

ĐẠI HỌC QUỐC GIA TP. HCM
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA HỆ THỐNG THÔNG TIN



BÁO CÁO ĐỒ ÁN
AN TOÀN VÀ BẢO MẬT HỆ THỐNG THÔNG TIN
ĐỀ TÀI:

INTRUSION DETECTION SYSTEM

Giảng viên: ThS. Hà Lê Hoài Trung

Lớp: IS335.P11.HTCL

Sinh viên thực hiện: Nguyễn Dương Chí Tâm - 21520439

Đỗ Hiền Thảo – 21520460

Huỳnh Mạnh Huy – 21520259

Nguyễn Trương Đình Giang – 21520215

Tp. Hồ Chí Minh, tháng 12 năm 2024

LỜI CẢM ƠN

Lời đầu tiên, chúng em xin chân thành cảm ơn đến toàn thể giảng viên trường Đại học Công nghệ Thông tin - Đại học Quốc gia TP.HCM cũng như nhà trường đã cùng tri thức, tâm huyết đã truyền đạt cho chúng em trong suốt thời gian thực hiện đồ án môn học.

Đặc biệt hơn hết, nhóm chúng em xin gửi lời cảm ơn chân thành đến thầy Hà Lê Hoài Trung – Giảng viên môn An toàn và bảo mật hệ thống thông tin đã trực tiếp giảng dạy, truyền đạt kiến thức, kinh nghiệm, hướng dẫn chúng em một cách cụ thể, tận tâm giúp chúng em hoàn thành tốt đồ án môn học của mình.

Ngoài ra, chúng em cũng gửi lời cảm ơn tập thể lớp IS335.P11.HTCL nói chung cũng như bạn thành viên trong nhóm nói riêng đã cùng nhau học tập, cùng nhau nghiên cứu, học hỏi để thực hiện đồ án một cách tốt nhất.

Cuối cùng, nhóm chúng em đã hoàn thành đồ án mang tên “Instrution Detection System”. Mặc dù đã vận dụng tối đa những gì đã học nhưng vẫn khó tránh khỏi sai sót, vì vậy nhóm chúng em rất mong nhận được sự góp ý từ cô để tiếp tục hoàn thiện dự án một cách tốt nhất. Qua đó, rút ra bài học và tích lũy thêm kinh nghiệm làm hành trang cho tương lai.

Chúng em xin hết và xin chân thành cảm ơn thầy và các bạn!

Trân trọng cảm ơn.

TP. Hồ Chí Minh, tháng 12 năm 2024

[illegible]

MỤC LỤC

LỜI CẢM ƠN.....	2
NHẬN XÉT CỦA GIẢNG VIÊN.....	3
DANH MỤC HÌNH ẢNH.....	7
CHƯƠNG I: GIỚI THIỆU ĐỀ TÀI.....	9
1.1 Bối cảnh đề tài.....	9
1.2 Lý do chọn đề tài.....	10
1.3 Mục tiêu chính	11
1.4 Lý do chọn dataset từ Kaggle	11
CHƯƠNG II: CƠ SỞ LÝ THUYẾT VỀ HỆ THỐNG PHÁT HIỆN XÂM NHẬP DỰA TRÊN HỌC MÁY	11
2.1 Intrusion Detection System (IDS) là gì?	11
<i>Phân loại IDS:</i>	12
2.1.1 Hệ thống phát hiện dựa trên chữ ký (<i>Signature-based Detection</i>).....	12
2.1.2 Hệ thống phát hiện bất thường (<i>Anomaly-based Detection</i>)	12
2.1.3 Hệ thống lai (<i>Hybrid IDS</i>)	12
2.2 Các loại tấn công phổ biến mà IDS phải đối mặt	12
2.2.1 Tấn công từ chối dịch vụ (<i>DoS/DDoS</i>)	12
2.2.2 Tấn công vào ứng dụng web.....	13
2.2.3 Tấn công brute force	13
2.2.4 Tấn công nội bộ (<i>Insider Threats</i>)	13
2.3 Vai trò của học máy trong Intrusion Detection System	14
2.3.1 Phân tích và học từ dữ liệu lớn	14
2.3.2 Phát hiện các mối đe dọa mới.....	14
2.3.3 Giảm cảnh báo sai (<i>False Positives</i>)	14
2.3.4 Tự động hóa và cải thiện hiệu suất.....	14
2.4 Các kỹ thuật học máy phổ biến trong IDS	14
2.4.1 Học có giám sát (<i>Supervised Learning</i>)	14

2.4.2 Học không giám sát (<i>Unsupervised Learning</i>)	15
2.4.3 Học sâu (<i>Deep Learning</i>)	15
2.4.4 Học tăng cường (<i>Reinforcement Learning</i>)	15
2.5 Một số mô hình IDS dựa trên học máy	15
2.5.1 Hệ thống phát hiện bất thường sử dụng <i>Deep Learning</i>	15
2.5.2 Phát hiện tấn công bằng <i>SVM</i>	16
2.5.3 <i>Gradient Boosting</i> cho IDS	16
2.5.4 Học không giám sát với <i>Clustering</i>	16
2.6 Thách thức và giải pháp	16
2.6.1 Thách thức	16
2.6.2 Giải pháp	16
CHƯƠNG III: DATASET	17
3.1. Giải pháp	17
3.3.1 Thông tin dataset	17
3.3.2 Mô tả thuộc tính	17
CHƯƠNG IV: MÔ HÌNH	21
4.1. Phân loại mô hình	21
4.1.1 <i>Decision Tree</i>	21
4.1.2 <i>Random Forest</i>	21
4.1.3 <i>Logistic Regression</i>	22
4.1.4 <i>Naïve Bayes</i>	23
4.2. Mô tả về kiến trúc	24
4.2.1. <i>Decision Tree</i>	24
4.2.2. <i>Random Forest</i>	24
4.2.3. <i>Logistic Regression</i>	25
4.2.4. <i>Naïve Bayes</i>	25
4.3. Lý do chọn lựa	26

4.3.1. <i>Decision Tree</i>	26
4.3.2. <i>Random Forest</i>	26
4.3.3. <i>Logistic Regression</i>	27
4.3.4. <i>Naïve Bayes</i>	27
CHƯƠNG V: XÂY DỰNG VÀ ĐÁNH GIÁ MÔ HÌNH	28
5.1. <i>Tiền xử lý dữ liệu</i>	28
5.1.1. <i>Upload và đọc dữ liệu</i>	28
5.1.2. <i>Thực hiện kiểm tra dữ liệu</i>	29
5.1.3. <i>Làm sạch dữ liệu</i>	31
5.1.4. <i>Phân loại dữ liệu</i>	32
5.1.5. <i>Loại bỏ đặc trưng phụ thuộc</i>	34
5.2. <i>Áp dụng các mô hình thuật toán cho dataset</i>	36
5.2.1. <i>Naïve Bayes</i>	36
5.2.2. <i>Decision Tree</i>	40
5.2.3. <i>Random Forest</i>	45
5.2.4. <i>Logistic Regression</i>	48
5.3. <i>So sánh độ chính xác của các thuật toán và kết quả</i>	52

DANH MỤC HÌNH ẢNH

Hình 1 Thực hiện import thư viện và upload dataset.....	28
Hình 2 Thực hiện đọc dữ liệu từ dataset.....	28
Hình 3 Thực hiện thêm tên cột cho dataset.....	29
Hình 4 Thực hiện sử dụng hàm df.info()	30
Hình 5 Thực hiện thống kê định lượng.....	31
Hình 6 Thực hiện thống kê định tính	31
Hình 7 Thực hiện kiểm tra dữ liệu NULL	32
Hình 8 Thực hiện kiểm tra trùng lặp.....	32
Hình 9 Thực hiện phân loại cột attack	33
Hình 10 Thực hiện thêm cột attack type	33
Hình 11 Thực hiện xem các cột để phân loại.....	33
Hình 12 Thực hiện chuyển đổi giá trị cột protocol_type.....	34
Hình 13 Thực hiện chuyển đổi giá trị flag feature mapping.....	34
Hình 14 Thực hiện drop cột service.....	34
Hình 15 Thực hiện xem lại dataset	34
Hình 16 Bản đồ heatmap của dữ liệu.....	35
Hình 17 Thực hiện xóa các cột có độ tương quan cao.....	35
Hình 18 Kết quả dữ liệu sau khi tiền xử lý	36
Hình 19 Thực hiện chạy mô hình Naïve Bayes	37
Hình 20 Thực hiện đánh giá mô hình	37
Hình 21 Thực hiện hiển thị kết quả grid search dưới dạng DataFrame.....	38
Hình 22 Thực hiện hiển thị kết quả grid search dưới dạng HeatMap.....	39
Hình 23 Thực hiện hiển thị kết quả grid search dưới dạng cột.....	40
Hình 24 Thực hiện chạy mô hình Decision tree	41
Hình 25 Thực hiện đánh giá mô hình	41
Hình 26 Thực hiện hiển thị kết quả grid search dưới dạng DataFrame.....	42
Hình 27 Thực hiện hiển thị kết quả grid search dưới dạng HeatMap.....	43
Hình 28 Thực hiện hiển thị kết quả grid search dưới dạng cột.....	44
Hình 29 Thực hiện hiển thị kết quả của cây quyết định	44
Hình 30 Thực hiện chạy mô hình Random Forest.....	45
Hình 31 Thực hiện đánh giá mô hình	46
Hình 32 Thực hiện hiển thị kết quả grid search dưới dạng DataFrame.....	46
Hình 33 Thực hiện hiển thị kết quả grid search dưới dạng HeatMap.....	47

Hình 34 Thực hiện hiển thị kết quả grid search dưới dạng cột.....	48
Hình 35 Thực hiện chạy mô hình Logistic regression.....	49
Hình 36 Thực hiện đánh giá mô hình	49
Hình 37 Thực hiện hiển thị kết quả grid search dưới dạng DataFrame.....	50
Hình 38 Thực hiện hiển thị kết quả grid search dưới dạng HeatMap.....	51
Hình 39 Thực hiện hiển thị kết quả grid search dưới dạng cột.....	52
Hình 40 Thực hiện so sánh độ chính xác các thuật toán.....	53

CHƯƠNG I: GIỚI THIỆU ĐỀ TÀI

1.1 Bối cảnh đề tài

Trong thế giới kết nối ngày nay, thông tin trở thành tài sản cốt lõi cho các tổ chức và cá nhân. Tuy nhiên, điều này cũng dẫn đến nguy cơ lớn từ các cuộc tấn công mạng, rò rỉ dữ liệu, và vi phạm bảo mật. Bảo vệ hệ thống thông tin trở thành một ưu tiên hàng đầu, và hệ thống phát hiện xâm nhập (Intrusion Detection System - IDS) được xem như một công cụ không thể thiếu.

Theo báo cáo từ **IBM Security X-Force năm 2023**, trung bình mỗi cuộc tấn công mạng thành công gây tổn thất khoảng **4,45 triệu USD** cho doanh nghiệp. Khoảng **39%** các tổ chức được khảo sát thừa nhận rằng hệ thống phát hiện sớm như IDS đã giúp giảm thiểu đến **30%** tổn thất từ các cuộc tấn công mạng.

Ví dụ thực tế:

a. Host-based IDS (HIDS): HIDS thường được triển khai trên máy chủ hoặc thiết bị cụ thể, phát hiện các hành vi không mong muốn như truy cập trái phép vào tệp tin quan trọng hoặc thay đổi không được cấp phép.

- **Ví dụ thực tế:**

Một ngân hàng triển khai HIDS trên máy chủ cơ sở dữ liệu. Khi hệ thống phát hiện một nhân viên nội bộ cố gắng sao chép toàn bộ dữ liệu khách hàng, HIDS cảnh báo ngay lập tức, giúp ngân hàng ngăn chặn rò rỉ thông tin nhạy cảm.

b. Network-based IDS (NIDS): NIDS giám sát toàn bộ lưu lượng mạng, phân tích các gói tin để phát hiện dấu hiệu tấn công mạng như DDoS, khai thác lỗ hổng, hoặc truyền dữ liệu bất hợp pháp.

- **Ví dụ thực tế:**

Một công ty thương mại điện tử sử dụng NIDS để phát hiện một lượng lớn yêu cầu bất thường từ nhiều địa chỉ IP. IDS nhanh chóng nhận diện đây là một cuộc tấn công từ chối dịch vụ (DDoS) và đưa ra cảnh báo, giúp đội ngũ IT kích hoạt biện pháp bảo vệ trước khi hệ thống bị sập.

c. Hybrid IDS: Hệ thống lai kết hợp giữa HIDS và NIDS, cung cấp khả năng giám sát đa chiều, hiệu quả cho các môi trường phức tạp.

- **Ví dụ thực tế:**

Trong môi trường điện toán đám mây của một doanh nghiệp lớn, Hybrid IDS giám sát cả lưu lượng mạng lẫn hoạt động trên các máy ảo. Điều này giúp phát hiện một máy ảo bị chiếm quyền và chặn đứng hành động xâm nhập từ tin tặc.

Hiệu quả của IDS: Số liệu và minh chứng thực tiễn

- **Tăng khả năng phát hiện tấn công:** Theo một nghiên cứu của Gartner (2022), việc triển khai IDS giúp các tổ chức phát hiện 95% các cuộc tấn công dựa trên chữ ký (signature-based attacks) và 75% các tấn công bất thường (anomaly-based attacks).
- **Giảm thời gian phát hiện mối đe dọa:** Palo Alto Networks báo cáo rằng hệ thống IDS hiện đại giúp giảm thời gian phát hiện tấn công xuống còn 4 giờ, so với trung bình 280 ngày nếu không sử dụng IDS (theo báo cáo của IBM).
- **Hỗ trợ phân tích sau sự cố:** Trong các cuộc tấn công lớn, IDS cung cấp nhật ký và dữ liệu truy vết giúp đội ngũ điều tra xác định nguồn gốc tấn công nhanh chóng.

Các nghiên cứu này cho thấy Hệ thống phát hiện xâm nhập (IDS) là công cụ bảo mật thiết yếu nhằm giúp các tổ chức bảo vệ dữ liệu và giảm thiểu tổn thất về mặt tài chính.

1.2 Lý do chọn đề tài

Với sự phát triển của chuyển đổi số, dữ liệu và hệ thống thông tin trở thành tài sản quan trọng của các tổ chức, nhưng cũng đồng thời là mục tiêu chính của tin tặc. Thông tin của mỗi cá nhân, tổ chức, doanh nghiệp, ... đều là những tài sản quý giá cần được bảo mật. Điều này không chỉ để đảm bảo được sự riêng tư của người dùng mà còn là việc cần thiết để tránh đi những rủi ro có thể xảy ra khi bị rò rỉ thông tin. Tuy nhiên, cũng bởi vì sự quan trọng và giá trị của thông tin, việc các tin tặc sử dụng thông tin hay cướp thông tin nhằm chiếm đoạt lợi ích từ các doanh nghiệp, tổ chức ngày càng trở nên tinh vi. Tổ chức không chỉ cần ngăn chặn các tấn công mà còn phải phát hiện và phản ứng nhanh chóng, giảm thiểu thiệt hại.

IDS (Intrusion Detection System) đóng vai trò quan trọng trong chiến lược này. IDS không chỉ phát hiện các cuộc tấn công phổ biến mà còn giám sát và nhận diện các hành vi bất thường, giúp phát hiện tấn công chưa từng được biết đến.

Nghiên cứu IDS mang lại cơ hội lớn để cải thiện hiệu quả của hệ thống bảo mật, từ đó giải quyết các vấn đề thực tiễn và đáp ứng nhu cầu cấp thiết của doanh nghiệp và tổ chức.

1.3 Mục tiêu chính

Mục tiêu chính: Xây dựng mô hình máy học để phân biệt giữa 'kết nối xấu' (xâm nhập/tấn công) và 'kết nối tốt (bình thường)'.

- Nghiên cứu cơ sở lý thuyết về IDS, phân tích các kỹ thuật phòng chống tấn công.
- Tìm hiểu về các mô hình phân loại.
- Nắm vững về cách phân loại giữa các dạng kết nối nhằm giúp cải thiện khả năng bảo vệ cá nhân và tổ chức khỏi tấn công mạng.
- Xây dựng mô hình phân loại.

