

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO CUỐI KÌ MÔN

NHẬP MÔN HỌC MÁY

Người hướng dẫn: **PGT.TS LÊ ANH CƯỜNG**

Người thực hiện: **NGUYỄN THỊ THU YẾN - 52000869**

NGUYỄN KHẮC VĂN - 52000868

TRƯƠNG THÔNG THẾ THÁI - 52000714

Lớp : 20050401

Khoá : 24

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO CUỐI KÌ MÔN

NHẬP MÔN HỌC MÁY

Người hướng dẫn: **PGS.TS LÊ ANH CƯỜNG**

Người thực hiện: **NGUYỄN THỊ THU YẾN - 52000869**

NGUYỄN KHẮC VĂN - 52000868

TRƯƠNG THÔNG THỂ THÁI - 52000714

Lớp : 20050401

Khoá : 24

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Nhóm chúng em xin gửi lời cảm ơn chân thành đến khoa công nghệ thông tin, nhà trường và đặc biệt là thầy **(PGS.TS) Lê Anh Cường** đã tận tình hướng dẫn nhóm em trong quá trình giảng dạy bộ môn Nhập môn Học máy (Machine learning)

Nhờ sự hướng dẫn và chỉ bảo tận tình của thầy, nhóm em đã có thể hoàn thành bài báo cáo của mình một cách chu đáo và đầy đủ. Nhóm em đã học hỏi được rất nhiều kiến thức và kinh nghiệm từ thầy, giúp nhóm có thêm hành trang và kỹ năng để áp dụng vào công việc sau này.

Em xin chân thành cảm ơn thầy đã dành thời gian quý báu để góp ý và chỉnh sửa bài báo cáo của nhóm. Nhóm em sẽ tiếp tục nỗ lực học tập và nghiên cứu để đạt được những thành tích tốt hơn trong tương lai.

Một lần nữa, nhóm em xin chân thành cảm ơn khoa, nhà trường và thầy.

Trân trọng!

ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi và được sự hướng dẫn của PGS.TS Lê Anh Cường. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 22 tháng 12 năm 2023

Tác giả

(ký tên và ghi rõ họ tên)

Nguyễn Khắc Văn

Nguyễn Thị Thu Yến

Trương Thông Thế Thái

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

TÓM TẮT

Câu 1 (3 điểm):

Trình bày một bài nghiên cứu, đánh giá của em về các vấn đề sau:

- 1) Tìm hiểu, so sánh các phương pháp Optimizer trong huấn luyện mô hình học máy;
- 2) Tìm hiểu về Continual Learning và Test Production khi xây dựng một giải pháp học máy để giải quyết một bài toán nào đó.

Câu 2: 4 điểm

Đưa ra một bài toán dự đoán có thể giải quyết bằng học máy (machine learning) với các yêu cầu sau:

- Số Feature/Attribute gồm nhiều kiểu: categorical và numerical;
 - Dữ liệu phải chưa được học, thực tập trên lớp và trong bài tập về nhà;
- 1) Phân tích thống kê trên dữ liệu, vẽ các đồ thị để hiểu bài toán, hiểu dữ liệu. Tìm hiểu các đặc trưng và đánh giá vai trò của các đặc trưng đối với mục tiêu bài toán;
 - 2) Ứng dụng các mô hình học máy cơ bản để giải quyết bài toán, bao gồm cả các mô hình thuộc Ensemble Learning;
 - 3) Sử dụng Feed Forward Neural Network và Recurrent Neural Network (hoặc mô thuộc loại này) để giải quyết bài toán;
 - 4) Áp dụng các kỹ thuật tránh Overfitting trên các mô hình của câu (2) và câu (3) để giải quyết bài toán;
 - 5) Sau khi huấn luyện xong mô hình thì muốn cải thiện độ chính xác, ta sẽ làm gì để giải quyết nó? Phân tích các trường hợp sai, đề ra giải pháp và thực hiện nó, sau đó đánh giá xem có cải tiến so với trước không.

MỤC LỤC

| | |
|--|-----------|
| LỜI CẢM ƠN | 1 |
| PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN | 3 |
| TÓM TẮT | 4 |
| MỤC LỤC BẢNG BIỂU, HÌNH VẼ..... | 7 |
| PHẦN 1 – CÂU 1 : NGHIÊN CỨU CÁ NHÂN | 8 |
| 1. Tìm hiểu, so sánh các phương pháp Optimizer | 8 |
| 1.1 Các phương pháp Optimizer trong huấn luyện mô hình học máy 8 | |
| 1.2 So sánh các phương pháp Optimizer trong huấn luyện mô hình học máy | 18 |
| 2. Tìm hiểu về Continual Learning và Test Production | 19 |
| 2.1 Continual Learning..... | 19 |
| 2.2 Test Production | 20 |
| 2.3 Kết hợp Continual Learning và Test Production giải quyết bài toán học máy..... | 22 |
| PHẦN 2 – CÂU 2 : XÂY DỰNG MÔ HÌNH HỌC MÁY – DỰ ĐOÁN PHÊ DUYỆT THẺ TÍN DỤNG TỰ ĐỘNG..... | 25 |
| 1. Giới thiệu dữ liệu sử dụng cho bài toán | 25 |
| 1.1 Tiếp dữ liệu: Credit Card Approval Prediction | 25 |
| 1.2 Đánh giá các feature của dữ liệu | 34 |
| 2. Xây dựng mô hình học máy | 35 |
| 2.1 Các mô hình sử dụng | 35 |
| 2.2 Phòng tránh Overfitting | 36 |
| 2. Thực hiện bài toán với các mô hình học máy | 37 |
| PHẦN 3 – KẾT LUẬN..... | 69 |
| ❖ Kiến thức sau nghiên cứu, tìm hiểu | 69 |

| | | |
|---|---------------------------|----|
| ❖ | Những điều cần lưu ý..... | 69 |
| | TÀI LIỆU THAM KHẢO..... | 71 |

MỤC LỤC BẢNG BIỂU, HÌNH VẼ

| | |
|---|----|
| Hình 1 Dữ liệu chung của tệp application_record.csv | 29 |
| Hình 2 Dữ liệu chung của tệp credit_record.csv | 30 |
| Hình 3 Thống kê số lượng khách hàng | 31 |
| Hình 4 Số liệu giới tính và sở hữu ô tô | 31 |
| Hình 5 Số liệu sở hữu tài sản | 31 |
| Hình 6 Thống kê số lượng trẻ em | 32 |
| Hình 7 Thống kê thu nhập hàng năm | 32 |
| Hình 8 Thống kê loại thu nhập | 32 |
| Hình 9 Thống kê trình độ học vấn | 33 |
| Hình 10 Thống kê số lượng khách hàng | 33 |
| Hình 11 Thống kê theo tháng | 33 |
| Hình 12 Thống kê theo trạng thái ghi nợ | 34 |
| Hình 13 Kết quả học mô hình Logistic Regression | 55 |
| Hình 14 Kết quả học mô hình Decision Tree | 56 |
| Hình 15 Kết quả học mô hình Random Forests | 58 |
| Hình 16 Kết quả học với KNN | 60 |
| Hình 17 Mô hình Random forest phòng tránh Overfitting | 63 |
| Hình 18 Tạo mô hình FNN có sử dụng tránh Overfitting | 64 |
| Hình 19 Tạo mô hình RNN có sử dụng tránh Overfitting | 64 |

PHẦN 1 – CÂU 1 : NGHIÊN CỨU CÁ NHÂN

1. Tìm hiểu, so sánh các phương pháp Optimizer

Khái niệm: Thuật toán tối ưu (Optimizer) trong học máy là một phần quan trọng của quá trình huấn luyện mô hình. Nhiệm vụ chính của thuật toán tối ưu là tìm ra giá trị tối ưu của các tham số mô hình (như trọng số và bias) bằng cách cập nhật chúng dựa trên gradient của hàm mất mát. Thuật toán tối ưu xác định cách thức cập nhật trọng số để hàm mất mát giảm dần và mô hình hội tụ tới giải pháp tối ưu. Mỗi thuật toán tối ưu có cách tiếp cận riêng để điều chỉnh tốc độ học, xử lý gradient, và quyết định cách thức cập nhật trọng số.

Optimizer trong lĩnh vực Machine Learning là một công cụ quan trọng để tối ưu hóa mô hình. Công việc chính của optimizer là điều chỉnh các thông số của mô hình để giảm thiểu hoặc tối ưu hóa hàm mất mát, từ đó cải thiện hiệu suất của mô hình.

Dưới đây là một số ứng dụng chính của optimizer:

- **Tối ưu hóa mô hình:** Optimizer được sử dụng để điều chỉnh các tham số của mô hình máy học để mô hình có thể học được từ dữ liệu huấn luyện và dự đoán tốt trên dữ liệu mới.
- **Học máy sâu:** Trong các mô hình học sâu như mạng nơ-ron sâu (deep neural networks), optimizer giúp điều chỉnh hàng trăm hoặc hàng nghìn tham số mô hình để tối ưu hóa hiệu suất.
- **Cải thiện hàm mất mát:** Một số optimizer như Adam, SGD (Stochastic Gradient Descent), RMSprop, ... được thiết kế để tối ưu hóa hàm mất mát, giúp mô hình học nhanh hơn và tránh các điểm tối ưu cục bộ.

Tại sao lại cần phải sử dụng optimizer?

- **Tốc độ học tập:** Optimizer giúp mô hình học từ dữ liệu một cách hiệu quả và nhanh chóng hơn, giảm thiểu thời gian huấn luyện.
- **Tránh trầm cảm cục bộ:** Một optimizer tốt có thể giúp mô hình tránh rơi vào các điểm tối ưu cục bộ và đi đến điểm tối ưu toàn cục tốt nhất.
- **Điều chỉnh tham số:** Nó giúp điều chỉnh tỷ lệ học tập, quy mô gradient và các tham số quan trọng khác, làm cho mô hình học tốt hơn.

1.1 Các phương pháp Optimizer trong huấn luyện mô hình học máy

1.1.1 Gradient Descent

Thuật toán Gradient Descent là một trong những phương pháp cơ bản và quan trọng nhất trong tối ưu hóa trong Machine Learning. Nó được sử dụng để tìm giá trị nhỏ nhất của một hàm mất mát bằng cách điều chỉnh các tham số của mô hình dựa trên gradient của hàm này.

- Cách hoạt động của Gradient Descent:

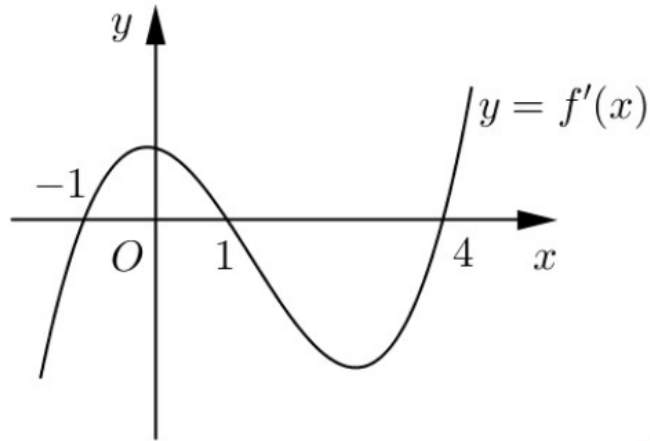
- Khởi tạo: Bắt đầu từ một điểm bất kỳ trong không gian tham số của mô hình .
- Tính gradient: Tính toán gradient của hàm mất mát (đạo hàm riêng theo từng tham số) dựa trên toàn bộ tập dữ liệu huấn luyện. Gradient được tính bằng cách sử dụng công thức đạo hàm của hàm mất mát tại điểm hiện tại.
- Điều chỉnh tham số: Di chuyển theo hướng ngược với gradient để giảm thiểu giá trị của hàm mất mát. Điều này có nghĩa là cập nhật các tham số theo hướng mà hàm mất mát giảm nhanh nhất.

CÔNG THỨC:

$$\text{tham số mới} = \text{tham số cũ} - \text{tỷ lệ học tập} \times \text{gradient}$$

Trong đó:

- + **Tham số mới:** Là giá trị mới của tham số .
 - + **Tham số cũ:** Là giá trị hiện tại của tham số.
 - + **Tỷ lệ học tập (Learning rate):** Là một siêu tham số quan trọng quyết định độ lớn của bước di chuyển mỗi khi cập nhật tham số. Điều này ảnh hưởng đến tốc độ học của mô hình. Nếu tỷ lệ học tập quá lớn, có thể dẫn đến việc vượt qua điểm tối ưu mong muốn hoặc dao động không ổn định; nếu quá nhỏ, có thể làm chậm quá trình học hoặc mắc kẹt ở điểm tối ưu cục bộ.
 - + **Gradient:** Là đạo hàm của hàm mất mát theo từng tham số. Gradient này xác định hướng và độ lớn mà hàm mất mát tăng nhanh nhất tại điểm hiện tại. Điều chỉnh tham số theo hướng ngược với gradient giúp giảm thiểu hàm mất mát.
 - Lặp lại quá trình: Lặp lại quá trình tính gradient và điều chỉnh tham số cho đến khi đạt được điều kiện dừng (ví dụ: số lần lặp, sự hội tụ, hoặc độ lớn gradient nhỏ hơn một ngưỡng nhất định).
- Áp dụng cho hàm 1 biến và hàm nhiều biến
 - a/ Gradient Descent cho hàm 1 biến



- Để tìm điểm cực tiểu của hàm số này, chúng ta có thể sử dụng thuật toán Gradient Descent.
- Ý tưởng của thuật toán Gradient Descent là bắt đầu từ một điểm x_0 bất kỳ, sau đó lặp đi lặp lại tiến một bước theo hướng ngược với gradient của hàm số tại điểm đó.
- Trong trường hợp này, gradient của hàm số là đường tiếp tuyến của hàm số tại điểm đó.

Vì vậy, bước lặp của thuật toán Gradient Descent được tính như sau:

$$x_{t+1} = x_t - \eta * \nabla f(x_t)$$

Trong đó:

- x_t là điểm tìm được sau vòng lặp thứ t
 - η là tốc độ học (learning rate)
 - $\nabla f(x_t)$ là gradient của hàm số $f(x)$ tại điểm x_t
- Với mỗi vòng lặp, điểm x_t sẽ di chuyển một bước theo hướng ngược với gradient của hàm số. Điều này sẽ giúp hàm số $f(x)$ giảm dần.
 - Nếu tốc độ học η được chọn hợp lý, thì thuật toán Gradient Descent sẽ hội tụ về điểm cực tiểu của hàm số.
 - Trong ví dụ này, chúng ta có thể chọn điểm x_0 là 1.0.

- Với tốc độ học $\eta = 0.1$, các bước lặp của thuật toán Gradient Descent được thể hiện trong bảng dưới đây:

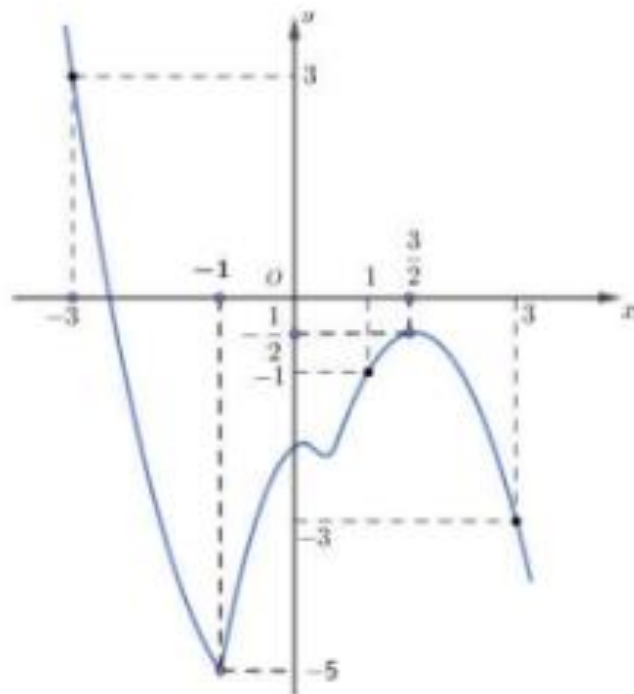
| Vòng lặp | x_t | $\nabla f(x_t)$ |
|----------|-------|-----------------|
| 0 | 1.0 | 1.0 |
| 1 | 0.9 | 0.9 |
| 2 | 0.81 | 0.81 |
| 3 | | 729 |

⇒ Như vậy, sau 5 vòng lặp, thuật toán Gradient Descent đã hội tụ về điểm cực tiểu của hàm số, $x^* = 0.0$.

b/ Gradient Descent cho hàm nhiều biến

Trong trường hợp hàm số có nhiều biến, thuật toán Gradient Descent cũng được áp dụng tương tự. Tuy nhiên, thay vì tính gradient của hàm số tại một điểm, chúng ta cần tính gradient của hàm số tại một vector.

Ví dụ, giả sử chúng ta có hàm số $y = f(x_1, x_2)$ như hình dưới đây:



Để tìm điểm cực tiểu của hàm số này, chúng ta có thể sử dụng thuật toán Gradient Descent như sau:

$$x_{t+1} = x_t - \eta * \nabla f(x_t)$$

Trong đó:

- x_t là vector tìm được sau vòng lặp thứ t
- η là tốc độ học (learning rate)
- $\nabla f(x_t)$ là gradient của hàm số $f(x_1, x_2)$ tại vector x_t

Gradient của hàm số $f(x_1, x_2)$ được tính như sau:

$$\nabla f(x_t) = [\nabla f_1(x_t), \nabla f_2(x_t)]$$

Trong đó:

- $\nabla f_1(x_t)$ là gradient của hàm số $f(x_1, x_2)$ theo biến x_1
- $\nabla f_2(x_t)$ là gradient của hàm số $f(x_1, x_2)$ theo biến x_2

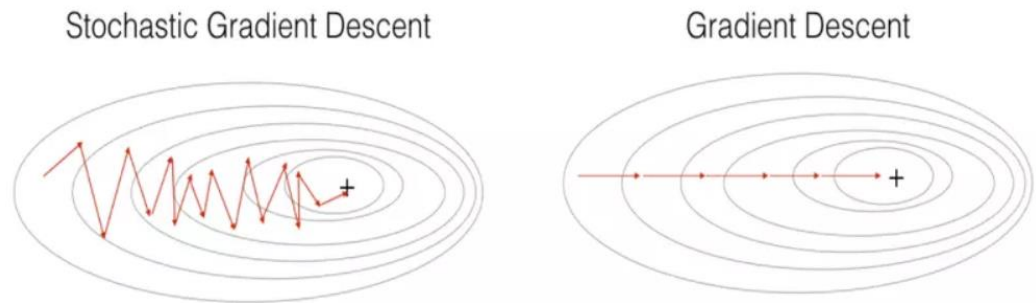
Với mỗi vòng lặp, vector x_t sẽ di chuyển một bước theo hướng ngược với gradient của hàm số. Điều này sẽ giúp hàm số $f(x_1, x_2)$ giảm dần.

