

BỘ CÔNG THƯƠNG  
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM  
TP.HCM  
KHOA: CÔNG NGHỆ THÔNG TIN



## KHOÁ LUẬN TỐT NGHIỆP

TÊN ĐỀ TÀI: NGHIÊN CỨU CÁC THUẬT TOÁN  
MÁY HỌC ỨNG DỤNG TRONG PHÁT HIỆN TẤN  
CÔNG DDOS

Giảng viên hướng dẫn: ThS. Trần Đắc Tốt

Nhóm sinh viên thực hiện:  
Lê Thành Trung – 2033180129  
Thái Hoàng Cường – 2033181005  
Nguyễn Hoàng Duy – 2033180056

TP. Hồ Chí Minh, ngày 01/09/2021

## ***Lời cảm ơn***

Sau 4 tháng nỗ lực thực hiện, khoá luận “Nghiên cứu các thuật toán máy học ứng dụng trong phát hiện tấn công DDoS”. Ngoài sự nỗ lực của bản thân các thành viên trong nhóm, chúng em còn nhận được sự khích lệ rất nhiều từ phía nhà trường, thầy cô, gia đình, bạn bè trong khoa. Chính điều này đã mang lại cho em sự động viên rất lớn để chúng em có thể hoàn thành tốt khoá luận này.

Trước hết, chúng em xin cảm ơn bố mẹ và những người thân yêu đã luôn động viên, ủng hộ, chăm sóc và tạo mọi điều kiện tốt nhất để con hoàn thành nhiệm vụ của mình.

Chúng em cũng xin gửi lời cảm ơn thầy cô trường Đại Học Công Nghiệp Thực Phẩm TP.HCM nói chung và thầy cô Khoa Công Nghệ Thông Tin nói riêng đã đem lại cho chúng em nguồn kiến thức vô cùng quý giá để chúng em có đủ kiến thức hoàn thành khoá luận cũng như làm hành trang bước vào đời. Đặc biệt, chúng em xin bày tỏ lòng biết ơn sâu sắc tới giảng viên hướng dẫn là Thạc sĩ Trần Đắc Tốt, người đã tận tâm hướng dẫn và đưa ra những hướng đi hữu ích giúp chúng em trong suốt quá trình học tập và hoàn thiện khoá luận.

Cuối cùng chúng em kính chúc quý thầy, cô dồi dào sức khỏe và thành công trong sự nghiệp cao quý.

Tp. Hồ Chí Minh, ngày 1 tháng 9 năm 2021

Sinh viên thực hiện  
Lê Thành Trung

Sinh viên thực hiện  
Thái Hoàng Cường

Sinh viên thực hiện  
Nguyễn Hoàng Duy

## ***Lời cam đoan***

Chúng em xin cam đoan những kết quả đạt được trong khoá luận này là do chúng em nghiên cứu, tổng hợp và thực hiện. Toàn bộ những điều được trình bày trong khoá luận là của chúng em và được tham khảo cũng tổng hợp từ các nguồn tài liệu khác nhau. Tất cả các tài liệu tham khảo, tổng hợp đều được trích dẫn với nguồn gốc rõ ràng.

Tp. Hồ Chí Minh, ngày 1 tháng 9 năm 2021

Sinh viên thực hiện  
Lê Thành Trung

Sinh viên thực hiện  
Thái Hoàng Cường

Sinh viên thực hiện  
Nguyễn Hoàng Duy

# Mục lục

MỞ ĐẦU .....	1
CHƯƠNG 1: TỔNG QUAN.....	3
1.1. Giới thiệu.....	3
1.2. Hướng tiếp cận.....	3
1.3. Khó khăn và thách thức .....	5
1.4. Hướng giải quyết.....	5
CHƯƠNG 2: TẤN CÔNG DDOS VÀ MẠNG BOTNET.....	6
2.1. Tấn công từ chối dịch vụ phân tán (DDoS) .....	6
2.1.1. Khái niệm.....	6
2.1.2. Các loại tấn công DDoS.....	6
2.1.2.1. Tấn công làm cạn kiệt băng thông (Bandwidth Deletion) .....	7
A. Flood attack.....	7
B. Amplification attack.....	8
2.1.2.2. Tấn công làm cạn kiệt tài nguyên (Resource Deletion) .....	9
2.2. Mạng Botnet .....	11
2.2.1. Khái niệm.....	11
2.2.2. Topology giao tiếp của mạng Botnet .....	12
2.2.2.1. Star.....	12
2.2.2.2. Multi – Server.....	13
2.2.2.3. Hierarchical .....	14
2.2.2.4. Random .....	15
CHƯƠNG 3: CƠ SỞ LÝ THUYẾT .....	17
3.1. Học có giám sát (Supervised Learning).....	17
3.2. Các thuật toán phân loại trong học có giám sát (Supervised Learning).....	17
3.2.1. Decision Tree .....	17
3.2.2. Random Forest .....	18
3.2.3. Naive Bayes .....	19
3.2.4. K-Nearest Neighbors (KNN) .....	20
3.2.5. Logistic Regression .....	22
3.2.6. XGBoost .....	22
3.3. Các kỹ thuật dùng trong giai đoạn tiền xử lý dữ liệu (preprocessing).....	23
3.3.1. Xử lý các giá trị rỗng trong tập dữ liệu .....	23
3.3.2. Xử lý các biến/ giá trị phân loại .....	25
3.3.3. Kỹ thuật Feature Selection.....	26
3.3.3.1. Phương pháp lọc (Filter Methods) .....	26
3.3.4. Xử lý tập dữ liệu mất cân bằng.....	28
3.3.4.1. Kỹ thuật Controlled Under-Sampling .....	28
3.3.5. Kỹ thuật Feature Scaling.....	30
3.3.5.1. Normalization .....	30

3.3.5.2. Standardization .....	31
3.3.5.3. Khi nào nên sử dụng kỹ thuật Normalization hay Standardization .....	31
3.4. Công cụ, thư viện, ngôn ngữ, môi trường cài đặt hỗ trợ trong quá trình xây dựng ứng dụng .....	32
3.4.1. Công cụ Tcpdump .....	32
3.4.2. Công cụ Argus.....	32
3.4.3. Python sys module.....	32
3.4.4. Thư viện subprocess .....	33
3.4.5. Ngôn ngữ lập trình Python.....	33
3.4.6. Môi trường cài đặt.....	33
CHƯƠNG 4: XÂY DỰNG ỨNG DỤNG VÀ MÔ HÌNH MÁY HỌC .....	34
4.1. Xây dựng mô hình máy học.....	34
4.2. Hiện thực cài đặt các công cụ, thư viện hỗ trợ.....	35
4.3. Xây dựng ứng dụng.....	36
4.3.1. Sơ đồ ứng dụng .....	36
4.3.2. Mã Python3.....	38
CHƯƠNG 5: THỰC NGHIỆM VÀ ỨNG DỤNG .....	41
5.1. Môi trường cài đặt.....	41
5.2. Bộ dữ liệu .....	41
5.2.1. Giới thiệu bộ dữ liệu CTU-13 .....	41
5.2.2. Quy trình tạo ra bộ dữ liệu CTU-13.....	43
5.2.3. Các cột/ tính năng của bộ dữ liệu .....	44
5.2.4. Tổng quát hoá bộ dữ liệu .....	45
5.3. Quá trình xử lý bộ dữ liệu.....	47
5.3.1. Quá trình tiền xử lý.....	47
5.3.1.1. Loại bỏ các cột/ tính năng không cần thiết .....	47
5.3.1.2. Xử lý các giá trị rỗng .....	47
5.3.1.3. Xử lý dữ liệu nhãn thô (raw label) .....	47
5.3.1.4. Xử lý các biến/ giá trị phân loại.....	48
5.3.1.5. Sử dụng kỹ thuật Feature Selection.....	51
5.3.1.6. Xử lý sự mất cân bằng trên bộ dữ liệu .....	52
5.3.1.7. Sử dụng kỹ thuật Feature Scaling .....	52
5.3.2. Tạo các kịch bản đào tạo mô hình máy học .....	53
5.4. Phương thức đánh giá.....	54
5.5. Kết quả thực nghiệm .....	56
5.5.1. Kết quả đánh giá trường hợp kịch bản sử dụng mã hoá với giá trị số.....	56
5.5.2. Kết quả đánh giá trường hợp kịch bản sử dụng mã hoá One-hot Encoding .....	57
5.6. Kết quả ứng dụng.....	61
5.7. Tổng kết và hướng phát triển .....	63
Tài liệu tham khảo .....	65

## DANH MỤC CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT

Viết tắt	Tiếng Anh	Tiếng Việt
DDoS	Distributed Denial Of Service	Distributed Denial Of Service
CTU-13	CTU-13	Bộ dữ liệu CTU-13
BI-LSTM	Bi-Directional Long Short-Term Memory	Bộ nhớ ngắn hạn dài hạn hai hướng
GMM	Gaussian Mixture Model	Mô hình Gaussian hỗn hợp
CNN	Convolutional Neural Network	Mạng thần kinh tích chập
LOIC	Low Orbit Ion Cannon	Ứng dụng kiểm tra mạng và tấn công từ chối dịch vụ mã nguồn mở
UDP	User datagram protocol	Giao thức datagram người dùng
ICMP	Internet Control Message Protocol	Giao thức thông báo kiểm soát Internet
IP	Internet Protocol Address	Địa chỉ giao thức internet
TCP	Transmission Control Protocol	Giao thức điều khiển truyền vận
SYN	Synchronize	Đồng bộ
ACK	Acknowledgment	Xác nhận
PSH	Push	Đẩy đi
IRC	Internet Relay Chat	Trò chuyện chuyển tiếp Internet
ADSL	Asymmetric Digital Subscriber Line	Đường dây thuê bao kỹ thuật số không đối xứng
C&C	Command-and-Control	Ra lệnh và kiểm soát
ANOVA	Analysis of Variance	Phân tích phương sai
LAN	Local area network	Mạng cục bộ
CTU	Chienkuo Technology University	Đại học Công nghệ Chienkuo

PCAP	Packet Capture	Bắt gói tin
TP	True Positive	Đúng tích cực
TN	True Negative	Đúng tiêu cực
FP	False Positive	Sai tích cực
FN	False Negative	Sai tiêu cực
TPR	True Positive Rate	Tỷ lệ đúng tích cực
FPR	False Positive Rate	Tỷ lệ sai tích cực
AUC	Area Under Curve	Khu vực dưới đường cong
ROC	Receiver Operating Characteristic	Những đặc thù có tác dụng nhận
DT	Decision Tree	Cây quyết định
RF	Random Forest	Rừng ngẫu nhiên
NB	Naive Bayes	Naive Bayes
XGB	XGBoost	XGBoost
KNN	K-Nearest Neighbors	K hàng xóm gần nhất
LG	Logistic Regression	Hồi quy logistic

## DANH MỤC CÁC BẢNG BIỂU

Bảng 5. 1: 13 kịch bản của bộ dữ liệu CTU-13 .....	41
Bảng 5. 2: Lượng dữ liệu trên mỗi kịch bản mạng botnet.....	42
Bảng 5. 3: Phân phối nhãn trong NetFlows cho mỗi kịch bản của bộ dữ liệu.....	43
Bảng 5. 4: Mô tả 15 cột/ tính năng của bộ dữ liệu .....	44
Bảng 5. 5: Trường hợp kịch bản sử dụng mã hoá với giá trị số .....	56
Bảng 5. 6: Trường hợp sử dụng kỹ thuật lựa chọn tính năng với Chi-Squared.....	57
Bảng 5. 7: Trường hợp sử dụng kỹ thuật lựa chọn tính năng với ANOVA .....	59

## DANH MỤC CÁC HÌNH ẢNH

Hình 2. 1: Các kỹ thuật tấn công DDoS.....	6
Hình 2. 2: Mô hình UDP Flood attack.....	7
Hình 2. 3: Mô hình ICMP Flood attack .....	8
Hình 2. 4: Mô hình Smurf attack.....	9
Hình 2. 5: Quá trình Three-way HandShake.....	10
Hình 2. 6: Quá trình tấn công SYN Flood.....	10
Hình 2. 7: Star C&C topology.....	13
Hình 2. 8: Multi – Server C&C topology.....	14
Hình 2. 9: Hierarchical C&C topology .....	15
Hình 2. 10: Random C&C topology .....	16
Hình 3. 1: Ví dụ về cấu trúc cây quyết định (Decision Tree) [37].....	18
Hình 3. 2: Ví dụ về mô hình rừng ngẫu nhiên (Random Forest) [38] .....	19

