

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



NGUYỄN TRUNG KIÊN - 52100427  
TRẦN THANH THÀNH TÀI - 52100998

# **TIẾP CẬN CÁC PHƯƠNG PHÁP HỌC MÁY ĐỂ PHÁT HIỆN BOTNET ẨN TRONG LƯU LƯỢNG MẠNG**

## **DỰ ÁN CÔNG NGHỆ THÔNG TIN**

## **MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG DỮ LIỆU**

TP. HỒ CHÍ MINH, NĂM 2025

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



NGUYỄN TRUNG KIÊN - 52100427  
TRẦN THANH THÀNH TÀI - 52100998

# **TIẾP CẬN CÁC PHƯƠNG PHÁP HỌC MÁY ĐỂ PHÁT HIỆN BOTNET ẨN TRONG LƯU LƯỢNG MẠNG**

## **DỰ ÁN CÔNG NGHỆ THÔNG TIN**

## **MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG DỮ LIỆU**

Người hướng dẫn:  
**TS. Trương Đình Tú**

TP. HỒ CHÍ MINH, NĂM 2025

## LỜI CẢM ƠN

Lời đầu tiên, nhóm chúng em xin chân thành cảm ơn Khoa Công nghệ thông tin – Trường Đại học Tôn Đức Thắng đã tạo điều kiện cho chúng em được tham gia học phần Dự án Công nghệ thông tin. Đây là cơ hội quý báu giúp chúng em vận dụng kiến thức đã học vào thực tế, đồng thời rèn luyện kỹ năng làm việc nhóm và quản lý dự án.

Đặc biệt, chúng em xin bày tỏ lòng biết ơn sâu sắc đến thầy TS.Trương Đình Tú – giảng viên hướng dẫn Dự án Công nghệ thông tin. Thầy là người truyền đạt kiến thức mà còn là người định hướng, đồng hành và hỗ trợ chúng em trong suốt quá trình thực hiện dự án. Những chia sẻ và kinh nghiệm thực tiễn mà thầy mang lại đã giúp chúng em mở rộng tư duy, nhìn nhận vấn đề một cách toàn diện hơn. Chúng em rất trân trọng sự tận tụy và tâm huyết của thầy đối với sinh viên. Dưới sự hướng dẫn của thầy, chúng em đã phần nào hiểu rõ hơn về quy trình làm việc trong môi trường chuyên nghiệp, cũng như những yêu cầu thực tế trong lĩnh vực công nghệ thông tin hiện nay.

Dù đã nỗ lực hết mình, nhưng do còn hạn chế về kinh nghiệm nên nhóm không tránh khỏi những thiếu sót. Kính mong thầy góp ý thêm để nhóm có thể hoàn thiện hơn trong tương lai.

Một lần nữa, chúng em xin gửi lời cảm ơn chân thành và sâu sắc đến thầy Trương Đình Tú, cùng toàn thể quý thầy cô trong Khoa Công nghệ thông tin đã luôn tận tâm giảng dạy, hỗ trợ và tạo điều kiện để chúng em hoàn thành tốt dự án này.

Xin trân trọng cảm ơn!

TP. Hồ Chí Minh, ngày 30 tháng 04 năm 2025

**Tác giả**

(Ký tên và ghi rõ họ tên)

*Nguyễn Trung Kiên*

*Trần Thanh Thành Tài*

## **CÔNG TRÌNH ĐƯỢC HOÀN THÀNH TẠI ĐẠI HỌC TÔN ĐỨC THẮNG**

Chúng em xin cam đoan đây là công trình nghiên cứu của riêng chúng em và được sự hướng dẫn khoa học của **TS. Trương Đình Tú**. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

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

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

TP. Hồ Chí Minh, ngày 30 tháng 04 năm 2025

**Tác giả**

(Ký tên và ghi rõ họ tên)

*Nguyễn Trung Kiên*

*Trần Thanh Thành Tài*

## TÓM TẮT

Nội dung chính của báo cáo gồm:

- **Tổng quan về phát hiện botnet ẩn trong lưu lượng mạng bằng phương pháp học máy:**
  - Botnet là hệ thống mạng lưới các thiết bị bị nhiễm mã độc, thường được sử dụng để tấn công mạng, gây ra các mối đe dọa nghiêm trọng về bảo mật.
  - Các phương pháp học máy được áp dụng để phân tích lưu lượng mạng, phát hiện các mẫu bất thường và ngăn chặn Botnet hiệu quả.
- **Mục tiêu dự án:**
  - Nghiên cứu và đánh giá các phương pháp học máy để phát hiện lưu lượng Botnet.
  - Xây dựng mô hình có độ chính xác cao, giảm tỷ lệ dương tính giả và âm tính giả.
  - Giảm thiểu rủi ro và tăng độ tin cậy của hệ thống.
  - Tối ưu hóa hiệu suất mô hình để áp dụng trong thời gian thực.
  - Phân tích và so sánh hiệu quả của các thuật toán khác nhau.
  - Tích hợp khả năng giám sát và cảnh báo tự động.
- **Phân tích và thiết kế hệ thống:**
  - Dự án bắt đầu bằng việc thu thập và tiền xử lý dữ liệu lưu lượng mạng từ các nguồn công khai như CTU-13.
  - Phân tích đặc trưng để lựa chọn các thuộc tính quan trọng phục vụ cho việc huấn luyện mô hình.
  - Thiết kế kiến trúc hệ thống bao gồm các giai đoạn: thu thập dữ liệu, tiền xử lý, huấn luyện mô hình và triển khai.
- **Thực nghiệm:**
  - Thử nghiệm các thuật toán học máy như Decision Tree, Random Forest và MLP.
  - Đánh giá hiệu suất thông qua các chỉ số như accuracy, precision, recall, và F1-score.
  - Trình bày kết quả và so sánh hiệu quả giữa các phương pháp.

# MỤC LỤC

<b>TÓM TẮT</b> . . . . .	<b>iii</b>
<b>DANH MỤC HÌNH VẼ, BẢNG BIỂU</b> . . . . .	<b>vi</b>
<b>DANH MỤC CÁC CHỮ VIẾT TẮT</b> . . . . .	<b>viii</b>
<b>CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI</b> . . . . .	<b>1</b>
1.1 Bối cảnh nghiên cứu . . . . .	1
1.2 Lý do chọn đề tài . . . . .	1
1.3 Mục tiêu nghiên cứu . . . . .	2
1.4 Nội dung nghiên cứu . . . . .	2
1.5 Đối tượng, phạm vi nghiên cứu . . . . .	3
1.6 Phương pháp nghiên cứu . . . . .	3
1.7 Ý nghĩa lý luận và thực tiễn . . . . .	3
<b>CHƯƠNG 2. CƠ SỞ LÝ THUYẾT</b> . . . . .	<b>4</b>
2.1 Tổng quan về botnet . . . . .	4
2.1.1 Khái niệm về botnet . . . . .	4
2.1.2 Cách thức hoạt động của botnet . . . . .	5
2.1.3 Mục đích sử dụng botnet . . . . .	6
2.1.4 Các loại botnet thông dụng hiện nay . . . . .	9
2.1.5 Tác hại của botnet . . . . .	11
2.2 Một số phương pháp phát hiện botnet . . . . .	12
2.2.1 Phương pháp phát hiện thông thường . . . . .	12
2.2.2 Phương pháp phát hiện bằng học máy . . . . .	13
2.3 Mô hình Decision Tree . . . . .	18
2.3.1 Tổng quan . . . . .	18
2.3.2 Cách xây dựng thuật toán . . . . .	21
2.3.3 Ưu điểm và nhược điểm . . . . .	22
2.3.4 Ứng dụng mô hình . . . . .	23
2.4 Mô hình Random Forest . . . . .	24
2.4.1 Tổng quan . . . . .	24
2.4.2 Cách xây dựng mô hình . . . . .	25
2.4.3 Ưu điểm và nhược điểm . . . . .	26
2.4.4 Ứng dụng mô hình . . . . .	27

<b>CHƯƠNG 3. MÔ HÌNH ĐỀ XUẤT</b>	<b>29</b>
3.1 Những công trình nghiên cứu liên quan	29
3.2 Mô hình đề xuất	31
3.3 Hướng tiếp cận	32
<b>CHƯƠNG 4. THỰC NGHIỆM</b>	<b>33</b>
4.1 Tập dữ liệu CTU-13	33
4.2 Cài đặt thực nghiệm	39
4.3 Trích xuất đặc trưng	39
4.4 Lựa chọn đặc trưng	40
4.5 Số liệu đánh giá	41
4.6 Kết quả đạt được	42
4.6.1 Đối với mô hình <i>Decision Tree</i>	42
4.6.2 Đối với mô hình <i>Random Forest</i>	43
4.6.3 Đối với mô hình <i>MLP</i>	44
<b>CHƯƠNG 5. TỔNG KẾT</b>	<b>46</b>
5.1 Những kết quả đạt được	46
5.2 Thuận lợi và hạn chế	47
5.2.1 Thuận lợi	47
5.2.2 Hạn chế	47
5.3 Hướng phát triển	48
<b>TÀI LIỆU THAM KHẢO</b>	<b>49</b>

## DANH MỤC HÌNH VẼ, BẢNG BIỂU

### Danh mục hình vẽ

Hình 2.1	Mô hình hoạt động của botnet . . . . .	5
Hình 2.2	Quá trình hoạt động của botnet . . . . .	6
Hình 3.3	Sơ đồ quy trình phát hiện botnet . . . . .	31
Hình 4.4	Tần suất của các NetFlows theo thời gian . . . . .	35
Hình 4.5	Các giá trị số gói tin phổ biến nhất . . . . .	36
Hình 4.6	Các giá trị byte phổ biến nhất . . . . .	36
Hình 4.7	Các giá trị byte phổ biến nhất . . . . .	37
Hình 4.8	Biểu đồ phân bố của 113 nhãn . . . . .	37
Hình 4.9	Bản đồ nhiệt tương quan giữa các đặc trưng . . . . .	38
Hình 4.10	Biểu đồ phân tán của totpkts và totbytes . . . . .	39
Hình 4.11	Đánh giá AUC với mô hình Random Forest . . . . .	43
Hình 4.12	So sánh kết quả của hai mô hình . . . . .	44

**Danh mục bảng biểu**

Bảng 4.1	Bảng thông tin CTU-13 . . . . .	33
Bảng 4.2	Bảng thông tin của các đặc trưng . . . . .	34
Bảng 4.3	Đánh giá chỉ số AUC . . . . .	42
Bảng 4.4	Kết quả sau khi thực nghiệm mô hình Decision Tree . . . . .	42
Bảng 4.5	Báo cáo phân loại của mô hình Decision Tree . . . . .	42
Bảng 4.6	Kết quả sau khi thực nghiệm mô hình Random Forest . . . . .	43
Bảng 4.7	Báo cáo phân loại của mô hình Random Forest . . . . .	43
Bảng 4.8	Báo cáo phân loại của mô hình MLP . . . . .	45
Bảng 5.9	Bảng so sánh kết quả với các báo cáo liên quan . . . . .	46

## DANH MỤC CÁC CHỮ VIẾT TẮT

ADASYN[64]	Adaptive Synthetic Sampling [64]
AI	Artificial Intelligence
AWS	Amazon Web Service
AUC	Area Under the Curve
ADASYN[64]	Adaptive Synthetic Sampling [64]
CART	Classification and Regression Tree
C&C	Command and Control
CHAID	Chi-squared Automatic Interaction Detection
CPU	Central Processing Unit
CTU-13	Cyber Threat Unit 13
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DDoS	Distributed Denial of Service
DNS	Domain Name System
DTI	Debt-to-Income Ratio
FAR	False Alarm Ratio
GNN	Graph Neural Networks
GPU	Graphic Processing Unit
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Sercue
ID3	Iterative Dichotomiser 3
Iot	Internet of Thing
IRC	Internet Relay Chat
KNN	K-Nearest Neighbor
LDA	Linear Discriminant Analysis
LSTM	Long Short-Term Memory
MLP	Multi-Layer Perceptron
MRI	Magnetic Resonance Imaging
OOB	Out of Bag
PCA	Principal Component Analysis
P2P	Peer to Peer
RAM	Random Access Memory
ROC	Receiver Operating Characteristic
SMS	Short Message Service
SSH	Secure Shell
SVM	Super Vector Machine
TCP	Transmission Control Protocol
t-SNE	t-distributed Stochastic Neighbor Embedding
UDP	User Datagram Protocol
UNSW-NB15	University of New South Wales - Network Behavior 2015
VPN	Virtual Private Network

# CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI

## 1.1 Bối cảnh nghiên cứu

Trong những năm gần đây, sự phát triển mạnh mẽ của Internet và các hệ thống mạng máy tính đã mang lại nhiều tiện ích cho xã hội, đồng thời cũng kéo theo những thách thức lớn về an ninh mạng. Trong số các mối đe dọa an ninh mạng hiện nay, Botnet được xem là một trong những hình thức tấn công nguy hiểm và phổ biến nhất. Botnet là mạng lưới gồm hàng nghìn, thậm chí hàng triệu thiết bị bị nhiễm mã độc như máy tính, điện thoại, IoT được điều khiển từ xa bởi botmaster để thực hiện các hoạt động độc hại như tấn công DDoS, phát tán thư rác, đánh cắp dữ liệu nhạy cảm, hoặc khai thác tiền mã hóa.

Theo báo cáo của các tổ chức an ninh mạng toàn cầu như Kaspersky, Fortinet và Cisco, số lượng các cuộc tấn công Botnet đã tăng đáng kể trong thập kỷ qua, gây thiệt hại hàng tỷ USD mỗi năm cho các doanh nghiệp và chính phủ. Đặc biệt, với sự phát triển của IoT, nhiều thiết bị kém bảo mật đã trở thành mục tiêu dễ dàng để hình thành các mạng Botnet quy mô lớn.

Các phương pháp phát hiện Botnet truyền thống chủ yếu dựa trên signature-based detection (phát hiện dựa trên mẫu nhận dạng) hoặc anomaly-based detection (phát hiện bất thường). Tuy nhiên, các phương pháp này có nhiều hạn chế:

- Signature-based chỉ phát hiện được các mẫu đã biết trước, trong khi Botnet liên tục biến đổi và sử dụng kỹ thuật mã hóa, làm giảm hiệu quả phát hiện.
- Anomaly-based có thể tạo ra nhiều cảnh báo giả (false positives), gây khó khăn cho việc phân tích thủ công.

Do đó, việc nghiên cứu và ứng dụng các phương pháp học máy vào phát hiện Botnet đang trở thành xu hướng quan trọng trong lĩnh vực an ninh mạng. Các thuật toán học máy có khả năng phân tích lượng lớn dữ liệu mạng, phát hiện các mẫu hành vi bất thường mà phương pháp truyền thống không thể nhận diện được.

## 1.2 Lý do chọn đề tài

Trong bối cảnh an ninh mạng ngày càng trở nên phức tạp, việc phát hiện và ngăn chặn các mối đe dọa từ Botnet đang trở thành thách thức lớn đối với các tổ chức và doanh nghiệp. Botnet không chỉ gây ra những thiệt hại nghiêm trọng về kinh tế mà còn đe dọa đến sự ổn định của hệ thống mạng toàn cầu. Các phương pháp phát hiện truyền thống tỏ ra kém hiệu quả trước sự biến đổi không ngừng của các mạng Botnet hiện đại, đòi hỏi phải có những giải pháp mới, thông minh và linh hoạt hơn.

Học máy đã chứng minh được tiềm năng to lớn trong việc giải quyết các bài toán phức tạp liên quan đến an ninh mạng. Khả năng phân tích lượng dữ liệu khổng lồ, phát hiện các mẫu bất thường và dự đoán các mối đe dọa tiềm ẩn khiến học máy trở thành công cụ lý tưởng để

đối phó với Botnet. Đặc biệt, với sự phát triển của các thuật toán học sâu, việc phát hiện các cuộc tấn công tinh vi trở nên khả thi hơn bao giờ hết.

Một trong những động lực chính thúc đẩy việc lựa chọn đề tài này là tính ứng dụng thực tiễn cao. Kết quả nghiên cứu không chỉ dừng lại ở mức lý thuyết mà có thể triển khai vào các hệ thống bảo mật thực tế, giúp các tổ chức chủ động phòng ngừa và giảm thiểu rủi ro. Hơn nữa, đề tài mở ra cơ hội để khám phá sâu hơn về khả năng kết hợp giữa học máy và các công nghệ an ninh mạng tiên tiến khác, từ đó xây dựng các giải pháp toàn diện hơn.

Bên cạnh đó, nghiên cứu này cũng đáp ứng nhu cầu cấp thiết về nguồn nhân lực chất lượng cao trong lĩnh vực an ninh mạng. Việc phát triển và hoàn thiện các mô hình học máy để phát hiện Botnet không chỉ góp phần nâng cao hiệu quả bảo mật mà còn tạo tiền đề cho những đổi mới công nghệ trong tương lai. Đây chính là lý do vì sao đề tài này không chỉ có ý nghĩa học thuật mà còn mang giá trị thực tiễn sâu sắc.

### 1.3 Mục tiêu nghiên cứu

- **Mục tiêu tổng quát:** Nghiên cứu và ứng dụng các phương pháp học máy để phát hiện lưu lượng Botnet một cách hiệu quả.
- **Mục tiêu cụ thể:**
  - Tìm hiểu các kỹ thuật học máy phù hợp để phân tích lưu lượng mạng.
  - Xây dựng mô hình có độ chính xác cao, giảm tỷ lệ false positive và false negative.
  - Đánh giá hiệu suất của các thuật toán khác nhau như Random Forest, SVM, Neural Networks,...
  - Phát triển giải pháp có khả năng giám sát và cảnh báo tự động.

### 1.4 Nội dung nghiên cứu

- Tổng quan về Botnet, các kỹ thuật tấn công và phương pháp phát hiện truyền thống.
- Các thuật toán học máy ứng dụng trong phát hiện lưu lượng bất thường.
- Thu thập và tiền xử lý dữ liệu từ các bộ dataset công khai như CTU-13, UNSW-NB15,...
- Phân tích đặc trưng để chọn lọc thuộc tính quan trọng.
- Huấn luyện và đánh giá các mô hình học máy.
- So sánh hiệu quả giữa các phương pháp.
- Xây dựng hệ thống giám sát và cảnh báo tự động.

### 1.5 Đối tượng, phạm vi nghiên cứu

- **Đối tượng:** Lưu lượng mạng bất thường do Botnet tạo ra.
- **Phạm vi:** Tập trung vào các phương pháp học máy, đặc biệt như Supervised và Unsupervised Learning. Sử dụng các bộ dữ liệu mạng công khai để huấn luyện và kiểm thử. Giới hạn trong môi trường mô phỏng và thử nghiệm lab trước khi áp dụng thực tế.

### 1.6 Phương pháp nghiên cứu

- **Nghiên cứu lý thuyết:**
  - Tiến hành tổng hợp và phân tích sâu các tài liệu học thuật, báo cáo chuyên ngành liên quan đến kiến trúc Botnet hiện đại, các phương pháp phát hiện truyền thống và xu hướng ứng dụng trí tuệ nhân tạo trong an ninh mạng.
  - Quá trình này sử dụng phương pháp phân tích tổng hợp để đánh giá toàn diện ưu nhược điểm của từng phương pháp, từ đó xác định hướng tiếp cận tối ưu cho nghiên cứu.
- **Nghiên cứu thực nghiệm:**
  - Triển khai theo quy trình khoa học bài bản bắt đầu từ việc thu thập và xử lý các bộ dữ liệu mẫu chuẩn như CTU-13 và UNSW-NB15. Áp dụng các kỹ thuật làm sạch dữ liệu để xử lý các giá trị thiếu và ngoại lai, đồng thời thực hiện chuẩn hóa dữ liệu và trích chọn đặc trưng quan trọng.
  - Trong giai đoạn xây dựng mô hình, nhiều thuật toán tiên tiến như Random Forest, SVM và Neural Networks được triển khai với các kỹ thuật tối ưu hóa tham số và phương pháp ensemble learning để nâng cao hiệu suất. Quá trình đánh giá được thực hiện nghiêm ngặt thông qua đa dạng các chỉ số và phân tích ma trận nhầm lẫn, đảm bảo tính khách quan và toàn diện.

### 1.7 Ý nghĩa lý luận và thực tiễn

#### Ý nghĩa lý luận

- Góp phần phát triển phương pháp phát hiện Botnet dựa trên học máy.
- Cung cấp cơ sở khoa học để so sánh hiệu quả của các thuật toán khác nhau.

#### Ý nghĩa thực tiễn

- Xây dựng giải pháp có thể tích hợp vào hệ thống bảo mật thực tế.
- Giúp các tổ chức chủ động phòng chống tấn công Botnet, giảm thiểu rủi ro an ninh mạng.

## CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

### 2.1 Tổng quan về botnet

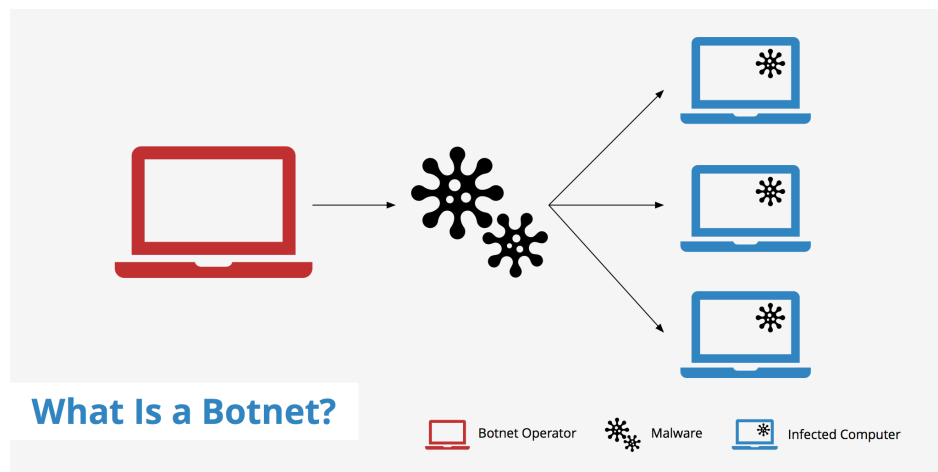
#### 2.1.1 Khái niệm về botnet

Botnet là một mạng lưới các thiết bị điện tử thường là máy tính, điện thoại thông minh hoặc thiết bị IoT bị nhiễm phần mềm độc hại và được kiểm soát từ xa bởi một tin tặc hoặc một nhóm tin tặc mà không có sự cho phép của chủ sở hữu thiết bị. Các thiết bị này được gọi là "bot" hoặc "zombie", hoạt động như một đội quân kỹ thuật số thực hiện các lệnh từ xa do kẻ tấn công được gọi là botmaster đưa ra bằng một cơ chế điều khiển và kiểm soát C&C.

Botnet được xây dựng dựa trên một hệ thống phân tán, trong đó các thành phần chính bao gồm:

- **Bot thiết bị bị nhiễm:** Đây là các thiết bị bị lây nhiễm phần mềm độc hại chẳng hạn như virus, trojan, worm, hoặc ransomware. Một khi bị nhiễm, thiết bị sẽ trở thành một phần của botnet và nhận lệnh từ botmaster.
- **Botmaster (Kẻ điều khiển):** Cá nhân hoặc nhóm đứng sau botnet chịu trách nhiệm phát tán phần mềm độc hại, quản lý botnet, và ra lệnh cho các bot.
- **Hệ thống C&C:** Đây là cơ sở hạ tầng liên lạc giữa botmaster và các bot. C&C có thể là một máy chủ tập trung, mạng ngang hàng (P2P), hoặc sử dụng các giao thức phi tập trung (như blockchain hoặc mạng xã hội) để gửi lệnh. Các giao thức phổ biến bao gồm IRC, HTTP hoặc DNS.
- **Phần mềm độc hại:** Các chương trình độc hại như Zeus, Mirai hoặc Conficker được sử dụng để lây nhiễm và kiểm soát thiết bị. Phần mềm này thường ẩn mình trong hệ thống khó bị phát hiện bởi phần mềm diệt virus.

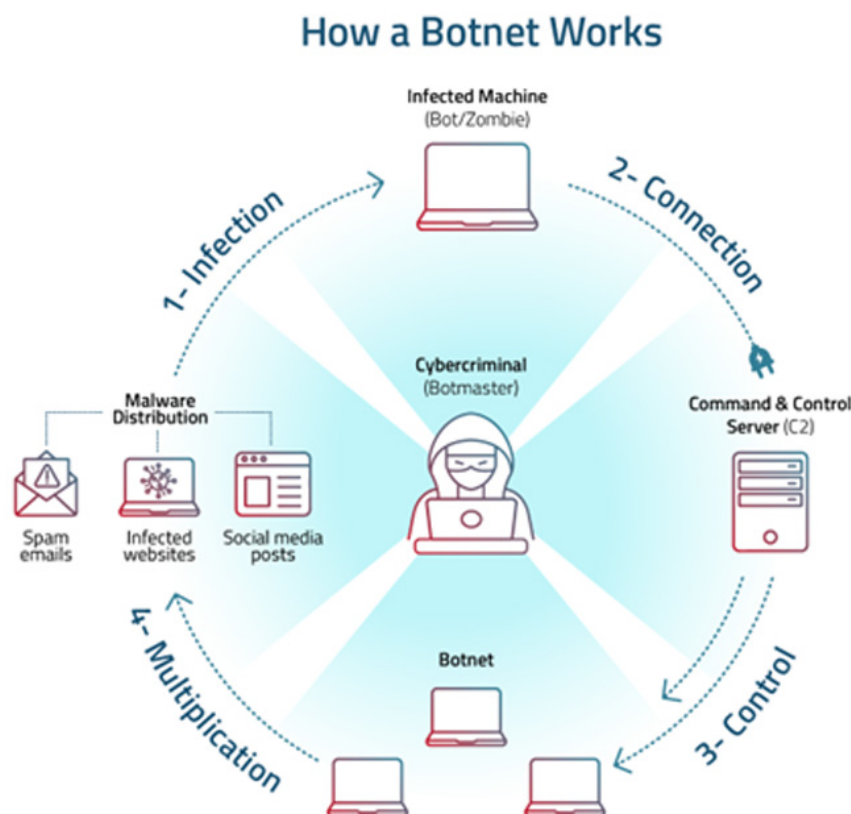
Botnet có thể bao gồm hàng trăm, hàng nghìn, thậm chí hàng triệu thiết bị tùy thuộc vào quy mô và mục đích của kẻ tấn công.



Hình 2.1: Mô hình hoạt động của botnet

### 2.1.2 Cách thức hoạt động của botnet

- **Tiến hành lây nhiễm vào thiết bị:**
  - Tin tặc sử dụng các phương pháp như email lừa đảo, tải xuống phần mềm độc hại từ các trang web không an toàn, khai thác lỗ hổng phần mềm (zero-day exploits) hoặc phát tán qua mạng xã hội.
  - Các thiết bị IoT như camera giám sát, bộ định tuyến thường là mục tiêu dễ bị tấn công do mật khẩu yếu hoặc thiếu cập nhật bảo mật.
- **Thiết lập kết nối với C&C:** Sau khi bị nhiễm, thiết bị sẽ liên lạc với máy chủ C&C để nhận lệnh. Botmaster có thể sử dụng các kỹ thuật như mã hóa hoặc ẩn danh thông qua Tor hoặc VPN để che giấu hoạt động.
- **Mở rộng mạng lưới:**
  - Một số botnet tự động lây lan, có thể thông qua worm, để tuyển mộ thêm các thiết bị khác vào mạng lưới.
  - Các botnet lớn như Mirai từng lây nhiễm hàng triệu thiết bị IoT bằng cách quét và tấn công các thiết bị có mật khẩu mặc định.
- **Vận hành và quản lý:** Botmaster sử dụng giao diện điều khiển để gửi lệnh đến các bot, chẳng hạn như thực hiện tấn công DDoS, thu thập dữ liệu, hoặc phát tán thêm mã độc.



Hình 2.2: Quá trình hoạt động của botnet

Hình 2.2 minh họa chi tiết về cách hoạt động của một botnet, một mạng lưới máy tính bị nhiễm mã độc được điều khiển bởi botmaster. Quá trình bắt đầu với giai đoạn lây nhiễm khi máy tính trở thành "zombie" thông qua các phương thức như email chứa mã độc, truy cập vào các trang web bị nhiễm hoặc tương tác với các bài đăng độc hại trên mạng xã hội. Tiếp theo là giai đoạn kết nối, khi máy bị nhiễm thiết lập liên lạc với máy chủ điều khiển của botmaster thường thông qua các kênh ẩn hoặc mã hóa để tránh bị phát hiện. Giai đoạn kiểm soát diễn ra khi botmaster sử dụng máy chủ C&C để gửi chỉ thị như thực hiện tấn công DDoS, gửi thư rác hoặc khai thác dữ liệu. Cuối cùng, giai đoạn nhân rộng xảy ra khi các máy bị nhiễm tiếp tục lây lan mã độc sang các thiết bị khác và mở rộng quy mô botnet.

### 2.1.3 Mục đích sử dụng botnet

- **Tấn công DDoS:** Botnet được dùng để thực hiện tấn công DDoS bằng cách huy động hàng nghìn hoặc thậm chí hàng triệu thiết bị bị nhiễm để gửi một lượng lớn lưu lượng truy cập giả mạo đến một máy chủ hoặc trang web mục tiêu làm quá tải hệ thống và gây tê liệt dịch vụ. Các cuộc tấn công này thường nhắm vào các nền tảng quan trọng như trang web thương mại điện tử, dịch vụ ngân hàng hoặc nhà cung cấp DNS nhằm gây gián đoạn hoạt động kinh doanh hoặc dịch vụ trực tuyến. Loại tấn công này không

chỉ gây thiệt hại tài chính mà còn làm tổn hại uy tín của các tổ chức bị nhắm mục tiêu khiến việc phòng chống DDoS trở thành ưu tiên hàng đầu trong an ninh mạng.

- **Khai thác tiền điện tử (Cryptojacking):** Botnet được sử dụng trong các hoạt động khai thác tiền điện tử bằng cách tận dụng tài nguyên tính toán của các thiết bị bị nhiễm như CPU hoặc GPU để đào các loại tiền điện tử như Bitcoin, Monero hoặc Ethereum mà không có sự đồng ý của chủ sở hữu thiết bị. Các botnet này thường cài đặt phần mềm khai thác lên máy tính, máy chủ hoặc thiết bị IoT sau đó sử dụng sức mạnh tính toán của chúng để giải các bài toán phức tạp tạo ra lợi nhuận cho botmaster. Cryptojacking không chỉ làm chậm thiết bị bị nhiễm mà còn tăng chi phí điện năng cho nạn nhân và đồng thời khó phát hiện vì hoạt động khai thác thường diễn ra âm thầm không gây gián đoạn rõ rệt.
- **Phát tán thư rác (Spam):** Botnet thường được sử dụng để phát tán thư rác qua email hoặc tin nhắn gửi hàng loạt thông điệp lừa đảo nhằm phát tán thêm mã độc dụ dỗ người dùng nhấp vào liên kết độc hại hoặc đánh cắp thông tin cá nhân. Các botnet này có khả năng gửi hàng triệu email mỗi ngày thường chứa tệp đính kèm độc hại hoặc liên kết dẫn đến trang web giả mạo, từ đó mở rộng mạng lưới lây nhiễm hoặc thực hiện các cuộc lừa đảo. Việc phát tán spam không chỉ gây phiền hà cho người dùng mà còn tạo điều kiện cho các cuộc tấn công tiếp theo, như đánh cắp thông tin hoặc phát tán ransomware, làm tăng nguy cơ an ninh mạng trên toàn cầu.
- **Đánh cắp dữ liệu:** Botnet được thiết kế để thu thập thông tin nhạy cảm từ các thiết bị bị nhiễm như thông tin đăng nhập (tài khoản ngân hàng, email), dữ liệu tài chính (số thẻ tín dụng), hoặc thông tin cá nhân (họ tên, địa chỉ) sau đó chuyển dữ liệu này về máy chủ C&C của botmaster. Các botnet này thường sử dụng các kỹ thuật như cài đặt Trojan, keylogging hoặc khai thác lỗ hổng để truy cập dữ liệu, sau đó bán thông tin trên dark web hoặc sử dụng để tống tiền nạn nhân. Việc đánh cắp dữ liệu không chỉ gây tổn thất tài chính mà còn làm tổn hại đến quyền riêng tư của người dùng đòi hỏi các biện pháp bảo mật mạnh mẽ hơn để bảo vệ thông tin cá nhân.
- **Tấn công ransomware:** Botnet dùng để phân phối phần mềm tống tiền, mã hóa dữ liệu trên thiết bị của nạn nhân và yêu cầu tiền chuộc bằng Bitcoin để khôi phục quyền truy cập. Các botnet này thường phát tán ransomware qua email lừa đảo, tệp đính kèm độc hại hoặc khai thác lỗ hổng hệ điều hành sau đó mã hóa các tệp quan trọng như tài liệu, ảnh, hoặc cơ sở dữ liệu của doanh nghiệp. Các cuộc tấn công ransomware không chỉ gây gián đoạn hoạt động mà còn tạo áp lực tài chính lớn cho nạn nhân đặc biệt khi họ không có bản sao lưu dữ liệu làm tăng nhu cầu phòng chống botnet trong an ninh mạng.
- **Gián điệp và giám sát:** Botnet có thể được sử dụng để thực hiện các hoạt động gián điệp và giám sát, ghi lại thao tác bàn phím, chụp ảnh màn hình, hoặc thậm chí kích hoạt

webcam và micro trên thiết bị bị nhiễm để theo dõi nạn nhân mà không bị phát hiện. Loại botnet này thu thập thông tin bí mật từ cá nhân, doanh nghiệp và tổ chức chính phủ sau đó sử dụng dữ liệu này để tống tiền, bán trên dark web hoặc hỗ trợ những cuộc tấn công lớn hơn. Các hoạt động gián điệp không chỉ xâm phạm quyền riêng tư mà còn gây nguy cơ an ninh quốc gia, đòi hỏi các biện pháp bảo mật tiên tiến để phát hiện và ngăn chặn.

- **Proxy ẩn danh:** Botnet có thể được sử dụng để tạo mạng proxy ẩn danh, che giấu danh tính của tin tặc khi thực hiện các hoạt động bất hợp pháp như tấn công mạng, mua bán dữ liệu trên dark web hoặc truy cập các trang web bị cấm. Các botnet này biến các thiết bị bị nhiễm thành các nút proxy cho phép botmaster định tuyến lưu lượng mạng qua nhiều địa chỉ IP khác nhau khiến việc truy vết trở nên cực kỳ khó khăn. Việc sử dụng proxy ẩn danh không chỉ hỗ trợ các hoạt động phạm tội mà còn làm phức tạp quá trình điều tra của cơ quan thực thi pháp luật, đòi hỏi sự hợp tác quốc tế để triệt phá. khác.

#### 2.1.4 Các loại botnet thông dụng hiện nay

Botnet được phân loại dựa trên hai tiêu chí chính là cấu trúc hoạt động của hệ thống C&C hoặc mục đích sử dụng cụ thể mà kẻ tấn công hướng tới cho phép phân loại botnet thành các nhóm khác nhau dựa trên cách chúng được tổ chức, vận hành, cũng như các hoạt động bất hợp pháp mà chúng được triển khai để thực hiện, từ đó giúp các chuyên gia bảo mật hiểu rõ hơn về cách thức hoạt động và phát triển các biện pháp đối phó hiệu quả.

- **Phân loại dựa trên cấu trúc máy chủ C&C:**

- **Botnet tập trung:** Botnet tập trung vận hành với một máy chủ C&C duy nhất đóng vai trò trung tâm để các bot kết nối và nhận lệnh, thường thông qua các giao thức phổ biến như IRC hoặc HTTP. Kết cấu này cho phép botmaster quản lý dễ dàng, truyền tải lệnh nhanh chóng đến hàng ngàn bot trong thời gian thực, rất phù hợp cho các cuộc tấn công cần phối hợp chặt chẽ như phát tán spam hoặc đánh cắp thông tin. Tuy nhiên, nhược điểm lớn nhất là tính dễ bị triệt phá, vì nếu máy chủ C&C dễ bị phát hiện và vô hiệu hóa bằng cách chặn IP hoặc tấn công ngược và toàn bộ mạng botnet có thể sụp đổ.
- **Botnet ngang hàng (P2P):** Botnet ngang hàng loại bỏ nhu cầu về máy chủ C&C. Thay vào đó các bot giao tiếp trực tiếp với nhau thông qua giao thức P2P, tạo ra một mạng lưới phân tán. Cơ chế này giúp botnet khó bị triệt phá hơn vì không có điểm yếu duy nhất để tấn công và khả năng phân tán tốt cho phép botnet duy trì hoạt động ngay cả khi một số nút bị vô hiệu hóa. Tuy nhiên, nhược điểm là việc quản lý trở nên phức tạp hơn đối với botmaster, và tốc độ truyền lệnh chậm hơn do phụ thuộc vào việc đồng bộ hóa giữa các bot.
- **Botnet lai:** Botnet lai kết hợp ưu điểm của cả botnet tập trung và P2P sử dụng các “siêu bot” làm trung gian để kết nối botmaster với các bot thông thường tạo ra một cấu trúc linh hoạt và bền vững. Các siêu bot đóng vai trò trung tâm cục bộ nhận lệnh từ botmaster và phân phối đến các bot khác thông qua giao thức P2P giúp tăng khả năng chống chịu khi bị tấn công và khó phát hiện hơn do không phụ thuộc hoàn toàn vào một máy chủ duy nhất. Tuy nhiên, việc triển khai botnet lai đòi hỏi thiết

kế phức tạp, đòi hỏi botmaster phải quản lý cả mạng lưới siêu bot và bot thường làm tăng độ khó trong vận hành.

- **Phân loại dựa trên loại thiết bị bị nhiễm:**

- **Botnet máy tính (PC-based):** Botnet máy tính thường tập trung vào các máy tính cá nhân chạy các hệ điều hành phổ biến như Windows, macOS, hoặc Linux bị lây nhiễm thông qua các vector tấn công như email lừa đảo chứa tệp đính kèm độc hại, phần mềm giả mạo hoặc khai thác lỗ hổng hệ điều hành chưa được vá. Loại botnet này khai thác sức mạnh tính toán và kết nối mạng của máy tính để thực hiện các nhiệm vụ như phát tán spam, đánh cắp thông tin cá nhân hoặc tấn công DDoS.
- **Botnet thiết bị IoT (IoT-based):** Botnet thiết bị IoT nhắm vào các thiết bị kết nối mạng như camera giám sát, bộ định tuyến, đèn thông minh hoặc các thiết bị gia dụng thông minh, vốn thường có bảo mật yếu hoặc thiếu cập nhật phần mềm. Lây nhiễm xảy ra chủ yếu do mật khẩu mặc định không được thay đổi, phần mềm firmware lỗi thời, hoặc khai thác các giao thức mạng như Telnet và SSH. Loại botnet này được sử dụng để thực hiện các cuộc tấn công DDoS quy mô lớn nhờ số lượng thiết bị lớn, hoặc khai thác dữ liệu từ các thiết bị bị nhiễm.
- **Botnet điện thoại di động (Mobile-based):** nhắm vào các thiết bị thông minh như smartphone hoặc tablet chủ yếu lây nhiễm qua các ứng dụng độc hại tải từ cửa hàng không chính thức, tin nhắn SMS lừa đảo chứa liên kết độc hại hoặc khai thác lỗ hổng trong hệ điều hành Android và iOS. Loại botnet này thường được thiết kế để đánh cắp dữ liệu cá nhân như danh bạ, tin nhắn, thông tin tài khoản, gửi tin nhắn rác với chi phí cao hoặc thực hiện các cuộc tấn công nhỏ lẻ nhắm vào người dùng cá nhân.
- **Botnet máy chủ (Server-based):** nhắm vào các máy chủ doanh nghiệp, trung tâm dữ liệu hoặc dịch vụ đám mây tận dụng sức mạnh tính toán lớn và băng thông cao của các hệ thống này để thực hiện các cuộc tấn công quy mô lớn đặc biệt là DDoS hoặc khai thác tài nguyên tính toán cho cryptojacking. Các máy chủ thường bị lây nhiễm thông qua các lỗ hổng quản lý như cấu hình sai, phần mềm không được cập nhật hoặc tấn công brute-force vào mật khẩu yếu, cho phép botnet kiểm soát các hệ thống quan trọng. Loại botnet này đặc biệt nguy hiểm vì có thể gây gián đoạn dịch vụ quan trọng như các cuộc tấn công DDoS nhắm vào các trang web thương mại điện tử hoặc dịch vụ đám mây như AWS và Google Cloud.

- **Phân loại dựa trên mức độ tự động hóa:**

- **Botnet thủ công:** botmaster trực tiếp ra lệnh cho các bot thông qua máy chủ C&C thường được sử dụng cho các chiến dịch ngắn hạn hoặc nhắm vào mục tiêu cụ thể như tấn công một trang web doanh nghiệp hoặc thực hiện thử nghiệm bảo mật.

Trong mô hình này, botmaster phải can thiệp thường xuyên để cập nhật lệnh, điều chỉnh chiến lược tấn công hoặc phản ứng với các biện pháp phòng thủ từ phía nạn nhân khiến nó ít hiệu quả với các cuộc tấn công quy mô lớn hoặc kéo dài.

