BỘ GIÁO DỤC VÀ ĐÀO TẠO
VIỆN HÀN LÂM KHOA HỌC VÀ CÔNG NGHỆ VIỆT NAM

HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

Trần Tuấn Long

PHÁT HIỆN WEBSHELL TRONG MÃ NGUỒN PHP BẰNG HỌC MÁY

LUẬN VĂN THẠC SĨ MÁY TÍNH

Hà Nội – 2026

BỘ GIÁO DỤC VÀ ĐÀO TẠO VIỆN HÀN LÂM KHOA HỌC
VÀ CÔNG NGHỆ VIỆT NAM

HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

Trần Tuấn Long

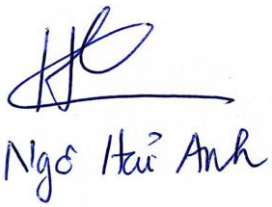
PHÁT HIỆN WEBSHELL TRONG MÃ NGUỒN PHP BẰNG HỌC MÁY

LUẬN VĂN THẠC SĨ MÁY TÍNH

Ngành: HỆ THỐNG THÔNG TIN

Mã số: 8480104

Người hướng dẫn Khoa học : TS. Ngô Hải Anh



Hà Nội – 2026

LỜI CAM ĐOAN

Tôi xin cam đoan đề tài nghiên cứu trong luận văn này là công trình nghiên cứu của tôi dựa trên những tài liệu, số liệu do chính tôi tự tìm hiểu và nghiên cứu. Đồng thời, các kết quả nêu trong luận văn là trung thực và chưa từng được công bố trong bất kỳ công trình nào khác.

Hà Nội, ngày 2 tháng 7 năm 2026

HỌC VIÊN

Trần Tuấn Long

LỜI CẢM ƠN

Luận văn được thực hiện tại Học viện Khoa học và Công nghệ – Viện Hàn lâm Khoa học và Công nghệ Việt Nam, dưới sự hướng dẫn của TS. Ngô Hải Anh. Tôi xin bày tỏ lòng biết ơn sâu sắc đến thầy về định hướng khoa học, người đã động viên, trao đổi nhiều kiến thức và chỉ bảo tôi vượt qua những khó khăn để hoàn thành luận văn này. Tôi trân trọng cảm ơn Học viện Khoa học và Công nghệ và Viện Công nghệ thông tin – Viện Hàn lâm Khoa học và Công nghệ Việt Nam đã tạo điều kiện thuận lợi cho tôi trong suốt quá trình nghiên cứu thực hiện luận văn.

Cuối cùng, tôi xin gửi lời cảm ơn sâu sắc tới gia đình, bạn bè, những người đã luôn ủng hộ, giúp đỡ và hỗ trợ tôi về mọi mặt để tôi yên tâm học tập đạt kết quả tốt.

Hà Nội, ngày 2 tháng 7 năm 2026

HỌC VIÊN

Trần Tuấn Long

MỤC LỤC

DANH MỤC CÁC BẢNG

DANH MỤC CÁC HÌNH VẼ, ĐỒ THỊ

MỞ ĐẦU

Lý do chọn đề tài

Trong bối cảnh các hệ thống ứng dụng web ngày càng phát triển, các hình thức tấn công nhằm chiếm quyền điều khiển máy chủ thông qua việc cài cắm *webshell* (mã độc web) đang trở nên phổ biến và gây hậu quả nghiêm trọng. Webshell thường được ngụy trang dưới dạng các tệp mã nguồn hợp lệ (ví dụ PHP), cho phép kẻ tấn công thực thi lệnh từ xa, thao tác tệp tin và dữ liệu, duy trì truy cập lâu dài, từ đó tạo điều kiện cho các hoạt động hậu khai thác. Đặc biệt, với các kỹ thuật *làm rối* (obfuscation), *mã hoá/biến đổi biểu diễn* (encoding/encryption) và biến thể cú pháp, việc phát hiện webshell bằng các cơ chế truyền thống dựa trên chữ ký hoặc luật đơn giản ngày càng gặp nhiều hạn chế.

Về tình hình nghiên cứu, các phương pháp phát hiện webshell có thể chia thành nhiều hướng như: *dựa trên chữ ký/luật* (signature/rule-based), *phân tích tĩnh theo chương trình* (program-analysis-based static detection), *phân tích động/xám động* (dynamic/gray-box), cũng như *học máy* (machine learning) và *học sâu* (deep learning). Mỗi hướng tiếp cận đều có ưu điểm và hạn chế riêng, đồng thời chịu tác động mạnh bởi dữ liệu đầu vào và chiến lược né tránh của webshell. Trên thực tế, hướng *học máy theo đặc trưng* (feature-based machine learning) cho phép xây dựng một quy trình phát hiện ở mức ứng dụng, có tính tái lập và có thể triển khai quét mã nguồn diện rộng mà không cần thực thi chương trình.

Từ các vấn đề đặt ra nêu trên, đề tài *Phát hiện webshell trong mã nguồn PHP bằng học máy* được lựa chọn nhằm xây dựng một quy trình phát hiện dựa trên *phân tích tĩnh* (static analysis) và *đặc trưng* (features) của mã, đồng thời đánh giá khách quan hiệu quả của các mô hình học máy cơ bản. Đề tài phù hợp với yêu cầu luận văn thạc sĩ theo định hướng ứng dụng, vừa đảm bảo tính khoa học (có quy trình, có đánh giá), vừa có ý nghĩa thực tiễn trong hỗ trợ rà soát an toàn mã nguồn.

Mục đích nghiên cứu

Mục đích của luận văn là xây dựng và đánh giá một quy trình phát hiện webshell trong mã nguồn PHP bằng học máy (machine learning) ở mức cơ bản, cụ thể: (i) Chuẩn hóa tập dữ liệu dạng *đặc trưng* (feature-level dataset) phục vụ bài toán phân

loại nhị phân `webshell`/`normal`; (ii) Xây dựng mô hình phát hiện dựa trên các thuật toán học máy phổ biến, dễ triển khai và dễ đánh giá kết quả đầu ra; (iii) Đánh giá khách quan mô hình bằng các chỉ số và hình minh họa chuẩn (precision/recall/F1, ma trận nhầm lẫn, ROC/PR), từ đó lựa chọn mô hình phù hợp trong phạm vi nghiên cứu.

Nội dung nghiên cứu

Luận văn tập trung vào các nội dung chính sau:

- Nghiên cứu tổng quan về webshell, đặc điểm và kỹ thuật né tránh; phân tích các hướng phát hiện webshell và định vị hướng nghiên cứu phù hợp (Chương 1).
- Thu thập, kiểm định chất lượng và tiền xử lý tập dữ liệu dạng CSV theo hướng *đặc trưng* (features); xây dựng pipeline tiền xử lý và chia tập train/validation/test theo *lấy mẫu phân tầng* (stratified sampling) để đảm bảo tái lập thí nghiệm (Chương 2).
- Xây dựng mô hình học máy cơ bản cho bài toán phân loại webshell; lựa chọn tham số đơn giản dựa trên tập validation và thiết kế quy trình suy luận (inference pipeline) trên dữ liệu mới (Chương 3).
- Mô phỏng và đánh giá kết quả trên tập test độc lập bằng các chỉ số định lượng và hình minh họa (confusion matrix, ROC, PR); so sánh mô hình và đưa ra kết luận lựa chọn mô hình phù hợp (Chương 4).

Cơ sở khoa học và tính thực tiễn của đề tài

Về cơ sở khoa học, luận văn dựa trên nền tảng của *phân tích tĩnh* (static analysis) và *học máy có giám sát* (supervised learning) cho bài toán *phân loại nhị phân* (binary classification). Cách tiếp cận dựa trên *đặc trưng* (features) cho phép biểu diễn mã nguồn dưới dạng vector số, từ đó áp dụng các mô hình học máy cơ bản như *hồi quy logistic* (logistic regression) và *rừng ngẫu nhiên* (random forest). Quy trình thực nghiệm được thiết kế theo nguyên tắc tách bạch train/validation/test và hạn chế *rò rỉ dữ liệu* (data leakage), đảm bảo kết quả đánh giá có ý nghĩa khoa học và có thể tái lập.

Về tính thực tiễn, kết quả của luận văn có thể hỗ trợ quá trình rà soát mã nguồn PHP trong các hoạt động như kiểm tra an toàn, đánh giá mã, hoặc hỗ trợ phản ứng sự cố. Việc sử dụng mô hình học máy cơ bản giúp giảm rào cản triển khai, dễ tích hợp, đồng thời các chỉ số đánh giá và hình minh họa giúp người vận hành hiểu rõ mức *báo động giả* (false positive) và *bỏ sót* (false negative) để lựa chọn phương án phù hợp.

Những đóng góp của luận văn

Trong phạm vi nghiên cứu, luận văn đạt được các đóng góp chính sau:

- Xây dựng quy trình xử lý tập dữ liệu dạng *đặc trưng* (feature-level dataset) cho phát hiện webshell PHP, bao gồm kiểm định chất lượng dữ liệu, sàng lọc đặc trưng cơ bản và chuẩn hóa thang đo.
- Thiết kế chia tập train/validation/test theo *lấy mẫu phân tầng* (stratified split) và lưu các *tạo phẩm dữ liệu* (data artifacts) nhằm đảm bảo tính tái lập và liên kết.
- Xây dựng và so sánh các mô hình học máy cơ bản (baseline và mô hình khuyến nghị), lựa chọn tham số đơn giản dựa trên validation.
- Đánh giá khách quan trên tập test bằng bộ chỉ số và hình minh họa chuẩn (precision/recall/F1, confusion matrix, ROC/PR), từ đó đưa ra kết luận lựa chọn mô hình phù hợp trong phạm vi dữ liệu nghiên cứu.

Đối tượng nghiên cứu

Đối tượng nghiên cứu của luận văn là bài toán phát hiện webshell trong mã nguồn PHP theo hướng *phân tích tĩnh* (static analysis). Cụ thể: (i) dữ liệu đầu vào là các mẫu mã PHP được biểu diễn dưới dạng *đặc trưng* (features) và nhãn `webshell/normal`; (ii) mô hình nghiên cứu là các thuật toán *học máy* (machine learning) ở mức cơ bản cho phân loại nhị phân; (iii) đối tượng đánh giá là hiệu quả phát hiện webshell trên tập test độc lập thông qua các chỉ số và hình minh họa chuẩn.

Phương pháp nghiên cứu

Luận văn sử dụng kết hợp các phương pháp nghiên cứu sau:

- *Phương pháp tổng quan tài liệu* (literature review): hệ thống hóa các hướng phát hiện webshell, phân tích ưu/nhược điểm và định vị hướng nghiên cứu phù hợp.
- *Phương pháp thực nghiệm* (experimental method): xây dựng pipeline tiền xử lý dữ liệu, huấn luyện mô hình, lựa chọn tham số trên validation và đánh giá trên test.
- *Phương pháp phân tích và so sánh* (analysis and comparison): so sánh mô hình dựa trên các chỉ số định lượng (precision/recall/F1, ROC-AUC, PR-AUC) và các hình minh họa (confusion matrix, ROC, PR), từ đó rút ra nhận xét và kết luận.
- *Công cụ và môi trường*: sử dụng ngôn ngữ Python để xử lý dữ liệu, huấn luyện và xuất dữ liệu đầu ra.

CHƯƠNG 1. TỔNG QUAN VÀ PHÂN TÍCH CÁC PHƯƠNG PHÁP PHÁT HIỆN WEBSHELL

1.1. Bối cảnh và vấn đề nghiên cứu

Webshell (mã độc web) là một dạng *cửa hậu* (backdoor) được cài vào ứng dụng web nhằm cho phép kẻ tấn công thực hiện các thao tác từ xa như thực thi lệnh, truy xuất hoặc sửa đổi dữ liệu, tải lên/tải xuống tệp và duy trì quyền truy cập trái phép. Webshell thường được ngụy trang dưới dạng các tệp mã nguồn hợp lệ (ví dụ PHP), khiến việc phát hiện trở nên khó khăn, đặc biệt khi kết hợp với các kỹ thuật làm rối (obfuscation) hoặc mã hóa chuỗi [1].

Việc phát hiện webshell gặp nhiều thách thức do mã độc có thể rất đa dạng về kích thước, hành vi gần giống với các đoạn mã quản trị hợp pháp và thường xuyên thay đổi để né tránh các phương pháp phát hiện truyền thống. Theo các khảo sát gần đây, PHP vẫn là nền tảng xuất hiện nhiều webshell nhất và xu hướng ứng dụng học máy trong phát hiện webshell ngày càng được quan tâm [1].

Các phương pháp phát hiện webshell hiện nay chủ yếu được chia thành hai hướng: *phân tích động* (dynamic analysis) và *phân tích tĩnh* (static analysis). Phân tích động quan sát hành vi khi chương trình được thực thi nên có khả năng phát hiện một số biến thể mới, tuy nhiên yêu cầu môi trường thực thi và chi phí tính toán cao. Ngược lại, phân tích tĩnh xử lý trực tiếp trên mã nguồn hoặc biểu diễn trung gian mà không cần thực thi chương trình, phù hợp với các hệ thống quét mã nguồn và quy trình kiểm thử phần mềm [2, 3].

Trong hướng phân tích tĩnh, các nghiên cứu tập trung vào ba nhóm phương pháp chính gồm: (i) phát hiện dựa trên chữ ký (signature-based), (ii) phân tích luồng dữ liệu nhiễm bẩn (taint analysis) và (iii) học máy dựa trên đặc trưng (feature-based machine learning). Phương pháp dựa trên chữ ký có tốc độ nhanh nhưng khó phát hiện các biến thể webshell mới; phân tích nhiễm bẩn có khả năng theo dõi luồng dữ liệu nhưng thường phức tạp và phụ thuộc môi trường phân tích; trong khi học máy tận dụng các đặc trưng thống kê, cấu trúc và hành vi để nâng cao khả năng tổng quát hóa đối với các mẫu chưa từng xuất hiện [2, 3].

Bên cạnh các mô hình học máy truyền thống, nhiều nghiên cứu gần đây đã áp

dụng các mô hình học sâu và mô hình biểu diễn mã nguồn như CNN, Transformer, CodeBERT hoặc Large Language Model (LLM) nhằm nâng cao hiệu quả phát hiện webshell. Tuy nhiên, các phương pháp này thường yêu cầu tập dữ liệu lớn, chi phí huấn luyện cao và khó triển khai trong các môi trường có nguồn lực hạn chế [4, 5].

Xuất phát từ các phân tích trên, luận văn tập trung nghiên cứu bài toán phát hiện webshell PHP theo hướng phân tích tĩnh kết hợp học máy dựa trên đặc trưng. Hướng tiếp cận này không yêu cầu thực thi chương trình, thuận lợi cho việc quét mã nguồn trên quy mô lớn, đồng thời vẫn đảm bảo khả năng diễn giải mô hình và chi phí triển khai phù hợp với điều kiện thực tế. Các chương tiếp theo lần lượt trình bày bộ dữ liệu và quy trình tiền xử lý (Chương 2), các mô hình phân loại được lựa chọn (Chương 3) và kết quả đánh giá thực nghiệm theo các chỉ số Accuracy, Precision, Recall, F1-score, ROC-AUC và PR-AUC (Chương 4).

1.2. Đặc điểm webshell trong PHP và kỹ thuật né tránh

1.2.1. Đặc điểm môi trường PHP và lý do webshell thường nhắm tới PHP

PHP là một ngôn ngữ kịch bản phía máy chủ có mức phổ biến cao trong các hệ thống web thực tế. Đặc trưng vận hành của PHP (chạy trong môi trường máy chủ web, tương tác trực tiếp với tham số HTTP, và có nhiều hàm thao tác hệ thống/tệp tin) tạo điều kiện thuận lợi cho kẻ tấn công cấy webshell nhằm duy trì truy cập. Các nghiên cứu và tổng quan gần đây đều xem PHP là một trong các mục tiêu trọng tâm của phát hiện webshell, đồng thời ghi nhận rằng sự đa dạng biến thể và kỹ thuật che giấu khiến việc phát hiện dựa trên luật đơn giản ngày càng khó khăn [3, 6].

Một yếu tố quan trọng khác là PHP có hệ sinh thái ứng dụng lớn (CMS, framework), đồng thời tồn tại nhiều bề mặt tấn công liên quan đến tải tệp, include, và xử lý dữ liệu từ người dùng. Sau khi khai thác thành công, kẻ tấn công thường cài webshell như một cơ chế *duy trì hiện diện* (persistence) và mở rộng hoạt động hậu khai thác (post-exploitation), bao gồm thực thi lệnh, thao tác cơ sở dữ liệu, tải thêm mã độc và che giấu dấu vết [6].

1.2.2. Các chức năng điển hình của webshell PHP

Xét theo góc độ hành vi, webshell PHP thường cung cấp một hoặc nhiều chức năng sau:

(i) *thực thi lệnh hệ điều hành* (system command execution) hoặc *thực thi mã động* (dynamic code execution), cho phép kẻ tấn công gửi lệnh qua tham số HTTP để thực thi trên máy chủ. Các cơ chế thực thi có thể dựa vào hàm nhạy cảm (ví dụ nhóm

hàm thực thi) hoặc cơ chế evaluate/interpret trong ngôn ngữ.

(ii) *thao tác tệp tin* (file operations), bao gồm đọc/ghi/xoá tệp, tải lên/tải xuống, nén/giải nén, và thay đổi quyền truy cập. Đây là nền tảng để kẻ tấn công tải thêm công cụ, trích xuất thông tin nhạy cảm hoặc phá hoại hệ thống.

(iii) *tương tác cơ sở dữ liệu* (database interaction), phục vụ đánh cắp dữ liệu, liệt kê bảng, hoặc chèn dữ liệu phục vụ leo thang.

(iv) *duy trì truy cập và chống phân tích* (persistence and anti-analysis), bao gồm tự xoá, vô hiệu một số cơ chế kiểm tra, hoặc thay đổi hành vi khi phát hiện môi trường phân tích [6].

Điểm đáng chú ý là cùng một mục tiêu thực thi lệnh, webshell có thể hiện thực theo nhiều kiểu: từ *one-liner* (one-line backdoor), webshell ngắn, đến webshell dài với cấu trúc phức tạp và cơ chế mã hoá hai chiều cho dữ liệu trao đổi [6, 7]. Sự khác biệt về độ dài và cấu trúc này ảnh hưởng trực tiếp đến cách thiết kế đặc trưng và mô hình học máy ở các chương sau.

1.2.3. Kỹ thuật né tránh dựa trên làm rối và mã hoá/giải mã

Một nhóm kỹ thuật né tránh phổ biến là *làm rối* (obfuscation) và *mã hoá/giải mã* (encoding/encryption). Mục tiêu là che giấu các chuỗi/hàm nhạy cảm và làm cho mã khó đọc đối với kiểm tra thủ công hoặc *dò mẫu* (pattern matching). Luận án về webshell và học sâu mô tả webshell có thể sử dụng các biến thể làm rối như chèn ký tự ngẫu nhiên, thêm đoạn mã chết (dead code), đổi tên biến/hàm, hoặc biến đổi luồng điều khiển để phá vỡ các quy tắc nhận dạng đơn giản; đồng thời kết hợp mã hoá/giải mã (ví dụ base64 và các cơ chế mã hoá đối xứng) để che giấu payload và thông điệp điều khiển [6].

Trong thực tế, *mã hoá base64* (base64 encoding) thường được dùng để che giấu chuỗi lệnh hoặc đoạn mã được thực thi. Khi kết hợp với cơ chế giải mã trong runtime, webshell có thể chỉ lộ ra hành vi độc hại tại thời điểm giải mã và gọi hàm thực thi. Điều này làm giảm hiệu quả của các bộ luật dựa trên từ khoá trực tiếp. Các nghiên cứu gần đây nhấn mạnh rằng nhiều mẫu webshell né được chữ ký bằng cách che giấu từ khoá thực thi hoặc tham số điều khiển, khiến mô hình phát hiện cần dựa trên các dấu hiệu thống kê/cấu trúc thay vì chỉ dựa trên chuỗi ký tự [6, 7].

Một hệ quả quan trọng cho thiết kế luận văn là: các kỹ thuật che giấu này thường tạo ra các biểu hiện thống kê bất thường như tăng mật độ dấu ngoặc kép/chuỗi,

tăng entropy, tăng độ dài dòng hoặc tăng mức "ngẫu nhiên" của ký tự. Vì vậy, nhóm đặc trưng thống kê và độ phức tạp chuỗi mà luận văn dùng ở Chương 2 (ví dụ entropy, độ dài từ lớn nhất, độ dài dòng) là hợp lý để phản ánh tác động của obfuscation/encoding lên mã nguồn.

1.2.4. Kỹ thuật né tránh bằng phá vỡ quy tắc và biến thể cú pháp

Bên cạnh obfuscation theo kiểu mã hoá chuỗi, một hướng né tránh quan trọng là phá vỡ trực tiếp các quy tắc phát hiện dựa trên mẫu. Nghiên cứu MMFDetect mô tả bối cảnh rule-based thường dựa vào các bước như nhận diện tag PHP, phát hiện hàm thực thi (ví dụ `eval`), và nhận diện tham số điều khiển (ví dụ `$_REQUEST`). Tuy nhiên, chỉ cần obfuscate tên hàm hoặc phá vỡ cấu trúc token (chèn comment, thay đổi cách viết) là có thể khiến so khớp thất bại [7]. Các kỹ thuật này không nhất thiết cần mã hoá payload; chỉ cần biến đổi cú pháp đủ để làm "mù" bộ luật.

Ngoài ra, một số biến thể có tính "đặc thù" (idiosyncratic) có thể lợi dụng đặc điểm của PHP như tự tăng (self-increment) và toán tử XOR để tạo hành vi tương đương nhưng không mang dấu hiệu cú pháp quen thuộc. Trong trường hợp này, phát hiện thuần văn bản có thể gặp khó khăn; từ đó, một số nghiên cứu đề xuất bổ sung biểu diễn thay thế hoặc kết hợp đa mô thức (ví dụ trích đặc trưng cấu trúc/ảnh hoá mã) để tăng khả năng nhận dạng [7].

Từ góc nhìn luận văn, điều này củng cố lựa chọn tránh phụ thuộc hoàn toàn vào chữ ký, và thay vào đó sử dụng tập đặc trưng đa dạng (thống kê, độ dài, dấu hiệu hành vi) kết hợp mô hình học máy cơ bản. Ở Chương 3, mô hình *rừng ngẫu nhiên* (random forest) được ưu tiên vì có khả năng học các tổ hợp điều kiện phi tuyến trên nhiều đặc trưng, phù hợp khi dấu hiệu bị "bẻ" khỏi dạng tuyến tính đơn giản.

1.2.5. Kỹ thuật né tránh bằng include và hàm tự định nghĩa

Một thách thức đặc trưng trong PHP là cơ chế *include* (include) và khả năng định nghĩa hàm tuỳ ý. Công trình WTA chỉ ra rằng các phương pháp dựa trên dò từ khoá/hàm nguy hiểm có thể bị vượt qua bằng *include-type webshell* (include-type webshell), trong đó đoạn mã độc được đặt ở tệp được include; hoặc bằng *user-defined function-type webshell* (webshell dùng hàm tự định nghĩa), nơi kẻ tấn công bọc lời gọi hàm nhạy cảm trong hàm do mình định nghĩa để tránh bị phát hiện bởi danh sách từ khoá/hàm nguy hiểm cố định [3]. Điều này làm nổi bật hạn chế của phân tích tĩnh "nông" (shallow static scan) nếu không thực hiện *phân tích liên thủ tục* (interprocedural analysis) hoặc không truy vết luồng dữ liệu.

Trong bối cảnh luận văn theo hướng học máy cơ bản, luận văn không triển khai taint analysis đầy đủ như WTA, nhưng các ý tưởng từ WTA vẫn rất hữu ích để định hướng đặc trưng: các webshell dạng include/udf thường để lại dấu hiệu về thao tác chuỗi, tham số điều khiển, hoặc chuỗi "giải mã rồi thực thi". Các dấu hiệu này có thể được phản ánh gián tiếp qua các đặc trưng thống kê/độ dài/entropy và một số đặc trưng đếm (nếu dataset có). Vì vậy, việc đánh giá mô hình ở Chương 4 cần xem xét kỹ *bỏ sót* (false negative), vì các biến thể include/udf là nhóm có khả năng gây bỏ sót cao nếu đặc trưng không đủ phân biệt.