Hình 3. 3: Ví dụ về phân loại với Naive Bayes [39].....	20
Hình 3. 4: Điểm dữ liệu mới cần phân loại [40].....	21
Hình 3. 5: Điểm dữ liệu mới được phân loại là A [40] .....	21
Hình 3. 6: Ví dụ về Logistic Regression [41].....	22
Hình 3. 7: Ví dụ loại bỏ cột/ tính năng chứa giá trị rỗng.....	23
Hình 3. 8: Ví dụ loại bỏ hàng/ mẫu chứa giá trị rỗng.....	24
Hình 3. 9: Ví dụ thay thế giá trị rỗng bằng giá trị trung bình cộng.....	24
Hình 3. 10: Ví dụ thay giá trị rỗng bằng giá trị khác có thêm cột/ tính năng .....	24
Hình 3. 11: Ví dụ loại bỏ các cột/ tính năng chứa biến phân loại .....	25
Hình 3. 12: Ví dụ sử dụng kỹ thuật mã hoá thứ tự (Ordinal Encoding) .....	25
Hình 3. 13: Ví dụ sử dụng kỹ thuật mã hoá One-hot Encoding.....	25
Hình 3. 14: Ví dụ sử dụng mã hoá với giá trị số.....	26
Hình 3. 15: Ví dụ khi sử dụng RandomUnderSampler [42].....	29
Hình 3. 16: Ví dụ về NearMiss-1 lựa chọn mẫu dựa trên khoảng cách [42] .....	29
Hình 3. 17: Ví dụ khi scale dữ liệu về khoảng (0, 1) [43].....	31
Hình 3. 18: Ví dụ khi thay đổi hình dạng dữ liệu với Standardization [43] .....	31
Hình 4. 1: Sơ đồ chung của quá trình xây dựng các mô hình máy học.....	34
Hình 4. 2: Cài đặt hoàn tất máy ảo Ubuntu .....	36
Hình 4. 3: Sơ đồ ứng dụng .....	37
Hình 4. 4: Các tập tin, thư mục sử dụng trong quá trình xây dựng ứng dụng.....	38
Hình 5. 1: Mô hình thu thập dữ liệu cho bộ dữ liệu CTU-13 .....	43
Hình 5. 2: Ví dụ tệp NetFlow .....	44
Hình 5. 3: Bộ dữ liệu CTU-13 trong khi đọc trong DataFrame.....	45
Hình 5. 4: Số lượng mẫu trong mỗi 3 lớp .....	46
Hình 5. 5: Thống kê cho các cột/ tính năng.....	46
Hình 5. 6: Thống kê cho từng cột/ tính năng.....	46
Hình 5. 7: Cột/ tính năng ‘Label’ với dữ liệu thô .....	47
Hình 5. 8: Cột/ tính năng ‘Label’ sau khi gán lại nhãn .....	48
Hình 5. 9: Số lượng giá trị mỗi loại trong cột/ tính năng ‘Dir’ .....	49
Hình 5. 10: Số lượng giá trị mỗi loại trong cột/ tính năng ‘Proto’ .....	49
Hình 5. 11: Số lượng giá trị mỗi loại trong cột/ tính năng ‘sTos’ .....	50
Hình 5. 12: Số lượng giá trị mỗi loại trong cột/ tính năng ‘dTos’ .....	50
Hình 5. 13: Bộ dữ liệu sau khi mã hoá One-hot Encoding.....	51
Hình 5. 14: Sơ đồ kịch bản.....	53
Hình 5. 15: Ví dụ về Confusion Matrix .....	55
Hình 5. 16: Ví dụ về ROC và AUC .....	55
Hình 5. 17: So sánh 2 confusion matrix của 2 mô hình .....	57
Hình 5. 18: Confusion matrix của Random Forest trong kịch bản (4.2) .....	58
Hình 5. 19: ROC-AUC của Random Forest trong kịch bản (4.2).....	59
Hình 5. 20: Confusion matrix của XGBoost trong kịch bản (4.2) .....	60
Hình 5. 21: ROC-AUC của XGBoost trong kịch bản (4.2).....	60
Hình 5. 22: Ví dụ chạy ứng dụng .....	61
Hình 5. 23: 4 thành phần chính trong kết quả hiển thị .....	61
Hình 5. 24: Kết quả cho trường hợp phát hiện lưu lượng Background-Normal .....	61
Hình 5. 25: Máy chủ gửi lệnh hướng dẫn tấn công cho botnet client .....	62
Hình 5. 26: Kết quả cho trường hợp phát hiện lưu lượng Botnet-DDoS.....	62
Hình 5. 27: Kết quả dự đoán bị sai cho lưu lượng bình thường.....	62
Hình 5. 28: Kết quả trong tệp log.txt .....	63
Hình 5. 29: Kết quả trong thư mục PCAP.....	63



## MỞ ĐẦU

Hiện nay, hệ thống mạng lưới Internet đã và đang phát triển hết sức kinh ngạc. Bằng chứng là nhiều quốc gia từ nhỏ đến lớn đều đang sử dụng mạng Internet để phục vụ cho lợi ích của riêng mình. Điều này góp một phần không nhỏ cho lợi ích kinh tế, chính trị của mỗi quốc gia. Không chỉ có vậy, mạng lưới Internet còn giúp cho cuộc sống của mỗi người trên thế giới được kết nối với nhau, giúp cho các công việc hằng ngày được tối ưu hoá hơn, được nhanh chóng hơn.

Song song với sự phát triển và những lợi ích ấy, thì vẫn tồn tại những mối nguy hại cho mạng lưới Internet. Đó là những mặt tối của mạng Internet, ví dụ như những cuộc tấn công vào hệ thống ngân hàng, trường học, tổ chức, cơ quan truyền thông nhằm đánh cắp những thông tin riêng tư, nhạy cảm nhằm trục lợi cho bản thân của kẻ xấu.

Hơn thế nữa là những cuộc tấn công từ chối dịch vụ phân tán (tên tiếng Anh là Distributed Denial Of Service – DDoS). Đây là những cuộc tấn công được biết đến từ những năm 1998. Và hiện nay, nó đang được phát triển với nhiều loại, hình thức tấn công mới, biến thể mới, gây ra thách thức lớn cho các nhà quản trị an ninh, bảo mật mạng.

Mô hình chung của các cuộc tấn công từ chối dịch vụ phân tán (DDoS) là dựa trên các hệ thống mạng Botnet. Botnet là mạng bao gồm các máy tính đã nhiễm mã độc và được sử dụng để phục vụ mục đích riêng của kẻ tấn công. Hậu quả của các cuộc tấn công DDoS có thể gây ra tổn thất lớn về kinh tế, về tính tiện lợi của mạng Internet. Ngoài ra, nó còn ảnh hưởng không chỉ cho các doanh nghiệp mà còn ảnh hưởng tới người dùng Internet. Nó gây ra sự mất kết nối giữa mọi người từ việc gây ra các cuộc tấn công từ chối dịch vụ.

Vì vậy việc phát hiện để ngăn chặn hay giảm thiểu các cuộc tấn công DDoS là một điều hết sức cần thiết trong giai đoạn Internet phát triển như hiện nay. Do lưu lượng của các cuộc tấn công từ chối dịch vụ chủ yếu tập trung từ các mạng Botnet nên nhóm đề xuất hướng nghiên cứu và ứng dụng các thuật toán máy học được học tập thông qua bộ dữ liệu lưu lượng mạng trong thực tế (bộ dữ liệu CTU-13 [2]) để phát hiện lưu lượng Botnet nói chung cũng như lưu lượng độc hại từ các cuộc tấn công từ chối dịch vụ phân tán (DDoS) hay tấn công từ chối dịch vụ (DoS) nói riêng. Từ đó, ứng dụng vào giải quyết bài toán phát hiện lưu lượng tấn công DDoS để giúp ích cho việc quản trị cũng như bảo mật hệ thống mạng của các cá nhân hay công ty, doanh nghiệp nhỏ, lẻ khác.

Nội dung khoá luận sẽ bao gồm những chương sau:

Chương 1: Tổng quan, giới thiệu về các phương pháp và hướng tiếp cận phát hiện tấn công từ chối dịch vụ phân tán (DDoS), ưu nhược điểm của các phương pháp và hướng tiếp cận đó, khó khăn và thách thức, đưa ra hướng tiếp cận của nhóm.

Chương 2: Tấn công DDoS và mạng Botnet, trình bày các lý thuyết liên quan đến tấn công từ chối dịch vụ phân tán (DDoS) và mạng Botnet.

Chương 3: Cơ sở lý thuyết, trình bày các cơ sở lý thuyết từ đề tài bao gồm các thuật toán máy học như Decision Tree, Random Forest, Naive Bayes, Logistic Regression, K-Nearest Neighbors, XGBoost, các phương pháp xử lý trên tập dữ liệu mất cân bằng, xử lý dữ liệu rỗng, xử lý biến/ giá trị phân loại, feature selection, feature scaling,...

Chương 4: Xây dựng ứng dụng và mô hình máy học, trình bày cách xây dựng ứng dụng cho bài toán phát hiện tấn công từ chối dịch vụ (DDoS), sơ đồ xây dựng mô hình máy học.

Chương 5: Kết quả thực nghiệm và ứng dụng, trình bày kết quả thực nghiệm, kết quả ứng dụng và nêu hướng phát triển của đề tài.

# CHƯƠNG 1

## TỔNG QUAN

### 1.1. Giới thiệu

Tấn công DDoS là một loại hình tấn công đã được thực hiện rất nhiều lần và tại nhiều nơi trên thế giới. Nó mang lại nhiều hậu quả cho người dùng Internet. Tuy nhiên, các nhà nghiên cứu đã tìm ra nhiều giải pháp giúp phát hiện và giảm thiểu tấn công DDoS. Họ đã giúp tạo ra một sự cân bằng nhất định trên Internet. Vì vậy, nhóm chúng tôi cũng muốn chia sẻ/ đóng góp một phần công sức của mình vào việc nghiên cứu này. Bài nghiên cứu của nhóm có thể sẽ không đạt được các kết quả như mong đợi nhưng hy vọng nó sẽ đóng góp một phần không nhỏ vào quá trình phát triển chung của việc nghiên cứu phòng chống tấn công DDoS.

Mục tiêu nghiên cứu của nhóm chúng tôi là nghiên cứu về các thuật toán máy học có giám sát và các kỹ thuật xử lý đi kèm, đồng thời tìm ra cách áp dụng mô hình máy học đã được đào tạo để từ đó có được kết quả là một ứng dụng/ công cụ có thể phát hiện được các cuộc tấn công DDoS.

### 1.2. Hướng tiếp cận

Một trong những hướng tiếp cận thông dụng của việc phát hiện tấn công DDoS là sử dụng 2 kỹ thuật dựa trên chữ ký (signature-based) và dựa trên bất thường (anomaly-based) được triển khai trên hệ thống phát hiện xâm nhập (IDS - Intrusion Detection System) như giới thiệu trong bài báo [3]. Trong đó các tác giả đã đề xuất một hệ thống phát hiện xâm nhập kết hợp cả 2 kỹ thuật Signature-based và Anomaly-based, một giải pháp có thể mở rộng để phát hiện sớm các cuộc tấn công flooding DDoS. Họ đã kết luận rằng IDS kết hợp được cho là mạnh mẽ hơn Anomaly-based IDS vì nó sử dụng các lợi thế của kỹ thuật Signature-based và việc tạo ra nhiều chữ ký hơn sẽ nâng cao hơn nữa hiệu suất tổng thể của IDS kết hợp.

Phương pháp phát hiện entropy là một phương pháp hiệu quả để phát hiện cuộc tấn công DDoS. Trong bài báo [4], các tác giả cải tiến thuật toán phát hiện entropy và đề xuất hai phương pháp phát hiện nâng cao dựa trên entropy tích lũy và thời gian tương ứng. Kết quả thử nghiệm cho thấy rằng những phương pháp này có thể phát hiện DDoS chính xác và hiệu quả hơn.

Một hướng tiếp cận khác được các tác giả trong bài báo [5] sử dụng để phát hiện và phân tích tấn công DDoS là thông qua Map Reduce trong Hadoop. Cụ thể, [5] sử dụng Hadoop framework để giải quyết vấn đề dữ liệu lớn do các cuộc tấn công DDoS gây ra. Đồng thời đề xuất thuật toán Counter-based để

phát hiện các cuộc tấn công DDoS để giải quyết vấn đề tỷ lệ dương tính giả cao và thuật toán Access pattern-based để phát hiện các cuộc tấn công DDoS có sử dụng hành vi của kẻ tấn công. Thuật toán Map reduce trong phần thuật toán Counter-based sẽ tạo ra các bản ghi có tỷ lệ phản hồi so với các yêu cầu lớn hơn unbalance ratio (một tham số của thuật toán Map reduce) và đánh dấu chúng là kẻ tấn công. Để tăng thêm hiệu suất của hệ thống, các tác giả [5] đã sử dụng đầu ra của thuật toán Counter-based làm đầu vào cho thuật toán Access pattern-based.

Tiếp theo là hướng tiếp cận liên quan đến thuật ngữ Machine Learning (học máy). Machine Learning đang là từ khoá rất nổi bật trong những năm gần đây vì tính ứng dụng cao của nó vào việc giải quyết các bài toán thực tế. Do đó, sẽ không có gì quá bất ngờ khi chúng ta thấy nó được ứng dụng vào bài toán phát hiện tấn công DDoS. Như trong bài báo [6], các tác giả đã đào tạo 7 mô hình Machine Learning bao gồm: Random Forest, Boosted Trees, Decision Tree, Naive Bayes, K-Nearest Neighbors, Logistic Regression, SVM Linear và 1 mô hình mạng thần kinh (Neural Network) trên tập dữ liệu CIC-IDS2017 [7]. Sau quá trình xử lý dữ liệu rộng và lựa chọn tính năng bài báo đánh giá mô hình Random Forest cho kết quả tốt nhất với độ chính xác cao nhất là 99,97% theo sau là 2 mô hình Boosted Trees và Decision Tree.