⇒ Nếu tốc độ học η được chọn hợp lý, thì thuật toán Gradient Descent sẽ hội tụ về điểm cực tiểu của hàm số.

- Các biến thể của Gradient Descent
 - Batch Gradient Descent: Tính gradient dựa trên toàn bộ tập dữ liệu huấn luyện. Nó có thể chậm với tập dữ liệu lớn vì việc tính toán gradient cho toàn bộ dữ liệu.
 - Stochastic Gradient Descent (SGD): Tính gradient dựa trên một điểm dữ liệu duy nhất ngẫu nhiên trong mỗi lần cập nhật. Nhanh hơn và thích hợp với dữ liệu lớn, nhưng có thể không ổn định hơn do độ nhiễu từ từng điểm dữ liệu.
 - Mini-batch Gradient Descent: Kết hợp cả hai phương pháp trên bằng việc tính gradient trên một lượng nhỏ dữ liệu (mini-batch) được chọn ngẫu nhiên từ tập dữ liệu.

1.1.2 Stochastic Gradient Descent

SGD là một biến thể của thuật toán Gradient Descent (GD) truyền thống. GD sử dụng toàn bộ tập dữ liệu huấn luyện để tính toán gradient của hàm mất mát tại mỗi bước cập nhật. SGD khắc phục vấn đề này bằng cách chỉ sử dụng một mẫu nhỏ ngẫu nhiên (batch) được chọn từ tập dữ liệu huấn luyện để tính toán gradient. Bằng cách lặp lại quá trình này với các batch khác nhau, SGD dần dần di chuyển các tham số của mô hình theo hướng giảm thiểu hàm mất mát tổng thể.



Cách hoạt động của SGD:

- Khởi tạo: Bắt đầu từ một điểm khởi tạo ngẫu nhiên hoặc được chọn trước định sẵn trong không gian các tham số của mô hình.
- Lặp lại quá trình: Đối với mỗi điểm dữ liệu hoặc mini-batch:
 - Tính gradient: Tính toán gradient của hàm mất mát chỉ dựa trên điểm dữ liệu này.
 - Điều chỉnh tham số: Cập nhật các tham số của mô hình theo hướng ngược với gradient và với một tỷ lệ học tập đã chọn trước:

$$\text{tham số mới} = \text{tham số cũ} - \text{tỷ lệ học tập} \times \text{gradient}$$

- Lặp lại cho đến khi điều kiện dừng được đáp ứng: Quá trình này có thể dừng sau một số lần lặp cố định, hoặc khi đạt được điều kiện dừng khác, chẳng hạn như sự hội tụ của gradient hoặc đạt được giá trị mất mát mong muốn.

1.1.3 Momentum

Trong ngữ cảnh của tối ưu hóa mô hình máy học, Momentum là một kỹ thuật được áp dụng để cải thiện quá trình hội tụ của thuật toán tối ưu hóa như Gradient Descent (GD) hoặc Stochastic Gradient Descent (SGD).

Trong quá trình tối ưu hóa, Gradient Descent thường có thể di chuyển chậm hoặc bị rơi vào "vùng phẳng" của hàm mất mát, nơi gradient rất nhỏ. Momentum giúp tăng tốc độ di chuyển qua các vùng phẳng hoặc vùng địa phương bằng cách tích lũy một "momentum" từ gradient của các bước trước đó.

Cách hoạt động của Momentum

- Tính toán gradient: Tính gradient của hàm mất mát tại điểm hiện tại.
- Tích lũy momentum: Sử dụng gradient hiện tại và kết hợp với momentum từ các bước trước đó theo một tỷ lệ được gọi là hệ số momentum (thường là một giá trị từ 0 đến 1).
- Điều chỉnh tham số: Cập nhật các tham số của mô hình bằng việc di chuyển theo hướng tích lũy của momentum.

Công thức của Momentum được tính như sau:

$$\begin{aligned}v(t+1) &= \beta v(t) + \eta g(x(t)) \\x(t+1) &= x(t) + v(t+1)\end{aligned}$$

Trong đó:

- $v(t)$ là biến momentum tại thời điểm t
- β là hệ số momentum, thường được đặt là 0,9 hoặc 0,99
- η là tốc độ học
- $g(x(t))$ là gradient của hàm mất mát tại điểm $x(t)$
- $x(t)$ là tham số của mô hình tại thời điểm t

1.1.4 Adagrad

Không giống như các thuật toán trước đó thì learning rate hầu như giống nhau trong quá trình training (learning rate là hằng số), Adagrad coi learning rate là 1 tham số. Tức là Adagrad sẽ cho learning rate biến thiên sau mỗi thời điểm t .

Adagrad là một kỹ thuật học máy tiên tiến, thực hiện giảm dần độ dốc bằng cách thay đổi tốc độ học tập. Adagrad được cải thiện hơn bằng cách cho trọng số học tập chính xác dựa vào đầu vào trước nó để tự điều chỉnh tỉ lệ học theo hướng tối ưu nhất thay vì với một tỉ lệ học duy nhất cho tất cả các nút.

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \cdot g_t$$

Trong công thức, G_t là ma trận đường chéo chứa bình phương của đạo hàm vectơ tham số tại vòng lặp t ; g_t là vectơ của độ dốc cho vị trí hiện tại và η là tỉ lệ học

1.1.5 RMSprop

RMSprop là một thuật toán tối ưu hóa được sử dụng để huấn luyện các mô hình học máy, đặc biệt là các mô hình mạng nơ ron nhân tạo.

RMSprop là một biến thể của thuật toán Adagrad, trong đó tổng bình phương của gradient được làm mờ theo thời gian. Điều này giúp RMSprop tránh được hiện tượng quá nhạy của Adagrad đối với các tham số có gradient nhỏ.

Giống như các thuật toán giảm độ dốc khác, RMSprop hoạt động bằng cách tính toán độ dốc của hàm mất mát theo các tham số của mô hình và cập nhật các tham số theo hướng ngược lại với độ dốc để giảm thiểu tổn thất. Tuy nhiên, RMSProp giới thiệu một số kỹ thuật bổ sung để cải thiện hiệu suất của quá trình tối ưu hóa.

Một tính năng chính là việc sử dụng đường trung bình động của gradient bình phương để chia tỉ lệ học tập cho từng tham số. Điều này giúp ổn định quá trình học và ngăn chặn sự dao động trong quỹ đạo tối ưu hóa.

Thuật toán có thể được tóm tắt bằng công thức RMSProp sau:

$$v_t = \text{decay_rate} * v_{t-1} + (1 - \text{decay_rate}) * \text{gradient}^2$$

$$\text{parameter} = \text{parameter} - \text{learning_rate} * \text{gradient} / (\text{sqrt}(v_t) + \text{epsilon})$$

Trong đó:

- v_t là đường trung bình động của bình phương gradient;
- decay_rate là siêu tham số kiểm soát tốc độ phân rã của đường trung bình động;
- learning_rate là siêu tham số kiểm soát kích thước bước cập nhật;
- gradient là gradient của hàm mất mát đối với tham số;
- epsilon là một hằng số nhỏ được thêm vào mẫu số để ngăn việc chia cho 0.

1.1.6 Adam

Adam được xem như là sự kết hợp của RMSprop và Stochastic Gradient Descent với động lượng. Adam là một phương pháp tỉ lệ học thích ứng, nó tính toán tỉ lệ học tập cá nhân cho các tham số khác nhau. Adam sử dụng ước tính của khoảng thời gian thứ nhất và thứ hai của độ dốc để điều chỉnh tỉ lệ học cho từng trọng số của mạng nơ-ron. Đánh giá các thuật toán tối ưu đối với mô hình mạng nơ-ron tích chập trong tác vụ nhận diện hình ảnh 76 Tuy nhiên, qua nghiên cứu thực nghiệm, trong một số trường hợp, Adam vẫn còn gặp phải nhiều thiếu sót so với thuật toán SGD. Thuật toán Adam được mô tả:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

Trong công thức (3.6), v_t là trung bình động của bình phương và m_t là trung bình động của gradient; β_1 và β_2 là tốc độ của di chuyển.

1.1.7 Adadelta

Adadelta là một biến thể khác của AdaGrad. Adadelta không có tham số tỉ lệ học. Thay vào đó, nó sử dụng tốc độ thay đổi của chính các tham số để điều chỉnh tỉ lệ học nghĩa là bằng cách giới hạn cửa sổ của gradient tích lũy trong quá khứ ở một số kích thước cố định của trọng số w .

$$g'_t = \sqrt{\frac{\Delta x_{t-1} + \epsilon}{s_t + \epsilon}} \cdot g_t$$

$$x_t = x_{t-1} - g'_t$$

$$\Delta x_t = \rho \Delta x_{t-1} + (1 - \rho) x_t^2$$

Từ công thức trên, Adadelta sử dụng 2 biến trạng thái: s_t để lưu trữ trung bình của khoảng thời gian thứ hai của gradient và Δx_t để lưu trữ trung bình của khoảng thời gian thứ 2 của sự thay đổi các tham số trong mô hình. g'_t : căn bậc hai thương của trung bình tốc độ thay đổi bình phương và trung bình mô-men bậc hai của gradient.

1.1.8 Nadam

Mô hình Nadam là một biến thể của thuật toán tối ưu hóa được sử dụng trong machine learning, kết hợp cả Adam (Adaptive Moment Estimation) và Nesterov Accelerated Gradient (NAG). Nadam kết hợp ưu điểm của cả hai phương pháp này để cải thiện quá trình hội tụ của mô hình học máy.

Cách hoạt động của Nadam

- Adam Optimization:
 - Sử dụng cả momentum và RMSprop để cập nhật gradient.
 - Tính toán gradient động (adaptive) cho từng tham số, kết hợp momentum để giảm độ dao động và RMSprop để điều chỉnh tỷ lệ học tập.
- Nesterov Accelerated Gradient (NAG):
 - Dự đoán điểm mà gradient sẽ đến sau một bước nhất định.
 - Tính toán gradient không chỉ dựa trên vị trí hiện tại mà còn dựa trên vị trí được dự đoán sau một bước.
- Kết hợp Adam và NAG:
 - Nadam kết hợp cả Adam và NAG bằng cách sử dụng cơ chế dự đoán của NAG để cập nhật tham số mô hình.
 - Điều này giúp Nadam kỹ thuật định hướng tốt hơn khi cập nhật tham số và giảm thiểu sự dao động, đặc biệt trong những khu vực có gradient ít ổn định.

1.2 So sánh các phương pháp Optimizer trong huấn luyện mô hình học máy

Điểm giống nhau của các phương pháp Optimizer:

- Cảm biến gradient: Tất cả các phương pháp Optimizer đều sử dụng gradient của hàm mất mát để cập nhật các tham số của mô hình. Gradient cho biết hướng mà hàm mất mát giảm nhanh nhất.
- Cập nhật lặp: Tất cả các phương pháp Optimizer đều cập nhật các tham số của mô hình lặp đi lặp lại. Mỗi lần cập nhật, các tham số được cập nhật theo một hướng nhỏ để giảm thiểu hàm mất mát.

Điểm khác biệt:

| Phương pháp | Mô tả | Ưu điểm | Nhược điểm |
|--|---|---|---|
| Gradient Descent | Cập nhật các tham số theo hướng giảm thiểu hàm mất mát | Đơn giản, hiệu quả | Có thể bị mắc kẹt ở các điểm tối thiểu cục bộ, cần tinh chỉnh tham số tốc độ học Chậm trên dữ liệu lớn |
| Stochastic Gradient Descent (SGD) | Sử dụng một mẫu nhỏ ngẫu nhiên để tính toán gradient | Hiệu quả hơn Gradient Descent với dữ liệu lớn, ổn định hơn Gradient Descent | Có thể bị dao động, cần tinh chỉnh tham số kích thước batch |
| Momentum | Sử dụng một biến momentum để lưu lại hướng của các bước cập nhật trước đó | Tăng độ ổn định, tăng tốc độ hội tụ | Cần tinh chỉnh tham số momentum |

| | | | |
|------------------|--|---|---|
| Adagrad | Tốc độ học của SGD được điều chỉnh dựa trên độ lớn của gradient | Tăng tốc độ hội tụ, giảm thiểu nguy cơ bị mắc kẹt | Có thể bị quá nhạy, cần tinh chỉnh tham số tốc độ học |
| RMSprop | Thay vì sử dụng tổng bình phương của gradient, RMSprop sử dụng trung bình di động của gradient | Ổn định hơn Adagrad, tăng tốc độ hội tụ | Cần tinh chỉnh tham số tốc độ học và hệ số giảm xóc |
| Nadam | Tích hợp Momentum vào RMSprop | Ổn định hơn RMSprop, tăng tốc độ hội tụ | Cần tinh chỉnh tham số tốc độ học, hệ số giảm xóc và momentum |
| Adam | Tích hợp Momentum và Adagrad | - Tích hợp cả momentum và RMSprop | - Cần điều chỉnh siêu tham số |
| Adadelata | Một biến thể của Adagrad, trong đó sử dụng trung bình di động của gradient | Ổn định hơn Adagrad, tăng tốc độ hội tụ | Có thể bị quá nhạy, cần tinh chỉnh tham số tốc độ học |

2. Tìm hiểu về Continual Learning và Test Production

2.1 Continual Learning

Continual Learning là một lĩnh vực của học máy tập trung vào việc xây dựng các mô hình có thể học hỏi và thích nghi với dữ liệu mới trong khi vẫn duy trì hiệu suất tốt trên dữ liệu cũ.

Trong thế giới thực, dữ liệu luôn thay đổi. Các mô hình học máy được đào tạo trên dữ liệu cũ có thể không thể dự đoán chính xác dữ liệu mới.

Continual Learning có thể giúp giải quyết vấn đề này bằng cách cho phép các mô hình học hỏi liên tục từ dữ liệu mới mà không làm giảm hiệu suất trên dữ liệu cũ.

Có nhiều kỹ thuật khác nhau có thể được sử dụng cho Continual Learning. Một số kỹ thuật phổ biến bao gồm:

- Forgetting prevention: Kỹ thuật này nhằm ngăn chặn mô hình quên dữ liệu cũ. Một số cách để thực hiện điều này bao gồm:
 - Data replay: Dữ liệu cũ được thêm vào tập dữ liệu mới để mô hình tiếp tục học hỏi từ nó.
 - Regularization: Các thuật toán regularizing có thể giúp mô hình tránh quên dữ liệu cũ.
- Incremental learning: Kỹ thuật này cho phép mô hình học hỏi từ dữ liệu mới mà không ảnh hưởng đến dữ liệu cũ. Một số cách để thực hiện điều này bao gồm:
 - Incremental gradient descent: Gradient của dữ liệu mới được thêm vào gradient của dữ liệu cũ để cập nhật các tham số của mô hình.
 - Elastic weight consolidation: Các tham số của mô hình được cố định một phần để tránh thay đổi quá nhiều.

Ưu điểm:

- Có thể giúp mô hình học hỏi và thích nghi với dữ liệu thay đổi
- Có thể cải thiện hiệu suất của mô hình theo thời gian
- Có thể giúp giảm thiểu rủi ro khi triển khai một mô hình học máy mới

Nhược điểm:

- Có thể phức tạp và khó thực hiện
- Có thể cần nhiều tài nguyên tính toán
- Có thể dẫn đến overfitting

2.2 Test Production

Test Production là một quá trình trong đó một mô hình học máy được triển khai trong môi trường sản xuất trong khi vẫn được giám sát và kiểm tra. Test

Production có thể giúp đảm bảo rằng mô hình học máy hoạt động chính xác trong môi trường sản xuất và có thể được điều chỉnh kịp thời nếu cần.

Có nhiều lợi ích của việc sử dụng Test Production, bao gồm:

- **Đảm bảo chất lượng:** Test Production có thể giúp đảm bảo rằng mô hình học máy hoạt động chính xác trong môi trường sản xuất.
- **Tăng cường học tập:** Test Production có thể giúp mô hình học máy học hỏi từ dữ liệu mới và cải thiện hiệu suất theo thời gian.
- **Giảm thiểu rủi ro:** Test Production có thể giúp giảm thiểu rủi ro khi triển khai một mô hình học máy mới.

Khi xây dựng một giải pháp học máy để giải quyết một bài toán nào đó, cần cân nhắc cả Continual Learning và Test Production. Continual Learning có thể giúp mô hình học máy thích ứng với dữ liệu thay đổi, trong khi Test Production có thể giúp đảm bảo rằng mô hình học máy hoạt động chính xác trong môi trường sản xuất.

Ưu điểm:

- Có thể giúp đảm bảo rằng mô hình học máy hoạt động chính xác trong môi trường sản xuất
- Có thể giúp mô hình học hỏi từ dữ liệu mới và cải thiện hiệu suất theo thời gian
- Có thể giúp giảm thiểu rủi ro khi triển khai một mô hình học máy mới

Nhược điểm:

- Có thể phức tạp và tốn kém
- Có thể cần nhiều tài nguyên tính toán
- Có thể dẫn đến overfitting

⇒ Kết hợp Continual Learning và Test Production là cách tiếp cận quan trọng để xây dựng và duy trì một mô hình học máy có khả năng học liên tục và tổng quát hóa tốt trên dữ liệu mới.

2.3 Kết hợp Continual Learning và Test Production giải quyết bài toán học máy

Để **xây dựng một phương pháp học máy**, chúng ta có thể tiếp cận với một quy trình dưới đây:

Bước 1: Thu thập và Chuẩn bị Dữ liệu

- Thu thập Dữ liệu: Thu thập dữ liệu từ nguồn phù hợp với bài toán cụ thể mà bạn muốn giải quyết.
- Tiền xử lý Dữ liệu: Chuẩn bị dữ liệu bằng cách loại bỏ nhiễu, xử lý dữ liệu bị thiếu, mã hóa các biến phân loại và thực hiện các bước tiền xử lý khác cần thiết.

Bước 2: Lựa chọn và Huấn luyện Mô hình

- Lựa chọn Mô hình: Chọn một mô hình phù hợp với bài toán của bạn. Điều này có thể là một mô hình học máy cơ bản như Linear Regression, Decision Trees, hoặc một mô hình phức tạp hơn như Neural Networks.
- Chia Dữ liệu: Chia dữ liệu thành tập huấn luyện và tập kiểm tra để đánh giá hiệu suất của mô hình.
- Huấn luyện Mô hình: Sử dụng tập huấn luyện để huấn luyện mô hình với các tham số tốt nhất.