1.2.6. Kỹ thuật né tránh bằng độ dài lớn và biến thể tiến hoá

Một xu hướng đáng chú ý là webshell ngày càng dài và phức tạp, tạo thách thức cho cả trích đặc trưng và mô hình hoá. Nghiên cứu về *học tăng dần* (incremental learning) mô tả rằng biến thể malware/webshell cập nhật nhanh hơn tốc độ cập nhật quy tắc truyền thống; đồng thời, các chuỗi biểu diễn (ví dụ opcode) có thể rất dài, chứa nhiều thành phần không liên quan làm giảm hiệu năng nếu không có cơ chế xử lý dài hiệu quả [8]. Nghiên cứu này cũng nhấn mạnh nhu cầu chuẩn hoá/giải mã nhiều lớp (URL/Base64) để giảm tác động của encoding obfuscation trước khi mô hình hoá hành vi [8].

Về mặt tổng quan, điều này gợi ý hai hướng: (i) cần biểu diễn mã theo cách "bền" trước obfuscation (ví dụ opcode/bytecode hoặc các đặc trưng thực thi), và (ii) cần cập nhật mô hình theo thời gian để theo kịp biến thể mới. Trong phạm vi luận văn hiện tại, luận văn tập trung vào một thiết kế thực nghiệm rõ ràng và tái lập (Chương 2–4), do đó chưa triển khai incremental learning. Tuy nhiên, các thảo luận về biến thể tiến hoá sẽ được nêu như hướng phát triển tương lai ở phần kết luận, đồng thời giúp giải thích vì sao các chỉ số đánh giá ở Chương 4 cần bao gồm ROC/PR để quan sát hành vi mô hình theo ngưỡng.

1.2.7. Thiết kế tập dữ liệu, mô hình và đánh giá của luận văn

Tổng hợp các đặc điểm và kỹ thuật né tránh nêu trên cho thấy phát hiện web-shell PHP là bài toán có tính đa dạng cao về hình thái mã và chiến lược che giấu. Điều này dẫn đến các mục đích tương ứng với cấu trúc của luận văn:

(i) Về dữ liệu (Chương 2): cần ưu tiên tập đặc trưng phản ánh cả dấu hiệu trực tiếp và gián tiếp của obfuscation (ví dụ entropy, độ dài chuỗi/dòng, mức độ "ngẫu nhiên"), vì đây là hệ quả thường gặp của kỹ thuật mã hoá/che giấu [6].

(ii) Về mô hình (Chương 3): sử dụng mô hình học máy cơ bản nhưng có khả

năng học tương tác đặc trưng (đặc biệt *rừng ngẫu nhiên* (random forest)) giúp mô hình hoá tốt hơn các trường hợp dấu hiệu webshell không nằm ở một từ khoá đơn lẻ mà nằm ở tổ hợp điều kiện [7].

(iii) Về đánh giá (Chương 4): do webshell có thể né bằng nhiều cách, chỉ số tổng hợp như accuracy không đủ; cần báo cáo precision/recall/F1 cùng ma trận nhầm lẫn và các đường cong ROC/PR để phân tích rõ đánh đổi giữa báo động giả và bỏ sót trong bối cảnh an toàn thông tin [6, 8].

Nhờ đó, Mục 1.2 tạo cầu nối trực tiếp từ bối cảnh tổng quan (Chương 1) sang các bước xử lý dữ liệu (Chương 2), xây dựng mô hình (Chương 3) và đánh giá (Chương 4) của luận văn.

1.2.8. Phân loại webshell và hàm ý đối với nghiên cứu

Trong thực tế, webshell không chỉ khác nhau về ngôn ngữ lập trình (PHP, ASP, JSP) mà còn rất đa dạng về kích thước, cấu trúc và cách thức hoạt động. Sự đa dạng này làm cho việc phát hiện webshell trở nên khó khăn, bởi cùng một phương pháp phát hiện có thể đạt hiệu quả với nhóm webshell này nhưng không phù hợp với nhóm khác. Do đó, việc phân loại webshell giúp lựa chọn hướng tiếp cận và bộ đặc trưng phù hợp cho bài toán phát hiện.

Một cách phân loại phổ biến là dựa trên độ dài và mức độ hoàn chỉnh của mã nguồn. Theo đó, webshell có thể được chia thành ba nhóm gồm: *one-liner backdoor*, *short backdoor* và *large backdoor* [1,7]. Webshell dạng *one-liner* thường chỉ gồm một câu lệnh thực thi nên rất ngắn gọn và khó phát hiện bằng quan sát trực tiếp. Webshell dạng *short* bổ sung thêm các cơ chế xác thực hoặc che giấu mã, trong khi webshell dạng *large* có thể tích hợp nhiều chức năng quản trị như quản lý tệp, thực thi lệnh hoặc truy cập cơ sở dữ liệu. Sự khác biệt lớn về kích thước và cấu trúc khiến các phương pháp phát hiện dựa trên chữ ký hoặc đặc trưng văn bản đơn thuần gặp nhiều hạn chế.

Bên cạnh đó, webshell cũng có thể được phân loại theo mức độ tương tác thành hai nhóm: *thụ động* (passive) và *chủ động* (active). Webshell thụ động chỉ thực thi khi nhận được yêu cầu hoặc tham số kích hoạt phù hợp, trong khi webshell chủ động có thể thực hiện thêm các hành vi nhằm duy trì quyền truy cập hoặc hỗ trợ các bước tấn công tiếp theo [1]. Mặc dù luận văn tập trung vào phân tích tĩnh, cách phân loại này giúp lý giải yêu cầu giảm số lượng webshell bị bỏ sót trong quá trình đánh giá mô hình.

Từ các phân tích trên có thể thấy webshell có tính đa dạng cao về kích thước,

cấu trúc và hành vi. Vì vậy, luận văn lựa chọn hướng tiếp cận dựa trên các đặc trưng tĩnh kết hợp học máy nhằm khai thác đồng thời nhiều dấu hiệu của mã nguồn thay vì chỉ dựa trên một số mẫu chữ ký cố định. Cách tiếp cận này phù hợp với bộ dữ liệu sử dụng trong nghiên cứu và tạo cơ sở cho việc xây dựng bộ đặc trưng, huấn luyện mô hình và đánh giá kết quả ở các chương tiếp theo.

1.2.9. Các kỹ thuật né tránh phát hiện webshell

Để tránh bị phát hiện, webshell thường sử dụng nhiều kỹ thuật che giấu mã nguồn và hành vi thực thi. Các kỹ thuật này làm giảm hiệu quả của các phương pháp phát hiện dựa trên chữ ký hoặc từ khóa cố định, đồng thời làm tăng độ khó của bài toán phát hiện webshell.

Một trong những kỹ thuật phổ biến nhất là *mã hóa và làm rối mã nguồn* (encoding/obfuscation), trong đó các chuỗi hoặc đoạn mã được mã hóa, chia nhỏ hoặc biến đổi nhằm che giấu ý nghĩa thực sự của chương trình. Các kỹ thuật này làm thay đổi phân bố ký tự, độ dài chuỗi và entropy của mã nguồn, khiến việc phát hiện bằng các quy tắc đơn giản trở nên khó khăn hơn [1, 7].

Bên cạnh đó, webshell có thể sử dụng các kỹ thuật *che giấu lời gọi hàm* (function call bypass), chẳng hạn thông qua hàm tự định nghĩa, tệp *include* hoặc các lớp trung gian nhằm tránh sử dụng trực tiếp các hàm nhạy cảm như `eval`, `system` hay `exec` [3]. Ngoài ra, việc thay đổi cú pháp, chèn mã dư thừa hoặc thay đổi cấu trúc chương trình cũng là những kỹ thuật thường gặp nhằm qua mặt các công cụ phát hiện dựa trên mẫu văn bản [7].

Sự khác biệt lớn về kích thước giữa các loại webshell, từ các webshell một dòng cho đến các webshell có đầy đủ chức năng quản trị, cũng làm giảm hiệu quả của các phương pháp phát hiện chỉ dựa trên một nhóm đặc trưng hoặc một quy tắc cố định. Điều này cho thấy việc kết hợp nhiều nhóm đặc trưng khác nhau sẽ giúp mô hình có khả năng thích nghi tốt hơn trước các biến thể webshell.

Từ các phân tích trên có thể thấy không tồn tại một dấu hiệu đơn lẻ đủ khả năng phát hiện hiệu quả mọi loại webshell. Vì vậy, luận văn lựa chọn hướng tiếp cận dựa trên tập hợp các đặc trưng tĩnh phản ánh nhiều khía cạnh của mã nguồn, sau đó sử dụng các mô hình học máy để khai thác đồng thời các đặc trưng này trong quá trình phân loại. Hướng tiếp cận này tạo cơ sở cho việc xây dựng bộ dữ liệu đặc trưng ở Chương 2, xây dựng mô hình ở Chương 3 và đánh giá kết quả ở Chương 4.

1.3. Tổng quan các phương pháp phát hiện webshell

1.3.1. Khung phân loại phương pháp và tiêu chí so sánh

Các phương pháp phát hiện webshell có thể được phân loại theo nhiều tiêu chí. Trong phạm vi luận văn, một khung phân loại thực dụng và thường dùng là dựa trên: (i) *dựa trên chữ ký/luật* (signature/rule-based), (ii) *phân tích tĩnh theo chương trình* (program-analysis-based static detection, ví dụ *taint analysis* (taint analysis)), (iii) *học máy dựa trên đặc trưng* (feature-based machine learning), (iv) *học sâu* (deep learning) trên các biểu diễn mã/biểu diễn trung gian, và (v) *phân tích động* (dynamic analysis) hoặc *xám động* (gray-box) khi quan sát hành vi thực thi. Khảo sát hệ thống hoá cho thấy các hướng này thường được kết hợp trong thực tế để bù trừ ưu/nhược điểm, do webshell có nhiều hình thái và nhiều chiến lược né tránh [1].

Để so sánh, luận văn sử dụng các tiêu chí: *khả năng tổng quát hoá* (generalization) trước biến thể mới, *chi phí triển khai* (deployment cost), *khả năng diễn giải* (interpretability), *mức độ phụ thuộc dữ liệu* (data dependency) và *bề mặt tấn công né tránh* (evasion surface). Các tiêu chí này cũng là cơ sở để luận văn lựa chọn hướng *phân tích tĩnh* (static analysis) kết hợp *học máy cơ bản* (basic machine learning) ở các chương sau.

1.3.2. Phát hiện dựa trên chữ ký/luật (signature/rule-based)

Phương pháp *dựa trên chữ ký* (signature-based) và *dựa trên luật* (rule-based) là hướng tiếp cận truyền thống và phổ biến trong vận hành. Cơ chế cốt lõi thường là: so khớp mẫu chuỗi, so khớp biểu thức chính quy, hoặc áp dụng các luật heuristic để đánh dấu những đoạn mã có dấu hiệu nguy hiểm (ví dụ xuất hiện token/hàm nhạy cảm, chuỗi được giải mã rồi thực thi). Ưu điểm của hướng này là đơn giản, chạy nhanh và hiệu quả với các mẫu đã biết. Khảo sát hệ thống hoá ghi nhận nhiều công cụ vận hành vẫn dựa vào chữ ký do tính dễ triển khai và phù hợp với quy trình phản ứng sự cố [1].

Tuy nhiên, nhược điểm lớn nhất là dễ bị né tránh bởi *làm rối* (obfuscation) và biến thể cú pháp. Chỉ cần thay đổi cách viết, chèn comment hoặc đóng gói lời gọi trong hàm tự định nghĩa, webshell có thể vượt qua nhiều bộ luật dựa trên mẫu bề mặt. Điều này đặc biệt rõ trong bối cảnh PHP, nơi include và biến thể cú pháp có thể làm "mù" cơ chế dò đơn giản. Vì vậy, hướng chữ ký/luật thường phù hợp như tầng lọc đầu (first-line filter), nhưng khó đảm bảo khả năng tổng quát hoá trước biến thể mới [1,3].

1.3.3. Phát hiện dựa trên phân tích tĩnh theo chương trình (program analysis)

Taint analysis (taint analysis) và phát hiện dựa trên luồng dữ liệu

Phân tích luồng dữ liệu nhiễm bẩn (taint analysis) là một nhánh quan trọng của *phân tích tĩnh* (static analysis), tập trung vào việc truy vết dữ liệu từ nguồn không tin cậy (ví dụ tham số HTTP) đến các điểm nhạy cảm (sink) như hàm thực thi lệnh hoặc thực thi mã. Nếu luồng dữ liệu "bẩn" đi đến sink mà không qua làm sạch phù hợp, đoạn mã bị xem là có nguy cơ webshell hoặc tấn công webshell. WTA là một ví dụ tiêu biểu áp dụng taint analysis cho PHP webshell và nhấn mạnh nhu cầu *phân tích liên thủ tục* (interprocedural analysis) để xử lý include và hàm tự định nghĩa [3, 9]. Ngoài ra, một khung taint analysis tĩnh khác cũng được đề xuất nhằm phát hiện tấn công dựa trên PHP webshell và thảo luận cách xây dựng chuỗi nguồn–sink trong ngữ cảnh web [10].

Ưu điểm của hướng program analysis là có cơ hội phát hiện các biến thể chưa gặp trước đó, vì dựa trên quan hệ ngữ nghĩa (semantic relation) thay vì mẫu chuỗi. Tuy nhiên, hạn chế là độ phức tạp triển khai và chi phí tính toán cao hơn; ngoài ra, phân tích tĩnh đầy đủ trong PHP gặp khó do tính động của ngôn ngữ (dynamic features) và việc thiếu thông tin khi phân tích không chạy chương trình. Trên thực tế, hướng này thường được dùng trong các công cụ chuyên sâu hoặc kết hợp với các tầng lọc khác [1, 3].

Phân tích chương trình cho webshell không tệp và webshell đa ngôn ngữ

Mặc dù luận văn tập trung vào PHP, cần lưu ý rằng webshell tồn tại ở nhiều ngôn ngữ (JSP, ASP,...) và có cả dạng *không tệp* (fileless) trong một số hệ sinh thái. Ví dụ, JShellDetector tập trung vào Java fileless webshell và áp dụng *phân tích chương trình* (program analysis) để phát hiện hành vi webshell khi payload không tồn tại dưới dạng tệp độc lập [11]. Các hướng như vậy cho thấy program analysis có vai trò quan trọng trong các môi trường mà chữ ký tệp không đủ, đồng thời gợi ý rằng thiết kế phát hiện cần xét cả bối cảnh ngôn ngữ và hình thái lưu trữ mã độc.

1.3.4. Phát hiện dựa trên học máy trong dữ liệu log và phiên làm việc

Bên cạnh phân tích mã nguồn, một hướng tiếp cận khác là phát hiện webshell dựa trên dữ liệu vận hành như log truy cập và phiên làm việc. Phương pháp này thường coi hành vi truy cập webshell tạo ra dấu vết khác thường (ví dụ chuỗi request/response, pattern tham số, tần suất truy cập), từ đó trích đặc trưng từ log và áp dụng học máy.

Nghiên cứu về phát hiện theo phiên (*session-based* (session-based)) sử dụng học máy trên web logs là ví dụ tiêu biểu cho hướng này [12].

Ưu điểm của phát hiện dựa trên log là có thể bắt được hành vi tấn công ngay cả khi mã webshell bị che giấu tốt trong hệ thống file. Tuy nhiên, nhược điểm là phụ thuộc mạnh vào chất lượng log, cấu hình ghi log và bối cảnh vận hành. Ngoài ra, một webshell thụ động hoặc sử dụng kênh điều khiển tinh vi có thể tạo ra ít dấu vết hoặc trộn lẫn với hành vi quản trị hợp lệ. Trong phạm vi luận văn, hướng log-based được xem là tham chiếu quan trọng, nhưng luận văn chọn hướng phân tích tĩnh mã nguồn để phù hợp mục tiêu xây dựng pipeline từ dữ liệu mã (Chương 2) đến mô hình và đánh giá (Chương 3–4).

1.3.5. Phát hiện dựa trên học máy theo đặc trưng (feature-based machine learning)

Học máy dựa trên đặc trưng (feature-based machine learning) là hướng tiếp cận trung gian giữa chữ ký và program analysis: thay vì so khớp mẫu cố định, phương pháp này trích một vector đặc trưng từ mã nguồn/biểu diễn trung gian và huấn luyện mô hình phân loại. Đặc trưng có thể gồm: đặc trưng thống kê (entropy, độ dài), đặc trưng tần suất token/hàm, hoặc đặc trưng phản ánh tính "thực thi được" của mã. Một ví dụ là nghiên cứu dựa trên *đặc tính dữ liệu thực thi* (executable data characteristics) của PHP code để phát hiện webshell, cho thấy việc thiết kế đặc trưng phù hợp có thể tăng khả năng phân biệt giữa mã lành tính và mã độc [2, 13].

Ưu điểm của hướng feature-based ML là: (i) tương đối dễ triển khai và mở rộng, (ii) phù hợp với quét diện rộng vì không cần chạy mã, và (iii) có thể đạt hiệu quả tốt nếu bộ đặc trưng đủ đại diện cho dấu hiệu webshell. Nhược điểm là chất lượng mô hình phụ thuộc mạnh vào cách trích đặc trưng và tính đại diện của dữ liệu huấn luyện; nếu đặc trưng quá "trực tiếp" (gần như chữ ký), mô hình có thể gặp nguy cơ *đẹp giả* do *độ rò rỉ nhãn* (label leakage). Đây là lý do luận văn ở Chương 2 nhấn mạnh kiểm định dữ liệu và ở Chương 4 dùng nhiều chỉ số để đánh giá khách quan.

1.3.6. Phát hiện dựa trên học sâu (deep learning) và biểu diễn mã

Học sâu trên biểu diễn opcode/bytecode và đặc trưng trung gian

Một nhánh học sâu đáng chú ý là sử dụng *biểu diễn trung gian* (intermediate representation) như opcode/bytecode để giảm phụ thuộc vào biến thể cú pháp và tăng tính ổn định trước obfuscation. Nghiên cứu SoICT 2019 đề xuất pipeline chuyển PHP sang opcode rồi huấn luyện mô hình học sâu để phân loại webshell [4]. Tương tự, một nghiên cứu khác sử dụng *đặc trưng bytecode* (bytecode features) kết hợp mạng CNN

cho phát hiện webshell [14]. Các hướng này gợi ý rằng việc trừu tượng hoá mã về biểu diễn thấp hơn có thể giúp mô hình học sâu bắt được cấu trúc thực thi tốt hơn so với văn bản thuần.

Học sâu theo chuỗi và cơ chế chú ý (sequence models and attention)

Ngoài CNN, các mô hình theo chuỗi (sequence models) như RNN/GRU/LSTM và cơ chế *chú ý* (attention) cũng được áp dụng. Một ví dụ là phương pháp dựa trên *Bidirectional GRU* (BiGRU) và attention cho phát hiện webshell [15]. Ưu điểm của hướng này là khả năng mô hình hóa phụ thuộc theo ngữ cảnh, tuy nhiên nhược điểm là khó xử lý tệp rất dài và dễ bị ảnh hưởng bởi "nhiễu" nếu không có cơ chế chọn vùng quan trọng.

Học sâu dựa trên mô hình tiền huấn luyện mã (CodeBERT)

Gần đây, các mô hình tiền huấn luyện cho mã nguồn như CodeBERT được khai thác để cải thiện biểu diễn ngữ nghĩa của mã. Một nghiên cứu dựa trên CodeBERT và mô hình học sâu cho phát hiện WebShell là ví dụ cho hướng này [16]. Nhìn chung, các hướng deep learning và pre-trained models có tiềm năng nâng hiệu năng, nhưng thường yêu cầu tài nguyên tính toán và thiết kế dữ liệu/huấn luyện phức tạp hơn. Vì yêu cầu luận văn theo hướng ứng dụng cơ bản và khả năng giải trình trước hội đồng, luận văn hiện tại chọn hướng feature-based ML và dùng deep learning như nền tham khảo trong tổng quan.

1.3.7. Phát hiện dựa trên phân tích động/xám động (dynamic/gray-box) và hệ thống hoá triển khai

Phân tích động (dynamic analysis) quan sát hành vi khi webshell được thực thi (ví dụ sandbox, hook hệ thống, theo dõi I/O), trong khi *xám động* (gray-box) kết hợp quan sát hành vi với một phần thông tin nội bộ/đặc trưng môi trường để tăng hiệu quả. SmartEagleEye là một hệ thống phát hiện webshell theo hướng *gray-box động* (dynamic gray-box) kết hợp học sâu, đặt trong bối cảnh môi trường đám mây [17]. Các hướng này cho thấy trong vận hành quy mô lớn, phát hiện webshell không chỉ là vấn đề mô hình mà còn là vấn đề hệ thống (thu thập tín hiệu, đồng bộ log, phản ứng sự cố).

Trong luận văn, hướng dynamic/gray-box được trình bày như tham chiếu quan trọng, nhưng không phải trọng tâm triển khai do yêu cầu môi trường và chi phí vận hành. Luận văn tập trung vào *phân tích tĩnh* kết hợp ML để có thể trình bày quy trình

rõ ràng, tái lập và phù hợp năng lực triển khai trên tập dữ liệu mã nguồn (Chương 2–4).

1.3.8. Sử dụng mô hình ngôn ngữ lớn (LLM) cho phát hiện webshell

Xu hướng gần đây là ứng dụng *mô hình ngôn ngữ lớn* (large language model – LLM) để hiểu và phân loại mã độc/webshell. Một công trình đặt câu hỏi liệu LLM có xử lý được bài toán webshell detection hay không và đề xuất khung nhận thức theo hành vi hàm (behavioral function-aware) để vượt qua các thách thức như độ dài tệp và nhiễu không liên quan [5]. Hướng này cho thấy tiềm năng kết hợp hiểu ngữ nghĩa và lập luận theo bối cảnh, nhưng đồng thời cũng nhấn mạnh rào cản thực tiễn: giới hạn ngữ cảnh và yêu cầu trích chọn đoạn mã liên quan.

Trong phạm vi luận văn, LLM được trình bày như hướng phát triển tương lai. Luận văn hiện tại ưu tiên một pipeline có tính tái lập cao và dễ giải trình dựa trên đặc trưng và mô hình ML cơ bản.

Kết luận

Từ tổng quan trên có thể rút ra một kết luận quan trọng: các phương pháp phát hiện webshell tạo thành một phổ lựa chọn từ chữ ký/luật (đơn giản nhưng dễ né) đến program analysis/dynamic analysis (mạnh hơn nhưng phức tạp), và các phương pháp ML/Deep Learning nằm ở giữa với mức đánh đổi khác nhau. Để phù hợp mục tiêu luận văn thạc sĩ theo hướng ứng dụng cơ bản và đảm bảo tính liên thông giữa các chương, luận văn lựa chọn hướng *phân tích tĩnh* (static analysis) dựa trên *đặc trưng* (feature-level dataset) và áp dụng mô hình ML cơ bản.

Cụ thể: Chương 2 hiện thực hoá hướng feature-based ML bằng cách sử dụng tập dữ liệu CSV đặc trưng và thiết kế pipeline tiền xử lý; Chương 3 triển khai và lựa chọn mô hình theo validation (baseline *hồi quy logistic* (logistic regression) và mô hình mạnh hơn *rừng ngẫu nhiên* (random forest)); và Chương 4 đánh giá khách quan trên test bằng ma trận nhầm lẫn và các chỉ số/đường cong ROC–PR, tương ứng với các tiêu chí đánh giá thường dùng trong các nghiên cứu ML về phát hiện webshell [2, 4, 15].

Bảng 1.3.1. Bảng tổng hợp các hướng phát hiện webshell và liên hệ với nội dung luận văn.

Nhóm phương pháp (Category)	*Ý tưởng chính* (Key idea)	*Ưu/nhược điểm* (Pros/Cons)	*Liên hệ với luận văn* (Link to Chapters 2–4)
Chữ ký/luật (signature/rule-based)	So khớp mẫu chuỗi/regex, luật heuristic theo từ khóa/hàm nhạy cảm. [1]	Nhanh, dễ triển khai; nhưng dễ né bởi *làm rối* (obfuscation), *mã hoá* (encoding), biến thể cú pháp.	Chương 2 dùng đặc trưng đa dạng để giảm phụ thuộc chữ ký; Chương 4 đánh giá theo nhiều chỉ số để tránh "đẹp giả".
Taint analysis (taint analysis)	Truy vết dữ liệu từ nguồn không tin cậy tới điểm nhạy cảm (sink), phát hiện luồng dữ liệu nguy hiểm. [3, 10]	Có khả năng tổng quát hoá tốt hơn chữ ký; nhưng phức tạp triển khai, chi phí phân tích cao, khó do tính động của PHP.	Luận văn không triển khai taint analysis đầy đủ, nhưng dùng đặc trưng phản ánh dấu hiệu gián tiếp (entropy/độ dài/hàm nhạy cảm) và ML cơ bản để đạt tính ứng dụng.
Phân tích chương trình (program analysis)	Phân tích cấu trúc/luồng thực thi, có thể áp dụng cho webshell không tệp hoặc đa ngôn ngữ. [11]	Mạnh trong một số bối cảnh đặc thù; nhưng thường yêu cầu mô hình hoá sâu và phụ thuộc môi trường/ngôn ngữ.	Luận văn tập trung PHP và feature-level; mục này làm tham chiếu học thuật để định vị phạm vi nghiên cứu.

