-1/m) * (y * log(y_{pred}) + (1-y) * log(1-y_{pred}))$$

B5. Gradient Descent: Tham số của mô hình được cập nhật theo hướng giảm hàm chi phí. Input: Hàm chi phí (J), learning rate (α), đạo hàm riêng của hàm chi phí theo từng trọng số.

$$w_i := w_i - \alpha \frac{\partial J}{\partial wi} \text{ (với α là learning rate)}$$

$$b := b - \alpha \frac{\partial J}{\partial b}$$

B6. Lặp Lại:

· Input: Dữ liệu huấn luyện, số lần lặp hoặc điều kiện dừng.

· Output: Các trọng số và bias đã được tối ưu.

⇒Hành Động: Lặp lại quá trình tính toán tổ hợp, dự đoán, hàm chi phí, và cập nhật tham số cho đến khi đạt được điều kiện dừng hoặc số lần lặp đủ.

B7. Đánh giá mô hình: Đánh giá mô hình bằng cách sử dụng các độ đo như độ chính xác (accuracy)

$$\text{Accuracy} = \frac{\text{Số mẫu dự đoán đúng}}{\text{Tổng số mẫu}}$$

B8. Áp dụng: tính xác suất của mô hình trên. Kết luận y_{pred} thuộc lớp nào dựa trên xác suất.

Ví dụ: Lớp A > 0.5, Lớp B < 0.5

=> y_pred = 0.67 => xác suất thuộc lớp A là 67%.

1.6. Ứng dụng mô tả thuật toán cho dữ liệu giải tay:

Giả sử chúng ta có một tập dữ liệu gồm các mẫu học sinh với số giờ học và số giờ ôn tập đã biết, cùng với kết quả đỗ hay không đỗ của từng học sinh. Sử dụng Logistic Regression để dự đoán xác suất một học sinh đỗ kỳ thi dựa trên số giờ học và số giờ ôn tập.

Số giờ học(x_1)	Số giờ ôn tập (x_2)	Đỗ/Không đỗ(y)
3	1	0
4	2	0
5	3	0
6	4	0
7	5	1
8	6	1
9	7	1
10	8	1
4	3	0
6	4	1
5	2	0
7	5	1
8	6	1
5	1	0
6	4	1

7	3	0
9	2	1
3	4	0
5	3	0
8	6	1

Bảng 7: Dữ liệu ứng dụng cho thuật toán

Bước 1: Khởi tạo các tham số ban đầu:

$w_1 = 0.5$

$w_2 = 0.3$

$b = 0$

Tính Tổ Hợp Tuyến Tính: $z=w_1*x_1+w_2*x_2+b$

$z_1=0.5*3+0.3*1+0=1.8$

$z_2=0.5*4+0.3*2+0=2.6$

$z_3=0.5*5+0.3*3+0=3.4$

$z_4=0.5*6+0.3*4+0=4.2$

$z_5=0.5*7+0.3*5+0=5.0$

$z_6=0.5*8+0.3*6+0=5.8$

$z_7=0.5*9+0.3*7+0=6.6$

$z_8=0.5*10+0.3*8+0=7.4$

$z_9=0.5*4+0.3*3+0=2.9$

$z_{10}=0.5*6+0.3*4+0=4.2$

$z_{11}=0.5*5+0.3*2+0=3.1$

$z_{12}=0.5*7+0.3*5+0=5.0$

$z_{13}=0.5*8+0.3*6+0=5.8$

$z_{14}=0.5*5+0.3*1+0=2.8$

$z_{15}=0.5*6+0.3*4+0=4.2$

$z_{16}=0.5*7+0.3*3+0=5.6$

$z_{17}=0.5*9+0.3*2+0=4.3$

$z_{18}=0.5*3+0.3*4+0=2.2$

$z_{19}=0.5*5+0.3*3+0=3.3$

$z_{20}=0.5*8+0.3*6+0=7.4$

Bước 2: Tính toán đầu ra dự đoán:

Sử dụng hàm sigmoid, ta tính toán giá trị dự đoán y_{pred} cho từng mẫu dữ liệu:

y_{pred} **= sigmoid(z) = 11+(e-z)**

y_{pred1} =**11+(e-1.8)**≈0.8576

y_{pred2} =**11+(e-2.6)**≈0.9309

y_{pred3} =**11+(e-3.4)**≈0.9677

y_{pred4} =**11+(e-4.2)**≈0.9852
y_{pred5} =**11+(e-5.0)**≈0.9933
y_{pred6} =**11+(e-**5.8)≈0.997
y_{pred7} =**11+(e-6.6)**≈0.9986
y_{pred8} =**11+(e-7.4)**≈0.9994
y_{pred9} =**11+(e-2.9)**≈0.9478
y_{pred10} =**11+(e-4.2)**≈0.9851
y_{pred11} =**11+(e-3.1)**≈0.9721
y_{pred12} =**11+(e-5.0)**≈0.9933
y_{pred13} =**11+(e-**5.8)≈0.997
y_{pred14} =**11+(e-2.8)**≈0.944
y_{pred15} =**11+(e-4.2)**≈0.985
y_{pred16} =**11+(e-**5.6)≈0.996
y_{pred17} =**11+(e-4.3)**≈0.987
y_{pred18} =**11+(e-**2.2)≈0.899
y_{pred19} =**11+(e-3.3)**≈0.965
y_{pred20} =**11+(e-7.4)**≈0.999

Bước 3: Tính toán hàm chi phí:

Sử dụng hàm chi phí Cross-Entropy Loss, ta tính toán hàm chi phí J giữa giá trị dự đoán y_{pred} và giá trị thực tế y cho toàn bộ dữ liệu:

$$J = (-1/m) * (y * log(y_{pred}) + (1-y) * log(1-y_{pred}))$$

J_1 =−(0*log(0.8576)+(1−0)*log(1−0.8576))≈0.1572
J_2 =−(0*log(0.9309)+(1−0)*log(1−0.9309))≈0.0452
J_3 =−(0*log(0.9677)+(1−0)*log(1−0.9677))≈0.0133
J_4 =−(0*log(0.9852)+(1−0)*log(1−0.9852))≈0.0035
J_5 =−(1*log(0.9933)+(1−1)*log(1−0.9933))≈0.0009
J_6 =−(1*log(0.997)+(1−1)*log(1−0.997))≈0.0002
J_7 =−(1*log(0.944)+(1−1)*log(1−0.944))≈0.0001
J_8 =−(1*log(0.9994)+(1−1)*log(1−0.9994))≈0.0001
J_9 =−(0*log(0.9478)+(1−0)*log(1−0.9478))≈0.044
J_{10} =−(1*log(0.985)+(1−1)*log(1−0.985))≈0.015
J_{11} =−(0*log(0.972)+(1−0)*log(1−0.972))≈0.028
J_{12} =−(1*log(0.985)+(1−1)*log(1−0.985))≈0.015
J_{13} =−(1*log(0.999)+(1−1)*log(1−0.999))≈0.001
J_{14} =−(0*log(0.944)+(1−0)*log(1−0.944))≈0.056
J_{15} =−(1*log(0.985)+(1−1)*log(1−0.985))≈0.015
J_{16} =−(0*log(0.996)+(1−0)*log(1−0.996))≈0.004
J_{17} =−(1*log(0.987)+(1−1)*log(1−0.987))≈0.013
J_{18} =−(0*log(0.899)+(1−0)*log(1−0.899))≈0.101
J_{19} =−(0*log(0.965)+(1−0)*log(1−0.965))≈0.035
J_{20} =−(1*log(0.999)+(1−1)*log(1−0.999))≈0.001

Bước 4: Sử dụng Gradient Descent để điều chỉnh các tham số:

Sử dụng gradient descent, ta tính toán đạo hàm riêng của hàm chi phí J theo từng tham số w và b, sau đó điều chỉnh các tham số dựa trên giá trị gradient để tiến tới giá trị tối ưu.

$w_i := w_i - \alpha \frac{\partial J}{\partial wi}$ (với α là learning rate)

$b := b - \alpha \frac{\partial J}{\partial b}$

Cho **α = 0.01:**

Đạo hàm mẫu 1:

$\frac{\partial J}{\partial wi1} = x_1 * (y_{pred} - y) = 3 * (0.857 - 0) \approx 2.571$

$\frac{\partial J}{\partial wi2} = x_2 * (y_{pred} - y) = 1 * (0.857 - 0) \approx 0.857$

$\frac{\partial J}{\partial b} = y_{pred} - y = 0.857 - 0 \approx 0.857$

Cập nhật tham số:

$w_1 := w_1 - \alpha \frac{\partial J}{\partial w1} = 0.5 - 0.01 \cdot 2.571 \approx 0.474$

$w_2 := w_2 - \alpha \frac{\partial J}{\partial w2} = 0.3 - 0.01 \cdot 0.857 \approx 0.292$

$b := b - \alpha \frac{\partial J}{\partial b} = 0 - 0.01 * 0.857 \approx -0.00857$

Đạo hàm mẫu 2:

$\frac{\partial J}{\partial wi1} = x_1 * (y_{pred} - y) \approx 3.5232$

$\frac{\partial J}{\partial wi2} = x_2 * (y_{pred} - y) \approx 1.7616$

$\frac{\partial J}{\partial b} = y_{pred} - y \approx 0.9569$

Cập nhật tham số:

$w_1 := w_1 - \alpha \frac{\partial J}{\partial w1} \approx 0.4648$

$w_2 := w_2 - \alpha \frac{\partial J}{\partial w2} \approx 0.2824$

$b := b - \alpha \frac{\partial J}{\partial b} \approx -0.008808$

Đạo hàm mẫu 3:

$\frac{\partial J}{\partial wi1} = x_1 * (y_{pred} - y) \approx 4.5765$

$\frac{\partial J}{\partial wi2} = x_2 * (y_{pred} - y) \approx 2.7459$

$\frac{\partial J}{\partial b} = y_{pred} - y \approx 0.9875$

Cập nhật tham số:

$w_1 := w_1 - \alpha \frac{\partial J}{\partial w1} \approx 0.419075$

$w_2 := w_2 - \alpha \frac{\partial J}{\partial w2} \approx 0.254941$

$b := b - \alpha \frac{\partial J}{\partial b} \approx -0.018958$

Đạo hàm mẫu 4:

$\frac{\partial J}{\partial wi1} = x_1 * (y_{pred} - y) \approx 5.5872$

$\frac{\partial J}{\partial wi2} = x_2 * (y_{pred} - y) \approx 3.7248$

$\frac{\partial J}{\partial b}$=$y_{pred}$-y=0.9965

Cập nhật tham số:

w_1:=w_1−α $\frac{\partial J}{\partial w1}$≈0.363202

w_2:=w_2−α $\frac{\partial J}{\partial w2}$≈0.217693

b:=b−α $\frac{\partial J}{\partial b}$≈−0.02827

Đạo hàm mẫu 5:

$\frac{\partial J}{\partial wi1}$=$x_1$*($y_{pred}$-y)≈−2.93

$\frac{\partial J}{\partial wi2}$=$x_2$*($y_{pred}$-y)≈−2.05

$\frac{\partial J}{\partial b}$=$y_{pred}$-y=−0.01

Cập nhật tham số:

w_1:=w_1−α $\frac{\partial J}{\partial w1}$≈0.5293

w_2:=w_2−α $\frac{\partial J}{\partial w2}$≈0.3205

b:=b−α $\frac{\partial J}{\partial b}$≈0.0001

Đạo hàm mẫu 6:

$\frac{\partial J}{\partial wi1}$=$x_1$*($y_{pred}$-y)≈−1.56

$\frac{\partial J}{\partial wi2}$=$x_2$*($y_{pred}$-y)≈−0.93

$\frac{\partial J}{\partial b}$=$y_{pred}$-y=−0.005

Cập nhật tham số:

w_1:=w_1−α $\frac{\partial J}{\partial w1}$≈0.5156

w_2:=w_2−α $\frac{\partial J}{\partial w2}$≈0.3093

b:=b−α $\frac{\partial J}{\partial b}$≈0.0001

Đạo hàm mẫu 7:

$\frac{\partial J}{\partial wi1}$=$x_1$*($y_{pred}$-y)≈−0.81

$\frac{\partial J}{\partial wi2}$=$x_2$*($y_{pred}$-y)≈−0.56

$\frac{\partial J}{\partial b}$=$y_{pred}$-y=−0.0018

Cập nhật tham số:

w_1:=w_1−α $\frac{\partial J}{\partial w1}$≈0.5081

w_2:=w_2−α $\frac{\partial J}{\partial w2}$≈0.3056

b:=b−α $\frac{\partial J}{\partial b}$≈0.000018

Đạo hàm mẫu 8:

$\frac{\partial J}{\partial wi1}$=$x_1$*($y_{pred}$-y)≈−0.067

$\frac{\partial J}{\partial wi2}$=$x_2$*($y_{pred}$-y)≈−0.0536

$\frac{\partial J}{\partial b}$=$y_{pred}$-y=−0.0067

Cập nhật tham số:

w_1:=w_1−α $\frac{\partial J}{\partial w1}$≈0.509667

w_2:=w_2−α $\frac{\partial J}{\partial w2}$≈0.308267

b:=b−α $\frac{\partial J}{\partial b}$≈0.000067

Đạo hàm mẫu 9:

$\frac{\partial J}{\partial wi1}$=$x_1$*($y_{pred}$-y)≈3.826

$\frac{\partial J}{\partial wi2}$=$x_2$*($y_{pred}$-y)≈2.8695

$\frac{\partial J}{\partial b}$=$y_{pred}$-y=0.9565

Cập nhật tham số:

w_1:=w_1−α $\frac{\partial J}{\partial w1}$≈0.46174

w_2:=w_2−α $\frac{\partial J}{\partial w2}$≈0.271305

b:=b−α $\frac{\partial J}{\partial b}$≈−0.009565

Đạo hàm mẫu 10:

$\frac{\partial J}{\partial wi1}$=$x_1$*($y_{pred}$-y)≈−0.132

$\frac{\partial J}{\partial wi2}$=$x_2$*($y_{pred}$-y)≈−0.088

$\frac{\partial J}{\partial b}$=$y_{pred}$-y=−0.022

Cập nhật tham số:

w_1:=w_1−α $\frac{\partial J}{\partial w1}$≈0.50132

w_2:=w_2−α $\frac{\partial J}{\partial w2}$≈0.30088

b:=b−α $\frac{\partial J}{\partial b}$≈0.00022

Đạo hàm mẫu 11:

$\frac{\partial J}{\partial wi1}$=$x_1$*($y_{pred}$-y)≈4.6205

$\frac{\partial J}{\partial wi2}$=$x_2$*($y_{pred}$-y)≈1.8482

$\frac{\partial J}{\partial b}$=$y_{pred}$-y=0.9241

Cập nhật tham số:

w_1:=w_1−α $\frac{\partial J}{\partial w1}$≈0.453795

w_2:=w_2−α $\frac{\partial J}{\partial w2}$≈0.281518

b:=b−α $\frac{\partial J}{\partial b}$≈−0.009241

Đạo hàm mẫu 12:

$\frac{\partial J}{\partial wi1}$=$x_1$*($y_{pred}$-y)≈−0.126

$\frac{\partial J}{\partial wi2}$=$x_2$*($y_{pred}$-y)≈−0.07

$\frac{\partial J}{\partial b}$=$y_{pred}$-y=−0.018

Cập nhật tham số:

w_1:=w_1−α $\frac{\partial J}{\partial w1}$≈0.50126

w_2:=w_2−α $\frac{\partial J}{\partial w2}$≈0.3007

b:=b−α $\frac{\partial J}{\partial b}$≈0.00018

Đạo hàm mẫu 13:

$\frac{\partial J}{\partial wi1}$=$x_1$*($y_{pred}$-y)≈7.864

$\frac{\partial J}{\partial wi2}$=$x_2$*($y_{pred}$-y)≈0.983

$\frac{\partial J}{\partial b}$=$y_{pred}$-y=0.983

Cập nhật tham số:

w_1:=w_1−α $\frac{\partial J}{\partial w1}$≈0.42136

w_2:=w_2−α $\frac{\partial J}{\partial w2}$≈0.29017

b:=b−α $\frac{\partial J}{\partial b}$≈−0.00983

Đạo hàm mẫu 14:

$\frac{\partial J}{\partial wi1}$=$x_1$*($y_{pred}$-y)≈−0.045

$\frac{\partial J}{\partial wi2}$=$x_2$*($y_{pred}$-y)≈−0.054

$\frac{\partial J}{\partial b}$=$y_{pred}$-y=−0.009

Cập nhật tham số:

w_1:=w_1−α $\frac{\partial J}{\partial w1}$≈0.50045

w_2:=w_2−α $\frac{\partial J}{\partial w2}$≈0.30054

b:=b−α $\frac{\partial J}{\partial b}$≈0.00009

Đạo hàm mẫu 15:

$\frac{\partial J}{\partial wi1}$=$x_1$*($y_{pred}$-y)≈5.784

$\frac{\partial J}{\partial wi2}$=$x_2$*($y_{pred}$-y)≈0.964

$\frac{\partial J}{\partial b}$=$y_{pred}$-y=0.964

Cập nhật tham số:

w_1:=w_1−α $\frac{\partial J}{\partial w1}$≈0.44116

w_2:=w_2−α $\frac{\partial J}{\partial w2}$≈0.29036

b:=b−α $\frac{\partial J}{\partial b}$≈−0.00964

Đạo hàm mẫu 16:

$\frac{\partial J}{\partial wi1}$=$x_1$*($y_{pred}$-y)≈−0.07

$\frac{\partial J}{\partial wi2}$=$x_2$*($y_{pred}$-y)≈−0.04

$\frac{\partial J}{\partial b}$=$y_{pred}$-y=−0.01

Cập nhật tham số:

w_1:=w_1−α $\frac{\partial J}{\partial w1}$≈0.5007

w_2:=w_2−α $\frac{\partial J}{\partial w2}$≈0.3004

b:=b−α $\frac{\partial J}{\partial b}$≈0.0001

Đạo hàm mẫu 17:

$\frac{\partial J}{\partial wi1}$=$x_1$*($y_{pred}$-y)≈8.955

$\frac{\partial J}{\partial wi2}$=$x_2$*($y_{pred}$-y)≈2.985

$\frac{\partial J}{\partial b}$=$y_{pred}$-y=0.995

Cập nhật tham số:

w_1:=w_1−α $\frac{\partial J}{\partial w1}$≈0.41045

w_2:=w_2−α $\frac{\partial J}{\partial w2}$≈0.26915

b:=b−α $\frac{\partial J}{\partial b}$≈−0.00995

Đạo hàm mẫu 18:

$\frac{\partial J}{\partial wi1}$=$x_1$*($y_{pred}$-y)≈−0.18

$\frac{\partial J}{\partial wi2}$=$x_2$*($y_{pred}$-y)≈−0.216

$\frac{\partial J}{\partial b}$=$y_{pred}$-y=−0.036

Cập nhật tham số:

w_1:=w_1−α $\frac{\partial J}{\partial w1}$≈0.5018

w_2:=w_2−α $\frac{\partial J}{\partial w2}$≈0.30216

b:=b−α $\frac{\partial J}{\partial b}$≈0.00036

Đạo hàm mẫu 19:

$\frac{\partial J}{\partial wi1}$=$x_1$*($y_{pred}$-y)≈7.936

$\frac{\partial J}{\partial wi2}$=$x_2$*($y_{pred}$-y)≈4.96

$\frac{\partial J}{\partial b}$=$y_{pred}$-y=0.992

Cập nhật tham số:

w_1:=w_1−α $\frac{\partial J}{\partial w1}$≈0.42064

w_2:=w_2−α $\frac{\partial J}{\partial w2}$≈0.2504

b:=b−α $\frac{\partial J}{\partial b}$≈−0.00992

Đạo hàm mẫu 20:

$\frac{\partial J}{\partial wi1}$=$x_1$*($y_{pred}$-y)≈−0.04

$\frac{\partial J}{\partial wi2}$=$x_2$*($y_{pred}$-y)≈−0.064

$\frac{\partial J}{\partial b}$=$y_{pred}$-y=−0.008

Cập nhật tham số:

w_1:=w_1−α $\frac{\partial J}{\partial w1}$≈0.5004

w_2:=w_2−α $\frac{\partial J}{\partial w2}$≈0.30064

b:=b−α $\frac{\partial J}{\partial b}$≈0.00008

Bước 5: Đánh giá mô hình

Dự đoán nhãn dựa trên ngưỡng 0.5:

y_{pred} = [0,0,0,0,1,1,1,1,0,1,0,1,0,0,1,0,1,0,0,1]

So sánh với nhãn thực tế:

y = [0,0,0,0,1,1,1,1,0,1,0,1,1,0,1,0,1,0,0,1]

Accuracy = $\frac{Số\ mẫu\ dự\ đoán\ đúng}{Tổng\ số\ mẫu} = \frac{19}{20}$

Vậy, Accuracy của mô hình trên tập dữ liệu kiểm thử là khoảng 95%

- **KẾT LUẬN:**

Với tập dữ liệu được cung cấp, có thể thấy rằng số giờ học và số giờ ôn tập có ảnh hưởng đến kết quả đỗ/trượt của học sinh. Cụ thể, học sinh càng học nhiều và ôn tập nhiều thì khả năng đỗ kỳ thi càng cao.

y_{pred1} ≈0.8576 Có xác suất khoảng 86% là thuộc lớp 1 (đỗ).
y_{pred2} ≈0.9309 Có xác suất khoảng 93% là thuộc lớp 1 (đỗ).
y_{pred3} ≈0.9677 Có xác suất khoảng 97% là thuộc lớp 1 (đỗ).
y_{pred4} ≈0.9852 Có xác suất khoảng 99% là thuộc lớp 1 (đỗ).
y_{pred5} ≈0.9933 Có xác suất khoảng 99% là thuộc lớp 1 (đỗ).
y_{pred6} ≈0.997 Có xác suất khoảng 100% là thuộc lớp 1 (đỗ).
y_{pred7} ≈0.9986 Có xác suất khoảng 100% là thuộc lớp 1 (đỗ).
y_{pred8} ≈0.9994 Có xác suất khoảng 100% là thuộc lớp 1 (đỗ).
y_{pred9} ≈0.9478 Có xác suất khoảng 95% là thuộc lớp 1 (đỗ).
y_{pred10} ≈0.9851 Có xác suất khoảng 99% là thuộc lớp 1 (đỗ).
y_{pred11} ≈0.9721 Có xác suất khoảng 97% là thuộc lớp 1 (đỗ).
y_{pred12} ≈0.9933 Có xác suất khoảng 100% là thuộc lớp 1 (đỗ).
y_{pred13} ≈0.997 Có xác suất khoảng 100% là thuộc lớp 1 (đỗ).
y_{pred14} ≈0.944 Có xác suất khoảng 95% là thuộc lớp 1 (đỗ).
y_{pred15} ≈0.985 Có xác suất khoảng 99% là thuộc lớp 1 (đỗ).
y_{pred16} ≈0.996 Có xác suất khoảng 100% là thuộc lớp 1 (đỗ).
y_{pred17} ≈0.987 Có xác suất khoảng 99% là thuộc lớp 1 (đỗ).
y_{pred18} ≈0.899 Có xác suất khoảng 90% là thuộc lớp 1 (đỗ).
y_{pred19} ≈0.965 Có xác suất khoảng 97% là thuộc lớp 1 (đỗ).
y_{pred20} ≈0.999 Có xác suất khoảng 100% là thuộc lớp 1 (đỗ).

CHƯƠNG 2: ỨNG DỤNG THUẬT TOÁN LOGISTIC PHÂN LOẠI LIỆU KHÁCH HÀNG SẼ ĐĂNG KÝ KHOẢN TIỀN GỬI CÓ KÌ HẠN HAY KHÔNG

2.1. DỮ LIỆU:

2.1.1. Giới thiệu về bộ dữ liệu và công cụ thực hiện khai thác:

- Tập dữ liệu chứa thông tin về tình trạng của khách hàng và các chiến dịch tiếp thị trực tiếp của một tổ chức ngân hàng Bồ Đào Nha. Mục tiêu phân loại là dự đoán khách hàng sẽ đăng ký một khoản tiền gửi có kỳ hạn.
- Công cụ sử dụng: Google Colab
- Ngôn ngữ sử dụng để phân tích: Python.
- Dataset sử dụng: bank-additional-full.csv
- Có 41188 điểm dữ liệu và 21 biến trong tập dữ liệu.

2.1.2. Xác định biến (ý nghĩa biến) và phân loại biến:

1 - age: độ tuổi (số)

2 - job : nghề nghiệp (phân loại: quản trị viên, công nhân, kinh doanh, giúp việc, quản lý, về hưu, tự làm chủ, dịch vụ, học sinh, kỹ thuật , thất nghiệp)

3 - marital : tình trạng hôn nhân (phân loại: đã ly hôn, kết hôn, độc thân)

4 - education: trình độ bậc giáo dục (phân loại: cơ bản, THPT, thất học, chuyên nghiệp, đại học)

5 - default: có vỡ nợ hay không? (phân loại: có, không)

6 - housing: có khoản vay nhà ở hay không? (phân loại: có, không)

7 - loan: có khoản vay cá nhân hay không? (phân loại: có, không)

Liên hệ cuối cùng của chiến dịch hiện tại:

8 - contact: loại liên lạc (phân loại: di dộng, điện thoại)

9 - month: tháng liên lạc cuối cùng trong năm (phân loại: 1, 2, 3,...)

10 - day_of_week: ngày liên lạc cuối cùng trong tuần (thứ 2,3,4,5,6)

11 - duration: thời hạn liên lạc cuối cùng, tính bằng giây (số). Lưu ý quan trọng: thuộc tính này ảnh hưởng cao đến mục tiêu đầu ra (VD: nếu thời lượng = 0 thì y = 'không'). Tuy nhiên, thời lượng không được biết đến trước khi một cuộc gọi được thực hiện. Ngoài ra, sau khi kết thúc cuộc gọi 'y' rõ ràng được biết đến. Ngoài ra, đầu vào này chỉ nên được đưa vào cho các mục đích điểm chuẩn và nên được loại bỏ nếu ý định là có mô hình dự đoán thực tế.

Các thuộc tính khác:

12 - campaign: số lượng liên hệ được thực hiện trong chiến dịch này và cho máy của khách (số, gồm liên hệ cuối cùng)

13 - pdays: số ngày trôi qua sau khi khách hàng được liên hệ lần cuối từ 1 chiến dịch trước đó (số, 999 nghĩa là máy của khách không được liên hệ trước đó)

14 - previous: số lượng liên hệ được thực hiên trước chiến dịch và cho máy của khách (số)

15 - poutcome: kết quả của chiến dịch tiếp thị trước đó (phân loại: thất bại, thành công, không tham gia)

Các thuộc tính bối cảnh xã hội và kinh tế:

16 - emp.var.rate: tỷ lệ biến đổi việc làm - hàng quý (số)

17 - cons.price.idx: chỉ số giá tiêu dùng - hàng tháng (số)

18 - cons.conf.idx: chỉ số niềm tin của người tiêu dùng - hàng tháng (số)

19 - euribor3m: tỷ lệ 3 tháng lãi suất tham chiếu euro liên ngân hàng - hàng (số)

20 - nr.employed: số lượng nhân viên - hàng quý (số)

Biến đầu ra (mục tiêu mong muốn):

21 - y - Khách hàng đã đăng ký một khoản tiền gửi có kỳ hạn chưa? (phân loại: có, không)

Bảng 8: Biến và phân loại biến

2.2. KHÁM PHÁ VÀ LÀM SẠCH DỮ LIỆU:

Thực hiện các bước theo mô hình phân tích dữ liệu:

2.2.1. Import thư viện liên quan:

- Đây là bước đầu tiên trong trước khi đi vào khai phá dữ liệu.

```
import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib as mpl

import matplotlib.pyplot as plt

import statsmodels.api as sm

import plotly
```

```
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from plotly.offline import iplot
```

Bảng 9: Import các thư viện liên quan

→ Tổng hợp những thư viện dùng trong quá trình phân tích:

- *NumPy* là thư viện mã nguồn mở cung cấp các đối tượng mảng đa chiều và các hàm để xử lý các đối tượng mảng.
- *Pandas* là thư viện mã nguồn mở cung cấp các đối tượng khung dữ liệu và các hàm để xử lý các dữ liệu bảng.
- *Seaborn* là thư viện mã nguồn mở cung cấp các giao diện đồ họa người dùng (GUI) cho các thư viện trực quan hóa dữ liệu khác, chẳng hạn như Matplotlib. Seaborn thường được sử dụng để tạo các biểu đồ và đồ thị dữ liệu trực quan và đẹp mắt.
- *Matplotlib* là thư viện mã nguồn mở cung cấp các hàm để tạo các biểu đồ và đồ thị dữ liệu.
- *Statsmodels* là thư viện mã nguồn mở cung cấp các hàm để phân tích thống kê dữ liệu.
- *Plotly* là thư viện mã nguồn mở cung cấp các hàm để tạo các biểu đồ và đồ thị dữ liệu tương tác.
- *Px* là thư viện con của Plotly cung cấp các hàm để tạo các biểu đồ và đồ thị dữ liệu theo phong cách Excel.
- *Ff* là thư viện con của Plotly cung cấp các hàm để tạo các biểu đồ và đồ thị dữ liệu tùy chỉnh.
- *Go* là thư viện con của Plotly cung cấp các hàm để tạo các biểu đồ và đồ thị dữ liệu từ các đối tượng JSON.
- Thư viện *from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error* cung cấp các hàm để đánh giá hiệu suất của các mô hình học máy.
- Thư viện *from sklearn.model_selection import train_test_split* cung cấp các hàm để chia dữ liệu thành dữ liệu đào tạo và dữ liệu kiểm tra.
- Thư viện *from sklearn.linear_model import LinearRegression* cung cấp một triển khai của mô hình hồi quy tuyến tính.
- Thư viện *from plotly.offline import iplot* cung cấp các hàm để hiển thị các biểu đồ và đồ thị dữ liệu trong trình duyệt web.

2.2.2. Load dữ liệu (file csv) vào Google Colab:

```
df = pd.read_csv('bank-additional-full.csv', sep=';')
df.head()
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no

5 rows × 21 columns

2.2.3. EDA – Khám phá dữ liệu:

EDA – Phân tích dữ liệu thăm dò là quá trình mô tả dữ liệu bằng các kỹ thuật thống kê và trực quan hoá nhằm tập trung vào các khía cạnh quan trọng của dữ liệu để tiếp tục phân tích. Điều này bao gồm cả việc kiểm tra tập dữ liệu từ nhiều góc độ, mô tả và tóm tắt nó mà không đưa ra bất kỳ giả định nào khác về nội dung của nó. EDA là một bước quan trọng cần phải thực hiện trước khi đi sâu vào mô hình thống kê hoặc học máy.

- ❖ **Thông tin dữ liệu (cột, hàng, kiểu dữ liệu):**

```
df.columns
```

```
Index(['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
       'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'emp.var.rate', 'cons.price.idx',
       'cons.conf.idx', 'euribor3m', 'nr.employed', 'y'],
      dtype='object')
```

```
list(df)
```

```
['age',
 'job',
 'marital',
 'education',
 'default',
 'housing',
 'loan',
 'contact',
 'month',
 'day_of_week',
 'duration',
 'campaign',
 'pdays',
 'previous',
 'poutcome',
 'emp.var.rate',
 'cons.price.idx',
 'cons.conf.idx',
 'euribor3m',
 'nr.employed',
 'y']
```

```
df.shape
```

```
(41188, 21)
```

Hình 1:Thông tin của dữ liệu cột, hàng.

--> **Kết quả:** Tập dữ liệu chứa 21 cột và 41188 dòng dữ liệu.

```
df.info(verbose = True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41188 non-null  int64
 1   job             41188 non-null  object
 2   marital         41188 non-null  object
 3   education       41188 non-null  object
 4   default         41188 non-null  object
 5   housing         41188 non-null  object
 6   loan            41188 non-null  object
 7   contact         41188 non-null  object
 8   month           41188 non-null  object
 9   day_of_week     41188 non-null  object
 10  duration        41188 non-null  int64
 11  campaign        41188 non-null  int64
 12  pdays           41188 non-null  int64
 13  previous        41188 non-null  int64
 14  poutcome        41188 non-null  object
 15  emp.var.rate    41188 non-null  float64
 16  cons.price.idx  41188 non-null  float64
 17  cons.conf.idx   41188 non-null  float64
 18  euribor3m       41188 non-null  float64
 19  nr.employed     41188 non-null  float64
 20  y               41188 non-null  object
dtypes: float64(5), int64(5), object(11)
memory usage: 6.6+ MB
```

Hình 2:Thông tin dữ liệu về kiểu dữ liệu

--> ***Kết quả:*** *Tập dữ liệu chứa 11 biến phân loại và 10 biến số.*

Note: Những biến phân loại sẽ được mã hóa số trước khi dựng mô hình phân loại.

- ❖ **Hiển thị từng giá trị và đếm số lượng của mỗi giá trị của mỗi cột**

```
for col in list(df.columns):

print("\nfor column : ", col,"\n")

print(df[col].value_counts())
```

ex:

```
for column :  job

admin.           6033
blue-collar      4507
technician       4104
services         2164
management       1682
entrepreneur      845
self-employed     773
housemaid         491
unemployed        451
retired           447
student           211
Name: job, dtype: int64

for column :  marital

married     12754
single       6372
divorced     2582
Name: marital, dtype: int64

for column :  education

university.degree      7075
high.school            5470
basic.9y               3323
professional.course    3135
basic.4y               1601
basic.6y               1097
illiterate                7
Name: education, dtype: int64

for column :  default

no     21705
yes        3
Name: default, dtype: int64

for column :  housing

yes    11665
no     10043
Name: housing, dtype: int64
```

Hình 3: Thông tin dữ liệu

❖ **Phân tích mọi điểm bất thường trong dữ liệu:**

- **Mô tả dữ liệu:**

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
age	41188.0	40.024060	10.421250	17.000	32.000	38.000	47.000	98.000
duration	41188.0	258.285010	259.279249	0.000	102.000	180.000	319.000	4918.000
campaign	41188.0	2.567593	2.770014	1.000	1.000	2.000	3.000	56.000
pdays	41188.0	962.475454	186.910907	0.000	999.000	999.000	999.000	999.000
previous	41188.0	0.172963	0.494901	0.000	0.000	0.000	0.000	7.000
emp.var.rate	41188.0	0.081886	1.570960	-3.400	-1.800	1.100	1.400	1.400
cons.price.idx	41188.0	93.575664	0.578840	92.201	93.075	93.749	93.994	94.767
cons.conf.idx	41188.0	-40.502600	4.628198	-50.800	-42.700	-41.800	-36.400	-26.900
euribor3m	41188.0	3.621291	1.734447	0.634	1.344	4.857	4.961	5.045
nr.employed	41188.0	5167.035911	72.251528	4963.600	5099.100	5191.000	5228.100	5228.100

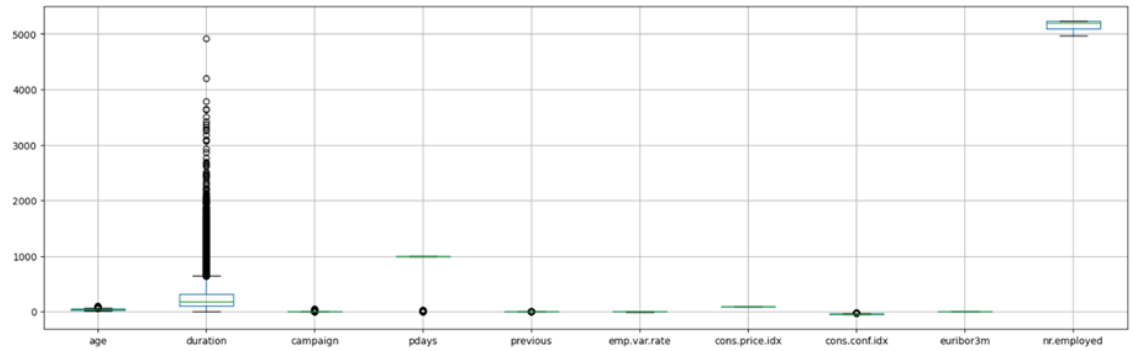
Hình 4: Mô tả dữ liệu

--> **Nhận xét:**

- Hầu hết các cột trong bộ dữ liệu đều lệch phải. Độ lệch chuẩn lớn
- Giá trị ngoại lệ ở các cột duration, age lớn. Cần xử lý trước khi đưa vào mô hình để tránh gây sai lệch kết quả.

❖ Kiểm tra giá trị ngoại lai (Outlier) trong các cột của bộ dữ liệu