Bước 3: Đánh giá và Tinh chỉnh Mô hình

- Đánh giá Mô hình: Đánh giá hiệu suất của mô hình trên tập kiểm tra bằng các độ đo thích hợp (accuracy, precision, recall, F1-score, etc.).
- Tinh chỉnh Tham số: Tối ưu hóa các tham số của mô hình để cải thiện hiệu suất.

Bước 4: Triển khai và Kiểm tra

- Triển khai Mô hình: Áp dụng mô hình đã huấn luyện vào môi trường thực tế.
- Kiểm tra và Đánh giá: Kiểm tra hiệu suất của mô hình trên dữ liệu thực tế để đảm bảo nó hoạt động hiệu quả.

Bước 5: Tối ưu hoá và Cải thiện

- Tối ưu hoá Mô hình: Liên tục cải thiện mô hình dựa trên phản hồi từ quá trình triển khai và kiểm tra thực tế.

- **Mở rộng và Scalability:** Nếu cần, mở rộng mô hình để xử lý dữ liệu lớn hơn hoặc mở rộng phạm vi ứng dụng.

Để kết hợp Continual Learning và Test Production, chúng ta cần thiết lập một quy trình làm việc liên tục và có cơ chế cập nhật đối với cả quá trình học và đánh giá. Dưới đây là các bước để kết hợp chúng:

Thiết lập Continuous Learning Loop:

- Chọn mô hình có khả năng học liên tục: Sử dụng mô hình học máy có khả năng học online hoặc có thể cập nhật liên tục từ dữ liệu mới.
- Sử dụng kỹ thuật Continual Learning:
 - Áp dụng kỹ thuật như Elastic Weight Consolidation (EWC), Synaptic Intelligence (SI) để bảo vệ và duy trì kiến thức đã học.
 - Sử dụng rehearsal hoặc buffer để lưu trữ và sử dụng lại dữ liệu cũ cho việc huấn luyện mô hình với dữ liệu mới.
- Đánh giá và Cập nhật thường xuyên:
 - Liên tục đánh giá hiệu suất của mô hình trên các bộ dữ liệu kiểm tra liên tục, bao gồm cả dữ liệu cũ và mới.
 - Cập nhật mô hình và kiến thức đã học dựa trên kết quả đánh giá.

Thiết lập Test Production:

- Tạo Continual Test Sets:
 - Tạo các bộ dữ liệu kiểm tra có tính liên tục, bao gồm dữ liệu mới và cũ, để đánh giá hiệu suất của mô hình theo thời gian.
 - Đảm bảo các bộ dữ liệu kiểm tra phản ánh thực tế và có thể thay đổi khi dữ liệu mới xuất hiện.
- Đánh giá và So sánh:
 - Sử dụng các phép đo hiệu suất phù hợp để đánh giá khả năng tổng quát hóa và hiệu suất của mô hình trên các bộ dữ liệu kiểm tra.
 - So sánh hiệu suất của mô hình trên các bộ dữ liệu kiểm tra với các tiêu chuẩn và theo dõi sự thay đổi của hiệu suất theo thời gian.

Tối ưu và Điều chỉnh:

- Cải thiện Continual Learning:
 - Điều chỉnh siêu tham số và kỹ thuật học liên tục để cải thiện hiệu suất và khả năng học của mô hình.
- Cải thiện Test Production:

- Liên tục cập nhật và cải thiện bộ dữ liệu kiểm tra để đảm bảo rằng nó phản ánh thực tế và thay đổi của dữ liệu mới.

⇒ Continual Learning và Test Production thường được tích hợp vào quy trình của bước cuối cùng, khi mô hình đã được huấn luyện và triển khai. Chúng giúp đảm bảo rằng mô hình không chỉ học từ dữ liệu mới mà còn duy trì khả năng tổng quát hóa và hiệu suất trên dữ liệu mới.

- Continual Learning:
 - Quá trình này thường xảy ra sau khi mô hình đã được huấn luyện. Liên tục học từ dữ liệu mới và duy trì kiến thức đã học trước đó mà không ghi đè hoặc quên đi thông tin quan trọng đã được học.
- Test Production:
 - Test Production được thực hiện để tạo ra các bộ dữ liệu kiểm tra liên tục, bao gồm cả dữ liệu mới và cũ, để đánh giá hiệu suất và khả năng tổng quát hóa của mô hình theo thời gian.

Những bước này giúp đảm bảo rằng mô hình không chỉ có khả năng học liên tục mà còn đánh giá và duy trì hiệu suất trên dữ liệu mới, giúp nó phản ánh thực tế và đáng tin cậy khi triển khai trong môi trường thực tế.

PHẦN 2 – CÂU 2 : XÂY DỰNG MÔ HÌNH HỌC MÁY – DỰ ĐOÁN PHÊ DUYỆT THẺ TÍN DỤNG TỰ ĐỘNG

1. Giới thiệu dữ liệu sử dụng cho bài toán

1.1 Tiếp dữ liệu: Credit Card Approval Prediction

❖ Lý do chọn đề tài:

Thẻ điểm tín dụng là một phương pháp kiểm soát rủi ro phổ biến trong ngành tài chính. Nó sử dụng thông tin và dữ liệu cá nhân do người đăng ký thẻ tín dụng gửi để dự đoán khả năng vỡ nợ và vay thẻ tín dụng trong tương lai. Ngân hàng có thể quyết định có cấp thẻ tín dụng cho người nộp đơn hay không. Điểm tín dụng có thể định lượng một cách khách quan mức độ rủi ro.

Nói chung, thẻ điểm tín dụng dựa trên dữ liệu lịch sử. Một khi gặp phải những biến động lớn về kinh tế. Các mô hình trong quá khứ có thể mất đi khả năng dự đoán ban đầu. Mô hình logistic là một phương pháp phổ biến để chấm điểm tín dụng. Bởi vì Logistic phù hợp với các nhiệm vụ phân loại nhị phân và có thể tính toán các hệ số của từng đặc điểm. Để thuận tiện cho việc hiểu và vận hành, thẻ điểm sẽ nhân hệ số hồi quy logistic với một giá trị nhất định (chẳng hạn như 100) và làm tròn số đó.

Hiện nay với sự phát triển của các thuật toán học máy. Các phương pháp dự đoán khác như Tăng cường, Rừng ngẫu nhiên và Máy vector hỗ trợ đã được đưa vào tính điểm thẻ tín dụng. Tuy nhiên, những phương pháp này thường không có tính minh bạch tốt. Có thể khó đưa ra lý do từ chối hoặc chấp nhận cho khách hàng và cơ quan quản lý.

❖ Mô tả

Tập dữ liệu "Credit Card Approval Prediction" là một tập dữ liệu phổ biến trong lĩnh vực học máy và dự đoán tín dụng. Tập dữ liệu này thường được sử dụng để dự đoán xem một đơn vay mượn thẻ tín dụng sẽ được phê duyệt hay từ chối dựa trên các đặc trưng khách hàng và lịch sử tài chính.

Thông thường, tập dữ liệu này chứa các thông tin như:

- Đặc trưng của khách hàng: Như tuổi, giới tính, thu nhập, nghề nghiệp, địa chỉ, học vấn, số lượng người phụ thuộc, v.v.
- Thông tin tài chính: Như số lượng thẻ tín dụng hiện có, lịch sử thanh toán, tỷ lệ nợ, điểm tín dụng, số tiền vay muốn, mục đích vay, v.v.
- Nhãn: Ghi chú về việc đơn vay mượn được phê duyệt (1) hoặc từ chối (0)

❖ **Nội dung và giải thích:**

Có hai bảng được kết nối bằng ID

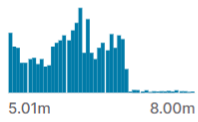
- application_record.csv chứa thông tin cá nhân của người đăng ký mà bạn có thể sử dụng làm tính năng dự đoán.
- credit_record.csv ghi lại hành vi sử dụng thẻ tín dụng của người dùng.


| application_record.csv | | |
|------------------------|----------------------|-----------|
| Tên tính năng | Giải trình | Bình luận |
| ID | Số lượng khách hàng | |
| CODE_GENDER | Giới tính | |
| FLAG_OWN_CAR | Có ô tô không | |
| FLAG_OWN_REALTY | Có tài sản nào không | |
| CNT_CHILDREN | số lượng trẻ em | |
| AMT_INCOME_TOTAL | Thu nhập hàng năm | |
| NAME_INCOME_TYPE | Loại thu nhập | |
| NAME_EDUCATION_TYPE | Trình độ học vấn | |

| | | |
|--------------------|-----------------------------|---|
| NAME_FAMILY_STATUS | Trình trạng hôn nhân | |
| NAME_HOUSING_TYPE | Cách sống | |
| DAYS_BIRTH | Sinh nhật | Đếm ngược từ ngày hiện tại (0), -1 nghĩa là ngày hôm qua |
| DAYS_EMPLOYED | Ngày bắt đầu làm việc | Đếm ngược từ ngày hiện tại (0). Nếu tích cực, nó có nghĩa là người hiện đang thất nghiệp. |
| FLAG_MOBIL | Có điện thoại di động không | |
| FLAG_WORK_PHONE | Có điện thoại cơ quan không | |
| FLAG_PHONE | Có điện thoại không | |
| FLAG_EMAIL | Có email không | |
| OCCUPATION_TYPE | Nghề nghiệp | |
| CNT_FAM_MEMBERS | Quy mô gia đình | |

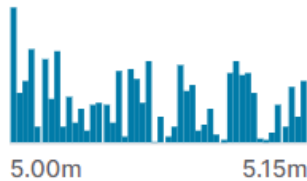
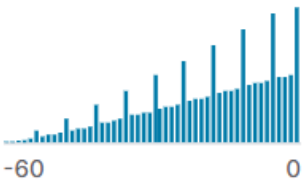
| | | |
|-------------------|------------|-----------|
| credit_record.csv | | |
| Tên tính năng | Giải trình | Bình luận |

| ID | Số lượng khách hàng | |
|----------------|---------------------|---|
| MONTHS_BALANCE | Ghi tháng | Tháng của dữ liệu được trích xuất là điểm bắt đầu, ngược lại, 0 là tháng hiện tại, -1 là tháng trước đó, v.v. |
| TATUS | Trạng thái | 0: Quá hạn 1-29 ngày 1: Quá hạn 30-59 ngày 2: Quá hạn 60-89 ngày 3: Quá hạn 90-119 ngày 4: Quá hạn 120-149 ngày 5: Nợ quá hạn hoặc nợ xấu, xóa nợ trên 150 ngày C: trả hết tháng đó X: Không vay trong tháng |

| application_record.csv (54.34 MB) | | | | | ↓ ↗ > |
|---|----------------|-------------------------|-------------------------|---|------------------|
| Detail Compact Column | | | | | 10 of 18 columns |
| ID | CODE_GENDER | FLAG_OWN_CAR | FLAG_OWN_REAL... | CNT_CHILDREN | |
| client number | gender | if users have a car | Is there a property | Number of children | |
|  | F 67% M 33% | true 0 0% false 0 0% | true 0 0% false 0 0% |  | |
| 5.01m 8.00m | | | | 0 19 | |
| 5008804 | M | Y | Y | 0 | |
| 5008805 | M | Y | Y | 0 | |
| 5008806 | M | Y | Y | 0 | |
| 5008808 | F | N | Y | 0 | |
| 5008809 | F | N | Y | 0 | |
| 5008810 | F | N | Y | 0 | |

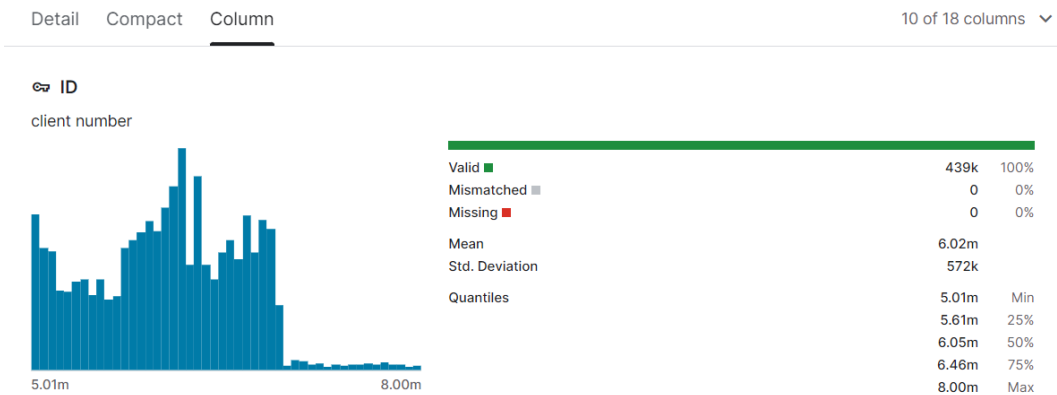
| application_record.csv (54.34 MB) | | | | | ↓ ↗ > |
|---|--|---|--|--|------------------|
| Detail Compact Column | | | | | 10 of 18 columns |
| # AMT_INCOME_TO... | NAME_INCOME_T... | NAME_EDUCATIO... | NAME_FAMILY_S... | NAME_HOUSING... | |
| Annual income | Income category | education level | Marital status | Way of living | |
|  | Working 52% Commercial assoc... 23% Other (111696) 25% | Secondary / seco... 69% Higher education 27% Other (19214) 4% | Married 68% Single / not married 13% Other (83458) 19% | House / apartment 90% With parents 4% Other (25649) 6% | |
| 19 26.1k 6.75m | | | | | |
| 427500.0 | Working | Higher education | Civil marriage | Rented apartment | |
| 427500.0 | Working | Higher education | Civil marriage | Rented apartment | |
| 112500.0 | Working | Secondary / secondary special | Married | House / apartment | |
| 270000.0 | Commercial associate | Secondary / secondary special | Single / not married | House / apartment | |
| 270000.0 | Commercial associate | Secondary / secondary special | Single / not married | House / apartment | |
| 270000.0 | Commercial associate | Secondary / secondary special | Single / not married | House / apartment | |

Hình 1 Dữ liệu chung của tệp application_record.csv

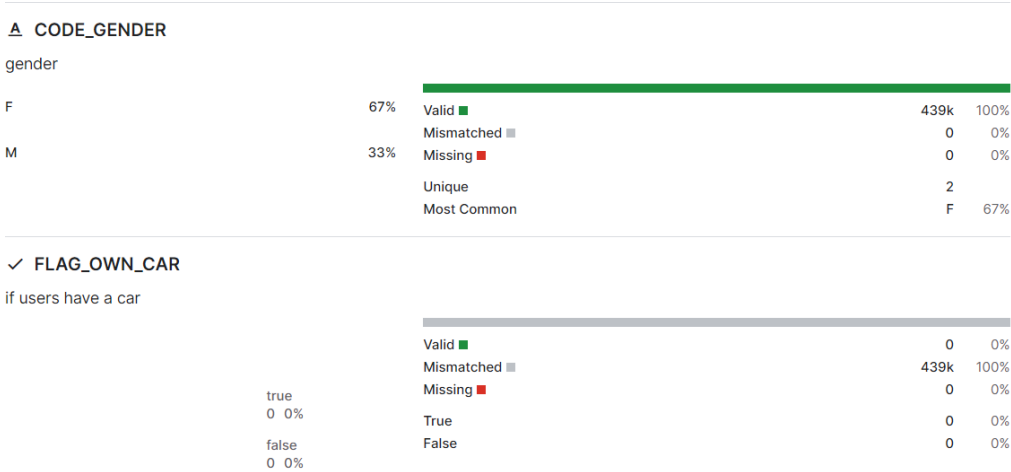
| credit_record.csv (15.37 MB) | | |
|---|--|--|
| Detail | Compact | Column |
| <div> <div>ID</div> <div>ID</div> </div> | <div> <div># MONTHS_BALAN...</div> <div>record month: The month of the extracted data is the starting point, backwards, 0 is the current month, -1 is the</div> </div> | <div> <div>STATUS</div> <div>0: 1-29 days past due 1: 30-59 days past due 2: 60-89 days overdue 3: 90-119 days overdue 4: 120-149 days overdue 5:</div> </div> |
|  |  | <div> <div>C 42%</div> <div>0 37%</div> <div>Other (223424) 21%</div> </div> |
| 5001711 | 0 | X |
| 5001711 | -1 | 0 |
| 5001711 | -2 | 0 |
| 5001711 | -3 | 0 |
| 5001712 | 0 | C |

Hình 2 Dữ liệu chung của tệp credit_record.csv

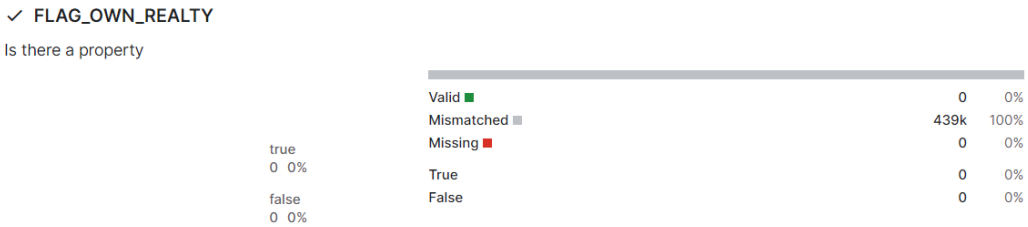
Các dữ liệu thống kê của application_record được thể hiện trong các hình sau:



Hình 3 Thống kê số lượng khách hàng



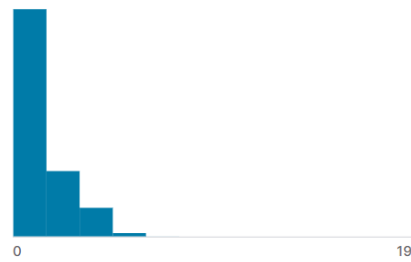
Hình 4 Số liệu giới tính và sở hữu ô tô



Hình 5 Số liệu sở hữu tài sản

CNT_CHILDREN

Number of children

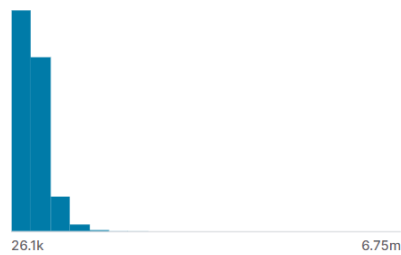


| | | |
|----------------|------|------|
| Valid | 439k | 100% |
| Mismatched | 0 | 0% |
| Missing | 0 | 0% |
| Mean | 0.43 | |
| Std. Deviation | 0.72 | |
| Quantiles | | |
| | 0 | Min |
| | 0 | 25% |
| | 0 | 50% |
| | 1 | 75% |
| | 19 | Max |

Hình 6 Thống kê số lượng trẻ em

AMT_INCOME_TOTAL

Annual income



| | | |
|----------------|-------|------|
| Valid | 439k | 100% |
| Mismatched | 0 | 0% |
| Missing | 0 | 0% |
| Mean | 188k | |
| Std. Deviation | 110k | |
| Quantiles | | |
| | 26.1k | Min |
| | 122k | 25% |
| | 161k | 50% |
| | 225k | 75% |
| | 6.75m | Max |