Bài báo [8] cũng đề cập đến hướng tiếp cận Machine Learning để phát hiện DDoS và thông qua thuật toán Multiple Linear Regression. Bộ dữ liệu CIC-IDS2017 [7] (dùng kịch bản Friday afternoon log file) được sử dụng và mô hình Multiple Linear Regression sau khi lựa chọn tính năng dựa trên Information Gain và Regression Analysis cho kết quả chính xác là 73,79%.

Bên cạnh hướng tiếp cận Machine Learning thì ta không thể không nhắc đến hướng tiếp cận với Deep Learning. Trong bài báo [9] có đề xuất một DDoS detection framework mới có tính năng Bi-Directional Long Short-Term Memory (tạm dịch là bộ nhớ ngắn hạn dài hạn hai hướng), một mô hình Gaussian Mixture Model (GMM) và học tập gia tăng (incremental learning). Hai bộ dữ liệu CIC-IDS2017 và CIC-DDoS2019 được sử dụng để đào tạo, thử nghiệm và đánh giá, kết quả thử nghiệm cho thấy BI-LSTM-GMM được đề xuất có thể đạt được recall, precision và độ chính xác (accuracy) lên đến 94%.

Khác với cách sử dụng mô hình kết hợp BI-LSTM-GMM như trong bài báo [9], các tác giả trong bài báo [10] sử dụng mô hình kết hợp CNN-BI-LSTM. Sau quá trình xếp hạng và chọn các tính năng đạt điểm cao nhất trong

tập dữ liệu CIC-DDoS2019, mô hình kết hợp CNN-BI-LSTM được đề xuất đã đạt được độ chính xác lên đến 94,52%.

### **1.3. Khó khăn và thách thức**

Bài toán về việc phát hiện tấn công từ chối dịch vụ phân tán (DDoS) bằng cách sử dụng các thuật toán máy học có những khó khăn và thách thức sau. Đầu tiên là vấn đề về tập dữ liệu đầu vào cho các thuật toán máy học, các tập dữ liệu này thường là các tập dữ liệu mô phỏng không phải trong thực tế, không được thiết kế theo một chuẩn nhất định và có thể có nhiều. Thứ hai, vấn đề mất cân bằng trong các tập dữ liệu khiến các thuật toán máy học thường học theo các lớp đa số, điều này gây ra dự đoán sai khi đưa vào dữ liệu đầu vào trong thực tế. Cuối cùng là việc tối ưu các thuật toán máy học mất rất nhiều thời gian và có thể không dự đoán chính xác một cách tuyệt đối với dữ liệu mới.

### **1.4. Hướng giải quyết**

Trong đề tài khoá luận này, ban đầu, nhóm chúng tôi dự định sẽ thực hiện hướng tiếp cận phát hiện tấn công từ chối dịch vụ phân tán (DDoS) dựa trên các thuật toán máy học và sử dụng các tập tin lưu lượng mạng pcap có chứa lưu lượng tấn công DDoS xen lẫn lưu lượng mạng lành tính làm dữ liệu đầu vào. Các tập tin pcap này sẽ bao gồm lưu lượng tấn công DDoS được trích xuất từ các công cụ như LOIC, Hping3,... Bên cạnh đó thì lưu lượng lành tính được tạo ra từ các giao tiếp thông thường trên mạng như lệnh ping kiểm tra kết nối giữa hai máy tính, hay lưu lượng của các gói tin HTTP khi truy cập web. Tuy nhiên, trong quá trình thực nghiệm, chúng tôi nhận ra vấn đề chênh lệch trong các công cụ DDoS là rất khác nhau làm cho các mô hình máy học không hiệu quả trên dữ liệu tấn công DDoS thực tế. Do đó, để giải quyết vấn đề này, nhóm chúng tôi đề xuất hướng tiếp cận phát hiện tấn công từ chối dịch vụ phân tán (DDoS) thông qua việc phát hiện lưu lượng mạng Botnet trên mô hình máy học. Ngoài ra, chúng tôi có áp dụng thêm các kỹ thuật xử lý dữ liệu cho bộ dữ liệu CTU-13 [2] nhằm tăng độ chính xác cho các mô hình học máy. Điều này mở rộng quy mô từ việc phát hiện tấn công DDoS thành phát hiện lưu lượng mạng Botnet bởi vì tấn công DDoS được thực hiện chủ yếu thông qua mạng Botnet. Giới hạn của đề tài là ưu tiên phát hiện lưu lượng tấn công DDoS từ mạng Botnet thực tế chứ không tập trung vào lưu lượng tấn công DDoS từ các công cụ LOIC hay Hping3,...

## CHƯƠNG 2

# TẤN CÔNG DDOS VÀ MẠNG BOTNET

### 2.1. Tấn công từ chối dịch vụ phân tán (DDoS)

#### 2.1.1. Khái niệm

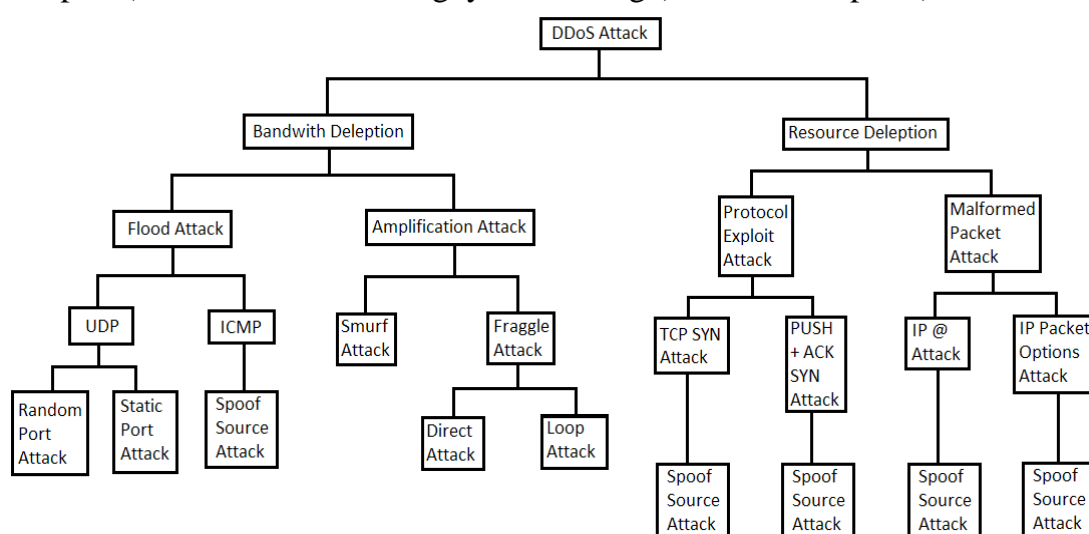
Tấn công từ chối dịch vụ phân tán (hay gọi tắt là DDoS) là cuộc tấn công trong đó nhiều hệ thống máy tính bị xâm nhập và đồng thời cùng tấn công vào một mục tiêu/ nạn nhân như một server hay một website... với mục đích xấu là làm tràn ngập các thông báo, các yêu cầu kết nối hoặc các gói tin không đúng định dạng được gửi đến hệ thống khiến nó chạy chậm lại hay gặp sự cố dẫn đến mất kết nối hoàn toàn. Điều này sẽ từ chối dịch vụ từ các người dùng hợp pháp trên mạng.

Tấn công DDoS trở nên phổ biến vì dễ dàng khai thác quyền truy cập và không cần phải tốn quá nhiều thời gian để thực thi. Những cuộc tấn công DDoS rất nguy hiểm vì chúng có thể nhanh chóng tiêu thụ một số lượng lớn các host trên Internet và khiến chúng trở nên vô dụng.

Một số tác hại của DDoS có thể kể đến bao gồm: mất đi lòng tin của khách hàng đối với các doanh nghiệp bị tấn công, gây ảnh hưởng đến sự an toàn mạng, tổn thất tài chính, từ chối dịch vụ người dùng hợp pháp, ...v...v.

#### 2.1.2. Các loại tấn công DDoS

Kỹ thuật tấn công DDoS có rất nhiều hình thức khác nhau từ đơn giản đến phức tạp, nhưng nếu nhìn dưới góc độ chuyên môn thì có thể chia thành hai loại dựa trên mục đích tấn công: Làm cạn kiệt băng thông (bandwidth depletion) và làm cạn kiệt tài nguyên hệ thống (resource depletion).



Hình 2. 1: Các kỹ thuật tấn công DDoS

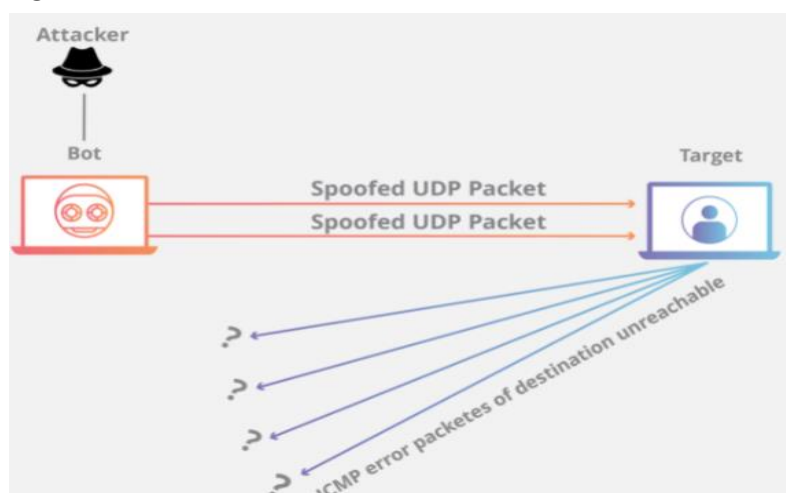
### 2.1.2.1. Tấn công làm cạn kiệt băng thông (Bandwidth Deletion)

#### A. Flood attack

Trong dạng tấn công Flood này, một lượng lớn các gói tin được gửi từ các botclient làm hệ thống mục tiêu bị treo do nhận quá nhiều yêu cầu và không thể đáp ứng các yêu cầu hợp lệ khác.

#### UDP Flood attack

Trong một cuộc tấn công UDP Flood, kẻ tấn công sẽ gửi các gói UDP giả mạo với số lượng lớn đến một máy chủ từ xa trên các cổng ngẫu nhiên của máy chủ mục tiêu bằng cách sử dụng dải IP nguồn lớn. Việc tràn ngập các gói UDP này khiến máy chủ phải kiểm tra nhiều lần để tìm các ứng dụng không tồn tại ở các cổng. Do đó, hệ thống không thể truy cập các ứng dụng hợp pháp và bất kỳ nỗ lực nào để truy cập vào chúng đều trả về phản hồi lỗi với gói ICMP "Destination Unreachable". Cuộc tấn công này tiêu tốn tài nguyên mạng và băng thông có sẵn.

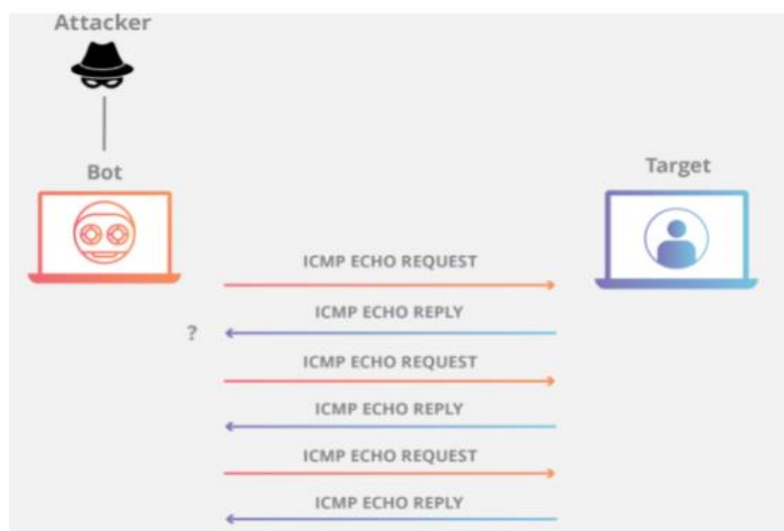


Hình 2. 2: Mô hình UDP Flood attack

Trong kiểu tấn công DDoS này, để tránh bị lộ vị trí của kẻ tấn công, chúng thường giả mạo địa chỉ IP nguồn trong các gói tin UDP.

#### ICMP Flood attack

ICMP là một giao thức hỗ trợ được sử dụng bởi các thiết bị mạng để kiểm tra thông tin hoạt động, lỗi và thông báo. Các yêu cầu và phản hồi ICMP tiêu tốn rất nhiều tài nguyên của thiết bị mạng. Lợi dụng điều này, kẻ tấn công sử dụng các yêu cầu ICMP để tấn công tràn ngập ICMP và không cần chờ phản hồi, do đó làm ngập tài nguyên của thiết bị.