(Còn tiếp ở trang sau)

Bảng 1.3.1. Bảng tổng hợp các hướng phát hiện webshell và liên hệ với nội dung luận văn (tiếp).

Nhóm phương pháp (Category)	*Ý tưởng chính* (Key idea)	*Ưu/nhược điểm* (Pros/Cons)	*Liên hệ với luận văn* (Link to Chapters 2–4)
ML trên log/phiên (log/session-based ML)	Trích đặc trưng hành vi từ web logs/phiên truy cập rồi huấn luyện mô hình phân loại. [12]	Bắt được hành vi kể cả khi mã bị che giấu; nhưng phụ thuộc chất lượng log, dễ nhiễu, khó áp dụng khi log thiếu.	Luận văn chọn *phân tích tĩnh* (static) để phù hợp mục tiêu xây dựng pipeline từ mã nguồn; các chỉ số đánh giá ở Chương 4 vẫn tương tự (precision/recall/F1).
ML theo đặc trưng (feature-based ML)	Chuyển mã thành vector *đặc trưng* (features) (thống kê, độ dài, dấu hiệu thực thi, đặc tính trung gian). [2]	Dễ triển khai, quét diện rộng; nhưng phụ thuộc chất lượng feature và dataset; có nguy cơ *rò rỉ nhãn* (label leakage) nếu feature quá trực tiếp.	Đây là hướng chính của luận văn: Chương 2 xây dựng/tiền xử lý dataset CSV; Chương 3 huấn luyện LR/RF; Chương 4 đánh giá test bằng confusion/ROC/PR.
Học sâu trên opcode/bytecode (DL on opcode/bytecode)	Dùng opcode/bytecode để giảm ảnh hưởng biến thể cú pháp, huấn luyện CNN/RNN trên chuỗi/ma trận biểu diễn. [4, 14]	Có tiềm năng tăng hiệu năng; nhưng cần dữ liệu lớn, tài nguyên tính toán, khó giải trình hơn.	Luận văn dùng ML cơ bản vì yêu cầu thạc sĩ ứng dụng; các nghiên cứu này làm nền so sánh và định hướng mở rộng tương lai.

(Còn tiếp ở trang sau)

Bảng 1.3.1. Bảng tổng hợp các hướng phát hiện webshell và liên hệ với nội dung luận văn (tiếp).

Nhóm phương pháp (Category)	*Ý tưởng chính* (Key idea)	*Ưu/nhược điểm* (Pros/Cons)	*Liên hệ với luận văn* (Link to Chapters 2–4)
Học sâu theo chuỗi + chú ý (sequence + attention)	Mô hình hoá mã/biểu diễn trung gian như chuỗi (Bi-GRU/LSTM) kết hợp *chú ý* (attention). [15]	Bắt ngữ cảnh tốt; nhưng khó với tệp rất dài và dễ bị nhiễu nếu không có cơ chế chọn vùng.	Luận văn ưu tiên pipeline đơn giản, tái lập; ROC/PR ở Chương 4 vẫn là chuẩn để so sánh với các hướng DL trong tương lai.
Học sâu + hệ thống xám động (dynamic gray-box + DL)	Kết hợp tín hiệu hành vi thực thi với mô hình học sâu trong hệ thống triển khai (cloud-oriented). [17]	Mạnh trong vận hành, nhưng chi phí triển khai cao, phụ thuộc môi trường, khó tái lập nếu thiếu hệ thống.	Luận văn chọn *phân tích tĩnh* để tái lập; mục này giúp nêu rõ khác biệt giữa nghiên cứu mô hình và triển khai hệ thống.
Mô hình ngôn ngữ lớn (LLM-based)	Dùng *LLM* để hiểu ngữ nghĩa và suy luận hành vi; cần xử lý giới hạn ngữ cảnh, chọn vùng mã quan trọng. [5]	Tiềm năng mạnh, nhưng chi phí và rủi ro dài/ngữ cảnh; khó áp dụng nếu không có pipeline chọn lọc.	Luận văn xem như hướng tương lai; các artifacts (pipeline + test evaluation) của luận văn là nền để so sánh khi mở rộng.
Khung nhẹ theo miền đặc thù (light-weight / domain-specific)	Thiết kế framework "nhẹ" cho các môi trường đặc thù (ví dụ mạng vệ tinh/UAV). [18]	Tối ưu tài nguyên, có ràng buộc miền; nhưng có thể kém tổng quát cho môi trường web thông thường.	Dùng làm tham chiếu cho tiêu chí triển khai (chi phí suy luận), liên hệ với lựa chọn RF/LR ở Chương 3.

Từ Bảng 1.3.1, có thể thấy hướng *học máy theo đặc trưng* (feature-based ML) là lựa chọn phù hợp để đảm bảo tính tái lập và tính ứng dụng ở mức cơ bản. Do đó, Chương 2 trình bày quy trình chuẩn hoá dữ liệu và tiền xử lý, Chương 3 xây dựng mô hình phân loại, và Chương 4 đánh giá khách quan trên tập test bằng các chỉ số và đồ thị chuẩn.

1.4. Nhận xét tổng hợp và lựa chọn hướng nghiên cứu

1.4.1. Nhận xét tổng hợp từ tổng quan phương pháp

Từ tổng quan ở Mục 1.3 và Bảng 1.3.1, có thể rút ra một số nhận xét quan trọng về bài toán phát hiện webshell trong PHP.

Thứ nhất, các phương pháp *dựa trên chữ ký/luật* (signature/rule-based) vẫn giữ vai trò thực tiễn do đơn giản và hiệu quả với mẫu đã biết, nhưng điểm yếu lớn là dễ bị né tránh bởi *làm rối* (obfuscation), *mã hoá/biến đổi biểu diễn* (encoding/encryption) và biến thể cú pháp, đặc biệt trong PHP với cơ chế include và hàm tự định nghĩa [1,3]. Điều này cho thấy nếu chỉ dựa vào chữ ký, hệ thống dễ rơi vào trạng thái "đuổi theo" biến thể mới và có nguy cơ bỏ sót các mẫu tinh vi.

Thứ hai, các hướng *phân tích tĩnh theo chương trình* (program analysis) như *taint analysis* (taint analysis) có tiềm năng tổng quát hoá tốt hơn do dựa trên quan hệ ngữ nghĩa và luồng dữ liệu, ví dụ WTA nhấn mạnh truy vết từ nguồn không tin cậy tới sink trong bối cảnh PHP webshell [3]. Tuy nhiên, việc triển khai đầy đủ thường đòi hỏi mô hình hoá ngôn ngữ sâu, xử lý tính động của PHP và có thể tốn kém tài nguyên; do đó, hướng này phù hợp với các công cụ chuyên sâu hoặc hệ thống phân tích bảo mật quy mô lớn hơn là một pipeline ứng dụng cơ bản.

Thứ ba, các phương pháp *học sâu* (deep learning) và các biểu diễn mã nâng cao (opcode/bytecode, mô hình chuỗi, mô hình tiền huấn luyện) cho thấy hiệu năng tiềm năng trong nhiều nghiên cứu [4,14–16]. Tuy nhiên, chúng thường yêu cầu dữ liệu lớn, chi phí huấn luyện cao và khó giải trình hơn. Với yêu cầu luận văn theo hướng ứng dụng học máy cơ bản, việc lựa chọn học sâu làm trọng tâm có thể làm tăng rủi ro khi giải trình trước hội đồng nếu không có nền tảng và tài nguyên thực nghiệm tương ứng.

Thứ tư, các hệ thống *phân tích động/xám động* (dynamic/gray-box) cho thấy hướng triển khai thực tế có thể rất mạnh khi kết hợp đa nguồn tín hiệu, nhưng đòi hỏi bối cảnh hệ thống (hạ tầng log, sandbox, phản ứng sự cố) và thường khó tái lập trong phạm vi luận văn cá nhân [17]. Tương tự, các hướng mới như *mô hình ngôn ngữ lớn*

(large language model – LLM) mở ra triển vọng hiểu mã theo ngữ nghĩa, nhưng vẫn gặp thách thức về độ dài, ngữ cảnh và cơ chế chọn lọc vùng mã liên quan [5].

Các nghiên cứu gần đây cũng bắt đầu khai thác học tăng dần (incremental learning) nhằm nâng cao khả năng thích nghi với các biến thể webshell mới [19].

Từ các phân tích trên có thể thấy rằng mỗi hướng tiếp cận đều có những ưu điểm và hạn chế riêng. Các phương pháp dựa trên chữ ký có tốc độ xử lý nhanh nhưng khó phát hiện các biến thể webshell mới; các phương pháp phân tích chương trình và phân tích động có khả năng phát hiện tốt hơn nhưng chi phí triển khai cao; trong khi các mô hình học sâu đòi hỏi tập dữ liệu lớn và tài nguyên tính toán đáng kể. Trong phạm vi các tài liệu được khảo sát, việc đánh giá các mô hình học máy truyền thống [13] có khả năng diễn giải tốt trên bộ dữ liệu đặc trưng webshell PHP công khai chưa được đề cập nhiều. Đây chính là khoảng trống mà luận văn tập trung nghiên cứu thông qua việc sử dụng các đặc trưng tĩnh kết hợp hai mô hình Logistic Regression và Random Forest để đánh giá khả năng phát hiện webshell.

1.4.2. Lựa chọn hướng nghiên cứu của luận văn

Từ các nhận xét tổng hợp trên, luận văn định vị hướng nghiên cứu theo nguyên tắc: ưu tiên một quy trình *tái lập* (reproducible), *dễ triển khai* (deployable) và *dễ giải trình* (explainable) ở mức cơ bản, đồng thời vẫn đủ sức phản ánh các dấu hiệu né tránh phổ biến của webshell PHP.

Cụ thể, luận văn lựa chọn hướng *phân tích tĩnh* (static analysis) kết hợp *học máy theo đặc trưng* (feature-based machine learning). Hướng này có ba điểm phù hợp:

(i) Có thể xử lý mã nguồn ở quy mô lớn mà không cần thực thi chương trình, phù hợp với quy trình kiểm tra mã và rà soát an toàn.

(ii) Cho phép thiết kế tập đặc trưng phản ánh đồng thời dấu hiệu trực tiếp (liên quan hàm nhạy cảm/từ khoá hành vi) và dấu hiệu gián tiếp (thống kê độ dài/entropy/độ phức tạp), giúp giảm phụ thuộc vào chữ ký thuần tuý và tăng khả năng phát hiện trước biến thể làm rối [2, 6].

(iii) Có thể áp dụng các mô hình ML cơ bản như *hồi quy logistic* (logistic regression) và *rừng ngẫu nhiên* (random forest) để cân bằng giữa hiệu năng và tính giải trình; đồng thời đánh giá khách quan theo các chỉ số chuẩn trong phân loại nhị phân như precision/recall/F1 và ROC/PR [8, 15].

Với định vị này, luận văn không nhằm thay thế các hệ thống phân tích động hoặc các phương pháp program analysis chuyên sâu, mà tập trung đề xuất và kiểm chứng một pipeline ứng dụng cơ bản, rõ ràng, có thể triển khai và tái lập trên dữ liệu dạng đặc trưng.

1.4.3. Mục đích nghiên cứu của luận văn

Dựa trên khảo sát và so sánh, có thể mô tả khoảng trống nghiên cứu như sau: nhiều công trình hoặc tập trung vào chữ ký/luật, hoặc tập trung vào các mô hình nâng cao (deep learning, hệ thống động) yêu cầu tài nguyên và độ phức tạp lớn. Trong khi đó, vẫn cần một quy trình ứng dụng ở mức cơ bản nhưng có tính chặt chẽ về dữ liệu, đánh giá và tái lập, nhằm phục vụ bối cảnh đào tạo và triển khai ban đầu.

Trong phạm vi luận văn, các đóng góp chính được mô tả như sau:

(i) Xây dựng quy trình chuẩn hoá dữ liệu dạng *đặc trưng* (feature-level dataset), kiểm định chất lượng dữ liệu và thiết kế chia train/validation/test nhất quán (Chương 2).

(ii) Xây dựng và lựa chọn mô hình ML cơ bản dựa trên tập validation, đảm bảo tránh *rò rỉ dữ liệu* (data leakage) và hạn chế *quá khớp* (overfitting) (Chương 3).

(iii) Đánh giá khách quan trên tập test bằng các chỉ số định lượng và hình minh hoạ chuẩn (confusion matrix, ROC, PR), từ đó đưa ra kết luận mô hình phù hợp trong bối cảnh phát hiện webshell PHP (Chương 4).

1.5. Kết luận

Chương 1 đã trình bày bối cảnh, đặc điểm kỹ thuật và tổng quan các hướng phát hiện webshell, đồng thời phân tích các điểm mạnh/yếu và bề mặt né tránh của từng nhóm phương pháp. Qua tổng hợp, có thể thấy phát hiện webshell là bài toán có tính đa dạng cao về hình thái và chiến lược che giấu; do đó các phương pháp dựa trên chữ ký/luật tuy hữu ích nhưng khó đảm bảo tổng quát hoá trước biến thể mới [1]. Các hướng program analysis và phân tích động có khả năng phát hiện mạnh hơn trong một số bối cảnh, nhưng đi kèm độ phức tạp và chi phí triển khai, trong khi các hướng học sâu và mô hình tiền huấn luyện có tiềm năng nhưng thường đòi hỏi dữ liệu/tài nguyên đáng kể [3, 16, 17].

Trên cơ sở đó, luận văn lựa chọn hướng *phân tích tĩnh* (static analysis) kết hợp *học máy theo đặc trưng* (feature-based machine learning) để đảm bảo tính tái lập, phù hợp yêu cầu ứng dụng học máy cơ bản, và vẫn phản ánh được các dấu hiệu né tránh thường gặp của webshell PHP. Định hướng này được cụ thể hoá trong các chương tiếp

theo như sau:

(i) Chương 2 trình bày quá trình thu thập, kiểm định và tiền xử lý tập dữ liệu dạng CSV, xây dựng pipeline chuẩn hoá và chia train/validation/test theo *lấy mẫu phân tầng* (stratified sampling). Đây là nền tảng để đảm bảo các thí nghiệm có thể tái lập và các mô hình được so sánh công bằng.

(ii) Chương 3 xây dựng các mô hình học máy cơ bản trên tập train và lựa chọn tham số dựa trên tập validation. Việc lựa chọn mô hình dựa trên tiêu chí phù hợp (ví dụ *F1-score* (F1-score) cho lớp dương `webshell`) giúp cân bằng giữa giảm bỏ sót và giảm báo động giả.

(iii) Chương 4 mô phỏng và đánh giá cuối trên tập test độc lập, báo cáo các chỉ số định lượng (accuracy, precision, recall, F1-score, ROC-AUC, PR-AUC) và minh hoạ bằng confusion matrix cùng các đường cong ROC/PR. Trên cơ sở đó, luận văn đưa ra nhận xét và kết luận về mô hình phù hợp trong phạm vi nghiên cứu, đồng thời nêu các giới hạn và hướng phát triển tương lai.

CHƯƠNG 2. THU THẬP, XỬ LÝ VÀ XÂY DỰNG TẬP DỮ LIỆU

2.1. Giới thiệu và mục tiêu xây dựng tập dữ liệu

Trong bài toán phát hiện webshell trong mã nguồn PHP, chất lượng *tập dữ liệu* (dataset) và quy trình *tiền xử lý* (preprocessing) có vai trò quyết định đối với khả năng huấn luyện, đánh giá và tái lập mô hình. Luận văn này lựa chọn hướng tiếp cận *phân tích tĩnh* (static analysis), trong đó mỗi mẫu mã được biểu diễn dưới dạng một vector *đặc trưng* (feature vector). Trên cơ sở đó, bài toán được mô hình hóa thành *phân loại nhị phân* (binary classification) gồm hai lớp: `webshell` và `normal`.

Mục tiêu của Chương 2 là xây dựng một quy trình xử lý dữ liệu ở mức cơ bản nhưng chặt chẽ, bao gồm: (i) mô tả nguồn dữ liệu và cấu trúc dữ liệu sử dụng trong luận văn, (ii) kiểm định *toàn vẹn dữ liệu* (data integrity) thông qua kiểm tra giá trị thiếu và trùng lặp, (iii) sàng lọc các *đặc trưng* (features) không mang thông tin (ví dụ *cột hằng* (constant columns)), (iv) chuẩn hóa thang đo đặc trưng để phù hợp với các mô hình học máy tuyến tính, và (v) thiết kế chia tập train/validation/test theo *lấy mẫu phân tầng* (stratified sampling) nhằm đảm bảo đánh giá khách quan.

Kết quả của chương này là các *tạo phẩm dữ liệu* (data artifacts) gồm tập dữ liệu đã tiền xử lý và *pipeline* (pipeline) tiền xử lý được lưu lại. Các tạo phẩm này được sử dụng xuyên suốt Chương 3 trong quá trình huấn luyện/lựa chọn mô hình và Chương 4 trong quá trình đánh giá cuối trên tập test.

Tiếp theo, Mục 2.2 trình bày nguồn dữ liệu, cấu trúc file CSV và các thống kê mô tả quan trọng của tập dữ liệu sử dụng trong luận văn.

2.2. Nguồn dữ liệu và mô tả cấu trúc tập dữ liệu

2.2.1. Nguồn dữ liệu và định nghĩa nhãn

Luận văn sử dụng một tập dữ liệu phục vụ bài toán phát hiện webshell trong mã nguồn PHP, được kế thừa từ nguồn dữ liệu công khai và được biểu diễn dưới dạng bảng *đặc trưng* (feature-level dataset) [20]. Mỗi mẫu dữ liệu được gán một nhãn thuộc tập $\{\texttt{webshell}, \texttt{normal}\}$, trong đó `webshell` biểu thị mã có hành vi webshell, còn `normal` biểu thị mã *bình thường* (benign/normal). Việc chuẩn hóa nhãn theo hai lớp giúp mô hình hóa bài toán dưới dạng *phân loại nhị phân* (binary classification), phù hợp với mục tiêu của luận văn và thuận lợi cho việc đánh giá bằng các chỉ số như

độ đúng (precision), *độ bao phủ* (recall) và *F1-score* (F1-score) ở các chương tiếp theo.

2.2.2. Cấu trúc file dataset dạng CSV

Tập dữ liệu sử dụng trong luận văn được lưu dưới dạng file CSV, trong đó mỗi dòng tương ứng một mẫu dữ liệu. Cột `label` là nhãn phân lớp, và các cột còn lại là các *đặc trưng* (features) dạng số (int/float) biểu diễn thuộc tính thống kê và dấu hiệu hành vi của mã nguồn. Dạng biểu diễn theo vector đặc trưng cho phép áp dụng trực tiếp các mô hình học máy cơ bản mà không cần xử lý cú pháp mã nguồn (parsing) hoặc biểu diễn theo chuỗi token ở mức nâng cao.

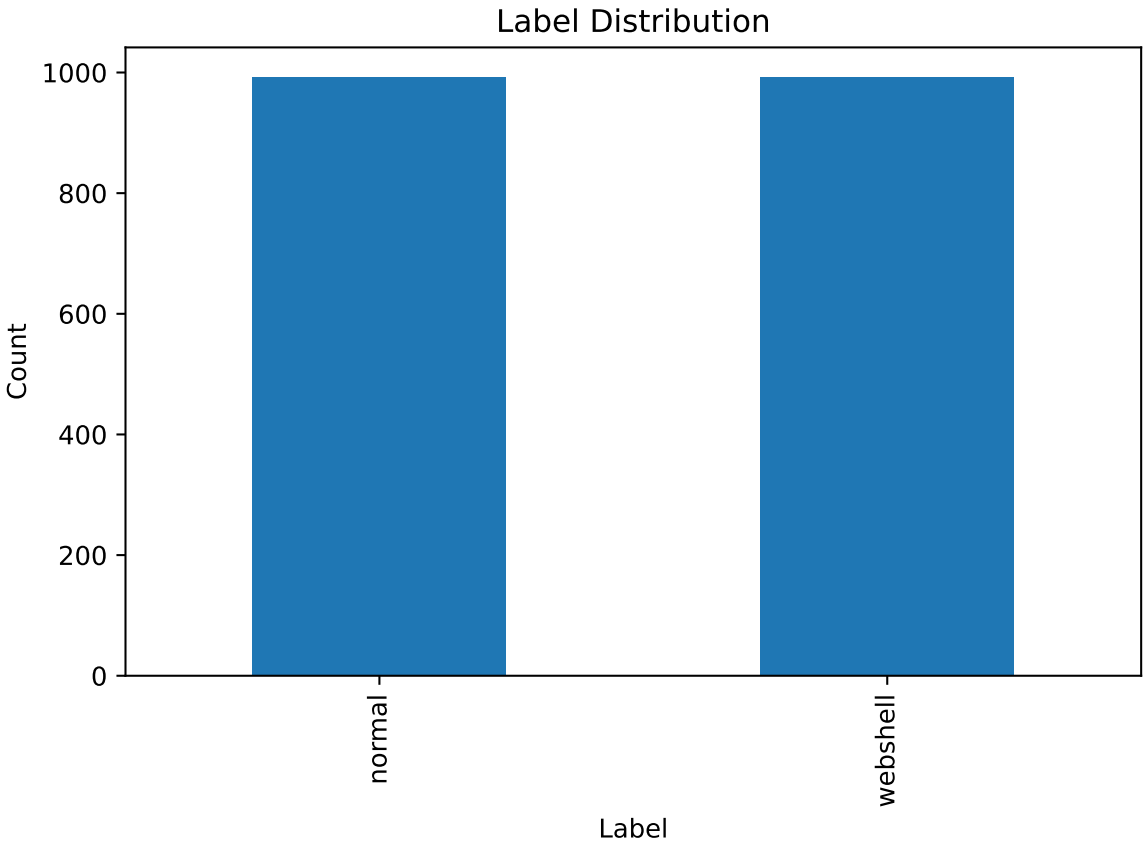
Để đảm bảo tính tái lập, các thống kê tổng quan của tập dữ liệu (số mẫu, số đặc trưng, số lớp, phân bố nhãn, kiểm tra thiếu dữ liệu và trùng lặp) được trích xuất tự động bằng ngôn ngữ lập trình Python và xuất ra dưới dạng bảng hoặc hình vẽ. Bảng 2.2.1 trình bày tổng quan tập dữ liệu CSV sử dụng trong luận văn.

Bảng 2.2.1. Tổng quan tập dữ liệu (dataset) dạng CSV sử dụng trong luận văn.

Thuộc tính (Attribute)	Giá trị (Value)
Số mẫu (Number of samples)	1984
Số cột (Number of columns)	101 (gồm 100 đặc trưng (features) + 1 nhãn (label))
Số lớp (Number of classes)	2
Phân bố nhãn (Class distribution)	normal: 992, webshell: 992
Số ô thiếu (Missing cells)	0
Số dòng trùng (Duplicate rows)	0

2.2.3. Phân bố nhãn và nhận xét ban đầu

Phân bố nhãn là một yếu tố quan trọng, vì dữ liệu mất cân bằng (class imbalance) có thể gây sai lệch khi huấn luyện và làm cho các chỉ số đánh giá trở nên thiếu tin cậy. Trong tập dữ liệu sử dụng, hai lớp `webshell` và `normal` có quy mô tương đương nhau. Hình 2.2.1 minh họa phân bố nhãn của tập dữ liệu và cho thấy dữ liệu được cân bằng theo hai lớp, qua đó thuận lợi cho việc so sánh mô hình ở Chương 3 và đánh giá ở Chương 4.



Hình 2.2.1. Phân bố nhãn `webshell` và `normal` trong tập dữ liệu.

2.3. Phân tích nhóm đặc trưng theo hướng phát hiện webshell

Do dữ liệu trong luận văn được biểu diễn dưới dạng bảng *đặc trưng* (feature-level dataset), trọng tâm của mục này là phân tích ý nghĩa của các nhóm đặc trưng nhằm lý giải cơ sở khoa học cho bài toán phân loại. Mỗi mẫu mã nguồn PHP được ánh xạ thành một vector đặc trưng $X \in \mathbb{R}^d$, trong đó mỗi thành phần phản ánh một khía cạnh của mã nguồn như mức độ phức tạp chuỗi, độ dài, dấu hiệu thực thi, hoặc các chỉ báo liên quan đến hành vi bất thường.