Hình 7 Thống kê thu nhập hàng năm

A NAME_INCOME_TYPE

Income category

| | | | | |
|----------------------|-----|-------------|---------|------|
| Working | 52% | Valid | 439k | 100% |
| | | Mismatched | 0 | 0% |
| | | Missing | 0 | 0% |
| Commercial associate | 23% | Unique | 5 | |
| Other (111696) | 25% | Most Common | Working | 52% |

A NAME_EDUCATION_TYPE

education level

| | | | | |
|-------------------------------|-----|-------------|-----------------|------|
| Secondary / secondary special | 69% | Valid | 439k | 100% |
| | | Mismatched | 0 | 0% |
| | | Missing | 0 | 0% |
| Higher education | 27% | Unique | 5 | |
| Other (19214) | 4% | Most Common | Secondary / ... | 69% |

Hình 8 Thống kê loại thu nhập

A NAME_FAMILY_STATUS

Marital status

| | | | | |
|----------------------|-----|-------------|---------|------|
| Married | 68% | Valid | 439k | 100% |
| | | Mismatched | 0 | 0% |
| Single / not married | 13% | Missing | 0 | 0% |
| Other (83458) | 19% | Unique | 5 | |
| | | Most Common | Married | 68% |

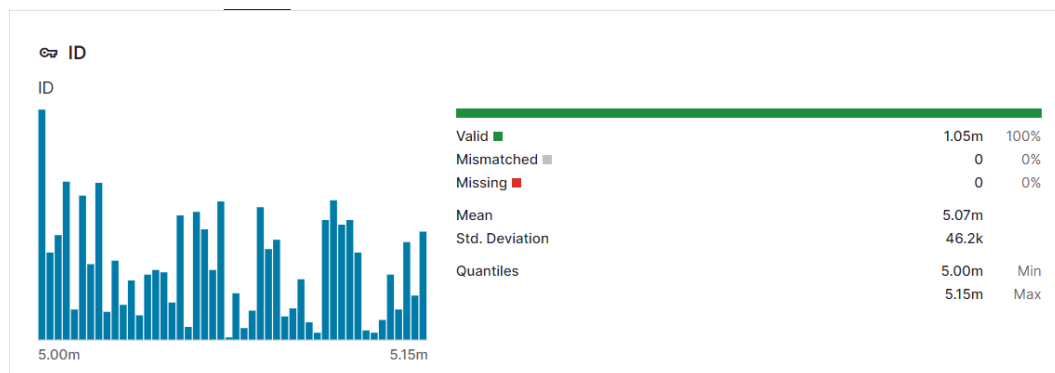
A NAME_HOUSING_TYPE

Way of living

| | | | | |
|-------------------|-----|-------------|----------------|------|
| House / apartment | 90% | Valid | 439k | 100% |
| | | Mismatched | 0 | 0% |
| With parents | 4% | Missing | 0 | 0% |
| Other (25649) | 6% | Unique | 6 | |
| | | Most Common | House / apa... | 90% |

Hình 9 Thống kê trình độ học vấn

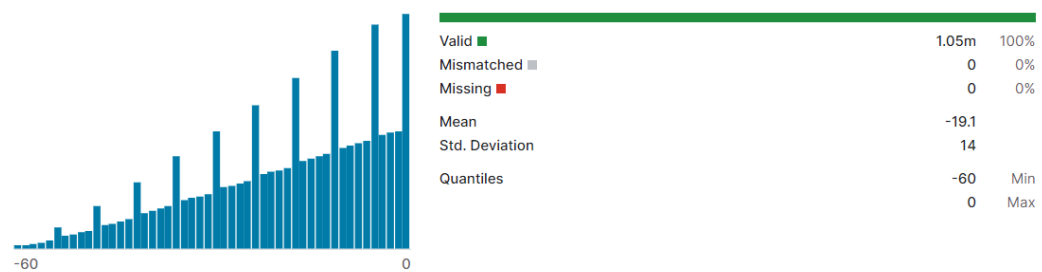
Các dữ liệu thống kê của credit_record được thể hiện qua các biểu đồ dưới đây:



Hình 10 Thống kê số lượng khách hàng

MONTHS_BALANCE

record month: The month of the extracted data is the starting point, backwards, 0 is the current month, -1 is the previous month, and so on



Hình 11 Thống kê theo tháng

A STATUS

0: 1-29 days past due 1: 30-59 days past due 2: 60-89 days overdue 3: 90-119 days overdue 4: 120-149 days overdue 5: Overdue or bad debts, write-offs for more than 150 days C: paid off that month X: No loan for the month

| | | | | |
|----------------|-----|--------------|-------|------|
| C | 42% | Valid ■ | 1.05m | 100% |
| | | Mismatched ■ | 0 | 0% |
| 0 | 37% | Missing ■ | 0 | 0% |
| Other (223424) | 21% | Unique | 8 | |
| | | Most Common | C | 42% |

Hình 12 Thống kê theo trạng thái ghi nợ

1.2 Đánh giá các feature của dữ liệu

Dữ liệu: Credit Card Approval Prediction

Để đánh giá vai trò của các cột trong dữ liệu đối với bài toán Dự đoán Phê duyệt thẻ tín dụng tự động, bạn có thể tập trung vào các cột có thể ảnh hưởng đến quyết định phê duyệt của hệ thống. Dưới đây là một số cột quan trọng mà bạn có thể xem xét:

- **AMT_INCOME_TOTAL** (Thu nhập hằng năm): Số tiền thu nhập hàng năm của khách hàng có thể là một yếu tố quan trọng để đánh giá khả năng thanh toán và đáng giá để cân nhắc trong quyết định phê duyệt.
- **NAME_EDUCATION_TYPE** (Loại hình giáo dục): Trình độ giáo dục của khách hàng có thể ảnh hưởng đến khả năng thanh toán và sự ổn định tài chính, có thể là một yếu tố quan trọng.
- **NAME_FAMILY_STATUS** (Tình trạng hôn nhân): Tình trạng hôn nhân của khách hàng có thể ảnh hưởng đến ổn định gia đình và khả năng chi trả nghĩa vụ tài chính.
- **DAYS_BIRTH** (Tuổi): Độ tuổi của khách hàng có thể đóng vai trò trong đánh giá tính ổn định tài chính và khả năng thanh toán.
- **DAYS_EMPLOYED** (Thời gian làm việc):
- Thời gian làm việc của khách hàng có thể là một yếu tố quan trọng, ảnh hưởng đến khả năng chi trả và ổn định tài chính.

- FLAG_OWN_CAR (Có xe ô tô):Việc sở hữu ô tô có thể là một yếu tố ảnh hưởng đến khả năng chi trả và tình trạng tài chính tổng thể.
 - FLAG_OWN_REALTY (Có bất động sản):Việc sở hữu bất động sản cũng có thể là một chỉ số cho sự ổn định tài chính và đáng giá để xem xét.
- ==> Từ đó chúng ta có thể xác định các feature quan trọng của dữ liệu để xây dựng mô hình học máy giải quyết bài toán.

2. Xây dựng mô hình học máy

2.1 Các mô hình sử dụng

- ❖ Decision Tree (Cây Quyết định):
 - Ưu Điểm:Dễ hiểu và trực quan: Cây quyết định có thể được biểu diễn dễ hiểu, giúp giải thích quyết định của mô hình một cách rõ ràng.
 - Khả năng xử lý dữ liệu phi cấp: Cây quyết định có thể xử lý dữ liệu không cần phải chia thành các phần tử cấp.
 - Ứng Dụng:Phù hợp cho việc tìm hiểu cấu trúc quyết định và giảm độ phức tạp của mô hình.
- ❖ K-Nearest Neighbors (KNN):
 - Ưu Điểm:Đơn giản và dễ triển khai: KNN không đòi hỏi giả định về phân phối dữ liệu và là một mô hình dễ triển khai.
 - Ứng Dụng:Hiệu quả cho các bài toán phân loại đơn giản và khi dữ liệu có cấu trúc lân cận.
- ❖ Gradient Boosting:
 - Ưu Điểm: Hiệu suất cao: Gradient Boosting thường cho kết quả rất tốt và có khả năng xử lý các tập dữ liệu lớn.
 - Ứng Dụng: Phù hợp cho các bài toán phức tạp, đặc biệt là khi có sự tương tác phức tạp giữa các đặc trưng.
- ❖ Random Forest:
 - Ưu Điểm: Ổn định và chống overfitting: Random Forest có khả năng giảm nguy cơ overfitting do sự đa dạng của các cây quyết định.
 - Ứng Dụng: Hiệu quả trong việc xử lý các tập dữ liệu lớn và nhiễu.
- ❖ Logistic Regression:
 - Ưu Điểm: Dễ hiểu và dễ triển khai: Logistic Regression là mô hình đơn giản nhưng mạnh mẽ, thường được sử dụng như một điểm xuất phát cho các mô hình phức tạp hơn.

- Ứng Dụng: Phù hợp cho các bài toán phân loại nhị phân và thường được sử dụng trong các hệ thống đánh giá rủi ro tín dụng.

❖ Feedforward Neural Network (FNN):

Ưu Điểm khi Áp Dụng:

- Khả Năng Học Phi Tuyến: FNN có khả năng học được các biểu diễn phi tuyến tính và phi tuyến tính, giúp nắm bắt được các mối quan hệ phức tạp trong dữ liệu.
- Phù Hợp Cho Dữ Liệu Phi Tuyến: Nếu dữ liệu của bạn có các mối quan hệ phi tuyến, FNN có thể là một lựa chọn tốt.
- Ứng Dụng: Phù hợp cho các bài toán phức tạp và dữ liệu có tính phi tuyến.

❖ Recurrent Neural Network (RNN):

Ưu Điểm khi Áp Dụng:

- Xử Lý Dữ Liệu Chuỗi Thời Gian: RNN được thiết kế để xử lý dữ liệu chuỗi thời gian, có thể hữu ích nếu dữ liệu của bạn có yếu tố thời gian.
- Chia Sẻ Trọng Số: RNN có khả năng chia sẻ trọng số qua các bước thời gian, giúp nắm bắt thông tin từ quá khứ.
- Ứng Dụng: Phù hợp cho các bài toán liên quan đến chuỗi thời gian, ví dụ như dự đoán sự phê duyệt thẻ tín dụng dựa trên dữ liệu lịch sử.

2.2 Phòng tránh Overfitting

Sau khi đã xây dựng các mô hình học máy, tiến hành áp dụng một số kỹ thuật để phòng tránh Overfitting tăng độ chính xác cho mô hình:

- Giảm Độ Phức Tạp của Mô Hình: Giảm độ sâu của cây quyết định, số lượng lớp và đơn vị ẩn trong các mô hình neural network có thể giúp giảm độ phức tạp của mô hình và nguy cơ overfitting.
- Dropout (Cho Neural Networks): Sử dụng kỹ thuật dropout trong các mạng neural. Dropout là quá trình ngẫu nhiên loại bỏ một số lượng đơn vị trong quá trình huấn luyện, giúp ngăn chặn overfitting.
- Tinh chỉnh Hyperparameters: Thực hiện tinh chỉnh hyperparameters một cách cẩn thận để điều chỉnh mô hình sao cho nó không quá phức tạp hoặc quá đơn giản

- Early Stopping: Sử dụng kỹ thuật early stopping để dừng quá trình huấn luyện khi hiệu suất trên tập validation không còn cải thiện nữa, giảm nguy cơ overfitting.

2. Thực hiện bài toán với các mô hình học máy

✓ 1.1 Đọc dữ liệu, thay đổi tên cột để chuẩn bị dữ liệu làm việc

```
[ ] 1 #import data
    2 data = pd.read_csv("credit-card-approval-prediction/application_record.csv")
    3 record = pd.read_csv("credit-card-approval-prediction/credit_record.csv")
```

```
[ ] 1 plt.rcParams['figure.facecolor'] = 'white'
```

```
1 # Tìm tất cả các tài khoản người dùng mở trong tháng
2 # nhóm DataFrame theo cột "ID" và tính giá trị tối thiểu của cột "MONTHS_BALANCE" cho mỗi nhóm
3 begin_month=pd.DataFrame(record.groupby(["ID"])["MONTHS_BALANCE"].agg(min))
4 # Đổi tên cột "MONTHS_BALANCE" trong DataFrame đầu tháng thành "begin_month1".
5 begin_month=begin_month.rename(columns={'MONTHS_BALANCE':'begin_month1'})
6 new_data=pd.merge(data,begin_month,how="left",on="ID") #merge to record data
```

Tạo một cột mới có tên là "dep_value" trong DataFrame ghi dữ liệu dựa trên giá trị trong cột "STATUS". Đặt "Yes" trong cột "dep_value" cho các hàng mà cột "STATUS" có giá trị '2', '3', '4' hoặc '5'.

```
[ ] 1 record['dep_value'] = None
    2 record['dep_value'][record['STATUS'] == '2'] = 'Yes'
    3 record['dep_value'][record['STATUS'] == '3'] = 'Yes'
    4 record['dep_value'][record['STATUS'] == '4'] = 'Yes'
    5 record['dep_value'][record['STATUS'] == '5'] = 'Yes'
```

Tạo một cột mới 'dep_value' trong DataFrame "cpunt" dựa trên số lần xuất hiện của 'dep_value' trong DataFrame "record". Sau đó, hợp nhất thông tin này vào DataFrame "new_data" dựa trên cột "ID" và tạo một cột mới 'target' dựa trên cột 'dep_value'.

```
[ ] 1 cpunt=record.groupby('ID').count()
    2 cpunt['dep_value'][cpunt['dep_value'] > 0]='Yes'
    3 cpunt['dep_value'][cpunt['dep_value'] == 0]='No'
    4 cpunt = cpunt[['dep_value']]
    5 new_data=pd.merge(new_data,cpunt,how='inner',on='ID')
    6 new_data['target']=new_data['dep_value']
    7 new_data.loc[new_data['target']=='Yes','target']=1
    8 new_data.loc[new_data['target']=='No','target']=0
```

In ấn số lượng giá trị duy nhất trong cột 'dep_value' của DataFrame "cpunt" và sau đó in số lượng giá trị chuẩn hóa.

```
1 print(cpunt['dep_value'].value_counts())
2 print(cpunt['dep_value'].value_counts(normalize=True))
```

```
No      45318
Yes       667
Name: dep_value, dtype: int64
No      0.985495
Yes     0.014505
Name: dep_value, dtype: float64
```

Đổi tên các cột trong DataFrame "new_data" bằng cách sử dụng phương pháp rename.

```
[ ] 1 new_data.rename(columns={'CODE_GENDER': 'Gender', 'FLAG_OWN_CAR': 'Car', 'FLAG_OWN_REALTY': 'Reality',
2                             'CNT_CHILDREN': 'ChldNo', 'AMT_INCOME_TOTAL': 'inc',
3                             'NAME_EDUCATION_TYPE': 'edutp', 'NAME_FAMILY_STATUS': 'famtp',
4                             'NAME_HOUSING_TYPE': 'houtp', 'FLAG_EMAIL': 'email',
5                             'NAME_INCOME_TYPE': 'inctp', 'FLAG_WORK_PHONE': 'wkphone',
6                             'FLAG_PHONE': 'phone', 'CNT_FAM_MEMBERS': 'famsize',
7                             'OCCUPATION_TYPE': 'occyp'
8                             }, inplace=True)
9 # Xử lý các giá trị bị thiếu trong new_data DataFrame bằng phương thức dropna.
10 new_data.dropna()
11 new_data = new_data.mask(new_data == 'NULL').dropna()
```

Tạo một bảng giá trị (IV) cho các biến trong DataFrame "new_data", sau đó loại bỏ một số biến được chỉ định trong danh sách tên.

Tại đây khởi tạo một bảng IV với tên biến trong cột biến và một cột giữ chỗ 'IV'. Sau đó, nó lặp qua danh sách tên và loại bỏ các hàng tương ứng với các biến được chỉ định từ bảng IV.


```

1 ivtable=pd.DataFrame(new_data.columns,columns=['variable'])
2 ivtable['IV']=None
3 namelist = ['FLAG_MOBIL','begin_month','dep_value','target','ID']
4
5 for i in namelist:
6     ivtable.drop(ivtable[ivtable['variable'] == i].index, inplace=True)
7 ivtable
8

```

| | variable | IV |
|----|---------------|------|
| 1 | Gender | None |
| 2 | Car | None |
| 3 | Reality | None |
| 4 | ChldNo | None |
| 5 | inc | None |
| 6 | inctp | None |
| 7 | edutp | None |
| 8 | famtp | None |
| 9 | houtp | None |
| 10 | DAYS_BIRTH | None |
| 11 | DAYS_EMPLOYED | None |
| 13 | wkphone | None |
| 14 | phone | None |
| 15 | email | None |
| 16 | occyp | None |
| 17 | famsize | None |
| 18 | begin_month1 | None |

▼ 1.2 Tạo một số hàm hỗ trợ

Định nghĩa một hàm `calc_iv` để tính Giá trị(IV) cho một đặc trưng cụ

- List item
- List item

thể trong DataFrame. Hàm này tính toán các thước đo khác nhau liên quan đến phân phối của đặc trưng đối với biến mục tiêu.

```

1 def calc_iv(df, feature, target, pr=False):
2     lst = []
3     df[feature] = df[feature].fillna("NULL")
4
5     for i in range(df[feature].nunique()):
6         val = list(df[feature].unique())[i]
7         lst.append([feature,
8                     val,
9                     df[df[feature] == val].count()[feature],
10                    df[(df[feature] == val) & (df[target] == 0)].count()[feature], # Good (think: Fraud == 0)
11                    df[(df[feature] == val) & (df[target] == 1)].count()[feature]]) # Bad (think: Fraud == 1)
12
13     data = pd.DataFrame(lst, columns=['Variable', 'Value', 'All', 'Good', 'Bad'])
14     data['Share'] = data['All'] / data['All'].sum()
15     data['Bad Rate'] = data['Bad'] / data['All']
16     data['Distribution Good'] = (data['All'] - data['Bad']) / (data['All'].sum() - data['Bad'].sum())
17     data['Distribution Bad'] = data['Bad'] / data['Bad'].sum()
18     data['WoE'] = np.log(data['Distribution Good'] / data['Distribution Bad'])
19
20     data = data.replace({'WoE': {np.inf: 0, -np.inf: 0}})
21
22     data['IV'] = data['WoE'] * (data['Distribution Good'] - data['Distribution Bad'])
23
24     data = data.sort_values(by=['Variable', 'Value'], ascending=[True, True])
25     data.index = range(len(data.index))
26