- **Botnet bán tự động:** botnet bán tự động kết hợp cả sự can thiệp của botmaster và khả năng tự động thực hiện một số tác vụ cơ bản như phát tán spam, tải xuống phần mềm độc hại bổ sung hoặc thực hiện các cuộc tấn công DDoS nhỏ lẻ nhằm vào nhiều mục tiêu cùng lúc. Trong mô hình này, botmaster thiết lập các kịch bản tự động để bot thực hiện, chẳng hạn như gửi hàng loạt email chứa malware, nhưng vẫn cần can thiệp định kỳ để thay đổi mục tiêu, cập nhật phần mềm độc hại hoặc điều chỉnh chiến lược khi bị phát hiện. Một botnet bán tự động chuyên phát tán spam qua email đòi hỏi botmaster điều chỉnh định kỳ để tránh bị các bộ lọc spam chặn cho thấy sự phụ thuộc vào sự giám sát của con người trong các giai đoạn quan trọng.
- **Botnet tự động:** Botnet tự động hoạt động gần như độc lập nhờ tích hợp các thuật toán thông minh hoặc AI, cho phép bot tự tìm kiếm mục tiêu mới, lây lan sang các thiết bị chưa bị nhiễm và thực hiện tấn công mà không cần can thiệp liên tục bởi botmaster. Loại botnet này sử dụng các kỹ thuật như học máy để phân tích hành vi mạng, tự điều chỉnh chiến lược dựa trên phản ứng của nạn nhân và khai thác lỗ hổng hệ thống theo thời gian thực, làm tăng hiệu quả và độ khó phát hiện. Tuy nhiên, sự phức tạp của AI cũng khiến botmaster khó kiểm soát hoàn toàn, dẫn đến rủi ro bị lạm dụng hoặc phát hiện do hành vi bất thường vượt ngoài dự đoán.

### 2.1.5 Tác hại của botnet

#### • Đối với cá nhân:

- Botnet gây ra nhiều vấn đề nghiêm trọng cho người dùng cá nhân. Chúng có thể đánh cắp thông tin nhạy cảm như mật khẩu, thông tin ngân hàng, hoặc dữ liệu cá nhân thông qua các kỹ thuật như keylogging hoặc chụp màn hình. Nạn nhân có nguy cơ mất tiền do lừa đảo tài chính, bị khóa dữ liệu bởi ransomware hoặc bị sử dụng tài nguyên thiết bị để đào tiền điện tử (cryptojacking).
- Ngoài ra, thiết bị bị nhiễm botnet thường chạy chậm, tiêu tốn CPU, RAM, hoặc băng thông làm giảm hiệu suất. Người dùng cũng phải chịu chi phí sửa chữa, khôi phục hệ thống hoặc mua phần mềm bảo mật mới. Hơn nữa, botnet có thể xâm phạm quyền riêng tư bằng cách kích hoạt webcam, ghi âm hoặc theo dõi hoạt động gây cảm giác bất an.

#### • Đối với doanh nghiệp:

- Đối với doanh nghiệp, botnet là mối đe dọa lớn đến hoạt động và uy tín. Các cuộc

tấn công DDoS từ botnet có thể làm tê liệt trang web, ứng dụng hoặc máy chủ dẫn đến gián đoạn kinh doanh và mất doanh thu.

- Botnet còn đánh cắp dữ liệu khách hàng hoặc thông tin nội bộ gây tổn hại danh tiếng và niềm tin. Doanh nghiệp phải đối mặt với chi phí lớn để khắc phục, triển khai hệ thống bảo mật mới hoặc trả tiền chuộc trong trường hợp ransomware. Ngoài ra, botnet chiếm dụng tài nguyên mạng hoặc máy chủ làm tăng chi phí vận hành như điện năng và băng thông. Nếu dữ liệu khách hàng bị rò rỉ, doanh nghiệp có thể vi phạm pháp lý dẫn đến kiện tụng hoặc phạt nặng.

- **Đối với xã hội:**

- Ở cấp độ xã hội, botnet gây ra những hậu quả nghiêm trọng hơn. Chúng có thể tấn công những hệ thống quan trọng như hệ thống y tế, giao thông hoặc chính phủ, đe dọa an ninh và trật tự xã hội. Ví dụ, một cuộc tấn công DDoS vào bệnh viện có thể làm gián đoạn chăm sóc y tế hoặc vào hệ thống bầu cử có thể gây bất ổn chính trị.
- Botnet còn thúc đẩy tội phạm mạng bằng cách hỗ trợ lừa đảo, rửa tiền hoặc buôn bán dữ liệu trên dark web. Các tổ chức và quốc gia phải chi hàng tỷ USD mỗi năm để chống lại botnet làm tăng gánh nặng kinh tế toàn cầu. Hơn nữa, các cuộc tấn công quy mô lớn làm giảm niềm tin vào thương mại điện tử và công nghệ số ảnh hưởng đến sự phát triển kinh tế trực tuyến.

## 2.2 Một số phương pháp phát hiện botnet

### 2.2.1 Phương pháp phát hiện thông thường

- **Phương pháp phát hiện dựa trên bất thường (Anomaly-based detection):** Phương pháp này giả định rằng mô hình giao tiếp của lưu lượng bot khác với lưu lượng không phải bot. Tuy nhiên, để tránh bị phát hiện, nhiều biến thể bot hiện đại có thể bắt chước mô hình giao tiếp của các máy chủ lành tính.
- **Phương pháp phát hiện dựa trên chữ ký (Signature-based detection):** Phương pháp này hiệu quả đối với các biến thể bot đã biết nhưng không hiệu quả với các biến thể bot chưa biết. Phương pháp này yêu cầu phải hiểu biết chi tiết về đặc điểm của lưu lượng bot. Bất kỳ đặc điểm mới hoặc chưa biết nào cũng sẽ qua mặt được phát hiện. Hầu hết các biến thể botnet mới sử dụng mã hóa khiến phương pháp dựa trên chữ ký hoàn toàn không hiệu quả.
- **Phương pháp phát hiện dựa trên honeypot (Honeypot-based detection):** Honeypot là các máy chủ hoặc thiết bị giả lập được thiết kế để thu hút botnet lây nhiễm. Khi botnet tấn công honeypot, các hoạt động như phương thức lây lan, địa chỉ C&C hoặc mã độc được ghi lại để phân tích. Honeypot có thể là thiết bị IoT giả lập hoặc máy chủ web để

bị tấn công. Phương pháp này cũng có thể phát hiện các botnet đã biết nhưng không hiệu quả đối với các botnet mới hoặc chưa biết.

- **Phương pháp phát hiện dựa trên nút (Node-based detection):** Phương pháp này hoạt động trên máy tính chủ. Nó cố gắng xác định phần mềm độc hại bot hoạt động trên máy tính bị xâm nhập bằng cách kiểm tra các dấu vết cấp độ máy khách như: lệnh gọi API, thay đổi tệp, nhật ký ứng dụng và hệ thống, tiến trình hoạt động, nhật ký phím, mức sử dụng tài nguyên,... Để tránh bị phát hiện, tin tặc cố gắng che giấu hoạt động của bot và sử dụng nhiều kỹ thuật kiên cường cấp máy khách như khả năng rootkit và mã hóa mã nguồn khiến việc phát hiện dựa trên nút trở nên khó khăn.
- **Phương pháp phát hiện dựa trên cộng đồng (Community-based detection):** Phương pháp này yêu cầu một đồ thị hoàn chỉnh về giao tiếp của bot. Tuy nhiên, rất khó để có được đồ thị giao tiếp hoàn chỉnh vì tin tặc liên tục cập nhật giao tiếp của botnet.
- **Phương pháp phát hiện dựa trên cấu trúc hoặc giao thức (Structure or protocol-based detection):** Phương pháp này dựa trên một giao thức cụ thể. Nhưng các botnet hiện đại rất linh hoạt và sử dụng nhiều giao thức khác ngoài TCP hoặc UDP để liên lạc. Do đó, phương pháp này cũng không hiệu quả.

Nhìn chung, những phương pháp phát hiện botnet thông thường chỉ cung cấp các giải pháp đơn giản, dễ triển khai và hiệu quả với botnet đã biết hoặc có hành vi rõ ràng. Nhưng những kỹ thuật này dễ bị vượt qua bởi botnet hiện đại sử dụng mã hóa, giao thức linh hoạt hoặc kỹ thuật che giấu như rootkit. Nó gây mất nhiều thời gian và kém hiệu quả với botnet phức tạp như P2P.

## 2.2.2 Phương pháp phát hiện bằng học máy

### 2.2.2.1 Định nghĩa học máy trong phát hiện botnet

Học máy (Machine Learning) là một nhánh của AI, tập trung vào việc phát triển các thuật toán và mô hình cho phép máy tính học hỏi và đưa ra quyết định hoặc dự đoán mà không cần được lập trình rõ ràng cho từng tác vụ. Thay vì dựa vào các quy tắc cố định, học máy sử dụng dữ liệu để “học” các mẫu, mối quan hệ, hoặc đặc trưng, từ đó cải thiện hiệu suất theo thời gian. Học máy là một kỹ thuật mạnh mẽ được áp dụng rộng rãi trong nhiều lĩnh vực như y tế, tài chính, giao thông và đặc biệt là an ninh mạng nhờ khả năng xử lý dữ liệu phức tạp và phát hiện các mẫu bất thường.

Cốt lõi của học máy là khả năng học từ kinh nghiệm (dữ liệu quá khứ) để dự đoán hoặc phân loại dữ liệu mới. Ví dụ, trong phát hiện botnet, học máy có thể phân biệt lưu lượng mạng của botnet như lưu lượng DDoS hoặc truy vấn DNS động với lưu lượng hợp pháp bằng cách phân tích các đặc trưng như tần suất gói tin, kích thước gói hoặc thời gian kết nối. Sự linh hoạt và khả năng thích nghi của học máy giúp nó trở thành công cụ lý tưởng để đối phó

với các mối đe dọa mạng ngày càng tinh vi, như botnet sử dụng mã hóa hoặc kỹ thuật che giấu.

#### 2.2.2.2 Các loại học máy

- **Học có giám sát (Supervised Learning):** Mô hình được huấn luyện trên dữ liệu đã gắn nhãn ví dụ như lưu lượng mạng được gắn nhãn là “botnet” hoặc “lành tính”. Các mô hình như SVM, Random Forest hoặc Neural Networks dự đoán nhãn cho dữ liệu mới dựa trên mẫu đã học. Trong phát hiện botnet, học có giám sát được dùng để phân loại lưu lượng hoặc nhận diện các mẫu email lừa đảo. Nhưng việc học có giám sát yêu cầu dữ liệu có nhãn lớn, phải trích xuất đặc trưng và dễ bị overfitting
- **Học không giám sát (Unsupervised Learning):** Mô hình phân tích dữ liệu không có nhãn để tìm các mẫu hoặc cụm bất thường. Các mô hình như K-Means Clustering hoặc DBSCAN có thể phát hiện lưu lượng botnet bằng cách nhóm các mẫu giao tiếp bất thường, chẳng hạn như các truy vấn DNS động của botnet Kelihos. Phương pháp này hữu ích khi dữ liệu botnet mới chưa được gắn nhãn. Bên cạnh đó, học không có giám sát cũng có những khuyết điểm như phải phụ thuộc vào đặc trưng tốt và tốn nhiều tài nguyên.

Nhìn chung, mỗi loại đều có ưu điểm và nhược điểm khác nhau. Học có giám sát chính xác với botnet đã biết nhưng cần nhãn và kém với botnet mới. Học không giám sát linh hoạt, phát hiện botnet chưa biết, nhưng dễ sai. Kết hợp các loại và cập nhật liên tục là cần thiết để đối phó botnet tinh vi.

#### 2.2.2.3 Cách thức hoạt động của học máy

Học máy thông qua một quy trình có hệ thống, bao gồm các bước từ thu thập dữ liệu, tiền xử lý, huấn luyện mô hình, đến đánh giá và triển khai. Mỗi giai đoạn đều đóng vai trò quan trọng trong việc đảm bảo mô hình có thể nhận diện chính xác các mẫu botnet, đặc biệt trong môi trường mạng phức tạp và đa dạng.

- **Thu thập dữ liệu:** Dữ liệu được thu thập từ nhiều nguồn, bao gồm lưu lượng mạng, tần suất gửi gói tin, thời gian phản hồi, kích thước gói tin, mức sử dụng CPU, truy vấn DNS và truy vấn giao thức HTTP/HTTPS. Các công cụ như Wireshark, Zeek hoặc hệ thống giám sát Splunk được sử dụng để ghi lại dữ liệu từ mạng thực tế, trong khi các bộ dữ liệu công khai như CTU-13 (chứa hơn 10 triệu gói tin từ botnet Neris, Rbot) cung cấp các mẫu chuẩn để huấn luyện. Việc thu thập đòi hỏi giám sát liên tục để đảm bảo dữ liệu đủ phong phú tránh thiên lệch mô hình.

- **Tiền xử lý dữ liệu:** Giai đoạn tiền xử lý dữ liệu nhằm mục đích làm sạch và chuẩn bị dữ liệu thô để sử dụng trong huấn luyện mô hình, đảm bảo chất lượng và tính nhất quán của dữ liệu đầu vào. Dữ liệu thô thường chứa nhiều, như gói tin lỗi, trùng lặp, hoặc giá trị thiếu, cần được loại bỏ bằng các công cụ như Pandas hoặc Apache Spark. Sau đó, dữ liệu được chuẩn hóa để đưa các giá trị về một thang đo chung giúp các thuật toán học máy hoạt động hiệu quả hơn. Quá trình trích xuất đặc trưng cũng được thực hiện, trong đó các đặc trưng quan trọng như tần suất kết nối (số gói/phút), kích thước gói tin (bytes), thời gian giữa các yêu cầu (milliseconds) và tần suất truy vấn DNS. Giai đoạn này đòi hỏi chuyên môn cao để chọn đặc trưng phù hợp, vì lựa chọn sai có thể dẫn đến mô hình học sai hoặc kém hiệu quả, như bỏ sót các mẫu giao tiếp bất thường nếu không bao gồm đặc trưng thời gian giữa các gói.
- **Huấn luyện mô hình:** Giai đoạn huấn luyện mô hình là bước cốt lõi, nơi các thuật toán học máy được sử dụng để học các mẫu liên quan đến botnet từ dữ liệu đã tiền xử lý. Dữ liệu được chia thành các tập huấn luyện (70%), xác thực (15%), và kiểm tra (15%) để đảm bảo mô hình được đánh giá chính xác. Các mô hình được đề xuất bao gồm Naive Bayes, KNN, LDA, Decision Tree, Random Forest và SVM.
- **Đánh giá và triển khai:** Sau khi huấn luyện, mô hình được đánh giá trên dữ liệu kiểm tra để đo lường hiệu suất và triển khai vào thực tế. Các chỉ số đánh giá bao gồm:
  - Accuracy: tỷ lệ dự đoán đúng trên tổng số mẫu (botnet hoặc lành tính), phản ánh độ chính xác tổng thể của mô hình.
  - Precision: tỷ lệ dự đoán botnet thực sự là botnet, giúp giảm dương tính giả.
  - Recall: tỷ lệ botnet được phát hiện đúng trên tổng số botnet thực tế, đảm bảo không bỏ sót mối đe dọa.
  - F1-score: giá trị trung bình hài hòa giữa precision và recall, cân bằng hiệu suất tổng thể.

#### 2.2.2.4 Các mô hình học máy phổ biến

Nhiều mô hình học máy được sử dụng để phát hiện botnet phụ thuộc vào loại dữ liệu, mục tiêu phát hiện và đặc điểm của môi trường mạng. Các thuật toán này được chia thành hai nhóm chính: học máy có giám sát và học máy không giám sát. Mỗi nhóm có những ưu điểm và ứng dụng riêng trong việc nhận diện các mối đe dọa botnet tinh vi.

##### • Đối với học có giám sát

- **SVM:** Là một thuật toán học có giám sát mạnh mẽ, hoạt động bằng cách tìm một siêu phẳng (hyperplane) tối ưu để phân tách các lớp dữ liệu trong trường hợp này là lưu lượng botnet và lành tính dựa trên các đặc trưng như tần suất gói tin, tỷ lệ

TCP/UDP hoặc thời gian phản hồi. SVM sử dụng các hàm kernel như linear, RBF để xử lý dữ liệu phi tuyến tính, cho phép nó phân loại hiệu quả các mẫu lưu lượng phức tạp mà không thể phân tách bằng một đường thẳng đơn giản. Mô hình phù hợp với các tập dữ liệu có kích thước vừa và nhỏ, nơi số lượng đặc trưng giới hạn, vì SVM có khả năng tối ưu hóa biên phân tách giữa các lớp, giảm nguy cơ sai lệch. Trong phát hiện botnet, SVM hiệu quả trong việc phân loại lưu lượng mạng chẳng hạn như nhận diện các mẫu email lừa đảo của botnet Emotet, bằng cách phân tích các đặc trưng như tiêu đề email hoặc tần suất gửi gói HTTPS.

- **Random Forest:** Là một thuật toán học có giám sát dựa trên tập hợp (ensemble), kết hợp nhiều cây quyết định để đưa ra dự đoán chính xác hơn, giảm nguy cơ overfitting so với một cây quyết định đơn lẻ. Mỗi cây quyết định được huấn luyện trên một tập con ngẫu nhiên của dữ liệu và đặc trưng, sau đó kết quả được tổng hợp qua cơ chế voting để phân loại. Random Forest phân tích các đặc trưng như kích thước gói tin, thời gian gửi, tần suất truy vấn DNS hoặc mức sử dụng CPU để phân loại botnet, đặc biệt mạnh trong việc xử lý dữ liệu không cân bằng chẳng hạn như lưu lượng lành tính chiếm đa số so với botnet.

#### • Đối với học không giám sát

- **K-Means Clustering:** Là một thuật toán học không giám sát chia dữ liệu thành k cụm dựa trên sự tương đồng giữa các mẫu, với mỗi cụm đại diện cho một nhóm hành vi mạng cụ thể, và các điểm không thuộc cụm chính được coi là bất thường, có khả năng là botnet. Thuật toán này phân tích các đặc trưng như tần suất truy vấn DNS, thời gian gửi gói hoặc kích thước gói để nhóm lưu lượng mạng không yêu cầu nhãn dữ liệu khiến nó phù hợp để phát hiện botnet mới chưa được ghi nhận.
- **DBSCAN:** Là một thuật toán học không giám sát khác nhóm dữ liệu dựa trên mật độ, nhận diện các cụm có mật độ cao và coi các điểm rời rạc hoặc có mật độ thấp là bất thường có khả năng là botnet. Thuật toán này không yêu cầu xác định trước số lượng cụm khiến nó hiệu quả với dữ liệu mạng không đồng nhất như lưu lượng botnet kết hợp giao thức HTTP và P2P. DBSCAN phân tích các đặc trưng như thời gian gửi gói, tần suất kết nối hoặc tỷ lệ TCP/UDP để phát hiện botnet.

#### 2.2.2.5 Ưu và nhược điểm trong phát hiện botnet bằng học máy

##### Ưu điểm

- **Nhận diện botnet mới và chưa biết:** Học máy có khả năng phát hiện các botnet không có trong cơ sở dữ liệu chữ ký, vượt xa các phương pháp truyền thống như phát hiện dựa trên chữ ký. Bằng cách phân tích các mẫu bất thường trong lưu lượng mạng, hành vi thiết bị, hoặc truy vấn DNS, các thuật toán như Random Forest, K-Means, hoặc LSTM có thể nhận diện các biến thể botnet mới mà không cần thông tin trước đó. Điều này

đặc biệt quan trọng trong bối cảnh botnet ngày càng sử dụng kỹ thuật che giấu như mã hóa hoặc tên miền động (DGA), giúp giảm thiểu rủi ro từ các cuộc tấn công zero-day.

- **Xử lý dữ liệu phức tạp và đa dạng:** Học máy có thể phân tích hiệu quả các loại dữ liệu phức tạp, từ lưu lượng mạng mã hóa, giao tiếp P2P đến nhật ký thiết bị hoặc hành vi người dùng. Các thuật toán học sâu như CNN hoặc Autoencoders tự động trích xuất đặc trưng từ dữ liệu thô, như kích thước gói tin, thời gian gửi, hoặc tần suất kết nối mà không cần can thiệp thủ công.
- **Giảm tỷ lệ dương tính giả:** So với phương pháp phát hiện dựa trên bất thường, học máy cải thiện độ chính xác bằng cách học các mẫu giao tiếp hoặc hành vi cụ thể của botnet, giảm nhầm lẫn với lưu lượng hợp pháp. Các thuật toán như Random Forest hoặc Neural Networks được huấn luyện trên dữ liệu đa dạng có thể phân biệt lưu lượng botnet với các hoạt động hợp pháp bất thường.
- **Áp dụng linh hoạt cho nhiều kịch bản:** Học máy có thể được triển khai trong nhiều tình huống phát hiện botnet, từ phân loại lưu lượng mạng, nhận diện hành vi thiết bị đến phân tích giao tiếp C&C. Các thuật toán như SVM, K-Means, hoặc LSTM áp dụng được cho cả botnet tập trung, P2P và IoT.