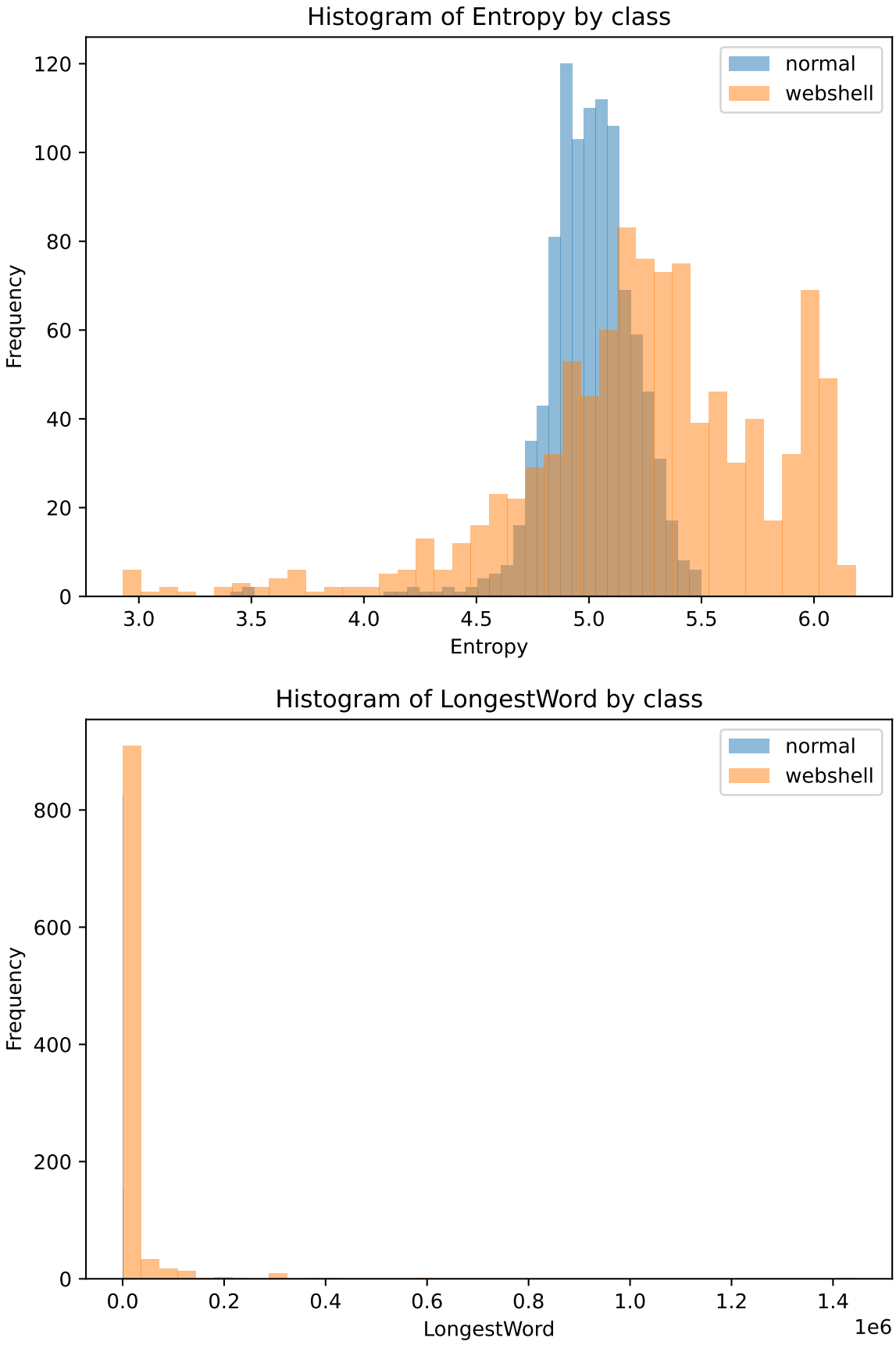
Các nhóm đặc trưng trong tập dữ liệu có thể được phân loại theo hướng trực quan như sau: (i) nhóm đặc trưng thống kê và độ phức tạp chuỗi, (ii) nhóm đặc trưng liên quan đến hàm nhạy cảm và dấu hiệu thực thi, và (iii) nhóm đặc trưng mang tính chữ ký/heuristic. Việc nhóm hóa này giúp luận văn trình bày rõ ràng, đồng thời thuận lợi cho việc diễn giải kết quả ở Chương 4.

2.3.1. Nhóm đặc trưng thống kê và độ phức tạp chuỗi

Nhóm đặc trưng thống kê và độ phức tạp chuỗi phản ánh xu hướng *làm rối* (obfuscation) và che giấu payload thường gặp trong webshell. Trong thực tế, webshell

có thể chứa các chuỗi dài (ví dụ chuỗi mã hóa), các dòng mã có độ dài bất thường, hoặc các ký tự đặc biệt xuất hiện với tần suất cao. Những đặc điểm này làm thay đổi đáng kể phân phối của các đặc trưng như *entropy* (entropy), *tỷ lệ nén* (compression ratio), *độ dài từ lớn nhất* (longest word), hoặc *độ dài dòng lớn nhất* (maximum line length).

Để minh họa sự khác biệt giữa hai lớp `webshell` và `normal`, luận văn sử dụng biểu đồ histogram theo lớp cho một số đặc trưng tiêu biểu. Hình 2.3.1 cho thấy phân phối của các đặc trưng này có xu hướng tách biệt, qua đó tạo cơ sở cho việc sử dụng các mô hình học máy cơ bản trong Chương 3.



Hình 2.3.1. So sánh phân phối theo lớp cho một số *đặc trưng* (feature) tiêu biểu.

2.3.2. Nhóm đặc trưng liên quan hàm nhạy cảm và dấu hiệu thực thi

Bên cạnh các đặc trưng thống kê, một nhóm đặc trưng quan trọng khác là các đặc trưng phản ánh dấu hiệu sử dụng những hàm nhạy cảm trong PHP. Trong bối cảnh mã độc và webshell, các hàm như `eval` (thực thi mã động), `system`/`exec` (thực thi lệnh hệ điều hành) hoặc các thao tác *mã hóa/giải mã* (encoding/decoding) như `base64_decode` thường được xem là chỉ báo mạnh cho hành vi nguy hiểm.

Trong tập dữ liệu, các đặc trưng thuộc nhóm này có thể xuất hiện dưới dạng biến đếm (số lần xuất hiện) hoặc biến nhị phân (có/không). Khi kết hợp với nhóm đặc trưng thống kê, mô hình học máy có thể học được cả hai khía cạnh: dấu hiệu hành vi "rõ" (ví dụ gọi hàm nhạy cảm) và dấu hiệu "gián tiếp" (ví dụ chuỗi dài, entropy cao).

2.3.3. Nhóm đặc trưng dạng chữ ký và lưu ý về độ rò rỉ nhãn

Một số đặc trưng trong tập dữ liệu có thể mang tính *chữ ký* (signature) hoặc *heuristic* (heuristic), tức là phản ánh mức độ "nghi ngờ" dựa trên các quy tắc nhận dạng có sẵn. Các đặc trưng dạng chữ ký có ưu điểm là giúp mô hình đạt hiệu năng cao trong nhiều trường hợp; tuy nhiên chúng cũng có thể gây ra nguy cơ *độ rò rỉ nhãn* (label leakage), khi mô hình dựa quá nhiều vào một dấu hiệu gần tương đương với nhãn thay vì học bản chất phân biệt giữa hai lớp.

Để đảm bảo tính khách quan mà vẫn đảm bảo cơ sở lý thuyết học máy cơ bản, luận văn thiết kế một thí nghiệm đối chứng ở các Chương 3, 4: huấn luyện mô hình trong hai kịch bản (có và không có nhóm đặc trưng dạng chữ ký), từ đó đánh giá mức độ phụ thuộc của mô hình vào các đặc trưng này. Thiết kế đối chứng giúp tăng độ tin cậy khoa học của kết quả mà không yêu cầu phương pháp học máy nâng cao.

2.4. Kiểm định chất lượng dữ liệu

Trước khi tiến hành huấn luyện mô hình học máy, việc kiểm định *chất lượng dữ liệu* (data quality) là cần thiết nhằm đảm bảo dữ liệu đầu vào có độ tin cậy và hạn chế các sai lệch do lỗi kỹ thuật. Trong phạm vi luận văn, kiểm định chất lượng tập trung vào ba nhóm tiêu chí cơ bản nhưng quan trọng: (i) kiểm tra *giá trị thiếu* (missing values), (ii) kiểm tra *trùng lặp* (duplicate samples), và (iii) sàng lọc các *đặc trưng* (features) không mang thông tin, điển hình là *cột hằng* (constant columns).

Các kiểm định trên được thực hiện tự động bằng ngôn ngữ Python để đảm bảo tính tái lập. Kết quả kiểm định sẽ được sử dụng trực tiếp trong quy trình tiền xử lý ở Mục 2.5 và góp phần đảm bảo thiết kế thí nghiệm ở Chương 3, Chương 4 diễn ra nhất

quán.

2.4.1. Kiểm tra giá trị thiếu và dữ liệu trùng

Trên tập dữ liệu CSV, luận văn kiểm tra tổng số ô dữ liệu bị thiếu nhằm đảm bảo các mô hình học máy không bị ảnh hưởng bởi các giá trị rỗng hoặc bị khuyết. Đồng thời, luận văn kiểm tra số lượng dòng trùng lặp để tránh trường hợp một mẫu xuất hiện nhiều lần, gây sai lệch trong quá trình huấn luyện và đánh giá.

Kết quả kiểm định cho thấy tập dữ liệu không có giá trị thiếu và không có dòng trùng lặp. Điều này cho phép thực hiện các bước tiền xử lý tiếp theo mà không cần áp dụng các kỹ thuật điền khuyết (imputation) hoặc loại bỏ mẫu trùng, từ đó giảm nguy cơ phát sinh sai lệch do xử lý bổ sung.

2.4.2. Phát hiện và loại bỏ cột hằng

Một số *đặc trưng* (features) trong tập dữ liệu có thể nhận cùng một giá trị trên toàn bộ mẫu (thường là 0). Các *cột hằng* (constant columns) này không đóng góp thông tin cho bài toán phân loại do *phương sai* (variance) bằng 0, đồng thời có thể gây nhiễu hoặc làm tăng chi phí tính toán không cần thiết trong quá trình huấn luyện.

Do đó, luận văn thực hiện bước sàng lọc đặc trưng cơ bản bằng cách loại bỏ các cột có số lượng giá trị khác nhau bằng 1. Sau bước sàng lọc này, số lượng đặc trưng hữu ích được giữ lại để đưa vào pipeline tiền xử lý ở Mục 2.5. Cách tiếp cận này phù hợp với mục tiêu *ứng dụng học máy* (applied machine learning) ở mức cơ bản và giúp mô hình ổn định hơn khi huấn luyện.

Danh sách đầy đủ các *cột hằng* (constant columns) bị loại bỏ được cung cấp trong Phụ lục.

2.4.3. Tổng hợp kết quả kiểm định chất lượng

Bảng 2.4.1 tổng hợp kết quả kiểm tra giá trị thiếu, dữ liệu trùng và số lượng *cột hằng* (constant columns) được phát hiện trong tập dữ liệu. Trên cơ sở đó, luận văn giữ lại các đặc trưng có biến thiên để phục vụ huấn luyện mô hình ở Chương 3.

Bảng 2.4.1. Kết quả kiểm định chất lượng và sàng lọc đặc trưng ở mức cơ bản.

Chỉ số kiểm định (Quality check)	Kết quả (Result)
Tổng số ô thiếu (Missing cells)	0
Số dòng trùng (Duplicate rows)	0
Số cột hằng (Constant columns)	17
Số đặc trưng sau loại cột hằng (Remaining features)	83

Bảng 2.5.1. Thiết kế chia tập dữ liệu.

Tập con (Subset)	Tỷ lệ (Ratio)	Số mẫu (Count)	Phân bố nhãn (Class distribution)
Train (huấn luyện) (training)	70%	1388	normal: 694, webshell: 694
Validation (xác thực) (validation)	15%	298	webshell: 149, normal: 149
Test (kiểm thử) (test)	15%	298	webshell: 149, normal: 149

2.5. Quy trình tiền xử lý dữ liệu cho học máy

2.5.1. Tách biến đầu vào và nhãn

Từ file CSV ban đầu, dữ liệu được tách thành hai thành phần: (i) ma trận đặc trưng X gồm tất cả các cột mô tả thuộc tính mã nguồn, và (ii) vector nhãn y lấy từ cột `label`. Việc tách riêng X và y nhằm đảm bảo tính rõ ràng giữa dữ liệu đầu vào và mục tiêu phân lớp, đồng thời là thao tác chuẩn trong quy trình huấn luyện mô hình học máy.

Trong luận văn, các đặc trưng trong X đều là biến số (int/float), phù hợp để áp dụng trực tiếp các mô hình học máy cơ bản như *hồi quy logistic* (logistic regression) và *rừng ngẫu nhiên* (random forest) ở Chương 3.

2.5.2. Chia tập train/validation/test theo lấy mẫu phân tầng

Để đánh giá mô hình một cách khách quan và hạn chế *quá khớp* (overfitting), tập dữ liệu được chia thành ba tập con: *huấn luyện* (training), *xác thực* (validation) và *kiểm thử* (test). Luận văn lựa chọn tỷ lệ chia 70/15/15, trong đó tập train dùng để huấn luyện tham số mô hình, tập validation dùng để lựa chọn mô hình/tham số ở Chương 3, và tập test chỉ sử dụng ở Chương 4 để báo cáo kết quả cuối.

Quá trình chia tập sử dụng *lấy mẫu phân tầng* (stratified sampling) nhằm giữ nguyên phân bố nhãn giữa các tập con. Cách thiết kế này giúp đảm bảo tập validation và test phản ánh đúng đặc tính dữ liệu ban đầu và tránh sai lệch khi so sánh mô hình.

2.5.3. Chuẩn hóa thang đo đặc trưng và xây dựng pipeline tiền xử lý

Do các *đặc trưng* (features) có thang đo khác nhau (ví dụ: đặc trưng độ dài có thể đạt giá trị rất lớn, trong khi một số đặc trưng khác chỉ nhận giá trị 0/1), luận văn áp dụng bước *chuẩn hóa* (standardization) để đưa các đặc trưng về cùng miền giá trị tương đối. Việc chuẩn hóa giúp các mô hình tuyến tính như *hồi quy logistic* (logistic regression) hoạt động ổn định hơn và giảm ảnh hưởng của sự chênh lệch thang đo.

Quy trình tiền xử lý được đóng gói thành một *pipeline* (pipeline) gồm hai bước chính: (i) loại bỏ các đặc trưng có *phương sai* (variance) bằng 0 (tương ứng với *cột hằng* (constant columns)), (ii) chuẩn hóa đặc trưng bằng StandardScaler (chuẩn hóa

theo trung bình và độ lệch chuẩn). Quan trọng hơn, pipeline chỉ được *fit* (fit) trên tập train và sau đó áp dụng nhất quán cho validation/test để tránh *rò rỉ dữ liệu* (data leakage) trong đánh giá.

2.5.4. Lưu trữ tạo phẩm dữ liệu để tái sử dụng

Để đảm bảo tính tái lập và duy trì sự nhất quán giữa các chương, luận văn lưu lại các *tạo phẩm dữ liệu* (data artifacts) sau khi tiền xử lý. Các tạo phẩm này được sinh tự động bằng Python và được sử dụng trực tiếp trong toàn bộ quy trình huấn luyện và đánh giá:

(i) `dataset_splits.npz`: chứa X_{train}, X_{val}, X_{test} sau tiền xử lý và các vector nhãn tương ứng, giúp Chương 3 và Chương 4 sử dụng đúng một thiết kế chia tập cố định.

(ii) `preprocess.joblib`: lưu *pipeline* (pipeline) tiền xử lý (loại cột hằng và chuẩn hóa), đảm bảo khi áp dụng cho dữ liệu mới hoặc khi tái chạy thí nghiệm sẽ thu được kết quả nhất quán.

(iii) `selected_features.csv`: danh sách đặc trưng được giữ lại sau bước sàng lọc, phục vụ việc truy vết và diễn giải đặc trưng khi cần thiết.

Việc lưu các tạo phẩm dữ liệu nêu trên giúp Chương 3 tập trung vào xây dựng mô hình và lựa chọn tham số trên tập train/validation, đồng thời giúp Chương 4 đánh giá cuối trên tập test mà không thay đổi bất kỳ bước tiền xử lý nào.

2.6. Kết luận

Chương 2 đã trình bày quy trình xây dựng tập dữ liệu dạng *đặc trưng* (feature-level dataset) phục vụ bài toán phát hiện webshell trong mã nguồn PHP. Trước hết, tập dữ liệu được mô tả rõ ràng về cấu trúc file CSV và phân bố nhãn (Mục 2.2), qua đó cung cấp bức tranh tổng quan về quy mô dữ liệu và tính cân bằng giữa hai lớp.

Tiếp theo, luận văn thực hiện kiểm định *chất lượng dữ liệu* (data quality) ở mức cơ bản nhưng cần thiết, bao gồm kiểm tra *giá trị thiếu* (missing values), *trùng lặp* (duplicates) và sàng lọc *cột hằng* (constant columns) (Mục 2.4). Các bước này nhằm đảm bảo dữ liệu đầu vào có độ tin cậy, giảm nhiễu và hạn chế sai lệch trong quá trình huấn luyện.

Trên cơ sở dữ liệu đã kiểm định, Chương 2 xây dựng quy trình *tiền xử lý* (preprocessing) cho học máy (Mục 2.5), bao gồm: chia tập train/validation/test theo *lấy mẫu phân tầng* (stratified sampling) với tỷ lệ 70/15/15, chuẩn hóa thang đo đặc

trưng và đóng gói toàn bộ quy trình dưới dạng *pipeline* (pipeline). Đặc biệt, luận văn lưu lại các *tạo phẩm dữ liệu* (data artifacts) gồm (i) tập dữ liệu đã chia và tiền xử lý, và (ii) pipeline tiền xử lý, nhằm đảm bảo khả năng tái lập và duy trì sự nhất quán giữa các chương.

Từ các kết quả của Chương 2, Chương 3 sẽ sử dụng tập train và validation (được tạo ra ở Mục 2.5.2) để xây dựng các mô hình học máy cơ bản, lựa chọn tham số phù hợp và đề xuất cải tiến ở mức quy trình/đặc trưng trên cùng một thiết kế tiền xử lý cố định. Sau đó, Chương 4 sẽ sử dụng tập test (được tách riêng từ Chương 2) để thực hiện mô phỏng, đánh giá cuối và so sánh mô hình bằng các chỉ số như *độ chính xác* (accuracy), *độ đúng* (precision), *độ bao phủ* (recall) và *F1-score* (F1-score), kèm theo các hình minh họa như ma trận nhầm lẫn và đường cong ROC/PR.

CHƯƠNG 3. NGHIÊN CỨU MÔ HÌNH PHÁT HIỆN WEBSHELL

3.1. Giới thiệu và mục tiêu mô hình phát hiện

Trên cơ sở tập dữ liệu đã được xây dựng và tiền xử lý ở Chương 2, chương này tập trung vào việc xây dựng mô hình học máy để phát hiện webshell trong mã nguồn PHP. Do mục tiêu của luận văn là theo hướng ứng dụng học máy ở mức cơ bản, luận văn lựa chọn các mô hình phổ biến, dễ diễn giải và phù hợp với dữ liệu dạng vector đặc trưng (feature vector), bao gồm: *hồi quy logistic* (logistic regression) và *rừng ngẫu nhiên* (random forest).

Mục tiêu của Chương 3 gồm: (i) xây dựng các mô hình cơ bản trên tập train, (ii) lựa chọn tham số đơn giản dựa trên tập validation để hạn chế *quá khớp* (overfitting), và (iii) lựa chọn mô hình/thiết lập tốt nhất làm cơ sở cho đánh giá cuối trong Chương 4. Toàn bộ quy trình huấn luyện và lựa chọn mô hình được thực hiện trên cùng một *pipeline* (pipeline) tiền xử lý và cùng thiết kế chia tập dữ liệu (70/15/15), nhằm đảm bảo các mô hình được so sánh một cách nhất quán.

Dữ liệu đầu vào cho huấn luyện và đánh giá trong chương này được lấy từ các *tạo phẩm dữ liệu* (data artifacts) đã được đề cập trong Chương 2, bao gồm `dataset_splits.npz` (các tập train/validation/test sau tiền xử lý) và `preprocess.joblib` (pipeline tiền xử lý).

3.2. Thiết kế thí nghiệm và cấu hình huấn luyện

3.2.1. Thiết kế dữ liệu huấn luyện và xác thực

Theo thiết kế ở Chương 2, tập dữ liệu được chia thành ba phần: train/validation/test theo tỷ lệ 70/15/15 và sử dụng *lấy mẫu phân tầng* (stratified sampling) để giữ ổn định phân bố nhãn. Trong Chương 3, luận văn chỉ sử dụng hai tập train và validation cho quá trình huấn luyện và lựa chọn mô hình. Tập test được giữ độc lập và chỉ sử dụng cho đánh giá cuối ở Chương 4, nhằm tránh *rò rỉ dữ liệu* (data leakage) và đảm bảo tính khách quan của kết quả.

Để đảm bảo tính nhất quán giữa các mô hình, toàn bộ dữ liệu đầu vào đã được tiền xử lý theo cùng một *pipeline* (pipeline) được mô tả từ Chương 2. Nhờ đó, mỗi mô hình được huấn luyện và đánh giá trên cùng không gian đặc trưng và cùng thang đo.

3.2.2. Tiêu chí lựa chọn mô hình và lý do lựa chọn

Trong bài toán phát hiện webshell, việc tối ưu hóa *độ chính xác* (accuracy) đơn thuần có thể không phản ánh đầy đủ chất lượng phát hiện, đặc biệt khi cân nhắc yêu cầu giảm báo động giả và giảm bỏ sót. Vì vậy, luận văn sử dụng *F1-score* (F1-score) trên tập validation làm tiêu chí chính để lựa chọn mô hình và tham số. Chỉ số F1-score kết hợp đồng thời *độ đúng* (precision) và *độ bao phủ* (recall), phù hợp để đánh giá năng lực phát hiện ở mức cảnh báo.

Việc lựa chọn hai mô hình Logistic Regression và Random Forest trong luận văn không nhằm chứng minh đây là các thuật toán có hiệu năng cao nhất đối với bài toán phát hiện webshell, mà nhằm đại diện cho hai hướng tiếp cận phổ biến trong học máy là mô hình tuyến tính và mô hình phi tuyến. Hai thuật toán đều đã được sử dụng rộng rãi trong các bài toán phân loại nhị phân, có cấu trúc tương đối đơn giản, dễ triển khai, dễ diễn giải kết quả và phù hợp với quy mô của bộ dữ liệu sử dụng trong nghiên cứu. Logistic Regression được lựa chọn làm mô hình cơ sở (baseline) do có tốc độ huấn luyện nhanh, khả năng diễn giải tốt và thường được sử dụng như một chuẩn so sánh trong các nghiên cứu phân loại. Trong khi đó, Random Forest có khả năng học được các quan hệ phi tuyến và sự tương tác giữa nhiều đặc trưng, phù hợp với bài toán phát hiện webshell khi các dấu hiệu độc hại thường không xuất hiện riêng lẻ mà là sự kết hợp của nhiều đặc trưng thống kê, đặc trưng hành vi và đặc trưng dạng chữ ký. Mặc dù các phương pháp học sâu hoặc các mô hình biểu diễn mã nguồn hiện đại (Transformer, CodeBERT, Graph Neural Network...) đã cho thấy nhiều kết quả tích cực trong các nghiên cứu gần đây, các phương pháp này thường yêu cầu bộ dữ liệu có quy mô lớn, chi phí huấn luyện cao và khó diễn giải hơn. Vì vậy, trong phạm vi luận văn, hai mô hình Logistic Regression và Random Forest được lựa chọn nhằm đảm bảo tính đơn giản, khả năng tái lập và thuận lợi trong quá trình phân tích kết quả.

3.2.3. Cấu hình huấn luyện mô hình

Luận văn lựa chọn hai mô hình học máy cơ bản như sau:

(i) *Hồi quy logistic* (logistic regression): mô hình tuyến tính, dễ triển khai và dễ diễn giải. Luận văn khảo sát tham số điều chuẩn C ở một tập giá trị nhỏ nhằm kiểm soát mức độ *điều chuẩn* (regularization).

(ii) *Rừng ngẫu nhiên* (random forest): mô hình dựa trên tập hợp cây quyết định, có khả năng mô hình hóa quan hệ phi tuyến tính giữa các đặc trưng. Luận văn khảo sát một số thiết lập cơ bản như số lượng cây và độ sâu tối đa nhằm cân bằng giữa hiệu

năng và độ phức tạp.

Việc giới hạn số lượng tham số khảo sát ở mức nhỏ giúp đảm bảo quy trình phù hợp với định hướng ứng dụng học máy của luận văn, đồng thời vẫn thể hiện được bước lựa chọn mô hình dựa trên dữ liệu validation thay vì lựa chọn cảm tính.