```
plt.figure(figsize=(20,6))
df.boxplot()
print()
```



Hình 5: Giá trị outline trong các cột của bộ dữ liệu

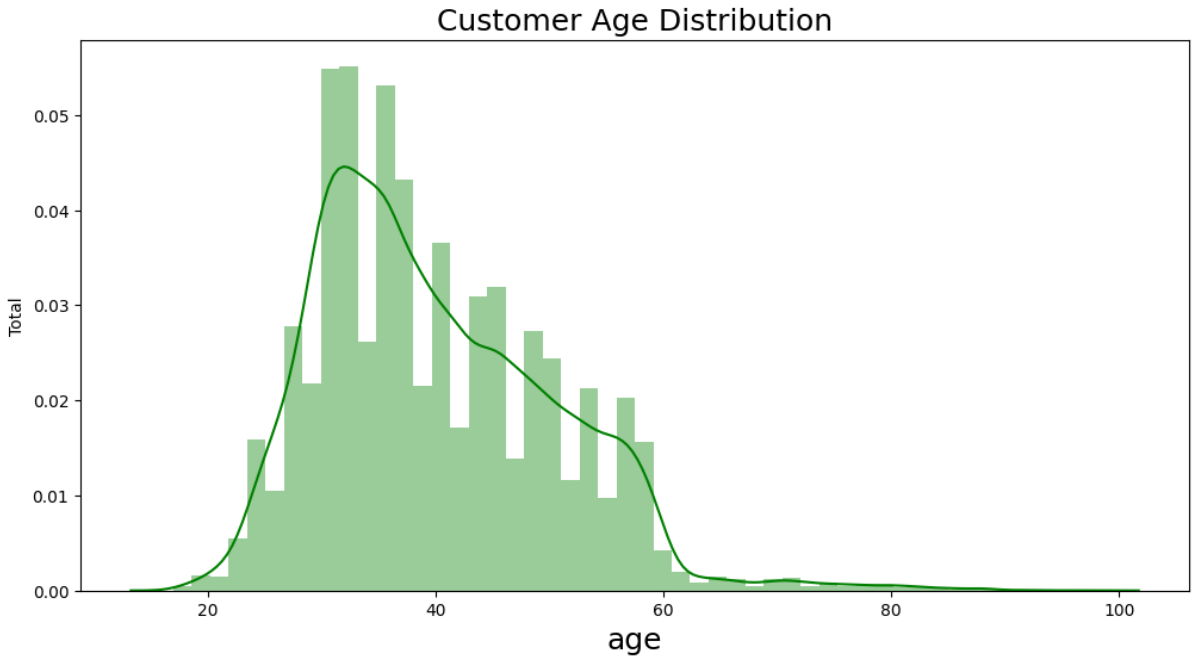
--> **Nhận xét:** Xuất hiện outlier ở những cột age, duration. Cần xử lý chúng.

❖ **Xét phân phối của dữ liệu:**

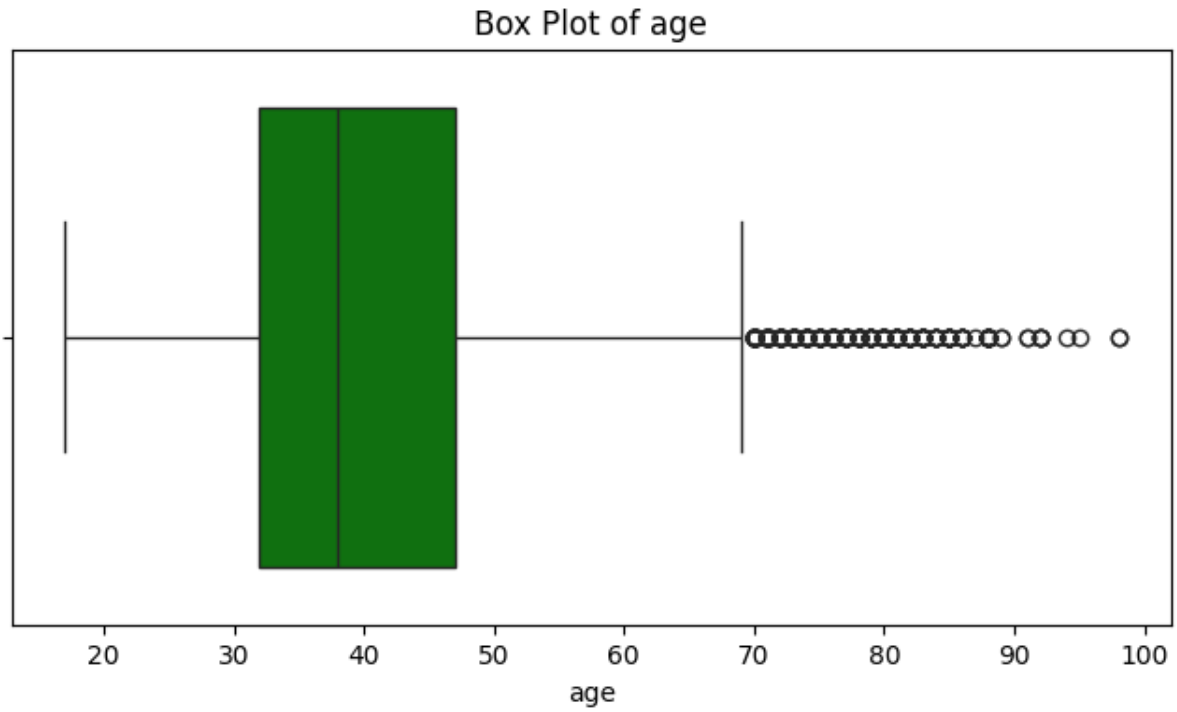
```
num_variables = df.select_dtypes(include=['float64',
'int64']).columns

for col in num_variables:

  plt.figure(figsize=(12,6))

  sns.distplot(df[col], color = 'green')

  plt.title('Phân phối', fontsize = 18)

  plt.xlabel(col, fontsize = 18)

  plt.ylabel('Tổng')

  plt.show()



  plt.figure(figsize=(8, 4))

  sns.boxplot(x=df[col], color = 'green')

  plt.title('Box Plot')

  plt.xlabel(col)
```

```
plt.show()
```

- **Ví dụ phân phối của cột ‘age’ và ‘duration’**



Biểu đồ 3: Biểu đồ phân phối độ tuổi của khách hàng



Hình 6: Phân phối của cột age và duration

→ ***Nhận xét:*** *Hầu hết hình dáng của tất cả các cột đều lệch phải và trung bình tập trung ở số thấp.*

- ❖ **Kiểm tra sự trùng lập dữ liệu (duplicated values):**

```
print('Tổng số dòng: ',df.shape[0])

print('Tổng số cột: ',df.shape[1])

dupplicate_row = len(df)-len(df.drop_duplicates())

print('Số dòng bị trùng lặp: ',dupplicate_row)
```

→ Kết quả: Tổng số dòng: 41188; Tổng số cột: 21; Số dòng bị trùng lặp: 12

- ❖ **Kiểm tra giá trị khuyết trong tập dữ liệu (missing values):**

Trong tập dữ liệu này, có các điểm dữ liệu có giá trị 'không xác định' **(unknown).** Chúng tôi sẽ trao đổi những giá trị này với các giá trị NaN và loại bỏ chúng:

```
df.replace('unknown', np.NaN, inplace=True)
```

```
df.isna().sum()
```

```
age                0
job              330
marital           80
education       1731
default         8597
housing          990
loan             990
contact            0
month              0
day_of_week        0
duration           0
campaign           0
pdays              0
previous           0
poutcome           0
emp.var.rate       0
cons.price.idx     0
cons.conf.idx      0
euribor3m          0
nr.employed        0
y                  0
dtype: int64
```

Hình 7: Mô tả dữ liệu

--> **Nhận xét:** Có giá trị khuyết ở những cột job, marital, education, default, housing, loan

- ❖ **Giá trị duy nhất của tất cả các biến phân loại:**

```
df_obj = df.select_dtypes('object')

df_uniques= pd.DataFrame([[i,len(df_obj[i].unique())] for i in df_obj.columns],
```

```
columns=['Variable','Unique Values']).set_index('Variable')

df_uniques
```

Variable	Unique Values
job	12
marital	4
education	8
default	3
housing	3
loan	3
contact	2
month	10
day_of_week	5
poutcome	3
y	2

Hình 8: Mô tả dữ liệu

--> Những biến phân loại sẽ được tiền xử lý bằng kĩ thuật mã hóa Dummy Encoding

2.2.4. Preprocessing – tiền xử lý dữ liệu:

- **Xóa các hàng có giá trị NaN:**

```
df.dropna(how='any', inplace=True)
df.isna().sum()
```

```
age             0
job             0
marital         0
education       0
default         0
housing         0
loan            0
contact         0
month           0
day_of_week     0
duration        0
campaign        0
pdays           0
previous        0
poutcome        0
emp.var.rate    0
cons.price.idx  0
cons.conf.idx   0
euribor3m       0
nr.employed     0
y               0
dtype: int64
```

Hình 9: Mô tả dữ liệu sau khi xóa giá trị NaN

- **Loại bỏ các bản ghi trùng lặp:**

```
df.drop_duplicates(keep=False, inplace=True)
dupplicate_row = len(df)-len(df.drop_duplicates())
print('Số dòng bị trùng lặp: ',dupplicate_row)
```

```
Số dòng bị trùng lặp:  0
```

Hình 10: Mô tả dữ liệu loại bỏ các gái trị trùng lặp

❖ **Xử lý outlier:**

```
continuous_column_list = ['age']

for column_name in continuous_column_list:
    while True:
        Q1 = df[column_name].quantile(0.25)
        Q3 = df[column_name].quantile(0.75)
        IQR = Q3 - Q1

        df_cleaned_outlier = df[~((df[column_name] < (Q1 - 1.5 * IQR)) | (df[column_name] > (Q3 + 1.5 * IQR)))]

        if len(df_cleaned_outlier) == len(df):
            break
        df = df_cleaned_outlier
print(df.shape[0])
print(df.shape[1])
```

```
30012
21
```

```
continuous_column_list = ['duration']

for column_name in continuous_column_list:
    while True:
        Q1 = df[column_name].quantile(0.25)
        Q3 = df[column_name].quantile(0.75)
        IQR = Q3 - Q1

        df_cleaned_outlier = df[~((df[column_name] < (Q1 - 1.5 * IQR)) | (df[column_name] > (Q3 + 1.5 * IQR)))]

        if len(df_cleaned_outlier) == len(df):
            break
        df = df_cleaned_outlier
print(df.shape[0])
print(df.shape[1])
```

```
26275
21
```

Hình 11: Xử lý outline

❖ **Drop những cột không có ý nghĩa cho mô hình:**

```
df1=df.drop(columns=['day_of_week','month','contact','poutcome','pdays'],axis=1)
df1
```

	age	job	marital	education	default	housing	loan	duration	campaign	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
0	56	housemaid	married	basic.4y	no	no	no	261	1	0	1.1	93.994	-36.4	4.857	5191.0	no
2	37	services	married	high.school	no	yes	no	226	1	0	1.1	93.994	-36.4	4.857	5191.0	no
3	40	admin.	married	basic.6y	no	no	no	151	1	0	1.1	93.994	-36.4	4.857	5191.0	no
4	56	services	married	high.school	no	no	yes	307	1	0	1.1	93.994	-36.4	4.857	5191.0	no
6	59	admin.	married	professional.course	no	no	no	139	1	0	1.1	93.994	-36.4	4.857	5191.0	no
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
41181	37	admin.	married	university.degree	no	yes	no	281	1	0	-1.1	94.767	-50.8	1.028	4963.6	yes
41182	29	unemployed	single	basic.4y	no	yes	no	112	1	1	-1.1	94.767	-50.8	1.028	4963.6	no
41184	46	blue-collar	married	professional.course	no	no	no	383	1	0	-1.1	94.767	-50.8	1.028	4963.6	no
41185	56	retired	married	university.degree	no	yes	no	189	2	0	-1.1	94.767	-50.8	1.028	4963.6	no
41186	44	technician	married	professional.course	no	no	no	442	1	0	-1.1	94.767	-50.8	1.028	4963.6	yes

26275 rows × 16 columns

Hình 12: Mô tả dữ liệu khi xóa các cột không ý nghĩa

❖ **Thay thế giá trị những cột có giá trị yes, no thành 1 và 0:**

```
df1.y.replace(('yes', 'no'), (1, 0), inplace=True)
df1.default.replace(('yes', 'no'), (1, 0), inplace=True)
df1.housing.replace(('yes', 'no'), (1, 0), inplace=True)
df1.loan.replace(('yes', 'no'), (1, 0), inplace=True)
df1
```

	age	job	marital	education	default	housing	loan	duration	campaign	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
0	56	housemaid	married	basic.4y	0	0	0	261	1	0	1.1	93.994	-36.4	4.857	5191.0	0
2	37	services	married	high.school	0	1	0	226	1	0	1.1	93.994	-36.4	4.857	5191.0	0
3	40	admin.	married	basic.6y	0	0	0	151	1	0	1.1	93.994	-36.4	4.857	5191.0	0
4	56	services	married	high.school	0	0	1	307	1	0	1.1	93.994	-36.4	4.857	5191.0	0
6	59	admin.	married	professional.course	0	0	0	139	1	0	1.1	93.994	-36.4	4.857	5191.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
41181	37	admin.	married	university.degree	0	1	0	281	1	0	-1.1	94.767	-50.8	1.028	4963.6	1
41182	29	unemployed	single	basic.4y	0	1	0	112	1	1	-1.1	94.767	-50.8	1.028	4963.6	0
41184	46	blue-collar	married	professional.course	0	0	0	383	1	0	-1.1	94.767	-50.8	1.028	4963.6	0
41185	56	retired	married	university.degree	0	1	0	189	2	0	-1.1	94.767	-50.8	1.028	4963.6	0
41186	44	technician	married	professional.course	0	0	0	442	1	0	-1.1	94.767	-50.8	1.028	4963.6	1

26275 rows × 16 columns

```
df1.default.replace(('unknown'), np.NaN, inplace = True)
df1.dropna(how='any', inplace=True)
df1.isna().sum()
```

```
age             0
job             0
marital         0
education       0
default         0
housing         0
loan            0
duration        0
campaign        0
previous        0
emp.var.rate    0
cons.price.idx  0
cons.conf.idx   0
euribor3m       0
nr.employed     0
y               0
dtype: int64
```

Hình 13: Mô tả dữ liệu khi thay thế các cột có giá trị

❖ **Dummy decoding - mã hóa biến phân loại:**

```
df2 = pd.get_dummies(df1)
df2.head()
```

	age	default	housing	loan	duration	campaign	previous	emp.var.rate	cons.price.idx	cons.conf.idx	...	marital_divorced	marital_married	marital_single	education_basic.4y	education_basic.6y	education_basic.9y	education_high.school
0	56	0	0	0	261	1	0	1.1	93.994	-36.4	...	0	1	0	1	0	0	0
2	37	0	1	0	226	1	0	1.1	93.994	-36.4	...	0	1	0	0	0	0	1
3	40	0	0	0	151	1	0	1.1	93.994	-36.4	...	0	1	0	0	1	0	0
4	56	0	0	1	307	1	0	1.1	93.994	-36.4	...	0	1	0	0	0	0	1
6	59	0	0	0	139	1	0	1.1	93.994	-36.4	...	0	1	0	0	0	0	0

5 rows × 34 columns