1.4 Lý do chọn dataset từ Kaggle

Nhóm quyết định chọn dataset từ Kaggle, một nền tảng trực tuyến nổi tiếng trong cộng đồng Machine Learning và Khoa học Dữ liệu.

Kaggle là một thư viện tập dữ liệu trực tuyến lớn, cho phép người dùng chia sẻ, tìm kiếm, và tương tác với các bộ dữ liệu đa dạng từ mọi lĩnh vực.

Với tài nguyên dữ liệu khổng lồ, Kaggle không chỉ là một nguồn tài nguyên miễn phí giúp người dùng tiết kiệm rất nhiều thời gian và công sức trong lúc tìm kiếm mà còn là một cộng đồng nơi người dùng có thể tìm hiểu, xây dựng mô hình, và tương tác với những chuyên gia hàng đầu trong lĩnh vực Machine Learning trên toàn thế giới. Dữ liệu trên Kaggle thường được đánh giá cao về chất lượng và đã được xử lý một cách kỹ lưỡng.

CHƯƠNG II: CƠ SỞ LÝ THUYẾT VỀ HỆ THỐNG PHÁT HIỆN XÂM NHẬP DỰA TRÊN HỌC MÁY

2.1 Intrusion Detection System (IDS) là gì?

Hệ thống phát hiện xâm nhập (Intrusion Detection System - IDS) là một thành phần quan trọng trong bảo mật mạng và hệ thống thông tin. IDS giúp phát hiện và ứng phó với các mối đe dọa bằng cách giám sát lưu lượng mạng, hành vi của người

dùng hoặc các thay đổi trong hệ thống để phát hiện các hành vi bất thường hoặc trái phép.

Phân loại IDS:

2.1.1 Hệ thống phát hiện dựa trên chữ ký (Signature-based Detection)

- Sử dụng cơ sở dữ liệu về các mẫu tấn công (signature) để phát hiện các mối đe dọa đã biết.
- **Ưu điểm:** Nhanh, chính xác với các tấn công đã biết.
- **Nhược điểm:** Không phát hiện được các tấn công mới hoặc chưa từng thấy.

2.1.2 Hệ thống phát hiện bất thường (Anomaly-based Detection)

- So sánh hành vi hiện tại với một đường cơ sở (baseline) để phát hiện hành vi bất thường.
- **Ưu điểm:** Phát hiện được các tấn công mới.
- **Nhược điểm:** Tỷ lệ cảnh báo sai cao, phụ thuộc nhiều vào dữ liệu huấn luyện.

2.1.3 Hệ thống lai (Hybrid IDS)

- Kết hợp cả hai phương pháp trên để tận dụng ưu điểm của chúng.

2.2 Các loại tấn công phổ biến mà IDS phải đối mặt

2.2.1 Tấn công từ chối dịch vụ (DoS/DDoS)

- **Mục tiêu:** Làm cạn kiệt tài nguyên của hệ thống, khiến hệ thống không thể phục vụ người dùng hợp lệ.
- **Hậu quả:**
 - Tạo lưu lượng giả mạo lớn nhằm chiếm băng thông mạng.
 - Tấn công làm cạn kiệt CPU, bộ nhớ hoặc tài nguyên khác của máy chủ.
- **Ví dụ thực tế:**
 - **Tấn công DDoS trên Dyn (2016):** Một cuộc tấn công phân tán từ chối dịch vụ (DDoS) nhắm vào dịch vụ DNS của Dyn, làm gián đoạn hàng loạt dịch vụ trực tuyến, bao gồm Twitter, Netflix, và Spotify. Các hacker

sử dụng hàng triệu thiết bị IoT bị xâm nhập để thực hiện cuộc tấn công này.

- **Tấn công DNS Amplification (2018):** Lợi dụng sự yếu kém trong cấu hình DNS để tạo ra các yêu cầu lớn và gửi tới máy chủ, gây quá tải bằng thông.

2.2.2 Tấn công vào ứng dụng web

- **Mục tiêu:** Khai thác lỗ hổng trong ứng dụng web như SQL Injection, Cross-Site Scripting (XSS), hoặc Remote Code Execution.
- **Hậu quả:**
 - Chiếm quyền kiểm soát hệ thống.
 - Đánh cắp thông tin nhạy cảm của người dùng.
- **Ví dụ thực tế:**
 - **SQL Injection (2014):** Một cuộc tấn công vào hệ thống của Yahoo, cho phép hacker lấy được thông tin đăng nhập và dữ liệu cá nhân của hàng triệu người dùng.
 - **XSS (2017):** Một tấn công XSS vào ngân hàng trực tuyến khiến kẻ tấn công có thể đánh cắp thông tin thẻ tín dụng từ người dùng khi họ đăng nhập vào trang web bị tấn công.

2.2.3 Tấn công brute force

- **Mục tiêu:** Đột nhập hệ thống bằng cách thử tất cả các tổ hợp mật khẩu hoặc khóa mã hóa.
- **Hậu quả:**
 - Lấy cắp thông tin đăng nhập.
 - Phá hoại hệ thống hoặc mã hóa dữ liệu (ransomware).
- **Ví dụ thực tế:**
 - **Tấn công vào Facebook (2018):** Một hacker đã thực hiện tấn công brute force để đoán mật khẩu tài khoản người dùng Facebook, chiếm đoạt hàng nghìn tài khoản thông qua việc thử mật khẩu đơn giản.

2.2.4 Tấn công nội bộ (Insider Threats)

- **Mục tiêu:** Tấn công từ bên trong, thường do nhân viên hoặc người có quyền truy cập gây ra.
- **Hậu quả:**

- Phá hoại dữ liệu, làm gián đoạn dịch vụ.
- Rò rỉ thông tin nhạy cảm ra bên ngoài.
- **Ví dụ thực tế:**
 - **Edward Snowden (2013):** Cựu nhân viên của NSA đã tiết lộ các tài liệu bí mật về các chương trình giám sát của chính phủ Mỹ, gây ảnh hưởng lớn đến an ninh quốc gia.
 - **Tấn công vào Target (2013):** Kẻ tấn công xâm nhập hệ thống của Target qua một nhà cung cấp bên ngoài, lợi dụng quyền truy cập của nhân viên để xâm nhập vào hệ thống và đánh cắp thông tin thẻ tín dụng của 40 triệu khách hàng.

2.3 Vai trò của học máy trong Intrusion Detection System

Học máy đã thay đổi cách thức hoạt động của IDS, mang lại các lợi ích sau:

2.3.1 Phân tích và học từ dữ liệu lớn

- Xử lý khối lượng lớn dữ liệu mạng trong thời gian thực, phân tích để tìm ra các mẫu hành vi bất thường.

2.3.2 Phát hiện các mối đe dọa mới

- Không như các phương pháp truyền thống, học máy có khả năng tự học các mẫu tấn công mới mà không cần có sẵn chữ ký.

2.3.3 Giảm cảnh báo sai (False Positives)

- Các mô hình học máy giúp giảm thiểu việc đánh dấu các hành vi hợp lệ là mối đe dọa.

2.3.4 Tự động hóa và cải thiện hiệu suất

- IDS dựa trên học máy có khả năng tự động hóa các quy trình phát hiện, giảm sự phụ thuộc vào con người.

2.4 Các kỹ thuật học máy phổ biến trong IDS

2.4.1 Học có giám sát (Supervised Learning)

- **Mô hình:** Dựa trên tập dữ liệu đã được gắn nhãn (bao gồm các mẫu "bình thường" và "bị tấn công") để huấn luyện.
- **Thuật toán phổ biến:**

- **Decision Trees và Random Forests:** Tạo cây quyết định để phân loại.
- **Support Vector Machines (SVM):** Tìm siêu phẳng phân chia dữ liệu giữa các lớp.
- **Naive Bayes:** Mô hình xác suất để dự đoán lớp của mẫu dữ liệu.
- **Logistic Regression (LR):** Mô hình phân loại tuyến tính có thể dự đoán xác suất của một sự kiện xảy ra. Trong IDS, LR có thể được sử dụng để phân loại lưu lượng mạng thành "bình thường" hoặc "xâm nhập" dựa trên các đặc trưng của lưu lượng đó.

2.4.2 Học không giám sát (Unsupervised Learning)

- **Mô hình:** Không cần dữ liệu gắn nhãn, thường dùng để phát hiện bất thường.
- **Thuật toán phổ biến:**
 - **Clustering (k-means, DBSCAN):** Nhóm các mẫu tương tự nhau và xác định các mẫu bất thường.
 - **Principal Component Analysis (PCA):** Giảm chiều dữ liệu để phát hiện các hành vi không điển hình.

2.4.3 Học sâu (Deep Learning)

- **Mô hình:** Áp dụng mạng nơ-ron nhân tạo để học các đặc trưng phức tạp từ dữ liệu.
- **Ứng dụng:**
 - **Recurrent Neural Networks (RNN):** Phân tích chuỗi thời gian trong lưu lượng mạng.
 - **Autoencoders:** Học các biểu diễn đặc trưng và phát hiện bất thường.

2.4.4 Học tăng cường (Reinforcement Learning)

- **Mô hình:** Dựa trên việc thử nghiệm và nhận phản hồi từ môi trường để học cách phát hiện và ngăn chặn xâm nhập.

2.5 Một số mô hình IDS dựa trên học máy

2.5.1 Hệ thống phát hiện bất thường sử dụng Deep Learning

- **Mô hình:** Dựa trên mạng LSTM để phân tích lưu lượng mạng theo thời gian.
- **Kết quả:** Phát hiện các tấn công DoS/DDoS với độ chính xác cao.

2.5.2 Phát hiện tấn công bằng SVM

- **Ứng dụng:** Phân loại các tấn công mạng thành nhiều loại (DoS, brute force, insider threats).
- **Đánh giá:** Hiệu quả cao nhưng tốn tài nguyên khi xử lý dữ liệu lớn.

2.5.3 Gradient Boosting cho IDS

- **Ứng dụng:** Phân tích dữ liệu gắn nhãn để phát hiện tấn công với độ chính xác cao.
- **Đánh giá:** Cần nhiều thời gian huấn luyện nhưng hiệu quả dự đoán vượt trội.

2.5.4 Học không giám sát với Clustering

- **Ứng dụng:** Phát hiện bất thường trong lưu lượng mạng không gắn nhãn.
- **Đánh giá:** Dễ triển khai nhưng khó tối ưu khi dữ liệu phức tạp.

2.6 Thách thức và giải pháp

2.6.1 Thách thức

- **Dữ liệu mất cân bằng:** Các mẫu tấn công thường ít hơn nhiều so với lưu lượng hợp lệ.
- **Thay đổi hành vi tấn công:** Tin tặc liên tục thay đổi kỹ thuật để vượt qua IDS.
- **Chi phí tính toán cao:** Học máy và học sâu đòi hỏi tài nguyên lớn để xử lý dữ liệu.

2.6.2 Giải pháp

- **Kỹ thuật xử lý dữ liệu mất cân bằng:** Sử dụng SMOTE, oversampling hoặc tập trung vào các mẫu xâm nhập hiếm.
- **Huấn luyện định kỳ:** Cập nhật mô hình với dữ liệu mới thường xuyên.
- **Sử dụng mô hình lai:** Kết hợp các thuật toán đơn giản với học sâu để cân bằng giữa hiệu quả và tốc độ.

CHƯƠNG III: DATASET

3.1. Giải pháp

3.3.1 Thông tin dataset

- **Tên dataset:** NSL-KDD
- **Thông tin chung dataset :** NSL-KDD là một tập dữ liệu tiêu chuẩn được thiết kế để đánh giá hiệu suất của các hệ thống phát hiện xâm nhập (IDS). Đây là phiên bản cải tiến của tập dữ liệu KDD CUP 1999, được khắc phục các nhược điểm lớn trong phiên bản gốc như dữ liệu trùng lặp và mất cân đối. NSL-KDD thường được sử dụng để thử nghiệm các thuật toán Machine Learning và Deep Learning trong lĩnh vực an ninh mạng.
- **Nguồn dataset:** <https://www.kaggle.com/code/eneskosar19/intrusion-detection-system-nsL-kdd/input>
- **Kích thước dataset:** Có tổng cộng 125972 hàng dữ liệu và 43 cột thuộc tính.

3.3.2 Mô tả thuộc tính

STT	Tên thuộc tính	Ý nghĩa	Loại dữ liệu
1	duration	Thời gian kết nối (tính bằng giây).	Numeric
2	protocol_type	Loại giao thức được sử dụng (tcp, udp, icmp).	Categorical
3	service	Dịch vụ mạng mà kết nối yêu cầu (http, ftp, smtp, ...)	Categorical
4	flag	Trạng thái của kết nối (SF, REJ, RSTO, ...)	Categorical
5	src_bytes	Số byte được gửi từ nguồn đến đích trong kết nối.	Numeric
6	dst_bytes	Số byte được gửi từ đích về nguồn trong kết nối.	Numeric
7	land	Chỉ báo xem kết nối có phải là "land attack" (1 nếu có, 0 nếu không). Land attack là khi địa chỉ IP nguồn và đích giống nhau.	Binary

8	wrong_fragment	Số lượng các mảnh (fragment) không hợp lệ trong kết nối.	Numeric
9	urgent	Số lượng gói tin có gắn cờ "urgent" (ưu tiên).	Numeric
10	hot	Số lượng các hành động "hot" (đáng ngờ), ví dụ như gọi hệ thống (system calls).	Numeric
11	num_failed_logins	Số lần đăng nhập không thành công.	Numeric
12	logged_in	Cờ đánh dấu xem kết nối đã đăng nhập thành công hay chưa (1: đã đăng nhập, 0: chưa).	Binary
13	num_compromised	Số lượng các phiên (session) bị xâm phạm.	Numeric
14	root_shell	Cờ cho biết kết nối đã mở shell với quyền root hay chưa (1: có, 0: không).	Binary
15	su_attempted	Số lần cố gắng thực hiện lệnh su (leo thang đặc quyền).	Numeric
16	num_root	Số lượng hành động được thực hiện với quyền root.	Numeric
17	num_file_creations	Số lượng tệp được tạo trong kết nối.	Numeric
18	num_shells	Số lần khởi tạo shell trong kết nối.	Numeric
19	num_access_files	Số lượng tệp bị truy cập trong kết nối.	Numeric
20	num_outbound_cmds	Số lệnh outbound được gửi (thường là 0 cho các kết nối không FTP).	Numeric
21	is_host_login	Cờ cho biết có phải đăng nhập host hay không (thường là 0 vì không có trường hợp nào được ghi nhận).	Binary

22	is_guest_login	Cờ cho biết có phải đăng nhập với tài khoản khách (guest) hay không (1: có, 0: không).	Binary
23	count	Số lượng kết nối tới cùng một host trong khoảng thời gian 2 giây.	Numeric
24	srv_count	Số lượng kết nối tới cùng một dịch vụ (service) trong khoảng thời gian 2 giây.	Numeric
25	serror_rate	Tỷ lệ kết nối có lỗi cờ SYN (SYN errors) trong số các kết nối đến cùng một host.	Numeric
26	srv_serror_rate	Tỷ lệ kết nối có lỗi cờ SYN tới cùng một dịch vụ (service).	Numeric
27	rerror_rate	Tỷ lệ kết nối có lỗi RESET (REJ errors) trong số các kết nối đến cùng một host.	Numeric
28	srv_rerror_rate	Tỷ lệ kết nối có lỗi RESET tới cùng một dịch vụ.	Numeric
29	same_srv_rate	Tỷ lệ kết nối đến cùng một dịch vụ (service).	Numeric
30	diff_srv_rate	Tỷ lệ kết nối tới các dịch vụ khác nhau.	Numeric
31	srv_diff_host_rate	Tỷ lệ kết nối tới các host khác nhau trên cùng một dịch vụ.	Numeric
32	dst_host_count	Số lượng kết nối tới cùng một host đích (destination host) trong khoảng thời gian nhất định.	Numeric

33	dst_host_srv_count	Số lượng kết nối tới cùng một dịch vụ trên cùng một host đích.	Numeric
34	dst_host_same_srv_rate	Tỷ lệ kết nối tới cùng một dịch vụ trên cùng một host đích.	Numeric
35	dst_host_diff_srv_rate	Tỷ lệ kết nối tới các dịch vụ khác nhau trên cùng một host đích.	Numeric
36	dst_host_same_src_port_rate	Tỷ lệ kết nối tới cùng một cổng nguồn (source port) trên cùng một host đích.	Numeric
37	dst_host_srv_diff_host_rate	Tỷ lệ kết nối tới các host khác nhau trên cùng một dịch vụ ở một host đích.	Numeric
38	dst_host_serror_rate	Tỷ lệ kết nối có lỗi SYN trên cùng một host đích.	Numeric
39	dst_host_srv_serror_rate	Tỷ lệ kết nối có lỗi SYN trên cùng một dịch vụ ở một host đích.	Numeric
40	dst_host_rerror_rate	Tỷ lệ kết nối có lỗi RESET trên cùng một host đích.	Numeric
41	dst_host_srv_rerror_rate	Tỷ lệ kết nối có lỗi RESET trên cùng một dịch vụ ở một host đích.	Numeric
42	attack	Loại tấn công hoặc trạng thái "normal". (neptune, smurf, ...)	Categorical
43	level	Chỉ số mức độ nghiêm trọng hoặc ưu tiên của kết nối	Numeric

CHƯƠNG IV: MÔ HÌNH

4.1. Phân loại mô hình

4.1.1 Decision Tree

Cây quyết định là một loại thuật toán học có giám sát thường được sử dụng trong học máy để mô hình hóa và dự đoán kết quả dựa trên dữ liệu đầu vào. Đây là một cấu trúc giống như cây, trong đó mỗi nút bên trong kiểm tra thuộc tính, mỗi nhánh tương ứng với giá trị thuộc tính và mỗi nút lá biểu diễn quyết định hoặc dự đoán cuối cùng. Thuật toán cây quyết định thuộc loại học có giám sát. Chúng có thể được sử dụng để giải quyết cả các vấn đề hồi quy và phân loại.