```

Định nghĩa một hàm `convert_dummy` để chuyển đổi một đặc trưng phân loại thành biến giả bằng cách sử dụng mã hóa one-hot.

```
[ ] 1 def convert_dummy(df, feature, rank=0):
2     pos = pd.get_dummies(df[feature], prefix=feature)
3     mode = df[feature].value_counts().index[rank]
4     biggest = feature + '_' + str(mode)
5     pos.drop([biggest], axis=1, inplace=True)
6     df.drop([feature], axis=1, inplace=True)
7     df=df.join(pos)
8     return df
```

Định nghĩa một hàm `get_category` để phân loại một cột số trong DataFrame thành các khoảng rời rạc. Hàm này sử dụng `pd.qcut` (cắt theo tỷ lệ) hoặc `pd.cut` (cắt theo chiều dài bằng nhau) dựa trên giá trị của tham số `qcut`

```
▶ 1 def get_category(df, col, binsnum, labels, qcut = False):
2     if qcut:
3         localdf = pd.qcut(df[col], q = binsnum, labels = labels) # quantile cut
4     else:
5         localdf = pd.cut(df[col], bins = binsnum, labels = labels) # equal-length cut
6
7     localdf = pd.DataFrame(localdf)
8     name = 'gp' + '_' + col
9     localdf[name] = localdf[col]
10    df = df.join(localdf[name])
11    df[name] = df[name].astype(object)
12    return df
```

Định nghĩa một hàm có tên là `plot_confusion_matrix` để trực quan hóa ma trận. Hàm này sử dụng Matplotlib để tạo một biểu đồ heatmap biểu diễn ma trận.

```
▶ 1 def plot_confusion_matrix(cm, classes,
2                             normalize=False,
3                             title='Confusion matrix',
4                             cmap=plt.cm.Blues):
5     if normalize:
6         cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
7
8     print(cm)
9
10    plt.imshow(cm, interpolation='nearest', cmap=cmap)
11    plt.title(title)
12    plt.colorbar()
13    tick_marks = np.arange(len(classes))
14    plt.xticks(tick_marks, classes)
15    plt.yticks(tick_marks, classes)
16
17    fmt = '.2f' if normalize else 'd'
18    thresh = cm.max() / 2.
19    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
20        plt.text(j, i, format(cm[i, j], fmt),
21                horizontalalignment="center",
22                color="white" if cm[i, j] > thresh else "black")
23
24    plt.tight_layout()
25    plt.ylabel('True label')
26    plt.xlabel('Predicted label')
```

✓ 1.3 Phân tích dữ liệu, tiền xử lý dữ liệu

Chuyển đổi cột 'Gender' trong DataFrame "new_data" từ giá trị phân loại ('F' và 'M') thành giá trị số học (0 và 1), sau đó tính Giá trị(IV) đã được biến đổi này.

```
1 new_data['Gender'] = new_data['Gender'].replace(['F','M'],[0,1])
2 print(new_data['Gender'].value_counts())
3 iv, data = calc_iv(new_data,'Gender','target')
4 ivtable.loc[ivtable['variable']=='Gender','IV']=iv
5 data.head()
```

```
0    15630
1     9504
Name: Gender, dtype: int64
This variable's IV is: 0.02520350452745081
0    15630
1     9504
Name: Gender, dtype: int64
```

| | Variable | Value | All | Good | Bad | Share | Bad Rate | Distribution Good | Distribution Bad | WoE | IV |
|---|----------|-------|-------|-------|-----|----------|----------|-------------------|------------------|-----------|----------|
| 0 | Gender | 0 | 15630 | 15400 | 230 | 0.621867 | 0.014715 | 0.623179 | 0.545024 | 0.134005 | 0.010473 |
| 1 | Gender | 1 | 9504 | 9312 | 192 | 0.378133 | 0.020202 | 0.376821 | 0.454976 | -0.188475 | 0.014730 |

Thay thế giá trị 'N' và 'Y' trong cột 'Car' bằng 0 và 1 tương ứng.

```
1 new_data['Car'] = new_data['Car'].replace(['N','Y'],[0,1])
2 print(new_data['Car'].value_counts())
3 iv, data=calc_iv(new_data,'Car','target')
4 ivtable.loc[ivtable['variable']=='Car','IV']=iv
5 data.head()
```

```
0    14618
1    10516
Name: Car, dtype: int64
This variable's IV is: 4.54248124999671e-06
0    14618
1    10516
Name: Car, dtype: int64
```

| | Variable | Value | All | Good | Bad | Share | Bad Rate | Distribution Good | Distribution Bad | WoE | IV |
|---|----------|-------|-------|-------|-----|----------|----------|-------------------|------------------|----------|----------|
| 0 | Car | 0 | 14618 | 14373 | 245 | 0.581603 | 0.016760 | 0.58162 | 0.580569 | 0.00181 | 0.000002 |
| 1 | Car | 1 | 10516 | 10339 | 177 | 0.418397 | 0.016831 | 0.41838 | 0.419431 | -0.00251 | 0.000003 |

Thay thế 'N' bằng 0 và 'Y' bằng 1 trong cột 'Reality', in số lượng giá trị, tính IV bằng cách sử dụng hàm calc_iv, cập nhật IV trong DataFrame "ivtable".

```
1 new_data['Reality'] = new_data['Reality'].replace(['N','Y'],[0,1])
2 print(new_data['Reality'].value_counts())
3 iv, data=calc_iv(new_data,'Reality','target')
4 ivtable.loc[ivtable['variable']=='Reality','IV']=iv
5 data.head()
```

1 16461
0 8673
Name: Reality, dtype: int64
This variable's IV is: 0.02744070350168343
1 16461
0 8673
Name: Reality, dtype: int64

| | Variable | Value | All | Good | Bad | Share | Bad Rate | Distribution Good | Distribution Bad | WoE | IV |
|---|----------|-------|-------|-------|-----|---------|----------|-------------------|------------------|-----------|----------|
| 0 | Reality | 0 | 8673 | 8494 | 179 | 0.34507 | 0.020639 | 0.34372 | 0.424171 | -0.210309 | 0.016920 |
| 1 | Reality | 1 | 16461 | 16218 | 243 | 0.65493 | 0.014762 | 0.65628 | 0.575829 | 0.130777 | 0.010521 |

Chuyển đổi cột 'phone' sang kiểu chuỗi, sau đó in giá trị chuẩn hóa, loại bỏ các hàng mà 'phone' là 'nan', và cuối cùng tính toán Giá trị(IV) bằng cách sử dụng hàm calc_iv.

```
[ ] 1 new_data['phone']=new_data['phone'].astype(str)
2 print(new_data['phone'].value_counts(normalize=True,sort=False))
3 new_data.drop(new_data[new_data['phone'] == 'nan' ].index, inplace=True)
4 iv, data=calc_iv(new_data,'phone','target')
5 ivtable.loc[ivtable['variable']=='phone','IV']=iv
6 data.head()
```

0 0.707209
1 0.292791
Name: phone, dtype: float64
This variable's IV is: 0.0005480495762639297
0 17775
1 7359
Name: phone, dtype: int64

| | Variable | Value | All | Good | Bad | Share | Bad Rate | Distribution Good | Distribution Bad | WoE | IV |
|---|----------|-------|-------|-------|-----|----------|----------|-------------------|------------------|-----------|----------|
| 0 | phone | 0 | 17775 | 17481 | 294 | 0.707209 | 0.016540 | 0.707389 | 0.696682 | 0.015251 | 0.000163 |
| 1 | phone | 1 | 7359 | 7231 | 128 | 0.292791 | 0.017394 | 0.292611 | 0.303318 | -0.035937 | 0.000385 |

In giá trị chuẩn hóa, sau đó chuyển đổi cột 'email' sang kiểu chuỗi, và cuối cùng tính Giá trị (IV) bằng cách sử dụng hàm `calc_iv`.

```
1 print(new_data['email'].value_counts(normalize=True,sort=False))
2 new_data['email']=new_data['email'].astype(str)
3 iv, data=calc_iv(new_data,'email','target')
4 ivtable.loc[ivtable['variable']=='email','IV']=iv
5 data.head()
```

```
0    0.89934
1    0.10066
Name: email, dtype: float64
This variable's IV is: 1.7343581493999816e-05
0    22604
1     2530
Name: email, dtype: int64
```

| | Variable | Value | All | Good | Bad | Share | Bad Rate | Distribution Good | Distribution Bad | WoE | IV |
|---|----------|-------|-------|-------|-----|---------|----------|-------------------|------------------|-----------|----------|
| 0 | email | 0 | 22604 | 22225 | 379 | 0.89934 | 0.016767 | 0.899361 | 0.898104 | 0.001398 | 0.000002 |
| 1 | email | 1 | 2530 | 2487 | 43 | 0.10066 | 0.016996 | 0.100639 | 0.101896 | -0.012407 | 0.000016 |

Xử lý cột 'wkphone' theo cách tương tự như mã nguồn trước. Đầu tiên, chuyển đổi 'wkphone' sang kiểu chuỗi, tính toán Giá trị(IV) bằng cách sử dụng hàm `calc_iv`, loại bỏ các hàng mà 'wkphone' là 'nan', và sau đó cập nhật IV trong DataFrame "ivtable".

```
1 new_data['wkphone']=new_data['wkphone'].astype(str)
2 iv, data = calc_iv(new_data,'wkphone','target')
3 new_data.drop(new_data[new_data['wkphone'] == 'nan' ].index, inplace=True)
4 ivtable.loc[ivtable['variable']=='wkphone','IV']=iv
5 data.head()
```

```
This variable's IV is: 0.002042429795148461
0    18252
1     6882
Name: wkphone, dtype: int64
```

| | Variable | Value | All | Good | Bad | Share | Bad Rate | Distribution Good | Distribution Bad | WoE | IV |
|---|----------|-------|-------|-------|-----|----------|----------|-------------------|------------------|-----------|----------|
| 0 | wkphone | 0 | 18252 | 17954 | 298 | 0.726188 | 0.016327 | 0.72653 | 0.706161 | 0.028436 | 0.000579 |
| 1 | wkphone | 1 | 6882 | 6758 | 124 | 0.273812 | 0.018018 | 0.27347 | 0.293839 | -0.071838 | 0.001463 |

Phân loại cột 'ChldNo' trong DataFrame "new_data" của bạn. Nếu giá trị trong cột 'ChldNo' lớn hơn hoặc bằng 2, bạn sẽ thay thế nó bằng '2More'.

```
[ ] 1 new_data.loc[new_data['ChldNo'] >= 2, 'ChldNo']='2More'
    2 print(new_data['ChldNo'].value_counts(sort=False))

0      15908
2More    3108
1       6118
Name: ChldNo, dtype: int64
```

Tính Giá trị (IV) cho cột 'ChldNo' trong DataFrame "new_data" sau khi phân loại nó thành các nhóm khác nhau.

```
[ ] 1 iv, data=calc_iv(new_data, 'ChldNo', 'target')
    2 ivtable.loc[ivtable['variable']=='ChldNo', 'IV']=iv
    3 data.head()
```

This variable's IV is: 0.0011214542503301935

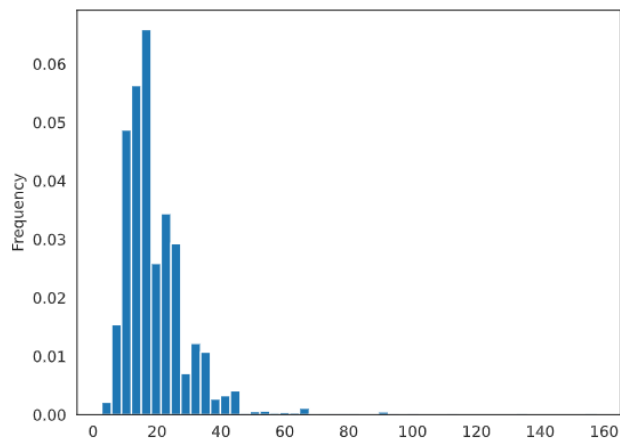
```
0      15908
1       6118
2More    3108
Name: ChldNo, dtype: int64
```

| | Variable | Value | All | Good | Bad | Share | Bad Rate | Distribution Good | Distribution Bad | WoE | IV |
|---|----------|-------|-------|-------|-----|----------|----------|-------------------|------------------|-----------|----------|
| 0 | ChldNo | 0 | 15908 | 15635 | 273 | 0.632928 | 0.017161 | 0.632689 | 0.646919 | -0.022243 | 0.000317 |
| 1 | ChldNo | 1 | 6118 | 6021 | 97 | 0.243415 | 0.015855 | 0.243647 | 0.229858 | 0.058259 | 0.000803 |
| 2 | ChldNo | 2More | 3108 | 3056 | 52 | 0.123657 | 0.016731 | 0.123665 | 0.123223 | 0.003580 | 0.000002 |

Thực hiện một số xử lý tiền xử lý trên cột 'inc' (thu nhập) trong DataFrame "new_data". Cụ thể, chuyển đổi cột 'inc' thành kiểu đối tượng, và chia giá trị cho 10,000

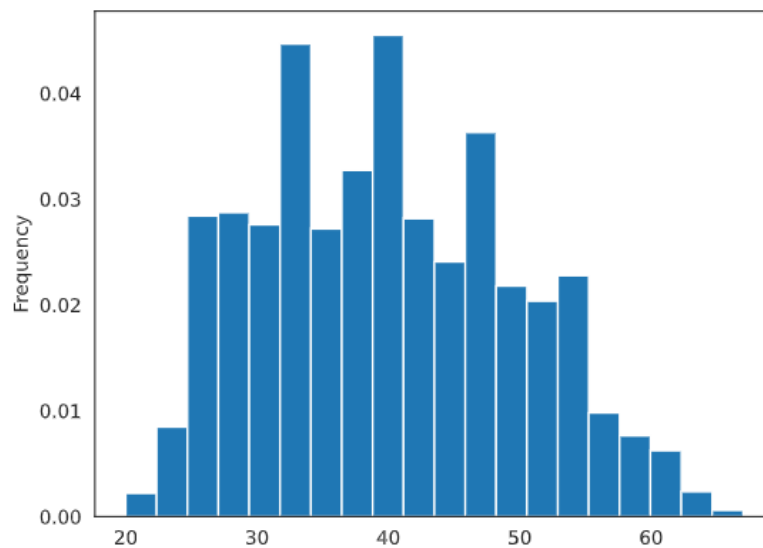
```
1 new_data['inc']=new_data['inc'].astype(object)
2 new_data['inc'] = new_data['inc']/10000
3 print(new_data['inc'].value_counts(bins=10,sort=False))
4 new_data['inc'].plot(kind='hist',bins=50,density=True)
```

```
(2.544, 18.18]      14663
(18.18, 33.66]      8464
(33.66, 49.14]      1637
(49.14, 64.62]       175
(64.62, 80.1]        124
(80.1, 95.58]         50
(95.58, 111.06]        4
(111.06, 126.54]        3
(126.54, 142.02]        6
(142.02, 157.5]         8
Name: inc, dtype: int64
<Axes: ylabel='Frequency'>
```



```
[ ] 1 new_data['Age'] = -(new_data['DAYS_BIRTH'])//365
    2 print(new_data['Age'].value_counts(bins=10,normalize=True,sort=False))
    3 new_data['Age'].plot(kind='hist',bins=20,density=True)
```

```
(19.951999999999998, 24.7]    0.025066
(24.7, 29.4]                 0.134280
(29.4, 34.1]                 0.169770
(34.1, 38.8]                 0.140805
(38.8, 43.5]                 0.173072
(43.5, 48.2]                 0.141880
(48.2, 52.9]                 0.099069
(52.9, 57.6]                 0.076550
(57.6, 62.3]                 0.032585
(62.3, 67.0]                 0.006923
Name: Age, dtype: float64
<Axes: ylabel='Frequency'>
```



Phân loại cột 'Age' thành năm nhóm bằng cách sử dụng hàm `get_category`, sau đó tính Giá trị (IV) cho biến nhóm mới ('gp_Age').

```
1 new_data = get_category(new_data,'Age',5, ["lowest","low","medium","high","highest"])
2 iv, data = calc_iv(new_data,'gp_Age','target')
3 ivtable.loc[ivtable['variable']=='DAYS_BIRTH','IV'] = iv
4 data.head()
```

```
This variable's IV is: 0.06593513858884348
medium    7916
low       7806
high      4414
lowest    4005
highest     993
Name: gp_Age, dtype: int64
```

| | Variable | Value | All | Good | Bad | Share | Bad Rate | Distribution Good | Distribution Bad | WoE | IV |
|---|----------|---------|------|------|-----|----------|----------|-------------------|------------------|-----------|----------|
| 0 | gp_Age | high | 4414 | 4323 | 91 | 0.175619 | 0.020616 | 0.174935 | 0.215640 | -0.209194 | 0.008515 |
| 1 | gp_Age | highest | 993 | 989 | 4 | 0.039508 | 0.004028 | 0.040021 | 0.009479 | 1.440361 | 0.043992 |
| 2 | gp_Age | low | 7806 | 7686 | 120 | 0.310575 | 0.015373 | 0.311023 | 0.284360 | 0.089625 | 0.002390 |
| 3 | gp_Age | lowest | 4005 | 3921 | 84 | 0.159346 | 0.020974 | 0.158668 | 0.199052 | -0.226754 | 0.009157 |
| 4 | gp_Age | medium | 7916 | 7793 | 123 | 0.314952 | 0.015538 | 0.315353 | 0.291469 | 0.078758 | 0.001881 |

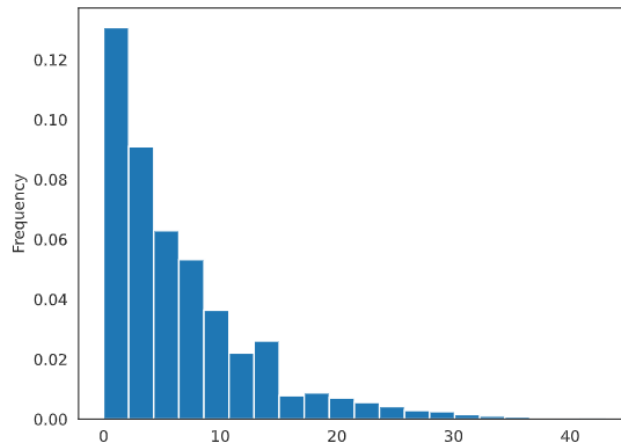
Code Text


```
[ ] 1 new_data = convert_dummy(new_data, 'gp_Age')
```