```
df2.describe().T
```

	count	mean	std	min	25%	50%	75%	max
age	26275.0	38.461808	9.331374	17.000	31.000	36.000	45.000	66.000
default	26275.0	0.000114	0.010685	0.000	0.000	0.000	0.000	1.000
housing	26275.0	0.543406	0.498122	0.000	0.000	1.000	1.000	1.000
loan	26275.0	0.156689	0.363514	0.000	0.000	0.000	0.000	1.000
duration	26275.0	182.322626	113.730354	0.000	94.000	159.000	252.000	489.000
campaign	26275.0	2.547441	2.791488	1.000	1.000	2.000	3.000	43.000
previous	26275.0	0.187897	0.511546	0.000	0.000	0.000	0.000	7.000
emp.var.rate	26275.0	-0.034204	1.593558	-3.400	-1.800	1.100	1.400	1.400
cons.price.idx	26275.0	93.527568	0.580488	92.201	93.075	93.444	93.994	94.767
cons.conf.idx	26275.0	-40.625522	4.711673	-50.800	-42.700	-41.800	-36.400	-26.900
euribor3m	26275.0	3.503394	1.760166	0.634	1.327	4.856	4.961	5.045
nr.employed	26275.0	5163.022013	73.567922	4963.600	5099.100	5191.000	5228.100	5228.100
y	26275.0	0.077564	0.267490	0.000	0.000	0.000	0.000	1.000
job_admin.	26275.0	0.291532	0.454477	0.000	0.000	0.000	1.000	1.000
job_blue-collar	26275.0	0.188544	0.391154	0.000	0.000	0.000	0.000	1.000
job_entrepreneur	26275.0	0.035890	0.186018	0.000	0.000	0.000	0.000	1.000
job_housemaid	26275.0	0.022493	0.148283	0.000	0.000	0.000	0.000	1.000
job_management	26275.0	0.077108	0.266767	0.000	0.000	0.000	0.000	1.000
job_retired	26275.0	0.025804	0.158553	0.000	0.000	0.000	0.000	1.000
job_self-employed	26275.0	0.036118	0.186587	0.000	0.000	0.000	0.000	1.000
job_services	26275.0	0.094234	0.292160	0.000	0.000	0.000	0.000	1.000
job_student	26275.0	0.019791	0.139283	0.000	0.000	0.000	0.000	1.000
job_technician	26275.0	0.183597	0.387162	0.000	0.000	0.000	0.000	1.000
job_unemployed	26275.0	0.024891	0.155795	0.000	0.000	0.000	0.000	1.000
marital_divorced	26275.0	0.113949	0.317755	0.000	0.000	0.000	0.000	1.000
marital_married	26275.0	0.571874	0.494817	0.000	0.000	1.000	1.000	1.000
marital_single	26275.0	0.314177	0.464196	0.000	0.000	0.000	1.000	1.000
education_basic.4y	26275.0	0.070942	0.256733	0.000	0.000	0.000	0.000	1.000

Hình 14: Mã hóa biến phân loại

2.2.5. Data Visullization:

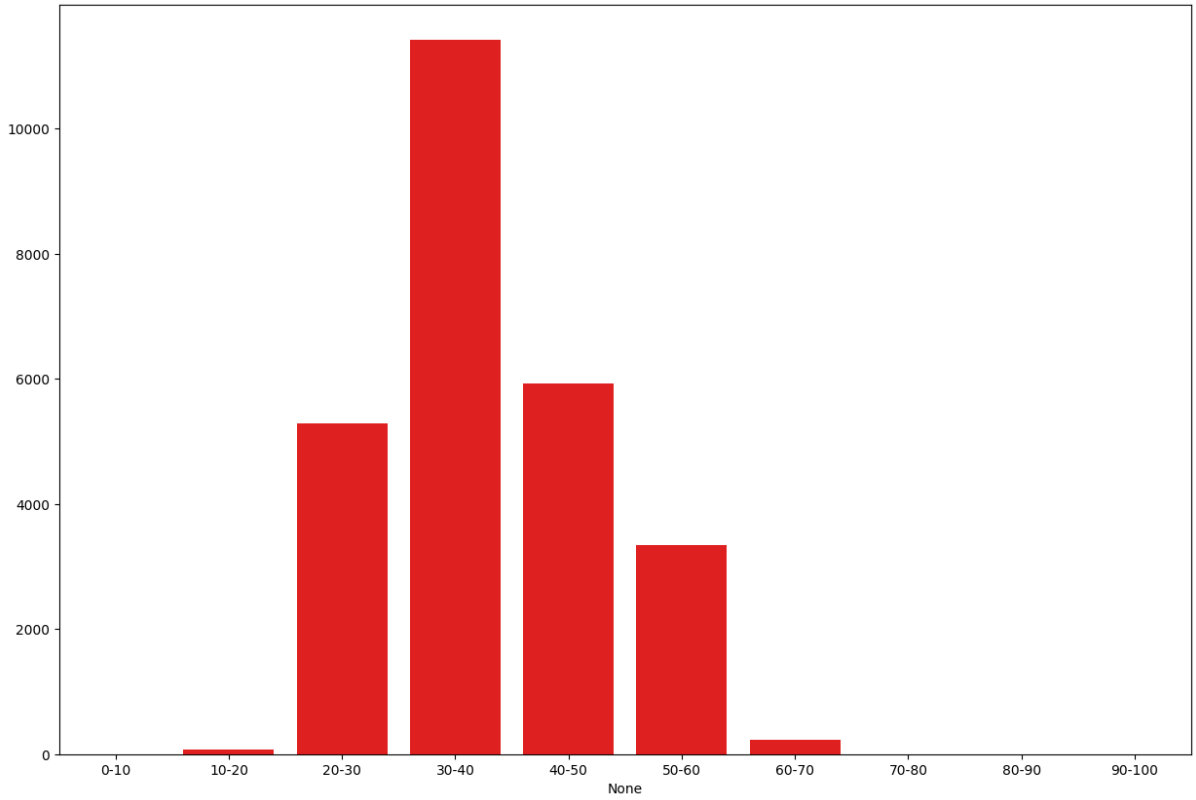
- **Nhóm độ tuổi nào sử dụng giao dịch ngân hàng nhiều nhất?**

```
age_bins = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
age_labels = ['0-10','10-20', '20-30', '30-40', '40-50', "50-60","60-70", "70-80", "80-90","90-100"]
df_visual = df
df_visual['groupage'] = pd.cut(df['age'], age_bins, labels = age_labels, right = True, include_lowest = True)
df_visual.head(10)
count_groupage = df['groupage'].value_counts()
count_groupage
```

```
30-40     11416
40-50      5926
20-30      5290
50-60      3335
60-70       235
10-20        73
0-10          0
70-80         0
80-90         0
90-100        0
Name: groupage, dtype: int64
```

Hình 15: Mô tả dữ liệu về nhóm độ tuổi

```
plt.figure(figsize=(15,10))
sns.barplot(x=count_groupage.index, y=count_groupage.values, color = 'red')
```



Biểu đồ 4: Biểu đồ độ tuổi khách hàng

→ *Nhận xét: Nhóm độ tuổi từ 30 - 40 có tỉ lệ sử dụng cao nhất*

❖ **Khách hàng trong ngân hàng đa số trong tình trạng hôn nhân như thế nào?**

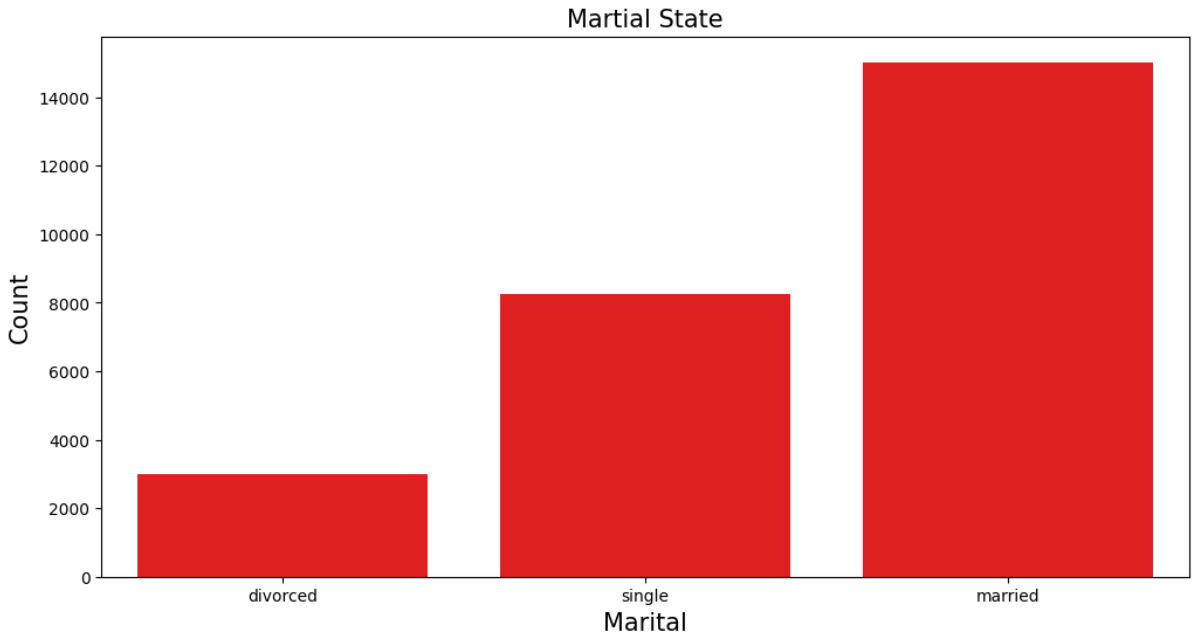
```
count_marital = df['marital'].value_counts().sort_values()
count_marital
```

```
divorced     2994
single       8255
married     15026
Name: marital, dtype: int64
```

```
plt.figure(figsize=(12, 6))
sns.countplot(data=df, x='marital', order=df['marital'].value_counts().index[::-1],
color='red')
plt.title('Martial State', fontsize=15)
plt.xlabel('Marital', fontsize=15)
plt.ylabel('Count', fontsize=15)
```

```
plt.show()
```

Hình 16: Dữ liệu về tình trạng hôn nhân



Biểu đồ 5: Biểu đồ thể hiện tình trạng hôn nhân của khách hàng

--> Nhận xét: Khách hàng của ngân hàng đa số là những người đã lập gia đình

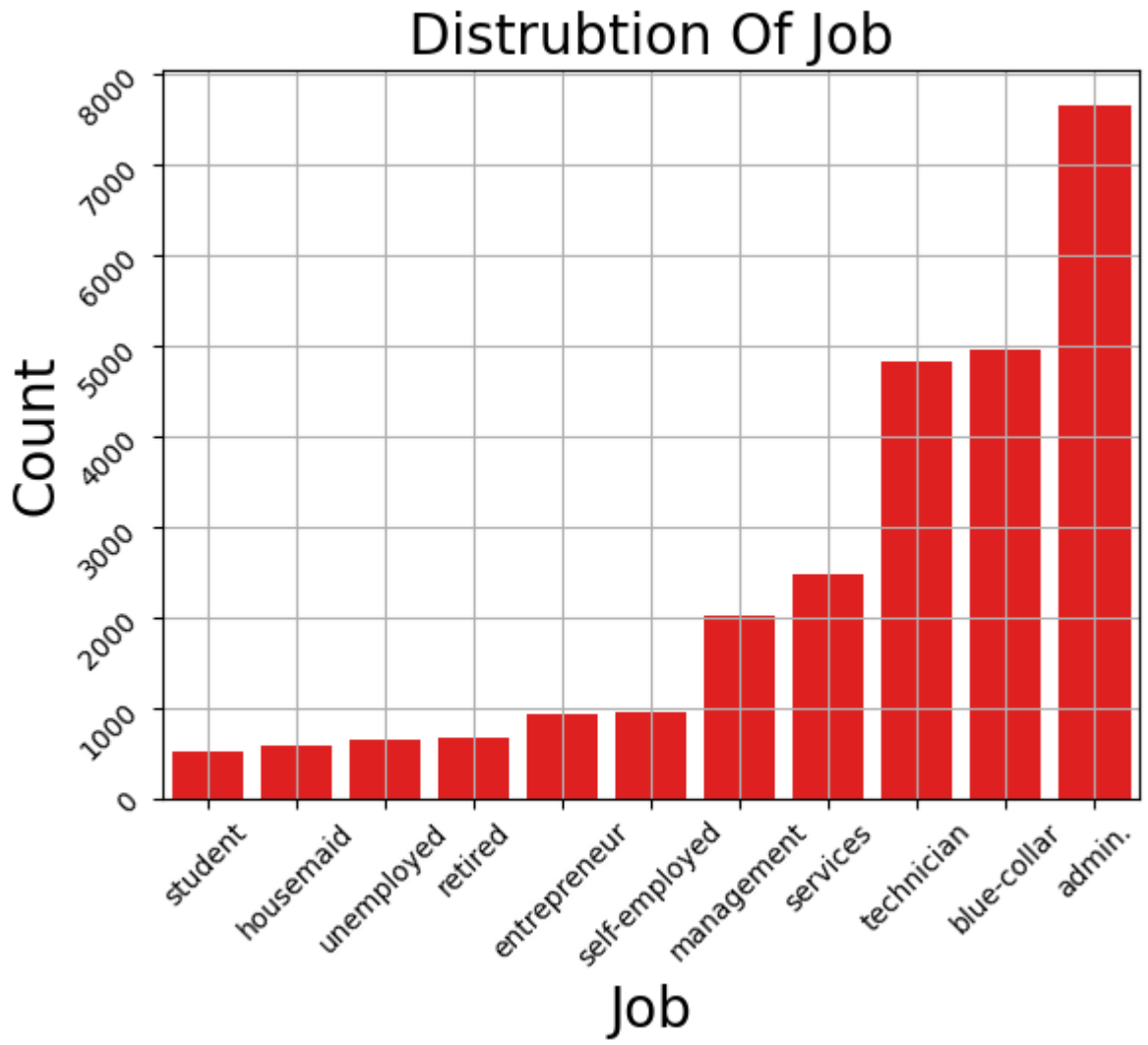
❖ **Trong ngân hàng, nghề nghiệp của khách hàng thường là công việc gì?**

```
count_job = df['job'].value_counts().sort_values()
count_job
```

```
student           520
housemaid         591
unemployed        654
retired           678
entrepreneur      943
self-employed     949
management       2026
services         2476
technician       4824
blue-collar      4954
admin.           7660
Name: job, dtype: int64
```

```
sns.countplot(data=df, x='job', order=df['job'].value_counts().index[::-1], color='red')
plt.grid(True)
plt.title('Distrubtion Of Job',fontsize=20)
plt.xlabel('Job',fontsize=20)
plt.ylabel('Count',fontsize=20)
plt.xticks(rotation=45)
plt.yticks(rotation=45)
plt.show()
```

Hình 17: Mô tả dữ liệu nghề nghiệp của khách hàng



Biểu đồ 6: Biểu đồ thể hiện nghề nghiệp khách hàng

→ *Nhận xét: Công việc của khách hàng đa số là quản trị viên, công ngân và kỹ thuật viên*

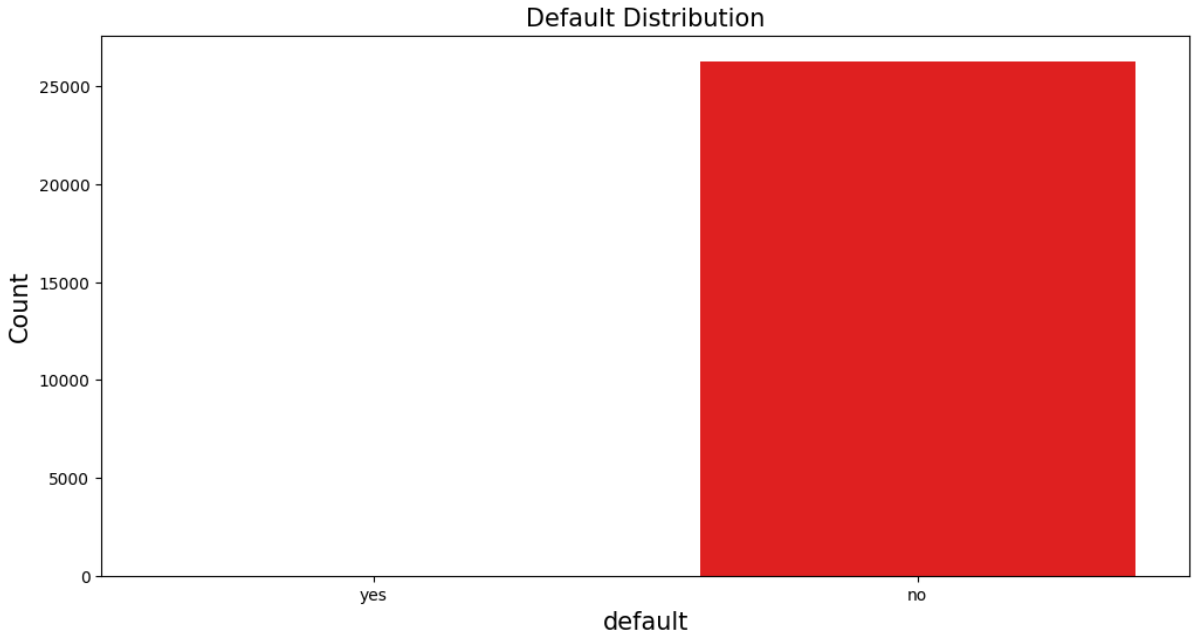
- **Tỉ lệ vỡ nợ trong ngân hàng:**

```
count_default = df['default'].value_counts().sort_values()
count_default
```

```
yes        3
no     26272
Name: default, dtype: int64
```

```
plt.figure(figsize=(12, 6))
sns.countplot(data=df, x='default', order=df['default'].value_counts().index[::-1],
color='red')
plt.title('Default Distribution', fontsize=15)
plt.xlabel('default', fontsize=15)
plt.ylabel('Count', fontsize=15)
plt.show()
```

Hình 18: Mô tả dữ liệu về tình trạng vỡ nợ của khách hàng



Biểu đồ 7: Biểu đồ thể hiện tình trạng vỡ nợ của khách hàng

→ Nhận xét: Xác suất để vỡ nợ là rất thấp, chỉ có 1 số người không có khả năng thanh đúng hạn

❖ **Trình độ bậc giáo dục phổ biến của khách hàng trong ngân hàng?**

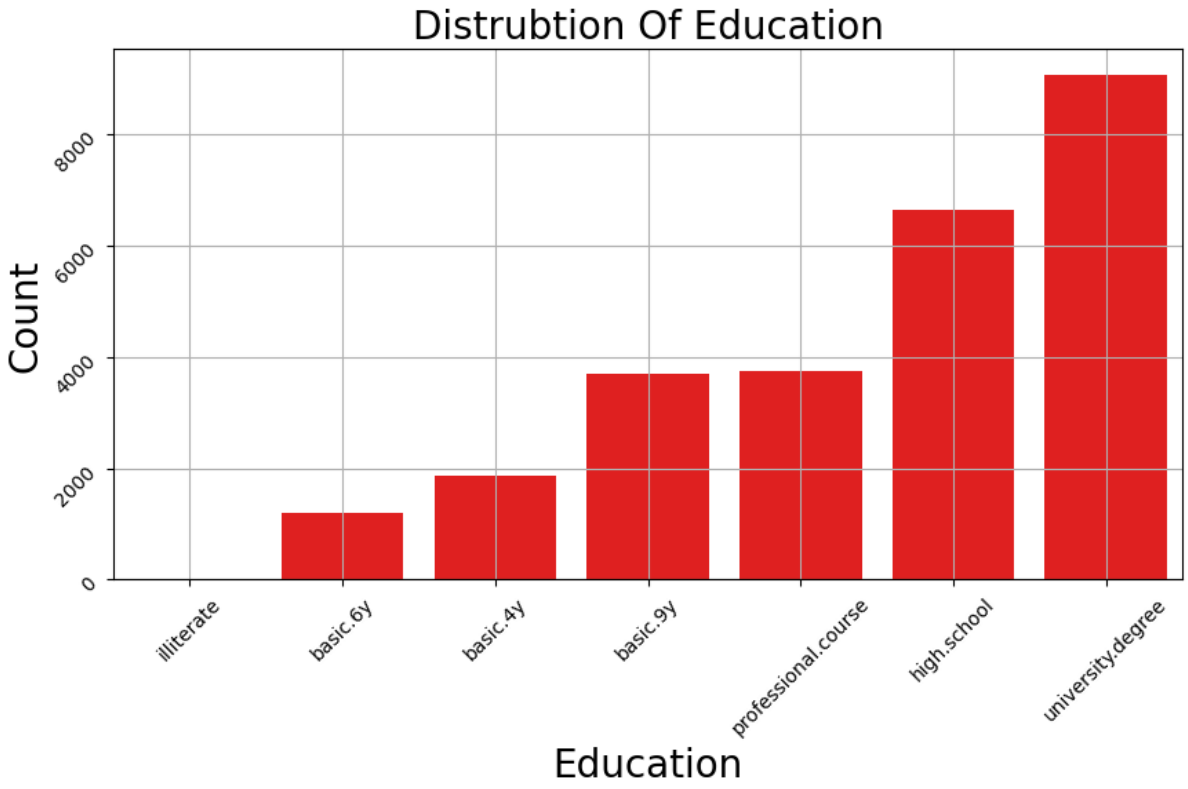
```
count_education = df['education'].value_counts().sort_values()
```