### Nhược điểm

- **Yêu cầu lượng dữ liệu huấn luyện lớn và chất lượng cao:** Học máy phụ thuộc vào dữ liệu gồm cả mẫu botnet và lành tính để xây dựng mô hình chính xác. Tuy nhiên, thu thập dữ liệu botnet mới thường khó khăn do cần phân tích chuyên sâu hoặc triển khai honeypot. Việc gắn nhãn dữ liệu cũng tốn thời gian và chi phí đặc biệt đối với các bộ dữ liệu lớn.
- **Phụ thuộc vào chất lượng dữ liệu:** Hiệu suất của học máy bị ảnh hưởng mạnh bởi chất lượng dữ liệu đầu vào. Dữ liệu sai lệch, không đại diện hoặc chứa nhiễu như gói tin lỗi, nhật ký không đầy đủ có thể dẫn đến mô hình dự đoán sai, bỏ sót botnet hoặc tạo dương tính giả.
- **Tốn tài nguyên tính toán và thời gian:** Học máy đòi hỏi sức mạnh tính toán lớn, thường cần GPU/CPU mạnh mẽ và thời gian huấn luyện kéo dài từ vài giờ đến vài ngày. Điều này khiến học máy không phù hợp với các tổ chức có nguồn lực hạn chế như doanh nghiệp nhỏ. Ngay cả khi triển khai, các mô hình phức tạp cần hệ thống mạnh để phân tích thời gian thực, làm tăng chi phí vận hành so với các phương pháp truyền thống như phát hiện dựa trên chữ ký.

## 2.3 Mô hình Decision Tree

### 2.3.1 Tổng quan

Decision Tree là một thuật toán học máy thuộc nhóm học có giám sát, được sử dụng cho cả nhiệm vụ phân loại và hồi quy. Thuật toán này mô phỏng quá trình ra quyết định của con người thông qua một cấu trúc dạng cây, trong đó mỗi nút đại diện cho một quyết định hoặc điều kiện, mỗi nhánh thể hiện kết quả của quyết định đó, và các nút lá biểu thị kết quả cuối cùng hoặc dự đoán.

Decision Tree hoạt động bằng cách chia tập dữ liệu thành các tập con dựa trên giá trị của các đặc trưng, với mục tiêu tạo ra các tập con đồng nhất nhất có thể. Decision Tree là một biểu diễn trực quan dưới dạng sơ đồ, giúp đơn giản hóa việc ra quyết định bằng cách chia nhỏ vấn đề phức tạp thành các bước nhỏ hơn, dễ hiểu và dễ diễn giải, đặc biệt phù hợp với những người không có nền tảng thống kê sâu.

Cấu trúc của một Decision Tree là một biểu diễn dạng cây phân cấp, trong đó mỗi thành phần đóng vai trò cụ thể trong việc xây dựng và ra quyết định từ dữ liệu. Cấu trúc này bao gồm các thành phần chính sau:

- **Root Node:** Nút gốc là điểm khởi đầu của Decision Tree, đại diện cho toàn bộ tập dữ liệu ban đầu trước khi thực hiện bất kỳ phân chia nào. Đây là nơi thuật toán bắt đầu quá trình ra quyết định, chọn một đặc trưng tối ưu để phân chia dữ liệu thành các tập con nhỏ hơn. Đặc trưng được chọn thường dựa trên các tiêu chí như Information Gain hoặc Gini Index. Nút gốc đóng vai trò quan trọng trong việc định hình toàn bộ cấu trúc cây, vì một lựa chọn không tối ưu có thể dẫn đến các nhánh kém hiệu quả, làm giảm độ chính xác của mô hình. Nút gốc là bước đầu tiên trong quá trình phân chia đệ quy, và việc chọn đặc trưng đúng tại đây là yếu tố quyết định hiệu suất của cây.
- **Branches:** Các nhánh là các đường nối giữa các nút trong cây, đại diện cho các kết quả có thể xảy ra của một điều kiện tại nút cha, dẫn dữ liệu đến các nút con tiếp theo hoặc nút lá. Mỗi nhánh tương ứng với một giá trị hoặc khoảng giá trị của đặc trưng tại nút cha. Các nhánh tạo nên luồng logic của cây, giúp dữ liệu di chuyển từ gốc đến lá, và số lượng nhánh tại mỗi nút phụ thuộc vào đặc trưng. Các nhánh là biểu diễn trực quan của các quy tắc ra quyết định, giúp người dùng dễ dàng theo dõi và hiểu cách mô hình đưa ra dự đoán.
- **Internal Nodes:** Là nút quyết định, là các nút trung gian trong cây, nằm giữa nút gốc và các nút lá, đại diện cho các điểm phân chia bổ sung dựa trên các đặc trưng khác nhau. Mỗi nút nội kiểm tra một điều kiện cụ thể trên một đặc trưng, chia dữ liệu thành các tập con nhỏ hơn để tăng độ thuần nhất của các lớp. Internal nodes là nơi thể hiện tính chất đệ quy của thuật toán, và số lượng internal nodes phụ thuộc vào độ phức tạp của dữ liệu và tiêu chí dừng được thiết lập như max\_depth hoặc min\_samples\_split.

- **Leaf Nodes:** Leaf Nodes là các nút cuối cùng trong cây, không thực hiện phân chia thêm và đại diện cho kết quả dự đoán cuối cùng của mô hình. Trong bài toán phân loại, mỗi nút lá tương ứng với một nhãn lớp, ví dụ "botnet" hoặc "lành tính", còn trong bài toán hồi quy, Leaf Nodes có thể là một giá trị số như dự đoán mức sử dụng CPU trung bình. Số lượng Leaf Nodes phụ thuộc vào độ phức tạp của cây và dữ liệu, và chúng là nơi cung cấp kết quả cuối cùng cho người dùng.
- **Splitting Criteria:** Tiêu chí phân chia là các thước đo định lượng được sử dụng để đánh giá và lựa chọn đặc trưng tối ưu tại mỗi nút trong Decision Tree, nhằm chia dữ liệu thành các tập con đồng nhất nhất có thể, tức là các tập con mà các mẫu trong đó thuộc về cùng một lớp hoặc có giá trị gần nhau. Mục tiêu chính của tiêu chí phân chia là giảm thiểu sự không thuần nhất hoặc bất định trong dữ liệu sau mỗi lần phân chia, từ đó tăng khả năng dự đoán chính xác của mô hình. Có nhiều thước đo phổ biến được sử dụng trong Decision Tree, bao gồm Information Gain, Entropy, Gini Index và một số tiêu chí khác như Chi-squared hoặc Variance Reduction. Mỗi tiêu chí có cách tiếp cận riêng để đánh giá độ thuần nhất.

- **Entropy** là một thước đo bất định được sử dụng trong các thuật toán như ID3 và C4.5. Entropy đo lường mức độ hỗn loạn của dữ liệu, với công thức:

$$H = - \sum_{i=1}^n p_i \log_2(p_i)$$

Trong đó  $p_i$  là tỷ lệ của lớp  $i$  trong tập dữ liệu, và  $n$  là số lượng lớp. Entropy có giá trị từ 0 (tập dữ liệu hoàn toàn thuần nhất, tất cả mẫu thuộc cùng một lớp) đến 1 (tập dữ liệu hoàn toàn hỗn loạn, các lớp phân bố đồng đều). Khi phân chia dữ liệu, thuật toán tính entropy trước và sau phân chia, từ đó xác định mức giảm bất định.

- **Information Gain** là một tiêu chí trực tiếp dựa trên entropy, đo lường mức giảm entropy sau khi phân chia dữ liệu theo một đặc trưng với công thức:

$$IG = H(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} H(S_v)$$

Trong đó  $H(S)$  là entropy của tập dữ liệu ban đầu,  $S_v$  là tập con sau khi phân chia theo giá trị  $v$  của đặc trưng  $A$ ,  $S_v$  là số mẫu trong tập con, và  $|S|$  là tổng số mẫu. Information Gain càng cao, đặc trưng đó càng được ưu tiên để phân chia, vì nó giúp giảm bất định nhiều nhất. Tiêu chí này đặc biệt hữu ích trong các bài toán phân loại, vì nó tập trung vào việc tối ưu hóa sự phân biệt giữa các lớp.

- **Gini index** là một thước đo khác, thường được sử dụng trong thuật toán CART (Classification and Regression Tree), đánh giá độ không thuần nhất của dữ liệu với công thức:

$$Gini = 1 - \sum_{i=1}^n (p_i)^2$$

Trong đó  $p_i$  là tỷ lệ của lớp  $i$  và  $n$  là số lượng lớp. Gini Index có giá trị từ 0 là lúc tập dữ liệu hoàn toàn thuần nhất đến 0.5 lúc tập dữ liệu hỗn loạn nhất, với hai lớp phân bố đồng đều. Khi phân chia, thuật toán tính Gini Index cho từng tập con sau phân chia và lấy trung bình có trọng số theo kích thước tập con, sau đó chọn đặc trưng có Gini Index tổng thấp nhất, vì điều đó cho thấy các tập con thuần nhất hơn. Gini Index được ưa chuộng trong các bài toán phân loại vì tính toán nhanh hơn entropy (không cần logarit) và hiệu quả trong các tình huống dữ liệu đơn giản.

- **Variance Reduction** là một tiêu chí được sử dụng trong các bài toán hồi quy, nơi mục tiêu là dự đoán giá trị liên tục thay vì nhãn lớp. Tiêu chí này đo lường mức giảm phương sai (variance) của giá trị mục tiêu sau khi phân chia, với công thức phương:

$$Var = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$$

Trong đó  $x_i$  là giá trị của mẫu  $i$ ,  $\bar{x}$  là giá trị trung bình và  $n$  là số lượng mẫu. Khi phân chia, thuật toán tính phương sai của tập dữ liệu ban đầu và các tập con, sau đó chọn đặc trưng làm giảm phương sai nhiều nhất, đảm bảo các giá trị trong mỗi tập con gần nhau hơn. Tiêu chí này giúp Decision Tree dự đoán chính xác hơn trong các bài toán như dự đoán mức sử dụng CPU của thiết bị.

- **Chi-squared** là một tiêu chí ít phổ biến hơn, thường được sử dụng trong các thuật toán như CHAID (Chi-squared Automatic Interaction Detection). Tiêu chí này sử dụng kiểm định Chi-squared để đo lường mức độ phụ thuộc thống kê giữa đặc trưng phân chia và biến mục tiêu, với công thức:

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

Trong đó  $O_i$  là tần suất quan sát và  $E_i$  là tần suất kỳ vọng. Đặc trưng có giá trị Chi-squared cao hơn sẽ được chọn, vì nó cho thấy mối quan hệ mạnh mẽ hơn với biến mục tiêu. Tiêu chí này phù hợp với dữ liệu phân loại, đặc biệt khi cần xác định các tương tác phức tạp giữa các đặc trưng.

### 2.3.2 Cách xây dựng thuật toán

Xây dựng Decision Tree là một quy trình có hệ thống nhằm tạo ra một mô hình dự đoán từ dữ liệu, giúp phân loại hoặc dự đoán giá trị dựa trên các đặc trưng. Quy trình này bao gồm các bước cụ thể để đảm bảo cây được xây dựng hiệu quả và có khả năng tổng quát hóa tốt:

1. **Chuẩn bị dữ liệu:** Bước đầu tiên là thu thập dữ liệu từ các nguồn phù hợp sau đó tiến hành làm sạch để loại bỏ các giá trị thiếu, trùng lặp, hoặc nhiễu có thể ảnh hưởng đến quá trình huấn luyện. Các đặc trưng quan trọng như tần suất hoạt động, kích thước gói tin, hoặc thời gian phản hồi được trích xuất từ dữ liệu thô để sử dụng trong phân tích. Dù Decision Tree không yêu cầu chuẩn hóa nghiêm ngặt nhưng một số trường hợp vẫn cần chuẩn hóa dữ liệu để đảm bảo tính đồng nhất đặc biệt khi làm việc với các đặc trưng có thang đo khác nhau.
2. **Chọn đặc trưng để phân chia:** Tại mỗi nút trong cây, thuật toán đánh giá tất cả các đặc trưng để chọn ra đặc trưng tối ưu dựa trên khả năng phân tách dữ liệu thành các tập con đồng nhất nhất tức là các tập con có các mẫu thuộc cùng một lớp hoặc giá trị gần nhau. Quá trình này được lặp lại ở mỗi nút, từ nút gốc đến các nút nội nhằm đảm bảo cây được xây dựng theo cách tối ưu hóa khả năng dự đoán. Việc chọn đặc trưng đóng vai trò quan trọng trong việc định hình cấu trúc cây và ảnh hưởng đến hiệu suất tổng thể.
3. **Xây dựng cây:** Sau khi chọn đặc trưng, thuật toán phân chia dữ liệu tại nút gốc thành các tập con dựa trên giá trị của đặc trưng đó, ví dụ chia thành các nhánh "Có" và "Không". Quá trình phân chia tiếp tục diễn ra đệ quy tại các nút nội mỗi lần chọn một đặc trưng mới để phân tách dữ liệu cho đến khi đạt tiêu chí dừng, chẳng hạn như khi tất cả mẫu trong một tập con thuộc cùng một lớp số lượng mẫu quá nhỏ, hoặc cây đạt độ sâu tối đa. Kết quả là một cấu trúc cây với các nhánh dẫn từ nút gốc đến các nút lá đại diện cho các kết quả dự đoán.
4. **Cắt tỉa (Pruning):** Để tránh cây trở nên quá phức tạp và quá khớp với dữ liệu huấn luyện, kỹ thuật cắt tỉa được áp dụng. Cắt tỉa trước thiết lập các giới hạn như độ sâu tối đa hoặc số mẫu tối thiểu trước khi xây dựng trong khi cắt tỉa sau loại bỏ các nhánh không cải thiện hiệu suất trên tập dữ liệu xác thực sau khi cây hoàn tất. Quá trình này giúp giảm độ phức tạp, tăng khả năng tổng quát hóa và đảm bảo mô hình hoạt động tốt hơn trên dữ liệu mới, đặc biệt trong các ứng dụng thực tế.
5. **Kiểm tra và tinh chỉnh:** Mô hình sau khi xây dựng được kiểm tra trên một tập dữ liệu kiểm tra độc lập để đánh giá hiệu suất sẽ sử dụng các chỉ số như độ chính xác, độ nhạy, độ đặc hiệu và F1 score. Dựa trên kết quả các tham số của cây như độ sâu tối đa, số mẫu tối thiểu để phân chia, và các ngưỡng dừng có thể được tinh chỉnh để cải thiện hiệu quả.

Kỹ thuật kiểm tra chéo thường được sử dụng để đảm bảo mô hình không chỉ phù hợp với dữ liệu huấn luyện mà còn hiệu quả trong các tình huống thực tế.

### 2.3.3 Ưu điểm và nhược điểm

#### 2.3.3.1 Ưu điểm

Decision Tree sở hữu nhiều ưu điểm nổi bật, khiến nó trở thành một công cụ quan trọng trong học máy:

- **Dễ hiểu và diễn giải:** Cấu trúc cây với các nút và nhánh mang tính trực quan, cho phép người dùng, kể cả những người không có nền tảng kỹ thuật sâu, dễ dàng hiểu cách mô hình đưa ra quyết định. Điều này đặc biệt hữu ích trong các lĩnh vực như y tế hoặc tài chính, nơi giải thích kết quả cho khách hàng hoặc bệnh nhân là cần thiết, nhờ tính chất "white box" của thuật toán.
- **Yêu cầu tiền xử lý tối thiểu:** Không giống một số thuật toán khác đòi hỏi chuẩn hóa dữ liệu hoặc mã hóa phức tạp, Decision Tree có thể hoạt động hiệu quả ngay cả với dữ liệu thô, miễn là các đặc trưng được định nghĩa rõ ràng. Điều này tiết kiệm thời gian và công sức trong giai đoạn chuẩn bị dữ liệu, đặc biệt khi làm việc với các tập dữ liệu lớn và đa dạng.
- **Xử lý được nhiều loại dữ liệu:** Thuật toán linh hoạt trong việc xử lý cả dữ liệu số và dữ liệu phân loại đồng thời có khả năng quản lý giá trị thiếu bằng cách bỏ qua hoặc thay thế chúng, giúp nó phù hợp với nhiều kịch bản thực tế khác nhau.
- **Hỗ trợ lựa chọn đặc trưng:** Decision Tree tự động xác định và ưu tiên các đặc trưng quan trọng thông qua quá trình xây dựng giúp giảm chiều dữ liệu mà không cần các kỹ thuật giảm chiều bổ sung như PCA. Điều này không chỉ cải thiện hiệu suất mà còn cung cấp thông tin giá trị về các yếu tố ảnh hưởng lớn đến kết quả dự đoán.

#### 2.3.3.2 Nhược điểm

Mặc dù có nhiều ưu điểm, Decision Tree cũng đối mặt với một số hạn chế nhất định:

- **Dễ bị overfitting:** Khi cây phát triển quá sâu với nhiều nhánh, nó có thể học chi tiết các đặc điểm ngẫu nhiên trong dữ liệu huấn luyện dẫn đến hiệu suất kém trên dữ liệu mới. Hiện tượng này thường xảy ra khi không có các biện pháp kiểm soát như cắt tỉa làm giảm khả năng tổng quát hóa của mô hình.
- **Không ổn định:** Một thay đổi nhỏ trong dữ liệu như thêm hoặc bớt một vài mẫu có thể làm thay đổi toàn bộ cấu trúc cây và kết quả dự đoán. Sự không ổn định này đòi hỏi việc kết hợp với các kỹ thuật như bagging để cải thiện độ tin cậy, đặc biệt trong các bài toán nhạy cảm với dữ liệu.

- **Hiệu suất kém với dữ liệu không cân bằng:** Khi tập dữ liệu có sự mất cân bằng lớn giữa các lớp, Decision Tree có thể thiên vị lớp chiếm ưu thế dẫn đến bỏ sót các mẫu thiểu số quan trọng, một vấn đề phổ biến trong phân loại mạng.
- **Hạn chế với bài toán hồi quy phức tạp:** Trong các bài toán dự đoán giá trị liên tục, Decision Tree có xu hướng mất thông tin khi chia dữ liệu thành các nhóm làm giảm độ chính xác so với các phương pháp hồi quy khác như hồi quy tuyến tính, đặc biệt khi dữ liệu có nhiều biến số phụ thuộc phức tạp.

#### 2.3.4 Ứng dụng mô hình

Decision Tree được áp dụng rộng rãi trong nhiều lĩnh vực nhờ tính linh hoạt, khả năng xử lý đa dạng loại dữ liệu, và cấu trúc trực quan dễ hiểu.

- **Phân loại lưu lượng mạng:** Thuật toán này được sử dụng để phát hiện botnet trong các mạng máy tính, đặc biệt hiệu quả khi phân tích lưu lượng bất thường trên các tập dữ liệu như CTU-13. Decision Tree có thể nhận diện các mẫu hành vi độc hại bằng cách phân tích các đặc trưng như tần suất gói tin và thời gian phản hồi. Ứng dụng này ngày càng quan trọng trong bối cảnh an ninh mạng, nơi việc phát hiện sớm các mối đe dọa là yếu tố then chốt để bảo vệ hệ thống.
- **Hệ thống tín dụng:** Trong ngành tài chính, Decision Tree hỗ trợ các tổ chức ngân hàng và tín dụng đánh giá rủi ro của khách hàng khi xin vay vốn. Thuật toán phân tích các yếu tố như lịch sử tín dụng, thu nhập hàng tháng, và tỷ lệ nợ trên thu nhập (DTI) để đưa ra quyết định chấp thuận hoặc từ chối, đảm bảo quy trình đánh giá đồng nhất và giảm thiểu rủi ro tài chính. Ứng dụng này không chỉ giúp tối ưu hóa việc phân bổ nguồn vốn mà còn hỗ trợ xây dựng các mô hình dự đoán xu hướng tín dụng trong tương lai.
- **Chẩn đoán y tế:** Decision Tree được ứng dụng trong y học để hỗ trợ chẩn đoán bệnh dựa trên các triệu chứng và dữ liệu bệnh nhân. Sự dễ hiểu của cây quyết định cho phép các bác sĩ giải thích kết quả cho bệnh nhân một cách rõ ràng, đồng thời hỗ trợ nghiên cứu y học trong việc phát triển các mô hình chẩn đoán tự động.
- **Hệ thống gợi ý:** Các nền tảng thương mại điện tử và dịch vụ trực tuyến như Amazon hoặc Netflix sử dụng Decision Tree để xây dựng hệ thống gợi ý sản phẩm hoặc nội dung. Thuật toán phân tích hành vi duyệt web, lịch sử mua sắm, và sở thích cá nhân để đề xuất các mặt hàng phù hợp, từ đó nâng cao trải nghiệm người dùng và tăng doanh thu. Ứng dụng này cũng được mở rộng sang các lĩnh vực như giáo dục trực tuyến, nơi gợi ý khóa học dựa trên tiến độ học tập.
- **Quản lý kinh doanh và chiến lược:** Decision Tree hỗ trợ các doanh nghiệp trong việc ra quyết định chiến lược như đánh giá khả năng mở rộng thị trường hoặc mở chi nhánh mới. Bằng cách phân tích các yếu tố như dân số khu vực, thu nhập bình quân đầu người

và nhu cầu thị trường, thuật toán cung cấp các kịch bản khả thi giúp nhà quản lý lựa chọn phương án tối ưu. Ứng dụng này còn được sử dụng để tối ưu hóa ngân sách tiếp thị và lập kế hoạch dài hạn.