Hình 2. 3: Mô hình ICMP Flood attack

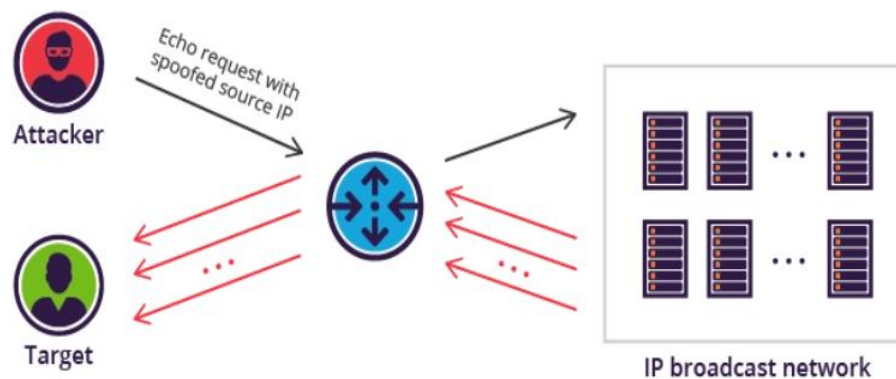
### **B. Amplification attack**

Một cuộc tấn công khuếch đại là cuộc tấn công nào mà kẻ tấn công có thể sử dụng một hệ số khuếch đại để nhân lên sức mạnh của nó. Kẻ tấn công sẽ sử dụng lệnh ping đến địa chỉ của một mạng khác nhưng với địa chỉ nguồn là địa chỉ của mục tiêu. Sau đó, các gói tin reply từ các máy trong mạng sẽ được phản hồi tới máy mục tiêu. Kẻ tấn công có thể tránh lộ vị trí bằng cách sử dụng mạng khuếch đại làm công cụ ẩn danh trên mạng.

### **Smurf attack**

Trong cuộc tấn công này một số lượng lớn các gói tin ICMP với IP nguồn giả mạo của nạn nhân được phát tới một mạng máy tính bằng địa chỉ IP broadcast. Theo mặc định, khi các máy tính trong mạng đó nhận được các gói tin ICMP request với IP nguồn là IP của nạn nhân thì chúng sẽ phản hồi điều này bằng cách gửi trả lời đến địa chỉ IP nguồn. Với số lượng lớn các máy trong mạng nhận và phản hồi các gói tin này, thì máy tính của nạn nhân sẽ bị ngập trong lưu lượng truy cập do nhận quá nhiều gói reply mà bản thân không yêu cầu từ mạng các máy tính kia. Điều này có thể làm chậm máy tính của nạn nhân đến mức không thể hoạt động được.





Hình 2. 4: Mô hình Smurf attack

Sức mạnh của dạng tấn công này phụ thuộc vào tỷ lệ khuếch đại, cụ thể là số lượng máy tính có trong mạng khuếch đại. Vì thế kẻ tấn công càng chiếm được càng nhiều hệ thống mạng khuếch đại thì cuộc tấn công sẽ càng gây ra hậu quả kinh khủng hơn so với việc chỉ có một mạng khuếch đại.

#### **Fraggle attack**

Fraggle attack chỉ khác với Smurf attack ở chỗ là nó sử dụng các gói tin UDP thay cho các gói tin ICMP.

#### **2.1.2.2. Tấn công làm cạn kiệt tài nguyên (Resource Depletion)**

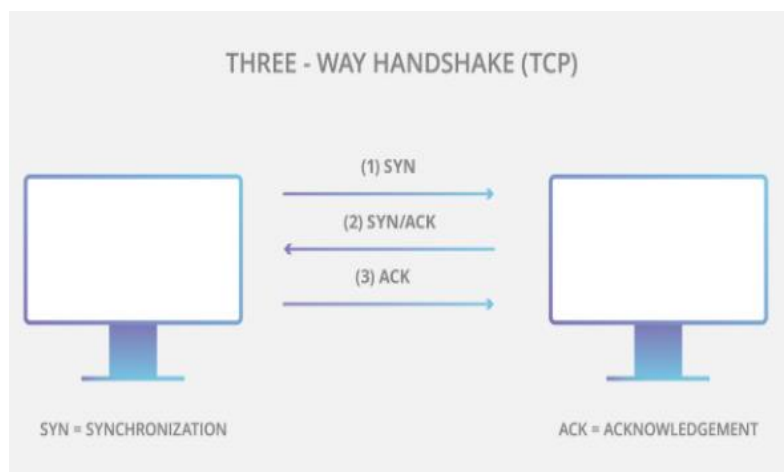
##### **TCP SYN Flood attack**

Tấn công TCP SYN Flood (còn gọi là SYN Flood) là một loại tấn công DDoS khai thác một phần của quá trình Three-way HandShake (bắt tay ba chiều TCP).

Về cơ bản, với SYN Flood, kẻ tấn công sẽ gửi các yêu cầu kết nối TCP nhanh hơn mức mà máy chủ mục tiêu có thể xử lý chúng.

Quá trình Three-way HandShake bao gồm 3 bước:

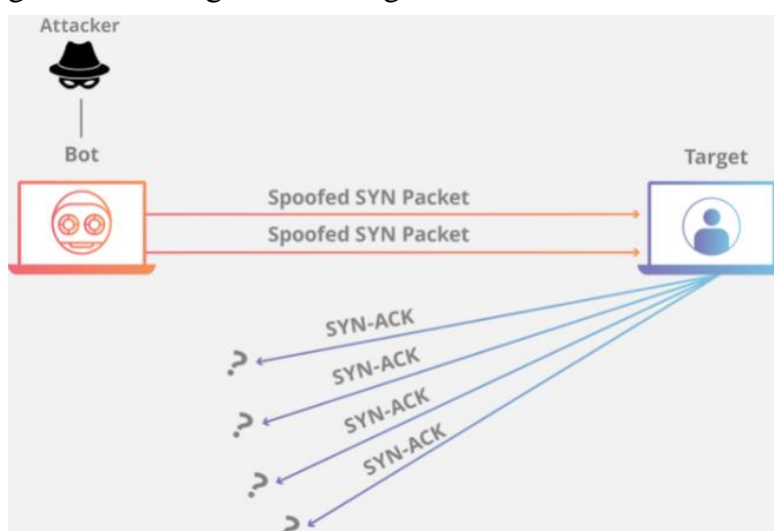
- Đầu tiên, client sẽ gửi một gói tin SYN đến server để bắt đầu kết nối.
- Sau đó, server phản hồi bằng gói SYN/ACK để thông báo cho client biết là server đã nhận được tín hiệu và chấp nhận kết nối từ client.
- Cuối cùng, client trả về một gói ACK để xác nhận việc nhận gói SYN/ACK từ server. Sau khi hoàn thành 3 bước gửi và nhận gói tin này, kết nối TCP sẽ mở và có thể gửi và nhận dữ liệu.



Hình 2. 5: Quá trình Three-way HandShake

Để tạo nên một cuộc tấn công SYN Flood, kẻ tấn công thường làm theo cách như sau:

- Đầu tiên, kẻ tấn công gửi một lượng lớn gói SYN đến máy chủ mục tiêu, thường là các địa chỉ IP giả mạo.
- Sau đó, máy chủ sẽ trả lời từng yêu cầu kết nối và để lại một cổng mở sẵn sàng nhận phản hồi.
- Trong khi máy chủ đợi gói ACK cuối cùng và gói này sẽ không bao giờ đến, kẻ tấn công tiếp tục gửi thêm gói SYN. Sự xuất hiện của mỗi gói SYN mới khiến máy chủ tạm thời duy trì kết nối cổng mở mới trong một khoảng thời gian nhất định và khi tất cả các cổng có sẵn đã được sử dụng, máy chủ sẽ không thể hoạt động bình thường.



Hình 2. 6: Quá trình tấn công SYN Flood

### **SYN – ACK Flood attack**

Kiểu tấn công này tương tự như kiểu tấn công SYN Flood, khác biệt duy nhất là kẻ tấn công khai thác giai đoạn thứ hai của quá trình Three-way

HandShake bằng cách gửi một số lượng lớn gói SYN – ACK đến máy mục tiêu để làm cạn kiệt tài nguyên của nó.

Theo logic đây là kiểu tấn công vector lợi dụng giao tiếp TCP trong đó máy chủ tạo ra gói tin SYN – ACK để xác nhận yêu cầu của máy khách. Để thực hiện tấn công, kẻ tấn công đã làm quá tải tài nguyên CPU RAM của máy chủ bằng cách gửi các gói tin SYN – ACK giả mạo.

### ***ACK và PUSH ACK Flood Attack***

Trong một phiên TCP đang hoạt động, ACK và PUSH ACK là các cờ được sử dụng để truyền thông tin đến và đi từ máy chủ và máy khách cho đến khi phiên kết thúc. Trong một cuộc tấn công ACK và PUSH ACK Flood, những kẻ tấn công gửi một lượng lớn các gói ACK và PUSH ACK giả mạo đến máy mục tiêu, khiến nó không hoạt động.

Máy chủ bị tấn công sẽ không thể xác định nguồn gốc của các gói tin bị làm sai lệch địa chỉ và máy chủ lúc đó sẽ lãng phí khả năng xử lý khi cố gắng xác định cách xử lý chúng.

## **2.2. Mạng Botnet**

### **2.2.1. Khái niệm**

Botnet [1] là một mạng bao gồm từ hàng trăm tới hàng triệu máy tính bị cài đặt mã độc. Các máy tính này vẫn hoạt động bình thường nhưng không biết rằng đã bị các kẻ xấu kiểm soát và điều khiển để đồng loạt tấn công từ chối dịch vụ vào một mục tiêu bất kì nào đó.

Khi đã chiếm được quyền điều khiển thông qua các mã độc đã được thực thi, kẻ tấn công sẽ sử dụng các máy tính này và lựa chọn thời điểm thích hợp để bắt đầu tiến hành tấn công từ chối dịch vụ. Với số lượng lớn các máy cùng tấn công vào một thời điểm, mục tiêu lúc này sẽ bị ngốn hết băng thông, dẫn tới tình trạng không thể đáp ứng các yêu cầu hợp lệ từ người dùng hợp pháp.

Tuy từ "Botnet" có thể dùng để chỉ một nhóm bot bất kỳ với mục đích tốt, nhưng nó cũng thường được dùng để chỉ một tập hợp các máy tính đang chạy các chương trình độc hại như là worm (sâu máy tính), trojan horse hay các backdoor. Các máy tính này có thể bị điều khiển từ xa thông qua các máy bot header chẳng hạn như máy chủ IRC với mục đích xấu.

Mỗi bot trong mạng botnet thường chạy ẩn và tuân theo chuẩn RFC 1459 (IRC). Thông thường, kẻ tạo botnet trước đó đã thỏa hiệp một loạt hệ thống bằng nhiều công cụ đa dạng. Các bot mới hơn có thể tự động quét môi trường của chúng và tự lây lan bản thân bằng cách sử dụng các lỗ hổng hệ

thống và mật khẩu yếu. Nếu một bot có thể quét và tự lây lan qua càng nhiều lỗ hổng hệ thống, thì nó càng trở nên nguy hiểm và gây khó khăn cho các nhà bảo mật an ninh mạng.

Các botnet dường như đã trở thành một phần quan trọng của Internet và chúng ngày càng khó phát hiện. Các mạng IRC truyền thống đã cấm truy cập đối với các botnet đã từng tồn tại ở đó. Do đó, để có thể điều khiển botnet thì cần phải tự tìm các server cho mình. Ngoài ra, botnet cũng có nhiều kiểu kết nối, chẳng hạn quay số (dial), ADSL và cáp.

### **2.2.2. Topology giao tiếp của mạng Botnet**

[11] Botnet có vô số loại hình dạng và kích cỡ. Do đó, chúng sử dụng một loạt các C&C topology để đối phó với các biện pháp phòng thủ, ngừng hoạt động hợp pháp và các nỗ lực chiếm quyền điều khiển. Sự phát triển này có nghĩa là một tội phạm điều hành mạng botnet sẽ có một số tùy chọn C&C topology được nghiên cứu kỹ lưỡng để làm cơ sở cho một mạng botnet mới. Các topology này đều có những ưu điểm và nhược điểm riêng.

Các Botnet C&C topology đã được tối ưu hóa để giảm thiểu sự cố mạng và lỗi hệ thống. C&C topology chính xác được lựa chọn bởi một tội phạm điều hành mạng botnet thường phản ánh rủi ro nhận thức được của cá nhân đó đối với việc tiếp tục truy cập lệnh và mô hình kinh doanh tài chính của mạng botnet đó.