Tính số năm làm việc ('worktm') dựa trên cột 'DAYS_EMPLOYED' trong DataFrame "new_data". Sau khi tính toán thời gian làm việc, xử lý các giá trị âm (có thể chỉ ra dữ liệu thiếu hoặc dữ liệu ngoại lệ) bằng cách thay thế chúng bằng NaN, và sau đó điền giá trị NaN bằng giá trị trung bình của cột 'worktm'.

```
1 new_data['worktm'] = -(new_data['DAYS_EMPLOYED']) // 365
2 new_data[new_data['worktm'] < 0] = np.nan # replace by na
3 new_data['DAYS_EMPLOYED']
4 new_data['worktm'].fillna(new_data['worktm'].mean(), inplace=True) #replace na by mean
5 new_data['worktm'].plot(kind='hist', bins=20, density=True)
```

<Axes: ylabel='Frequency'>



Phân loại cột 'worktm' (số năm làm việc) thành năm nhóm bằng cách sử dụng hàm get_category, sau đó tính Giá trị (IV) cho biến nhóm mới ('gp_worktm').

```
1 new_data = get_category(new_data, 'worktm', 5, ["lowest", "low", "medium", "high", "highest"])
2 iv, data = calc_iv(new_data, 'gp_worktm', 'target')
3 ivtable.loc[ivtable['variable'] == 'DAYS_EMPLOYED', 'IV'] = iv
4 data.head()
```

This variable's IV is: 0.04022152230816303

```
lowest    18254
low       4987
medium    1378
high       425
highest     90
```

Name: gp_worktm, dtype: int64

| | Variable | Value | All | Good | Bad | Share | Bad Rate | Distribution Good | Distribution Bad | WoE | IV |
|---|-----------|---------|-------|-------|-----|----------|----------|-------------------|------------------|-----------|----------|
| 0 | gp_worktm | high | 425 | 423 | 2 | 0.016909 | 0.004706 | 0.017117 | 0.004739 | 1.284186 | 0.015895 |
| 1 | gp_worktm | highest | 90 | 90 | 0 | 0.003581 | 0.000000 | 0.003642 | 0.000000 | 0.000000 | 0.000000 |
| 2 | gp_worktm | low | 4987 | 4921 | 66 | 0.198416 | 0.013234 | 0.199134 | 0.156398 | 0.241573 | 0.010324 |
| 3 | gp_worktm | lowest | 18254 | 17916 | 338 | 0.726267 | 0.018516 | 0.724992 | 0.800948 | -0.099635 | 0.007568 |
| 4 | gp_worktm | medium | 1378 | 1362 | 16 | 0.054826 | 0.011611 | 0.055115 | 0.037915 | 0.374082 | 0.006434 |

```
1 new_data = convert_dummy(new_data, 'gp_worktm')
2 new_data['famsize'].value_counts(sort=False)
```

```
2.0    12697
1.0     4263
5.0      307
3.0     5216
4.0     2576
6.0       51
15.0       3
7.0       18
20.0        1
9.0         2
Name: famsize, dtype: int64
```

Thao tác trên cột 'famsize' trong DataFrame "new_data". Sau đó, chuyển đổi nó thành kiểu số nguyên, tạo một cột mới 'famsizegp', phân loại nó, và sau đó tính Giá trị(IV) cho biến đã phân loại.

```
[ ] 1 new_data['famsize']=new_data['famsize'].astype(int)
2 new_data['famsizegp']=new_data['famsize']
3 new_data['famsizegp']=new_data['famsizegp'].astype(object)
4 new_data.loc[new_data['famsizegp']>=3, 'famsizegp']='3more'
5 iv, data=calc_iv(new_data, 'famsizegp', 'target')
6 ivtable.loc[ivtable['variable']=='famsize', 'IV']=iv
7 data.head()
```

This variable's IV is: 0.006156138510778323

```
2    12697
3more    8174
1         4263
```

Name: famsizegp, dtype: int64

| | Variable | Value | All | Good | Bad | Share | Bad Rate | Distribution Good | Distribution Bad | WoE | IV |
|---|-----------|-------|-------|-------|-----|----------|----------|-------------------|------------------|-----------|----------|
| 0 | famsizegp | 1 | 4263 | 4179 | 84 | 0.169611 | 0.019704 | 0.169108 | 0.199052 | -0.163028 | 0.004882 |
| 1 | famsizegp | 2 | 12697 | 12489 | 208 | 0.505172 | 0.016382 | 0.505382 | 0.492891 | 0.025027 | 0.000313 |
| 2 | famsizegp | 3more | 8174 | 8044 | 130 | 0.325217 | 0.015904 | 0.325510 | 0.308057 | 0.055108 | 0.000962 |

```
[ ] 1 new_data = convert_dummy(new_data, 'famsizegp')
```

Chỉnh sửa giá trị của cột 'inctp' bằng cách kết hợp các hạng mục 'Pensioner' và 'Student' thành 'State servant'. Sau các sửa đổi này, tính Giá trị (IV) cho biến 'inctp'.

```
1 print(new_data['inctp'].value_counts(sort=False))
2 print(new_data['inctp'].value_counts(normalize=True, sort=False))
3 new_data.loc[new_data['inctp']=='Pensioner', 'inctp']='State servant'
4 new_data.loc[new_data['inctp']=='Student', 'inctp']='State servant'
5 iv, data=calc_iv(new_data, 'inctp', 'target')
6 ivtable.loc[ivtable['variable']=='inctp', 'IV']=iv
7 data.head()
```

```
Working      15622
Commercial associate    7052
State servant    2437
Student         10
Pensioner       13
Name: inctp, dtype: int64
Working      0.621549
Commercial associate    0.280576
State servant    0.096960
Student       0.000398
Pensioner     0.000517
Name: inctp, dtype: float64
This variable's IV is: 5.159303327851404e-05
Working      15622
Commercial associate    7052
State servant    2460
Name: inctp, dtype: int64
```

| | Variable | Value | All | Good | Bad | Share | Bad Rate | Distribution Good | Distribution Bad | WoE | IV |
|---|----------|----------------------|-------|-------|-----|----------|----------|-------------------|------------------|-----------|----------|
| 0 | inctp | Commercial associate | 7052 | 6933 | 119 | 0.280576 | 0.016875 | 0.280552 | 0.281991 | -0.005115 | 0.000007 |
| 1 | inctp | State servant | 2460 | 2418 | 42 | 0.097875 | 0.017073 | 0.097847 | 0.099526 | -0.017013 | 0.000029 |
| 2 | inctp | Working | 15622 | 15361 | 261 | 0.621549 | 0.016707 | 0.621601 | 0.618483 | 0.005028 | 0.000016 |

Phân loại lại cột 'occyp' (loại nghề nghiệp) trong DataFrame "new_data" bằng cách nhóm một số nghề nghiệp cụ thể vào các danh mục rộng hơn. Sau quá trình phân loại lại này, tính Giá trị (IV) cho biến 'occyp' đã được sửa đổi

```
1 new_data.loc[(new_data['occyp']=='Cleaning staff') | (new_data['occyp']=='Cooking staff') | (new_data['occyp']=='Drivers') |
2 new_data.loc[(new_data['occyp']=='Accountants') | (new_data['occyp']=='Core staff') | (new_data['occyp']=='HR staff') | (new
3 new_data.loc[(new_data['occyp']=='Managers') | (new_data['occyp']=='High skill tech staff') | (new_data['occyp']=='IT staff'
4 print(new_data['occyp'].value_counts())
5 iv, data=calc_iv(new_data, 'occyp', 'target')
6 ivtable.loc[ivtable['variable']=='occyp', 'IV']=iv
7 data.head()
```

```
Laborwk      10496
officewk     10183
hightecwk    4455
Name: occyp, dtype: int64
This variable's IV is: 0.004820472062853304
Laborwk      10496
officewk     10183
hightecwk    4455
Name: occyp, dtype: int64
```

| | Variable | Value | All | Good | Bad | Share | Bad Rate | Distribution Good | Distribution Bad | WoE | IV |
|---|----------|-----------|-------|-------|-----|----------|----------|-------------------|------------------|-----------|----------|
| 0 | occyp | Laborwk | 10496 | 10311 | 185 | 0.417602 | 0.017626 | 0.417247 | 0.438389 | -0.049428 | 0.001045 |
| 1 | occyp | hightecwk | 4455 | 4375 | 80 | 0.177250 | 0.017957 | 0.177039 | 0.189573 | -0.068404 | 0.000857 |
| 2 | occyp | officewk | 10183 | 10026 | 157 | 0.405148 | 0.015418 | 0.405714 | 0.372038 | 0.086652 | 0.002918 |

```
[ ] 1 new_data = convert_dummy(new_data, 'occyp')
```

Tính Giá trị (IV) cho cột 'houtp' (loại nhà ở) trong DataFrame "new_data".

```
1 iv, data=calc_iv(new_data,'houtp','target')
2 ivtable.loc[ivtable['variable']=='houtp','IV']=iv
3 data.head()
```

This variable's IV is: 0.0073275026880227365

```
House / apartment      22102
With parents           1430
Municipal apartment    812
Rented apartment       439
Office apartment       199
Co-op apartment        152
Name: houtp, dtype: int64
```

| | Variable | Value | All | Good | Bad | Share | Bad Rate | Distribution Good | Distribution Bad | WoE | IV |
|---|----------|---------------------|-------|-------|-----|----------|----------|-------------------|------------------|-----------|----------|
| 0 | houtp | Co-op apartment | 152 | 149 | 3 | 0.006048 | 0.019737 | 0.006029 | 0.007109 | -0.164705 | 0.000178 |
| 1 | houtp | House / apartment | 22102 | 21738 | 364 | 0.879367 | 0.016469 | 0.879654 | 0.862559 | 0.019624 | 0.000335 |
| 2 | houtp | Municipal apartment | 812 | 793 | 19 | 0.032307 | 0.023399 | 0.032090 | 0.045024 | -0.338655 | 0.004380 |
| 3 | houtp | Office apartment | 199 | 194 | 5 | 0.007918 | 0.025126 | 0.007850 | 0.011848 | -0.411619 | 0.001646 |
| 4 | houtp | Rented apartment | 439 | 433 | 6 | 0.017466 | 0.013667 | 0.017522 | 0.014218 | 0.208939 | 0.000690 |

Sắp xếp DataFrame "ivtable" theo Giá trị (IV) giảm dần, sau đó chỉnh sửa tên biến để đọc dễ hiểu hơn.

```
1 ivtable=ivtable.sort_values(by='IV',ascending=False)
2 ivtable.loc[ivtable['variable']=='DAYS_BIRTH','variable']='agegp'
3 ivtable.loc[ivtable['variable']=='DAYS_EMPLOYED','variable']='worktmgp'
4 ivtable.loc[ivtable['variable']=='inc','variable']='incgp'
5 ivtable
```

| | variable | IV |
|----|--------------|----------|
| 10 | agegp | 0.065935 |
| 8 | famtp | 0.043137 |
| 11 | worktmgp | 0.040222 |
| 3 | Reality | 0.027441 |
| 1 | Gender | 0.025204 |
| 7 | edutp | 0.010362 |
| 9 | houtp | 0.007328 |
| 17 | famsize | 0.006156 |
| 16 | occyp | 0.00482 |
| 13 | wkphone | 0.002042 |
| 4 | ChldNo | 0.001121 |
| 14 | phone | 0.000548 |
| 6 | inctp | 0.000052 |
| 15 | email | 0.000017 |
| 2 | Car | 0.000005 |
| 5 | incgp | None |
| 18 | begin_month1 | None |

Định nghĩa biến mục tiêu Y và ma trận đặc trưng X cho một nhiệm vụ mô hình hóa dự đoán. Ở đây, Y đại diện cho biến mục tiêu ('target'), và X bao gồm một lựa chọn các đặc trưng từ DataFrame "new_data" của bạn

```
1 Y = new_data['target']
2 X = new_data[['ID', 'Gender', 'Reality', 'ChldNo_1', 'ChldNo_2More', 'wkphone',
3             'gp_Age_high', 'gp_Age_highest', 'gp_Age_low',
4             'gp_Age_lowest', 'gp_worktm_high', 'gp_worktm_highest',
5             'gp_worktm_low', 'gp_worktm_medium', 'occyp_hightecwk',
6             'occyp_officewk', 'famsizegp_1', 'famsizegp_3more',
7             'houtp_Co-op apartment', 'houtp_Municipal apartment',
8             'houtp_Office apartment', 'houtp_Rented apartment',
9             'houtp_With parents', 'edutp_Higher education',
10            'edutp_Incomplete higher', 'edutp_Lower secondary', 'famtp_Civil marriage',
11            'famtp_Separated', 'famtp_Single / not married', 'famtp_Widow']]
```

Thực hiện tái chọn mẫu sử dụng Phương pháp Tổng hợp Thấp hơn Thiểu số (SMOTE) để cân bằng phân phối lớp trong biến mục tiêu Y của bạn.

```
[ ] 1 Y = Y.astype('int')
2 X_balance, Y_balance = SMOTE().fit_resample(X, Y)
3 X_balance = pd.DataFrame(X_balance, columns = X.columns)
```

2.1 Giải quyết bài toán bằng các mô hình học máy

Chia dữ liệu đã được tái chọn mẫu thành các tập huấn luyện và kiểm thử bằng cách sử dụng hàm train_test_split.

```
[ ] 1 X_train, X_test, y_train, y_test = train_test_split(X_balance, Y_balance,
2             stratify=Y_balance, test_size=0.3,
3             random_state = 10086)
```

```
1 X_train.head()
```

| | ID | Gender | Reality | ChldNo_1 | ChldNo_2More | wkphone | gp_Age_high | gp_Age_highest | gp_Age_low | gp_Age_lowest | ... | houtp_Office apartment | houtp_Rented apartment | houtp_With parents | edutp_Higher education |
|-------|--------------|----------|---------|----------|--------------|----------|-------------|----------------|------------|---------------|-----|------------------------|------------------------|--------------------|------------------------|
| 32515 | 5.125401e+06 | 0.000000 | 0.0 | 0.703311 | 0.0 | 0.296689 | 0.296689 | 0.0 | 0.703311 | 0.0 | ... | 0 | 0 | 0 | 0 |
| 14471 | 5.114645e+06 | 0.000000 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.000000 | 0.0 | 1.000000 | 0.0 | ... | 0 | 0 | 0 | 0 |
| 16400 | 5.096682e+06 | 0.000000 | 0.0 | 0.000000 | 0.0 | 1.0 | 0.000000 | 0.0 | 1.000000 | 0.0 | ... | 0 | 0 | 0 | 0 |
| 29956 | 5.054360e+06 | 0.630845 | 0.0 | 0.000000 | 0.0 | 1.0 | 0.369155 | 0.0 | 0.630845 | 0.0 | ... | 0 | 0 | 0 | 0 |
| 39904 | 5.135484e+06 | 0.626703 | 1.0 | 0.000000 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.000000 | 1.0 | ... | 0 | 0 | 0 | 1 |

5 rows x 30 columns

Trích xuất cột 'ID' từ tập huấn luyện và testing

```
[ ] 1 X_train_ID = X_train['ID']
    2 X_test_ID = X_test['ID']
```

```
[ ] 1 X_test_ID = X_test['ID']
    2
```

```
1 X_test.head()
```

| | ID | Gender | Reality | ChldNo_1 | ChldNo_2More | wkphone | gp_Age_high | gp_Age_highest | gp_Age_low | gp_Age_lowest | ... | houtp_Office apartment | houtp_Rented apartment | houtp_With parents | educ |
|-------|--------------|---------|----------|----------|--------------|----------|-------------|----------------|------------|---------------|-----|---------------------------|---------------------------|-----------------------|------|
| 24205 | 5.149085e+06 | 0.00000 | 1.000000 | 0.0 | 0.000000 | 0.0 | 0.000000 | 0.0 | 0.000000 | 0.0 | ... | 0 | 0 | 0 | |
| 39804 | 5.068813e+06 | 0.00000 | 0.140911 | 0.0 | 0.000000 | 0.859089 | 0.140911 | 0.0 | 0.859089 | 0.0 | ... | 0 | 0 | 0 | |
| 10003 | 5.061323e+06 | 0.00000 | 1.000000 | 1.0 | 0.000000 | 0.0 | 0.000000 | 0.0 | 0.000000 | 0.0 | ... | 0 | 0 | 0 | |
| 45484 | 5.024128e+06 | 1.00000 | 0.294786 | 0.0 | 0.705214 | 0.294786 | 0.000000 | 0.0 | 0.705214 | 0.0 | ... | 0 | 0 | 0 | |
| 25645 | 5.046483e+06 | 0.66108 | 0.338920 | 1.0 | 0.000000 | 0.0 | 0.000000 | 0.0 | 0.000000 | 0.0 | ... | 0 | 0 | 0 | |

5 rows x 30 columns

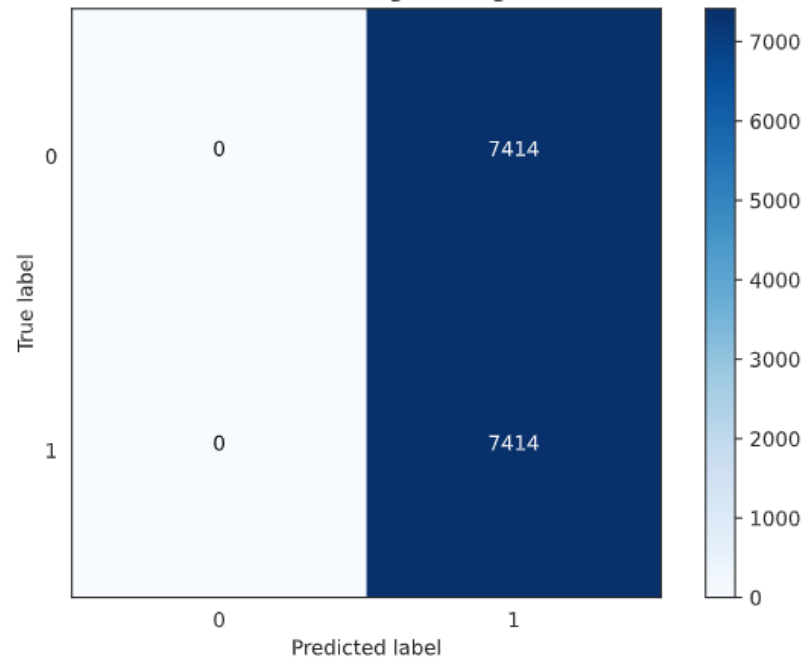
training logistic regression model

```
1 #Logistic Regression
2 model = LogisticRegression(C=0.8,
3                             random_state=0,
4                             solver='lbfgs')
5 model.fit(X_train, y_train)
6 y_predict = model.predict(X_test)
7
8 print('Accuracy Score is {:.5}'.format(accuracy_score(y_test, y_predict)))
9 print(pd.DataFrame(confusion_matrix(y_test,y_predict)))
10
11 sns.set_style('white')
12 class_names = ['0','1']
13 plot_confusion_matrix(confusion_matrix(y_test,y_predict),
14                       classes= class_names,
15                       title='Confusion Matrix: Logistic Regression')
```