## 2.4 Mô hình Random Forest

### 2.4.1 Tổng quan

Random Forest là một thuật toán học máy thuộc nhóm học có giám sát được thiết kế dựa trên nguyên tắc kết hợp nhiều cây quyết định để nâng cao độ chính xác và độ tin cậy trong việc dự đoán. Thuật toán này là một phần của phương pháp ensemble learning, cụ thể sử dụng kỹ thuật bagging (Bootstrap Aggregating), trong đó các cây quyết định được huấn luyện độc lập trên các tập con ngẫu nhiên của dữ liệu huấn luyện và kết quả cuối cùng được tổng hợp thông qua cơ chế bỏ phiếu trong phân loại hoặc lấy trung bình trong hồi quy. Random Forest nổi bật với khả năng xử lý các tập dữ liệu phức tạp, giảm thiểu nguy cơ quá khớp so với các cây quyết định đơn lẻ và cung cấp các công cụ đánh giá độ quan trọng của đặc trưng khiến nó trở thành một trong những thuật toán được ưa chuộng nhất trong học máy hiện đại.

Cấu trúc của Random Forest bao gồm các thành phần chính liên quan đến tập hợp các cây quyết định và cách chúng được tích hợp để đưa ra kết quả:

- **Individual Decision Trees:** Mỗi cây quyết định trong Random Forest là một đơn vị độc lập, được huấn luyện trên một tập con ngẫu nhiên của dữ liệu, với cấu trúc và độ sâu có thể khác nhau tùy thuộc vào cách dữ liệu và đặc trưng được chọn. Sự đa dạng giữa các cây là yếu tố cốt lõi giúp tăng cường hiệu suất tổng thể của rừng.
- **Bootstrap Samples:** Đây là các tập con dữ liệu được tạo ra bằng cách lấy ngẫu nhiên các mẫu từ sampling with replacement để đảm bảo rằng mỗi cây được huấn luyện trên một tập dữ liệu khác nhau, từ đó tăng tính đa dạng và giảm nguy cơ thiên lệch trong mô hình.
- **Feature Subset:** Tại mỗi nút của mỗi cây, chỉ có một tập con ngẫu nhiên của các đặc trưng được xem xét để thực hiện phân chia thay vì sử dụng toàn bộ đặc trưng giúp giảm sự tương quan giữa các cây và tránh tình trạng các cây sao chép lẫn nhau, từ đó cải thiện khả năng tổng quát hóa.
- **Aggregation Mechanism:** Kết quả từ tất cả các cây quyết định được tổng hợp để đưa ra dự đoán cuối cùng. Trong bài toán phân loại, cơ chế này sử dụng bỏ phiếu đa số để chọn lớp có số phiếu cao nhất. Còn trong hồi quy, nó lấy trung bình của các giá trị dự đoán từ các cây, đảm bảo một kết quả ổn định và chính xác hơn.
- **Out-of-Bag (OOB) Samples:** Đây là các mẫu không được sử dụng trong quá trình huấn luyện của một cây cụ thể, được giữ lại để đánh giá hiệu suất của cây đó một cách nội

bộ. Tập OOB đóng vai trò như một tập kiểm tra tự động, cho phép đánh giá mô hình mà không cần tập dữ liệu kiểm tra riêng biệt.

#### 2.4.2 Cách xây dựng mô hình

Xây dựng Random Forest là một quy trình phức tạp nhưng có hệ thống nhằm tạo ra một mô hình dự đoán mạnh mẽ bằng cách kết hợp nhiều cây quyết định. Quy trình này bao gồm nhiều bước từ chuẩn bị dữ liệu, tạo các tập con ngẫu nhiên, huấn luyện các cây, đến tổng hợp kết quả và tinh chỉnh mô hình để đảm bảo hiệu suất tối ưu.

1. **Chuẩn bị dữ liệu:** Bước đầu tiên là thu thập dữ liệu từ các nguồn phù hợp sau đó tiến hành làm sạch dữ liệu để loại bỏ các giá trị thiếu, trùng lặp hoặc nhiễu có thể ảnh hưởng đến hiệu suất của mô hình. Quá trình này thường bao gồm việc xử lý các giá trị thiếu bằng cách thay thế chúng bằng giá trị trung bình hoặc giá trị phổ biến nhất loại bỏ các ngoại lệ nếu cần, và đảm bảo dữ liệu được định dạng đúng để phù hợp với thuật toán. Tiếp theo, các đặc trưng quan trọng như tần suất gói tin, thời gian phản hồi hoặc các thuộc tính khác được trích xuất từ dữ liệu thô, sử dụng các công cụ như Pandas hoặc Spark để chuẩn bị tập dữ liệu hoàn chỉnh. Mặc dù Random Forest không yêu cầu chuẩn hóa nghiêm ngặt, một số trường hợp có thể cần chuẩn hóa hoặc mã hóa các biến phân loại để đảm bảo tính nhất quán, đặc biệt khi làm việc với các tập dữ liệu có thang đo khác nhau.
2. **Tạo mẫu bootstrap:** Sau khi dữ liệu được chuẩn bị, Random Forest sử dụng kỹ thuật bootstrap để tạo ra các tập con dữ liệu ngẫu nhiên cho từng cây quyết định. Quá trình này lấy mẫu từ sampling with replacement, là một mẫu có thể được chọn nhiều lần hoặc không được chọn trong một tập con cụ thể. Thông thường, mỗi tập con có kích thước bằng tập dữ liệu gốc nhưng do lấy mẫu ngẫu nhiên, khoảng 1/3 số mẫu sẽ không được sử dụng trong một tập con cụ thể, được gọi là mẫu ngoài bao (OOB). Các mẫu OOB này sẽ được sử dụng để đánh giá hiệu suất nội bộ sau này. Việc tạo mẫu bootstrap đảm bảo rằng mỗi cây được huấn luyện trên một tập dữ liệu khác nhau giúp tăng tính đa dạng và giảm nguy cơ thiên lệch, một yếu tố quan trọng giúp Random Forest đạt hiệu suất cao.
3. **Xây dựng cây:** Mỗi cây quyết định trong rừng được huấn luyện độc lập trên một tập con dữ liệu bootstrap với một đặc điểm quan trọng là tại mỗi nút của cây, chỉ một tập con ngẫu nhiên của các đặc trưng được xem xét để thực hiện phân chia. Quá trình xây dựng cây sử dụng các tiêu chí phân chia như Gini Index hoặc Entropy để chọn đặc trưng tối ưu tại mỗi nút, phân chia dữ liệu thành các tập con đồng nhất hơn và lặp lại đệ quy cho đến khi đạt tiêu chí dừng như độ sâu tối đa, số mẫu tối thiểu tại mỗi nút hoặc tất cả mẫu trong một tập con thuộc cùng một lớp. Quá trình này được lặp lại cho từng cây trong rừng.
4. **Tổng hợp kết quả:** Sau khi tất cả các cây được huấn luyện, Random Forest tổng hợp

kết quả từ các cây để đưa ra dự đoán cuối cùng. Trong bài toán phân loại, cơ chế bỏ phiếu đa số được sử dụng: mỗi cây đưa ra một dự đoán và nhãn được chọn nhiều nhất từ tất cả các cây sẽ là kết quả cuối cùng. Trong bài toán hồi quy, kết quả là giá trị trung bình của các dự đoán từ tất cả các cây giúp giảm thiểu sai số và tăng độ ổn định. Quá trình tổng hợp này tận dụng sức mạnh của tập hợp giúp giảm thiểu tác động của các cây riêng lẻ có hiệu suất kém và cung cấp một dự đoán tổng thể đáng tin cậy hơn so với một cây quyết định đơn lẻ. Đây là một trong những lý do chính khiến Random Forest được đánh giá cao trong các bài toán thực tế.

5. **Kiểm tra và tinh chỉnh:** Cuối cùng, mô hình được kiểm tra và tinh chỉnh để đảm bảo hiệu suất tối ưu. Random Forest có lợi thế là có thể sử dụng mẫu ngoài bao (OOB) để đánh giá hiệu suất nội bộ mà không cần tập kiểm tra riêng biệt vì mỗi cây không được huấn luyện trên khoảng 1/3 dữ liệu (mẫu OOB). Các chỉ số như độ chính xác, độ nhạy, độ đặc hiệu, điểm F1-score được tính toán trên tập OOB hoặc một tập kiểm tra độc lập để đánh giá hiệu suất tổng thể. Dựa trên kết quả, các tham số của mô hình như số cây (`n_estimators`), độ sâu tối đa (`max_depth`), số đặc trưng tối đa tại mỗi nút (`max_features`) hoặc số mẫu tối thiểu để phân chia (`min_samples_split`) có thể được điều chỉnh để cải thiện hiệu quả. Các kỹ thuật như Grid Search hoặc Random Search thường được sử dụng để tìm kiếm tổ hợp tham số tối ưu đảm bảo mô hình không chỉ hoạt động tốt trên dữ liệu huấn luyện mà còn hiệu quả trên các kịch bản thực tế như phân loại lưu lượng mạng hoặc dự đoán tài chính.

### 2.4.3 Ưu điểm và nhược điểm

#### 2.4.3.1 Ưu điểm

Random Forest mang lại nhiều lợi thế nổi bật, khiến nó trở thành một thuật toán được ưa chuộng trong học máy:

- **Giảm overfitting hiệu quả:** Bằng cách kết hợp nhiều cây quyết định thông qua kỹ thuật bagging, Random Forest giảm thiểu nguy cơ quá khớp so với một cây quyết định đơn lẻ đảm bảo mô hình không chỉ học tốt trên dữ liệu huấn luyện mà còn có khả năng tổng quát hóa mạnh mẽ trên các tập dữ liệu mới, đặc biệt trong các bài toán phức tạp như phân loại hình ảnh hoặc dự đoán tài chính.
- **Khả năng xử lý dữ liệu lớn:** Thuật toán có thể xử lý các tập dữ liệu có số lượng lớn đặc trưng và mẫu một cách hiệu quả nhờ việc sử dụng tập con ngẫu nhiên của dữ liệu và đặc trưng trong quá trình huấn luyện. Điều này làm cho Random Forest trở thành lựa chọn lý tưởng cho các ứng dụng thực tế như phân tích dữ liệu lớn trong mạng, tài chính và khoa học dữ liệu, nơi dữ liệu thường có kích thước khổng lồ và độ phức tạp cao.
- **Đánh giá độ quan trọng của đặc trưng:** Random Forest cung cấp một công cụ tích hợp để đo lường mức độ quan trọng của từng đặc trưng thông qua chỉ số giảm độ không

thuần nhất (Gini importance) hoặc mức độ giảm độ chính xác khi loại bỏ đặc trưng giúp các nhà phân tích hiểu rõ hơn về dữ liệu và tập trung vào các yếu tố ảnh hưởng lớn đến kết quả dự đoán, một lợi thế quan trọng trong các nghiên cứu khoa học hoặc tối ưu hóa mô hình.

- **Khả năng xử lý dữ liệu không cân bằng:** Nhờ cơ chế tổng hợp kết quả qua bỏ phiếu hoặc trung bình, Random Forest có thể xử lý tốt các tập dữ liệu không cân bằng giúp giảm thiểu thiên lệch về lớp chiếm ưu thế, một vấn đề thường gặp trong các bài toán như phát hiện gian lận, phân loại botnet hoặc chẩn đoán y khoa.

#### 2.4.3.2 Nhược điểm

Mặc dù có nhiều ưu điểm, Random Forest cũng đối mặt với một số hạn chế đáng kể:

- **Tốn tài nguyên tính toán:** Việc huấn luyện đồng thời nhiều cây quyết định đòi hỏi lượng lớn tài nguyên tính toán gồm bộ nhớ và thời gian xử lý, đặc biệt khi làm việc với dữ liệu rất lớn hoặc khi số cây ( $n_{\text{estimators}}$ ) được đặt ở mức cao để đảm bảo độ chính xác. Điều này có thể gây khó khăn khi triển khai trên các hệ thống có tài nguyên hạn chế hoặc trong các ứng dụng thời gian thực.
- **Khó diễn giải:** Mặc dù mỗi cây quyết định riêng lẻ dễ hiểu nhưng sự kết hợp của nhiều cây trong Random Forest làm cho toàn bộ mô hình trở thành một "hộp đen", khiến việc giải thích chi tiết cách mỗi dự đoán được đưa ra trở nên phức tạp. Điều này là một hạn chế trong các lĩnh vực yêu cầu tính minh bạch cao như y tế hoặc luật pháp.
- **Không tối ưu cho dữ liệu tuyến tính:** Random Forest hoạt động tốt nhất với dữ liệu phi tuyến tính nhưng kém hiệu quả khi mối quan hệ giữa đặc trưng và biến mục tiêu mang tính tuyến tính vì nó không tận dụng được các đặc tính toán học tuyến tính như hồi quy tuyến tính hoặc mô hình tuyến tính tổng quát, dẫn đến hiệu suất giảm trong một số trường hợp cụ thể.
- **Thời gian huấn luyện lâu:** So với các mô hình đơn giản như Logistic Regression hoặc một cây quyết định đơn lẻ, Random Forest cần thời gian huấn luyện dài hơn do phải xây dựng và tổng hợp kết quả từ nhiều cây, đặc biệt khi số cây được tăng lên để cải thiện độ chính xác hoặc khi làm việc với các tập dữ liệu có số lượng đặc trưng lớn làm tăng chi phí tính toán và thời gian triển khai.

#### 2.4.4 Ứng dụng mô hình

Random Forest được áp dụng rộng rãi trong nhiều lĩnh vực khác nhau nhờ khả năng xử lý dữ liệu lớn, độ chính xác cao và tính linh hoạt trong các kịch bản thực tế.

- **Phân loại và phát hiện bất thường trong mạng:** Random Forest là một công cụ quan trọng trong lĩnh vực an ninh mạng được sử dụng để phát hiện botnet trên các tập dữ liệu

tiêu chuẩn như CTU-13. Thuật toán phân tích các đặc trưng phức tạp như tần suất gói tin, thời gian phản hồi và các truy vấn DNS bất thường để nhận diện các mẫu hành vi độc hại từ botnet. Ứng dụng này không chỉ giúp phát hiện sớm các cuộc tấn công DDoS hoặc spam mà còn hỗ trợ các hệ thống phòng thủ tự động hóa giúp giảm thiểu thời gian phản ứng và tăng cường bảo mật cho các tổ chức lớn.

- **Phân tích y sinh:** Random Forest đóng vai trò quan trọng trong phân loại hình ảnh y tế như phát hiện khối u trong hình ảnh MRI hoặc dự đoán bệnh ung thư dựa trên biểu hiện gen và các yếu tố sinh học khác. Ngoài ra, nó còn được sử dụng để hỗ trợ chẩn đoán bệnh tim từ các thông số như huyết áp, tuổi tác và kết quả xét nghiệm máu cung cấp các dự đoán đáng tin cậy cho các bác sĩ. Tính linh hoạt của Random Forest cho phép xử lý các tập dữ liệu đa chiều và không đồng nhất giúp hỗ trợ nghiên cứu y học trong việc phát triển các công cụ chẩn đoán tiên tiến.
- **Hệ thống gợi ý nâng cao:** Các nền tảng lớn như Netflix, Spotify và Amazon áp dụng Random Forest để xây dựng hệ thống gợi ý cá nhân hóa phân tích sâu dữ liệu về sở thích người dùng, lịch sử xem hoặc nghe và tương tác trước đó để đề xuất nội dung phù hợp. Ứng dụng này không chỉ nâng cao trải nghiệm người dùng mà còn hỗ trợ các công ty phân tích xu hướng thị hiếu, điều chỉnh chiến lược kinh doanh và tối ưu hóa danh mục sản phẩm hoặc dịch vụ để đáp ứng nhu cầu thị trường một cách hiệu quả.
- **Quản lý tài nguyên thiên nhiên:** Random Forest được sử dụng để phân loại và dự đoán các khu vực tự nhiên như rừng, đất nông nghiệp và vùng đất dễ bị xói mòn dựa trên dữ liệu vệ tinh, thông tin khí hậu và các yếu tố địa lý khác. Ứng dụng này hỗ trợ các nhà khoa học và nhà hoạch định chính sách lập kế hoạch bảo tồn, quản lý tài nguyên bền vững và dự đoán tác động của biến đổi khí hậu.

## CHƯƠNG 3. MÔ HÌNH ĐỀ XUẤT

### 3.1 Những công trình nghiên cứu liên quan

Trong lúc tìm hiểu về đề tài dự án phát botnet, nhóm em đã tìm được rất nhiều tài liệu liên quan đề xuất những ý tưởng:

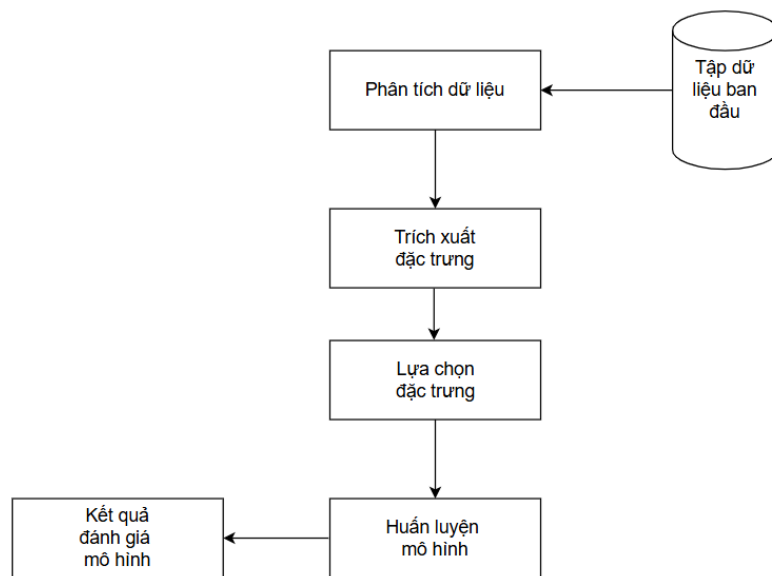
- **Zhang et al. (2022)** trong bài báo "**Botnet detection using graph-based feature clustering**" đã đề xuất một phương pháp phát hiện botnet dựa trên phân tích đồ thị mạng, sử dụng bộ dữ liệu CTU-13 để xây dựng các đặc trưng như mức độ trung tâm và kết nối giữa các nút. Phương pháp này kết hợp thuật toán Random Forest để phân loại lưu lượng botnet, đạt độ chính xác 94.3% và F1-score 93.5%. Kết quả nổi bật trong việc nhận diện các botnet như Neris và Rbot, nhưng hạn chế nằm ở việc phụ thuộc vào dữ liệu NetFlow có nhãn, dẫn đến hiệu suất giảm khi xử lý các botnet chưa được ghi nhận trước đó.
- **Velasco-Mata et al. (2023)** trong bài "**Real-time Botnet Detection on Large Network Bandwidths Using Machine Learning**" đã giới thiệu một hệ thống phát hiện botnet thời gian thực, sử dụng kết hợp các thuật toán học máy như Random Forest, XGBoost và phân tích hành vi dựa trên lưu lượng mạng. Nghiên cứu được thử nghiệm trên tập dữ liệu CICIDS2017, đạt độ chính xác 96.5% và F1-score 92.6%. Mô hình này nổi bật với khả năng phân tích thời gian thực trên băng thông lớn, nhưng hạn chế nằm ở yêu cầu tài nguyên tính toán cao, không phù hợp cho các thiết bị biên có tài nguyên hạn chế. Ngoài ra, hiệu suất giảm khi xử lý các botnet mới chưa có trong dữ liệu huấn luyện.
- **Kim et al. (2024)** trong bài "**Real-Time Botnet Detection with Gradient Boosting on CTU-13**" đề xuất sử dụng thuật toán Gradient Boosting trên bộ dữ liệu CTU-13 để phát hiện botnet dựa trên các đặc trưng thời gian thực như tần suất gói tin và độ trễ giao tiếp. Mô hình đạt độ chính xác 97.1% và F1-score 96.4%, phù hợp cho phân tích mạng doanh nghiệp. Hạn chế chính là mô hình nhạy cảm với nhiễu dữ liệu, dẫn đến tăng dương tính giả khi lưu lượng mạng có hoạt động bất thường không phải botnet, như lưu lượng video trực tuyến.
- **Patel et al. (2025)** trong bài "**Unsupervised Clustering for Botnet Detection on CTU-13 Dataset**" tập trung vào phương pháp phát hiện botnet dựa trên phân cụm không giám sát, sử dụng K-Means trên dữ liệu CTU-13 để nhóm các mẫu lưu lượng bất thường. Mô hình đạt tỷ lệ phát hiện 93.7% và giảm 15% thời gian xử lý so với các phương pháp phân loại truyền thống. Tuy nhiên, phương pháp này gặp khó khăn trong việc phân biệt botnet với lưu lượng bình thường trong các kịch bản phức tạp, và hiệu suất phụ thuộc mạnh vào việc chọn số lượng cụm (k) ban đầu.