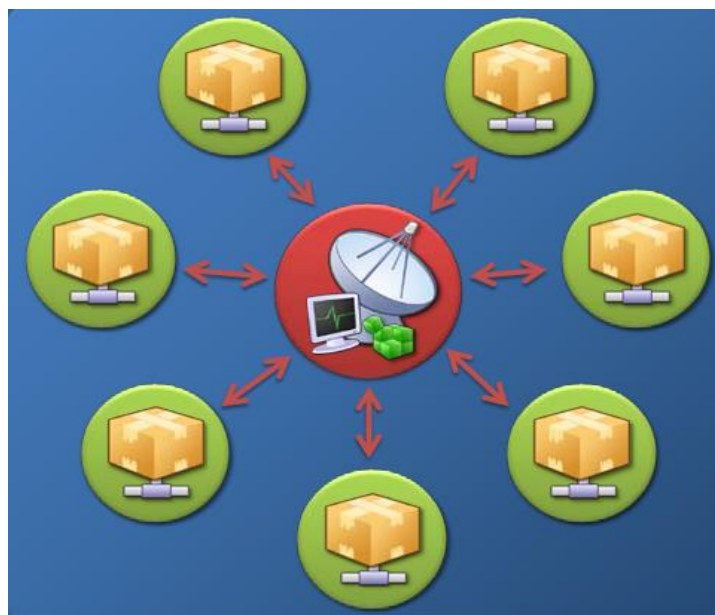
Các loại C&C topology trong thực tế thường có các loại sau:

- Star
- Multi-server
- Hierarchical
- Random

#### **2.2.2.1. Star**

Star topology [11] dựa trên một máy chủ C&C tập trung và duy nhất để giao tiếp với tất cả các bot trong mạng. Mỗi bot sẽ được nhận các lệnh hướng dẫn trực tiếp từ máy chủ C&C tập trung và duy nhất này.

Khi một máy tính nạn nhân trở thành một bot, nó thường được định cấu hình sẵn thành “phone home” đối với máy chủ C&C trung tâm này, sau đó nó sẽ tự động trở thành một thành viên trong mạng botnet và liên tục đợi và nhận lệnh hướng dẫn mới từ máy chủ C&C trung tâm.



Hình 2. 7: Star C&C topology

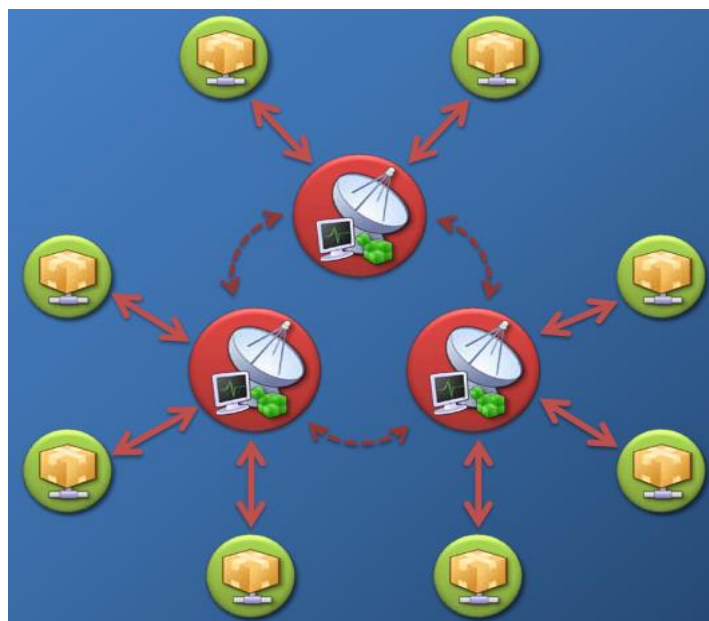
Ưu điểm	Nhược điểm
<b>Tốc độ kiểm soát</b> Giao tiếp trực tiếp một cách nhanh chóng giữa máy chủ C&C và các bot.	<b>Điểm lỗi duy nhất</b> Nếu máy chủ C&C trung tâm bị chặn hoặc bị vô hiệu hóa, thì mạng botnet sẽ trở nên vô dụng.

#### 2.2.2.2. Multi – Server

Multi – Server C&C topology [11] là mô hình mở rộng dựa trên Star C&C topology, trong đó bao gồm nhiều máy chủ C&C được sử dụng để cung cấp lệnh hướng dẫn cho các bot trong mạng. Nhiều máy chủ C&C này sẽ giao tiếp với nhau khi chúng quản lý mạng botnet. Nếu một máy chủ C&C riêng lẻ bị lỗi hoặc bị vô hiệu hoá hoàn toàn, thì mạng botnet vẫn có thể tiếp tục hoạt động nhờ vào các máy chủ C&C còn lại.

Tuy nhiên, người điều hành mạng botnet này cần phải có kế hoạch chi tiết để xây dựng Multi – Server C&C. Một lợi ích khác là các bot giống nhau có thể được sử dụng cho cả Star C&C topology hoặc Multi – Server C&C topology.

Việc đặt nhiều máy chủ C&C giữa các vị trí khác nhau có thể tăng tốc độ liên lạc với các bot có vị trí tương tự. Không chỉ vậy, việc đặt các máy chủ C&C đồng thời ở nhiều quốc gia làm cho mạng botnet trở nên quy mô lớn hơn và làm tăng khả năng chống lại việc bị shutdown.



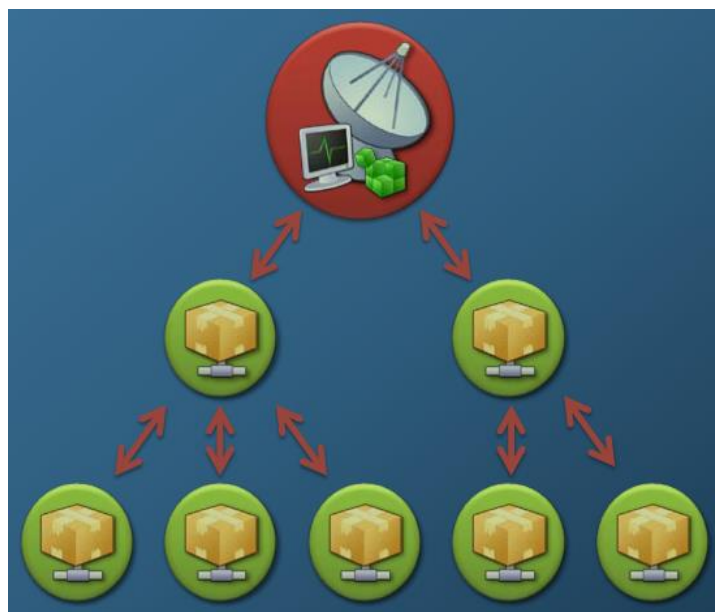
Hình 2. 8: Multi – Server C&C topology

Ưu điểm	Nhược điểm
<p><b>Không có điểm lỗi nào</b> Nếu một máy chủ C&amp;C bất kỳ bị vô hiệu hóa, mạng botnet vẫn có thể duy trì nhờ vào các máy chủ C&amp;C còn lại.</p> <p><b>Tối ưu hóa vị trí địa lý</b> Nhiều máy chủ C&amp;C được đặt ở nhiều nơi khác nhau làm tăng tốc độ liên lạc giữa các bot trong mạng botnet.</p>	<p><b>Yêu cầu lập kế hoạch trước</b> Cần có sự chuẩn bị và hiểu biết để xây dựng cơ sở hạ tầng Multi – Server C&amp;C.</p>

### 2.2.2.3. Hierarchical

Hierarchical topology [11] phản ánh động lực của các phương pháp được sử dụng trong quá trình thỏa hiệp và lan truyền của chính các bot. Các bot có khả năng ủy quyền các lệnh C&C mới cho các bot thế hệ sau. Tuy nhiên, các lệnh hướng dẫn được cập nhật thường gặp phải các vấn đề về độ trễ khiến người điều hành mạng botnet khó sử dụng botnet cho các hoạt động yêu cầu thời gian thực.

Hierarchical botnet (botnet phân cấp) có nghĩa là không có bot nào nhận biết được vị trí của toàn bộ mạng botnet. Cấu hình này gây ra thách thức, khó khăn cho các nhà bảo mật mạng trong việc mô tả cũng như ước tính kích thước tổng thể của mạng botnet. Hierarchical topology cũng tạo điều kiện thuận lợi cho việc tạo ra các mạng botnet nhỏ từ các mạng botnet lớn để bán lại hoặc cho thuê cho những ai cần sử dụng cho mục đích cá nhân của họ.



Hình 2. 9: Hierarchical C&C topology

Ưu điểm	Nhược điểm
<p><b>Che giấu/ ẩn mạng botnet</b> Việc chặn hoặc chiếm quyền điều khiển của các bot sẽ làm hiển thị tất cả các bot khác trong mạng botnet và sẽ không làm phát hiện ra máy chủ C&amp;C.</p> <p><b>Dễ bán lại hoặc cho thuê</b> Một người điều hành mạng botnet có thể dễ dàng cắt các phần của mạng botnet của họ để cho thuê hoặc bán lại cho các cá nhân, tổ chức khác có nhu cầu sử dụng.</p>	<p><b>Độ trễ lệnh</b> Bởi vì các lệnh phải đi qua nhiều nhánh giao tiếp trong mạng botnet, có thể có độ trễ cao với các lệnh cập nhật được nhận bởi các bot thế hệ sau. Sự chậm trễ này làm cho một số bot không thể đồng bộ tham gia vào một cuộc tấn công trong thời gian thực. Đồng thời cũng làm giảm chất lượng của mạng botnet.</p>

#### 2.2.2.4. Random

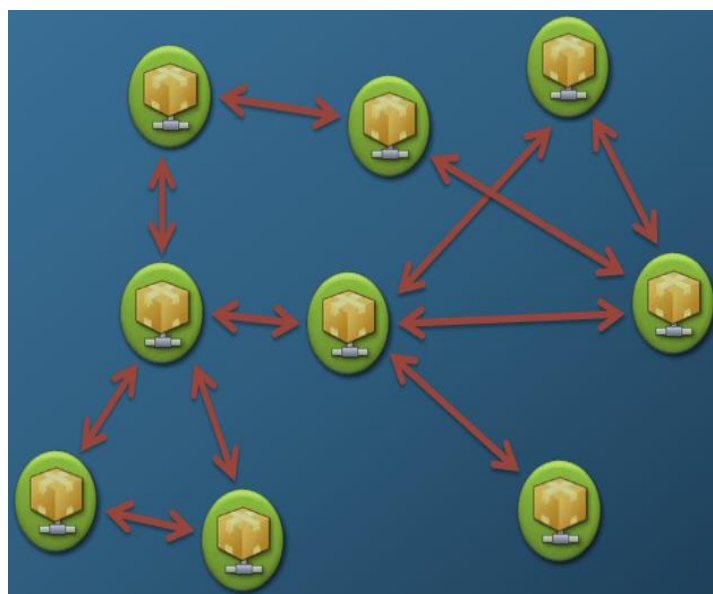
Các mạng botnet sử dụng Random topology [11] sẽ không có cơ sở hạ tầng máy chủ C&C tập trung. Thay vào đó, các lệnh được đưa vào mạng botnet có thể thông qua bất kỳ bot nào. Các lệnh này thường được “ký” là có thẩm quyền, điều này yêu cầu các bot tự động truyền các lệnh cho tất cả các bot khác đang tham gia vào mạng botnet.

Random topology có khả năng chống bị vô hiệu hoá hoàn toàn và chiếm quyền điều khiển cao vì chúng không cần dùng tới máy chủ C&C tập trung và sử dụng nhiều đường giao tiếp giữa các bot. Tuy nhiên, chúng ta thường dễ dàng xác định các bot còn lại của mạng botnet bằng cách theo dõi một bot và phân tích các giao tiếp của nó với các máy khác bên ngoài. Và nếu phát hiện có



sự bất thường trong các giao tiếp thì có thể xác định ra các bot khác trong mạng bonet.

Độ trễ của lệnh là một vấn đề còn tồn tại đối với Random topology. Tuy nhiên, nhiều liên kết giao tiếp giữa các bot làm cho độ trễ ít bất lợi hơn so với Hierarchical topology.



Hình 2. 10: Random C&C topology

Ưu điểm	Nhược điểm
<b>Có khả năng phục hồi cao</b> Việc thiếu cơ sở hạ tầng máy chủ C&C tập trung và nhiều liên kết giao tiếp giữa các bot khiến nó rất dễ khôi phục lại khi bị vô hiệu hoá.	<b>Độ trễ lệnh</b> Bản chất đặc biệt của các liên kết giữa các bot làm cho giao tiếp C&C không thể đoán trước được, điều này có thể dẫn đến mức độ trễ cao đối với một số cụm bot.
	<b>Truy tìm botnet</b> Việc giám sát, phân tích các thông tin liên lạc từ một máy chủ duy nhất bị bot xâm nhập có thể liệt kê các bot khác của mạng botnet.

Kết luận chương: chúng ta có thể thấy được mối quan hệ giữa các cuộc tấn công DDoS và các mạng Botnet được sử dụng trong quá trình tấn công. Chúng có mối quan hệ phụ thuộc lẫn nhau, tấn công DDoS thông qua mạng Botnet để thực hiện, còn mạng Botnet được xây dựng để thực hiện các cuộc tấn công quy mô lớn, trong đó có tấn công DDoS.