Cây quyết định hoạt động như một biểu đồ cây, trong đó:

- **Mỗi nút trong cây** đại diện cho một thuộc tính (hoặc đặc trưng) được dùng để phân loại.
- **Các nhánh** đại diện cho các điều kiện hoặc quy tắc dựa trên giá trị của thuộc tính.
- **Lá cây** (leaf nodes) đại diện cho kết quả hoặc giá trị dự đoán (nhóm hoặc lớp).

Quá trình học của cây quyết định liên quan đến việc chia tập dữ liệu thành các tập con dựa trên tiêu chí phân chia (như *Information Gain*, *Gini Index*, hoặc *Entropy*) để tối ưu hóa việc phân chia.

4.1.2 Random Forest

Random Forest là một thuật toán học máy thuộc nhóm **ensemble learning** (học kết hợp), được sử dụng cho cả bài toán phân loại và hồi quy. Thuật toán này được phát triển dựa trên ý tưởng kết hợp nhiều cây quyết định (decision trees) để cải thiện độ chính xác và độ ổn định của mô hình.

Ưu điểm:

- **Độ chính xác cao:** Kết hợp nhiều cây giúp giảm sai số do quá khớp hoặc chưa khớp (bias-variance tradeoff).
- **Tính đa dạng:** Các cây được xây dựng từ dữ liệu và thuộc tính ngẫu nhiên nên ít tương quan, tăng tính tổng quát.
- **Khả năng xử lý dữ liệu phức tạp:** Hiệu quả với dữ liệu có nhiều thuộc tính hoặc mối quan hệ phi tuyến tính.

- **Chống overfitting:** Random Forest giảm thiểu nguy cơ quá khớp nhờ việc chọn ngẫu nhiên cả dữ liệu và thuộc tính.
- **Xác định tầm quan trọng của thuộc tính:** Thuật toán cung cấp thông tin về mức độ ảnh hưởng của từng thuộc tính đến kết quả.

Nhược điểm: Dù có độ chính xác khá cao nhưng cây quyết định tồn tại những hạn chế lớn đó là:

- **Chi phí tính toán cao:** Xây dựng và dự đoán với nhiều cây quyết định đòi hỏi tài nguyên tính toán lớn.
- **Khó diễn giải:** Kết quả từ Random Forest thường khó giải thích hơn so với một cây quyết định đơn lẻ.
- **Hiệu quả giảm với dữ liệu độ nhiễu cao:** Nếu dữ liệu có quá nhiều nhiễu, các cây quyết định có thể học các mẫu không thực sự tồn tại.

4.1.3 Logistic Regression

Logistic Regression là một thuật toán học máy có giám sát, thường được sử dụng cho bài toán phân loại nhị phân. Thuật toán này dựa trên mô hình hồi quy tuyến tính nhưng sử dụng hàm sigmoid để dự đoán xác suất của một lớp cụ thể.

Nguyên lý hoạt động:

- Logistic Regression mô hình hóa mối quan hệ giữa các thuộc tính đầu vào (XXX) và xác suất của một lớp đầu ra (Y) bằng công thức:

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n)}}$$

Trong đó, β_0, β_1, \dots là các hệ số cần tối ưu.

- Hàm sigmoid giới hạn giá trị đầu ra trong khoảng $[0, 1]$ đại diện cho xác suất.
- Ngưỡng phân loại (thường là 0.5) được sử dụng để quyết định liệu một mẫu thuộc lớp 1 hay lớp 0.

Ứng dụng trong IDS:

- Logistic Regression có thể được sử dụng để phân loại các kết nối mạng thành "bình thường" hoặc "tấn công" dựa trên các đặc trưng như thời gian kết nối, giao thức, số byte gửi đi và nhận về.

4.1.4 Naïve Bayes

Naïve Bayes là một nhóm các thuật toán phân loại dựa trên định lý Bayes và giả định rằng các đặc trưng (thuộc tính) của dữ liệu là độc lập với nhau.

Mặc dù giả định này không thực tế trong hầu hết các trường hợp, Naive Bayes vẫn hoạt động hiệu quả trong nhiều tình huống.

Lý thuyết Bayes mô tả xác suất của một sự kiện A xảy ra khi biết sự kiện B đã xảy ra.

Công thức:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Trong đó:

- $P(A|B)$: Xác suất xảy ra sự kiện A khi biết B.
- $P(B|A)$: Xác suất xảy ra sự kiện B khi biết A.
- $P(A)$: Xác suất xảy ra sự kiện A.
- $P(B)$: Xác suất xảy ra sự kiện B.

Ưu điểm:

- **Đơn giản và nhanh chóng:** Tính toán dễ dàng, phù hợp với các bài toán lớn.
- **Hiệu quả trên tập dữ liệu nhỏ:** Dễ dàng huấn luyện ngay cả khi dữ liệu hạn chế.
- **Xác suất rõ ràng:** Đầu ra của mô hình là các xác suất, dễ diễn giải.

Nhược điểm:

- Giả sử các tính năng là độc lập, điều này **không phải lúc nào cũng đúng** trong dữ liệu thực tế.
- **Có thể bị ảnh hưởng bởi những thuộc tính không liên quan.**
- Có thể gán xác suất bằng 0 cho các sự kiện không nhìn thấy được, dẫn đến **khả năng khái quát kém.**

4.2. Mô tả về kiến trúc

4.2.1. Decision Tree

- **Nút gốc (Root Node):**
 - Đây là nút đầu tiên của cây, đại diện cho toàn bộ tập dữ liệu.
 - Tại đây, thuộc tính (hoặc đặc trưng) tốt nhất sẽ được chọn để bắt đầu phân chia dữ liệu.
- **Nút bên trong (Internal Nodes):**
 - Các nút trung gian đại diện cho các điều kiện kiểm tra được sử dụng để tiếp tục phân chia dữ liệu.
 - Mỗi nút sẽ áp dụng một quy tắc dựa trên một đặc trưng cụ thể
 - Kết quả kiểm tra sẽ dẫn đến các nhánh con.
- **Nhánh (Branch):**
 - Một nhánh kết nối hai nút trong cây, thể hiện kết quả của điều kiện kiểm tra tại nút gốc hoặc nút bên trong.
- **Lá (Leaf Nodes):**
 - Đây là các nút cuối cùng của cây, không tiếp tục phân chia dữ liệu nữa.
 - Tại mỗi lá, một giá trị dự đoán hoặc một lớp phân loại sẽ được lưu lại.

4.2.2. Random Forest

Random Forest hoạt động dựa trên hai ý tưởng chính:

1. Bootstrap Aggregation (Bagging): Khi xây dựng mỗi cây quyết định thành viên, một tập dữ liệu con được tạo ra bằng cách lấy mẫu ngẫu nhiên có hoàn lại (with replacement). Mỗi tập dữ liệu con có kích thước bằng với kích thước của tập dữ liệu ban đầu nhưng có sự khác biệt về thành phần các mẫu được sử dụng để huấn luyện một cây quyết định độc lập.
Dự đoán cuối cùng được thực hiện bằng cách:
 - Phân loại: Lấy số đông (majority vote) của tất cả các cây.
 - Hồi quy: Lấy trung bình dự đoán của các cây.
2. Random Feature Selection: Ở mỗi nút của một cây quyết định, Random Forest không xem xét tất cả các thuộc tính (features) mà chỉ chọn một tập con ngẫu nhiên. Điều này giúp giảm sự tương quan giữa các cây và làm tăng tính đa dạng của mô hình.

4.2.3. Logistic Regression

Cấu trúc và hoạt động:

1. Đầu vào:

- Các đặc trưng đầu vào (X_1, X_2, \dots, X_n) được chuẩn hóa hoặc mã hóa phù hợp để đảm bảo tính ổn định khi tính toán.

2. Hàm tuyến tính:

- Tạo ra điểm số tuyến tính (z) từ dữ liệu:

$$z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n$$

3. Hàm sigmoid:

- Chuyển đổi điểm số tuyến tính thành xác suất:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

4. Ngưỡng quyết định:

- Nếu xác suất $P \geq 0.5$, mẫu được dự đoán thuộc lớp "tấn công"; ngược lại, thuộc lớp "bình thường."

Cách áp dụng trong IDS:

- Các đặc trưng như giao thức, trạng thái kết nối, và số lần đăng nhập thất bại được sử dụng để tính toán điểm số tuyến tính.
- Kết quả phân loại giúp phát hiện tấn công dựa trên ngưỡng xác suất đã xác định.

4.2.4. Naïve Bayes

1. Đầu vào: Tập dữ liệu mạng, bao gồm các đặc trưng như địa chỉ IP, giao thức, kích thước gói, thời gian, và nhãn (xâm nhập hoặc bình thường).
2. Xử lý dữ liệu: Chuyển đổi dữ liệu thô thành các đặc trưng có thể tính toán xác suất, xử lý dữ liệu mất cân bằng và chuẩn hóa dữ liệu nếu cần.
3. Mô hình xác suất: Naive Bayes giả định rằng tất cả các đặc trưng là độc lập. Dựa trên định lý Bayes:

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)}$$

Trong đó:

- $P(X|C)$: Xác suất của đặc trưng X khi biết lớp C .
 - C : Lớp (bình thường hoặc xâm nhập).
 - X : Tập các đặc trưng.
 - $P(C)$: Xác suất tiên nghiệm của lớp C .
4. Dự đoán lớp: Mô hình dự đoán lớp có xác suất lớn nhất $P(C|X)$ cho từng gói hoặc kết nối mạng.

4.3. Lý do chọn lựa

4.3.1. *Decision Tree*

Cây quyết định được sử dụng rộng rãi trong học máy vì một số lý do:

- Cây quyết định rất linh hoạt trong việc mô phỏng các quá trình ra quyết định phức tạp vì tính dễ hiểu và linh hoạt của nó.
- Việc miêu tả các kịch bản lựa chọn phức tạp có tính đến nhiều nguyên nhân và kết quả khác nhau được thực hiện thông qua cấu trúc phân cấp của chúng.
- Chúng cung cấp những hiểu biết dễ hiểu về logic quyết định, cây quyết định đặc biệt hữu ích cho các nhiệm vụ liên quan đến phân loại và hồi quy.
- Chúng thành thạo với cả dữ liệu số và dữ liệu phân loại, và có thể dễ dàng thích ứng với nhiều tập dữ liệu khác nhau nhờ khả năng lựa chọn tính năng tự động.
- Cây quyết định cũng cung cấp hình ảnh trực quan đơn giản, giúp hiểu và làm sáng tỏ các quy trình quyết định cơ bản trong một mô hình.

4.3.2. *Random Forest*

- Tính hiệu quả cao trong việc phát hiện xâm nhập nhờ khả năng tổng hợp dự đoán từ nhiều cây quyết định, giúp cải thiện độ chính xác và giảm overfitting.
- Mô hình hoạt động tốt trên dữ liệu phức tạp và mất cân bằng, điển hình trong hệ thống phát hiện xâm nhập.
- Có khả năng xử lý nhanh trên dữ liệu lớn, chống nhiễu hiệu quả, và cung cấp thông tin về tầm quan trọng của các đặc trưng, giúp nâng cao khả năng giải thích.

4.3.3. Logistic Regression

- Đơn giản và hiệu quả:
 - Logistic Regression dễ triển khai và cung cấp kết quả phân loại nhanh chóng với độ chính xác cao trên dữ liệu tuyến tính hoặc gần tuyến tính.
- Khả năng giải thích:
 - Mô hình cung cấp thông tin rõ ràng về tác động của từng đặc trưng đến kết quả dự đoán, giúp dễ dàng hiểu và phân tích.
- Tương thích tốt với IDS:
 - Với các đặc trưng như thời gian kết nối, số byte truyền tải, và tỉ lệ lỗi trong mạng, Logistic Regression hoạt động tốt khi cần phân loại tấn công dựa trên các tiêu chí đã biết.
- Hiệu quả trên dữ liệu nhỏ:
 - Với các tập dữ liệu có kích thước nhỏ hoặc vừa, Logistic Regression vẫn đạt hiệu quả cao mà không đòi hỏi tài nguyên tính toán lớn.

4.3.4. Naïve Bayes

- Tính đơn giản, tốc độ xử lý cao, và khả năng hoạt động tốt trên dữ liệu lớn, đặc biệt phù hợp với hệ thống phát hiện xâm nhập thời gian thực.
- Thuật toán này dựa trên cơ sở xác suất, giúp dự đoán nhanh và hiệu quả ngay cả với các tập dữ liệu có đặc trưng đa chiều hoặc mất cân bằng (số lượng mẫu xâm nhập ít hơn mẫu bình thường).
- Mặc dù giả định các đặc trưng là độc lập (naive), mô hình vẫn đạt hiệu quả cao trong nhiều trường hợp IDS thực tế.

CHƯƠNG V: XÂY DỰNG VÀ ĐÁNH GIÁ MÔ HÌNH

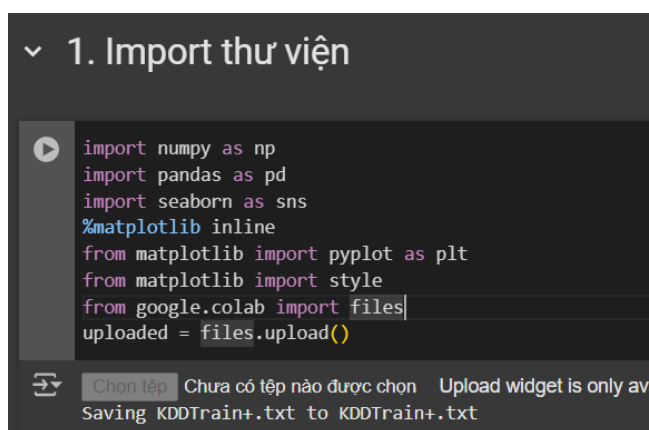
5.1. Tiền xử lý dữ liệu

5.1.1. Upload và đọc dữ liệu

Ta thực hiện việc sử dụng thư viện upload file của Google Colab để up dữ liệu từ máy tính.

```
from google.colab import files
```

```
uploaded = files.upload()
```



Hình 1 Thực hiện import thư viện và upload dataset

Tiếp theo ta thực hiện việc đọc dữ liệu từ dataset

```
df_0 = pd.read_csv("KDDTrain+.txt")
```

```
df = df_0.copy()
```

```
df.head()
```

The screenshot shows the output of the 'df.head()' command in a Google Colab notebook. The output is a table with 5 rows and 43 columns. The columns are labeled with various network-related attributes and a final column labeled 'normal'. The first row shows a normal connection (normal: 20). The second row shows a normal connection (normal: 15). The third row shows a normal connection (normal: 19). The fourth row shows a normal connection (normal: 21). The fifth row shows a normal connection (normal: 21).