3.2.4. Mô hình hóa bài toán phân loại nhị phân

Với mỗi mẫu mã nguồn PHP, sau bước trích xuất và tiền xử lý ở Chương 2, dữ liệu được biểu diễn bởi một vector đặc trưng $x \in \mathbb{R}^d$ và nhãn tương ứng $y \in \{0, 1\}$, trong đó $y = 1$ đại diện cho lớp `webshell` và $y = 0$ đại diện cho lớp `normal`. Bài toán đặt ra là học một hàm phân loại $f(\cdot)$ sao cho:

$$\hat{y} = f(x),$$

trong đó $\hat{y}$ là nhãn dự đoán. Trong phạm vi luận văn, các mô hình được lựa chọn đều thuộc nhóm *học có giám sát* (supervised learning), nghĩa là mô hình học tham số từ dữ liệu đã gán nhãn.

Ngoài dự đoán nhãn, một số mô hình (ví dụ *hồi quy logistic* (logistic regression)) có thể sinh ra xác suất dự đoán $p(y = 1 \mid x)$. Giá trị xác suất này có thể hỗ trợ điều chỉnh ngưỡng cảnh báo trong triển khai thực tế, tuy nhiên trong luận văn ngưỡng phân loại mặc định được sử dụng theo thiết lập của thư viện (thường là 0.5) để đảm bảo tính đơn giản và nhất quán trong so sánh mô hình.

3.3. Huấn luyện và lựa chọn mô hình trên tập validation

3.3.1. Quy trình huấn luyện và so sánh mô hình

Trên cơ sở dữ liệu đã được tiền xử lý và chia tập ở Chương 2, luận văn tiến hành huấn luyện các mô hình học máy cơ bản trên tập train và đánh giá tạm thời trên tập validation. Quy trình thực nghiệm được thực hiện theo nguyên tắc sau: (i) mô hình được huấn luyện (fit) trên tập train, (ii) dự đoán trên tập validation để tính chỉ số *F1-score* (F1-score), và (iii) lựa chọn thiết lập tham số có F1-score tốt nhất. Tập test không tham gia vào quá trình lựa chọn nhằm đảm bảo đánh giá cuối ở Chương 4 mang tính khách quan.

Đối với *hồi quy logistic* (logistic regression), luận văn khảo sát tham số điều chuẩn C với một số giá trị tiêu biểu nhằm quan sát ảnh hưởng của mức độ điều chuẩn đến khả năng phân loại. Đối với *rừng ngẫu nhiên* (random forest), luận văn khảo sát một số thiết lập cơ bản gồm số lượng cây và độ sâu tối đa, là các tham số có ảnh hưởng

trực tiếp đến độ phức tạp và khả năng tổng quát hóa của mô hình.

3.3.2. Kết quả lựa chọn mô hình/tham số

Bảng 3.3.1 trình bày kết quả lựa chọn mô hình và tham số đơn giản dựa trên *F1-score* (F1-score) trên tập validation. Mỗi mô hình được báo cáo kèm thiết lập tham số tốt nhất và giá trị F1-score tương ứng. Thiết lập được lựa chọn ở mục này sẽ được giữ cố định để đánh giá cuối trên tập test trong Chương 4.

Bảng 3.3.1. Kết quả lựa chọn mô hình/tham số dựa trên tập validation.

Mô hình (Model)	Thiết lập (Setting)	F1 trên validation (Validation F1)
Logistic Regression	C=10.0	0.9324
Random Forest	n=100, depth=10	0.9695

3.3.3. Kết luận lựa chọn mô hình

Từ kết quả trên tập validation, luận văn lựa chọn hai mô hình với thiết lập tham số tương ứng để đưa vào đánh giá cuối. Trong Chương 4, các mô hình sẽ được đánh giá trên tập test bằng các chỉ số *độ chính xác* (accuracy), *độ đúng* (precision), *độ bao phủ* (recall), *F1-score* (F1-score), đồng thời minh họa bằng ma trận nhầm lẫn và các đường cong ROC/PR nhằm quan sát hành vi phân loại ở các ngưỡng khác nhau.

3.4. Phân tích mô hình hồi quy logistic

3.4.1. Nguyên lý và biểu diễn xác suất

Hồi quy logistic (logistic regression) là một mô hình tuyến tính phổ biến cho bài toán *phân loại nhị phân* (binary classification). Với một mẫu đầu vào được biểu diễn bởi vector đặc trưng $x \in \mathbb{R}^d$, mô hình ước lượng xác suất mẫu thuộc lớp dương (`webshell`) thông qua hàm sigmoid:

$$p(y = 1 \mid x) = \sigma(z) = \frac{1}{1 + e^{-z}}, \quad z = w^\top x + b,$$

trong đó $w \in \mathbb{R}^d$ là vector trọng số và b là hệ số chệch (bias). Dự đoán nhãn được suy ra dựa trên ngưỡng τ:

$$\hat{y} = \begin{cases} 1, & p(y = 1 \mid x) \geq \tau, \\ 0, & p(y = 1 \mid x) < \tau. \end{cases}$$

Trong nghiên cứu này, ngưỡng phân loại được sử dụng là $\tau = 0.5$, tương ứng với thiết lập mặc định của thư viện Scikit-learn. Khi xác suất dự đoán $p(y = 1|x) \geq 0.5$, mẫu được phân loại là webshell; ngược lại sẽ được phân loại là mẫu bình thường. Giá trị

ngưỡng này phù hợp với bộ dữ liệu cân bằng giữa hai lớp được sử dụng trong luận văn và giúp thuận lợi khi so sánh với các nghiên cứu liên quan.

Đối với bài toán phát hiện webshell, lớp `webshell` được xem là lớp dương (positive class). Do đó, các chỉ số *độ đúng* (precision), *độ bao phủ* (recall) và *F1-score* (F1-score) trong các chương sau đều được tính theo lớp dương này.

3.4.2. Hàm mất mát và vai trò của điều chuẩn

Mục tiêu huấn luyện của hồi quy logistic là tìm bộ tham số (w, b) sao cho xác suất dự đoán phù hợp nhất với nhãn thật trên tập huấn luyện. Trong thực tế, mô hình thường kết hợp *điều chuẩn* (regularization) để hạn chế *quá khớp* (overfitting), đặc biệt khi dữ liệu có nhiều đặc trưng hoặc các đặc trưng có tương quan. Về mặt trực giác, điều chuẩn giúp tránh trường hợp mô hình "phóng đại" trọng số của một số đặc trưng để khớp chặt dữ liệu huấn luyện nhưng lại kém tổng quát trên dữ liệu mới.

Trong thư viện scikit-learn, tham số C điều khiển mức độ điều chuẩn theo dạng nghịch đảo: C càng lớn thì điều chuẩn càng yếu, C càng nhỏ thì điều chuẩn càng mạnh. Trong luận văn, C được khảo sát trên một tập giá trị nhỏ và lựa chọn dựa trên hiệu năng validation (Mục 3.3), nhằm đảm bảo quy trình lựa chọn tham số đơn giản, có cơ sở thực nghiệm và dễ giải trình.

3.4.3. Tính phù hợp của hồi quy logistic với dữ liệu đặc trưng

Với dữ liệu dạng vector đặc trưng, hồi quy logistic có một số ưu điểm phù hợp với mục tiêu luận văn: (i) đơn giản và hiệu quả, chi phí huấn luyện thấp; (ii) dễ triển khai trong thực tế do tốc độ suy luận nhanh; (iii) có khả năng diễn giải mức độ ảnh hưởng của đặc trưng thông qua trọng số w (mức độ đóng góp tương đối).

Mặc dù là mô hình tuyến tính, hồi quy logistic vẫn có thể đạt hiệu quả tốt nếu các đặc trưng đầu vào đã phản ánh được sự khác biệt giữa hai lớp. Trong bài toán phát hiện webshell, các đặc trưng thường kết hợp cả dấu hiệu trực tiếp (ví dụ sự xuất hiện của các hàm nhạy cảm) và dấu hiệu gián tiếp (ví dụ độ dài chuỗi, độ phức tạp thống kê). Sự kết hợp này làm tăng khả năng tách biệt tuyến tính giữa hai lớp trong không gian đặc trưng, qua đó giúp hồi quy logistic trở thành một mô hình nền tảng (baseline) phù hợp.

3.4.4. Giới hạn của hồi quy logistic trong bối cảnh phát hiện webshell

Do bản chất tuyến tính, hồi quy logistic có thể gặp hạn chế khi ranh giới phân tách giữa hai lớp có dạng phi tuyến mạnh hoặc khi hành vi webshell bị che giấu theo

nhiều biến thể phức tạp. Ngoài ra, mô hình có thể nhạy với các đặc trưng có phân phối lệch và ngoại lệ nếu không chuẩn hóa thang đo. Vì vậy, luận văn kết hợp sử dụng pipeline chuẩn hóa đặc trưng (Mục 2.5) và đồng thời sử dụng một mô hình phi tuyến (*rừng ngẫu nhiên* (random forest)) để so sánh và bổ sung góc nhìn.

3.5. Phân tích mô hình rừng ngẫu nhiên

3.5.1. Nguyên lý mô hình và trực giác hoạt động

Rừng ngẫu nhiên (random forest) là một mô hình *tổ hợp* (ensemble) dựa trên nhiều *cây quyết định* (decision tree). Mỗi cây quyết định học một tập các quy tắc phân tách dữ liệu dựa trên các đặc trưng đầu vào để đi đến dự đoán nhãn. Khi dự đoán, rừng ngẫu nhiên kết hợp kết quả của nhiều cây (thường bằng bỏ phiếu đa số đối với phân loại) để đưa ra dự đoán cuối cùng.

Trực giác quan trọng của rừng ngẫu nhiên là giảm sai lệch do một cây đơn lẻ bằng cách tạo ra nhiều cây "đa dạng" và tổng hợp chúng. Độ đa dạng này đến từ hai cơ chế ngẫu nhiên: (i) mỗi cây được huấn luyện trên một mẫu dữ liệu được lấy lại có hoàn (bootstrap sampling), và (ii) tại mỗi nút phân tách, mô hình chỉ xét một tập con ngẫu nhiên của các đặc trưng. Nhờ đó, mô hình có xu hướng giảm *phương sai* (variance) so với cây quyết định đơn, và cải thiện khả năng tổng quát hóa.

Random Forest được xây dựng từ nhiều cây quyết định (decision tree) độc lập. Mỗi cây không được huấn luyện trên toàn bộ tập dữ liệu mà trên một tập dữ liệu con được tạo bằng kỹ thuật Bootstrap (lấy mẫu có hoàn lại). Với phương pháp này, một mẫu dữ liệu có thể được lựa chọn nhiều lần trong cùng một tập huấn luyện, trong khi một số mẫu khác có thể không được lựa chọn. Điều này tạo ra sự khác biệt giữa các cây quyết định, góp phần làm giảm hiện tượng quá khớp (overfitting). Bên cạnh việc lấy mẫu ngẫu nhiên dữ liệu, tại mỗi nút của cây quyết định, Random Forest chỉ lựa chọn ngẫu nhiên một tập con các đặc trưng để tìm phép chia tốt nhất thay vì sử dụng toàn bộ đặc trưng. Cơ chế này giúp giảm sự tương quan giữa các cây quyết định và nâng cao khả năng tổng quát hóa của mô hình.

3.5.2. Cơ chế biểu diễn quan hệ phi tuyến và tương tác đặc trưng

Khác với *hồi quy logistic* (logistic regression) vốn tạo ranh giới phân tách tuyến tính trong không gian đặc trưng, rừng ngẫu nhiên có thể mô hình hóa các quan hệ phi tuyến và các tương tác phức tạp giữa các đặc trưng. Điều này đặc biệt phù hợp với bài toán phát hiện webshell trong thực tế, khi hành vi webshell có thể được che giấu bằng nhiều kỹ thuật *làm rối* (obfuscation) và biến thể mã khác nhau.

Trong bối cảnh dữ liệu dạng vector đặc trưng, các quy tắc phân tách của cây quyết định có thể xem như các điều kiện dạng "nếu–thì" (if–then) trên các đặc trưng, ví dụ: nếu một số đặc trưng về độ dài chuỗi/entropy vượt ngưỡng và đồng thời xuất hiện một dấu hiệu thực thi nhạy cảm, mẫu có xu hướng thuộc lớp `webshell`. Cách biểu diễn theo quy tắc này giúp rừng ngẫu nhiên có khả năng xử lý các mẫu mà dấu hiệu không hoàn toàn tuyến tính.

3.5.3. Tham số cơ bản và ý nghĩa trong luận văn

Trong phạm vi luận văn, mô hình rừng ngẫu nhiên được khảo sát với một số tham số cơ bản nhằm đảm bảo tính đơn giản và dễ giải trình: (i) số lượng cây (*number of trees* (n_estimators)), và (ii) độ sâu tối đa của cây (*maximum depth* (max_depth)).

Số lượng cây lớn hơn thường giúp mô hình ổn định hơn do giảm dao động giữa các lần huấn luyện, tuy nhiên cũng làm tăng chi phí tính toán. Độ sâu tối đa của cây ảnh hưởng trực tiếp đến độ phức tạp của mô hình: cây càng sâu càng có khả năng khớp dữ liệu huấn luyện chi tiết hơn, nhưng nguy cơ *quá khớp* (overfitting) cũng tăng. Do đó, luận văn lựa chọn tham số dựa trên *F1-score* (F1-score) trên tập validation (Mục 3.3), đảm bảo việc điều chỉnh tham số có cơ sở thực nghiệm và không sử dụng tập test trong giai đoạn lựa chọn.

3.5.4. Tính phù hợp và giới hạn của rừng ngẫu nhiên

Rừng ngẫu nhiên có một số ưu điểm phù hợp với mục tiêu phát hiện webshell: (i) mô hình hóa quan hệ phi tuyến tốt, (ii) tương đối bền vững trước nhiễu và outlier do cơ chế tổng hợp nhiều cây, và (iii) không quá nhạy với việc chuẩn hóa thang đo như mô hình tuyến tính. Trong luận văn, mặc dù pipeline tiền xử lý vẫn chuẩn hóa thang đo để đảm bảo nhất quán với các mô hình khác, rừng ngẫu nhiên vẫn có thể tận dụng tốt các đặc trưng đa dạng để phân biệt giữa hai lớp.

Tuy nhiên, rừng ngẫu nhiên cũng có một số giới hạn: (i) chi phí suy luận thường cao hơn so với hồi quy logistic, đặc biệt khi số lượng cây lớn; (ii) mức độ diễn giải trực tiếp thường khó hơn mô hình tuyến tính, do dự đoán là kết quả tổng hợp nhiều cây; và (iii) nếu thiết lập tham số không hợp lý (ví dụ cây quá sâu), mô hình có thể tăng nguy cơ quá khớp. Vì vậy, luận văn sử dụng chiến lược lựa chọn tham số đơn giản dựa trên validation để cân bằng giữa hiệu năng và khả năng tổng quát hóa.

3.6. Thảo luận về tiêu chí lựa chọn và nguy cơ sai lệch

3.6.1. Lựa chọn chỉ số đánh giá: vì sao ưu tiên F1-score

Trong bài toán phát hiện webshell, hai dạng sai sót đều có chi phí: (i) *báo động giả* (false positive) làm tăng khối lượng công việc rà soát mã nguồn và có thể gây "nhiễu" cho người quản trị, và (ii) *bỏ sót* (false negative) có thể dẫn tới rủi ro an ninh nghiêm trọng do webshell không bị phát hiện. Do đó, nếu chỉ tối ưu *độ chính xác* (accuracy) thì có thể không phản ánh đầy đủ chất lượng mô hình trong bối cảnh an toàn thông tin.

Vì lý do trên, luận văn lựa chọn *F1-score* (F1-score) làm tiêu chí chính để so sánh và lựa chọn mô hình trên tập validation. F1-score là trung bình điều hòa của *độ đúng* (precision) và *độ bao phủ* (recall), phản ánh mức cân bằng giữa giảm báo động giả và giảm bỏ sót. Trong luận văn, lớp `webshell` được xem là lớp dương (positive class) khi tính các chỉ số precision/recall/F1.

3.6.2. Nguy cơ quá khớp và các dấu hiệu cần lưu ý

Quá khớp (overfitting) xảy ra khi mô hình học quá sát các mẫu trong tập huấn luyện, dẫn đến hiệu năng cao trên train nhưng giảm đáng kể trên dữ liệu mới. Nguy cơ này thường tăng khi mô hình có độ phức tạp cao hoặc khi điều chỉnh tham số quá nhiều lần dựa trên một tập validation cố định.

Trong luận văn, một số biện pháp cơ bản được sử dụng để hạn chế quá khớp: (i) thiết kế chia dữ liệu train/validation/test tách biệt ngay từ Chương 2, (ii) giới hạn không gian tham số khảo sát ở mức nhỏ để phù hợp định hướng ứng dụng cơ bản, và (iii) lựa chọn mô hình dựa trên tập validation thay vì dựa trên tập train. Đối với mô hình *rừng ngẫu nhiên* (random forest), việc giới hạn độ sâu tối đa cũng góp phần giảm nguy cơ quá khớp do cây quyết định quá phức tạp.

Một dấu hiệu thực nghiệm thường gặp của quá khớp là sự chênh lệch lớn giữa F1-score trên train và F1-score trên validation. Trong luận văn, việc sử dụng validation cho lựa chọn mô hình (Bảng 3.3.1) giúp cung cấp một điểm kiểm tra độc lập để quan sát xu hướng này.

3.6.3. Nguy cơ rò rỉ dữ liệu và cách kiểm soát

Rò rỉ dữ liệu (data leakage) là một trong những nguyên nhân phổ biến dẫn đến kết quả "đẹp giả", khi thông tin từ tập đánh giá vô tình được sử dụng trong quá trình huấn luyện hoặc lựa chọn mô hình. Một số dạng rò rỉ thường gặp gồm: (i) chuẩn hóa

đặc trưng trên toàn bộ dữ liệu trước khi chia tập, (ii) lựa chọn đặc trưng dựa trên toàn bộ dữ liệu, hoặc (iii) sử dụng tập test để điều chỉnh tham số.

Trong luận văn, nguy cơ rò rỉ dữ liệu được kiểm soát theo nguyên tắc: mọi bước có thể học tham số từ dữ liệu, ví dụ *chuẩn hóa* (standardization) hoặc sàng lọc đặc trưng theo *phương sai* (variance) chỉ được fit trên tập train. Sau đó, các phép biến đổi được áp dụng nhất quán cho validation/test bằng cùng một *pipeline* (pipeline). Ngoài ra, tập test được giữ độc lập và chỉ dùng cho báo cáo kết quả cuối ở Chương 4, tránh việc "tối ưu" mô hình theo test.

3.6.4. Thiên lệch dữ liệu và giới hạn của bộ dữ liệu

Mặc dù tập dữ liệu được cân bằng theo hai lớp, vẫn có thể tồn tại *thiên lệch* (bias) do nguồn dữ liệu và cách dữ liệu được thu thập/tiền xử lý. Ví dụ, nếu các mẫu webshell trong tập dữ liệu chủ yếu thuộc một số họ webshell phổ biến hoặc sử dụng một số kỹ thuật làm rối điển hình, mô hình có thể học tốt trên các kiểu đó nhưng kém tổng quát với các biến thể mới.

Bên cạnh đó, dữ liệu cân bằng giúp thuận lợi cho nghiên cứu và so sánh mô hình, nhưng trong triển khai thực tế, tỷ lệ webshell trong mã nguồn thường thấp hơn nhiều so với mã bình thường. Do đó, khi áp dụng thực tế, có thể cần điều chỉnh ngưỡng cảnh báo hoặc ưu tiên giảm bỏ sót tùy bối cảnh. Trong phạm vi luận văn, các thí nghiệm được thiết kế nhằm so sánh mô hình một cách nhất quán trên cùng tập dữ liệu chuẩn, và phần đánh giá ở Chương 4 sẽ cung cấp thêm góc nhìn thông qua ROC-AUC/PR-AUC và ma trận nhầm lẫn.

3.6.5. Đặc trưng mang tính chữ ký và nguy cơ đẹp giả

Một lưu ý quan trọng trong phát hiện webshell là một số đặc trưng có thể mang tính *chữ ký* (signature) hoặc *heuristic* (heuristic). Các đặc trưng này có thể giúp mô hình đạt hiệu năng cao nhanh chóng, nhưng cũng tiềm ẩn nguy cơ mô hình phụ thuộc quá nhiều vào dấu hiệu gần tương đương nhãn. Trong luận văn, để đảm bảo kết quả có ý nghĩa, phần đánh giá ở Chương 4 sẽ được trình bày theo nhiều chỉ số (precision/recall/F1 và ROC/PR), đồng thời luận văn thảo luận rõ giới hạn của dữ liệu và mô hình trong bối cảnh phát hiện dựa trên đặc trưng tĩnh.

3.7. Quy trình áp dụng mô hình cho dữ liệu mới

3.7.1. Yêu cầu nhất quán về không gian đặc trưng

Trong triển khai thực tế, mục tiêu của hệ thống là đánh giá một đoạn mã PHP mới và đưa ra dự đoán `webshell` hoặc `normal`. Tuy nhiên, do mô hình trong luận văn được huấn luyện trên dữ liệu dạng *vector đặc trưng* (feature vector), dữ liệu đầu vào của giai đoạn suy luận (inference) phải được biểu diễn đúng theo cùng không gian đặc trưng như dữ liệu huấn luyện.

Nói cách khác, nếu $x \in \mathbb{R}^d$ là vector đặc trưng sau tiền xử lý, thì một mẫu mã nguồn mới phải được trích xuất đặc trưng theo cùng bộ tiêu chí (cùng tên đặc trưng, cùng định nghĩa) và sau đó đi qua đúng quy trình tiền xử lý đã sử dụng khi huấn luyện. Điều này giúp bảo đảm tính nhất quán và tránh sai lệch do khác biệt thang đo hoặc thiếu đặc trưng.

3.7.2. Pipeline suy luận tổng quát

Quy trình suy luận tổng quát được mô tả theo các bước sau:

(i) *Thu thập dữ liệu đầu vào* (input acquisition): lấy mã nguồn PHP cần kiểm tra (ví dụ file hoặc đoạn mã).

(ii) *Trích xuất đặc trưng* (feature extraction): chuyển mã nguồn PHP sang dạng bảng đặc trưng theo đúng định nghĩa dataset. Trong phạm vi luận văn, bước này được giả định đã thực hiện để tạo ra dữ liệu dạng CSV/row tương ứng với một mẫu.

(iii) *Tiền xử lý đặc trưng* (preprocessing): áp dụng pipeline tiền xử lý đã lưu để (a) loại bỏ các đặc trưng không phù hợp (ví dụ *cột hằng* (constant features) đã bị loại trong huấn luyện), và (b) chuẩn hóa thang đo đặc trưng bằng StandardScaler. Bước này bảo đảm dữ liệu mới có cùng thang đo và cùng cách biểu diễn với dữ liệu huấn luyện.

(iv) *Dự đoán* (prediction): đưa vector đặc trưng sau tiền xử lý vào mô hình đã huấn luyện để thu được nhãn dự đoán $\hat{y} \in \{\texttt{webshell}, \texttt{normal}\}$. Với các mô hình có khả năng sinh xác suất, hệ thống có thể thu được thêm giá trị $p(\texttt{webshell} \mid x)$ để hỗ trợ diễn giải hoặc điều chỉnh ngưỡng cảnh báo.

(v) *Xuất kết quả* (output): trả về nhãn dự đoán và (nếu có) xác suất/tín hiệu hỗ trợ để phục vụ ra quyết định.

3.7.3. Sử dụng các tạo phẩm dữ liệu đã lưu để tái lập suy luận

Để đảm bảo quy trình suy luận có thể tái lập và nhất quán với quá trình huấn luyện, luận văn lưu và sử dụng các tệp *tạo phẩm dữ liệu* (data artifacts) sau đây được sinh ra sau quá trình xử lý bằng ngôn ngữ Python:

(i) `preprocess.joblib`: lưu pipeline tiền xử lý gồm sàng lọc đặc trưng theo phương sai và chuẩn hóa thang đo. Khi áp dụng cho dữ liệu mới, pipeline này giúp đảm bảo dữ liệu được biến đổi giống hệt như trong huấn luyện.

(ii) `best_lr.joblib` và `best_rf.joblib`: lưu mô hình hồi quy logistic và rừng ngẫu nhiên tương ứng với thiết lập tham số tốt nhất trên tập validation (Mục 3.3). Trong triển khai, có thể sử dụng một mô hình hoặc kết hợp hai mô hình tùy mục tiêu ưu tiên (ví dụ ưu tiên tốc độ hoặc ưu tiên khả năng mô hình hóa phi tuyến).