## CHƯƠNG 3

# CƠ SỞ LÝ THUYẾT

### 3.1. Học có giám sát (Supervised Learning)

Học có giám sát (tiếng Anh là Supervised Learning) là thuật toán dự đoán đầu ra (output) của một dữ liệu mới (new input) dựa trên các cặp (input, output) đã biết từ trước. Cặp dữ liệu này còn được gọi là (data, label), tức (dữ liệu, nhãn). Supervised Learning là nhóm phổ biến nhất trong các thuật toán máy học [18].

Nói theo cách toán học, học có giám sát là khi chúng ta có một tập hợp biến đầu vào  $X = \{x_1, x_2, \dots, x_n\}$  và một tập hợp nhãn tương ứng  $Y = \{y_1, y_2, \dots, y_n\}$ , trong đó  $x_i, y_i$  là các vector. Các cặp dữ liệu biết trước  $(x_i, y_i)$  được gọi là tập training data (dữ liệu huấn luyện). Từ tập training data này, chúng ta cần tạo ra một hàm số ánh xạ mỗi phần tử từ tập  $X$  sang một phần tử (xấp xỉ) tương ứng của tập  $Y$ , sao cho:

$$y_i \approx f(x_i), \forall i = 1, 2, \dots, N$$

Mục đích là xấp xỉ hàm số  $f$  thật tốt để khi có một dữ liệu  $x$  mới, chúng ta có thể tính được nhãn tương ứng của nó  $y = f(x)$  [18].

Học có giám sát được chia thành 2 dạng bài toán chính sau:

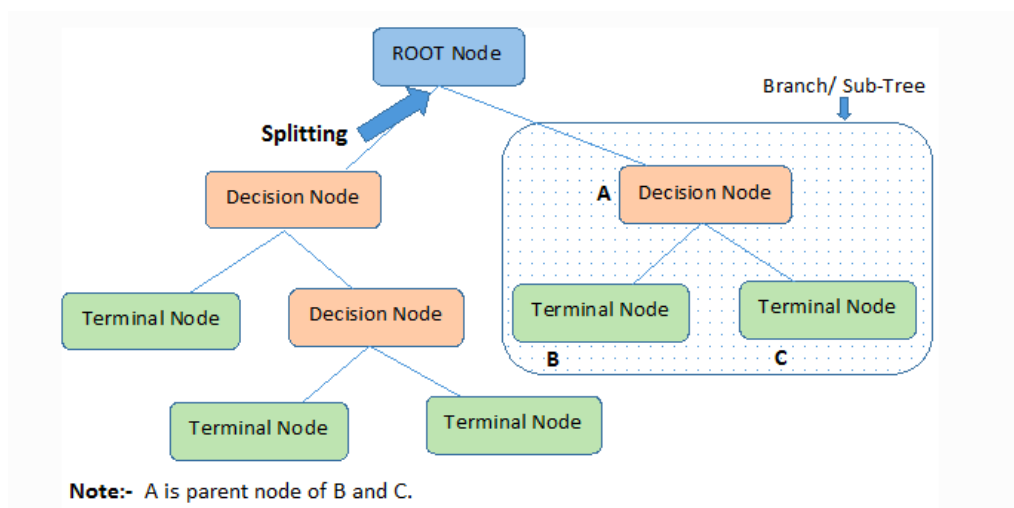
- Phân loại (Classification): Một bài toán được gọi là phân loại nếu các label của input data được chia thành một số hữu hạn nhóm. Ví dụ: Gmail xác định xem một email có phải là spam hay không; các hãng tín dụng xác định xem khách hàng có khả năng thanh toán nợ hay không [18].
- Hồi quy (Regression): Trường hợp này, label sẽ không chia thành các nhóm mà là một giá trị cụ thể. Ví dụ: một căn nhà rộng  $x$  m<sup>2</sup>, có  $y$  phòng ngủ và cách trung tâm thành phố  $z$  km sẽ có giá là bao nhiêu [18].

### 3.2. Các thuật toán phân loại trong học có giám sát (Supervised Learning)

#### 3.2.1. Decision Tree

[19] Decision Tree (tạm dịch là cây quyết định) là một phương pháp học tập có giám sát phi tham số được sử dụng để phân loại và hồi quy. Mục đích là tạo ra một mô hình dự đoán giá trị của một biến mục tiêu bằng cách tìm hiểu các quy tắc quyết định đơn giản được suy ra từ các cột/ tính năng dữ liệu.

Việc phân tách quyết định được thực hiện bằng cách sử dụng Gini index, Chi-Square, Information Gain (thông tin thu được) hoặc Reduction in Variance (giảm phương sai).



Hình 3. 1: Ví dụ về cấu trúc cây quyết định (Decision Tree) [37]

Ưu điểm:

- Dễ hiểu và dễ hình dung.
- Có thể xử lý cả dữ liệu số và dữ liệu biến/ phân loại.
- Có thể xử lý vấn đề đa nhãn lớp (multi-class) với bài toán phân loại nhiều lớp.
- Có thể đánh giá một mô hình bằng cách sử dụng các bài kiểm tra thống kê.

Nhược điểm:

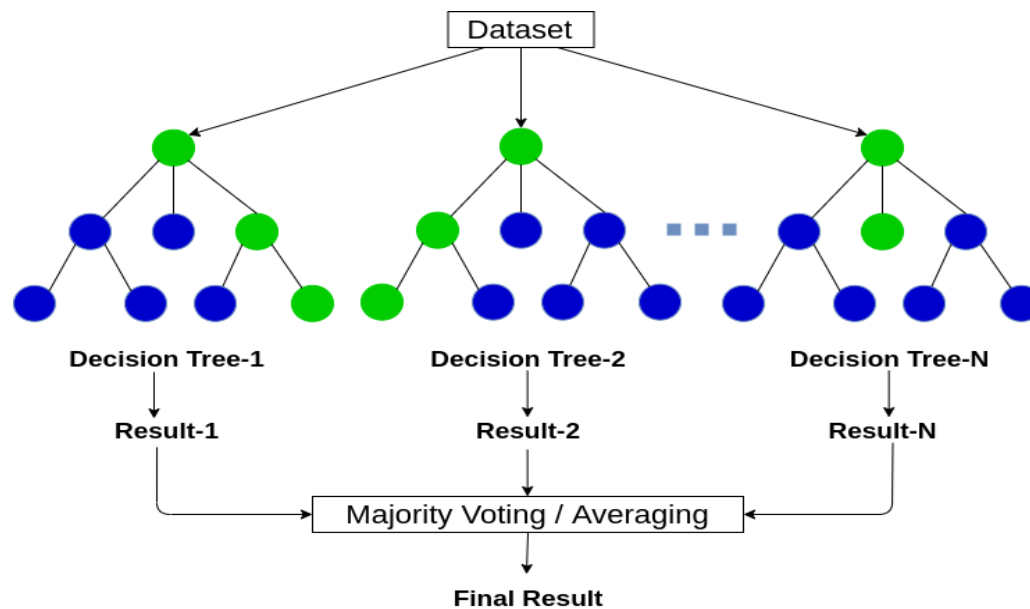
- Có thể tạo cây quá phức tạp mà không tổng quát hóa dữ liệu tốt sẽ xảy ra trường hợp overfitting.
- Việc thay đổi dữ liệu có thể khiến cây quyết định có cấu trúc không tốt, điều này có thể ảnh hưởng đến độ chính xác của mô hình máy học.
- Cây quyết định có thể là một cây thiên vị nếu một số lớp chiếm ưu thế hơn so với những lớp còn lại.

### 3.2.2. Random Forest

Thuật toán Random Forest (rừng ngẫu nhiên) là một rừng tập hợp nhiều Decision Tree làm cho nó trở nên mạnh mẽ hơn và mang lại độ chính xác cao hơn. Các cột/ tính năng ngẫu nhiên được chọn trong quá trình cảm ứng. Dự đoán được thực hiện bằng cách tổng hợp đa số phiếu bầu của từng cây cho phân loại hoặc lấy trung bình cho hồi quy trong tổng số các dự đoán của tập hợp [12]. Mỗi cây được tạo như sau:

- Bằng cách lấy mẫu N ngẫu nhiên, nếu số trường hợp trong tập huấn luyện là N nhưng có sự thay thế, từ dữ liệu ban đầu. Mẫu này sẽ được sử dụng làm bộ đào tạo để trồng cây.

- Đối với  $M$  số biến đầu vào, biến  $m$  được chọn sao cho  $m < M$  được xác định tại mỗi nút,  $m$  biến được chọn ngẫu nhiên từ  $M$  và phân tách tốt nhất trên  $m$  này được sử dụng để tách nút. Trong quá trình rừng phát triển, giá trị của  $m$  không đổi.
- Mỗi cây được phát triển ở mức độ lớn nhất có thể. Không sử dụng cắt tỉa.

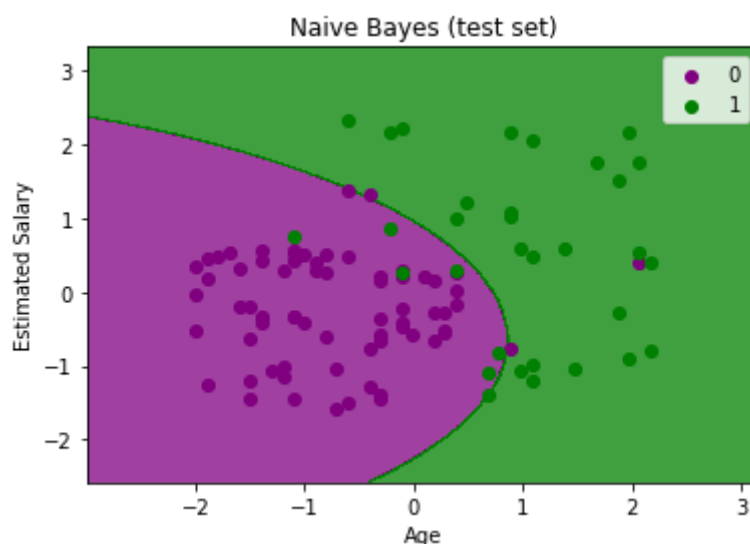


Hình 3. 2: Ví dụ về mô hình rừng ngẫu nhiên (Random Forest) [38]

### 3.2.3. Naive Bayes

Thuật toán Naive Bayes là một thuật toán học có giám sát, dựa trên định lý Bayes và được sử dụng để giải các bài toán phân loại. Nó chủ yếu được sử dụng trong phân loại văn bản bao gồm tập dữ liệu đào tạo với số chiều lớn [20].

Bộ phân loại Naive Bayes là một trong những thuật toán phân loại đơn giản và hiệu quả nhất giúp xây dựng các mô hình học máy nhanh có thể đưa ra dự đoán nhanh chóng. Nó còn là một bộ phân loại theo xác suất, có nghĩa là nó dự đoán trên cơ sở xác suất của một đối tượng [20].



Hình 3. 3: Ví dụ về phân loại với Naive Bayes [39]

Một nhược điểm của Naive Bayes là nó giả định rằng tất cả các cột/ tính năng là độc lập hoặc không liên quan, vì vậy nó không thể tìm hiểu mối quan hệ giữa các cột/ tính năng [20].

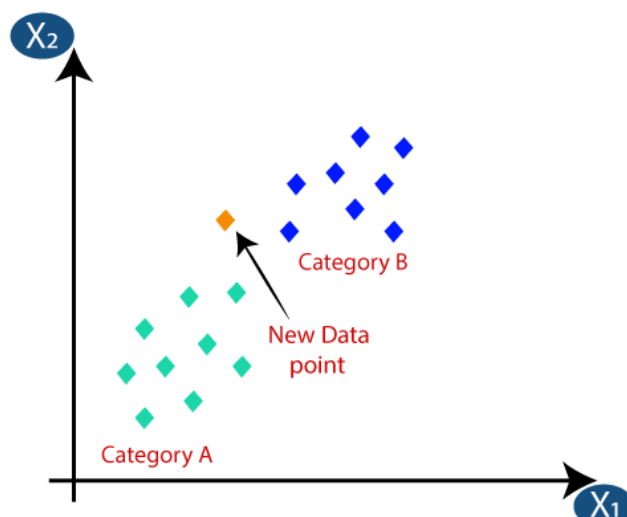
#### 3.2.4. K-Nearest Neighbors (KNN)

[21] Thuật toán K-Nearest Neighbors (KNN) là một kỹ thuật học có giám sát (supervised learning) dùng để phân loại quan sát mới bằng cách tìm điểm tương đồng giữa quan sát mới này với dữ liệu sẵn có. Thuật toán K-NN giả định sự giống nhau giữa trường hợp/ dữ liệu mới và các trường hợp có sẵn và đặt trường hợp mới vào danh mục giống nhất với các danh mục có sẵn.

Cách thức hoạt động của K-Nearest Neighbors:

- Bước 1: Chọn số K cho neighbors (hàng xóm).
- Bước 2: Tính khoảng cách Euclidean của K số hàng xóm.
- Bước 3: Lấy K hàng xóm gần nhất theo khoảng cách Euclidean được tính toán.
- Bước 4: Trong số K lân cận này, đếm số điểm dữ liệu trong mỗi loại.
- Bước-5: Gán các điểm dữ liệu mới cho danh mục đó mà số lượng hàng xóm là tối đa.