```
count_education
```

```
illiterate               10
basic.6y               1210
basic.4y               1864
basic.9y               3699
professional.course    3753
high.school            6658
university.degree      9081
Name: education, dtype: int64
```

```
plt.figure(figsize=(10,5))
sns.countplot(data=df, x='education', order=df['education'].value_counts().index[::-1], color='red')
plt.grid(True)
plt.title('Distrubtion Of Education',fontsize=20)
plt.xlabel('Education',fontsize=20)
plt.ylabel('Count',fontsize=20)
plt.xticks(rotation=45)
plt.yticks(rotation=45)
plt.show()
```

Hình 19:Mô tả dữ liệu trình độ giáo dục của khách hàng



Biểu đồ 8:Biểu đồ thể hiện trình độ giáo dục phổ biến của khách hàng

→ Nhận xét: Khách hàng đa số là những người đã tốt nghiệp cấp 3 và đại học trở lên.

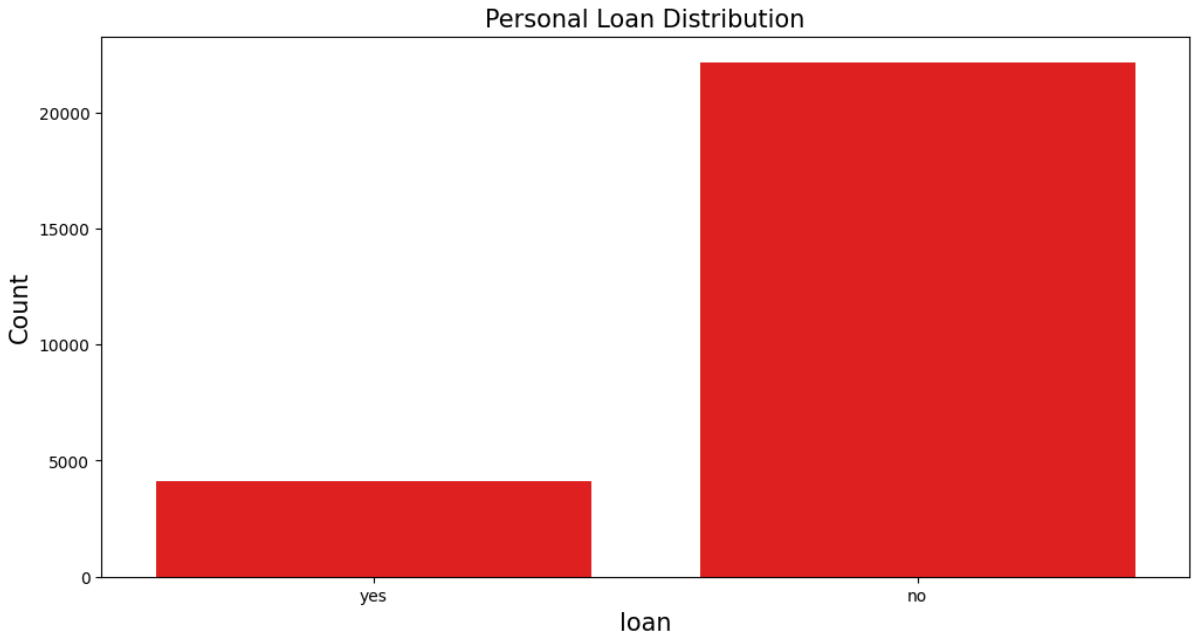
❖ **Khoản vay cá nhân của khách hàng trong ngân hàng nhiều hay ít?**

```
count_loan = df['loan'].value_counts().sort_values()
count_loan
```

```
yes     4117
no     22158
Name: loan, dtype: int64
```

```
plt.figure(figsize=(12, 6))
sns.countplot(data=df, x='loan', order=df['loan'].value_counts().index[::-1],
color='red')
plt.title('Personal Loan Distribution', fontsize=15)
plt.xlabel('loan', fontsize=15)
plt.ylabel('Count', fontsize=15)
plt.show()
```

Hình 20: Mô tả dữ liệu khoản vay cá nhân của khách hàng



Biểu đồ 9:Biểu đồ thể hiện khoản vay cá nhân của khách hàng

→ *Nhận xét: Khách hàng ít khi có khoản vay cá nhân, có lẽ hầu như là tiền gửi.*

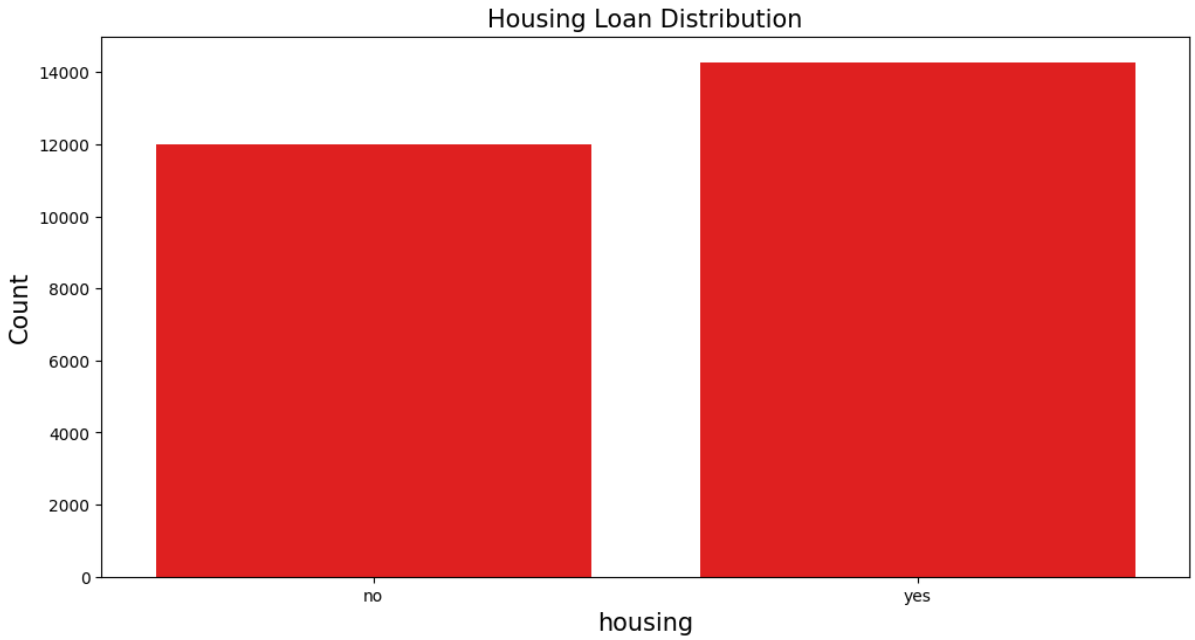
❖ **Khoản vay mua nhà (khoản vay thế chấp) của khách hàng**

```
count_housing = df['housing'].value_counts().sort_values()
count_housing
```

```
no     11997
yes    14278
Name: housing, dtype: int64
```

```
plt.figure(figsize=(12, 6))
sns.countplot(data=df, x='housing', order=df['housing'].value_counts().index[::-1],
color='red')
plt.title('Housing Loan Distribution', fontsize=15)
plt.xlabel('housing', fontsize=15)
plt.ylabel('Count', fontsize=15)
plt.show()
```

Hình 21: Mô tả dữ liệu về khoản vay nhà của khách hàng



Biểu đồ 10: Biểu đồ thể hiện khoản vay nhà của khách hàng

→ *Nhận xét: Mọi người không vay cá nhân nhưng có tỉ lệ vay nhà (khoảng vay thế chấp) cao.*

❖ **Kết quả của chiến dịch marketing trước đó?**

```
count_poutcome = df['poutcome'].value_counts().sort_values()
count_poutcome
```

```
success          938
failure         2989
nonexistent    22348
Name: poutcome, dtype: int64
```

```
labels = ['Success', 'Failure', 'Nonexistent']
colors = ["yellow", "blue", "red"]
plt.title('Kết quả của những chiến dịch marketing trước đó')
plt.pie(count_poutcome, labels=labels, colors=colors)
plt.show()
```

Hình 22: Mô tả dữ liệu về chiến dịch marketing trước đó

Biểu đồ 11:Biểu đồ thể hiện kết quả của chiến dịch marketing trước đó

→ Nhận xét: Tỉ lệ lệ thành công chiếm tỉ trọng rất ít cho thấy hiệu quả của chiến dịch marketing chưa thực sự tốt

- **Mục tiêu mong muốn của ngân hàng (Đầu ra): Lượng khách đã đăng ký một khoản tiền gửi có thời hạn:**

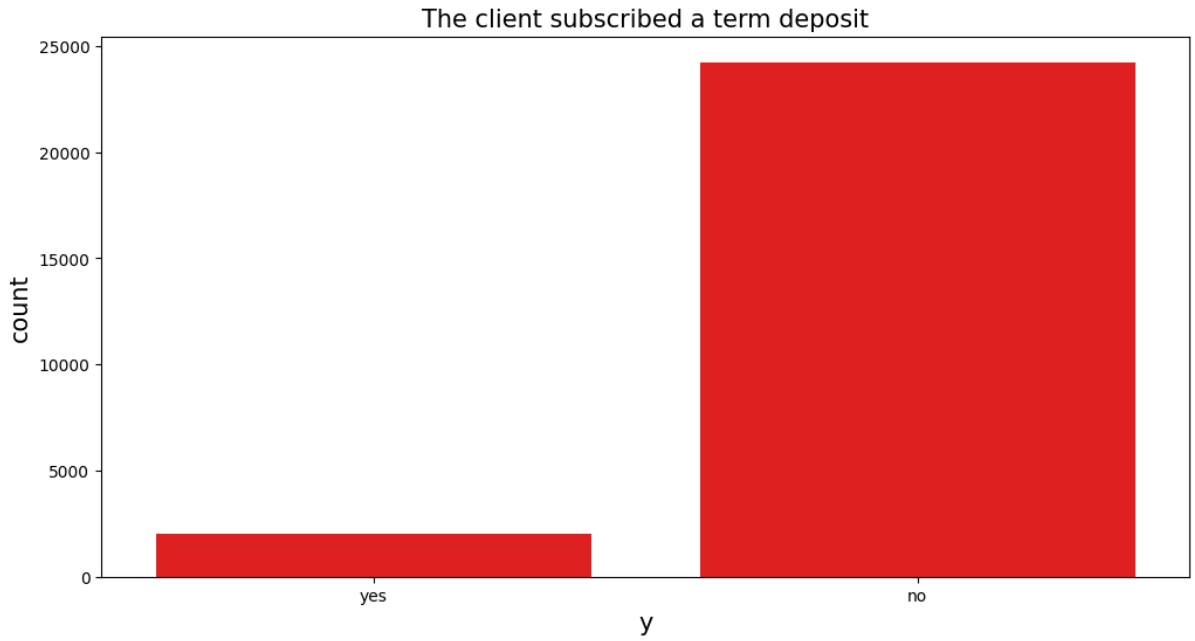
```
count_y = df['y'].value_counts().sort_values()
count_y
```

```
yes     2038
no     24237
Name: y, dtype: int64
```

```
plt.figure(figsize=(12, 6))
sns.countplot(data=df, x='y', order=df['y'].value_counts().index[::-1], color='red')
plt.title('The client subscribed a term deposit', fontsize=15)
plt.xlabel('y', fontsize=15)
plt.ylabel('count', fontsize=15)
```

```
plt.show()
```

Hình 23: Mô tả dữ liệu lượng khách hàng đã ký kết một khoản tiền gửi có kỳ hạn



Biểu đồ 12:Thể hiện lượng khách hàng đã ký kết một khoản tiền gửi có kỳ hạn

→ Nhận xét: Lượng khách hàng đã ký kết một khoản tiền gửi có kỳ hạn là rất ít

2.3. XÂY DỰNG MÔ HÌNH PHÂN LOẠI:

Mục tiêu là phân loại **xác suất khách hàng đăng ký khoản tiền gửi có thời hạn hay không (yes/no)** dựa trên việc khai thác bộ dữ liệu này và dùng thuật toán Logistic Regression để phân lớp.

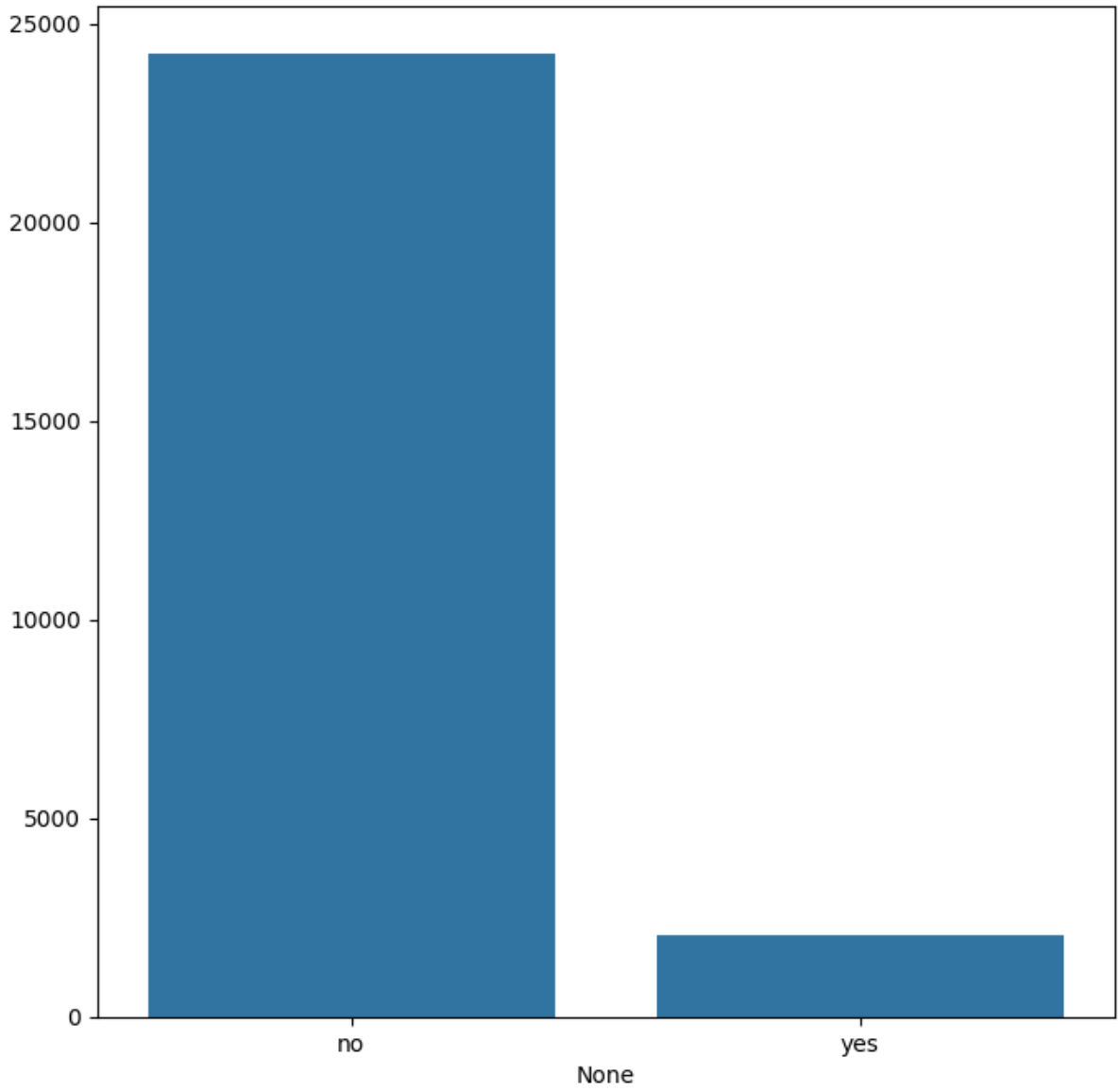
2.3.1. Xét sự chênh lệch biến mục tiêu Y:

```
countY = df['y'].value_counts()
countY
```

```
no     24237
yes     2038
Name: y, dtype: int64
```

```
plt.figure(figsize=(8,8))
sns.barplot(x=countY.index, y=countY.values)
```

Hình 24: Mô tả dữ liệu sự chênh lệch mục tiêu biến Y



Biểu đồ 13:Thể hiện sự chênh lệch mục tiêu biến Y

→ Có sự chênh lệch với tỉ lệ rất lớn ở biến mục tiêu Y

→ Logistic regression như đã nói ở yêu cầu: nó đòi hỏi không có sự chênh lệch về tỉ lệ giá trị bên trong cột

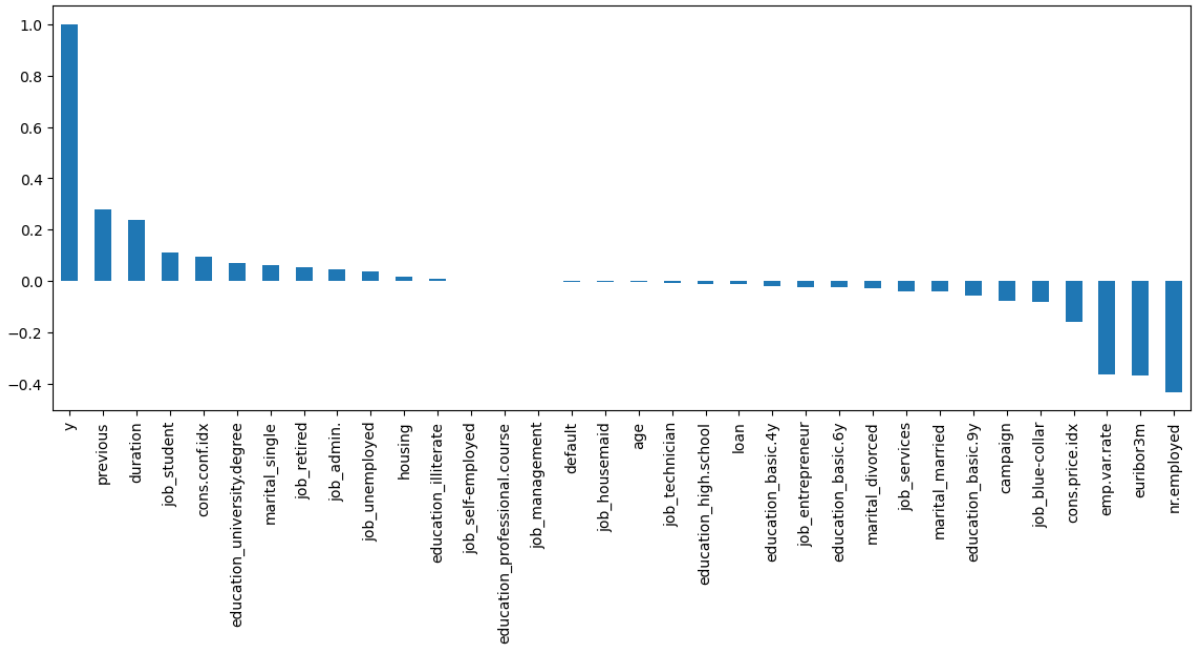
→ Cần có biện pháp để giải quyết mất cân bằng mẫu.

2.3.2. Correlation (xét tương quan của biến mục tiêu Y với những biến độc lập còn lại):

Logistic regression không cho phép có hiện tượng đa cộng tuyến xảy ra, ta cần xét mối quan hệ để xem xét quan hệ giữa các biến tránh xảy ra hiện tượng đa cộng tuyến trước khi đưa vào chạy mô hình.