	0	tcp	ftp_data	SF	491	0.1	0.2	0.3	0.4	0.5	...	0.17	0.03	0.17.1	0.00.6	0.00.7	0.00.8	0.05	0.00.9	normal	20
0	0	udp	other	SF	146	0	0	0	0	0	...	0.00	0.60	0.88	0.00	0.00	0.00	0.0	0.00	normal	15
1	0	tcp	private	S0	0	0	0	0	0	0	...	0.10	0.05	0.00	0.00	1.00	1.00	0.0	0.00	neptune	19
2	0	tcp	http	SF	232	8153	0	0	0	0	...	1.00	0.00	0.03	0.04	0.03	0.01	0.0	0.01	normal	21
3	0	tcp	http	SF	199	420	0	0	0	0	...	1.00	0.00	0.00	0.00	0.00	0.00	0.0	0.00	normal	21
4	0	tcp	private	REJ	0	0	0	0	0	0	...	0.07	0.07	0.00	0.00	0.00	0.00	1.0	1.00	neptune	21

5 rows x 43 columns

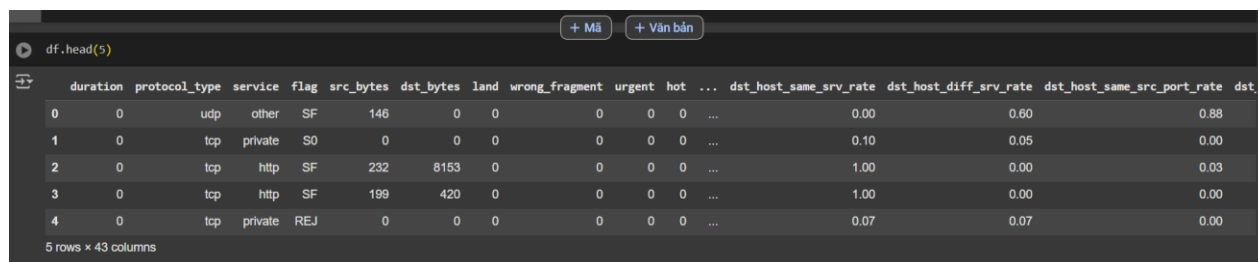
Hình 2 Thực hiện đọc dữ liệu từ dataset

Bởi vì dataset này không có tên cột, nên ta thêm các tên cột tương ứng cho dữ liệu.

Bằng lệnh:

```
columns =
(['duration','protocol_type','service','flag','src_bytes','dst_bytes','land','wrong_fragment',
'urgent','hot','num_failed_logins','logged_in','num_compromised','root_shell','su
_attempted','num_root','num_file_creations','num_shells','num_access_files','num
_outbound_cmds','is_host_login','is_guest_login','count','srv_count','serror_rate','srv
_serror_rate','rerror_rate','srv_rerror_rate','same_srv_rate','diff_srv_rate','srv_diff_h
ost_rate','dst_host_count','dst_host_srv_count','dst_host_same_srv_rate','dst_host_d
iff_srv_rate','dst_host_same_src_port_rate','dst_host_srv_diff_host_rate','dst_host
_serror_rate','dst_host_srv_serror_rate','dst_host_rerror_rate','dst_host_srv_rerror_ra
te','attack','level'])

df.columns = columns
```



	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_same_srv_rate	dst_host_diff_srv_rate	dst_host_same_src_port_rate	dst_host_rerror_rate
0	0	udp	other	SF	146	0	0	0	0	0	...	0.00	0.60	0.88	
1	0	tcp	private	S0	0	0	0	0	0	0	...	0.10	0.05	0.00	
2	0	tcp	http	SF	232	8153	0	0	0	0	...	1.00	0.00	0.03	
3	0	tcp	http	SF	199	420	0	0	0	0	...	1.00	0.00	0.00	
4	0	tcp	private	REJ	0	0	0	0	0	0	...	0.07	0.07	0.00	

Hình 3 Thực hiện thêm tên cột cho dataset

5.1.2 Thực hiện kiểm tra dữ liệu

Ta sẽ thực hiện sử dụng các lệnh sau để kiểm tra dữ liệu. Ta sử dụng hàm **df.info()** sẽ cho ta biết định dạng và số lượng quan sát not-null của mỗi trường trong dataframe.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 125972 entries, 0 to 125971
Data columns (total 43 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   duration                                  125972 non-null  int64
1   protocol_type                             125972 non-null  object
2   service                                   125972 non-null  object
3   flag                                       125972 non-null  object
4   src_bytes                                 125972 non-null  int64
5   dst_bytes                                 125972 non-null  int64
6   land                                      125972 non-null  int64
7   wrong_fragment                           125972 non-null  int64
8   urgent                                    125972 non-null  int64
9   hot                                       125972 non-null  int64
10  num_failed_logins                         125972 non-null  int64
11  logged_in                                 125972 non-null  int64
12  num_compromised                           125972 non-null  int64
13  root_shell                                125972 non-null  int64
14  su_attempted                              125972 non-null  int64
15  num_root                                  125972 non-null  int64
16  num_file_creations                        125972 non-null  int64
17  num_shells                                125972 non-null  int64
18  num_access_files                          125972 non-null  int64
19  num_outbound_cmds                         125972 non-null  int64
20  is_host_login                             125972 non-null  int64
21  is_guest_login                            125972 non-null  int64
22  count                                      125972 non-null  int64
23  srv_count                                 125972 non-null  int64
24  serror_rate                               125972 non-null  float64
25  srv_serror_rate                           125972 non-null  float64
26  rerror_rate                               125972 non-null  float64
27  srv_rerror_rate                           125972 non-null  float64
28  same_srv_rate                             125972 non-null  float64
29  diff_srv_rate                             125972 non-null  float64
30  srv_diff_host_rate                       125972 non-null  float64
31  dst_host_count                             125972 non-null  int64
32  dst_host_srv_count                        125972 non-null  int64
33  dst_host_same_srv_rate                   125972 non-null  float64
34  dst_host_diff_srv_rate                   125972 non-null  float64
35  dst_host_same_src_port_rate              125972 non-null  float64
36  dst_host_srv_diff_host_rate              125972 non-null  float64
37  dst_host_serror_rate                     125972 non-null  float64
38  dst_host_srv_serror_rate                 125972 non-null  float64
39  dst_host_rerror_rate                     125972 non-null  float64
40  dst_host_srv_rerror_rate                 125972 non-null  float64
41  attack                                    125972 non-null  object
42  level                                    125972 non-null  int64
dtypes: float64(15), int64(24), object(4)
memory usage: 41.3+ MB
```

Hình 4 Thực hiện sử dụng hàm `df.info()`

Tiếp theo ta sử dụng `df.describe()` để thống kê định lượng.

```
# Thống kê định lượng
df.describe().T
```

duration	125972.0	287.146929	2.604526e+03	0.0	0.00	0.00	0.00	4.290800e+04
src_bytes	125972.0	45567.100824	5.870354e+06	0.0	0.00	44.00	276.00	1.379964e+09
dst_bytes	125972.0	19779.271433	4.021285e+06	0.0	0.00	0.00	516.00	1.309937e+09
land	125972.0	0.000198	1.408613e-02	0.0	0.00	0.00	0.00	1.000000e+00
wrong_fragment	125972.0	0.022688	2.535310e-01	0.0	0.00	0.00	0.00	3.000000e+00
urgent	125972.0	0.000111	1.436608e-02	0.0	0.00	0.00	0.00	3.000000e+00
hot	125972.0	0.204411	2.149977e+00	0.0	0.00	0.00	0.00	7.700000e+01
num_failed_logins	125972.0	0.001222	4.523932e-02	0.0	0.00	0.00	0.00	5.000000e+00
logged_in	125972.0	0.395739	4.890107e-01	0.0	0.00	0.00	1.00	1.000000e+00
num_compromised	125972.0	0.279253	2.394214e+01	0.0	0.00	0.00	0.00	7.479000e+03
root_shell	125972.0	0.001342	3.660299e-02	0.0	0.00	0.00	0.00	1.000000e+00
su_attempted	125972.0	0.001103	4.515456e-02	0.0	0.00	0.00	0.00	2.000000e+00
num_root	125972.0	0.302194	2.439971e+01	0.0	0.00	0.00	0.00	7.468000e+03
num_file_creations	125972.0	0.012669	4.839370e-01	0.0	0.00	0.00	0.00	4.300000e+01
num_shells	125972.0	0.000413	2.218122e-02	0.0	0.00	0.00	0.00	2.000000e+00
num_access_files	125972.0	0.004096	9.936995e-02	0.0	0.00	0.00	0.00	9.000000e+00
num_outbound_cmds	125972.0	0.000000	0.000000e+00	0.0	0.00	0.00	0.00	0.000000e+00

Hình 5 Thực hiện thống kê định lượng

Ta sử dụng lệnh `df.describe(include='object').T` để thống kê định tính.

```
# Thống kê định tính
df.describe(include='object').T
```

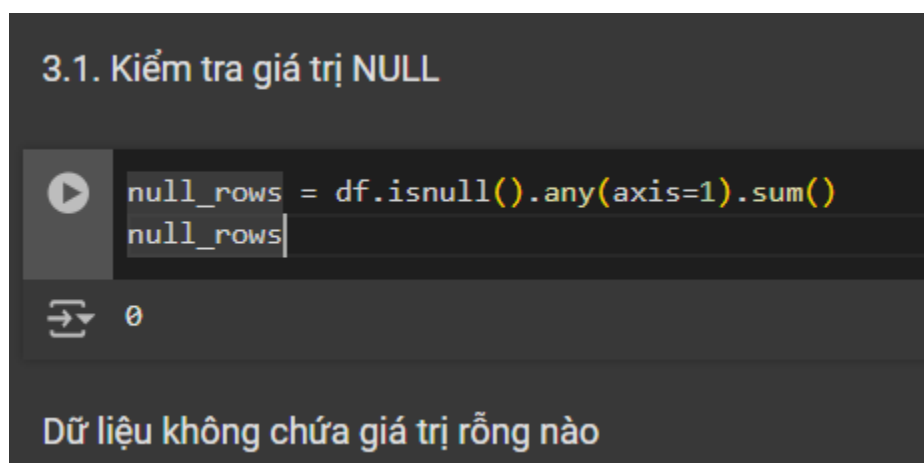
	count	unique	top	freq
protocol_type	125972	3	tcp	102688
service	125972	70	http	40338
flag	125972	11	SF	74944
attack	125972	23	normal	67342

Hình 6 Thực hiện thống kê định tính

5.1.3. Làm sạch dữ liệu

5.1.3.1 Kiểm tra dữ liệu rỗng

Ta sử dụng `null_rows = df.isnull().any(axis=1).sum` để kiểm tra dữ liệu có bị rỗng hay bị thiếu hay không.

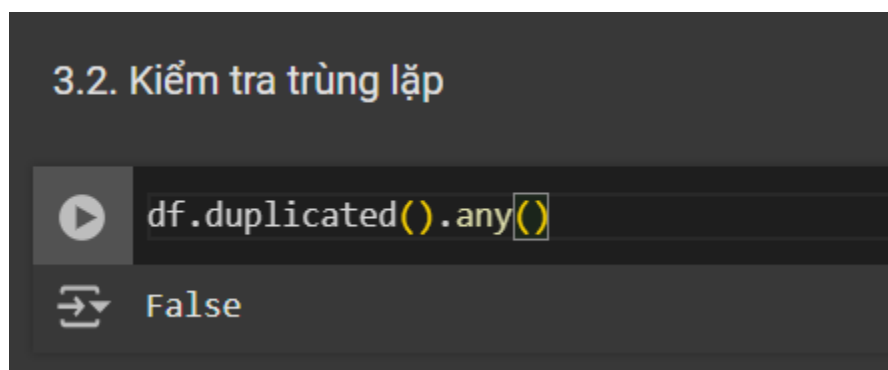


Hình 7 Thực hiện kiểm tra dữ liệu NULL

⇒ Ta có thể thấy giá trị bằng 0 chứng tỏ là dữ liệu không có giá trị rỗng.

5.1.3.2. Kiểm tra trùng lặp

Tiếp theo ta sử dụng `df.duplicated().any()` để kiểm tra xem dữ liệu có bị trùng lặp hay không.



Hình 8 Thực hiện kiểm tra trùng lặp

⇒ Kết quả của lệnh khi chạy là ra false nên dữ liệu không bị trùng lặp.

5.1.4. Phân loại dữ liệu

Ta cần phân loại dữ liệu để có thể dễ dàng hơn trong việc chạy các mô hình.

Bước đầu tiên ta cần phân loại các loại tấn công sang các dạng chính tương ứng như **normal**, **dos**, **u2r**, **r2l**, **probe**.


```
attacks_types = {
    'normal': 'normal',
    'back': 'dos',
    'buffer_overflow': 'u2r',
    'ftp_write': 'r2l',
    'guess_passwd': 'r2l',
    'imap': 'r2l',
    'ipsweep': 'probe',
    'land': 'dos',
    'loadmodule': 'u2r',
    'multihop': 'r2l',
    'neptune': 'dos',
    'nmap': 'probe',
    'perl': 'u2r',
    'phf': 'r2l',
    'pod': 'dos',
    'portsweep': 'probe',
    'rootkit': 'u2r',
    'satan': 'probe',
    'smurf': 'dos',
    'spy': 'r2l',
    'teardrop': 'dos',
    'warezclient': 'r2l',
    'warezmaster': 'r2l',
}
```

Hình 9 Thực hiện phân loại cột attack

```
#Thêm cột attack type
df['Attack Type'] = df.attack.apply(lambda r:attacks_types[r])
df.drop('attack', axis = 1, inplace = True)
df.head()
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_same_srv_rate	dst_host_diff_srv_rate	dst_host_same_src_port_rate	dst
0	0	udp	other	SF	146	0	0	0	0	0	...	0.00	0.60	0.88	
1	0	tcp	private	S0	0	0	0	0	0	0	...	0.10	0.05	0.00	
2	0	tcp	http	SF	232	8153	0	0	0	0	...	1.00	0.00	0.03	
3	0	tcp	http	SF	199	420	0	0	0	0	...	1.00	0.00	0.00	
4	0	tcp	private	REJ	0	0	0	0	0	0	...	0.07	0.07	0.00	

5 rows x 43 columns

Hình 10 Thực hiện thêm cột attack type

Ta sử dụng lệnh `cat_features = df.select_dtypes(include='object').columns` để xem các cột có giá trị **object** để phân loại.

```
[ ] cat_features = df.select_dtypes(include='object').columns
cat_features
```

```
Index(['protocol_type', 'service', 'flag', 'Attack Type'], dtype='object')
```

Hình 11 Thực hiện xem các cột để phân loại

Ta thực hiện chuyển đổi giá trị cột `protocol_type`

```
# chuyển đổi protocol_type
pmap = {'icmp':0, 'tcp':1, 'udp':2}
df['protocol_type'] = df['protocol_type'].map(pmap)
```

Hình 12 Thực hiện chuyển đổi giá trị cột `protocol_type`

Ta thực hiện chuyển đổi giá trị cột `flag` feature mapping

```
# chuyển đổi flag feature mapping
fmap = {'SF':0, 'S0':1, 'REJ':2, 'RSTR':3, 'RSTO':4, 'SH':5, 'S1':6, 'S2':7, 'RSTOS0':8, 'S3':9, 'OTH':10}
df['flag'] = df['flag'].map(fmap)
```