Ví dụ cần phân loại một điểm dữ liệu mới thuộc loại A hoặc B.



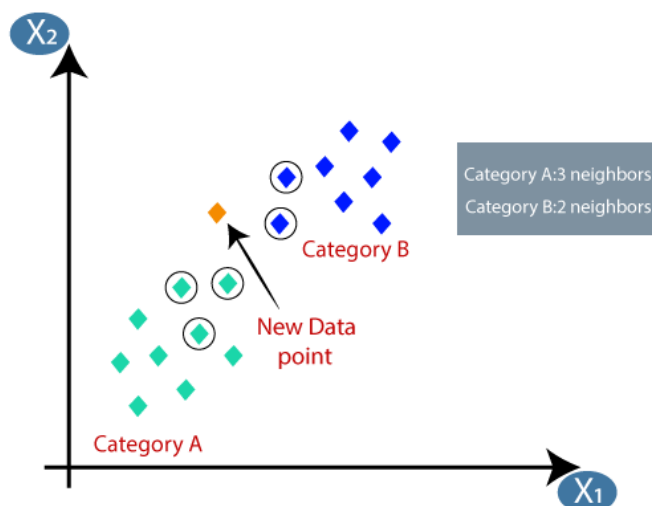
Hình 3. 4: Điểm dữ liệu mới cần phân loại [40]

Đầu tiên, chọn số lượng hàng xóm với  $k = 5$ . Tiếp theo, chúng ta tính toán khoảng cách Euclidean giữa các điểm dữ liệu. Công thức khoảng cách Euclidean như sau:

$$d = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

Ngoài ra còn có các đại lượng khoảng cách khác như Manhattan, Minkowski.

Sau khi tính toán khoảng cách Euclidean, chúng ta đã có được những hàng xóm gần nhất là 3 hàng xóm gần nhất trong loại A và 2 người hàng xóm gần nhất trong loại B.



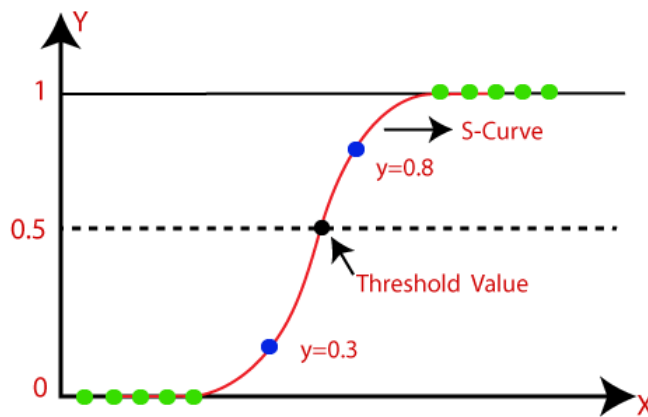
Hình 3. 5: Điểm dữ liệu mới được phân loại là A [40]

Cuối cùng, chúng ta có thể thấy 3 hàng xóm gần nhất thuộc loại A, do đó điểm dữ liệu mới này phải thuộc loại A.

### 3.2.5. Logistic Regression

[22] Thuật toán Logistic Regression mặc dù trong tên gọi có chứa cụm từ Regression (Hồi quy) nhưng nó cũng được dùng cho bài toán phân loại. Nó dự đoán biến phụ thuộc phân loại bằng cách sử dụng một tập hợp các biến độc lập nhất định.

Logistic Regression dự đoán đầu ra của một biến phụ thuộc phân loại. Do đó, kết quả phải là một giá trị phân loại hoặc rời rạc như Yes hoặc No, 0 hoặc 1, đúng hoặc Sai. Nhưng thay vì đưa ra giá trị chính xác là 0 và 1, nó cung cấp các giá trị xác suất nằm giữa 0 và 1.



Hình 3. 6: Ví dụ về Logistic Regression [41]

Trong thuật toán Logistic Regression có sử dụng Sigmoid Function là một hàm toán học được sử dụng để ánh xạ các giá trị dự đoán với xác suất. Nó sẽ ánh xạ bất kỳ giá trị thực nào thành một giá trị khác trong khoảng (0, 1). Điều này tạo ra một đường cong giống hình chữ S cũng chính là Sigmoid Function.

Công thức của Sigmoid Function:

$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1} = 1 - S(-x)$$

Bên cạnh đó, khái niệm giá trị ngưỡng (threshold value), xác định xác suất là 0 hoặc 1. Chẳng hạn như các giá trị trên giá trị ngưỡng có xu hướng là 1 và giá trị dưới giá trị ngưỡng có xu hướng bằng 0 như ví dụ trong hình trên.

### 3.2.6. XGBoost

XGBoost [13] là một tập hợp cây quyết định (Decision Tree Ensemble) dựa trên tăng cường độ dốc (Gradient Boosting). XGBoost xây dựng một mở rộng bổ sung của hàm mục tiêu (objective function) bằng cách giảm thiểu hàm mất mát (loss function). Nó thường dựa vào cây quyết định (Decision Tree) làm bộ phân loại cơ sở, một biến thể của hàm mất mát được sử dụng để kiểm soát độ phức tạp của cây là:

$$L_{\text{xgb}} = \sum_{i=1}^N L(y_i, F(x_i)) + \sum_{m=1}^M \Omega(h_m)$$

$$\Omega(h) = \gamma T + \frac{1}{2} \lambda \|w\|^2$$

Trong đó:

- T là số lá của cây
- $w$  là các điểm output của các lá.
- $\gamma$  là giá trị kiểm soát mức tăng giảm tổn thất tối thiểu cần thiết để tách một nút bên trong. Giá trị  $\gamma$  càng cao thì cây càng đơn giản.

[13] Ngoài ra, XGBoost còn thực hiện một số phương pháp để tăng tốc độ đào tạo của cây quyết định (Decision Tree) mà không liên quan đến độ chính xác của tổng thể. Nó sẽ tập trung vào việc giảm độ phức tạp tính toán để tìm ra sự phân tách tốt nhất. Các thuật toán tìm kiếm phân tách thường sẽ liệt kê tất cả các phân tách ứng viên và chọn một giải pháp có mức tăng cao nhất và yêu cầu thực hiện quét tuyến tính trên từng thuộc tính đã sắp xếp để tìm ra phân tách tốt nhất cho mỗi nút. Do đó, để tránh tình trạng dữ liệu lặp lại trong mọi nút, thuật toán XGBoost sử dụng cấu trúc dựa trên cột nén cụ thể, trong đó dữ liệu được lưu trữ được sắp xếp trước. Theo cách này, mỗi thuộc tính chỉ cần được sắp xếp đúng 1 lần.

Cuối cùng, thuật toán XGBoost hoạt động tốt trong các cuộc thi học máy vì khả năng xử lý mạnh mẽ của nhiều loại dữ liệu, mối quan hệ, phân phối và nhiều loại siêu tham số có thể được tinh chỉnh. Bên cạnh đó, XGBoost được sử dụng cho các vấn đề hồi quy, phân loại và xếp hạng.


### 3.3. Các kỹ thuật dùng trong giai đoạn tiền xử lý dữ liệu (preprocessing)

#### 3.3.1. Xử lý các giá trị rỗng trong tập dữ liệu

Đối với tập dữ liệu có chứa các giá trị rỗng thì chúng ta thường có 4 kỹ thuật sau:

- Loại bỏ các cột/ tính năng có chứa 1 hoặc nhiều giá trị rỗng.

Name	Age
Trung	21
Cường	22
Duy	23
NaN	25




Age
21
22
23
25

Hình 3. 7: Ví dụ loại bỏ cột/ tính năng chứa giá trị rỗng

Ưu điểm của kỹ thuật này là đơn giản dễ thực hiện nhưng có nguy cơ làm mất đi các cột/ tính năng quan trọng của tập dữ liệu gây ra hiệu suất thấp khi sử dụng các mô hình máy học.

- Loại bỏ các hàng/ mẫu có chứa 1 hoặc nhiều giá trị rỗng.

Name	Age
Trung	21
NaN	22
Duy	23
NaN	25




Name	Age
Trung	21
Duy	23

Hình 3. 8: Ví dụ loại bỏ hàng/ mẫu chứa giá trị rỗng

Cũng giống với kỹ thuật loại bỏ cột/ tính năng ở phía trên là đơn giản dễ thực hiện nhưng khác là kỹ thuật này giữ lại tất cả các cột/ tính năng có thể quan trọng của tập dữ liệu. Tuy nhiên, nhược điểm của phương pháp này có thể gây ra sự mất cân bằng trên các lớp trong tập dữ liệu.

- Thay thế giá trị rỗng bằng 1 giá trị khác.

Name	Age
Trung	21
Cường	NaN
Duy	NaN
Thành	25




Name	Age
Trung	21
Cường	23
Duy	23
Thành	25

Hình 3. 9: Ví dụ thay thế giá trị rỗng bằng giá trị trung bình cộng

Có thể thay thế bằng nhiều giá trị như khác nhau như 0, 1, 2,... hoặc trung bình cộng của các giá trị trong cột/ tính năng đó. Giá trị được thay thế sẽ không chính xác trong hầu hết các trường hợp, nhưng nó thường dẫn đến các mô hình máy học chính xác hơn và sẽ tốt hơn so với 2 kỹ thuật trên.

- Thay thế giá trị rỗng bằng 1 giá trị khác có bổ sung cột/ tính năng mới.

Name	Age
Trung	21
Cường	NaN
Duy	NaN
Thành	25



Name	Age	Miss_value
Trung	21	False
Cường	23	True
Duy	23	True
Thành	25	False

Hình 3. 10: Ví dụ thay giá trị rỗng bằng giá trị khác có thêm cột/ tính năng




Kỹ thuật này cũng thay thế giá trị rỗng bằng 1 giá trị khác như kỹ thuật trên. Không chỉ vậy nó còn thêm vào 1 cột/ tính năng mới cho biết đâu là giá trị đã bị thay thế và không bị thay thế.

### 3.3.2. Xử lý các biến/ giá trị phân loại

Để xử lý tập dữ liệu có chứa các giá trị là biến phân loại thì chúng ta có 4 kỹ thuật sau:

- Loại bỏ các cột/ tính năng chứa biến phân loại.

Name	Age	Sex
Trung	21	Nam
Cường	23	Nam
Duy	24	Nam
Thanh	25	Nữ




Name	Age
Trung	21
Cường	23
Duy	24
Thanh	25

Hình 3. 11: Ví dụ loại bỏ các cột/ tính năng chứa biến phân loại

Kỹ thuật này đơn giản để thực hiện nhưng cần phải xem xét kỹ trước khi thực hiện để tránh loại bỏ các cột/ tính năng chứa thông tin quan trọng.

- Sử dụng kỹ thuật mã hoá thứ tự (Ordinal Encoding).

Protocol
TCP
UDP
TCP
ICMP
UDP




Protocol
0
1
0
2
1

Hình 3. 12: Ví dụ sử dụng kỹ thuật mã hoá thứ tự (Ordinal Encoding)

Kỹ thuật mã hoá thứ tự (Ordinal Encoding) sẽ mã hoá mỗi giá trị phân loại thành mỗi giá trị số nguyên duy nhất.

- Sử dụng kỹ thuật mã hoá One-hot Encoding.

Protocol
TCP
UDP
TCP
ICMP
UDP

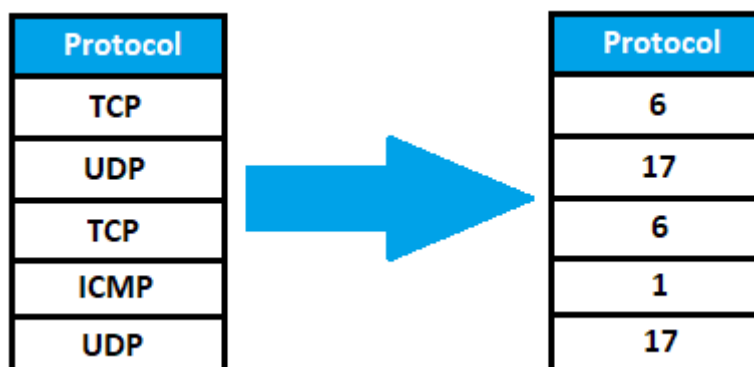


Protocol_TCP	Protocol_UDP	Protocol_ICMP
1	0	0
0	1	0
1	0	0
0	0	1
0	1	0

Hình 3. 13: Ví dụ sử dụng kỹ thuật mã hoá One-hot Encoding

Kỹ thuật này tạo ra các cột/ tính năng mới chỉ chứa 2 giá trị là 0 và 1. Giá trị sẽ là 1 nếu tương ứng với cột/ tính năng gốc và ngược lại sẽ là 0 nếu không tương ứng. Nhược điểm của kỹ thuật này là tạo ra 1 số lượng lớn các cột/ tính năng làm tăng kích thước tập dữ liệu và số lượng đầu vào của bài toán. Do đó, chỉ nên sử dụng trên các cột/ tính năng có chứa ít hơn 15 biến phân loại.

- Sử dụng mã hoá với giá trị số.