Accuracy Score is 0.5

```
0 1
0 0 7414
1 0 7414
[[ 0 7414]
 [ 0 7414]]
```

Confusion Matrix: Logistic Regression




Kiểm tra độ dài của các nhãn dự đoán (`y_predict`), nhãn thực tế (`y_test`), và cột 'ID' trong tập kiểm thử (`X_test_ID`).

```
[ ] 1 len(y_predict), len(y_test), len(X_test_ID)

(14828, 14828, 14828)
```

Tạo một DataFrame mới, `final_prediction_comparison`, để so sánh các nhãn dự đoán (`y_predict`) với nhãn thực tế (`y_test`) cùng với các giá trị 'ID' tương ứng từ tập kiểm thử (`X_test_ID`).

```
▶ 1 data_dict = {'ID':X_test_ID, 'y_test':y_test, 'y_predict':y_predict}
  2 final_prediction_comparison = pd.DataFrame(data_dict)
  3 final_prediction_comparison
```

 ID y_test y_predict

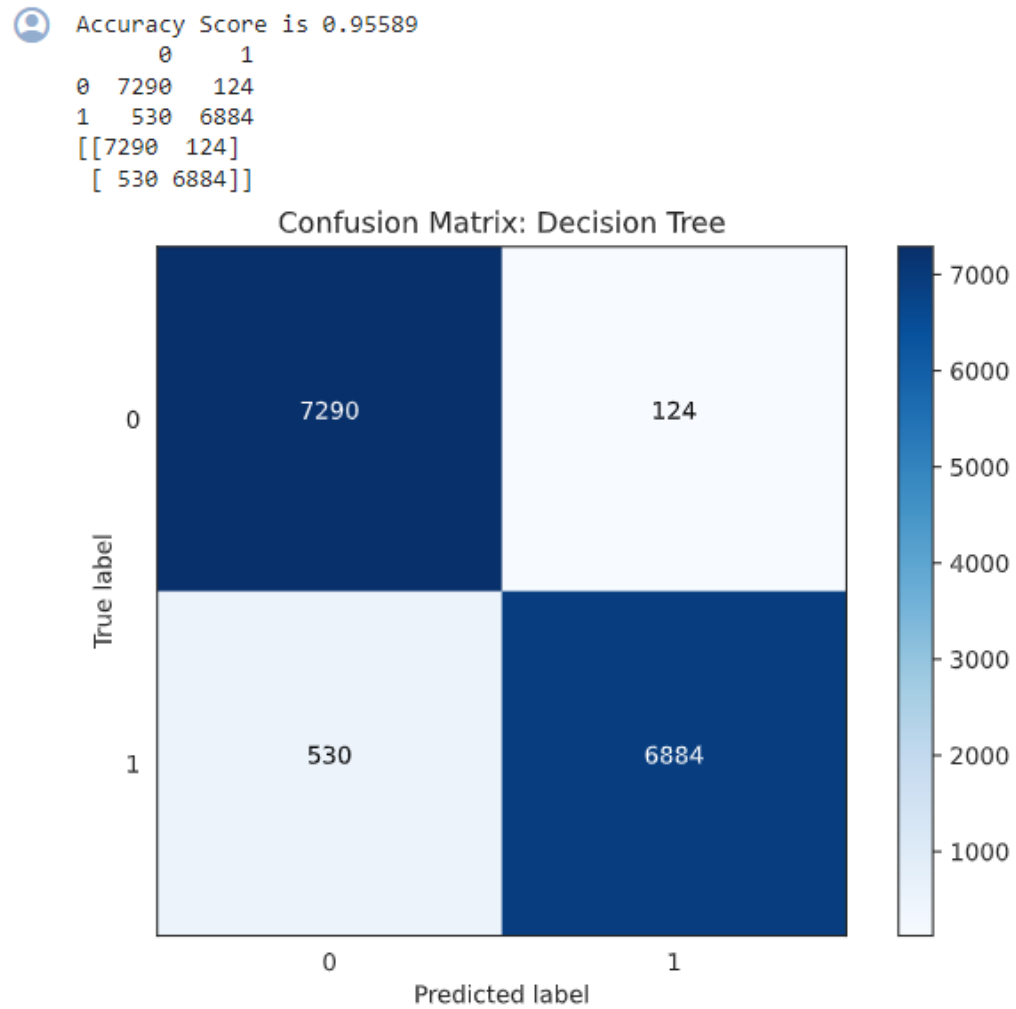
| | | | |
|-------|--------------|-----|-----|
| 24205 | 5.149085e+06 | 0 | 1 |
| 39804 | 5.068813e+06 | 1 | 1 |
| 10003 | 5.061323e+06 | 0 | 1 |
| 45484 | 5.024128e+06 | 1 | 1 |
| 25645 | 5.046483e+06 | 1 | 1 |
| ... | ... | ... | ... |
| 41229 | 5.045886e+06 | 1 | 1 |
| 24933 | 5.036469e+06 | 1 | 1 |
| 7233 | 5.045966e+06 | 0 | 1 |
| 39047 | 5.105078e+06 | 1 | 1 |
| 34162 | 5.045874e+06 | 1 | 1 |

14828 rows x 3 columns

Hình 13 Kết quả học mô hình Logistic Regression

training a decision tree model

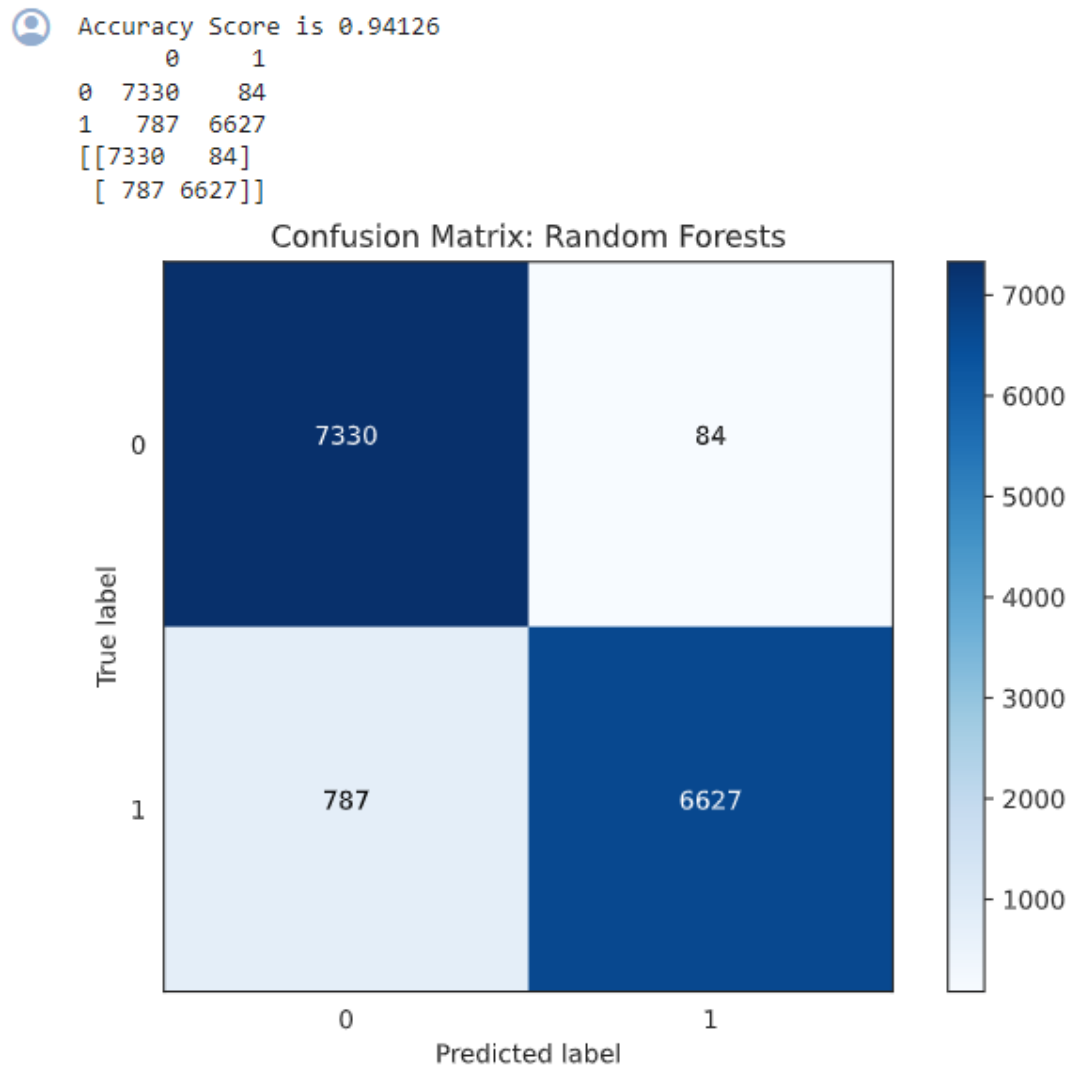
```
▶ 1 #decision tree
  2 model = DecisionTreeClassifier(max_depth=12,
  3                               min_samples_split=8,
  4                               random_state=1024)
  5 model.fit(X_train, y_train)
  6 y_predict = model.predict(X_test)
  7
  8 dt_accuracy_before = accuracy_score(y_test, y_predict)
  9 print('Accuracy Score is {:.5}'.format(accuracy_score(y_test, y_predict)))
 10 print(pd.DataFrame(confusion_matrix(y_test, y_predict)))
 11
 12 plot_confusion_matrix(confusion_matrix(y_test, y_predict),
 13                       classes=class_names,
 14                       title='Confusion Matrix: Decision Tree')
```



Hình 14 Kết quả học mô hình Decision Tree

random forest

```
1 #random forest
2 model = RandomForestClassifier(n_estimators=250,
3                               max_depth=12,
4                               min_samples_leaf=16
5                               )
6 model.fit(X_train, y_train)
7 y_predict = model.predict(X_test)
8
9 rf_accuracy_before = accuracy_score(y_test, y_predict)
10 print('Accuracy Score is {:.5}'.format(accuracy_score(y_test, y_predict)))
11 print(pd.DataFrame(confusion_matrix(y_test,y_predict)))
12
13 plot_confusion_matrix(confusion_matrix(y_test,y_predict),
14                       classes=class_names,
15                       title='Confusion Matrix: Random Forests')
```



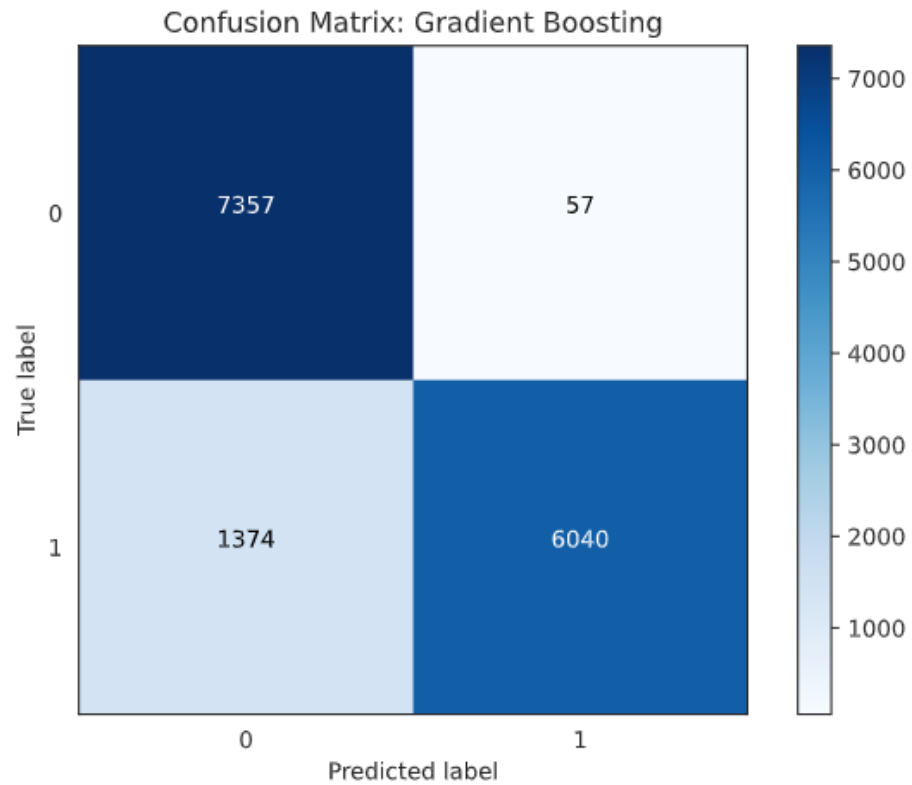
Hình 15 Kết quả học mô hình Random Forests

Gradient Boosting Accuracy Score is 0.90349

```

      0      1
0  7357    57
1  1374   6040
[[7357    57]
 [1374   6040]]

```



```

1 from sklearn.neighbors import KNeighborsClassifier
2
3 # K-Nearest Neighbors (KNN)
4 knn_model = KNeighborsClassifier(n_neighbors=5)
5
6 knn_model.fit(X_train, y_train)
7 y_knn_predict = knn_model.predict(X_test)
8
9 knn_accuracy_before = accuracy_score(y_test, y_knn_predict)
10 print(knn_accuracy_before)
11 print('KNN Accuracy Score is {:.5}'.format(accuracy_score(y_test, y_knn_predict)))
12 print(pd.DataFrame(confusion_matrix(y_test, y_knn_predict)))
13
14
15 plot_confusion_matrix(confusion_matrix(y_test, y_knn_predict),
16                       classes=class_names,
17                       title='Confusion Matrix: K-Nearest Neighbors')

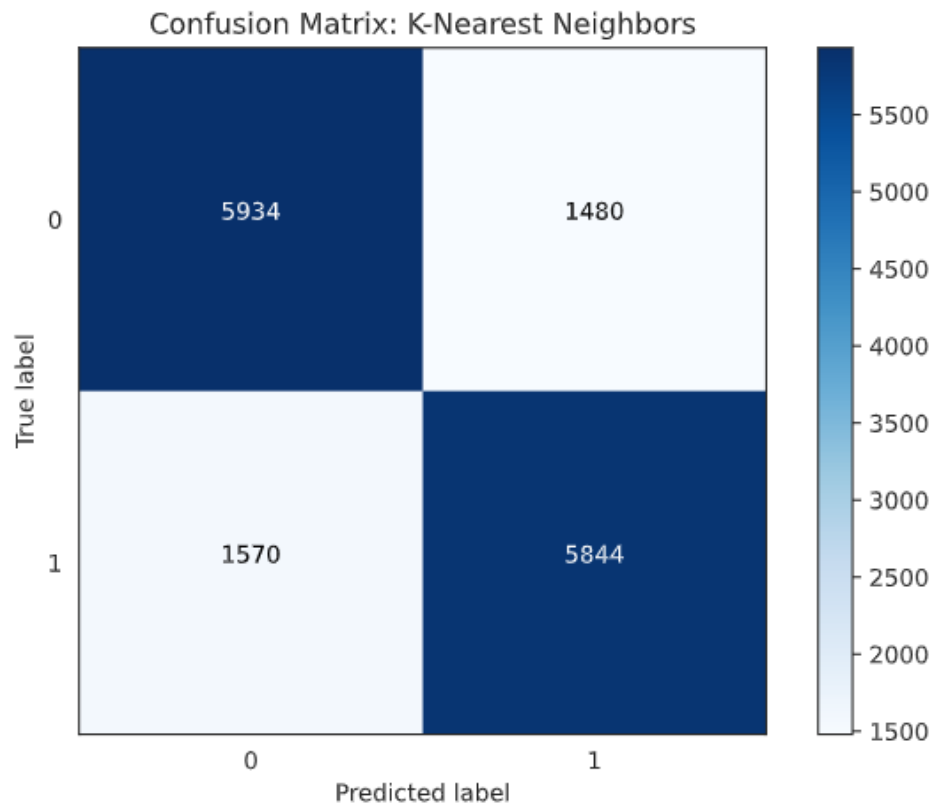
```

Xây dựng mô hình học máy KNN

0.7943080658214189
 KNN Accuracy Score is 0.79431

| | 0 | 1 |
|---|------|------|
| 0 | 5934 | 1480 |
| 1 | 1570 | 5844 |

```
[[5934 1480]
 [1570 5844]]
```



Hình 16 Kết quả học với KNN

3.1 Sử dụng Feed Forward Neural Network và Recurrent Neural Network để giải quyết bài toán

```
[ ] 1 # Chuyển đổi dữ liệu đầu vào sang kiểu dữ liệu float32
2 X_train_FNN = np.asarray(X_train).astype(np.float32)
3 X_test_FNN = np.asarray(X_test).astype(np.float32)
4 y_train_FNN = np.asarray(y_train).astype(np.float32)
5 y_test_FNN = np.asarray(y_test).astype(np.float32)
6
7 fnn_model = Sequential([
8     Dense(64, input_dim=X_train.shape[1], activation='relu'),
9     Dense(32, activation='relu'),
10    Dense(1, activation='sigmoid')
11 ])
12
13 fnn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
14
15
16 fnn = fnn_model.fit(X_train_FNN, y_train_FNN, epochs=10, batch_size=32, validation_data=(X_test_FNN, y_test_FNN))
17 fnn_loss, fnn_accuracy = fnn_model.evaluate(X_test_FNN, y_test_FNN)
18
19 print(f'FNN Accuracy Score is: {fnn_accuracy:.4f}')
```

```
▶ 1 X_train_3D = X_train_FNN.reshape((X_train.shape[0], 1, X_train.shape[1]))
2 X_test_3D = X_test_FNN.reshape((X_test_FNN.shape[0], 1, X_test_FNN.shape[1]))
3 y_train_RNN = np.asarray(y_train_FNN).astype(np.float32)
4 y_test_RNN = np.asarray(y_test_FNN).astype(np.float32)
5 # Tạo mô hình RNN
6 rnn_model = Sequential([
7     SimpleRNN(50, activation='relu', input_shape=(1, X_train_FNN.shape[1])),
8     Dense(1, activation='sigmoid')
9 ])
10
11 rnn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
12 rnn = rnn_model.fit(X_train_3D, y_train_RNN, epochs=10, batch_size=32, validation_data=(X_test_3D, y_test_RNN))
13
14 rnn_loss, rnn_accuracy = rnn_model.evaluate(X_test_3D, y_test_RNN)
15 print(f'RNN Accuracy Score is: {rnn_accuracy:.4f}')
```