```
dfy = df2.corr()['y']
dfy
```

```
plt.figure(figsize=(14,5))
dfy.sort_values(ascending = False).plot(kind='bar')
```



Biểu đồ 14:Thể hiện tương quan của biến mục tiêu Y với những biến độc lập còn lại.

→ Không có xuất hiện hiện tượng đa cộng tuyến khi xét tương quan của biến mục tiêu Y với những biến độc lập còn lại.

2.3.3. Code thủ công triển khai chạy mô hình:

```
import pandas as pd
```

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import RFE
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix
from statsmodels.stats.outliers_influence import variance_inflation_factor
from imblearn.over_sampling import SMOTE
from sklearn import metrics

class LogisticRegressionCustom:
    def __init__(self, learning_rate=0.01, num_iterations=5000):
        self.learning_rate = learning_rate
        self.num_iterations = num_iterations
        self.weights = None
        self.bias = None

    def sigmoid(self, z):
        return 1 / (1 + np.exp(-z))

    def initialize_weights_and_bias(self, num_features):
        self.weights = np.zeros((num_features, 1))
        self.bias = 0

    def compute_cost(self, y, y_pred):
        m = y.shape[0]
        cost = -1 / m * np.sum(y * np.log(y_pred) + (1 - y) * np.log(1 - y_pred))
        return cost

    def gradient_descent(self, X, y, y_pred):
        m = X.shape[0]
        dw = 1 / m * np.dot(X.T, (y_pred - y))
        db = 1 / m * np.sum(y_pred - y)
        return dw, db

    def train(self, X, y):
        num_samples, num_features = X.shape
        self.initialize_weights_and_bias(num_features)

        costs = []
```

```
        for i in range(self.num_iterations):
            z = np.dot(X, self.weights) + self.bias
            y_pred = self.sigmoid(z)

            cost = self.compute_cost(y, y_pred)
            costs.append(cost)

            dw, db = self.gradient_descent(X, y, y_pred)

            self.weights -= self.learning_rate * dw
            self.bias -= self.learning_rate * db

            if i % 100 == 0:
                print(f"Iteration {i}, Cost: {cost}")

        plt.figure(figsize=(8, 6))
        plt.plot(range(0, self.num_iterations), costs, marker='o', linestyle='-', color='b')
        plt.title('Cost vs Iteration')
        plt.xlabel('Number of Iteration')
        plt.ylabel('Cost')
        plt.grid(True)
        plt.show()

    def predict(self, X):
        z = np.dot(X, self.weights) + self.bias
        y_pred = self.sigmoid(z)
        return y_pred

def preprocess_data(df):
    X = df.drop(['y'], axis=1)
    Y = df['y']
    x_train, x_test, y_train, y_test = train_test_split(X, Y, train_size=0.7, test_size=0.3,
random_state=100)

    scaler = StandardScaler()
    x_train[['age', 'duration', 'campaign', 'previous']] = scaler.fit_transform(
        x_train[['age', 'duration', 'campaign', 'previous']])

    return x_train, x_test, y_train, y_test

def oversample_smote(x_train, y_train):
    smote = SMOTE()
```

```
    x_train_os, y_train_os = smote.fit_resample(x_train, y_train)

    return x_train_os, y_train_os

def build_logistic_regression(x_train, y_train, feature_subset=None):
    if feature_subset is not None:
        x_train_sm = np.column_stack((np.ones(len(x_train)), x_train[feature_subset]))
    else:
        x_train_sm = np.column_stack((np.ones(len(x_train)), x_train))

    # Use your custom logistic regression
    log_model_custom = LogisticRegressionCustom(learning_rate=0.001, num_iterations=1000)
    log_model_custom.train(x_train_sm, y_train.values.reshape(-1, 1))

    return log_model_custom, x_train_sm

def select_features_rfe(x_train, y_train, step=20):
    logreg = LogisticRegression()
    rfe = RFE(logreg, step=step)
    rfe = rfe.fit(x_train, y_train)

    return rfe.support_, x_train.columns[rfe.support_]

def evaluate_model(y_true, y_pred, threshold=0.5):
    y_pred_final = pd.DataFrame({'Sub': y_true.values, 'Sub_prob': y_pred.flatten()})
    y_pred_final['predict'] = y_pred_final['Sub_prob'].map(lambda x: 1 if x > threshold else 0)

    confusion = confusion_matrix(y_pred_final.Sub, y_pred_final.predict)
    accuracy = accuracy_score(y_pred_final.Sub, y_pred_final.predict)
    precision = precision_score(y_pred_final.Sub, y_pred_final.predict)
    recall = recall_score(y_pred_final.Sub, y_pred_final.predict)
    f1 = f1_score(y_pred_final.Sub, y_pred_final.predict)

    print("Confusion Matrix:")
    print(confusion)
    print("\nMetrics:")
    print(f"Accuracy: {accuracy:.2f}")
    print(f"Precision: {precision:.2f}")
    print(f"Recall: {recall:.2f}")
    print(f"F1 Score: {f1:.2f}")

    return confusion, accuracy
```

```
def draw_roc_curve(actual, probs):
    fpr, tpr, thresholds = metrics.roc_curve(actual, probs, drop_intermediate=False)
    auc_score = metrics.roc_auc_score(actual, probs)
    plt.figure(figsize=(5, 5))
    plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % auc_score)
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic Example')
    plt.legend(loc='lower right')
    plt.show()

def find_optimal_cutoff(y_true, y_prob):
    numbers = [float(x) / 10 for x in range(10)]
    cutoff_df = pd.DataFrame(columns=['prob', 'accuracy', 'sensi', 'speci'])

    for i in numbers:
        y_pred = (y_prob > i).astype(int)
        cm = confusion_matrix(y_true, y_pred)
        total = sum(sum(cm))
        accuracy = (cm[0, 0] + cm[1, 1]) / total
        speci = cm[0, 0] / (cm[0, 0] + cm[0, 1])
        sensi = cm[1, 1] / (cm[1, 0] + cm[1, 1])
        cutoff_df.loc[i] = [i, accuracy, sensi, speci]

    return cutoff_df

def print_model_parameters(model, feature_names):
    print("Model Parameters:")
    for feature, weight in zip(feature_names, model.weights):
        print(f"{feature}: {weight[0]}")
    print(f"Bias: {model.bias}")


def main():
    df = df2
    x_train, x_test, y_train, y_test = preprocess_data(df)
    x_train_os, y_train_os = oversample_smote(x_train, y_train)
```

```
    # Build logistic regression model
    rfe_support, selected_features = select_features_rfe(x_train_os, y_train_os)
    log_model_custom, x_train_sm = build_logistic_regression(x_train_os, y_train_os,
feature_subset=selected_features)

    # Evaluate model on training data
    y_train_pred = log_model_custom.predict(x_train_sm)
    confusion_train, accuracy_train = evaluate_model(y_train_os, y_train_pred)

    # Draw ROC curve
    draw_roc_curve(y_train_os, y_train_pred)

    # Find optimal cutoff
    cutoff_df = find_optimal_cutoff(y_train_os, y_train_pred)
    optimal_cutoff = cutoff_df.loc[cutoff_df['accuracy'].idxmax()]['prob']

    # Evaluate model on test data
    scaler = StandardScaler()
    x_test[['age', 'duration', 'campaign', 'previous']] = scaler.fit_transform(x_test[['age', 'duration',
'campaign', 'previous']])
    x_test = x_test[selected_features]
    x_test_sm = np.column_stack((np.ones(len(x_test)), x_test))
    y_test_pred = log_model_custom.predict(x_test_sm)

    # Evaluate final model on test data with optimal cutoff
    y_test_df = pd.DataFrame(y_test)
    y_test_df['Cust_id'] = y_test_df.index
    y_pred_1 = pd.DataFrame(y_test_pred)
    y_test_df.reset_index(drop=True, inplace=True)
    y_pred_1.reset_index(drop=True, inplace=True)
    y_pred_final = pd.concat([y_test_df, y_pred_1], axis=1)
    y_pred_final.rename(columns={'y': 'Sub', 0: 'Probability'}, inplace=True)
    y_pred_final['predict'] = (y_pred_final['Probability'] > optimal_cutoff).astype(int)
    accuracy_test = accuracy_score(y_pred_final.Sub, y_pred_final.predict)
    y_pred_final = pd.concat([x_test, y_test, y_pred_final], axis=1)
    y_pred_final.dropna(how='any', inplace=True)
    y_pred_final = y_pred_final.drop(columns=['Sub', 'Cust_id'], axis=1)

    print(f"\nTraining Accuracy: {accuracy_train:.2f}")
    print(f"Optimal Cutoff for Test Data: {optimal_cutoff:.2f}")
    print(f"Test Accuracy with Optimal Cutoff: {accuracy_test:.2f}")
    print_model_parameters(log_model_custom, selected_features)
```

```
    return y_pred_final

if __name__ == "__main__":
    result_df = main()
    print("\nResult DataFrame:")
    print(result_df)
```

Hình 25: Mô tả dữ liệu triển khai code thủ công

2.3.4 GIẢI THÍCH CHI TIẾT TỪNG BƯỚC:

1. Import những thư viện liên quan:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection importtrain_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import RFE
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, confusion_matrix
from statsmodels.stats.outliers_influence import variance_inflation_factor
from imblearn.over_sampling import SMOTE
from sklearn import metrics
```

Hình 26: Mô tả dữ liệu import thư viện

2. Tạo Class (lớp) chứa tất cả các hàm tính toán các thông số trong mô hình hồi quy:

```
class LogisticRegressionCustom:
    def __init__(self, learning_rate=0.01, num_iterations=5000):
        self.learning_rate = learning_rate
        self.num_iterations = num_iterations
        self.weights = None
        self.bias = None

    def sigmoid(self, z):
        return 1 / (1 + np.exp(-z))

    def initialize_weights_and_bias(self, num_features):
        self.weights = np.zeros((num_features, 1))
        self.bias = 0

    def compute_cost(self, y, y_pred):
        m = y.shape[0]
```

```
    cost = -1 / m * np.sum(y * np.log(y_pred) + (1 - y) * np.log(1 - y_pred))
    return cost

def gradient_descent(self, X, y, y_pred):
    m = X.shape[0]
    dw = 1 / m * np.dot(X.T, (y_pred - y))
    db = 1 / m * np.sum(y_pred - y)
    return dw, db

def train(self, X, y):
    num_samples, num_features = X.shape
    self.initialize_weights_and_bias(num_features)

    costs = []

    for i in range(self.num_iterations):
        z = np.dot(X, self.weights) + self.bias
        y_pred = self.sigmoid(z)

        cost = self.compute_cost(y, y_pred)
        costs.append(cost)
        dw, db = self.gradient_descent(X, y, y_pred)

        self.weights -= self.learning_rate * dw
        self.bias -= self.learning_rate * db

        if i % 100 == 0:
            print(f"Iteration {i}, Cost: {cost}")

    plt.figure(figsize=(8, 6))
    plt.plot(range(0, self.num_iterations), costs, marker='o', linestyle='-', color='b')
    plt.title('Cost vs Iteration')
    plt.xlabel('Number of Iteration')
    plt.ylabel('Cost')
    plt.grid(True)
    plt.show()

def predict(self, X):
    z = np.dot(X, self.weights) + self.bias
    y_pred = self.sigmoid(z)        return y_pred
```

Hình 27: Mô tả dữ liệu tạo class

Giải thích: Chứa các hàm bên trong class:

- Hàm __init__(self, learning_rate=0.01, num_iterations=5000): Hàm khởi tạo lớp, khởi tạo các thuộc tính learning_rate (tỷ lệ học tập) và num_iterations (số lần lặp).
- Hàm sigmoid(self, z): Hàm sigmoid, được sử dụng để chuyển đổi giá trị z thành giá trị dự đoán y.

```
def sigmoid(self, z):
    return 1 / (1 + np.exp(-z))
```

Hình 28: Mô tả dữ liệu hàm Sigmoid

Hàm sigmoid: Hàm sigmoid biến đổi giá trị đầu vào z từ bất kỳ giá trị nào thành một giá trị nằm trong khoảng từ 0 đến 1. Khi z càng lớn, y càng gần với 1. Khi z càng nhỏ, y càng gần với 0.

- **Truyền giá trị đầu vào z.**
- **Hàm sigmoid được tính:** y = 1 / (1 + e^(-z))
- **Trong đó** e là số Euler, có giá trị xấp xỉ là 2,71828.

- Hàm initialize_weights_and_bias(self, num_features): Hàm khởi tạo các trọng số và độ lệch của mô hình, ban đầu được đặt thành 0.

```
def initialize_weights_and_bias(self, num_features):
    self.weights = np.zeros((num_features, 1))
    self.bias = 0
```

Hình 29: Khởi tạo trọng số và độ lệch của mô hình

- **Trọng số w:**Mảng w thường được sử dụng để lưu trữ các trọng số (weights) trong thuật toán.
  - **Tạo mảng numpy np.zeros () chứa kích thước num_features (hàng) và 1 cột. Tạo thành một vector cột.**
  - **0.01 là giá trị ban đầu được điền vào tất cả các phần tử trong mảng.**
- **Độ lệch bias (b) gán bằng 0**

- Hàm compute_cost(self, y, y_pred): Hàm tính toán chi phí của mô hình, được định nghĩa là hàm mất mát của thuật toán hồi quy logistic.

```
def compute_cost(self, y, y_pred):
    m = y.shape[0]
    cost = -1 / m * np.sum(y * np.log(y_pred) + (1 - y) * np.log(1 - y_pred))
    return cost
```

Hình 30: Mô tả dữ liệu hàm mất mát

Hàm này có hai tham số đầu vào là y và y_pred. Tham số y là tập dữ liệu thực tế, còn tham số y_pred là tập dữ liệu dự đoán của mô hình.
Hàm compute_cost thực hiện theo các bước sau:

1. Tính toán số lượng mẫu dữ liệu trong tập dữ liệu thực tế.
2. Tính toán chi phí cho từng mẫu dữ liệu trong tập dữ liệu thực tế.

3. Tổng hợp chi phí của từng mẫu dữ liệu thành một tổng chi phí.

Chi phí cho từng mẫu dữ liệu được tính theo công thức sau:
cost = -y * log(y_pred) - (1 - y) * log(1 - y_pred)
Trong đó:

- y là giá trị thực tế của mẫu dữ liệu.
- y_pred là giá trị dự đoán của mẫu dữ liệu.
- log là hàm logarit tự nhiên.

Công thức này thể hiện sự khác biệt giữa giá trị thực tế và giá trị dự đoán của mô hình. Chi phí càng nhỏ thì mô hình càng chính xác.

❖ Hàm gradient_descent(self, x, y, y_pred): Hàm cập nhật trọng số w và độ lệch bias của mô hình ở bên trên, sử dụng **phương pháp gradient descent.**

```
def gradient_descent(self, X, y, y_pred):
    m = X.shape[0]
    dw = 1 / m * np.dot(X.T, (y_pred - y))
    db = 1 / m * np.sum(y_pred - y)
    return dw, db
```

Hình 31: Mô tả dữ liệu phương pháp Gradient Descent

Giải thích: Mỗi lần lặp lại của thuật toán gradient descent, hàm gradient_descent sẽ cập nhật các tham số w và b theo hướng gradient của hàm chi phí. Hướng gradient này thể hiện hướng mà hàm chi phí giảm khi thay đổi các tham số của mô hình.

Hàm gradient_descent thực hiện theo các bước sau:

1. Tính toán số lượng mẫu dữ liệu trong tập dữ liệu thực tế.
2. Tính toán gradient của hàm chi phí đối với từng tham số của mô hình.
3. Cập nhật từng tham số của mô hình theo hướng gradient.

Gradient của hàm chi phí đối với từng tham số của mô hình được tính theo công thức sau:
dw = 1 / m * np.dot(X.T, (y_pred - y))
db = 1 / m * np.sum(y_pred - y)
Trong đó:

- m là số lượng mẫu dữ liệu trong tập dữ liệu thực tế.
- X là tập dữ liệu đầu vào.
- y_pred là tập dữ liệu dự đoán của mô hình.
- y là tập dữ liệu thực tế.

❖ Hàm train(self, x, y): Hàm huấn luyện mô hình, thực hiện lặp lại phương pháp gradient descent cho đến khi chi phí của mô hình giảm xuống dưới một ngưỡng nhất định.