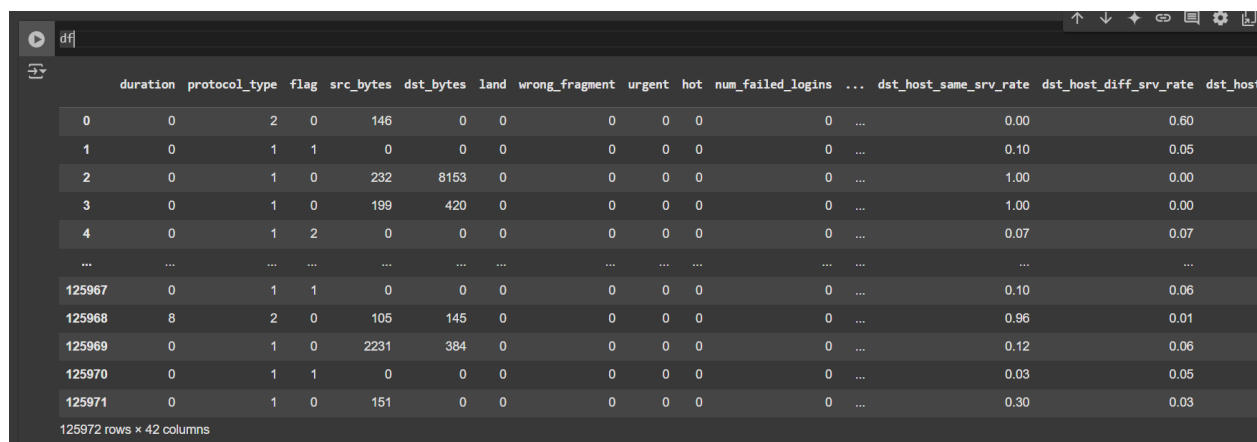
Hình 13 Thực hiện chuyển đổi giá trị `flag` feature mapping

Ta drop cột `service` vì giá trị cột này không cần thiết khi chạy dữ liệu

```
# Drop cột service
df.drop('service', axis = 1, inplace = True)
```

Hình 14 Thực hiện drop cột `service`

Ta xem lại giá trị các cột sau khi chuyển đổi dữ liệu:

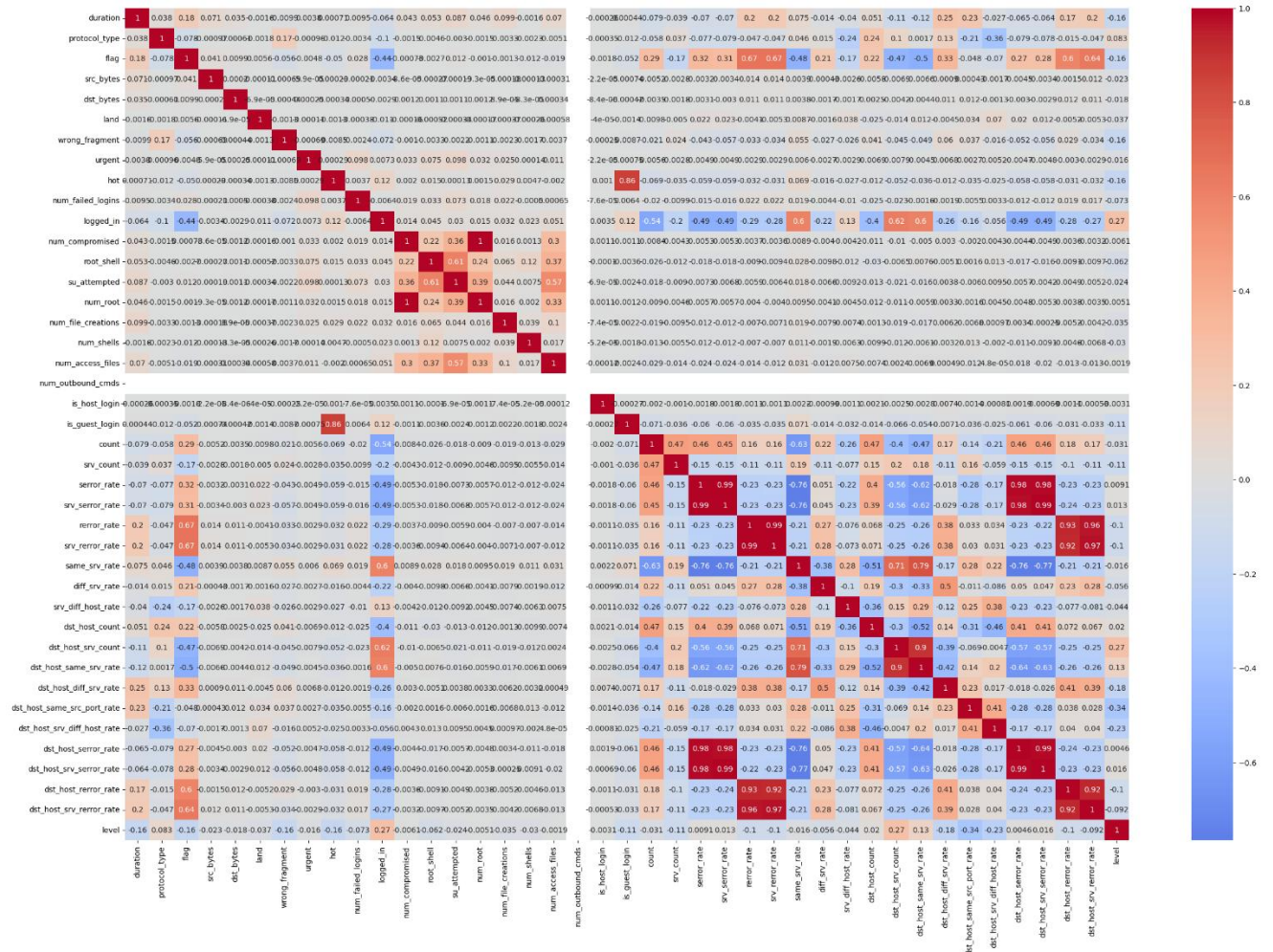


	duration	protocol_type	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	...	dst_host_same_srv_rate	dst_host_diff_srv_rate	dst_host
0	0	2	0	146	0	0	0	0	0	0	...	0.00	0.60	
1	0	1	1	0	0	0	0	0	0	0	...	0.10	0.05	
2	0	1	0	232	8153	0	0	0	0	0	...	1.00	0.00	
3	0	1	0	199	420	0	0	0	0	0	...	1.00	0.00	
4	0	1	2	0	0	0	0	0	0	0	...	0.07	0.07	
...
125967	0	1	1	0	0	0	0	0	0	0	...	0.10	0.06	
125968	8	2	0	105	145	0	0	0	0	0	...	0.96	0.01	
125969	0	1	0	2231	384	0	0	0	0	0	...	0.12	0.06	
125970	0	1	1	0	0	0	0	0	0	0	...	0.03	0.05	
125971	0	1	0	151	0	0	0	0	0	0	...	0.30	0.03	

Hình 15 Thực hiện xem lại dataset

5.1.5. Loại bỏ đặc trưng phụ thuộc

Ta thực hiện tính toán ma trận tương quan bằng lệnh **correlation** để kiểm tra các dữ liệu có độ tương quan lớn để có thể loại bớt cột giúp gia tăng tốc độ xử lý cũng như tránh dự đoán sai.



Hình 16 Bản đồ heatmap của dữ liệu

Ta sẽ drop các cột có độ tương quan cao.

```
df.drop('num_root', axis = 1, inplace = True)
df.drop('is_guest_login', axis = 1, inplace = True)
df.drop('dst_host_srv_count', axis = 1, inplace = True)
df.drop('dst_host_same_srv_rate', axis = 1, inplace = True)
df.drop('dst_host_error_rate', axis = 1, inplace = True)
df.drop('dst_host_srv_error_rate', axis = 1, inplace = True)
df.drop('dst_host_diff_error_rate', axis = 1, inplace = True)
df.drop('dst_host_srv_error_rate', axis = 1, inplace = True)
```

Hình 17 Thực hiện xóa các cột có độ tương quan cao

Kết quả sau khi ta thực hiện xóa cột, dữ liệu còn 125972 dòng và 34 cột.

	duration	protocol_type	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	...	srv_error_rate	same_srv_rate	diff_srv_rate	srv_diff_host_rate
0	0	2	0	146	0	0	0	0	0	0	...	0.0	0.08	0.15	0.00
1	0	1	1	0	0	0	0	0	0	0	...	0.0	0.05	0.07	0.00
2	0	1	0	232	8153	0	0	0	0	0	...	0.0	1.00	0.00	0.00
3	0	1	0	199	420	0	0	0	0	0	...	0.0	1.00	0.00	0.09
4	0	1	2	0	0	0	0	0	0	0	...	1.0	0.16	0.06	0.00
...
125967	0	1	1	0	0	0	0	0	0	0	...	0.0	0.14	0.06	0.00
125968	8	2	0	105	145	0	0	0	0	0	...	0.0	1.00	0.00	0.00
125969	0	1	0	2231	384	0	0	0	0	0	...	0.0	1.00	0.00	0.00
125970	0	1	1	0	0	0	0	0	0	0	...	0.0	0.06	0.05	0.00
125971	0	1	0	151	0	0	0	0	0	0	...	0.0	1.00	0.00	0.00

Hình 18 Kết quả dữ liệu sau khi tiền xử lý

5.2. Áp dụng các mô hình thuật toán cho dataset

5.2.1. Naïve Bayes

Ta thực hiện chia dữ liệu thành 2 tập huấn luyện và kiểm tra và sử dụng grid search để tìm mô hình tốt nhất.

```
df_copy = df.copy()
X = df_copy.drop(["Attack Type"], axis=1)
y = df_copy["Attack Type"]

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

param_grid_nb = {
    'alpha': [0.1, 0.5, 1.0, 5.0, 10.0],
    'fit_prior': [True, False]
}

# Khởi tạo và huấn luyện mô hình Native Bayes
model = MultinomialNB()

grid_search_nb = GridSearchCV(
    estimator=MultinomialNB(),
    param_grid=param_grid_nb,
    scoring='accuracy',
    cv=5,
    verbose=2,
    n_jobs=-1
)

grid_search_nb.fit(X_train, y_train)

# Dự đoán kết quả trên tập kiểm tra bằng mô hình đã đào tạo
nb_predictions = grid_search_nb.predict(X_test)

# In ra thông số tốt nhất và kết quả đạt được
print("Mô hình tốt nhất:", grid_search_nb.best_params_)

Fitting 5 folds for each of 10 candidates, totalling 50 fits
Mô hình tốt nhất: {'alpha': 0.1, 'fit_prior': True}
```

Hình 19 Thực hiện chạy mô hình Naïve Bayes

Ta thực hiện việc đánh giá mô hình

```
[ ] # Đánh giá mô hình
accuracy_nb = accuracy_score(y_test, nb_predictions)
report = classification_report(y_test, nb_predictions)
print("Accuracy: ", accuracy_nb)
print("Report: ", report)
```

➡ Accuracy: 0.41627857747671465

Report:

	precision	recall	f1-score	support
dos	0.54	0.98	0.70	13941
normal	0.98	0.10	0.19	20014
probe	0.00	0.00	0.00	3526
r2l	0.03	0.08	0.04	291
u2r	0.00	0.70	0.00	20
accuracy			0.42	37792
macro avg	0.31	0.37	0.19	37792
weighted avg	0.72	0.42	0.36	37792

Hình 20 Thực hiện đánh giá mô hình

Ta sử dụng hàm DataFrame để xem kết quả của grid search

```

import pandas as pd

# Lấy kết quả từ GridSearchCV
cv_results = grid_search_nb.cv_results_

# Tạo DataFrame với các cột cần thiết
results_df = pd.DataFrame({
    'mean_test_score': cv_results['mean_test_score'],
    'std_test_score': cv_results['std_test_score'],
    'param_alpha': cv_results['param_alpha'],
    'param_fit_prior': cv_results['param_fit_prior']
})

# Hiển thị DataFrame
print("Kết quả Grid Search:")
print(results_df)

# Sắp xếp theo độ chính xác trung bình (giảm dần)
results_df_sorted = results_df.sort_values(by='mean_test_score', ascending=False)
print("\nTop 5 kết quả tối ưu:")
print(results_df_sorted.head())

```

→ Kết quả Grid Search:

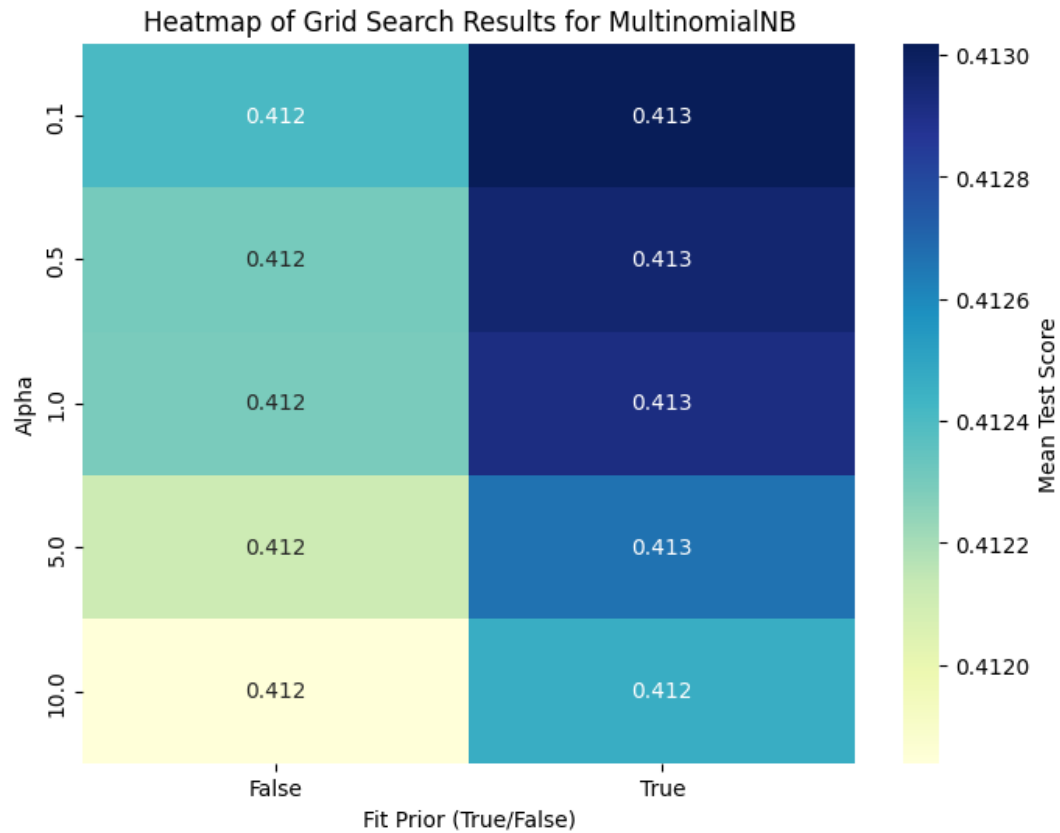
	mean_test_score	std_test_score	param_alpha	param_fit_prior
0	0.413019	0.005733	0.1	True
1	0.412406	0.005812	0.1	False
2	0.412962	0.005740	0.5	True
3	0.412304	0.005805	0.5	False
4	0.412917	0.005745	1.0	True
5	0.412293	0.005804	1.0	False
6	0.412667	0.005709	5.0	True
7	0.412112	0.005803	5.0	False
8	0.412463	0.005751	10.0	True
9	0.411839	0.005751	10.0	False

Top 5 kết quả tối ưu:

	mean_test_score	std_test_score	param_alpha	param_fit_prior
0	0.413019	0.005733	0.1	True
2	0.412962	0.005740	0.5	True
4	0.412917	0.005745	1.0	True
6	0.412667	0.005709	5.0	True
8	0.412463	0.005751	10.0	True