4.1 Áp dụng các kỹ thuật tránh Overfitting các mô hình học máy

```

1 model = DecisionTreeClassifier(
2     max_depth=15,
3     min_samples_split=12,
4     min_samples_leaf=12,
5     random_state=1024
6 )
7
8 model.fit(X_train, y_train)
9 y_predict = model.predict(X_test)
10
11
12 dt_accuracy_after = accuracy_score(y_test, y_predict)
13 print(' Decision Tree Accuracy Score is {:.5}'.format(dt_accuracy_after))
14 print(pd.DataFrame(confusion_matrix(y_test, y_predict)))
15
16 plot_confusion_matrix(confusion_matrix(y_test, y_predict),
17                       classes=class_names,
18                       title='Confusion Matrix: Decision Tree')
19
20 print('\n Decision Tree Accuracy Score before is ', dt_accuracy_before)
21 print(' Decision Tree Accuracy Score after is ', dt_accuracy_after)

```

Decision Tree

```

[ ] 1 # Adjusted parameters to prevent overfitting
2 gb_model = GradientBoostingClassifier(n_estimators=200, learning_rate=0.05, max_depth=5, min_samples_split=5,
3                                     min_samples_leaf=2, random_state=42)
4
5 gb_model.fit(X_train, y_train)
6 y_gb_predict = gb_model.predict(X_test)
7
8 gb_accuracy_after = accuracy_score(y_test, y_gb_predict)
9 print('Gradient Boosting Accuracy Score is {:.5}'.format(gb_accuracy_after))
10 print(pd.DataFrame(confusion_matrix(y_test, y_gb_predict)))
11
12 plot_confusion_matrix(confusion_matrix(y_test, y_gb_predict),
13                       classes=class_names,
14                       title='Confusion Matrix: Gradient Boosting')
15
16 print('\n Gradient Boosting Accuracy Score before is ', gb_accuracy_before)
17 print(' Gradient Boosting Accuracy Score after is ', gb_accuracy_after)

```

Gradient Boosting phòng tránh Overfitting


```

1 #random forest
2 model = RandomForestClassifier(n_estimators=350,
3                               max_depth=15,
4                               min_samples_leaf=20
5                               )
6 model.fit(X_train, y_train)
7 y_predict = model.predict(X_test)
8
9 rf_accuracy_after = accuracy_score(y_test, y_predict)
10 print('Random forest Accuracy Score is {:.5}'.format(accuracy_score(y_test, y_predict)))
11 print(pd.DataFrame(confusion_matrix(y_test,y_predict)))
12
13 plot_confusion_matrix(confusion_matrix(y_test,y_predict),
14                       classes=class_names,
15                       title='Confusion Matrix: Random Forests')
16
17
18 print('\nRandom forest Accuracy Score before is ', rf_accuracy_before)
19 print('Random forest Accuracy Score after is ', rf_accuracy_after)
20

```

Hình 17 Mô hình Random forest phòng tránh Overfitting

```

1 # K-Nearest Neighbors (KNN)
2 knn_model = KNeighborsClassifier(n_neighbors=5, weights='uniform') # Điều chỉnh số láng giềng và trọng số nếu cần thiết
3 knn_model.fit(X_train, y_train)
4
5 y_knn_predict = knn_model.predict(X_test)
6 knn_accuracy_after = accuracy_score(y_test, y_knn_predict)
7 print('KNN Accuracy Score is {:.5}'.format(knn_accuracy_after))
8 print(pd.DataFrame(confusion_matrix(y_test, y_knn_predict)))
9
10 plot_confusion_matrix(confusion_matrix(y_test, y_knn_predict),
11                       classes=class_names,
12                       title='Confusion Matrix: K-Nearest Neighbors')
13
14 print('\nKNN Accuracy Score before is ', knn_accuracy_before)
15 print('KNN Accuracy Score after is ', knn_accuracy_after)

```

4.2 Áp dụng các kỹ thuật tránh Overfitting cho FNN và RNN

```
[ ] 1 def plot_comparison(history1, history2, metric='accuracy'):
2     plt.plot(history1.history[metric], label='Without Overfitting')
3     plt.plot(history2.history[metric], label='With Overfitting')
4     plt.title(f'Model Comparison - {metric}')
5     plt.xlabel('Epoch')
6     plt.ylabel(metric)
7     plt.legend()
8     plt.show()
9
```

```
1 # Tạo mô hình FNN có sử dụng tránh Overfitting
2 fnn_model = Sequential([
3     Dense(64, input_dim=X_train.shape[1], activation='relu'),
4     Dropout(0.5),
5     Dense(32, activation='relu'),
6     Dropout(0.5),
7     Dense(1, activation='sigmoid')
8 ])
9
10 fnn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
11 fnn_dropout = fnn_model.fit(X_train_FNN, y_train_FNN, epochs=10, batch_size=32, validation_data=(X_test_FNN, y_test_FNN))
12
13
14 fnn_loss, fnn_accuracy_dropout = fnn_model.evaluate(X_test_FNN, y_test_FNN)
15 print(f'FNN Accuracy Score is: {fnn_accuracy:.4f}')
```

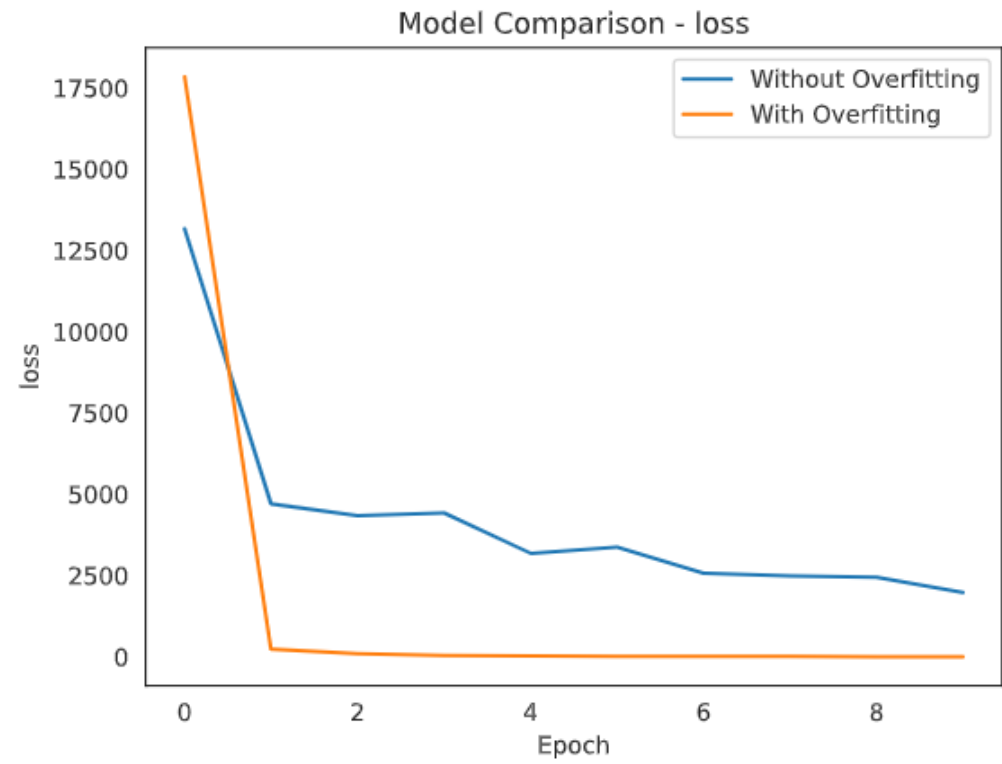
Hình 18 Tạo mô hình FNN có sử dụng tránh Overfitting

```
1 # Tạo mô hình FNN có sử dụng tránh Overfitting
2 rnn_model = Sequential([
3     Dropout(0.5),
4     SimpleRNN(50, activation='relu', input_shape=(1, X_train_FNN.shape[1])),
5     Dense(1, activation='sigmoid')
6 ])
7
8 rnn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
9
10
11 rnn_dropout = rnn_model.fit(X_train_3D, y_train_RNN, epochs=10, batch_size=32, validation_data=(X_test_3D, y_test_RNN))
12
13 rnn_loss, rnn_accuracy = rnn_model.evaluate(X_test_3D, y_test_RNN)
14 print(f'RNN Accuracy Score is: {rnn_accuracy:.4f}')
```

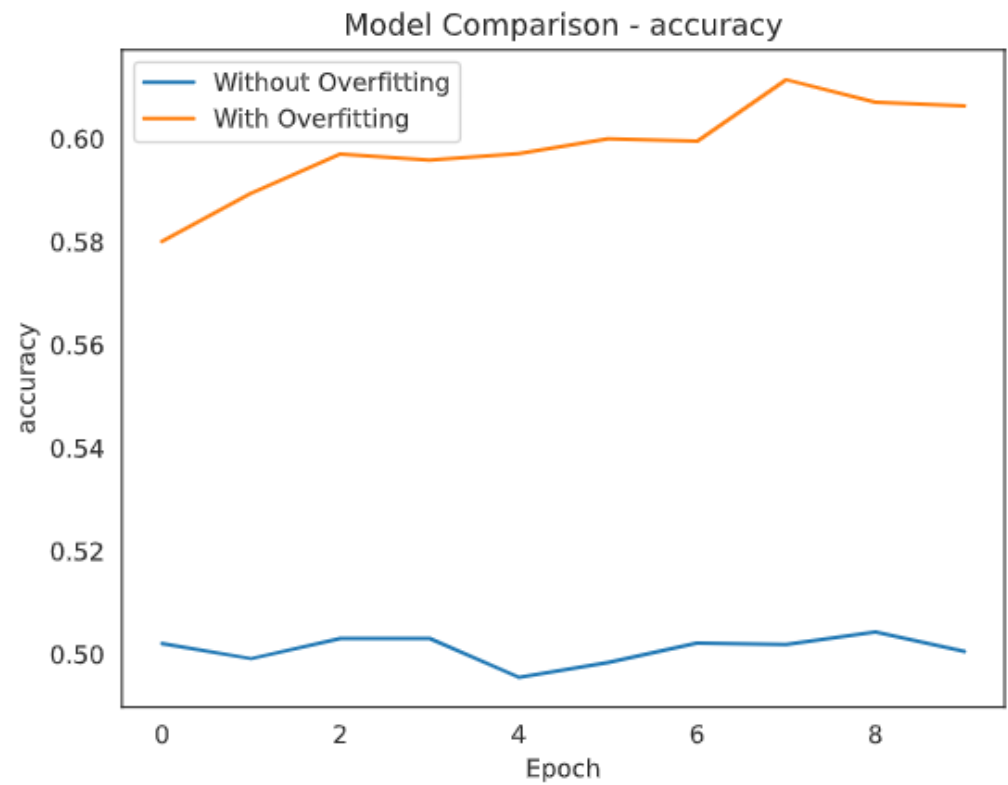
Hình 19 Tạo mô hình RNN có sử dụng tránh Overfitting

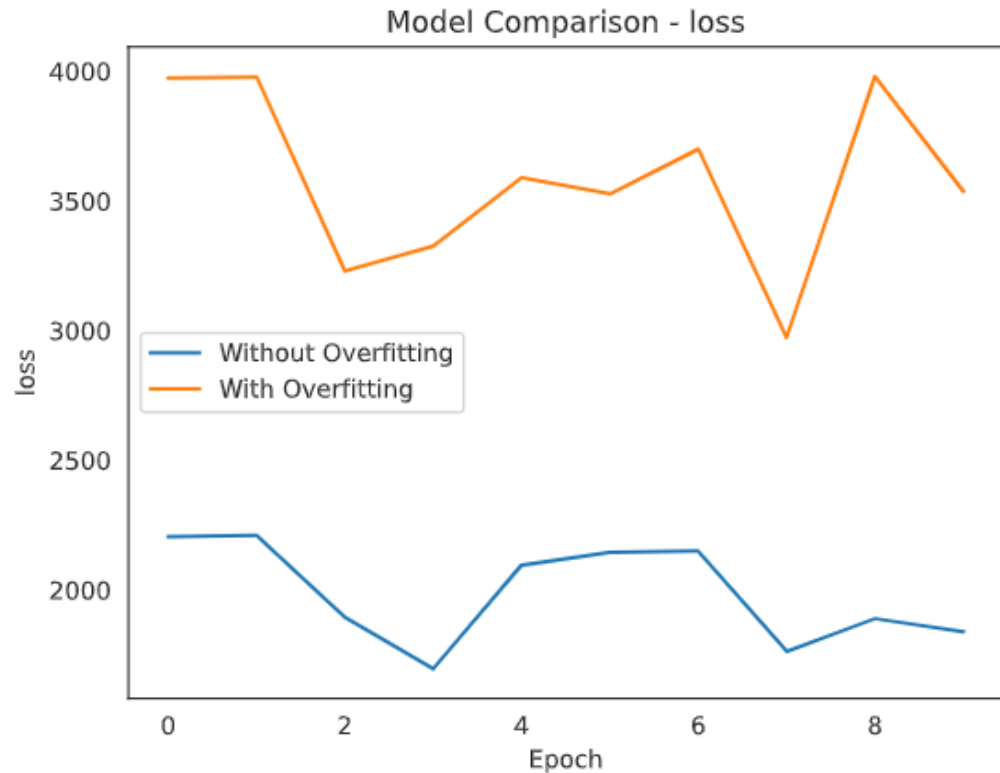
```
1 # So sánh độ chính xác  
2 plot_comparison(fnn, fnn_dropout, metric='accuracy')  
3 # So sánh hàm mất mát  
4 plot_comparison(fnn, fnn_dropout, metric='loss')
```





```
1 # So sánh độ chính xác
2 plot_comparison(rnn, rnn_dropout, metric='accuracy')
3 # So sánh hàm mất mát
4 plot_comparison(rnn, rnn_dropout, metric='loss')
5
```





5.1 Các giải pháp để cải tiến mô hình

Để cải thiện độ chính xác của mô hình sau khi đã huấn luyện, chúng ta có thể thực hiện các bước sau:

1. Phân tích Trường hợp Sai: Xem xét các trường hợp mà mô hình dự đoán sai. Phân tích các false positives và false negatives để hiểu rõ hơn về lý do mô hình đưa ra các quyết định không chính xác.
2. Tinh chỉnh Tham số Mô hình: Kiểm tra và điều chỉnh các tham số của mô hình. Sử dụng kỹ thuật tinh chỉnh hyperparameters như Grid Search hoặc Random Search để tìm ra các giá trị tham số tốt nhất.
3. Tăng cường Dữ liệu (Data Augmentation): Tăng kích thước của tập dữ liệu bằng cách thêm các biến thể của các mẫu hiện có hoặc tạo dữ liệu tổng hợp. Điều này có thể giúp mô hình học được từ nhiều biến thể hơn và cải thiện khả năng tổng quát hóa.
4. Điều chỉnh các yếu tố như độ sâu của cây (đối với cây quyết định), số lượng cây (đối với rừng ngẫu nhiên), hay số lượng layer và units (đối với mạng nơ-ron) để kiểm soát overfitting và underfitting.

Biện pháp đã sử dụng trong bài toán:

1. Tinh chỉnh Tham số Mô hình: Kiểm tra và điều chỉnh các tham số của mô hình. Sử dụng kỹ thuật tinh chỉnh hyperparameters như Grid Search hoặc Random Search để tìm ra các giá trị tham số tốt nhất.
2. Điều chỉnh các yếu tố như độ sâu của cây (đối với cây quyết định), số lượng cây (đối với rừng ngẫu nhiên), hay số lượng layer và units (đối với mạng nơ-ron) để kiểm soát overfitting và underfitting.

=> Kết quả đã được cải thiện, mô hình học chính xác hơn:

PHẦN 3 – KẾT LUẬN

1. Kiến thức sau nghiên cứu, tìm hiểu

Qua việc nghiên cứu và áp dụng các mô hình tối ưu để giải quyết các vấn đề trong bài toán học máy, ta đã khám phá và thực hiện một loạt các khía cạnh quan trọng về Machine Learning, và sau đây là những điểm quan trọng mà nhóm em đã tìm hiểu được:

- Khái niệm, ứng dụng của các phương pháp tối ưu:
 - Tối ưu hóa mô hình: Optimizer được sử dụng để điều chỉnh các tham số của mô hình máy học để mô hình có thể học được từ dữ liệu huấn luyện và dự đoán tốt trên dữ liệu mới.
 - Học máy sâu: Trong các mô hình học sâu như mạng nơ-ron sâu (deep neural networks), optimizer giúp điều chỉnh hàng trăm hoặc hàng nghìn tham số mô hình để tối ưu hóa hiệu suất.
 - Cải thiện hàm mất mát: Một số optimizer như Adam, SGD (Stochastic Gradient Descent), RMSprop, ... được thiết kế để tối ưu hóa hàm mất mát, giúp mô hình học nhanh hơn và tránh các điểm tối ưu cục bộ.
- Phương pháp, giải thuật để tối ưu mô hình học máy: Chúng ta đã tìm hiểu về các phương pháp tối ưu (Optimizer) như Gradient Descent, Adam, SGD (Stochastic Gradient Descent), RMSprop,... Mỗi mô hình có cách tiếp cận riêng và yêu cầu hiểu biết cụ thể về cách chúng hoạt động.
- Phân tích, so sánh các phương pháp tối ưu: Trong việc so sánh, phải xem xét mục tiêu, ưu điểm, nhược điểm của từng phương pháp. Việc so sánh này giúp chúng ta chọn ra mô hình tốt nhất cho từng tình huống cụ thể.

Ngoài ra nhóm đã giải quyết bài toán học máy dựa trên tập dữ liệu Credit Card Approval Prediction bao gồm thống kê, phân tích, ứng dụng các mô hình cơ bản...

2. Những điều cần lưu ý

- Sự quan trọng của việc tiền xử lý dữ liệu: Dữ liệu sạch và chuẩn hoá là yếu tố quyết định cho hiệu suất của mô hình.

- Khả năng chọn mô hình phù hợp: Việc hiểu rõ loại bài toán và dữ liệu giúp chọn ra mô hình phù hợp.
- Quá trình đánh giá và điều chỉnh mô hình: Kiểm tra hiệu suất và tối ưu hóa mô hình là một phần quan trọng của quá trình học máy.
- Overfitting và giải quyết nó: Overfitting là một vấn đề thường gặp và cần được kiểm soát thông qua các biện pháp như Regularization.
- Những kiến thức và kinh nghiệm thu thập từ môn học này sẽ có giá trị lớn khi áp dụng vào các dự án thực tế và nghiên cứu tiếp theo trong lĩnh vực Machine Learning.

TÀI LIỆU THAM KHẢO

Tiếng Việt

3. Slide môn Nhập môn học máy của trường đại học Tôn Đức Thắng
4. <https://viblo.asia/p/optimizer-hieu-sau-ve-cac-thuat-toan-toi-uu-gdsgdadam-Obq5QQ9E5D8>

Tiếng Anh

- [1] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back propagating errors, nature 323 (1986) 533536.
- [2] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, L. D. Jackel, Handwritten digit recognition with a back-propagation network, in: Advances in neural information processing systems, pp. 396404.
- [3] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (1997) 17351780.
- [4] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770778.
- [5] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 47004708