- **Nguyen et al. (2022)** trong bài báo "**Botnet Detection Using Ensemble Learning on CTU-13 Dataset**" đã đưa ra một kỹ thuật phát hiện botnet dựa trên học máy kết hợp (ensemble learning), sử dụng các thuật toán như Random Forest và Gradient Boosting để phân tích lưu lượng mạng trên bộ dữ liệu CTU-13. Nghiên cứu tập trung vào các đặc trưng như tần suất gói tin, thời gian giữa các gói, và số lượng kết nối đến IP lạ để nhận diện các botnet như Neris và Rbot, thường thực hiện các cuộc tấn công DDoS hoặc spam. Kết quả cho thấy mô hình đạt độ chính xác 95.8% và F1-score 94.6%, với khả năng giảm dương tính giả đáng kể so với các phương pháp đơn lẻ. Tuy nhiên, hạn chế của nghiên cứu nằm ở việc phụ thuộc vào dữ liệu có nhãn, gây khó khăn khi áp dụng cho các botnet mới chưa được ghi nhận, và thời gian xử lý tăng khi số lượng đặc trưng lớn, đòi hỏi tối ưu hóa thêm để triển khai thời gian thực.
- **Hoang et al. (2023)** trong bài báo "**Improving Botnet Detection with Random Forest and Feature Engineering on CTU-13**" đã giới thiệu một phương pháp nâng cao phát hiện botnet dựa trên thuật toán Random Forest, kết hợp kỹ thuật trích xuất đặc trưng trên bộ dữ liệu CTU-13. Nghiên cứu tập trung vào các đặc trưng được cải tiến như tỷ lệ gói tin outbound/inbound, thời gian giữa các kết nối, và số lượng cổng bất thường để nhận diện các botnet như Zeus và Rbot. Mô hình đạt độ chính xác 96.0% và độ đặc hiệu 94.5% sau khi điều chỉnh tham số như số cây ( $n\_estimators = 200$ ), vượt qua các phương pháp cơ sở. Tuy nhiên, hạn chế của nghiên cứu là mô hình gặp khó khăn khi xử lý dữ liệu thời gian thực với lưu lượng lớn, và hiệu suất giảm khi botnet sử dụng kỹ thuật ngụy trang, cần tích hợp thêm các phương pháp học không giám sát.
- **Joshi và Ranjan et.al.(2021)** trong bài báo "**Botnet Detection Using Machine Learning Algorithms**" đã gợi ý một khung phát hiện botnet sử dụng các thuật toán học máy. Họ đã sử dụng bốn thuật toán học máy, bao gồm: KNN, Logistic Regression, SVM và Decision Tree. Bộ dữ liệu được sử dụng trong các thí nghiệm của họ là CTU-13, nhưng kịch bản cụ thể không được đề cập. Họ đã sử dụng thuật toán chọn lọc đặc trưng và đã chọn sáu đặc trưng cho các thí nghiệm của mình. Tuy nhiên, thuật toán chọn lọc đặc trưng được sử dụng trong các thí nghiệm của họ không được đề cập. Họ đã đánh giá hiệu suất của mô hình bằng cách sử dụng độ chính xác, độ chính xác dự đoán, độ nhạy và điểm F1. Tuy nhiên, phương pháp của họ được đánh giá trên bộ dữ liệu CTU-13 kịch bản 2, nơi chỉ có các nút bot IRC có sẵn. Do đó, không rõ phương pháp này hoạt động tốt như thế nào trên một bộ dữ liệu chứa các nút bot P2P.
- **Ibrahim et al.(2021)**, trong bài báo "**Multilayer Framework for Botnet Detection Using Machine Learning Algorithms**" đã so sánh ba bộ phân loại có giám sát: KNN, SVM và Multilayer Perceptron. Họ đã áp dụng một thuật toán chọn lọc đặc trưng và đã chọn ra năm đặc trưng. Họ đã đánh giá mô hình của mình bằng cách sử dụng các giá trị độ chính xác (accuracy), độ chính xác dự đoán (precision), độ nhạy (Recall), điểm

F1 (F1-score) và tỷ lệ âm tính giả (FNR) trong các kịch bản khác nhau của bộ dữ liệu CTU-13. Tuy nhiên, hiệu suất của phương pháp của họ trong kịch bản chứa botnet P2P không đạt yêu cầu.

Các nghiên cứu trên cho thấy phát hiện botnet bằng học máy mang lại hiệu quả cao, với độ chính xác từ 95% đến 99.77%. Tuy nhiên, các hạn chế phổ biến bao gồm yêu cầu tài nguyên tính toán lớn, phụ thuộc vào dữ liệu huấn luyện, khó diễn giải kết quả, và hiệu suất giảm khi xử lý botnet mã hóa hoặc botnet mới.

### 3.2 Mô hình đề xuất



Hình 3.3: Sơ đồ quy trình phát hiện botnet

Hình 3.3 minh họa quy trình phát hiện botnet, bao gồm các giai đoạn: phân tích dữ liệu từ các luồng NetFlow của bộ CTU-13, trích xuất đặc trưng quan trọng như số gói tin và tổng byte, lựa chọn đặc trưng để loại bỏ dữ liệu không cần thiết, huấn luyện mô hình bằng các thuật toán như Decision Tree và Random Forest, đánh giá hiệu suất qua các chỉ số Accuracy và F1-Score, và cuối cùng triển khai đầu ra để phát hiện botnet trong thời gian thực.

### 3.3 Hướng tiếp cận

#### Chọn tập dữ liệu

Phần đầu tiên của phương pháp là thu thập dữ liệu luồng mạng (*traffic flow*). Điều này có thể được thực hiện bằng cách lấy dữ liệu thực tế từ một tổ chức và trích xuất các luồng NetFlow. Các nhà nghiên cứu sử dụng nhiều bộ dữ liệu khác nhau để kiểm chứng phương pháp của họ. Tuy nhiên, các bộ dữ liệu này có một số khuyết điểm. Một số bộ dữ liệu không có sẵn công khai. Trong trường hợp không có dữ liệu thực tế, ta có thể sử dụng các bộ dữ liệu công khai như CTU-13, KDDCUP99, CIC-IDS-2017...

Chúng em chọn sử dụng bộ dữ liệu **CTU-13** vì tính khả dụng cao và vì nó đã được sử dụng rộng rãi trong các nghiên cứu tương tự. Phần 4.1 trình bày chi tiết hơn về tập dữ liệu. Dữ liệu sau đó được phân tích thống kê để xác định các đặc trưng phổ biến.

#### Trích xuất đặc trưng

Sau khi chọn tập dữ liệu, bước tiếp theo là xác định và trích xuất các đặc trưng. Đây là bước rất quan trọng trong toàn bộ quy trình. Dữ liệu NetFlow bao gồm nhiều đặc trưng phân loại (*categorical*) cần được mã hóa thành giá trị số hoặc nhị phân giúp xử lý dữ liệu số nhằm tối ưu hóa hiệu suất và tương thích với các mô hình học máy. Tuy nhiên, điều này có thể làm ma trận đầu vào quá lớn gây lỗi bộ nhớ.

#### Lựa chọn đặc trưng

Khi đã chọn được bộ dữ liệu, bước tiếp theo là xác định và trích xuất các đặc trưng – đây là một phần quan trọng trong phương pháp của chúng tôi. Dữ liệu NetFlow bao gồm các đặc trưng dạng phân loại (*categorical*) cần được mã hóa thành các giá trị số. Một phương pháp phổ biến để chuyển đổi các giá trị phân loại thành đặc trưng số là sử dụng one-hot encoding. Mặc dù điều này làm tăng dung lượng bộ nhớ cần thiết để lưu trữ, phần cứng của chúng tôi vẫn có khả năng xử lý được vì trong bộ dữ liệu có 7 đặc trưng dạng phân loại.

Vì dữ liệu mất cân bằng phương pháp luận này đã sử dụng xác thực chéo (*cross-validation*) *k*-fold phân tầng để đạt được hiệu suất chính xác hơn cho bộ phân loại. Phương pháp này hoạt động bằng cách chia tập dữ liệu thành *k* fold, trong đó *k* - 1 fold được sử dụng để huấn luyện và fold còn lại được sử dụng để kiểm tra. Sau đó, quá trình này được lặp lại *k* lần sao cho mỗi fold được sử dụng để kiểm tra ít nhất một lần và để huấn luyện *k* - 1 lần.

#### Phát hiện Botnet

Kế đến, đó là lần lượt huấn luyện các mô hình Decision Tree, Random Forest và MLP. Sau cùng là kiểm tra mô hình để xác định khả năng phát hiện lưu lượng mạng botnet trong bộ dữ liệu **CTU-13**. Hiệu quả của các mô hình được đánh giá dựa trên hiệu số đánh giá.

## CHƯƠNG 4. THỰC NGHIỆM

### 4.1 Tập dữ liệu CTU-13

CTU-13 là một tập dữ liệu công khai bao gồm 13 kịch bản với số lượng máy tính bị nhiễm khác nhau và 7 họ botnet thật thực hiện nhiều hoạt động độc hại khác nhau như tấn công DDoS, quét cổng, gian lận nhấp chuột, gửi thư rác, v.v. Bảng 4.1 tóm tắt thời lượng dữ liệu, số luồng, số bot và loại bot trong mỗi tập con.

Bảng 4.1: Bảng thông tin CTU-13

ID	Period (hrs)	Flows	Bot	Total	Activity
1	6.15	2,824,637	Neris	1	IRC, SPAM, CF
2	4.21	1,808,123	Neris	1	IRC, SPAM, CF
3	66.85	4,710,639	Rbot	1	IRC, PS
4	4.21	1,121,077	Rbot	1	IRC, DDoS
5	11.63	129,833	Virut	1	SPAM, PS
6	2.18	558,920	Menti	1	PS
7	0.38	114,078	Sogou	1	HTTP
8	19.5	2,954,231	Murlo	1	PS
9	5.18	2,753,885	Neris	10	IRC, SPAM, CF, PS
10	4.75	1,309,792	Rbot	10	IRC, DDoS
11	0.26	107,252	Rbot	3	IRC, DDoS
12	1.21	325,472	NSIS.ay	3	IRC, P2P
13	16.36	1,925,150	Virut	1	HTTP, SPAM, PS

Bảng 4.1 trình bày thông tin tổng quan về các hoạt động của botnet trong khoảng thời gian từ CTU-13, với các cột chính bao gồm: "ID" là mã định danh của từng bản ghi, "Period (hrs)" là thời gian hoạt động tính bằng giờ, "Flows" là số lượng luồng dữ liệu, "Bot" là tên của botnet cụ thể, "Total" là số lượng bot trong mạng và "Activity" là các loại hoạt động mà botnet thực hiện (như IRC, SPAM, CF, DDoS, PS, HTTP, P2P). Ví dụ, bản ghi ID 1 cho thấy botnet Neris hoạt động trong 6.15 giờ với 2.824.637 luồng dữ liệu, có 1 bot và thực hiện các hoạt động IRC, SPAM, CF.

Trong tập dữ liệu CTU-13, lưu lượng mạng botnet chỉ chiếm khoảng 1.5% tổng lưu lượng mạng. Điều này cho thấy tập dữ liệu này cực kỳ mất cân bằng. Vì phương pháp của chúng em tập trung vào việc phát hiện các máy bị nhiễm trong mạng thay vì các luồng bị nhiễm.

Thông tin về tập dữ liệu bao gồm:

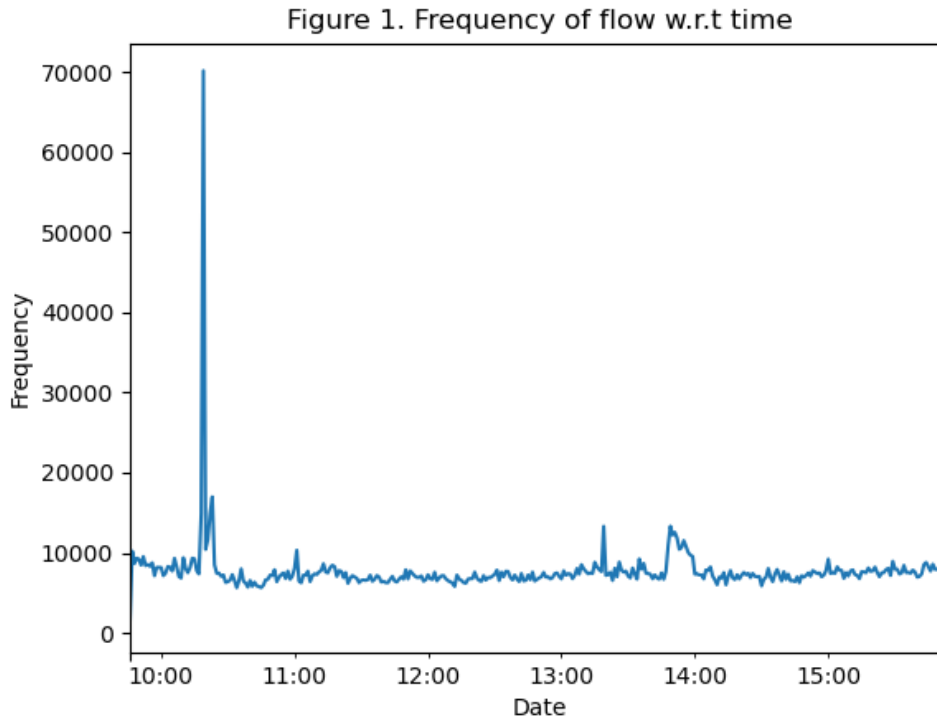
- Tác giả: nhóm Stratosphere IPS được tại Đại học CTU
- Năm: 2011
- Số lượng bản ghi: 20,643,076
- Số lượng đặc trưng: 15
- Dung lượng: 74.3GB

Bảng 4.2: Bảng thông tin của các đặc trưng

Đặc trưng	Thông tin
StartTime	Thời gian ghi nhận dữ liệu, chỉ ra thời điểm chúng được ghi lại (h:m:s)
Dur	Thời lượng của luồng mạng liên quan (tính bằng giây)
Proto	Giao thức được sử dụng: esp, igmp, pim, udp, tcp, rtp, ipx, ipv6-icmp, ipv6, spx, unas, rdp, arp, udt, rarp, icmp
SrcAddr	Địa chỉ IP nguồn
Sport	Số cổng tại nguồn
Dir	Hướng của luồng mạng: -> (ra), <- (vào), <-> (đi hai chiều), <?> (chưa biết), who (tra cứu danh tính), <? (nguồn chưa biết), ?> (đích chưa biết)
DstAddr	Địa chỉ IP đích
Dport	Số cổng tại đích
State	Trạng thái giao dịch liên quan đến giao thức
sTos	Loại dịch vụ nguồn
dTos	Loại dịch vụ đích
TotPkts	Tổng số gói tin được truyền
TotBytes	Tổng số byte được truyền
SrcBytes	Số byte được truyền từ nguồn đến đích
Label	Nhãn bao gồm các luồng là bình thường, nền (background), hoặc botnet

Bảng 4.2 mô tả các đặc trưng của dữ liệu lưu lượng mạng được sử dụng trong bộ dữ liệu CTU-13. Bảng cung cấp một khung cơ bản để hiểu cấu trúc dữ liệu, từ đó hỗ trợ các bước tiền xử lý và huấn luyện mô hình.

Sau đây là mô tả chi tiết của các đặc trưng:



Hình 4.4: Tần suất của các NetFlows theo thời gian

Trong hình 4.4 là thời gian bắt đầu của các NetFlows được phân bố đồng đều trong suốt quá trình thực nghiệm, ngoại trừ vào khoảng 10:20 khi có một đỉnh của số lượng NetFlows bắt đầu. Hình này cho thấy rằng phần lớn các giao tiếp chỉ tồn tại trong một khoảng thời gian rất ngắn (ít hơn 1 ms đối với một nửa trong số đó). Cũng có thể nhận thấy các đỉnh tần suất xuất hiện mỗi khi đạt đến một phút tròn.

**StartTime:** Ngày bắt đầu từ 09:46:53 đến 15:54:07 10/08/2011.

**Dur:** Thời gian tồn tại của luồng giao tiếp được tính bằng giây.

**Proto:** Giao thức vận chuyển được sử dụng phản ánh sự đa dạng trong lưu lượng mạng.

**SrcAddr:** Địa chỉ IP nguồn với tổng cộng là 542.093 địa chỉ cho thấy tập trung vào một số IP phổ biến, với phần còn lại phân bố rải rác.

**Sport:** Cổng nguồn với tổng số là 64.752.

**Dir:** Hướng của luồng bao gồm 78% song công và 22% từ nguồn đến đích.

**DstAddr:** Địa chỉ IP đích gồm có 119.296 địa chỉ cho thấy sự phân tán cao.

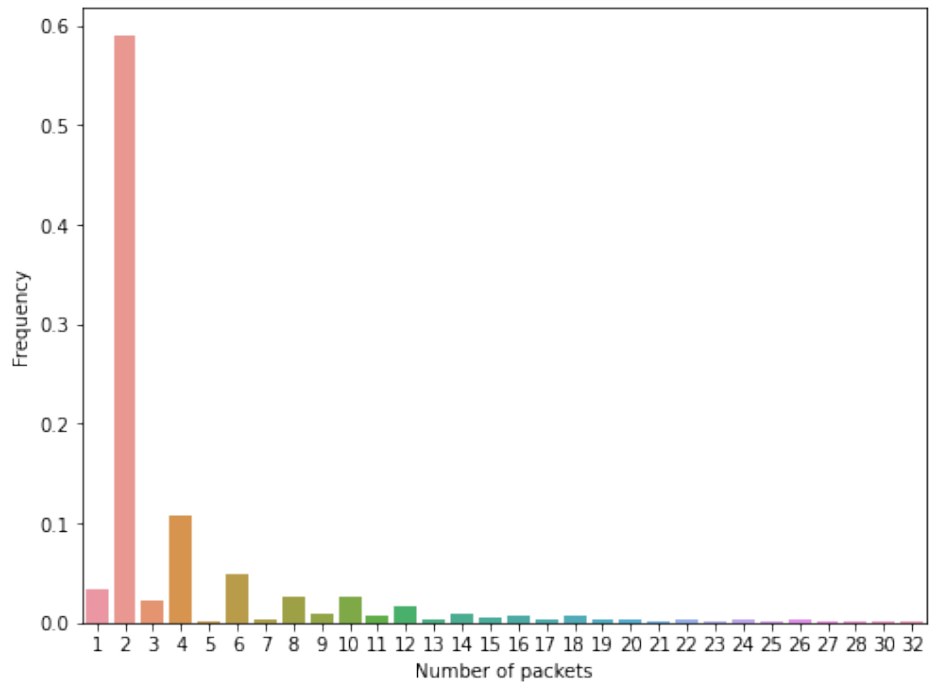
**Dport:** Cổng đích với tổng số là 73.786 .

**State:** Trạng thái giao dịch với 230 trạng thái khác nhau.

**sTos:** Giá trị TOS của nguồn là 0 đối với 99,9% các giao tiếp.

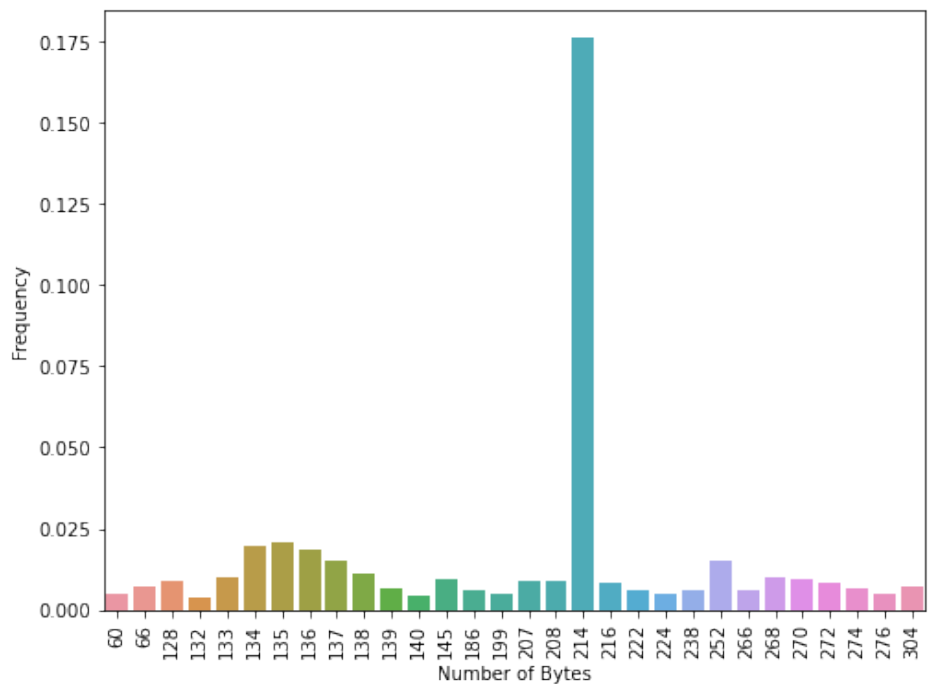
**dTos:** Giá trị TOS của đích là 0 đối với 99,99% các giao tiếp.