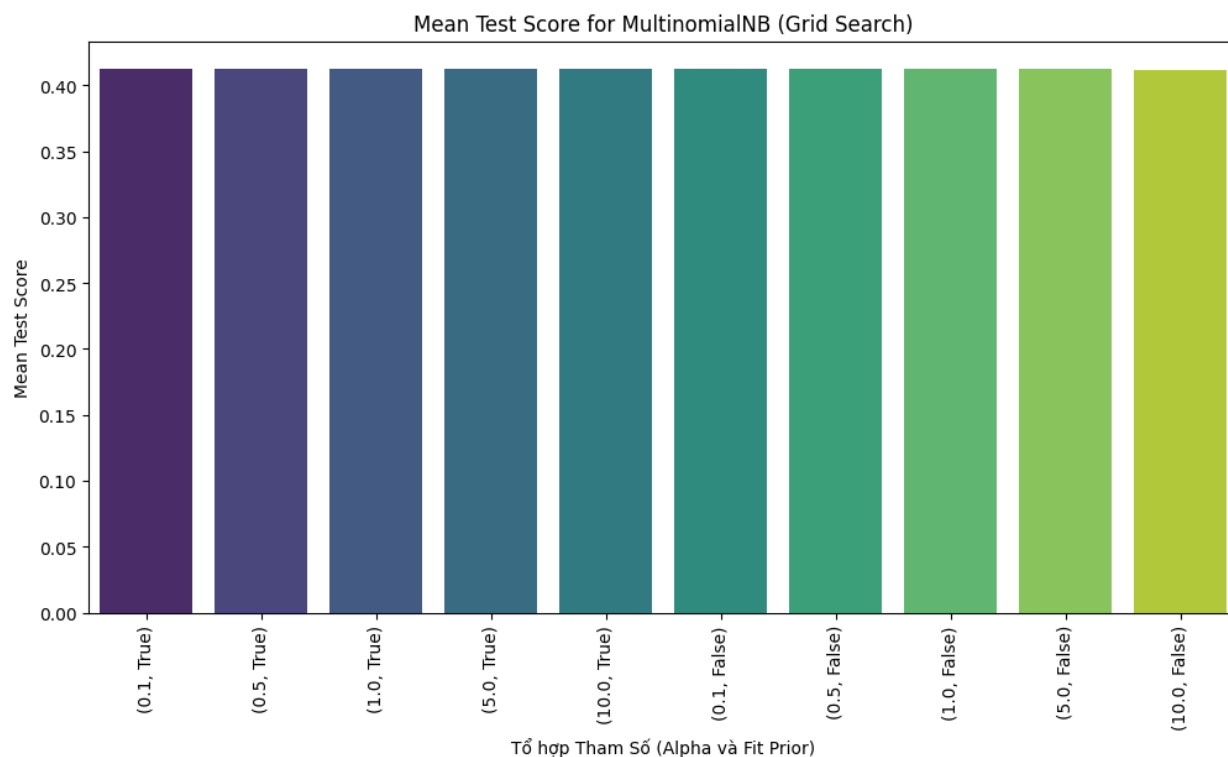
Hình 21 Thực hiện hiển thị kết quả grid search dưới dạng DataFrame

Ta hiển thị kết quả dưới dạng heatmap để dễ quan sát



Hình 22 Thực hiện hiển thị kết quả grid search dưới dạng HeatMap

Ta hiển thị dưới dạng biểu đồ.



Hình 23 Thực hiện hiển thị kết quả grid search dưới dạng cột

5.2.2. Decision Tree

Ta thực hiện chia dữ liệu thành 2 tập huấn luyện và kiểm tra và sử dụng grid search để tìm mô hình tốt nhất

```
df_copy = df.copy()
X = df_copy.drop(["Attack Type"], axis=1)
y = df_copy["Attack Type"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=43)

# Tạo một lưới các giá trị tham số cần thử
param_grid = {
    'max_depth': [3, 5, 7, 10],
    'min_samples_split': [2, 5, 10],
    'criterion': ['gini', 'entropy']
}

model = DecisionTreeClassifier(random_state=0)

# Huấn luyện mô hình và tìm tham số tốt nhất
grid_search = GridSearchCV(model, param_grid, cv=5)
grid_search.fit(X_train, y_train)

# Lấy mô hình tốt nhất
best_model = grid_search.best_estimator_
print("Best Parameters:", grid_search.best_params_)

/usr/local/lib/python3.10/dist-packages/numpy/ma/core.py:2820: RuntimeWarning: invalid value encountered in cast
_data = np.array(data, dtype=dtype, copy=copy,
Best Parameters: {'criterion': 'gini', 'max_depth': 10, 'min_samples_split': 5}
```


Hình 24 Thực hiện chạy mô hình Decision tree

Ta thực hiện đánh giá mô hình

```
# Huấn luyện mô hình với các tham số tối ưu
best_model.fit(X_train, y_train)

# Đánh giá mô hình
clfd_predict = best_model.predict(X_test)
clfd_score = metrics.accuracy_score(y_test, clfd_predict)
print("Độ chính xác: ", clfd_score)
print("Báo cáo: ", metrics.classification_report(y_test, clfd_predict))
```

Độ chính xác: 0.99714240355612

Báo cáo:

	precision	recall	f1-score	support
dos	1.00	1.00	1.00	4543
normal	1.00	1.00	1.00	6770
probe	0.99	0.99	0.99	1176
r2l	0.90	0.98	0.94	107
u2r	0.00	0.00	0.00	2
accuracy			1.00	12598
macro avg	0.78	0.79	0.79	12598
weighted avg	1.00	1.00	1.00	12598

Hình 25 Thực hiện đánh giá mô hình

Hiển thị kết quả của grid search dưới dạng DataFrame

```

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Lấy thông tin kết quả từ Grid Search
results = grid_search.cv_results_

# Tạo DataFrame từ kết quả
results_df = pd.DataFrame({
    'mean_test_score': results['mean_test_score'], # Độ chính xác trung bình từ cross-validation
    'param_max_depth': results['param_max_depth'],
    'param_min_samples_split': results['param_min_samples_split'],
    'param_criterion': results['param_criterion']
})

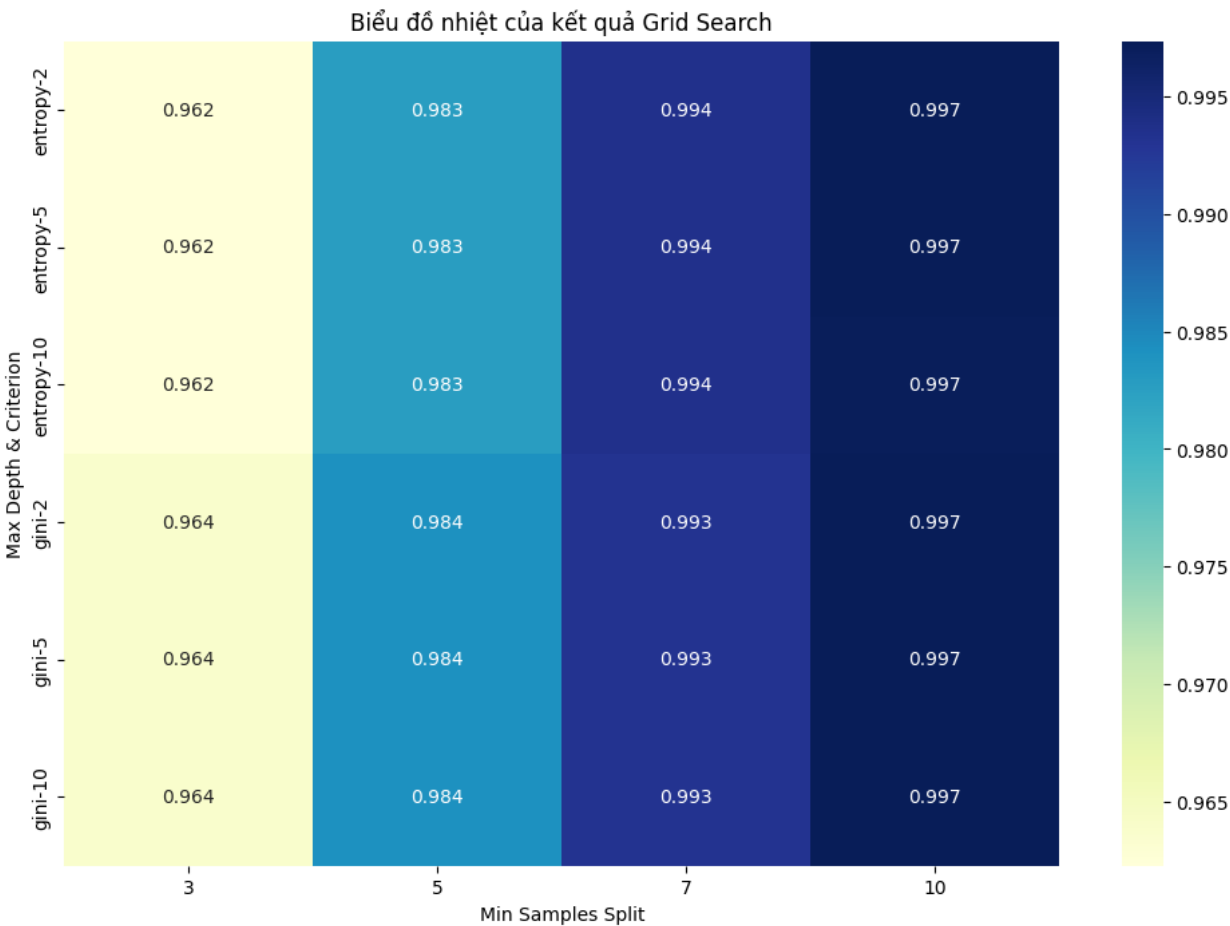
# Hiển thị dữ liệu kết quả
print(results_df)

```

	mean_test_score	param_max_depth	param_min_samples_split	param_criterion
0	0.963757	3	2	gini
1	0.963757	3	5	gini
2	0.963757	3	10	gini
3	0.984079	5	2	gini
4	0.984079	5	5	gini
5	0.984079	5	10	gini
6	0.993208	7	2	gini
7	0.993235	7	5	gini
8	0.993182	7	10	gini
9	0.997310	10	2	gini
10	0.997354	10	5	gini
11	0.997319	10	10	gini
12	0.962275	3	2	entropy
13	0.962275	3	5	entropy
14	0.962275	3	10	entropy
15	0.982721	5	2	entropy
16	0.982721	5	5	entropy
17	0.982721	5	10	entropy
18	0.993552	7	2	entropy
19	0.993561	7	5	entropy
20	0.993526	7	10	entropy
21	0.997230	10	2	entropy
22	0.997239	10	5	entropy
23	0.997125	10	10	entropy

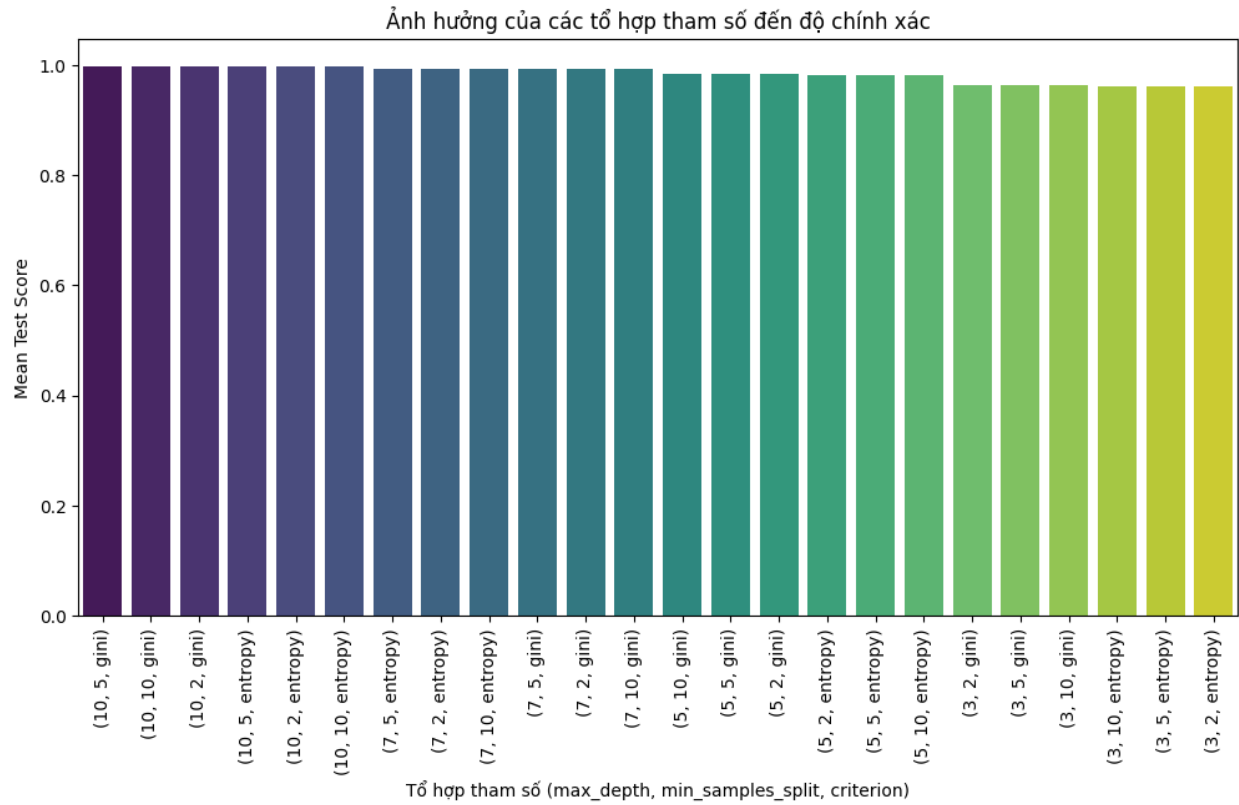
Hình 26 Thực hiện hiển thị kết quả grid search dưới dạng DataFrame

Hiển thị kết quả dưới dạng biểu đồ nhiệt.



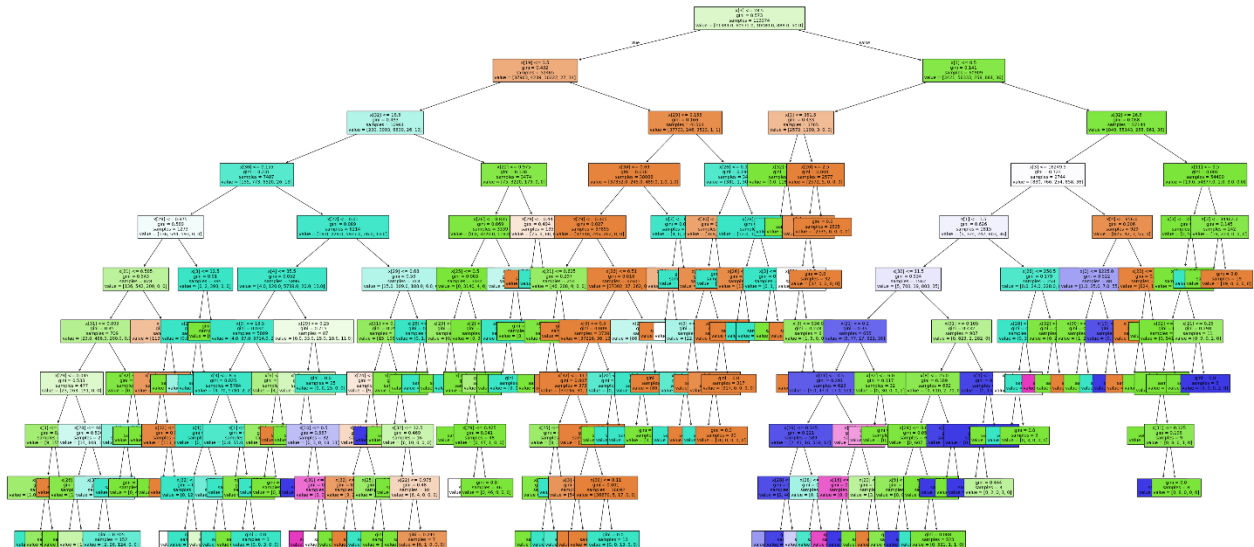
Hình 27 Thực hiện hiển thị kết quả grid search dưới dạng HeatMap

Hiển thị kết quả dưới dạng biểu đồ cột



Hình 28 Thực hiện hiển thị kết quả grid search dưới dạng cột

Hiển thị kết quả cây quyết định bằng biểu đồ.