```
def train(self, X, y):
    num_samples, num_features = X.shape
    self.initialize_weights_and_bias(num_features)

    costs = []

    for i in range(self.num_iterations):
```

```
        z = np.dot(X, self.weights) + self.bias
        y_pred = self.sigmoid(z)

        cost = self.compute_cost(y, y_pred)
        costs.append(cost)

        dw, db = self.gradient_descent(X, y, y_pred)
        self.weights -= self.learning_rate * dw
        self.bias -= self.learning_rate * db

        if i % 100 == 0:
            print(f"Iteration {i}, Cost: {cost}")

    plt.figure(figsize=(8, 6))
    plt.plot(range(0, self.num_iterations), costs, marker='o', linestyle='-', color='b')
    plt.title('Cost vs Iteration')
    plt.xlabel('Number of Iteration')
    plt.ylabel('Cost')
    plt.grid(True)
    plt.show()
```

Hình 32: Dữ liệu hàm Train

Hàm train thực hiện theo các bước sau:

1. Khởi tạo các tham số của mô hình bằng hàm initialize_weights_and_bias.
2. Lặp lại num_iterations lần:
    - Tính toán chi phí của mô hình bằng hàm compute_cost.
    - Cập nhật các tham số của mô hình bằng hàm gradient_descent.
    - In chi phí của mô hình.

Hàm train được sử dụng để đào tạo một mô hình hồi quy logistic bằng thuật toán gradient descent.
Trong ảnh, hàm train được sử dụng để đào tạo một mô hình hồi quy logistic với hai tham số w và b. Tham số w là trọng số của các đặc trưng đầu vào, còn tham số b là chặn của mô hình.
Mỗi lần lặp lại của thuật toán gradient descent, hàm train sẽ tính toán chi phí của mô hình và cập nhật các tham số w và b theo hướng gradient của hàm chi phí. Quá trình này sẽ lặp lại cho đến khi hàm chi phí đạt đến một giá trị nhỏ nhất nào đó.
Cuối cùng, hàm train sẽ trả về mô hình đã được đào tạo.

❖ Hàm cuối cùng trong class là predict:

```
def predict(self, X):
    z = np.dot(X, self.weights) + self.bias
    y_pred = self.sigmoid(z)
    return y_pred
```

Hình 33: Hàm cuối cùng trong class predict

Hàm predict thực hiện theo các bước sau:

1. Tính toán giá trị đầu ra của mô hình cho mẫu dữ liệu mới bằng hàm sigmoid.
2. Nếu giá trị đầu ra lớn hơn 0.5, thì trả về nhãn 1.
3. Nếu giá trị đầu ra nhỏ hơn 0.5, thì trả về nhãn 0.

Hàm predict được sử dụng để dự đoán nhãn của một mẫu dữ liệu mới dựa trên mô hình hồi quy logistic đã được đào tạo.
Trong ảnh, hàm predict được sử dụng để dự đoán nhãn của một mẫu dữ liệu mới với hai đặc trưng đầu vào. Tham số X của hàm predict là mẫu dữ liệu mới có hai đặc trưng.
Hàm predict sẽ tính toán giá trị đầu ra của mô hình cho mẫu dữ liệu mới bằng hàm sigmoid. Giá trị đầu ra này sẽ nằm trong khoảng từ 0 đến 1.
Nếu giá trị đầu ra lớn hơn 0.5, thì hàm predict sẽ trả về nhãn 1. Điều này có nghĩa là mẫu dữ liệu mới thuộc lớp 1.
Nếu giá trị đầu ra nhỏ hơn 0.5, thì hàm predict sẽ trả về nhãn 0. Điều này có nghĩa là mẫu dữ liệu mới thuộc lớp 0.

4. Khởi tạo Hàm tiền xử lý, chia tập train test và xử lý bất cân bằng dữ liệu:

```
def preprocess_data(df):
    X = df.drop(['y'], axis=1)
    Y = df['y']
    x_train, x_test, y_train, y_test = train_test_split(X, Y, train_size=0.7, test_size=0.3,
random_state=100)

    scaler = StandardScaler()
    x_train[['age', 'duration', 'campaign', 'previous']] = scaler.fit_transform(
        x_train[['age', 'duration', 'campaign', 'previous']])

    return x_train, x_test, y_train, y_test

def oversample_smote(x_train, y_train):
    smote = SMOTE()
    x_train_os, y_train_os = smote.fit_resample(x_train, y_train)

    return x_train_os, y_train_os
```

Hình 34: Hàm tiền xử lý

Giải thích:

❖ **Khởi tạo hàm tiền xử lý:**

- **Xác định biến độc lập X và biến phụ thuộc (mục tiêu) Y:**

```
# Biến độc lập (tất cả các biến trừ biến 'y') và biến phụ thuộc y
X = df2.drop(['y'],1)
Y = df2['y']
```

- **Sau đó Chia tập train và tập test:**
- **Tiếp theo dùng thư viện StandardScaler để tiêu chuẩn các biến age, duration, campaign, previous về cùng một dạng số giúp dễ dàng thực hiện chạy model.**

❖ **Khởi tạo hàm oversample_smote giúp Xử lý bất cân xứng dữ liệu: Logistic Regression không cho phép có sự chênh lệch mẫu đưa vào chạy model. Do đó cần xử lý bước này.**

Sử dụng thư viện SMOTE để xử lý bất cân xứng dữ liệu: là phương pháp sinh mẫu nhằm gia tăng kích thước mẫu của nhóm thiểu số trong trường hợp xảy ra mất cân bằng mẫu

Trong hàm này, xử lý mất cân bằng mẫu giữa X_train_os, y_train_os.

5. Khởi tạo hàm xây dựng mô hình hồi quy Logistic:

```
def build_logistic_regression(x_train, y_train, feature_subset=None):
    if feature_subset is not None:
        x_train_sm = np.column_stack((np.ones(len(x_train)), x_train[feature_subset]))
    else:
        x_train_sm = np.column_stack((np.ones(len(x_train)), x_train))

    # Use your custom logistic regression
    log_model_custom = LogisticRegressionCustom(learning_rate=0.001,
num_iterations=1000)
    log_model_custom.train(x_train_sm, y_train.values.reshape(-1, 1))

    return log_model_custom, x_train_sm

def select_features_rfe(x_train, y_train, step=20):
    logreg = LogisticRegression()
    rfe = RFE(logreg, step=step)
    rfe = rfe.fit(x_train, y_train)

    return rfe.support_, x_train.columns[rfe.support_]
```

Hình 35: Hàm xây dựng mô hình hồi quy

Giải thích:

a. Hàm **build_logistic_regression** được sử dụng để xây dựng một mô hình hồi quy logistic. Hàm này có hai tham số đầu vào là x_train và y_train. Tham số x_train là tập dữ liệu đầu vào, và tham số y_train là tập dữ liệu thực tế.

Hàm build_logistic_regression thực hiện theo các bước sau:

1. Kiểm tra xem tham số feature_subset có được cung cấp hay không.

2. Nếu feature_subset được cung cấp, thì sử dụng các đặc trưng trong feature_subset để xây dựng mô hình.
3. Nếu feature_subset không được cung cấp, thì sử dụng tất cả các đặc trưng trong x_train để xây dựng mô hình.

Trong trường hợp feature_subset được cung cấp, hàm build_logistic_regression sẽ sử dụng các đặc trưng trong feature_subset để xây dựng mô hình hồi quy logistic. Hàm build_logistic_regression sẽ khởi tạo các tham số của mô hình bằng các giá trị ngẫu nhiên, và sau đó sử dụng thuật toán gradient descent để đào tạo mô hình.

Trong trường hợp feature_subset không được cung cấp, hàm build_logistic_regression sẽ sử dụng tất cả các đặc trưng trong x_train để xây dựng mô hình hồi quy logistic. Hàm build_logistic_regression sẽ khởi tạo các tham số của mô hình bằng các giá trị ngẫu nhiên, và sau đó sử dụng thuật toán gradient descent để đào tạo mô hình.

b. Hàm select_features_rfe được sử dụng để lựa chọn các đặc trưng quan trọng nhất trong một tập dữ liệu. Hàm này có ba tham số đầu vào là x_train, y_train, và step. Tham số x_train là tập dữ liệu đầu vào, tham số y_train là tập dữ liệu thực tế, và tham số step là số đặc trưng cần lựa chọn.

Hàm select_features_rfe thực hiện theo các bước sau:

1. Tạo một đối tượng RFE với số đặc trưng cần lựa chọn là step.
2. Huấn luyện đối tượng RFE trên tập dữ liệu (x_train, y_train).
3. Lấy ra danh sách các đặc trưng được giữ lại.

Hàm select_features_rfe sử dụng thuật toán RFE (Recursive Feature Elimination) để lựa chọn các đặc trưng quan trọng nhất trong một tập dữ liệu. Thuật toán RFE sẽ bắt đầu với tất cả các đặc trưng trong tập dữ liệu, và sau đó loại bỏ dần các đặc trưng có ảnh hưởng nhỏ nhất đến mô hình.

Hàm select_features_rfe sử dụng để lựa chọn 20 đặc trưng quan trọng nhất trong tập dữ liệu (x_train, y_train). Hàm select_features_rfe sẽ trả về danh sách 20 đặc trưng được giữ lại.

6. Khởi tạo hàm chứa các chỉ số đánh giá mô hình:

```
def evaluate_model(y_true, y_pred, threshold=0.5):
    y_pred_final = pd.DataFrame({'Sub': y_true.values, 'Sub_prob': y_pred.flatten()})
    y_pred_final['predict'] = y_pred_final['Sub_prob'].map(lambda x: 1 if x > threshold
else 0)

    confusion = confusion_matrix(y_pred_final.Sub, y_pred_final.predict)
    accuracy = accuracy_score(y_pred_final.Sub, y_pred_final.predict)
    precision = precision_score(y_pred_final.Sub, y_pred_final.predict)
    recall = recall_score(y_pred_final.Sub, y_pred_final.predict)
```

```
    f1 = f1_score(y_pred_final.Sub, y_pred_final.predict)

    print("Confusion Matrix:")
    print(confusion)
    print("\nMetrics:")
    print(f"Accuracy: {accuracy:.2f}")
    print(f"Precision: {precision:.2f}")
    print(f"Recall: {recall:.2f}")
    print(f"F1 Score: {f1:.2f}")
    return confusion, accuracy
```

Hình 36: Hàm chứa các chỉ số đánh giá mô hình

Giải thích: Hàm này xử lý các chỉ số giúp đánh giá hiệu suất của mô hình bao gồm:

Hàm này được sử dụng để đánh giá hiệu suất của một mô hình. Hàm này có hai tham số đầu vào là y_true và y_pred. Tham số y_true là tập dữ liệu thực tế, và tham số y_pred là tập dữ liệu dự đoán của mô hình.

Hàm evaluate_model thực hiện theo các bước sau:

1. Tính toán ma trận nhầm lẫn.
2. Tính toán các chỉ số đánh giá hiệu suất.

Ma trận nhầm lẫn là một bảng tóm tắt kết quả dự đoán của mô hình. Ma trận nhầm lẫn có thể được sử dụng để tính toán các chỉ số đánh giá hiệu suất.

Các chỉ số đánh giá hiệu suất thường được sử dụng bao gồm:

- Accuracy: Tỷ lệ mẫu được dự đoán chính xác.
- Precision: Tỷ lệ mẫu được dự đoán là dương thực sự là dương.
- Recall: Tỷ lệ mẫu dương thực sự được dự đoán là dương.
- F1-score: Trung bình cộng của precision và recall, nhân với hệ số 2.

7. **Khởi tạo hàm vẽ đường cong mô hình, optimal giới hạn phân loại**

```
def draw_roc_curve(actual, probs):
    fpr, tpr, thresholds = metrics.roc_curve(actual, probs, drop_intermediate=False)
    auc_score = metrics.roc_auc_score(actual, probs)
    plt.figure(figsize=(5, 5))
    plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % auc_score)
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
```

```
    plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic Example')
    plt.legend(loc='lower right')
    plt.show()
def find_optimal_cutoff(y_true, y_prob):
    numbers = [float(x) / 10 for x in range(10)]
    cutoff_df = pd.DataFrame(columns=['prob', 'accuracy', 'sensi', 'speci'])

    for i in numbers:
        y_pred = (y_prob > i).astype(int)
        cm = confusion_matrix(y_true, y_pred)
        total = sum(sum(cm))
        accuracy = (cm[0, 0] + cm[1, 1]) / total
        speci = cm[0, 0] / (cm[0, 0] + cm[0, 1])
        sensi = cm[1, 1] / (cm[1, 0] + cm[1, 1])
        cutoff_df.loc[i] = [i, accuracy, sensi, speci]
    return cutoff_df
```

Hình 37: Khởi tạo hàm optimal giới hạn phân loại

Giải thích:

8. **Khởi tạo hàm in trọng số sau khi đạo hàm:**

```
def print_model_parameters(model, feature_names):
    print("Model Parameters:")
    for feature, weight in zip(feature_names, model.weights):
        print(f"{feature}: {weight[0]}")
    print(f"Bias: {model.bias}")
```

Hình 38: Khởi tạo hàm in trong phân loại

Giải thích:

- Hàm draw_roc_curve được dùng để vẽ đường đường cong ROC (Receiver Operating Characteristic). Đường cong ROC là một công cụ biểu diễn hiệu suất của một mô hình phân loại. Nó hiển thị mối quan hệ giữa tỷ lệ dương tính giả (FPR) và tỷ lệ dương tính thật (TPR) của mô hình.
- Hàm find_optimal_cutoff có nhiệm vụ thực hiện xử lý ngưỡng giá trị tối ưu cho kết quả mô hình. Cụ thể:
    - **Bước 1: Lặp lại các giá trị ngưỡng**
    - Hàm for i in numbers: lặp lại các giá trị ngưỡng từ 0 đến 1 với bước nhảy là 0,1.
    - **Bước 2: Tính toán độ chính xác, độ nhạy và độ đặc hiệu**
    - Hàm y_pred = (y_prob > i).astype(int) sử dụng thư viện numpy để chuyển đổi xác suất dự đoán thành dự đoán.

- Hàm cm = confusion_matrix(y_true, y_pred) sử dụng thư viện sklearn để tính toán ma trận nhầm lẫn.
- Hàm accuracy = (cm[0, 0] + cm[1, 1]) / total tính toán độ chính xác của mô hình.
- Hàm sensi = cm[1, 1] / (cm[1, 0] + cm[1, 1]) tính toán độ nhạy của mô hình.
- Hàm speci = cm[0, 0] / (cm[0, 0] + cm[0, 1]) tính toán độ đặc hiệu của mô hình.

• **Bước 3: Chọn giá trị ngưỡng tối ưu**

- Hàm cutoff_df.loc[i] = [i, accuracy, sensi, speci] lưu trữ độ chính xác, độ nhạy và độ đặc hiệu của mô hình cho giá trị ngưỡng i trong DataFrame cutoff_df.
- Hàm cutoff_df = cutoff_df.loc[cutoff_df['accuracy'].idxmax()] chọn giá trị ngưỡng có độ chính xác cao nhất.

- Hàm Print_model_parameter: được sử dụng để in các tham số của một mô hình. Hàm này có hai đối số:
  - model: Một mô hình máy học.
  - feature_names: Một danh sách các tên thuộc tính của mô hình.

  Hàm hoạt động như sau:
  - Đầu tiên, hàm in tiêu đề "Model Parameters:".
  - Sau đó, hàm lặp qua từng thuộc tính trong danh sách feature_names.
  - Đối với mỗi thuộc tính, hàm in tên thuộc tính và giá trị trọng số của nó.

9. Hàm main (chứa tất cả hàm con) _hàm xử lý chính:

```
def main():
    df = df2
    x_train, x_test, y_train, y_test = preprocess_data(df)
    x_train_os, y_train_os = oversample_smote(x_train, y_train)

    # Build logistic regression model
    rfe_support, selected_features = select_features_rfe(x_train_os, y_train_os)
    log_model_custom, x_train_sm = build_logistic_regression(x_train_os, y_train_os,
feature_subset=selected_features)

    # Evaluate model on training data
    y_train_pred = log_model_custom.predict(x_train_sm)
    confusion_train, accuracy_train = evaluate_model(y_train_os, y_train_pred)

    # Draw ROC curve
    draw_roc_curve(y_train_os, y_train_pred)

    # Find optimal cutoff
    cutoff_df = find_optimal_cutoff(y_train_os, y_train_pred)
    optimal_cutoff = cutoff_df.loc[cutoff_df['accuracy'].idxmax()]['prob']

    # Evaluate model on test data
```

```
    scaler = StandardScaler()
    x_test[['age', 'duration', 'campaign', 'previous']] = scaler.fit_transform(x_test[['age',
'duration', 'campaign', 'previous']])
    x_test = x_test[selected_features]
    x_test_sm = np.column_stack((np.ones(len(x_test)), x_test))
    y_test_pred = log_model_custom.predict(x_test_sm)