**TotPkts:** Tổng số gói tin của giao dịch với tối thiểu là 1 gói và tối đa là 2.686.731 gói.



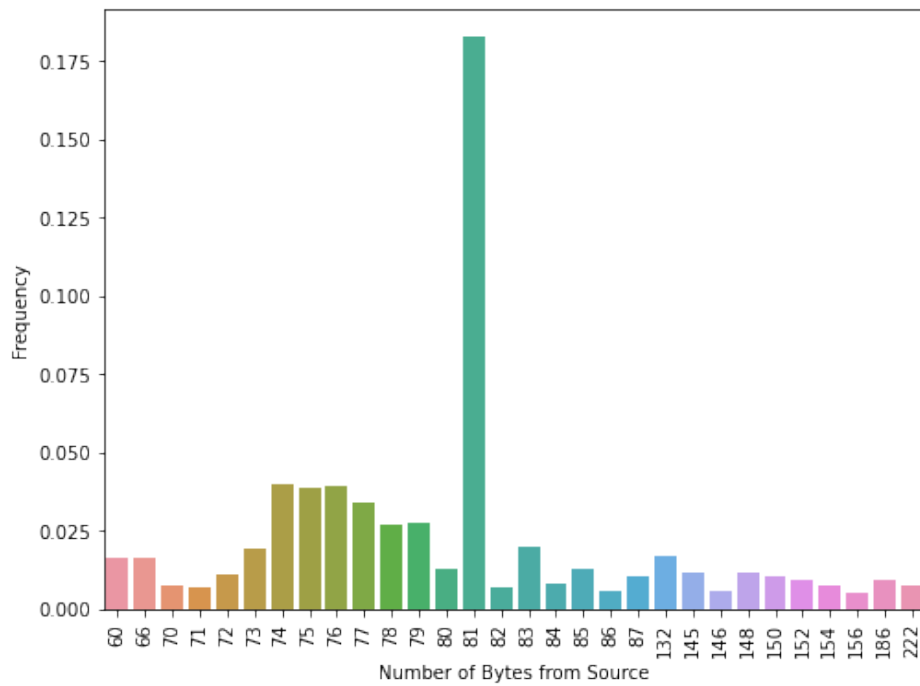
Hình 4.5: Các giá trị số gói tin phổ biến nhất

**TotBytes:** Tổng số byte của giao dịch với tối thiểu là 60 byte và tối đa là 2.689.640.464 byte.



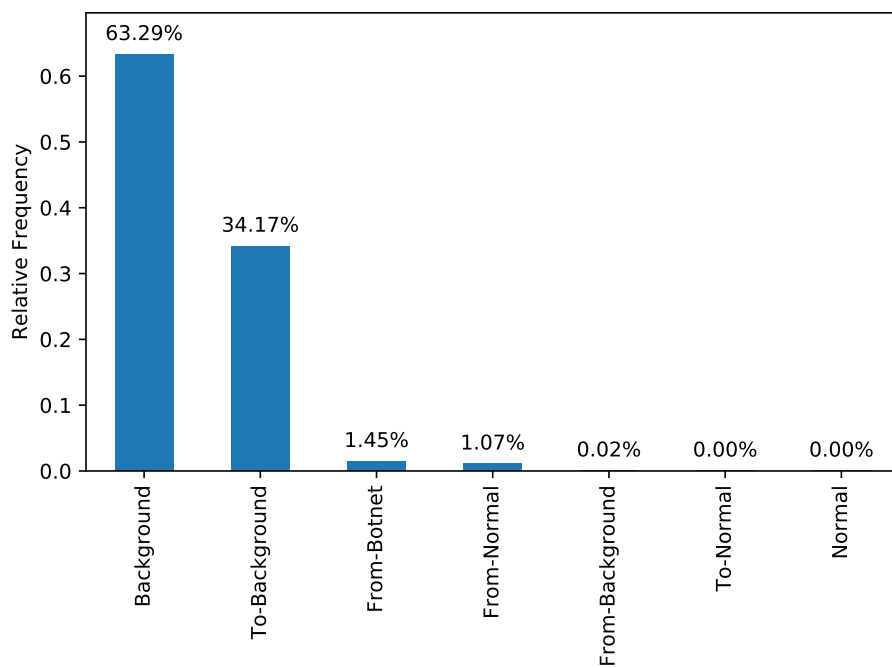
Hình 4.6: Các giá trị byte phổ biến nhất

**SrcBytes:** Tổng số byte từ nguồn với tối thiểu là 0 byte và tối đa là 2.635.366.235 byte.

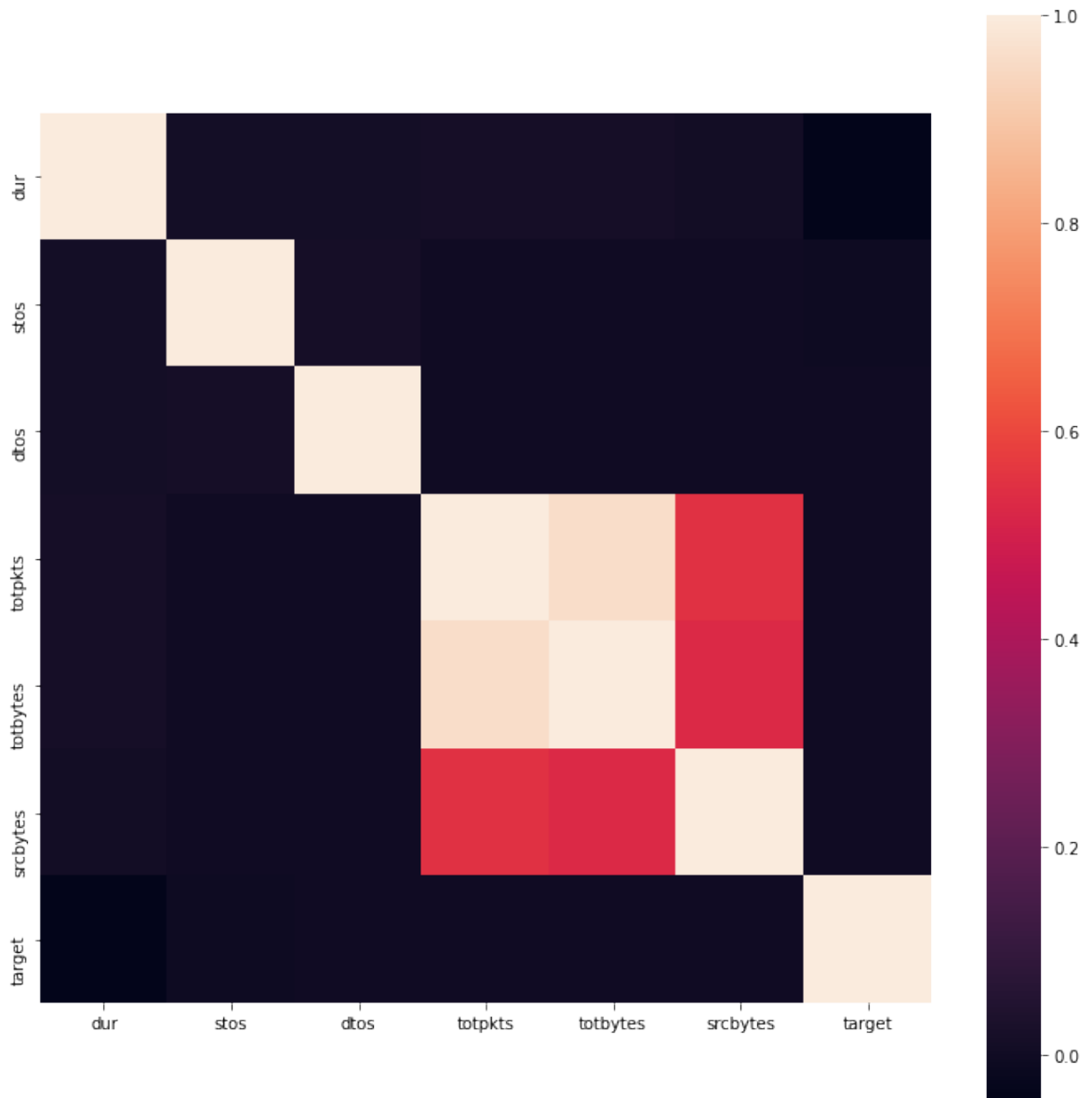


Hình 4.7: Các giá trị byte phổ biến nhất

**Label:** Phân bố 113 nhãn thể hiện sự mất cân bằng, với lớp botnet chiếm thiểu số (khoảng 2%).



Hình 4.8: Biểu đồ phân bố của 113 nhãn

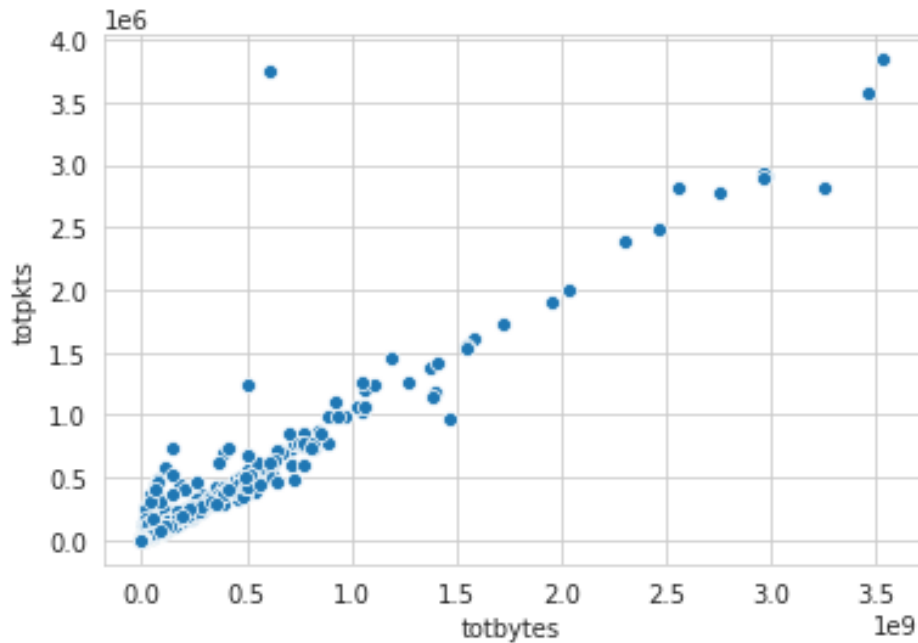


Hình 4.9: Bản đồ nhiệt tương quan giữa các đặc trưng

Trong hình 4.9 chúng ta có thể suy ra những đặc trưng nào có tương quan cao nhất với nhau.

Các ô màu be nhạt biểu thị các tương quan dương cao nhất, trong khi các ô màu đen biểu thị các tương quan âm cao nhất. Tương quan cao được phát hiện ở tất cả các đặc trưng TotPkts, TotBytes nhưng điều này không bất ngờ vì chúng được suy ra từ các phép đo thống kê của cùng một cột trong tập dữ liệu thô.

Liên quan đến phương pháp lọc, chúng ta có thể sử dụng bản đồ nhiệt để xác định các đặc trưng có tương quan cao trong tập con chứa các đặc trưng tốt nhất và chọn ra những đặc trưng đại diện tốt nhất.



Hình 4.10: Biểu đồ phân tán của totpkts và totbytes

Đa số các điểm tập trung ở vùng totpkts thấp (dưới 1) và totbytes thấp (dưới 1), tạo thành một đám mây dày đặc ở góc dưới bên trái.

Khi totpkts tăng, totbytes cũng có xu hướng tăng, thể hiện qua xu hướng đi lên của các điểm từ trái sang phải. Điều này cho thấy một tương quan dương giữa hai biến.

Biểu đồ này có tác dụng chính là trực quan hóa mối quan hệ tuyến tính dương mạnh giữa totpkts và totbytes, hỗ trợ trong quá trình EDA để hiểu cấu trúc dữ liệu và đưa ra quyết định về việc chọn đặc trưng hoặc xử lý ngoại lệ. Với xu hướng tăng rõ ràng và hệ số tương quan cao, nó xác nhận rằng hai đặc trưng này có thể bổ trợ hoặc trùng lặp nhau.

## 4.2 Cài đặt thực nghiệm

Mô hình đề xuất được chạy trên máy tính với CPU Intel Core i5-10210U với 16GB RAM. Hệ điều hành sử dụng là Windows 10 với trình soạn thảo VS Code sử dụng Python 3.13.2, Keras, Numpy và Matplotlib.

## 4.3 Trích xuất đặc trưng

CTU-13 Dataset chứa các cột kiểu số nguyên, số thực, đối tượng và cột phân loại. Các cột như Start Time, Source và Destination IP address, cũng như Source và Destination port có độ đa dạng (cardinality) cao, trong khi các cột như sTos và dTos có độ đa dạng rất thấp. Để giải quyết các vấn đề này, cần thực hiện tiền xử lý trên tập dữ liệu CTU-13 nhằm làm cho nó tương thích với việc huấn luyện và dự đoán bằng máy học.

### One Hot Encoding

Cột phân loại là cột chứa các danh mục và có độ đa dạng tối thiểu. Trong tập dữ liệu CTU-13, có 4 cột được xác định là cột phân loại, bao gồm 'Dir', 'Proto', 'sTos' và 'dTos'. Cột 'Dir' có 7 danh mục, 'Proto' có 15 danh mục, 'sTos' có 6 danh mục và 'dTos' có 5 danh mục. One Hot Encoding là quá trình chuyển đổi cột phân loại thành các vectơ gồm 0 và 1. Một cột có 2 hoặc 3 lớp sẽ có độ dài vectơ tương ứng là 2 hoặc 3. Việc chuyển đổi một cột có 5 lớp thành vectơ 0 và 1 với độ dài 5 có thể dẫn đến vấn đề đa cộng tuyến (multicollinearity). Vấn đề này xảy ra do cung cấp thông tin dư thừa và các biến dự đoán có độ tương quan cao. Để giải quyết, có thể bỏ một lớp đã được mã hóa One Hot của một cột. Do đó, một cột có 5 danh mục sẽ có vectơ độ dài 4 thay vì 5. Trong trường hợp CTU-13, số cột được mã hóa One Hot cho 4 cột phân loại sẽ là 29 cột.

### Label Encoding

Cột mục tiêu trong tập dữ liệu CTU-13 là cột Label. Cột Label có thể được phân loại thành ba lớp: background, botnet, và normal. Để bất kỳ mô hình máy học nào hoạt động chính xác, tất cả các biến dự đoán và biến phản hồi cần phải là số. Để thực hiện điều này, cột Label cần được chuyển đổi thành số. Label Encoding là quá trình gán nhãn cho các danh mục chuỗi bắt đầu từ 0. Trong trường hợp CTU-13, các nhãn background, botnet và normal được ánh xạ lần lượt thành 0, 1 và 2. Tuy nhiên, sự phân bố của các nhãn này không cân bằng, như đã thảo luận trong phần V.C.

### Dropping Columns

Các cột có độ đa dạng rất cao và các cột không thể chuyển đổi thành giá trị số có thể bị loại bỏ. Trong tập dữ liệu CTU-13, các cột như 'StartTime', 'SrcAddr', 'Sport', 'DstAddr', 'Dport', và 'State' đã bị bỏ để giảm số lượng đặc trưng trong tập dữ liệu.

### Kỹ thuật Scaling

Các mô hình máy học thường gặp vấn đề khi thực hiện tính toán trên các số lớn do lượng tính toán bên trong rất lớn. Nếu không được xử lý đúng cách, điều này có thể dẫn đến rò rỉ bộ nhớ, thời gian huấn luyện tăng, thời gian dự đoán chậm và mô hình không mở rộng tốt khi số lượng đặc trưng tăng lên. Một lần chạy ban đầu của mô hình cơ sở trên các giá trị chưa được chuẩn hóa đã dẫn đến việc huấn luyện bị dừng do cạn kiệt tài nguyên bộ nhớ. Để xử lý các trường hợp như vậy, cần phải chuẩn hóa các giá trị này để nằm trong một phạm vi nhất định. Vấn đề này thường xảy ra với các cột có nhiều giá trị và phạm vi lớn. Bảng 4 cho thấy phạm vi của 4 cột như vậy từ tập dữ liệu CTU-13. Các cột này đã được chuẩn hóa để nằm trong khoảng từ 0 đến 1.

## 4.4 Lựa chọn đặc trưng

Sử dụng Threshold giúp nhận diện các mẫu mà mô hình không chắc chắn trong việc phân loại. Những mẫu này có thể thuộc về lớp mới (chưa được huấn luyện) hoặc là dữ liệu bất thường. Điều này rất hữu ích để phát hiện các loại botnet mới chưa từng xuất hiện trong dữ

liệu huấn luyện. Cải thiện độ chính xác của phân loại bằng cách không gán nhãn cụ thể cho các mẫu không chắc chắn (nhãn "other"), mô hình tránh được việc phân loại sai, từ đó tăng độ tin cậy cho các dự đoán còn lại đồng thời cũng tăng khả năng tổng quát hóa của mô hình.

Cách đơn giản và dễ nhất để loại bỏ các cột dư thừa là xóa các cột có nhiều giá trị null. Trong trường hợp tập dữ liệu CTU-13, các cột có giá trị null bao gồm 'Sport', 'Dport', 'State', 'sTos' và 'dTos'. Các cột 'Sport', 'Dport' và 'State' đã bị loại bỏ do bản chất của chúng không thể chuyển đổi thành giá trị số. Các giá trị null trong 'sTos' và 'dTos' đã được thay thế bằng giá trị trung vị. Trong tập dữ liệu CTU-13, giá trị null được tìm thấy trong các cột 'Sport', 'Dport', 'State', 'sTos' và 'dTos'. Rõ ràng, 'Sport', 'Dport' và 'State' đã bị loại bỏ. Giá trị null trong cột 'sTos' đã được thay thế bằng giá trị xuất hiện nhiều nhất (mode) của cột, là 0.0. Cột 'dTos', như được thấy trong phần ngưỡng phương sai tiếp theo, đã bị loại bỏ.

#### 4.5 Số liệu đánh giá

Để so sánh kết quả của hai phương pháp, cần xác định một số chỉ số để đánh giá hiệu suất của các phương pháp đó. Cách phổ biến là xem xét số lượng dương tính giả nghĩa là số lượng lưu lượng mạng bình thường bị phân loại nhầm thành botnet, âm tính giả tức là số lượng lưu lượng bất thường bị gán nhầm là lưu lượng bình thường và độ chính xác của mô hình phân loại đa lớp. Mô hình này dùng để xác định loại botnet cụ thể sau khi lưu lượng bất thường được phát hiện. Để thực hiện điều này, có thể sử dụng năm chỉ số đánh giá.

- the Recall:  $R = \frac{TP}{TP + FN}$
- the Precision:  $P = \frac{TP}{TP + FP}$
- the  $f_1$  Score:  $f_1 = \left( \frac{R^{-1} + P^{-1}}{2} \right)^{-1} = 2 \times \frac{R \times P}{R + P}$
- AUC score: Diện tích dưới đường cong ROC
- Accuracy:  $Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$
- FAR là số lượng cảnh báo sai (false alarms) so với tổng số cảnh báo hoặc tín hiệu được phát ra được tính bằng công thức:  $FAR = \frac{1}{2} \left( \frac{FP}{FP + TN} + \frac{FN}{FN + TP} \right)$

Chú thích:

- TP: Số lượng botnet (hoặc sự kiện) được hệ thống phát hiện đúng là botnet thực sự.
- FN: Số lượng botnet thực sự nhưng bị hệ thống bỏ sót, nghĩa là không được phát hiện.
- FP: lưu lượng bình thường nhưng bị hệ thống nhầm là botnet (báo động sai).
- TN: lưu lượng bình thường được hệ thống phân loại đúng là không phải botnet.

Bảng 4.3: Đánh giá chỉ số AUC

S/No.	AUC Range	Classifications
1	$0.90 < AUC \leq 1.00$	Very good
2	$0.80 < AUC \leq 0.90$	Good
3	$0.70 < AUC \leq 0.80$	Poor
4	$0.60 < AUC \leq 0.70$	Very poor

## 4.6 Kết quả đạt được

### 4.6.1 Đối với mô hình Decision Tree

Bảng 4.4: Kết quả sau khi thực nghiệm mô hình Decision Tree

Dataset	Model	AUC	F1-Score	False Alarm Rate
Train	DT	0.975	0.954	0.024
Test	DT	0.965	0.936	0.034

Bảng 4.5: Báo cáo phân loại của mô hình Decision Tree

	Precision	Recall	F1-Score	Support
Lớp 0	1.00	1.00	1.00	4097792
Lớp 1	0.93	0.94	0.936	79049
Accuracy			0.9976	4176841
Macro Avg	0.97	0.97	0.97	4176841
Weighted Avg	0.9976	0.9976	0.9976	4176841

Báo cáo phân loại cho thấy:

Lớp 0 (dự đoán là lưu lượng bình thường): Precision, recall, và F1-score đều đạt 1.00, cho thấy mô hình phân loại hoàn hảo cho lớp này. Với support là 4,097,792, đây là lớp chiếm đa số là lưu lượng mạng bình thường.