Hình 29 Thực hiện hiển thị kết quả của cây quyết định

5.2.3. Random Forest

Ta thực hiện chia dữ liệu thành 2 tập huấn luyện và kiểm tra và sử dụng grid search để tìm mô hình tốt nhất

```
df_copy = df.copy()
X = df_copy.drop(["Attack Type"], axis=1)
y = df_copy["Attack Type"]

# Split test and train data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

[ ] from sklearn.ensemble import RandomForestClassifier

# Định nghĩa bộ tham số cần tối ưu
param_grid_rf = {
    'n_estimators': [50, 100],
    'max_depth': [None, 10],
    'min_samples_split': [2, 5]
}

# Khởi tạo mô hình Random Forest
rf = RandomForestClassifier(random_state=42)

# Thiết lập GridSearchCV
grid_search_rf = GridSearchCV(estimator=rf, param_grid=param_grid_rf, scoring='accuracy', cv=5, verbose=2, n_jobs=-1)

# Huấn luyện mô hình với GridSearch
grid_search_rf.fit(X_train, y_train)

# Lấy mô hình tốt nhất
best_model = grid_search_rf.best_estimator_
print("Best Parameters:", grid_search_rf.best_params_)

Fitting 5 folds for each of 8 candidates, totalling 40 fits
Best Parameters: {'max_depth': None, 'min_samples_split': 2, 'n_estimators': 100}
```

Hình 30 Thực hiện chạy mô hình Random Forest

Ta thực hiện đánh giá mô hình

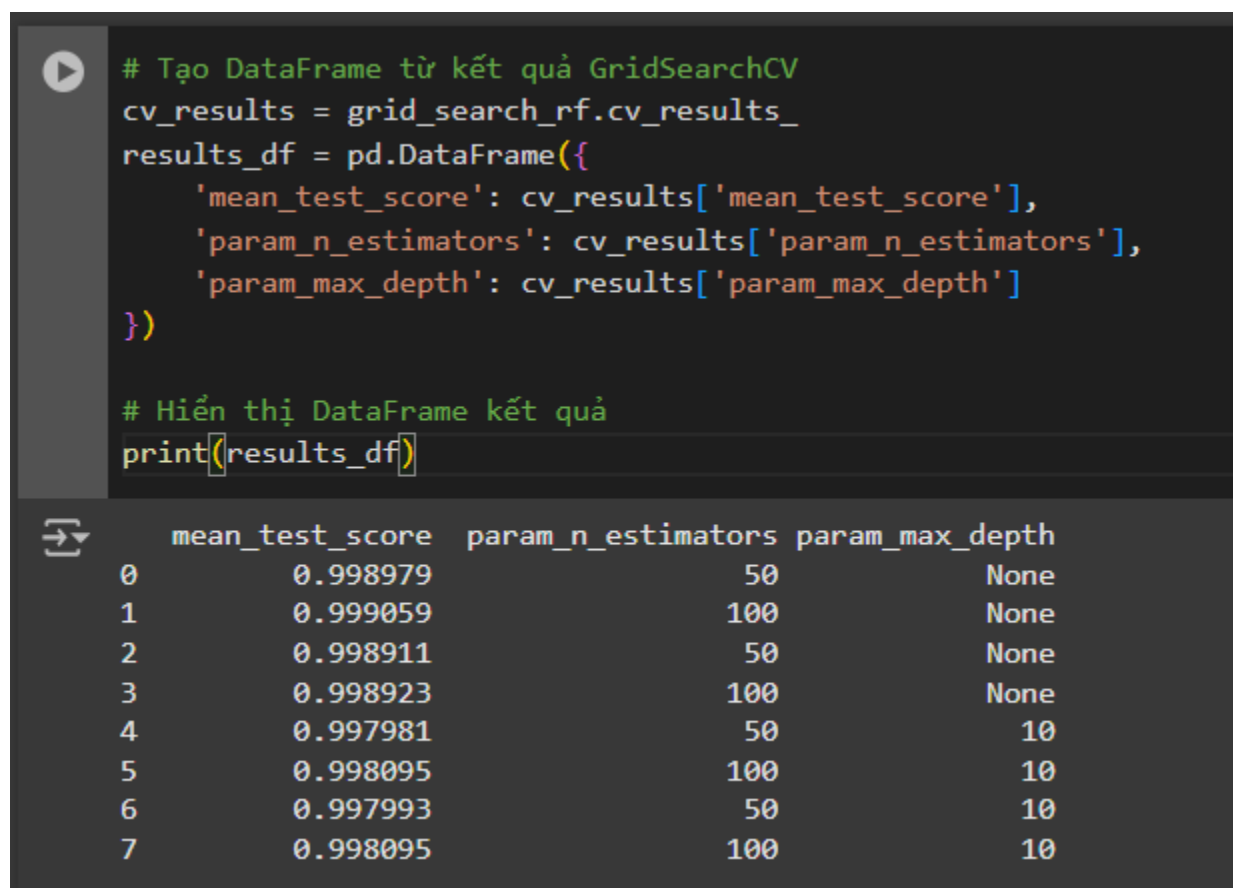
```
# Đánh giá mô hình
rf_predict = best_model.predict(X_test)
rf_score = metrics.accuracy_score(y_test, rf_predict)
print("Độ chính xác:", rf_score)
print("Báo cáo: ", metrics.classification_report(y_test, rf_predict))

Độ chính xác: 0.999417866215072
Báo cáo:
```

	precision	recall	f1-score	support
dos	1.00	1.00	1.00	13941
normal	1.00	1.00	1.00	20014
probe	1.00	1.00	1.00	3526
r2l	0.99	0.98	0.98	291
u2r	0.88	0.75	0.81	20
accuracy			1.00	37792
macro avg	0.97	0.95	0.96	37792
weighted avg	1.00	1.00	1.00	37792

Hình 31 Thực hiện đánh giá mô hình

Hiển thị kết quả của grid search dưới dạng DataFrame



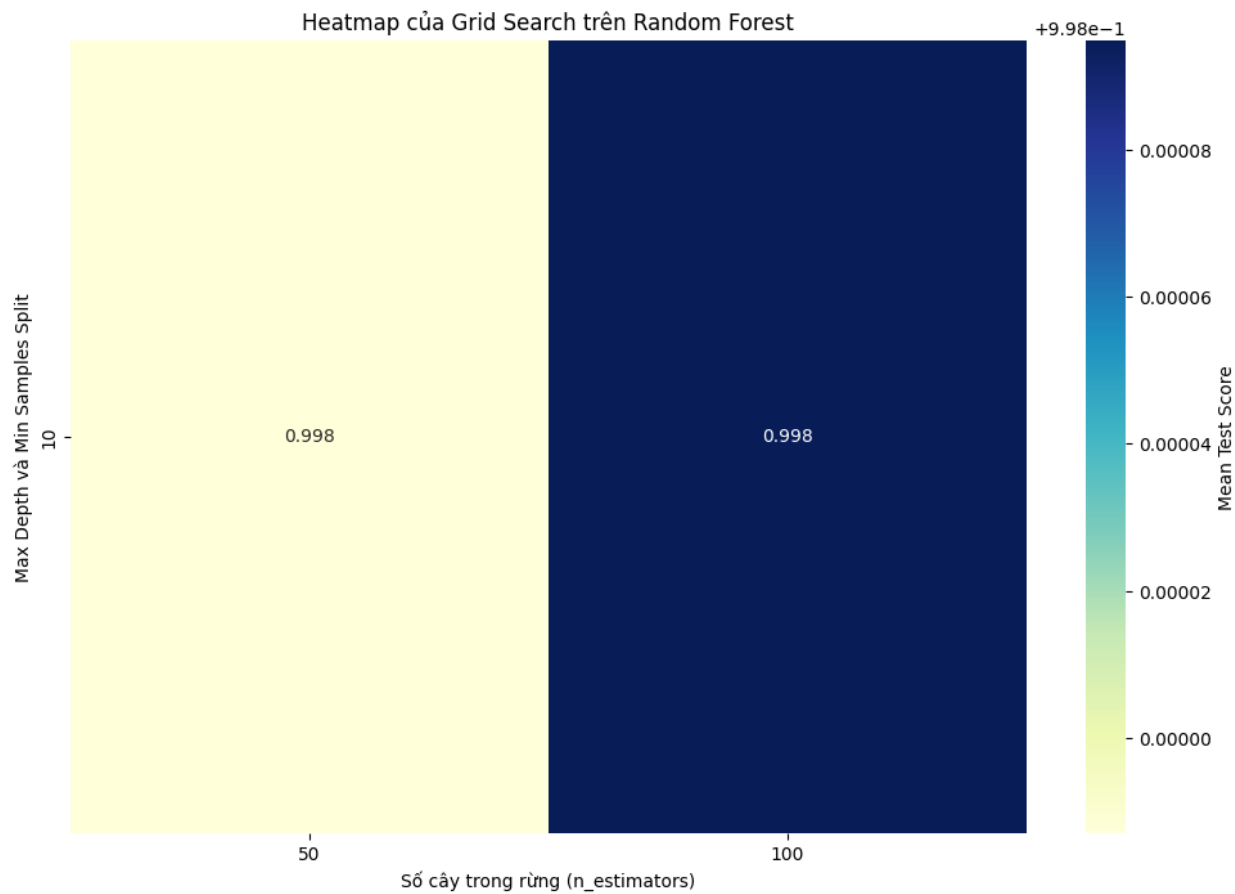
```
# Tạo DataFrame từ kết quả GridSearchCV
cv_results = grid_search_rf.cv_results_
results_df = pd.DataFrame({
    'mean_test_score': cv_results['mean_test_score'],
    'param_n_estimators': cv_results['param_n_estimators'],
    'param_max_depth': cv_results['param_max_depth']
})

# Hiển thị DataFrame kết quả
print(results_df)
```

	mean_test_score	param_n_estimators	param_max_depth
0	0.998979	50	None
1	0.999059	100	None
2	0.998911	50	None
3	0.998923	100	None
4	0.997981	50	10
5	0.998095	100	10
6	0.997993	50	10
7	0.998095	100	10

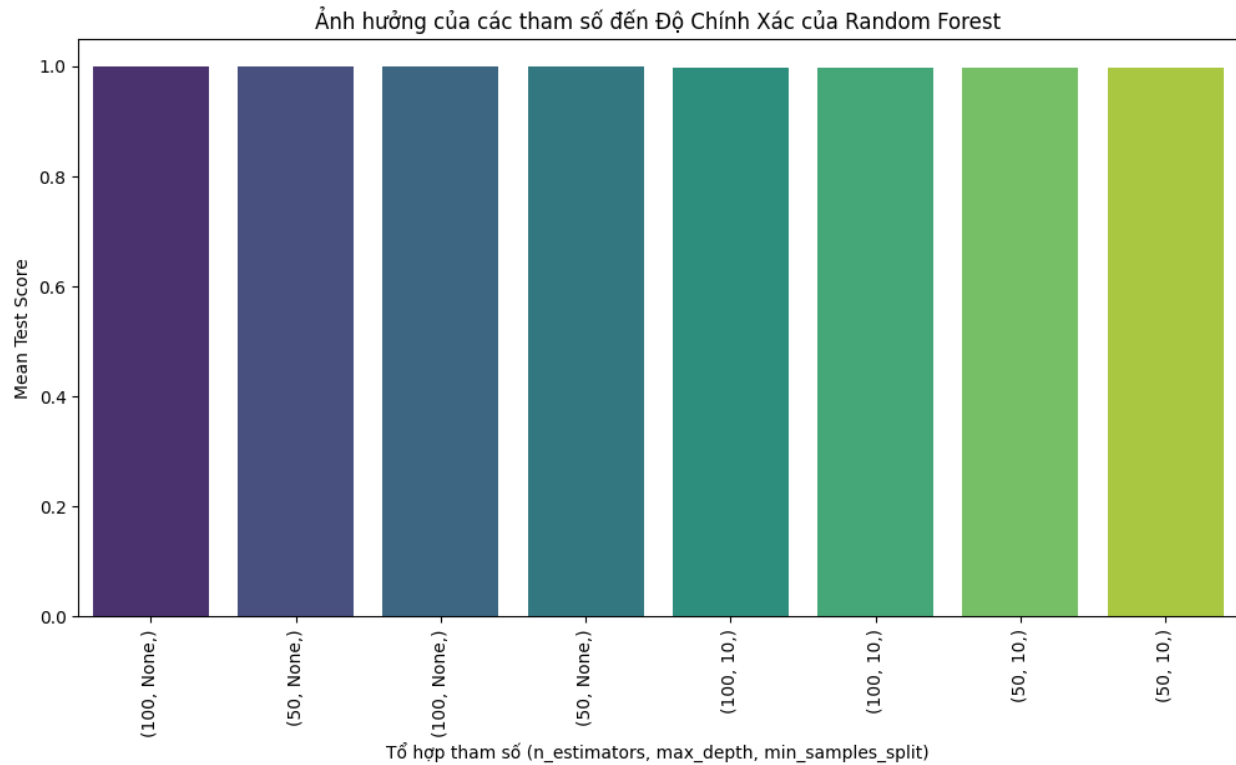
Hình 32 Thực hiện hiển thị kết quả grid search dưới dạng DataFrame

Hiển thị kết quả dưới dạng heatmap



Hình 33 Thực hiện hiển thị kết quả grid search dưới dạng HeatMap

Hiển thị kết quả dưới dạng cột.



Hình 34 Thực hiện hiển thị kết quả grid search dưới dạng cột

5.2.4. Logistic Regression

Ta thực hiện chia dữ liệu thành 2 tập huấn luyện và kiểm tra và sử dụng grid search để tìm mô hình tốt nhất


```

df_copy = df.copy()
X = df_copy.drop(["Attack Type"], axis=1)
y = df_copy["Attack Type"]

# Scale features for Logistic Regression
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

# Split test and train data
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)

[ ] param_grid_lr = {
    'C': [0.1, 1, 10],
    'penalty': ['l2'],
    'solver': ['liblinear']
}

# Khởi tạo mô hình Logistic Regression
lr = LogisticRegression(max_iter=200, random_state=42)

# Thiết lập GridSearchCV
grid_search_lr = GridSearchCV(estimator=lr, param_grid=param_grid_lr, scoring='accuracy', cv=5, verbose=2, n_jobs=-1)

# Huấn luyện mô hình với GridSearch
grid_search_lr.fit(X_train, y_train)
# In ra các tham số tốt nhất
print("Mô hình tốt nhất:", grid_search_lr.best_params_)

Fitting 5 folds for each of 3 candidates, totalling 15 fits
Mô hình tốt nhất: {'C': 10, 'penalty': 'l2', 'solver': 'liblinear'}

```

Hình 35 Thực hiện chạy mô hình Logistic regression

Ta thực hiện việc đánh giá mô hình

```

# Đánh giá mô hình
lr_predict = grid_search_lr.predict(X_test)
lr_score = metrics.accuracy_score(y_test, lr_predict)
print("Độ chính xác:", lr_score)
print("Báo cáo: \n", metrics.classification_report(y_test, lr_predict))

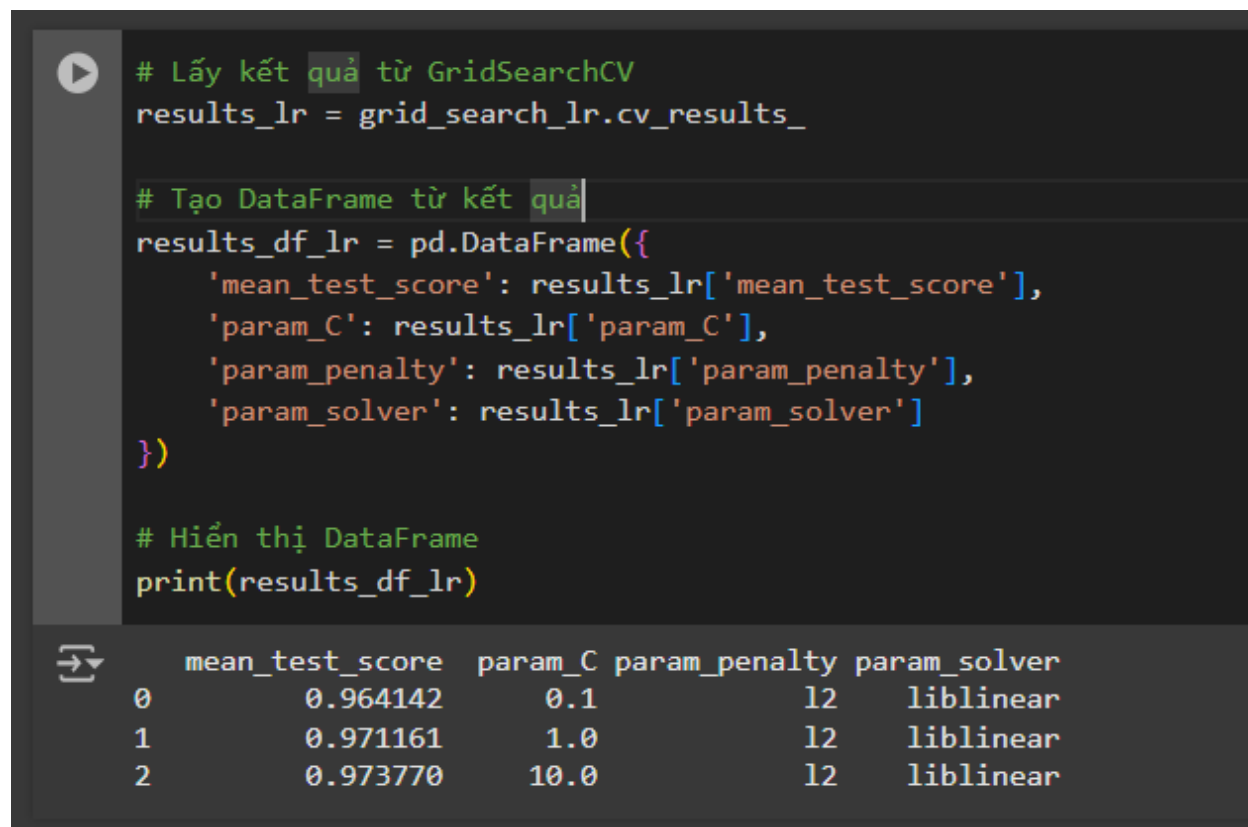
Độ chính xác: 0.9707080863674852
Báo cáo:

```

	precision	recall	f1-score	support
dos	0.98	0.98	0.98	13941
normal	0.97	0.98	0.97	20014
probe	0.92	0.91	0.92	3526
r2l	0.85	0.83	0.84	291
u2r	0.82	0.45	0.58	20
accuracy			0.97	37792
macro avg	0.91	0.83	0.86	37792
weighted avg	0.97	0.97	0.97	37792

Hình 36 Thực hiện đánh giá mô hình

Hiển thị kết quả của grid search dưới dạng DataFrame



```
# Lấy kết quả từ GridSearchCV
results_lr = grid_search_lr.cv_results_

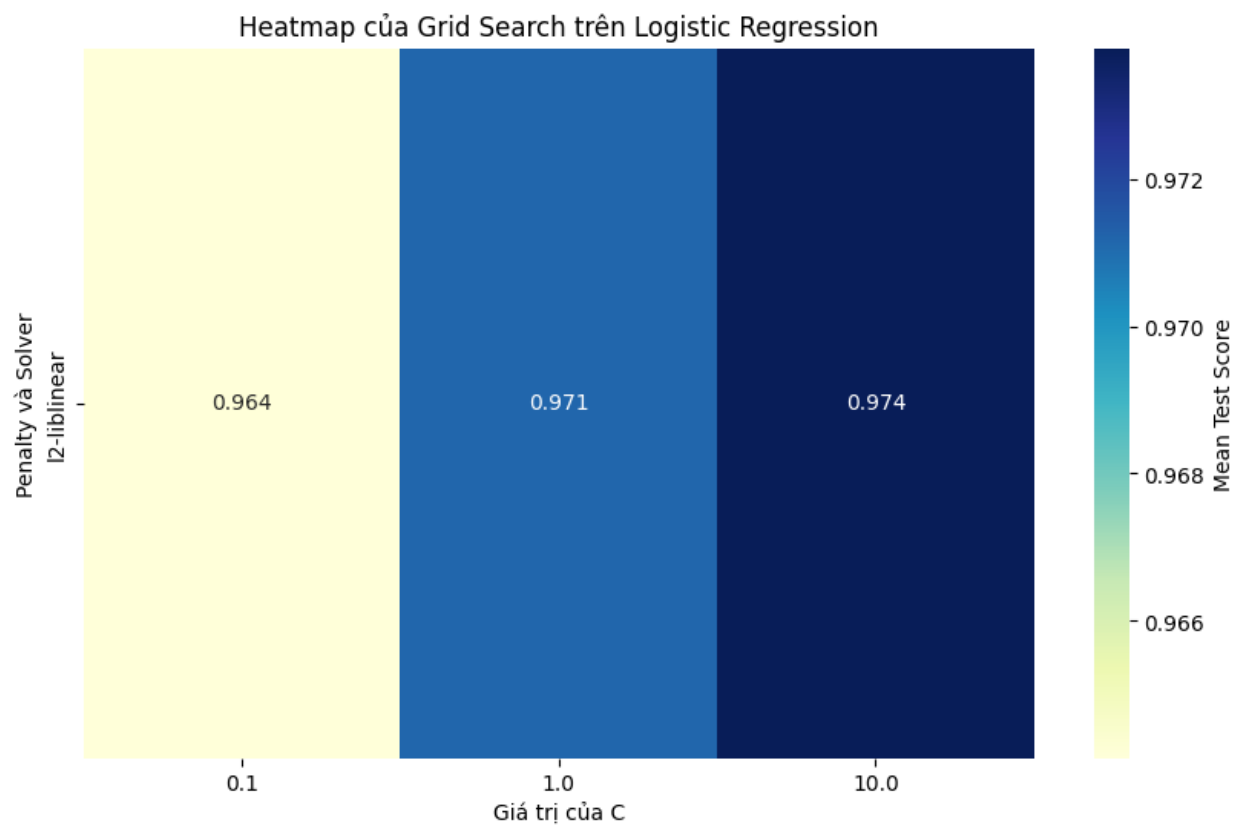
# Tạo DataFrame từ kết quả
results_df_lr = pd.DataFrame({
    'mean_test_score': results_lr['mean_test_score'],
    'param_C': results_lr['param_C'],
    'param_penalty': results_lr['param_penalty'],
    'param_solver': results_lr['param_solver']
})

# Hiển thị DataFrame
print(results_df_lr)
```

	mean_test_score	param_C	param_penalty	param_solver
0	0.964142	0.1	12	liblinear
1	0.971161	1.0	12	liblinear
2	0.973770	10.0	12	liblinear

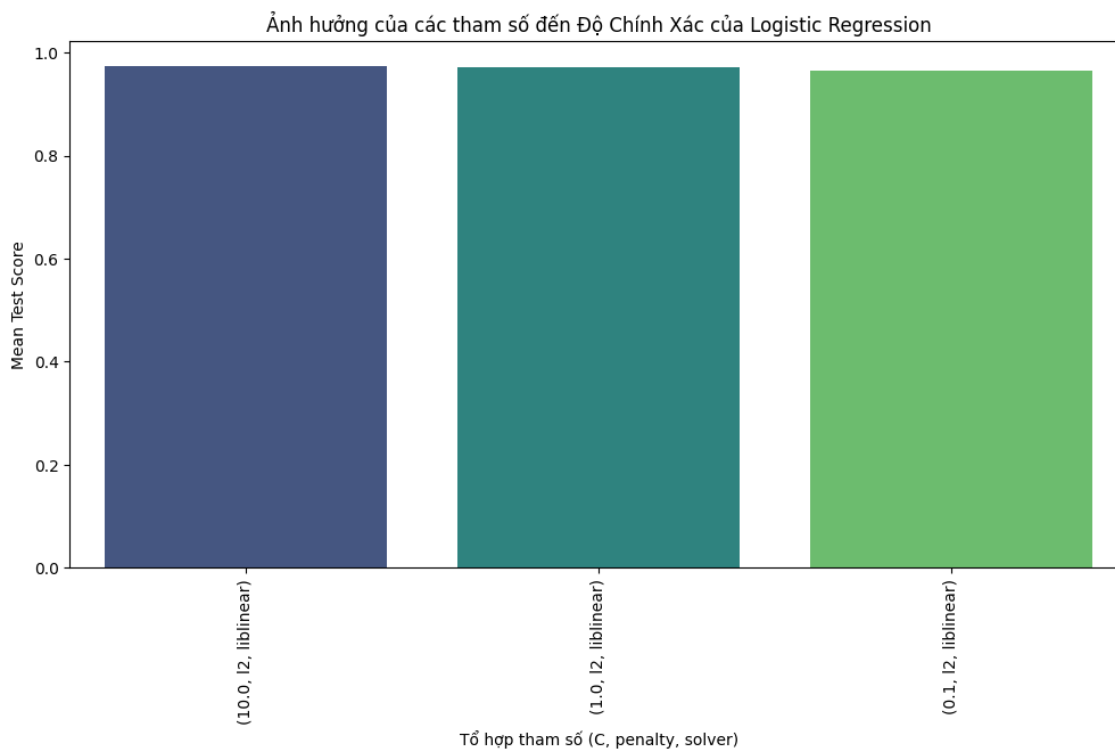
Hình 37 Thực hiện hiển thị kết quả grid search dưới dạng DataFrame

Hiển thị kết quả grid search dưới dạng HeatMap



Hình 38 Thực hiện hiển thị kết quả grid search dưới dạng HeatMap

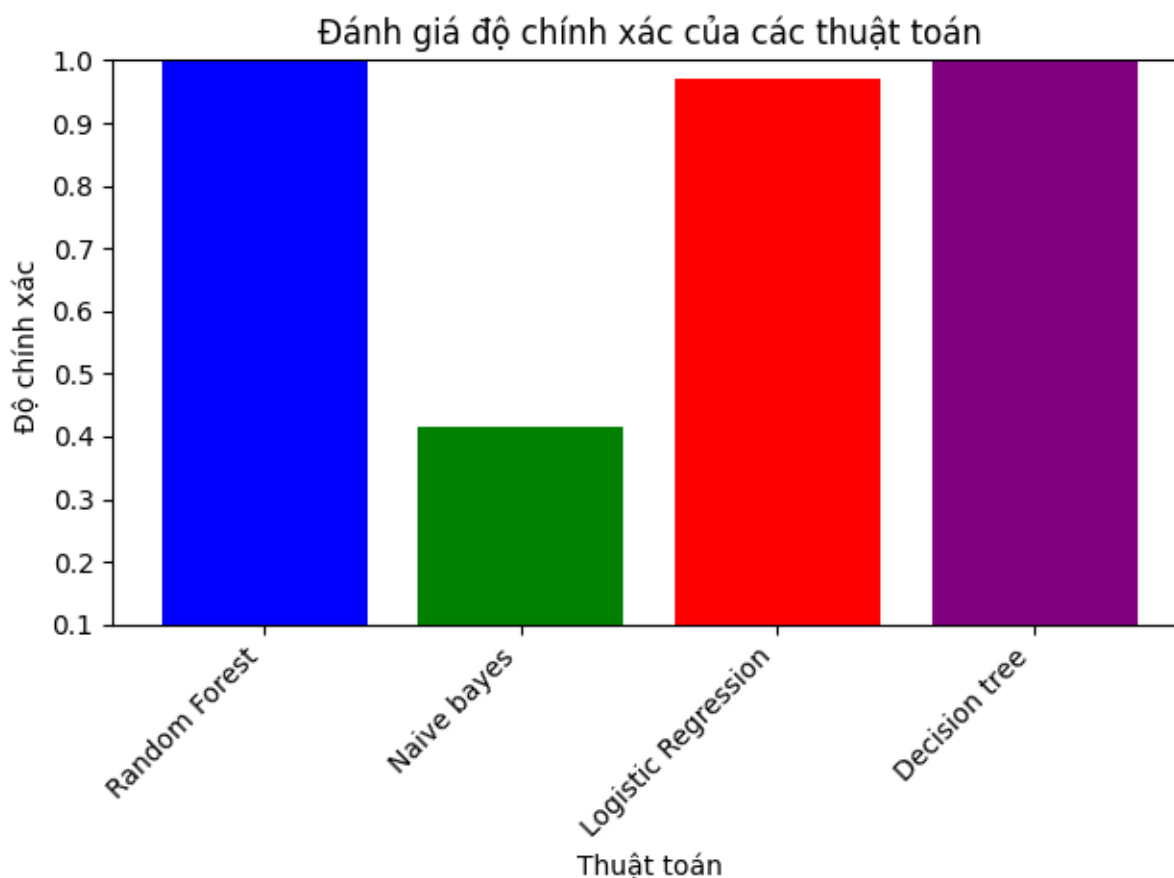
Hiển thị kết quả grid search dưới dạng cột



Hình 39 Thực hiện hiển thị kết quả grid search dưới dạng cột

5.3. So sánh độ chính xác của các thuật toán và đánh giá

Dựa theo các kết quả mà ta đã nhận được sau khi chạy các mô hình.



Hình 40 Thực hiện so sánh độ chính xác các thuật toán

⇒ Ta có thể thấy được độ chính xác của Naïve Byes là thấp nhất trong 3 thuật toán còn lại. Với 3 thuật toán Random Forest, Logisitic Regression, Decision Tree có độ chính xác khá cao .

Đánh giá các mô hình:

	Độ chính xác	Recall	F1-score	Precision	Hạn chế
Naïve Bayes	Không tốt	Cao	Cao	Cao	Đòi hỏi thời gian và tài nguyên tính toán lớn.
Random Forest	Tốt	Cao	Cao	Cao	Nhạy cảm với dữ liệu nhiễu.
Logistic Regression	Tốt	Cao	Cao	Cao	Đòi hỏi thời gian và tài

					nguyên tính toán lớn.
Decision Tree	Tốt	Cao	Cao	Cao	Đòi hỏi thời gian và tài nguyên tính toán lớn.

CHƯƠNG VI: KẾT LUẬN

Hệ thống phát hiện xâm nhập (IDS) là một thành phần quan trọng trong việc bảo vệ các hệ thống mạng và tài nguyên thông tin trước những mối đe dọa an ninh mạng ngày càng phức tạp.

Với sự kết hợp của các phương pháp phân tích truyền thống và các thuật toán học máy hiện đại, IDS có khả năng phát hiện các cuộc tấn công nhanh chóng, chính xác và giảm thiểu thiệt hại.

Hướng phát triển:

- **Tích hợp IDS với các hệ thống tự động hóa và bảo mật khác** như IPS (Intrusion Prevention System) để nâng cao khả năng phản ứng tức thời.
- **Áp dụng các công nghệ tiên tiến** như xử lý ngôn ngữ tự nhiên (NLP) để cải thiện hiệu suất phát hiện tấn công phức tạp.
- **Xây dựng các mô hình học sâu** có khả năng xử lý thời gian thực nhằm phát hiện, ngăn chặn các cuộc tấn công zero-day.

CHƯƠNG VII: BẢNG PHÂN CÔNG CÔNG VIỆC

	Nguyễn Dương Chí Tâm (21520439)	Huỳnh Mạnh Huy (21520259)	Đỗ Hiền Thảo (21520460)	Nguyễn Trương Đình Giang (21520215)
Tìm hiểu và thực hiện chương 1, 5	X			
Tìm hiểu và thực hiện chương 2, 5		X		

Tìm hiểu và thực hiện chương 3, 5				X
Tìm hiểu và thực hiện chương 4, 5			X	
Chỉnh sửa và viết báo cáo	X	X	X	X
Mức độ hoàn thành	100%	100%	100%	100%

CHƯƠNG VIII: TÀI LIỆU THAM KHẢO

[1] Kaggle, Intrusion Detection System NSL-KDD.

<https://www.kaggle.com/code/eneskosar19/intrusion-detection-system-nsl-kdd/notebook>

[2] GeeksforGeeks. (n.d.). Intrusion Detection System using Machine Learning Algorithms. <https://www.geeksforgeeks.org/intrusion-detection-system-using-machine-learning-algorithms/>

[3] GeeksforGeeks. (n.d.). Understanding Logistic Regression. <https://www.geeksforgeeks.org/understanding-logistic-regression/>

[4] Scholarhub. (n.d.). Random Forest. <https://scholarhub.vn/topic/random%20forest>

[5] GeeksforGeeks. (n.d.). Naive Bayes Classifiers. <https://www.geeksforgeeks.org/naive-bayes-classifiers/>