    # Evaluate final model on test data with optimal cutoff
    y_test_df = pd.DataFrame(y_test)
    y_test_df['Cust_id'] = y_test_df.index
    y_pred_1 = pd.DataFrame(y_test_pred)
    y_test_df.reset_index(drop=True, inplace=True)
    y_pred_1.reset_index(drop=True, inplace=True)
    y_pred_final = pd.concat([y_test_df, y_pred_1], axis=1)
    y_pred_final.rename(columns={'y': 'Sub', 0: 'Probability'}, inplace=True)
    y_pred_final['predict'] = (y_pred_final['Probability'] > optimal_cutoff).astype(int)
    accuracy_test = accuracy_score(y_pred_final.Sub, y_pred_final.predict)
    y_pred_final = pd.concat([x_test, y_test, y_pred_final], axis=1)
    y_pred_final.dropna(how='any', inplace=True)
    y_pred_final = y_pred_final.drop(columns=['Sub', 'Cust_id'], axis=1)

    print(f"\nTraining Accuracy: {accuracy_train:.2f}")
    print(f"Optimal Cutoff for Test Data: {optimal_cutoff:.2f}")
    print(f"Test Accuracy with Optimal Cutoff: {accuracy_test:.2f}")
    print_model_parameters(log_model_custom, selected_features)
    return y_pred_final

if __name__ == "__main__":
    result_df = main()
    print("\nResult DataFrame:")
    print(result_df)
```

Hình 39: Mô tả dữ liệu hàm main

Giải thích: Hàm này có nhiệm vụ khởi tạo chương trình và thực hiện các công việc chính của chương trình.

Hàm này sẽ gọi tên của những hàm xử lý con bên trên, tổng hợp vào hàm chính để lý chung. Do những hàm bên trên chỉ là khai báo trước khi đưa vào hoạt động chung trong hàm Main.

Hàm này thực hiện các công việc sau (các công việc đã được khai báo ở những hàm phía trên:

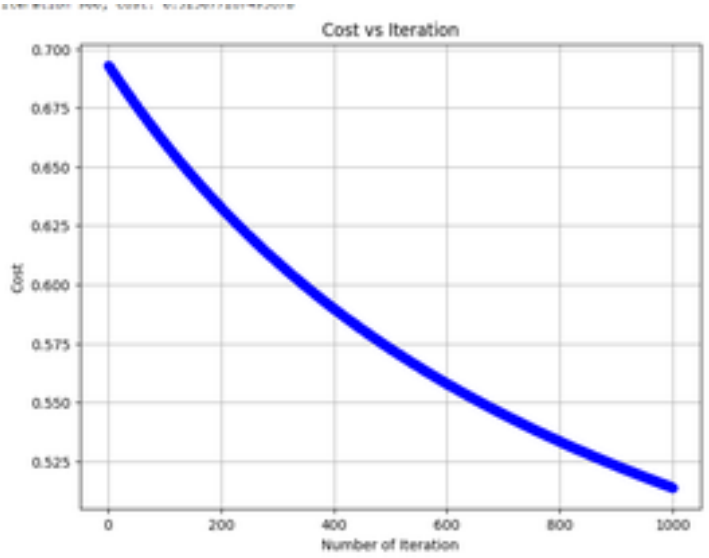
- Đầu tiên, hàm tải dữ liệu từ tệp df2

- Tiếp theo, hàm thực hiện tiền xử lý dữ liệu, bao gồm loại bỏ các giá trị thiếu, chuẩn hóa dữ liệu và chọn các thuộc tính quan trọng nhất.
- Sau đó, hàm xây dựng mô hình hồi quy logistic.
- Tiếp theo, hàm đánh giá mô hình trên tập dữ liệu đào tạo, bao gồm tính toán độ chính xác và vẽ đường cong ROC.
- Tiếp theo, hàm tìm ngưỡng tối ưu cho mô hình.
- Cuối cùng, hàm đánh giá mô hình trên tập dữ liệu thử nghiệm, bao gồm tính toán độ chính xác với ngưỡng tối ưu.

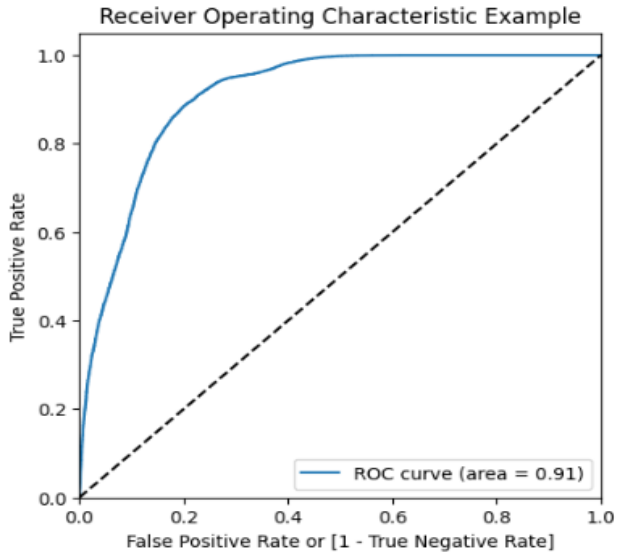
2.3.4 KẾT QUẢ:

a. KẾT QUẢ CHẠY MÔ HÌNH:

```
Iteration 0, Cost: 0.6931471805599452
Iteration 100, Cost: 0.7882189194099944
Iteration 200, Cost: 0.7492825180791831
Iteration 300, Cost: 0.7090689702858909
Iteration 400, Cost: 0.6679550523098322
Iteration 500, Cost: 0.626359306771245
Iteration 600, Cost: 0.5849523167612085
Iteration 700, Cost: 0.5456295893918663
Iteration 800, Cost: 0.5167417790398883
Iteration 900, Cost: 0.5038567020291623
```




```
Confusion Matrix:
[[12296  4696]
 [ 1005 15987]]

Metrics:
Accuracy: 0.83
Precision: 0.77
Recall: 0.94
F1 Score: 0.85
```




```
Training Accuracy: 0.81
Optimal Cutoff for Test Data: 0.60
Test Accuracy with Optimal Cutoff: 0.82
Model Parameters:
housing: -0.04101252138954872
loan: -0.04491337477462106
duration: -0.028188059032239903
campaign: 0.17027462936807608
emp.var.rate: -0.04982767950761279
job_blue-collar: -0.3530121571181157
job_management: -0.03987533561763162
job_services: -0.012459087559332541
marital_divorced: -0.01995231049994109
marital_married: -0.021120585643830413
marital_single: -0.0476997121324584
education_basic.4y: -0.008080446888835137
education_basic.6y: -0.013210010720085406
education_basic.9y: -0.009697636421271291
education_high.school: -0.029945342461886477
education_university.degree: -0.034903606757763155
Bias: -0.04101252138954871
```

```
Result DataFrame:
      housing  loan  duration  campaign  emp.var.rate  job_blue-collar  \
4135      1.0   0.0 -0.011522 -0.185165           1.1              0.0
3023      1.0   0.0  0.446710 -0.185165           1.1              1.0
1803      0.0   0.0 -0.478566 -0.555543           1.1              0.0
5920      1.0   0.0 -1.095417  0.555590           1.1              0.0
4593      1.0   0.0 -1.139478 -0.185165           1.1              0.0
...       ...   ...       ...       ...           ...              ...
3968      0.0   0.0  0.508395 -0.555543           1.1              1.0
5903      1.0   0.0 -0.575500 -0.185165           1.1              1.0
6723      0.0   1.0  2.024087 -0.555543           1.1              0.0
6049      0.0   0.0 -0.681246 -0.185165           1.1              0.0
1304      1.0   0.0 -0.064395 -0.555543           1.1              1.0

      job_management  job_services  marital_divorced  marital_married  \
4135             0.0           0.0               0.0              0.0
3023             0.0           0.0               0.0              1.0
1803             0.0           0.0               0.0              1.0
5920             0.0           1.0               0.0              1.0
4593             0.0           0.0               0.0              1.0
...              ...           ...               ...              ...
3968             0.0           0.0               0.0              1.0
5903             0.0           0.0               0.0              1.0
6723             1.0           0.0               0.0              0.0
6049             0.0           0.0               0.0              1.0
1304             0.0           0.0               1.0              0.0

      marital_single  education_basic.4y  education_basic.6y  \
4135             1.0                 0.0                 0.0
3023             0.0                 1.0                 0.0
1803             0.0                 0.0                 0.0
5920             0.0                 0.0                 0.0
4593             0.0                 1.0                 0.0
...              ...                 ...                 ...
3968             0.0                 1.0                 0.0
5903             0.0                 0.0                 0.0
6723             1.0                 0.0                 0.0
6049             0.0                 1.0                 0.0
1304             0.0                 0.0                 0.0
```

	education_basic.9y	education_high.school	education_university.degree	\
4135	0.0	0.0	1.0	
3023	0.0	0.0	0.0	
1803	0.0	1.0	0.0	
5920	0.0	1.0	0.0	
4593	0.0	0.0	0.0	
...	...	...	...	
3968	0.0	0.0	0.0	
5903	1.0	0.0	0.0	
6723	0.0	0.0	1.0	
6049	0.0	0.0	0.0	
1304	0.0	0.0	0.0	

	y	Probability	predict
4135	0.0	0.330397	0.0
3023	0.0	0.721244	1.0
1803	0.0	0.361749	0.0
5920	0.0	0.329177	0.0
4593	0.0	0.378893	0.0
...	...	...	...
3968	0.0	0.349861	0.0
5903	0.0	0.560573	0.0
6723	0.0	0.651126	1.0
6049	0.0	0.732049	1.0
1304	0.0	0.429585	0.0

[1349 rows x 19 columns]

Hình 40: Mô tả dữ liệu kết quả mô hình

ĐÁNH GIÁ MÔ HÌNH VÀ KẾT LUẬN:

Các tiêu chí đánh giá:

- **Ngưỡng tối ưu cho mô hình (optimal cutoff for test data) = 0.6:** Ngưỡng tối ưu cho mô hình là **0.6**. Với ngưỡng này, mô hình sẽ dự đoán khách hàng đăng ký khoản vay khi xác suất đăng ký của khách hàng lớn hơn hoặc bằng 0.6

⇨ **Các chỉ số đánh giá hiệu suất mô hình:**

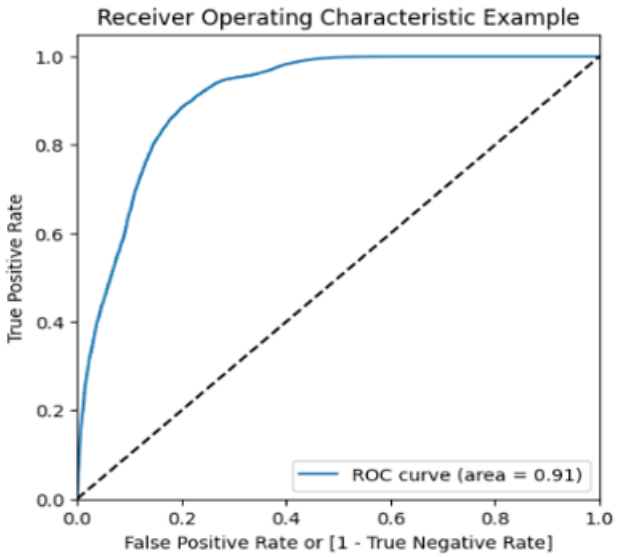
- **Accurancy = 0.81 (81%) => Mô hình có thể dự đoán khá chính xác liệu một khách hàng có đăng ký khoản vay hay không.** Độ chính xác của mô hình trên tập dữ liệu thử nghiệm là **81%** (accurancy)
- Độ chính xác khi test dữ liệu phân loại với ngưỡng tối ưu (0.6) = 0.82 (82%) => Cho thấy độ chính xác chạy mô hình phân loại cao. Mô hình có ý nghĩa và chính xác nếu sử dụng.
- và F1-score để đánh giá khả năng dự đoán của mô hình là 0.84 => Khả năng dự đoán của mô hình đạt mức cao 84%.
- Ngoài ra còn những chỉ số:

• Độ chính xác dự đoán tích cực (precision) = 0.74

• Độ chính xác nhận dạng tích cực (recall) = 0.96

⇨ Những chỉ số trên đều ra kết quả cao > 0.7 cho thấy mô hình hồi quy Logistic phân loại chính xác và đều đo lường ra kết quả dương. Mô hình có ý nghĩa phân loại và dự đoán cao.

- Xem xét biểu đồ ROC (Receiver Operating Characteristic) và tính toán diện tích dưới đường cong ROC (AUC) để đánh giá khả năng phân loại của mô hình.



Biểu đồ 15: Biểu đồ ROCE

Nhìn chung, biểu đồ ROCE cho thấy mô hình có hiệu suất khá tốt. Nhìn vào biểu đồ có thể thấy được:

- True positive rate: True positive rate (TPR) là tỷ lệ phần trăm các khách hàng sẽ đăng ký khoản vay mà mô hình dự đoán đúng là sẽ đăng ký khoản vay. Trong trường hợp này, TPR là khoảng 80%. Điều này cho thấy mô hình có thể dự đoán chính xác khoảng 80% các khách hàng sẽ đăng ký khoản vay.
- True negative rate: True negative rate (TNR) là tỷ lệ phần trăm các khách hàng sẽ không đăng ký khoản vay mà mô hình dự đoán đúng là sẽ không đăng ký khoản vay. Trong trường hợp này, TNR là khoảng 90%. Điều này cho thấy mô hình có thể dự đoán chính xác khoảng 90% các khách hàng sẽ không đăng ký khoản vay.
- Độ chính xác: Độ chính xác (accuracy) là tỷ lệ phần trăm các dự đoán của mô hình là chính xác. Trong trường hợp này, độ chính xác là khoảng 81%. Điều này cho thấy mô hình có thể dự đoán chính xác khoảng 81% các khách hàng, bao gồm cả các khách hàng sẽ đăng ký khoản vay và các khách hàng sẽ không đăng ký khoản vay.

❖ Nhóm cũng đưa ra kết quả y_predict về phân loại khách hàng về mục **tiêu mong muốn của ngân hàng (Đầu ra): Lượng khách đã đăng ký một khoản tiền gửi có thời hạn (0 và 1)**

	y	Probability	predict
4135	0.0	0.330397	0.0
3023	0.0	0.721244	1.0
1803	0.0	0.361749	0.0
5920	0.0	0.329177	0.0
4593	0.0	0.378893	0.0
...	...	...	...
3968	0.0	0.349861	0.0
5903	0.0	0.560573	0.0
6723	0.0	0.651126	1.0
6049	0.0	0.732049	1.0
1304	0.0	0.429585	0.0

PHÂN TÍCH ƯU – KHUYẾT ĐIỂM VỀ HIỆU SUẤT (SO SÁNH VỚI HAI THUẬT TOÁN SVM (SUPPORT VECTOR MACHINE) VÀ DICISION TREE:

- **Chia bộ dữ liệu thành tập train _ test như cũ:**

```
# Biến độc lập (tất cả các biến trừ biến 'y') và biến phụ thuộc y
X = df2.drop(['y'],1)
Y = df2['y']
```

```
<ipython-input-38-bb8d2a14e308>:2: FutureWarning:

In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only.
```

```
#Chia bộ dữ liệu thành tập train và tập test
x_train, x_test, y_train, y_test = train_test_split(X, Y, train_size=0.7, test_size=0.3, random_state=123)
```

```
scaler = StandardScaler()

x_train[['age','duration','campaign','previous']] = scaler.fit_transform(x_train[['age','duration','campaign','previous']])
```

```
x_train.shape, y_train.shape
```

```
((18392, 33), (18392,))
```

```
y_train.value_counts()
```

```
0    16983
1     1409
Name: y, dtype: int64
```

Hình 41: Mô tả dữ liệu chia tập Train - Test như cũ

- Import thư viện Decision tree và SVM chạy mô hình:
  - DECISION TREE

```
from sklearn import tree
dtc = tree.DecisionTreeClassifier()
dtc.fit(x_train, y_train.ravel())

dtc_predict_test = dtc.predict(x_test)

print("Confusion Matrix")
print("{0}".format(metrics.confusion_matrix(y_test, dtc_predict_test, labels = [1, 0])))
print("")

accuracy_dtc = metrics.accuracy_score(y_test, dtc_predict_test)
print("Accuracy: {0:.4f}".format(accuracy_dtc))
print()

print("Classification Report")
print(metrics.classification_report(y_test, dtc_predict_test, labels = [1, 0]))
```

```
Confusion Matrix
[[ 428  201]
 [1611 5643]]

Accuracy: 0.7701

Classification Report
              precision    recall  f1-score   support

           1       0.21      0.68      0.32       629
           0       0.97      0.78      0.86      7254

    accuracy                           0.77      7883
   macro avg       0.59      0.73      0.59      7883
weighted avg       0.91      0.77      0.82      7883
```

Hình 42: Mô tả dữ liệu về cây quyết định

- SVM

```
from sklearn import svm

svmc = svm.SVC()
svmc.fit(x_train, y_train.ravel())

svmc_predict_test = svmc.predict(x_test)

print("Confusion Matrix")
print("{0}".format(metrics.confusion_matrix(y_test, svmc_predict_test, labels = [1, 0])))
print("")

accuracy_svmc = metrics.accuracy_score(y_test, svmc_predict_test)
print("Accuracy: {0:.4f}".format(accuracy_svmc))
print()

print("Classification Report")
print(metrics.classification_report(y_test, svmc_predict_test, labels = [1, 0]))
```

```
Confusion Matrix
[[   0  629]
 [   0 7254]]

Accuracy: 0.9202

Classification Report
              precision    recall  f1-score   support

           1       0.00      0.00      0.00       629
           0       0.92      1.00      0.96      7254

    accuracy                           0.92      7883
   macro avg       0.46      0.50      0.48      7883
weighted avg       0.85      0.92      0.88      7883
```

Hình 43: Mô tả dữ liệu về SVM

*** KẾT LUẬN SO SÁNH HIỆU SUẤT:
Những thông số của Logistic:

```
Confusion Matrix:
[[12296  4696]
 [ 1005 15987]]

Metrics:
Accuracy: 0.83
Precision: 0.77
Recall: 0.94
F1 Score: 0.85
```

```
results = pd.DataFrame([
    {'Algorithm' : 'Decision Tree', 'Accuracy' : accuracy_dtc*100},
    {'Algorithm' : 'Support Vector Machine', 'Accuracy' : accuracy_svmc*100}
])

results.sort_values(by=['Accuracy'], ascending=False)
```

	Algorithm	Accuracy
1	Support Vector Machine	92.020804
0	Decision Tree	77.013827

Hình 44: Mô tả dữ liệu về kết luận so sánh hiệu suất (thông số của SVM, Decision Tree)

- Nhìn chung Logistic regression có độ chính xác cao accurancy = 0.8 > decision tree = 0.7. Nhưng < hơn thuật toán SVM = 0.92
- **Ưu điểm** có thể thấy của Thuật toán Logistic là kết quả ra sẽ biết chính xác biến X ảnh hưởng biến mục tiêu Y như thế nào vì recall của Logistic regression sẽ cao
- **Nhược điểm:** Có thể thấy thuật toán Logistic Regression sẽ phù hợp với những tập dữ liệu nhỏ. Do thuật toán SVM sẽ phù hợp với tập dữ liệu lớn hơn.
- Nếu dữ liệu quá lớn và quá phức tạp thì thuật toán Logistic sẽ chạy rất chậm và xác suất phân loại độ chính xác sẽ không cao và đúng.

FILE GOOGLE COLAB PHÂN TÍCH:
https://colab.research.google.com/drive/1A7xqZFk81v8ZgsGoQn7p4kZf-39eunoP?usp=sharing

HẾT