(iii) `selected_features.csv`: lưu danh sách đặc trưng được giữ lại sau bước sàng lọc. Tệp này giúp kiểm tra tính nhất quán của dữ liệu đầu vào (ví dụ đảm bảo dữ liệu mới có đủ các đặc trưng cần thiết) và hỗ trợ truy vết khi phân tích lỗi dự đoán.

Nhờ cơ chế lưu các tạo phẩm dữ liệu, toàn bộ quy trình từ tiền xử lý đến suy luận có thể được tái chạy nhất quán giữa các lần thực nghiệm và phù hợp với yêu cầu tái lập trong nghiên cứu khoa học.

3.7.4. Liên kết sang giai đoạn đánh giá cuối

Trong Chương 4, các mô hình đã lựa chọn sẽ được đánh giá trên tập test độc lập bằng các chỉ số tổng hợp và hình minh họa (ma trận nhầm lẫn, ROC/PR). Các kết quả đánh giá này phản ánh trực tiếp chất lượng của quy trình suy luận nêu trên khi áp dụng cho dữ liệu chưa từng xuất hiện trong huấn luyện và lựa chọn mô hình.

3.8. Kết luận

Chương 3 đã trình bày đầy đủ quy trình xây dựng mô hình phát hiện webshell theo hướng *ứng dụng học máy* (applied machine learning) ở mức cơ bản, bám sát dữ liệu dạng *vector đặc trưng* (feature vector) được tạo ra ở Chương 2. Trọng tâm của chương là thiết kế một quy trình huấn luyện và lựa chọn mô hình nhất quán, có khả năng tái lập, đồng thời tránh các sai lệch thường gặp trong thực nghiệm học máy.

Trước hết, chương đã xác lập rõ nguyên tắc tách biệt train/validation/test và sử dụng tập validation để lựa chọn mô hình/tham số, nhằm hạn chế *quá khớp* (overfitting)

và tránh *rò rỉ dữ liệu* (data leakage). Quy trình này đảm bảo rằng tập test được giữ độc lập và chỉ dùng cho đánh giá cuối, qua đó giúp các kết luận ở Chương 4 có giá trị khách quan. Bên cạnh đó, luận văn sử dụng *F1-score* (F1-score) làm tiêu chí lựa chọn mô hình, phản ánh cân bằng giữa *độ đúng* (precision) và *độ bao phủ* (recall) trong bối cảnh phát hiện mã độc, trong đó lớp `webshell` được xem là lớp dương (positive class).

Tiếp theo, chương đã triển khai hai mô hình học máy cơ bản nhưng có ý nghĩa so sánh rõ ràng: *hồi quy logistic* (logistic regression) và *rừng ngẫu nhiên* (random forest). Hồi quy logistic đóng vai trò mô hình tuyến tính nền tảng, có ưu điểm về tính đơn giản, tốc độ suy luận và khả năng diễn giải tương đối. Rừng ngẫu nhiên bổ sung góc nhìn phi tuyến, có khả năng mô hình hóa tương tác đặc trưng và xử lý các trường hợp mà ranh giới phân tách không tuyến tính. Việc đặt hai mô hình này cạnh nhau giúp luận văn vừa đáp ứng yêu cầu cơ bản về lý thuyết học máy, vừa làm cơ sở để thảo luận về ưu/nhược điểm của các hướng tiếp cận.

Kết quả lựa chọn mô hình và tham số đơn giản dựa trên tập validation đã được tổng hợp trong Bảng 3.3.1. Trên cơ sở đó, luận văn lưu lại mô hình đã lựa chọn và *pipeline* (pipeline) tiền xử lý dưới dạng các *tạo phẩm dữ liệu* (data artifacts), đảm bảo tính nhất quán khi áp dụng suy luận trên dữ liệu mới (Mục 3.7). Việc chuẩn hóa quy trình suy luận bằng các tệp lưu trữ mô hình/pipeline cũng góp phần tăng tính ứng dụng và tái lập của nghiên cứu.

Từ các kết quả của Chương 3, Chương 4 sẽ tiến hành mô phỏng và đánh giá trên tập test độc lập. Cụ thể, Chương 4 sẽ trình bày: (i) Bảng chỉ số đánh giá trên tập test cho từng mô hình (precision, recall, F1-score, accuracy và các chỉ số tổng hợp như ROC-AUC/PR-AUC), (ii) Ma trận nhầm lẫn để quan sát trực tiếp số lượng báo động giả và bỏ sót, (iii) Đường cong ROC và PR để đánh giá hành vi phân loại theo ngưỡng và làm rõ khả năng phân biệt lớp trong bối cảnh phát hiện webshell.

CHƯƠNG 4. MÔ PHỎNG VÀ ĐÁNH GIÁ KẾT QUẢ

4.1. Thiết kế đánh giá và chỉ số

4.1.1. Nguyên tắc đánh giá trên tập test độc lập

Mục tiêu của chương này là đánh giá khách quan khả năng phát hiện webshell của các mô hình đã lựa chọn ở Chương 3. Để đảm bảo tính khách quan, toàn bộ kết quả trong chương được báo cáo trên tập *kiểm thử* (test) độc lập đã được tách ra ngay từ Chương 2 theo thiết kế 70/15/15. Tập test không tham gia vào bất kỳ bước lựa chọn mô hình hay tham số nào ở Chương 3, do đó kết quả trên test phản ánh khả năng tổng quát hóa của mô hình trên dữ liệu chưa từng xuất hiện trong quá trình huấn luyện.

Trong luận văn, việc tiền xử lý dữ liệu (loại đặc trưng phương sai bằng 0 và chuẩn hóa thang đo) được đóng gói thành *pipeline* (pipeline) và chỉ được *fit* (fit) trên tập train. Tập validation và test chỉ được *transform* (transform) bằng pipeline đã fit nhằm tránh *rò rỉ dữ liệu* (data leakage). Nhờ đó, các đánh giá trình bày trong chương có tính nhất quán và tái lập.

4.1.2. Ma trận nhầm lẫn và các khái niệm TP/FP/TN/FN

Để phân tích chi tiết hành vi phân loại, luận văn sử dụng *ma trận nhầm lẫn* (confusion matrix) với bốn đại lượng cơ bản: *dương tính đúng* (true positive - TP), *dương tính sai* (false positive - FP), *âm tính đúng* (true negative - TN) và *âm tính sai* (false negative - FN). Trong bối cảnh phát hiện webshell, lớp `webshell` được xem là lớp dương (positive class). Do đó: TP là số mẫu webshell dự đoán đúng, FP là số mẫu normal bị dự đoán nhầm thành webshell (báo động giả), FN là số mẫu webshell bị dự đoán nhầm thành normal (bỏ sót), và TN là số mẫu normal dự đoán đúng.

Ma trận nhầm lẫn cho phép quan sát trực tiếp mức độ báo động giả và bỏ sót, từ đó hỗ trợ thảo luận ưu, nhược điểm của từng mô hình.

4.1.3. Các chỉ số đánh giá chính

Để đánh giá toàn diện, luận văn sử dụng nhóm chỉ số cơ bản và phổ biến trong bài toán phân loại nhị phân:

(i) *Độ chính xác* (accuracy): tỷ lệ dự đoán đúng trên toàn bộ mẫu.

(ii) *Độ đúng* (precision): tỷ lệ mẫu dự đoán là webshell mà thực sự là webshell,

chỉ số này phản ánh mức độ báo động giả.

(iii) *Độ bao phủ* (recall): tỷ lệ webshell được phát hiện đúng trên tổng số webshell, chỉ số này phản ánh mức độ bỏ sót.

(iv) *F1-score* (F1-score): trung bình điều hòa của precision và recall, phản ánh cân bằng giữa báo động giả và bỏ sót.

Ngoài ra, để quan sát hành vi phân loại theo ngưỡng và đánh giá chất lượng phân biệt lớp, luận văn sử dụng: (v) ROC-AUC và đường cong ROC (*receiver operating characteristic*), (vi) PR-AUC và đường cong PR (*precision–recall*). Các chỉ số và đường cong này hỗ trợ so sánh mô hình một cách trực quan, đặc biệt khi cân nhắc mức độ ưu tiên giữa giảm bỏ sót và giảm báo động giả.

4.1.4. Thiết lập thực nghiệm và tái lập kết quả

Toàn bộ kết quả trong chương được sinh tự động bằng kịch bản thực nghiệm được lập trình bằng ngôn ngữ Python, dựa trên cùng một tập test cố định và cùng pipeline tiền xử lý.

4.2. Đánh giá định lượng trên tập test

4.2.1. Kết quả đánh giá của mô hình hồi quy logistic

Trong mục này, luận văn trình bày kết quả đánh giá định lượng của mô hình *hồi quy logistic* (logistic regression) trên tập *kiểm thử* (test). Các chỉ số bao gồm *độ chính xác* (accuracy), *độ đúng* (precision), *độ bao phủ* (recall), *F1-score* (F1-score), cùng các chỉ số tổng hợp ROC-AUC và PR-AUC như đã mô tả ở Mục 4.1.3. Các chỉ số này cung cấp cái nhìn tổng quát về năng lực phát hiện webshell của mô hình trong điều kiện dữ liệu chưa từng xuất hiện trong huấn luyện.

Bảng 4.2.1 tổng hợp kết quả định lượng của mô hình hồi quy logistic trên tập test như sau.

Bảng 4.2.1. Kết quả đánh giá trên tập test của mô hình hồi quy logistic (Logistic Regression).

Chỉ số (Metric)	Giá trị (Value)
Accuracy (độ chính xác) (accuracy)	0.8960
Precision (độ đúng) (precision)	0.9338
Recall (độ bao phủ) (recall)	0.8523
F1-score (F1-score)	0.8912
ROC-AUC (ROC-AUC)	0.9337
PR-AUC (PR-AUC)	0.9487

4.2.2. Kết quả đánh giá của mô hình rừng ngẫu nhiên

Tương tự, mô hình *rừng ngẫu nhiên* (random forest) được đánh giá trên cùng tập test độc lập nhằm đảm bảo tính so sánh công bằng giữa các mô hình. Do rừng ngẫu nhiên có khả năng mô hình hóa quan hệ phi tuyến và tương tác đặc trưng, việc đánh giá định lượng giúp quan sát liệu mô hình này có cải thiện được cân bằng giữa *báo động giả* (false positive) và *bỏ sót* (false negative) so với mô hình tuyến tính hay không.

Bảng 4.2.2 tổng hợp kết quả định lượng của mô hình rừng ngẫu nhiên trên tập test như sau.

Bảng 4.2.2. Kết quả đánh giá trên tập test của mô hình rừng ngẫu nhiên (Random Forest).

Chỉ số (Metric)	Giá trị (Value)
Accuracy (độ chính xác) (accuracy)	0.9430
Precision (độ đúng) (precision)	0.9459
Recall (độ bao phủ) (recall)	0.9396
F1-score (F1-score)	0.9428
ROC-AUC (ROC-AUC)	0.9875
PR-AUC (PR-AUC)	0.9896

4.2.3. Nhận xét tổng quát

Từ các kết quả định lượng trên tập test, có thể nhận xét rằng cả hai mô hình đều thể hiện khả năng phân loại webshell ở mức nhất định, tuy nhiên mỗi mô hình có thể có ưu thế khác nhau tùy theo tiêu chí ưu tiên. Trong bối cảnh an toàn thông tin, việc lựa chọn mô hình không chỉ dựa trên giá trị tổng hợp như accuracy hay F1-score, mà còn cần xem xét trực tiếp mức *báo động giả* (false positive) và *bỏ sót* (false negative).

Do đó, các mục tiếp theo sẽ phân tích chi tiết hơn thông qua *ma trận nhầm lẫn* (confusion matrix) và các đường cong ROC/PR nhằm làm rõ hành vi dự đoán của mô hình theo ngưỡng và hỗ trợ lựa chọn mô hình phù hợp cho kịch bản triển khai.

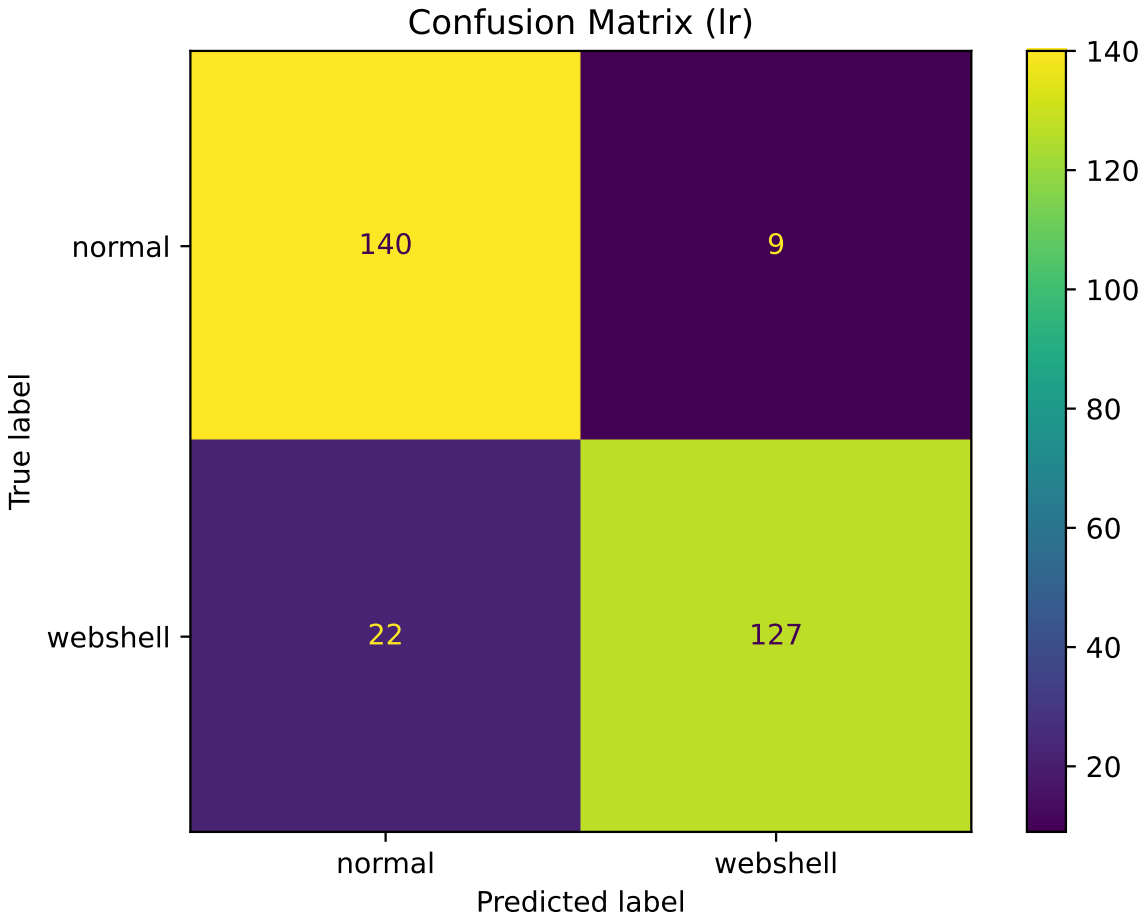
4.3. Phân tích ma trận nhầm lẫn

4.3.1. Ma trận nhầm lẫn của mô hình hồi quy logistic

Mặc dù các chỉ số tổng hợp như *F1-score* (F1-score) hay ROC-AUC cung cấp cái nhìn tổng quát, *ma trận nhầm lẫn* (confusion matrix) cho phép phân tích trực tiếp hai loại sai lệch quan trọng trong phát hiện webshell: *báo động giả* (false positive) và *bỏ sót* (false negative). Trong bối cảnh an toàn thông tin, bỏ sót webshell thường gây rủi ro cao hơn do mẫu độc hại có thể tiếp tục tồn tại trong hệ thống; trong khi đó báo

động giả làm tăng khối lượng kiểm tra thủ công.

Hình 4.3.1 trình bày ma trận nhầm lẫn của mô hình *hồi quy logistic* (logistic regression) trên tập test. Thông qua ma trận này, có thể quan sát số lượng mẫu webshell được phát hiện đúng (TP), số lượng mẫu `normal` bị cảnh báo nhầm (FP), số lượng webshell bị bỏ sót (FN) và số lượng normal dự đoán đúng (TN). Kết quả này bổ trợ cho các chỉ số định lượng ở Bảng 4.2.1 và giúp luận văn thảo luận rõ hơn về đặc tính lỗi của mô hình tuyến tính.



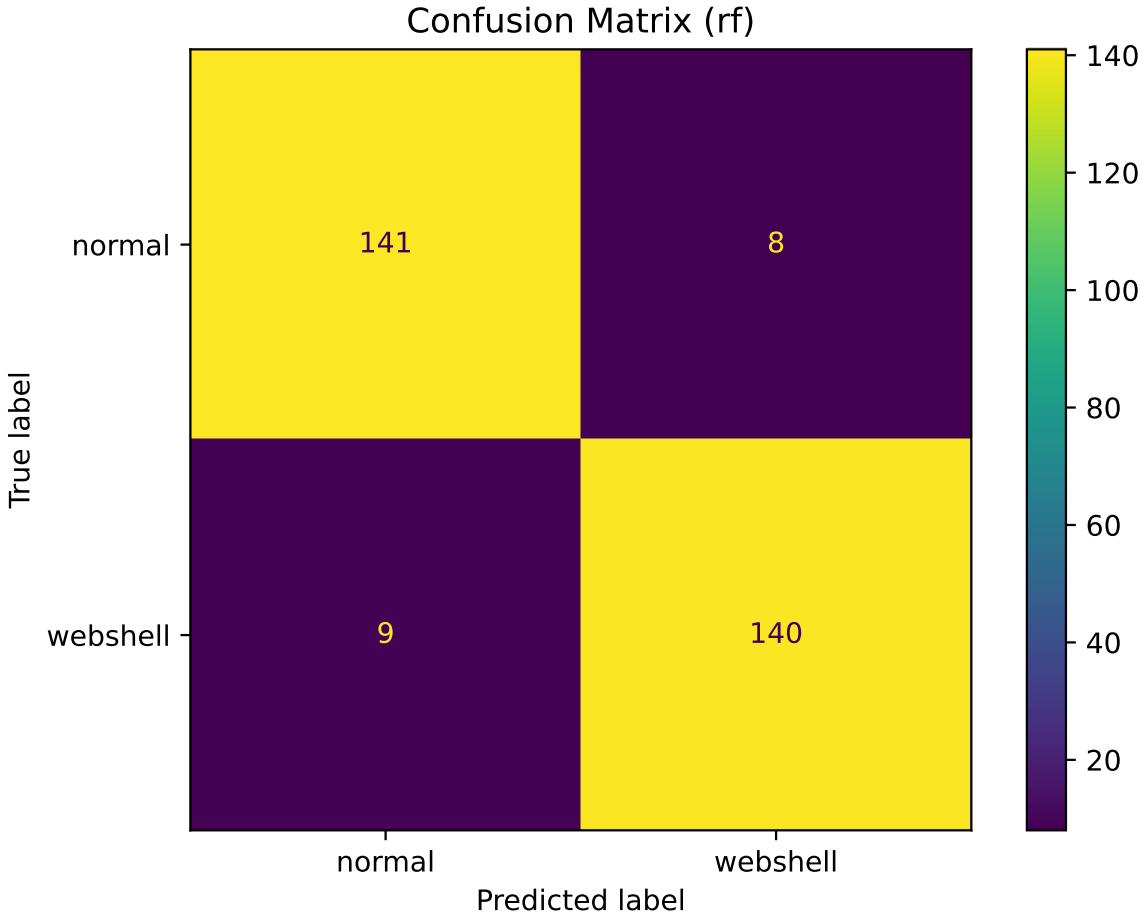
Hình 4.3.1. Ma trận nhầm lẫn (confusion matrix) của mô hình *hồi quy logistic* (logistic regression) trên tập test.

4.3.2. Ma trận nhầm lẫn của mô hình rừng ngẫu nhiên

Hình 4.3.2 trình bày ma trận nhầm lẫn của mô hình *rừng ngẫu nhiên* (random forest) trên tập test. Do rừng ngẫu nhiên có khả năng mô hình hóa quan hệ phi tuyến và kết hợp nhiều cây quyết định, mô hình này có thể biểu hiện khác biệt về mức báo động giả và bỏ sót so với hồi quy logistic.

Việc so sánh hai ma trận nhầm lẫn giúp làm rõ mô hình nào phù hợp hơn với

mục tiêu triển khai. Chẳng hạn, nếu ưu tiên giảm bỏ sót webshell (giảm FN), mô hình có FN thấp hơn sẽ được ưu tiên; ngược lại, nếu ưu tiên giảm báo động giả (giảm FP) để giảm tải kiểm tra thủ công, mô hình có FP thấp hơn sẽ phù hợp hơn. Phân tích này cho phép diễn giải kết quả theo hướng gắn với bối cảnh an toàn thông tin thay vì chỉ dựa trên một chỉ số tổng hợp.



Hình 4.3.2. Ma trận nhầm lẫn (confusion matrix) của mô hình *rừng ngẫu nhiên* (random forest) trên tập test.

4.3.3. Nhận xét và liên kết sang phân tích ROC/PR

Từ ma trận nhầm lẫn của hai mô hình, luận văn có thể rút ra nhận xét định tính về sự đánh đổi giữa báo động giả và bỏ sót. Tuy nhiên, ma trận nhầm lẫn chỉ phản ánh hiệu năng tại một ngưỡng phân loại cụ thể. Trong khi đó, ở các kịch bản triển khai khác nhau, ngưỡng cảnh báo có thể cần điều chỉnh để ưu tiên giảm bỏ sót hoặc giảm báo động giả.

Vì vậy, các mục tiếp theo sẽ sử dụng đường cong ROC và PR để đánh giá hành vi phân loại của mô hình theo nhiều ngưỡng khác nhau, qua đó cung cấp cơ sở lựa chọn mô hình và ngưỡng phù hợp cho bối cảnh phát hiện webshell.

4.4. Phân tích đường cong ROC và PR

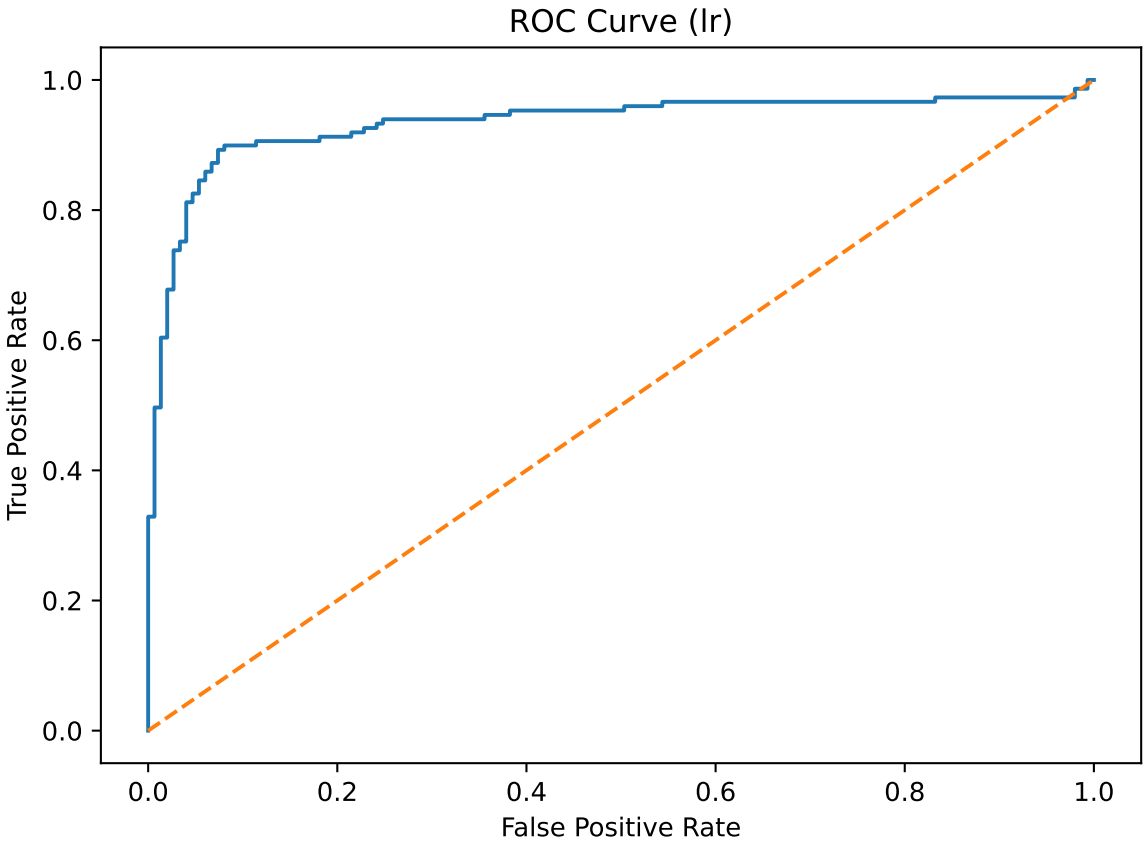
4.4.1. Đường cong ROC và chỉ số ROC-AUC

Đường cong ROC (receiver operating characteristic) mô tả mối quan hệ giữa *tỷ lệ dương tính giả* (false positive rate – FPR) và *tỷ lệ dương tính đúng* (true positive rate – TPR) khi thay đổi ngưỡng phân loại. Trong đó, TPR tương ứng với *độ bao phủ* (recall) đối với lớp dương webshell, còn FPR phản ánh mức báo động giả trên lớp `normal`. Chỉ số *ROC-AUC* (ROC-AUC) là diện tích dưới đường ROC, thể hiện khả năng phân biệt hai lớp một cách tổng quát theo nhiều ngưỡng khác nhau: ROC-AUC càng cao thì mô hình càng có khả năng xếp hạng mẫu webshell cao hơn mẫu normal.