Lớp 1 (dự đoán là botnet): Precision đạt 0.93, recall đạt 0.94, và F1-score đạt 0.94, với support là 79,049, cho thấy mô hình hiệu quả trong việc phát hiện lưu lượng botnet, dù đây là lớp thiểu số.

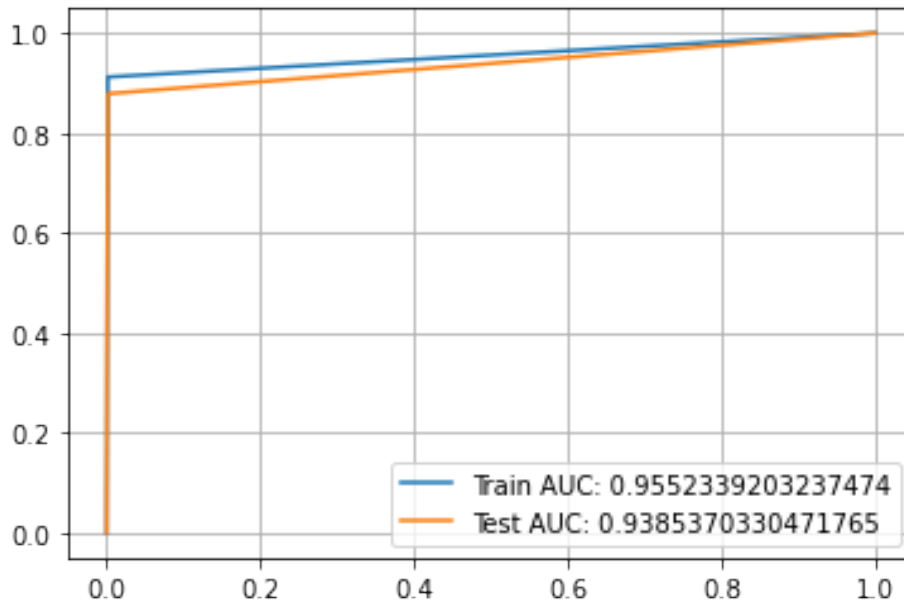
Độ chính xác tổng thể: Đạt 99.76%, phản ánh hiệu suất cao, nhưng cần lưu ý rằng với tập dữ liệu mất cân bằng, accuracy có thể không phản ánh đầy đủ hiệu suất trên lớp thiểu số.

Macro Avg và Weighted Avg: Macro Avg (0.97) cho thấy hiệu suất trung bình không trọng số giữa hai lớp, trong khi Weighted Avg (0.9976) tính đến số lượng support, gần với accuracy do lớp 0 chiếm ưu thế.

#### 4.6.2 Đối với mô hình Random Forest

Bảng 4.6: Kết quả sau khi thực nghiệm mô hình Random Forest

Dataset	Model	AUC	F1-Score	False Alarm Rate
Train	RF	0.955	0.925	0.044
Test	RF	0.938	0.896	0.061



Hình 4.11: Đánh giá AUC với mô hình Random Forest

Bảng 4.7: Báo cáo phân loại của mô hình Random Forest

	Precision	Recall	F1-Score	Support
Lớp 0	0.9976	0.9984	0.9980	4,097,130
Lớp 1	0.9144	0.8787	0.8962	79,711
Accuracy			0.9961	4,176,841
Macro Avg	0.9560	0.9386	0.9471	4,176,841
Weighted Avg	0.9961	0.9961	0.9961	4,176,841

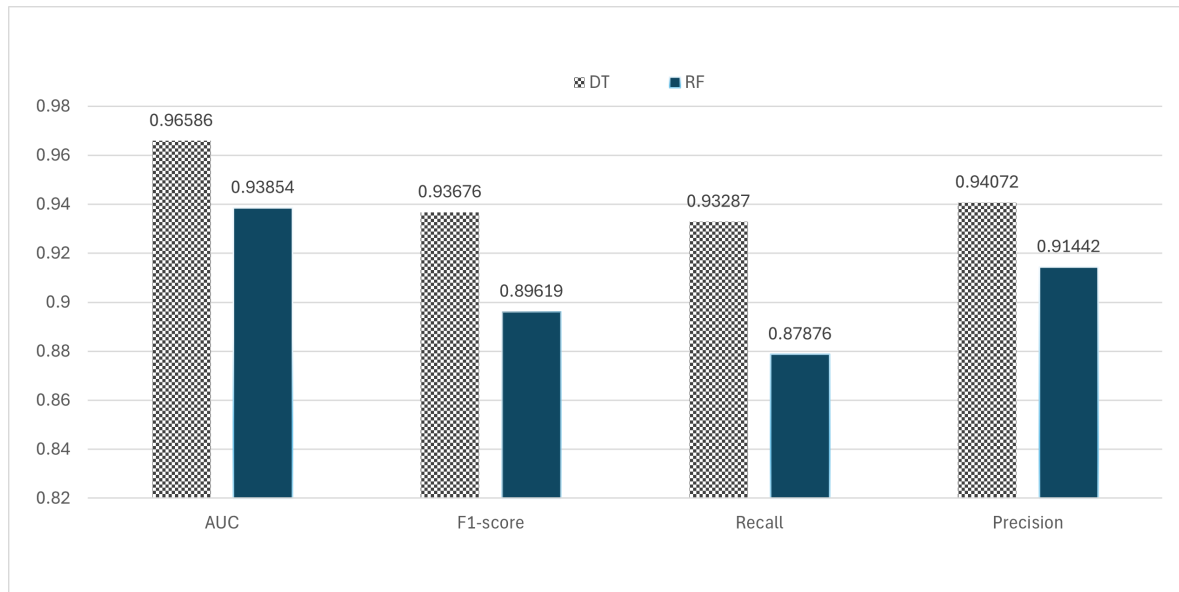
Lớp 0: Precision (0.9976), recall (0.9984) và F1-score (0.9980) đều rất cao, cho thấy mô hình gần như hoàn hảo trong việc phân loại lưu lượng bình thường. Với support là 4,097,130, đây là lớp chiếm đa số trong tập dữ liệu.

Lớp 1: Precision đạt 0.9144, recall đạt 0.8787 và F1-score đạt 0.8962, với support là 79,711. Điều này cho thấy mô hình hiệu quả trong việc phát hiện botnet, nhưng có một

số trường hợp botnet bị bỏ sót (false negatives) và một số lưu lượng bình thường bị phân loại sai thành botnet.

Độ chính xác tổng thể (Accuracy): Đạt 99.61%, phản ánh hiệu suất cao. Tuy nhiên, do tập dữ liệu mất cân bằng (lớp 0 chiếm ưu thế), độ chính xác này có thể bị ảnh hưởng bởi hiệu suất trên lớp 0.

Macro Avg và Weighted Avg: Macro Avg (0.9471) tính trung bình không trọng số của F1-score giữa hai lớp, cho thấy hiệu suất cân bằng giữa các lớp. Weighted Avg (0.9961) phản ánh hiệu suất tổng thể, gần với độ chính xác do lớp 0 chiếm ưu thế.



Hình 4.12: So sánh kết quả của hai mô hình

#### 4.6.3 Đối với mô hình MLP

Mô hình Mạng nơ-ron truyền thẳng nhiều lớp được sử dụng để huấn luyện các mô hình phân loại. Cụ thể, mô hình MLP này gồm 6 tầng ẩn, với số lượng nút ẩn (hidden nodes) tương ứng cho từng tầng là: 128, 1024, 512, 128, 128, 10. Việc sử dụng nhiều tầng với số lượng nút lớn cho phép mô hình học được các đặc trưng phức tạp và tinh vi trong dữ liệu.

Hàm kích hoạt ReLU được sử dụng cho tất cả các tầng ẩn nhằm tăng khả năng học phi tuyến và giảm thiểu hiện tượng vanishing gradient, từ đó giúp mô hình học nhanh và hiệu quả hơn. Tại tầng đầu ra, hàm kích hoạt Softmax được sử dụng để chuẩn hóa các giá trị đầu ra thành phân phối xác suất, đảm bảo rằng tổng xác suất các lớp bằng 1 và thuận tiện cho việc xác định lớp đầu ra cuối cùng (tức là các loại botnet).

Mô hình MLP này được huấn luyện trên tập dữ liệu lưu lượng mạng và được thiết lập để phân loại thành bốn lớp tương ứng với các loại botnet phổ biến: Virut, Neris, Murlo và Rbot. Mỗi lớp đại diện cho một loại hành vi tấn công cụ thể, với số lượng mẫu khác nhau trong tập dữ liệu.

Bảng 4.8: Báo cáo phân loại của mô hình MLP

	<b>Precision</b>	<b>Recall</b>	<b>F1-Score</b>	<b>Support</b>
Lớp 0	0.74	0.58	0.65	12370
Lớp 1	0.91	0.87	0.89	55588
Lớp 2	1.00	0.98	0.99	1805
Lớp 3	0.58	0.88	0.70	9885
Accuracy			0.83	79648
Macro Avg	0.81	0.83	0.81	79648
Weighted Avg	0.85	0.83	0.83	79648

Với độ chính xác tổng thể 83%, mô hình cho thấy khả năng phân loại tốt, đặc biệt trên lớp Neris (F1-score 0.89) và Murlo (F1-score 0.99). Tuy nhiên, hiệu suất trên lớp Virut (F1-score 0.65) và Rbot (F1-score 0.70) thấp hơn.

## CHƯƠNG 5. KẾT LUẬN

### 5.1 Những kết quả đạt được

Trong các nghiên cứu được đề xuất, thì dưới đây sẽ là kết quả của toàn bộ quá trình dự án này so với các bài báo liên quan.

Bảng 5.9: Bảng so sánh kết quả với các báo cáo liên quan

Sl. No.	Tác giả dự án liên quan	Độ chính xác
1	Zhang et al. (2022)	94.3%
2	Padmavathia và Muthukumar (2022)	99.98%
3	Ibrahim et al. (2021)	95.91%
4	Velasco-Mata et al. (2023)	96.50%
5	Wang et al. (2020)	99.94%
6	Joshi và Ranjan (2021)	97%
7	Mô hình đề xuất (MLP)	83.23%
8	Mô hình đề xuất (Decision Tree)	99.76%
9	Mô hình đề xuất (Random Forest)	99.61%

Hiệu suất của các mô hình trong dự án này Decision Tree với độ chính xác 99.76%, đây là mô hình có hiệu suất cao nhất trong dự án này. Random Forest đạt độ chính xác 99.61%, rất gần với mô hình Decision Tree, cho thấy hiệu quả mạnh mẽ của phương pháp học tập tổng hợp dựa trên cây quyết định. Điều này cho thấy các mô hình dựa trên Decision Tree và Random Forest rất hiệu quả cho việc phát hiện botnet trong bộ dữ liệu mà chúng em tìm hiểu.

Multi-Layer Perceptron (MLP): Chỉ đạt 83.23%, thấp hơn đáng kể so với Decision Tree và Random Forest, đồng thời là mô hình có hiệu suất kém nhất trong bảng. Xét thấy độ chính xác thấp nên mô hình này không phù hợp hoặc chưa được tối ưu hóa tốt cho bài toán này. Có thể nguyên nhân là do kích thước tập dữ liệu không đủ lớn, cấu trúc mạng chưa phù hợp hoặc quá trình huấn luyện chưa tối ưu.

Trong quá trình thực hiện dự án "Tiếp cận các phương pháp học máy để phát hiện botnet ẩn trong lưu lượng mạng" thì đây là toàn bộ kết quả mà nhóm em đã đạt được:

- Phần đầu tiên trong phương pháp nghiên cứu là tiến hành thu thập dữ liệu lưu lượng mạng. Việc này có thể được thực hiện bằng cách lấy dữ liệu thực tế từ một tổ chức và trích xuất các luồng.
- Tiếp theo, nhóm em đã tiến hành phân tích, xử lý và xác định các tập dữ liệu cần thiết cho việc huấn luyện mô hình. Việc chọn lựa và xử lý dữ liệu là một bước quan trọng, ảnh hưởng trực tiếp đến hiệu suất của mô hình. Cụ thể, nhóm em đã áp dụng các kỹ

thuật như Variance Threshold để loại bỏ các đặc trưng có phương sai thấp, One Hot Encoding để mã hóa các nhãn phân loại, và k-fold cross-validation để đánh giá hiệu suất mô hình một cách đáng tin cậy trên các tập con dữ liệu..

- Sau đó, nhóm em bắt đầu huấn luyện các mô hình Decision Tree, Random Forest, MLP. Các mô hình này được triển khai nhằm đánh giá hiệu quả trong việc phân loại và phát hiện các luồng dữ liệu botnet.
- Giai đoạn cuối cùng trong phương pháp là kiểm tra hiệu quả của các mô hình đã huấn luyện để xác định khả năng phát hiện lưu lượng mạng botnet trên bộ dữ liệu CTU-13. Hiệu suất của từng mô hình được đánh giá dựa trên các chỉ số như độ chính xác accuracy, precision, recall, F1-score,... Nhờ đó, chúng em có thể đưa ra nhận định khách quan về mô hình nào phù hợp và hiệu quả nhất trong việc phát hiện các hoạt động botnet.

## 5.2 Thuận lợi và hạn chế

### 5.2.1 Thuận lợi

Các phương pháp học máy đã mang lại nhiều thuận lợi đáng kể trong quá trình nghiên cứu phát hiện botnet ẩn trong lưu lượng mạng. Các thuật toán như Random Forest và Decision Tree thể hiện khả năng phân loại hiệu quả các mẫu lưu lượng bất thường trên bộ dữ liệu CTU-13. Chúng khai thác tốt các đặc trưng như tần suất gói tin, thời gian giữa các gói, và số lượng kết nối đến IP lạ, giúp nhận diện chính xác các botnet như Neris và Zeus.

Các phương pháp ensemble như Random Forest giúp giảm tỷ lệ dương tính giả đáng kể so với các phương pháp đơn lẻ. Điều này cải thiện độ tin cậy của hệ thống phát hiện botnet trong môi trường mạng phức tạp. Khả năng xử lý dữ liệu không cân bằng cũng là một lợi thế cho phép các thuật toán nhận diện botnet ẩn ngay cả khi dữ liệu chứa nhiều hoặc mẫu hành vi tinh vi.

Việc tích hợp các thư viện học máy hiện đại tạo điều kiện thuận lợi cho quá trình triển khai. Những công cụ này cung cấp các chức năng tối ưu hóa và đánh giá hiệu suất sẵn có, giúp rút ngắn thời gian phát triển mô hình và tăng hiệu quả nghiên cứu.

### 5.2.2 Hạn chế

Mặc dù đạt được nhiều kết quả tích cực, nghiên cứu vẫn đối mặt với một số hạn chế cần giải quyết. Các mô hình học máy phụ thuộc mạnh vào dữ liệu có nhãn. Điều này gây khó khăn khi phát hiện các botnet mới chưa được ghi nhận trong bộ dữ liệu CTU-13, đặc biệt khi botnet sử dụng kỹ thuật ngụy trang hoặc giao thức mã hóa để che giấu hành vi.

Một vấn đề khác là thời gian xử lý tăng đáng kể khi số lượng đặc trưng hoặc mẫu dữ liệu lớn. Hạn chế này làm giảm khả năng triển khai thời gian thực trong các hệ thống mạng có lưu lượng cao ảnh hưởng đến tính ứng dụng thực tế của mô hình trong các môi trường đòi hỏi phản hồi nhanh.

Ngoài ra, hiệu suất của các mô hình như Decision Tree và Random Forest giảm khi dữ liệu có tỷ lệ nhiễu cao hoặc khi botnet áp dụng các kỹ thuật ẩn mình tinh vi. Những yếu tố này làm giảm độ chính xác và độ tin cậy của hệ thống trong các kịch bản phức tạp.

Cuối cùng, yêu cầu tài nguyên tính toán cao với các phương pháp ensemble như Random Forest đòi hỏi phần cứng mạnh mẽ. Điều này gây khó khăn khi triển khai trên các hệ thống có nguồn lực hạn chế như thiết bị IoT hoặc mạng doanh nghiệp nhỏ, làm hạn chế phạm vi ứng dụng của nghiên cứu.

### 5.3 Hướng phát triển

Dựa trên các thuận lợi và hạn chế đã phân tích, dự án phát hiện botnet ẩn trong lưu lượng mạng bằng phương pháp học máy mở ra nhiều hướng phát triển. Một hướng quan trọng là tích hợp học không giám sát hoặc bán giám sát, chẳng hạn như sử dụng các thuật toán clustering hoặc autoencoder. Cách tiếp cận này có thể giúp phát hiện các botnet mới chưa được ghi nhận nhằm giảm sự phụ thuộc vào dữ liệu có nhãn và tăng khả năng thích nghi với các mối đe dọa mới.

Bên cạnh đó, tối ưu hóa hiệu suất thời gian thực là một hướng phát triển cần thiết. Các kỹ thuật như giảm chiều dữ liệu, tính toán song song (parallel computing) hoặc triển khai trên GPU có thể giúp mô hình đáp ứng yêu cầu xử lý nhanh trong các hệ thống mạng lớn, nâng cao khả năng ứng dụng thực tế.

Hơn nữa, phát triển các phương pháp giải mã giao thức mã hóa hoặc phân tích hành vi ẩn dựa trên các đặc trưng cấp cao như chuỗi thời gian hoặc đồ thị mạng sẽ giúp cải thiện khả năng phát hiện botnet sử dụng kỹ thuật ngụy trang. Điều này đặc biệt quan trọng trong bối cảnh các botnet ngày càng tinh vi.

Cuối cùng, kết hợp các thuật toán học máy với các phương pháp học sâu, như LSTM hoặc GNN, giúp tăng cường khả năng phân tích mẫu hành vi phức tạp của botnet. Sự kết hợp này có thể mở ra hướng tiếp cận mới để nâng cao độ chính xác và hiệu quả trong các ứng dụng an ninh mạng, đáp ứng các thách thức hiện tại và tương lai.

## TÀI LIỆU THAM KHẢO

1. Garcia, S., Grill, M., Stiborek, J., & Zunino, A. (2014). An empirical comparison of botnet detection methods. *Computers & Security*, 45, 100-123.
2. Strayer, W. T., Lapsley, D., Walsh, R., & Livadas, C. (2008). Botnet detection based on network behavior. In *Botnet Detection: Countering the Largest Security Threat* (pp. 1-24).
3. Beigi, E. B., Jazi, H. H., Stakhanova, N., & Ghorbani, A. A. (2014). Towards effective feature selection in machine learning-based botnet detection approaches. In *2014 IEEE Conference on Communications and Network Security* (pp. 247-255).
4. Velasco-Mata, J., & Others. (2021). Efficient detection of botnet traffic by features selection and decision trees. *arXiv preprint arXiv:2107.02896*
5. Hassan, W., Hosseini, S. E., & Pervez, S. (2024). Real-time anomaly detection in network traffic using graph neural networks and random forest. In Koucheryavy, Y., & Aziz, A. (Eds.), *Proceedings of the Conference* (pp. XXX-XXX).
6. Stratosphere Laboratory. (2011). The CTU-13 dataset: A labeled dataset with botnet, normal and background traffic. Stratosphere IPS.
7. Baruah, S., Borah, D. J., & Deka, V. (2023). Detection of peer-to-peer botnet using machine learning techniques and ensemble learning algorithm. *International Journal of Information Security and Privacy*, 17(1). <https://doi.org/10.4018/IJISP.319303>
8. Padhiar, S., & Patel, R. (2023). Performance evaluation of botnet detection using machine learning techniques. *International Journal of Electrical and Computer Engineering*, 13(6), 6827–6835. <https://doi.org/10.11591/ijece.v13i6.pp6827-6835>
9. Miller, S., & Busby-Earle, C. C. R. (2016, December). The role of machine learning in botnet detection. In *2016 International Conference for Internet Technology and Secured Transactions (ICITST)* (pp. 318–323). IEEE. <https://doi.org/10.1109/ICITST.2016.7856730>
10. Tikekar, P. C., Sherekar, S. S., & Kumar, J. (2024). An approach for detection of botnet based on machine learning classifier. *SN Computer Science*, 5(300). <https://doi.org/10.1007/s42979-024-02636-4>

11. Nguyen, H.-T., Ngo, Q.-D., Nguyen, D.-H., & Le, V.-H. (2020). PSI-rooted subgraph: A novel feature for IoT botnet detection using classifier algorithms. *ICT Express*, 6(2), 128–138. <https://doi.org/10.1016/j.ictex.2019.12.001>
12. Delplace, A., Hermoso, S., & Anandita, K. (2019, May 17). Cyber attack detection thanks to machine learning algorithms (arXiv:2001.06309v1). University of Queensland. <https://arxiv.org/abs/2001.06309v1>