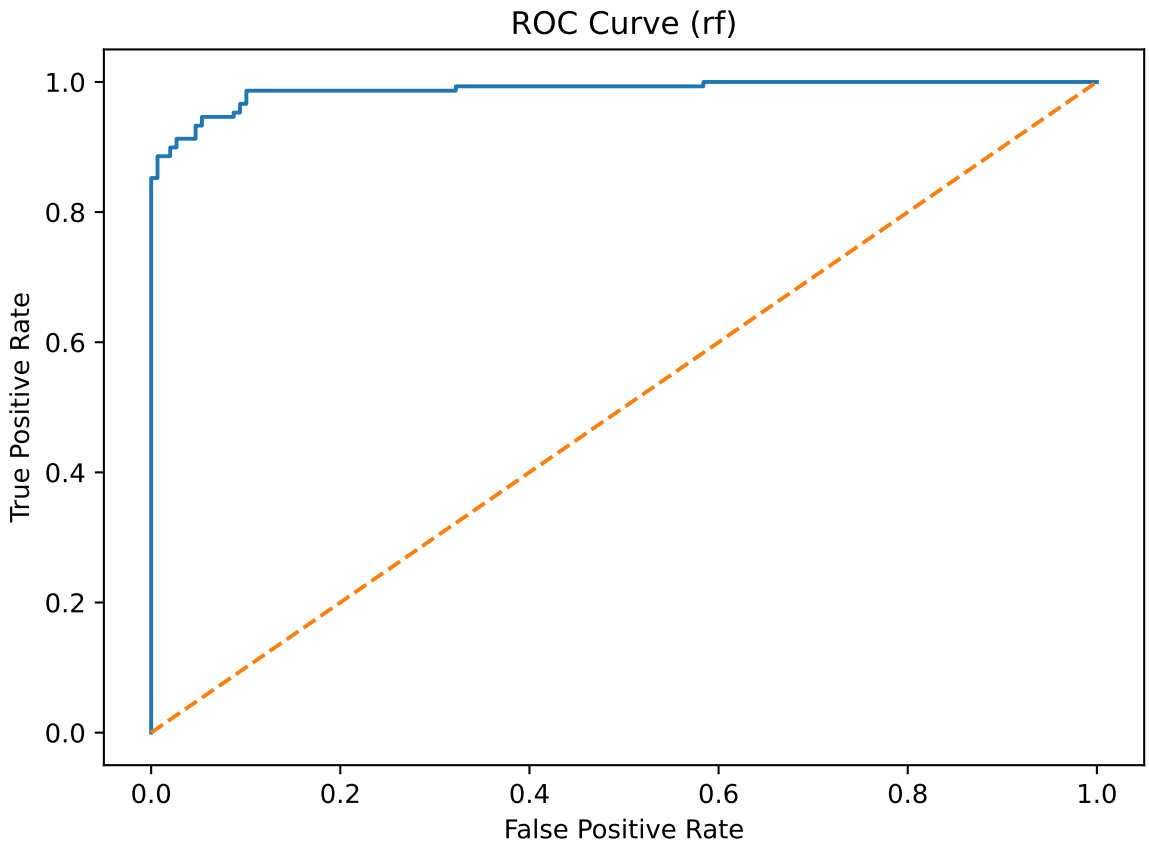
Trong luận văn, ROC được sử dụng nhằm quan sát hành vi phân loại khi điều chỉnh ngưỡng cảnh báo. Điều này quan trọng trong bối cảnh triển khai thực tế, khi người vận hành có thể ưu tiên giảm bỏ sót (tăng TPR) hoặc giảm báo động giả (giảm FPR) tùy theo yêu cầu hệ thống.

4.4.2. Đường cong ROC của các mô hình

Hình 4.4.1 và Hình 4.4.2 lần lượt trình bày đường cong ROC của mô hình *hồi quy logistic* (logistic regression) và mô hình *rừng ngẫu nhiên* (random forest) trên tập test. Các đường cong này bổ trợ cho chỉ số ROC-AUC đã được báo cáo trong Bảng 4.2.1 và Bảng 4.2.2, đồng thời cung cấp minh họa trực quan về sự đánh đổi giữa bỏ sót và báo động giả theo ngưỡng.



Hình 4.4.1. Đường cong ROC của mô hình *hồi quy logistic* (logistic regression) trên tập test.



Hình 4.4.2. Đường cong ROC của mô hình *rừng ngẫu nhiên* (random forest) trên tập test.

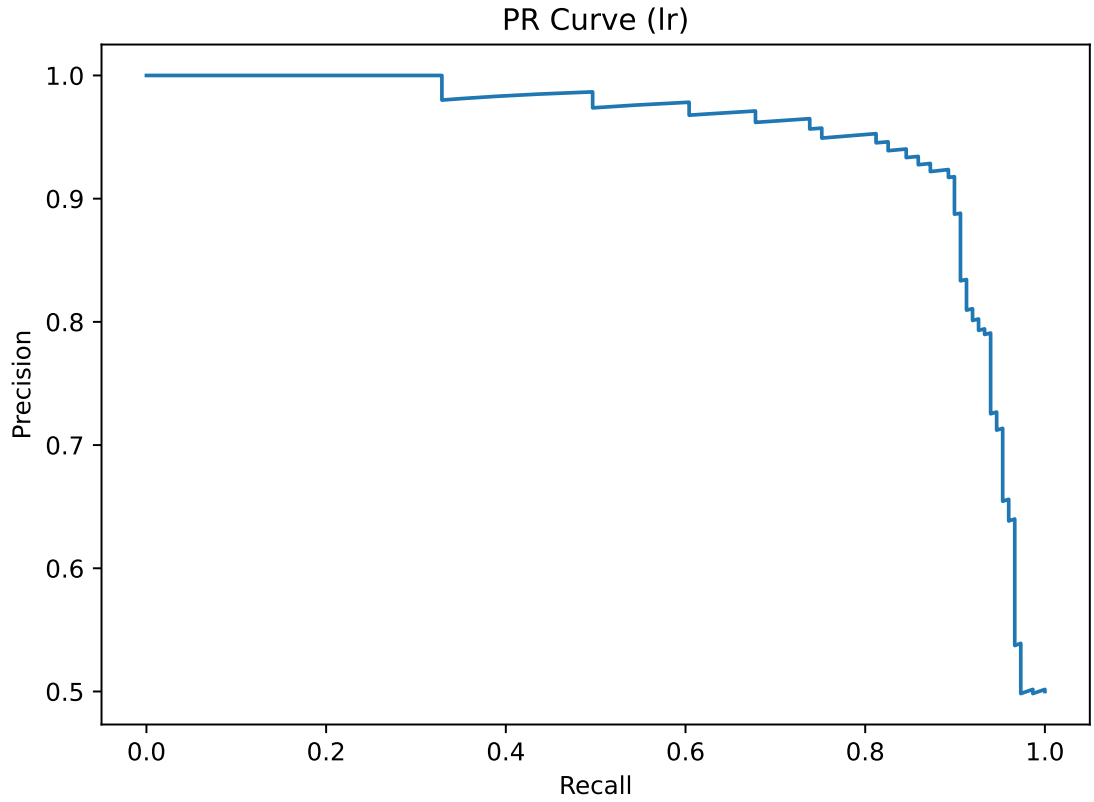
4.4.3. Đường cong PR và chỉ số PR-AUC

Đường cong PR (precision–recall) mô tả mối quan hệ giữa *độ đúng* (precision) và *độ bao phủ* (recall) khi thay đổi ngưỡng phân loại. So với ROC, đường PR thường được sử dụng để nhấn mạnh trực tiếp đánh đổi giữa báo động giả và bỏ sót, đặc biệt khi quan tâm đến lớp dương `webshell`. Chỉ số *PR-AUC* (PR-AUC) là diện tích dưới đường PR, phản ánh chất lượng phát hiện lớp dương theo nhiều ngưỡng.

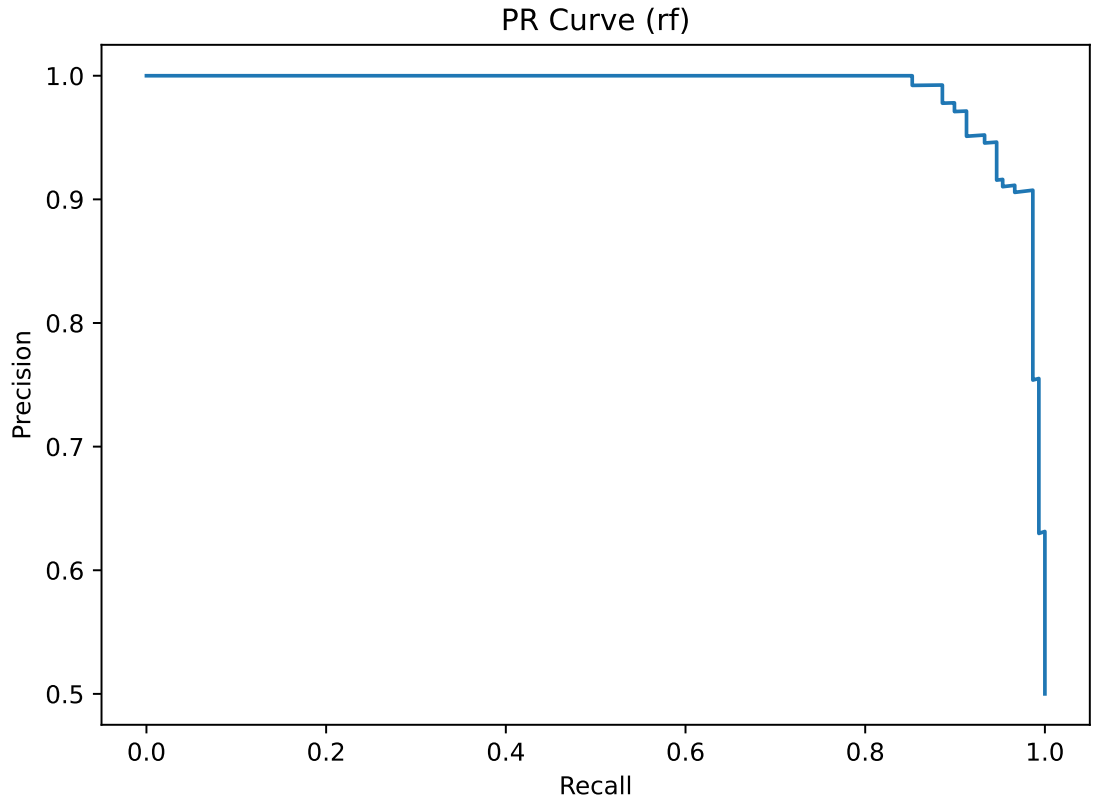
Trong bối cảnh phát hiện webshell, việc đồng thời quan sát PR-AUC và đường PR giúp luận văn đánh giá rõ hơn liệu mô hình có duy trì được precision tốt khi tăng recall hay không, từ đó hỗ trợ lựa chọn mô hình phù hợp với mục tiêu ưu tiên giảm bỏ sót hoặc giảm báo động giả.

4.4.4. Đường cong PR của các mô hình

Hình 4.4.3 và Hình 4.4.4 lần lượt trình bày đường cong PR của hai mô hình trên tập test. Các đường cong này bổ trợ cho chỉ số PR-AUC trong Bảng 4.2.1 và Bảng 4.2.2, đồng thời cung cấp minh họa trực quan để so sánh hành vi precision–recall giữa các mô hình.



Hình 4.4.3. Đường cong PR của mô hình *hồi quy logistic* (logistic regression) trên tập test.



Hình 4.4.4. Đường cong PR của mô hình *rừng ngẫu nhiên* (random forest) trên tập test.

4.4.5. Nhận xét tổng hợp từ ROC/PR

Kết hợp kết quả từ ma trận nhầm lẫn (Mục 4.3) và các đường cong ROC/PR, luận văn có thể đánh giá mô hình không chỉ tại một ngưỡng cố định mà còn theo xu hướng thay đổi ngưỡng. Điều này đặc biệt hữu ích trong bối cảnh triển khai, khi ngưỡng cảnh báo có thể điều chỉnh để ưu tiên giảm bỏ sót webshell hoặc giảm báo động giả.

Phần tiếp theo sẽ tổng hợp các kết quả và đưa ra nhận xét cuối cùng về mô hình phù hợp, đồng thời nêu các giới hạn và hướng phát triển trong tương lai.

4.5. Tổng hợp, so sánh và đánh giá kết quả thử nghiệm

4.5.1. Tổng hợp kết quả định lượng và định tính

Trong phần này, luận văn tổng hợp kết quả đánh giá trên tập *kiểm thử* (test) theo cả hai hướng định lượng và định tính. Nhìn chung, mô hình *rừng ngẫu nhiên* (random forest) thể hiện ưu thế nhất quán hơn so với *hồi quy logistic* (logistic regression) trên các chỉ số quan trọng đối với bài toán phát hiện webshell, trong khi mô hình hồi quy logistic vẫn đóng vai trò mô hình nền tảng (baseline) để tham chiếu về tính đơn giản và chi phí tính toán.

Kết quả đánh giá trên tập *kiểm thử* (test) đã được trình bày theo hai hướng: (i) đánh giá định lượng thông qua các chỉ số tổng hợp (Mục 4.2), và (ii) đánh giá định tính thông qua phân tích *ma trận nhầm lẫn* (confusion matrix) và các đường cong ROC/PR (Mục 4.3 và Mục 4.4).

Về mặt định lượng, Bảng 4.2.1 và Bảng 4.2.2 cho thấy hai mô hình *hồi quy logistic* (logistic regression) và *rừng ngẫu nhiên* (random forest) đều đạt được các chỉ số ở mức phù hợp cho bài toán phát hiện webshell dựa trên *đặc trưng tĩnh* (static features). Các chỉ số *precision* (precision), *recall* (recall) và *F1-score* (F1-score) phản ánh trực tiếp mức đánh đổi giữa báo động giả và bỏ sót, trong khi ROC-AUC và PR-AUC cung cấp góc nhìn tổng quát về khả năng phân biệt lớp theo nhiều ngưỡng khác nhau.

Về mặt định tính, Hình 4.3.1 và Hình 4.3.2 giúp quan sát trực tiếp số lượng *báo động giả* (false positives) và *bỏ sót* (false negatives) của từng mô hình tại ngưỡng phân loại mặc định. Đồng thời, các đường cong ROC (Hình 4.4.1, Hình 4.4.2) và PR (Hình 4.4.3, Hình 4.4.4) cung cấp thông tin về hành vi mô hình khi thay đổi ngưỡng, từ đó hỗ trợ lựa chọn mô hình/thiết lập phù hợp với ưu tiên triển khai.

Từ các kết quả tổng hợp trên, phần tiếp theo tập trung phân tích các khía cạnh triển khai và vận hành của hai mô hình, trong đó nhấn mạnh vì sao *rừng ngẫu nhiên* (random forest) là lựa chọn phù hợp hơn trong bối cảnh ưu tiên giảm *bỏ sót* (false negative) khi phát hiện webshell, đồng thời vẫn xem xét chi phí suy luận và mức *báo động giả* (false positive).

4.5.2. So sánh hai mô hình về khả năng triển khai

Trong bối cảnh phát hiện webshell, lựa chọn mô hình triển khai cần cân nhắc đồng thời hai yếu tố: (i) hiệu năng phát hiện, đặc biệt là giảm *bỏ sót* (false negative) đối với lớp `webshell`, và (ii) chi phí vận hành, bao gồm thời gian suy luận và mức *báo động giả* (false positive) dẫn đến chi phí rà soát thủ công. Do đó, việc so sánh mô hình trong luận văn không chỉ dựa trên một chỉ số tổng hợp, mà dựa trên tập hợp các chỉ số (Bảng 4.2.2, Bảng 4.2.1) và hành vi lỗi quan sát được ở ma trận nhầm lẫn (Hình 4.3.2, Hình 4.3.1).

Hồi quy logistic (logistic regression) có ưu điểm nổi bật về tính đơn giản, tốc độ suy luận nhanh và khả năng tích hợp thuận tiện vào các quy trình quét mã nguồn quy mô lớn. Với dữ liệu dạng *vector đặc trưng* (feature vector), mô hình tuyến tính cũng dễ giải trình ở mức tổng quan. Tuy nhiên, do giới hạn tuyến tính, mô hình có thể kém linh hoạt khi gặp các biến thể webshell có quan hệ phi tuyến hoặc tương tác đặc trưng phức tạp; điều này thể hiện ở hiệu năng phát hiện thấp hơn so với mô hình rừng ngẫu nhiên trên tập test (Bảng 4.2.1).

Rừng ngẫu nhiên (random forest) có khả năng mô hình hóa quan hệ phi tuyến và tương tác giữa các đặc trưng, nhờ đó thể hiện ưu thế rõ rệt về hiệu năng phát hiện trên tập test. Đặc biệt, việc đạt *recall* (recall) và *F1-score* (F1-score) cao hơn cho thấy mô hình có khả năng giảm bỏ sót webshell tốt hơn trong kịch bản triển khai thực tế, nơi bỏ sót thường gây rủi ro an ninh lớn. Mặc dù chi phí suy luận của rừng ngẫu nhiên có thể cao hơn hồi quy logistic, kết quả đánh giá cho thấy lợi ích về mặt giảm rủi ro an ninh là đáng kể trong phạm vi dữ liệu nghiên cứu.

Từ các phân tích trên, luận văn định hướng sử dụng *rừng ngẫu nhiên* (random forest) như mô hình triển khai khuyến nghị trong phạm vi nghiên cứu, trong khi *hồi quy logistic* (logistic regression) được sử dụng như mô hình nền tảng (baseline) để so sánh và làm mốc tham chiếu về tính đơn giản và chi phí tính toán.

4.5.3. Kết luận lựa chọn mô hình

Dựa trên kết quả đánh giá trên tập *kiểm thử* (test), luận văn lựa chọn *rừng ngẫu nhiên* (random forest) làm mô hình phát hiện webshell chính. Cơ sở của quyết định này thể hiện rõ qua các chỉ số định lượng trong Bảng 4.2.2 và Bảng 4.2.1.

Cụ thể, mô hình rừng ngẫu nhiên đạt *F1-score* (F1-score) bằng 0.9428, cao hơn đáng kể so với 0.8912 của *hồi quy logistic* (logistic regression). Bên cạnh đó, *độ bao phủ* (recall) của rừng ngẫu nhiên đạt 0.9396, trong khi hồi quy logistic chỉ đạt 0.8523. Trong bối cảnh an toàn thông tin, recall cao hơn đồng nghĩa với việc giảm *bỏ sót* (false negative) đối với lớp `webshell`, qua đó giảm rủi ro để webshell tồn tại mà không bị phát hiện. Đồng thời, rừng ngẫu nhiên cũng duy trì *độ đúng* (precision) ở mức cao (0.9459), cho thấy mô hình không đánh đổi quá nhiều bằng cách tăng *báo động giả* (false positive).

Ngoài các chỉ số tại ngưỡng phân loại mặc định, các chỉ số tổng hợp theo ngưỡng cũng ủng hộ lựa chọn này. Mô hình rừng ngẫu nhiên đạt ROC-AUC = 0.9875 và PR-AUC = 0.9896, đều cao hơn đáng kể so với ROC-AUC = 0.9337 và PR-AUC = 0.9487 của hồi quy logistic. Điều này cho thấy rừng ngẫu nhiên có khả năng phân biệt hai lớp `webshell` và `normal` tốt hơn một cách ổn định theo nhiều ngưỡng, phù hợp cho các kịch bản triển khai cần điều chỉnh ngưỡng cảnh báo.

Từ các phân tích trên, luận văn kết luận mô hình *rừng ngẫu nhiên* (random forest) là lựa chọn phù hợp hơn để sử dụng làm mô hình phát hiện webshell trong phạm vi dữ liệu và quy trình tiền xử lý của nghiên cứu. Mô hình *hồi quy logistic* (logistic regression) vẫn có giá trị như một mô hình nền tảng (baseline) nhờ tính đơn giản và tốc độ, tuy nhiên không được chọn làm mô hình khuyến nghị chính do hiệu năng phát hiện thấp hơn rõ rệt trên tập test.

4.5.4. Giới hạn và hướng phát triển

Mặc dù đạt được kết quả khả quan trên tập test, nghiên cứu vẫn còn một số giới hạn:

Thứ nhất, bộ dữ liệu sử dụng trong luận văn chỉ gồm 1.984 mẫu và được cân bằng giữa hai lớp (992 webshell và 992 mẫu bình thường), trong khi ở môi trường thực tế, webshell thường chiếm tỷ lệ rất nhỏ so với tổng số tệp mã nguồn. Vì vậy, kết quả thực nghiệm trên bộ dữ liệu cân bằng chưa phản ánh đầy đủ hiệu quả của mô hình khi triển khai trong điều kiện dữ liệu lệch lớp.

Thứ hai, dữ liệu được biểu diễn theo các *đặc trưng tĩnh* (static features) nên hiệu quả của mô hình phụ thuộc vào chất lượng của quá trình trích xuất đặc trưng. Các đặc trưng này chưa phản ánh được các thông tin động trong quá trình thực thi chương trình như luồng dữ liệu, lời gọi hệ thống hoặc hành vi thời gian chạy, do đó khả năng phát hiện một số biến thể webshell sử dụng kỹ thuật làm rối phức tạp còn hạn chế.

Thứ ba, mô hình mới được đánh giá trên một bộ dữ liệu công khai duy nhất và chưa được kiểm chứng trên các bộ dữ liệu độc lập hoặc các mẫu webshell có mức độ obfuscation khác nhau. Do đó, khả năng tổng quát hóa của mô hình đối với các biến thể webshell mới vẫn cần được tiếp tục đánh giá trong các nghiên cứu tiếp theo.

Các nghiên cứu gần đây cho thấy xu hướng phát hiện webshell đang chuyển dần sang các mô hình học sâu và các mô hình biểu diễn mã nguồn hiện đại như Transformer, CodeBERT hoặc Graph Neural Network. Các phương pháp này có khả năng tự học đặc trưng từ mã nguồn và có tiềm năng nâng cao khả năng phát hiện các biến thể webshell mới, tuy nhiên thường yêu cầu bộ dữ liệu lớn, chi phí huấn luyện cao và tài nguyên tính toán đáng kể. Trong phạm vi của luận văn, Logistic Regression và Random Forest vẫn là lựa chọn phù hợp nhờ khả năng diễn giải tốt, tốc độ huấn luyện nhanh và thuận lợi khi triển khai trên bộ dữ liệu có quy mô vừa phải.

Trong tương lai, hướng phát triển có thể bao gồm: mở rộng bộ dữ liệu và đánh giá trên các tập dữ liệu độc lập nhằm kiểm chứng khả năng tổng quát hóa của mô hình; kết hợp các đặc trưng động với các đặc trưng tĩnh; khảo sát ảnh hưởng của dữ liệu lệch lớp và các ngưỡng phân loại khác nhau; đồng thời nghiên cứu, so sánh với các mô hình hiện đại như Transformer, CodeBERT hoặc Graph Neural Network trên các bộ dữ liệu có quy mô lớn hơn.

DANH MỤC CÔNG TRÌNH ĐÃ CÔNG BỐ

[CB1] Trần Tuấn Long, Nguyễn Thị Như Quỳnh, Nguyễn Văn Tăng, Ngô Hải Anh. *Nghiên cứu phương pháp phát hiện webshell bằng học máy*. Kỷ yếu Hội thảo quốc gia lần thứ XXVIII: Một số vấn đề chọn lọc của Công nghệ thông tin và truyền thông, Ninh Bình, 577–582, 15–16/11/2025.

TÀI LIỆU THAM KHẢO

1. Abdelhakim Hannousse and Salima Yahiouche, *Handling webshell attacks: A systematic mapping and survey*, Computers & Security, 2021, 108, 102366.
2. Zulie Pan, Yuanchao Chen, Yu Chen, Yi Shen, and Xuanzhen Guo, *Webshell Detection Based on Executable Data Characteristics of PHP Code*, Wireless Communications and Mobile Computing, 2021, 2021(1), 5533963.
3. Jiazhen Zhao, Yuliang Lu, Xin Wang, Kailong Zhu, and Lu Yu, *WTA: A Static Taint Analysis Framework for PHP Webshell*, Applied Sciences, 2021, 11(16).
4. Ngoc-Hoa Nguyen, Viet-Ha Le, Van-On Phung, and Phuong-Hanh Du, *Toward a Deep Learning Approach for Detecting PHP Webshell*, In Proceedings of the 10th International Symposium on Information and Communication Technology, 2019, SoICT '19, page 514–521, New York, NY, USA. Association for Computing Machinery.
5. Feijiang Han, Jiaming Zhang, Chuyi Deng, Jianheng Tang, and Yunhuai Liu, *Can LLMs Handle WebShell Detection? Overcoming Detection Challenges with Behavioral Function-Aware Framework*, In Second Conference on Language Modeling (COLM), 2025.
6. Le Viet Ha. *Enhancing Webshell Detection with Deep Learning-Powered Methods*. Ph.d. dissertation, Vietnam National University Hanoi, University of Engineering and Technology, 2024.
7. Yifan Zhang, Haiyan Kang, and Qiang Wang, *MMFDetect: Webshell Evasion Detect Method Based on Multimodal Feature Fusion*, Electronics, 2025, 14(3).
8. Zhiqiang Wang, Haoyu Wang, Shaowei Yuan, and Zhiang Tian, *Incremental learning research for webshell detection*, Journal of King Saud University Computer and Information Sciences, Oct 2025, 37(8), 249.
9. Khaled Suwais, Adnan A. Hnaif, and Sally Almanasra, *An Alternative Static Taint Analysis Framework to Detect PHP Web Shell-Based Web Attacks*, International Journal of Advances in Soft Computing and its Applications, 2023, 15(3).

10. Khaled Suwais, Adnan A. Hnaif, and Sally Almanasra, *An Alternative Static Taint Analysis Framework to Detect PHP Web Shell-Based Web Attacks*, International Journal of Advances in Soft Computing and its Applications, 2023, 15(3). Print ISSN: 2710-1274, Online ISSN: 2074-8523.

11. Xuyan Song, Yiting Qin, Xinyao Liu, Baojiang Cui, and Junsong Fu, *Jshelldetector: A java fileless webshell detector based on program analysis*, Computers, Materials and Continua, 2023, 75(1), 2061–2078.

12. Yixin Wu, Yuqiang Sun, Cheng Huang, Peng Jia, and Luping Liu, *Session-Based Webshell Detection Using Machine Learning in Web Logs*, Security and Communication Networks, 2019, 2019(1), 3093809.

13. Shun Fu, Hao Li, Panpan Zhu, Jian Tong, Jinye Yang, and Ji Xu, *WeDIGAR: A Light-Weighted Webshell Detection Framework for Satellite and UAV Networks*, Electronics, 2025, 14(21).

14. Dian Anggraini and Abba Suganda Girsang, *Webshell Detection Based on Bytecode Feature with Convolutional Neural Network*, Journal of Theoretical and Applied Information Technology, 2023, 101(18). ISSN 1992-8645.

15. Zhiqiang Liu, Daofeng Li, and Lulu Wei, *A New Method for WebShell Detection Based on Bidirectional GRU and Attention Mechanism*, Security and Communication Networks, 2022, 2022(1), 3434920.

16. Guan-Yu Wang, Hung-Jui Ko, Chang-Po Chiang, and Wei-Jen Wang, *WebShell Detection Based on CodeBERT and Deep Learning Model*, In Proceedings of the 2024 5th International Conference on Computing, Networks and Internet of Things, 2024, CNIOT '24, page 484–489, New York, NY, USA. Association for Computing Machinery.

17. Xin Liu, Yingli Zhang, Qingchen Yu, Jiajun Min, Jun Shen, Rui Zhou, and Qingguo Zhou, *SmartEagleEye: A Cloud-Oriented Webshell Detection System Based on Dynamic Gray-Box and Deep Learning*, Tsinghua Science and Technology, 2024, 29(3), 766–783.

18. Shun Fu, Hao Li, Panpan Zhu, Jian Tong, Jinye Yang, and Ji Xu, *WeDIGAR: A Light-Weighted Webshell Detection Framework for Satellite and UAV Networks*, Electronics, 2025, 14(21).

19. Zhiqiang Wang, Haoyu Wang, Shaowei Yuan, and Zhiang Tian, *Incremental Learning Research for Webshell Detection*, Journal of King Saud University Computer and Information Sciences, 2025, 37(8), 249.

20. Abdelhakim Hannousse. *Cleaned PHP Webshell dataset*. `https://github.com/hannousse/Cleaned-PHP-Webshell-dataset`, 2026. Accessed: January, 2026.

MÃ NGUỒN PYTHON SỬ DỤNG TRONG LUẬN VĂN

```python
import os, json, warnings
warnings.filterwarnings("ignore")

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from google.colab import drive

from sklearn.model_selection import train_test_split
from sklearn.feature_selection import VarianceThreshold
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.metrics import (
    confusion_matrix, ConfusionMatrixDisplay,
    accuracy_score, precision_score, recall_score, f1_score,
    roc_auc_score, roc_curve,
    average_precision_score, precision_recall_curve
)
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
import joblib

# 0. Mount Google Drive.
drive.mount('/content/drive')

# 1. Config paths.
DRIVE_ROOT = "/content/drive/MyDrive/webshell/dataset"
INPUT_CSV = os.path.join(DRIVE_ROOT, "php_webshell_dataset.csv")

OUT_ROOT = os.path.join(DRIVE_ROOT, "output_data")
OUT_FIG = os.path.join(OUT_ROOT, "figures")
OUT_TAB = os.path.join(OUT_ROOT, "tables")
OUT_ART = os.path.join(OUT_ROOT, "artifacts")
OUT_LOG = os.path.join(OUT_ROOT, "logs")

for p in [OUT_ROOT, OUT_FIG, OUT_TAB, OUT_ART, OUT_LOG]:
    os.makedirs(p, exist_ok=True)

def safe_print(msg):
    print(msg)

# 2. Helpers: LaTeX table writer.
def write_latex_table_full(filepath, caption, label, df, col_format):
    tabular = df.to_latex(index=False, escape=True, column_format=col_format)
```

```
    tabular = tabular.replace("\\toprule", "\\hline").replace("\\midrule",
    ↪  "\\hline").replace("\\bottomrule", "\\hline")
    content = f"""% Auto-generated by Colab (one-cell thesis script)
\\begin{{table}}[!t]
\\centering
\\resizebox{{\\linewidth}}{{!}}{{%
{tabular}
}}%
\\caption{{{caption}}}
\\label{{{label}}}
\\end{{table}}
"""
    with open(filepath, "w", encoding="utf-8") as f:
        f.write(content)

def save_bar_label_distribution(series_counts, out_pdf):
    plt.figure()
    series_counts.plot(kind="bar")
    plt.title("Label Distribution")
    plt.xlabel("Label")
    plt.ylabel("Count")
    plt.tight_layout()
    plt.savefig(out_pdf, bbox_inches="tight")
    plt.close()

def save_hist_by_class(df, label_col, feature, out_pdf, bins=40):
    if feature not in df.columns:
        return False
    plt.figure()
    for cls in sorted(df[label_col].unique()):
        df[df[label_col] == cls][feature].plot(kind="hist", bins=bins, alpha=0.5, label=cls)
    plt.title(f"Histogram of {feature} by class")
    plt.xlabel(feature)
    plt.ylabel("Frequency")
    plt.legend()
    plt.tight_layout()
    plt.savefig(out_pdf, bbox_inches="tight")
    plt.close()
    return True

def plot_and_save_confusion(y_true, y_pred, labels, title, out_pdf):
    cm = confusion_matrix(y_true, y_pred, labels=labels)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels)
    plt.figure()
    disp.plot(values_format="d")
    plt.title(title)
    plt.tight_layout()
    plt.savefig(out_pdf, bbox_inches="tight")
    plt.close()
    return cm

```

```python
def compute_metrics(y_true, y_pred, y_score, positive_label):
    acc = accuracy_score(y_true, y_pred)
    prec = precision_score(y_true, y_pred, pos_label=positive_label, zero_division=0)
    rec = recall_score(y_true, y_pred, pos_label=positive_label, zero_division=0)
    f1v = f1_score(y_true, y_pred, pos_label=positive_label, zero_division=0)

    y_bin = (np.array(y_true) == positive_label).astype(int)
    roc_auc = roc_auc_score(y_bin, y_score)
    pr_auc = average_precision_score(y_bin, y_score)
    return {
        "Accuracy": float(acc),
        "Precision": float(prec),
        "Recall": float(rec),
        "F1": float(f1v),
        "ROC-AUC": float(roc_auc),
        "PR-AUC": float(pr_auc)
    }

def save_roc_curve(y_true, y_score, positive_label, out_pdf, title):
    y_bin = (np.array(y_true) == positive_label).astype(int)
    fpr, tpr, _ = roc_curve(y_bin, y_score)
    plt.figure()
    plt.plot(fpr, tpr)
    plt.plot([0, 1], [0, 1], linestyle="--")
    plt.title(title)
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.tight_layout()
    plt.savefig(out_pdf, bbox_inches="tight")
    plt.close()

def save_pr_curve(y_true, y_score, positive_label, out_pdf, title):
    y_bin = (np.array(y_true) == positive_label).astype(int)
    prec, rec, _ = precision_recall_curve(y_bin, y_score)
    plt.figure()
    plt.plot(rec, prec)
    plt.title(title)
    plt.xlabel("Recall")
    plt.ylabel("Precision")
    plt.tight_layout()
    plt.savefig(out_pdf, bbox_inches="tight")
    plt.close()

# 3. Load dataset.
if not os.path.exists(INPUT_CSV):
    raise FileNotFoundError(f"Cannot find input CSV at: {INPUT_CSV}")

df = pd.read_csv(INPUT_CSV)
if "label" not in df.columns:
    raise ValueError("CSV must contain a 'label' column.")

```

```python
labels = sorted(df["label"].unique().tolist())
if len(labels) != 2:
    raise ValueError(f"Expected binary labels, got: {labels}")

positive_label = "webshell" if "webshell" in labels else labels[1]

X_raw = df.drop(columns=["label"]).copy()
y_raw = df["label"].copy()

missing_cells = int(df.isna().sum().sum())
duplicate_rows = int(df.duplicated().sum())
label_counts = df["label"].value_counts()

n_samples = int(df.shape[0])
n_cols = int(df.shape[1])
n_features = int(n_cols - 1)
n_classes = int(df["label"].nunique())

nunique = X_raw.nunique(dropna=False)
constant_cols = nunique[nunique == 1].index.tolist()
n_constant = int(len(constant_cols))

# 4. CHAPTER 2 outputs.
safe_print("=== Chapter 2: Generating tables, figures, artifacts ===")

fig_label = os.path.join(OUT_FIG, "fig_label_distribution.pdf")
save_bar_label_distribution(label_counts, fig_label)

fig_entropy = os.path.join(OUT_FIG, "fig_feature_entropy_hist.pdf")
fig_longest = os.path.join(OUT_FIG, "fig_feature_longestword_hist.pdf")
entropy_ok = save_hist_by_class(df, "label", "Entropy", fig_entropy)
longest_ok = save_hist_by_class(df, "label", "LongestWord", fig_longest)

tab1 = pd.DataFrame([
    ["Số mẫu (Number of samples)", str(n_samples)],
    ["Số cột (Number of columns)", f"{n_cols} (gồm {n_features} đặc trưng (features) + 1 nhãn (label))"],
    ["Số lớp (Number of classes)", str(n_classes)],
    ["Phân bố nhãn (Class distribution)", ", ".join([f"{k}: {v}" for k, v in label_counts.to_dict().items()])],
    ["Số ô thiếu (Missing cells)", str(missing_cells)],
    ["Số dòng trùng (Duplicate rows)", str(duplicate_rows)],
], columns=["Thuộc tính (Attribute)", "Giá trị (Value)"])

write_latex_table_full(
    filepath=os.path.join(OUT_TAB, "tab_dataset_summary.tex"),
    caption="Tổng quan tập dữ liệu CSV sử dụng trong luận văn.",
    label="tab:dataset-summary",
    df=tab1,
    col_format="|p{0.40\\linewidth}|p{0.55\\linewidth}|"
)
```

```
196
197 tab2 = pd.DataFrame([
198     ["Tổng số ô thiếu (Missing cells)", str(missing_cells)],
199     ["Số dòng trùng (Duplicate rows)", str(duplicate_rows)],
200     ["Số cột hằng (Constant columns)", str(n_constant)],
201     ["Số đặc trưng sau loại cột hằng (Remaining features)", str(n_features - n_constant)],
202 ], columns=["Chỉ số kiểm định (Quality check)", "Kết quả (Result)"])
203
204 write_latex_table_full(
205     filepath=os.path.join(OUT_TAB, "tab_data_quality.tex"),
206     caption="Kết quả kiểm định chất lượng và sàng lọc đặc trưng ở mức cơ bản.",
207     label="tab:data-quality",
208     df=tab2,
209     col_format="|p{0.55\\linewidth}|p{0.35\\linewidth}|"
210 )
211
212 pd.Series(constant_cols, name="constant_column").to_csv(os.path.join(OUT_ART,
    ↪  "constant_columns.csv"), index=False)
213
214 X_temp, X_test, y_temp, y_test = train_test_split(
215     X_raw, y_raw, test_size=0.15, random_state=42, stratify=y_raw
216 )
217 X_train, X_val, y_train, y_val = train_test_split(
218     X_temp, y_temp, test_size=(0.15 / 0.85), random_state=42, stratify=y_temp
219 )
220
221 split_info = pd.DataFrame([
222     ["Train (huấn luyện) (training)", "70%", f"{X_train.shape[0]}",
223      ", ".join([f"{k}: {v}" for k, v in y_train.value_counts().to_dict().items()])],
224     ["Validation (xác thực) (validation)", "15%", f"{X_val.shape[0]}",
225      ", ".join([f"{k}: {v}" for k, v in y_val.value_counts().to_dict().items()])],
226     ["Test (kiểm thử) (test)", "15%", f"{X_test.shape[0]}",
227      ", ".join([f"{k}: {v}" for k, v in y_test.value_counts().to_dict().items()])],
228 ], columns=["Tập con (Subset)", "Tỷ lệ (Ratio)", "Số mẫu (Count)", "Phân bố nhãn (Class
    ↪  distribution)"])
229
230 write_latex_table_full(
231     filepath=os.path.join(OUT_TAB, "tab_split_summary.tex"),
232     caption="Thiết kế chia tập dữ liệu phục vụ Chương 3 và Chương 4.",
233     label="tab:split-summary",
234     df=split_info,
235
        ↪  col_format="|p{0.30\\linewidth}|p{0.12\\linewidth}|p{0.15\\linewidth}|p{0.38\\linewidth}|"
236 )
237
238 preprocess = Pipeline(steps=[
239     ("var", VarianceThreshold(threshold=0.0)),
240     ("scaler", StandardScaler())
241 ])
242
243 X_train_p = preprocess.fit_transform(X_train)
```

```
X_val_p = preprocess.transform(X_val)
X_test_p = preprocess.transform(X_test)

selected_mask = preprocess.named_steps["var"].get_support()
selected_features = X_train.columns[selected_mask].tolist()

np.savez(
    os.path.join(OUT_ART, "dataset_splits.npz"),
    X_train=X_train_p, X_val=X_val_p, X_test=X_test_p,
    y_train=y_train.values, y_val=y_val.values, y_test=y_test.values
)
joblib.dump(preprocess, os.path.join(OUT_ART, "preprocess.joblib"))
pd.Series(selected_features, name="selected_feature").to_csv(os.path.join(OUT_ART, "selected_features.csv"), index=False)

# 5. CHAPTER 3: Basic ML models, simple selection on validation.
safe_print("\n=== Chapter 3: Training basic ML models and selecting by validation F1 ===")

lr_C_list = [0.1, 1.0, 10.0]
rf_n_list = [100, 300]
rf_depth_list = [None, 10]

best_lr, best_lr_info, best_lr_f1 = None, None, -1.0
for C in lr_C_list:
    lr = LogisticRegression(max_iter=2000, C=C, solver="liblinear")
    lr.fit(X_train_p, y_train)
    y_val_pred = lr.predict(X_val_p)
    f1v = f1_score(y_val, y_val_pred, pos_label=positive_label, zero_division=0)
    if f1v > best_lr_f1:
        best_lr_f1 = float(f1v)
        best_lr = lr
        best_lr_info = {"model": "LogisticRegression", "C": C, "val_F1": float(f1v)}

best_rf, best_rf_info, best_rf_f1 = None, None, -1.0
for n_estimators in rf_n_list:
    for max_depth in rf_depth_list:
        rf = RandomForestClassifier(
            n_estimators=n_estimators,
            max_depth=max_depth,
            random_state=42,
            n_jobs=-1
        )
        rf.fit(X_train_p, y_train)
        y_val_pred = rf.predict(X_val_p)
        f1v = f1_score(y_val, y_val_pred, pos_label=positive_label, zero_division=0)
        if f1v > best_rf_f1:
            best_rf_f1 = float(f1v)
            best_rf = rf
            best_rf_info = {"model": "RandomForest", "n_estimators": n_estimators,
            ↪  "max_depth": str(max_depth), "val_F1": float(f1v)}
```

```python
joblib.dump(best_lr, os.path.join(OUT_ART, "best_lr.joblib"))
joblib.dump(best_rf, os.path.join(OUT_ART, "best_rf.joblib"))

sel_df = pd.DataFrame([
    ["Logistic Regression", f"C={best_lr_info['C']}", f"{best_lr_info['val_F1']:.4f}"],
    ["Random Forest", f"n={best_rf_info['n_estimators']}, depth={best_rf_info['max_depth']}", f"{best_rf_info['val_F1']:.4f}"],
], columns=["Mô hình (Model)", "Thiết lập (Setting)", "F1 trên validation (Validation F1)"])

write_latex_table_full(
    filepath=os.path.join(OUT_TAB, "tab_model_selection.tex"),
    caption="Kết quả lựa chọn mô hình/tham số đơn giản dựa trên tập validation.",
    label="tab:model-selection",
    df=sel_df,
    col_format="|p{0.30\\linewidth}|p{0.45\\linewidth}|p{0.18\\linewidth}|"
)

# 6. CHAPTER 4: Final evaluation on TEST set.
safe_print("\n=== Chapter 4: Final evaluation on TEST set (confusion matrix, ROC/PR, metrics tables) ===")

def eval_and_export(model, short_name):
    y_pred = model.predict(X_test_p)

    if hasattr(model, "predict_proba"):
        proba = model.predict_proba(X_test_p)
        class_list = model.classes_.tolist()
        pos_idx = class_list.index(positive_label)
        y_score = proba[:, pos_idx]
    else:
        if hasattr(model, "decision_function"):
            y_score = model.decision_function(X_test_p)
            y_score = (y_score - y_score.min()) / (y_score.max() - y_score.min() + 1e-12)
        else:
            y_score = (y_pred == positive_label).astype(float)

    fig_cm = os.path.join(OUT_FIG, f"fig_confusion_{short_name}.pdf")
    plot_and_save_confusion(y_test.values, y_pred, labels, f"Confusion Matrix ({short_name})", fig_cm)

    fig_roc = os.path.join(OUT_FIG, f"fig_roc_{short_name}.pdf")
    save_roc_curve(y_test.values, y_score, positive_label, fig_roc, f"ROC Curve ({short_name})")

    fig_pr = os.path.join(OUT_FIG, f"fig_pr_{short_name}.pdf")
    save_pr_curve(y_test.values, y_score, positive_label, fig_pr, f"PR Curve ({short_name})")

    m = compute_metrics(y_test.values, y_pred, y_score, positive_label)

    metrics_df = pd.DataFrame([
        ["Accuracy (độ chính xác) (accuracy)", f"{m['Accuracy']:.4f}"],
```

```
        ["Precision (độ đúng) (precision)", f"{m['Precision']:.4f}"],
        ["Recall (độ bao phủ) (recall)", f"{m['Recall']:.4f}"],
        ["F1-score (F1-score)", f"{m['F1']:.4f}"],
        ["ROC-AUC (ROC-AUC)", f"{m['ROC-AUC']:.4f}"],
        ["PR-AUC (PR-AUC)", f"{m['PR-AUC']:.4f}"],
    ], columns=["Chỉ số (Metric)", "Giá trị (Value)"])

    tab_path = os.path.join(OUT_TAB, f"tab_test_metrics_{short_name}.tex")
    write_latex_table_full(
        filepath=tab_path,
        caption=f"Kết quả đánh giá trên tập test của mô hình {short_name}.",
        label=f"tab:test-metrics-{short_name}",
        df=metrics_df,
        col_format="|p{0.60\\linewidth}|p{0.30\\linewidth}|"
    )

    return {
        "model": short_name,
        "metrics": m,
        "fig_cm": fig_cm,
        "fig_roc": fig_roc,
        "fig_pr": fig_pr,
        "tab_metrics": tab_path
    }

res_lr = eval_and_export(best_lr, "lr")
res_rf = eval_and_export(best_rf, "rf")

# 7. Logs + what to include in LaTeX.
run_summary = {
    "input_csv": INPUT_CSV,
    "output_root": OUT_ROOT,
    "chapter2": {
        "tables": {
            "tab_dataset_summary": os.path.join(OUT_TAB, "tab_dataset_summary.tex"),
            "tab_data_quality": os.path.join(OUT_TAB, "tab_data_quality.tex"),
            "tab_split_summary": os.path.join(OUT_TAB, "tab_split_summary.tex"),
        },
        "figures": {
            "fig_label_distribution": fig_label,
            "fig_feature_entropy_hist": fig_entropy if entropy_ok else None,
            "fig_feature_longestword_hist": fig_longest if longest_ok else None,
        }
    },
    "chapter3": {
        "model_selection_table": os.path.join(OUT_TAB, "tab_model_selection.tex"),
        "best_lr_info": best_lr_info,
        "best_rf_info": best_rf_info,
    },
    "chapter4": {
        "lr": res_lr,
```

```
        "rf": res_rf
    }
}

with open(os.path.join(OUT_LOG, "run_summary.json"), "w", encoding="utf-8") as f:
    json.dump(run_summary, f, ensure_ascii=False, indent=2)

safe_print("\n=== FINISHED. Outputs are in: /content/drive/MyDrive/output_data ===\n")
safe_print("CHAPTER 2 (Tables):")
safe_print("- \\input{output_data/tables/tab_dataset_summary.tex}   % Table 2.1")
safe_print("- \\input{output_data/tables/tab_data_quality.tex}      % Table 2.2")
safe_print("- \\input{output_data/tables/tab_split_summary.tex}     % Table 2.3")
safe_print("\nCHAPTER 2 (Figures):")
safe_print("- \\includegraphics[width=\\linewidth]{output_data/figures/fig_label_distribution.pdf} % Figure 2.1")
if entropy_ok:
    safe_print("- \\includegraphics[width=\\linewidth]{output_data/figures/fig_feature_entropy_hist.pdf} % Figure 2.2a")
if longest_ok:
    safe_print("- \\includegraphics[width=\\linewidth]{output_data/figures/fig_feature_longestword_hist.pdf} % Figure 2.2b")

safe_print("\nCHAPTER 3 (Table):")
safe_print("- \\input{output_data/tables/tab_model_selection.tex} % model selection on validation")

safe_print("\nCHAPTER 4 (Tables):")
safe_print("- \\input{output_data/tables/tab_test_metrics_lr.tex} % LR metrics on test")
safe_print("- \\input{output_data/tables/tab_test_metrics_rf.tex} % RF metrics on test")

safe_print("\nCHAPTER 4 (Figures):")
safe_print("- \\includegraphics[width=\\linewidth]{output_data/figures/fig_confusion_lr.pdf} % confusion matrix LR")
safe_print("- \\includegraphics[width=\\linewidth]{output_data/figures/fig_confusion_rf.pdf} % confusion matrix RF")
safe_print("- \\includegraphics[width=\\linewidth]{output_data/figures/fig_roc_lr.pdf} % ROC LR")
safe_print("- \\includegraphics[width=\\linewidth]{output_data/figures/fig_roc_rf.pdf} % ROC RF")
safe_print("- \\includegraphics[width=\\linewidth]{output_data/figures/fig_pr_lr.pdf} % PR LR")
safe_print("- \\includegraphics[width=\\linewidth]{output_data/figures/fig_pr_rf.pdf} % PR RF")
safe_print("\nLog file:")
safe_print(f"- {os.path.join(OUT_LOG, 'run_summary.json')}")
```