Hình 3. 14: Ví dụ sử dụng mã hoá với giá trị số

Kỹ thuật này có phần tương tự kỹ thuật mã hoá thứ tự (Ordinal Encoding) nhưng thay vì lựa chọn giá trị số theo thứ tự liên tiếp nhau thì ta có thể gán giá trị số không theo thứ tự tùy mục đích của mình. Ví dụ trên thay thế các giá trị trong cột/ tính năng ‘Protocol’ thành các giá trị số tương ứng của các giao thức đó (với TCP là 6, UDP là 17 và ICMP là 1).

### 3.3.3. Kỹ thuật Feature Selection

Kỹ thuật lựa chọn tính năng (Feature Selection) là kỹ thuật chọn ra các cột/ tính năng sao cho các cột/ tính năng này quan trọng nhất đối với mô hình máy học. Đồng thời, nó còn giúp giảm đi các cột/ tính năng nhiễu, không cần thiết có thể làm tăng thời gian đào tạo và gây ra hiệu suất thấp trên các mô hình máy học. Do đó, đây là một kỹ thuật quan trọng trong quá trình tiền xử lý dữ liệu.

#### 3.3.3.1. Phương pháp lọc (Filter Methods)

Các phương pháp bộ lọc (Filter methods) đánh giá mức độ phù hợp của các yếu tố dự báo bên ngoài các mô hình dự đoán và sau đó chỉ lập mô hình các yếu tố dự báo vượt qua một số tiêu chí. Ví dụ, đối với các bài toán phân loại, mỗi dự báo có thể được đánh giá riêng để kiểm tra xem có mối quan hệ hợp lý giữa nó và các lớp được quan sát hay không. Khi đó, chỉ những yếu tố dự đoán có mối quan hệ quan trọng mới được đưa vào một mô hình phân loại [14].

Filter methods đánh giá mức độ liên quan của các tính năng bằng cách chỉ xem xét các thuộc tính nội tại của dữ liệu. Trong hầu hết các trường hợp,

điểm mức độ liên quan của tính năng được tính toán và các tính năng cho điểm thấp sẽ bị xóa. Sau đó, tập hợp con các tính năng này được trình bày như là đầu vào cho thuật toán phân loại. Ưu điểm của Filter methods là chúng dễ dàng chia tỷ lệ thành các tập dữ liệu có chiều rất cao, tính toán đơn giản và nhanh chóng, và chúng không phụ thuộc vào thuật toán phân loại. Do đó, việc lựa chọn đối tượng địa lý chỉ cần được thực hiện một lần và sau đó các bộ phân loại khác nhau có thể được đánh giá [15].

[23] Tác giả Jason Brownlee đã xem xét một số biện pháp thống kê đơn biến có thể được sử dụng cho việc lựa chọn tính năng dựa trên các phương pháp lọc. Cụ thể là đối với trường hợp đầu vào là biến số, đầu ra là biến phân loại hoặc đầu vào là biến phân loại, đầu ra là biến số thì có thể dùng phương pháp thống kê ANOVA. Ngoài ra, đối trường hợp đầu vào và đầu ra đều là biến phân loại thì nên dùng phương pháp Chi-Squared.

### ***Phương pháp Chi-Squared***

Chi-Squared là kiểm định giả thuyết thống kê có giá trị để thực hiện khi thống kê kiểm định được phân phối Chi-Squared theo giả thuyết rỗng, cụ thể là kiểm định Pearson's Chi-Squared và các biến thể của chúng [24]. Chi-Squared kiểm tra sự khác biệt giữa giá trị quan sát được và giá trị kỳ vọng. Chi-Square cho thấy hoặc theo một cách nào đó kiểm tra mối quan hệ giữa hai biến phân loại có thể được tính toán bằng cách sử dụng tần suất quan sát đã cho và tần suất kỳ vọng [25].

Công thức của Chi-Squared:

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

Trong đó:

- $c$  là độ tự do.
- $O_i$  là giá trị quan sát được.
- $E_i$  là giá trị kỳ vọng.

### ***Phương pháp ANOVA***

ANOVA là viết tắt của Analysis of Variance (phân tích phương sai), nó là một kỹ thuật thống kê chỉ ra sự khác biệt giữa hai hoặc nhiều means hoặc các thành phần thông qua các bài kiểm tra ý nghĩa. Nó cũng cho chúng ta thấy một cách để so sánh nhiều means của một số quần thể. Thử nghiệm Anova được thực hiện bằng cách so sánh hai loại biến thiên, sự thay đổi giữa các trung bình của mẫu, cũng như sự thay đổi trong mỗi mẫu [26].

ANOVA Coefficient (hiệu số Anova - F) hay F-test được sử dụng để so sánh các yếu tố của độ lệch tổng với công thức sau:

$$F = \frac{MSB}{MSW}$$

Trong đó:

- MSB là tổng bình phương trung bình giữa các nhóm.
- MSW là tổng bình phương trung bình trong các nhóm.

### 3.3.4. Xử lý tập dữ liệu mất cân bằng

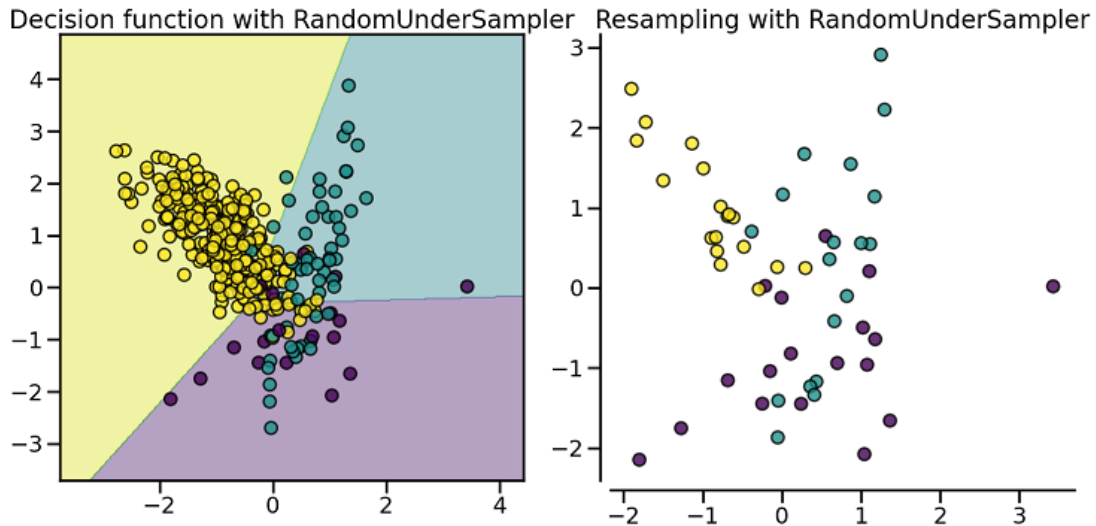
Các mô hình máy học thường có xu hướng học tập nghiêng về các lớp mục tiêu có nhiều dữ liệu/ sample hơn trên tập dữ liệu. Do đó, cần phải làm cho tập dữ liệu trở nên cân bằng hay ít nhất là không có sự chênh lệch quá lớn giữa các lớp mục tiêu. Để giải quyết vấn đề này, chúng ta có thể sử dụng các kỹ thuật cân bằng dữ liệu được trình bày dưới đây.

#### 3.3.4.1. Kỹ thuật Controlled Under-Sampling

Kỹ thuật Controlled Under-Sampling (tạm dịch là lấy mẫu dưới có kiểm soát) nằm trong nhóm thuật toán Prototype Selection (thuật toán lựa chọn mẫu từ tập S ban đầu và tạo ra tập S' mới sao cho  $|S'| < |S|$  và  $S' \subset S$ ). Kỹ thuật Controlled Under-Sampling cho phép thực hiện chiến lược lấy mẫu trong đó người dùng chỉ định số lượng mẫu trong S'. Trong kỹ thuật này còn có 2 kỹ thuật nhỏ khác thường dùng là RandomUnderSampler và NearMiss.

##### *RandomUnderSampler*

RandomUnderSampler là kỹ thuật được sử dụng để thực hiện việc cân bằng bộ dữ liệu một cách nhanh chóng và dễ dàng. Nó cân bằng bộ dữ liệu bằng cách chọn ngẫu nhiên một tập hợp con dữ liệu từ bộ dữ liệu ban đầu cho các lớp mục tiêu. Vì thuộc kỹ thuật Controlled Under-Sampling (lấy mẫu dưới có kiểm soát) nên ta có thể chỉ định số lượng mẫu cho mỗi lớp mục tiêu thông qua tham số `sampling_strategy` trong lớp RandomUnderSampler từ thư viện `imblearn.under_sampling`.



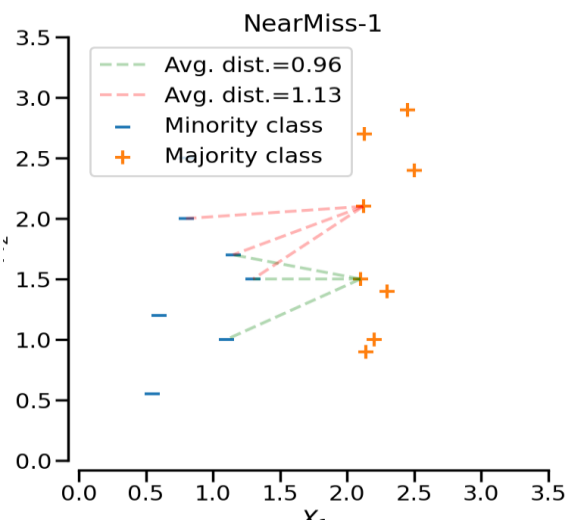
Hình 3. 15: Ví dụ khi sử dụng RandomUnderSampler [42]

### NearMiss

Near Miss là kỹ thuật dựa trên thuật toán nearest neighbors (tạm dịch là láng giềng gần nhất). Nó lựa chọn mẫu dựa trên khoảng cách của các mẫu trong lớp đa số (các lớp mục tiêu có nhiều mẫu nhất) đến các mẫu trong lớp thiểu số (các lớp mục tiêu có ít mẫu nhất) [16, 27].

NearMiss có 3 phiên bản bao gồm NearMiss-1, NearMiss-2 và NearMiss-3.

- NearMiss-1 lựa chọn các mẫu từ lớp đa số có khoảng cách trung bình nhỏ nhất đến 3 mẫu gần nhất từ lớp thiểu số.
- NearMiss-2 lựa chọn các mẫu từ lớp đa số có khoảng cách trung bình nhỏ nhất đến 3 mẫu xa nhất từ lớp thiểu số.
- NearMiss-3 lựa chọn một số lượng mẫu nhất định cho lớp đa số với mỗi mẫu trong lớp thiểu số gần nhất.



Hình 3. 16: Ví dụ về NearMiss-1 lựa chọn mẫu dựa trên khoảng cách [42]

Tương tự như RandomUnderSampler, ta có thể dùng tham số `sampling_strategy` để chỉ định số lượng mẫu cho mỗi lớp mục tiêu và lựa chọn phiên bản cho NearMiss thông qua tham số `version` với 3 giá trị 1, 2, 3 ứng với từng phiên bản NearMiss.

### 3.3.5. Kỹ thuật Feature Scaling

Feature Scaling là một kỹ thuật đề cập đến việc chia tỷ lệ (scaling) hay chuẩn hoá dữ liệu sao cho nó phù hợp nhất với mô hình học máy. Vì dữ liệu đầu vào thường có sự chênh lệch khoảng giá trị hay chênh lệch giữa các đơn vị đo trên các cột/ tính năng đầu vào, do đó, đây được xem là một kỹ thuật quan trọng trong giai đoạn tiền xử lý. Thông thường có 2 kiểu chia tỷ lệ (scaling) thông dụng là normalization và standardization.

#### 3.3.5.1. Normalization

Normalization là một kỹ thuật chia tỷ lệ (scaling) trong đó các giá trị được dịch chuyển và thay đổi tỷ lệ để chúng nằm trong khoảng từ 0 đến 1. Nó còn được gọi với cái tên là Min-Max scaling [28].

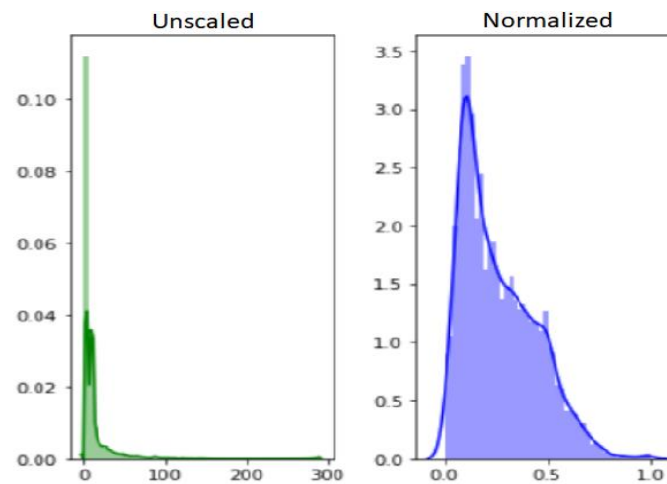
Công thức của kỹ thuật normalization:

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

Trong đó:

- $X_{\max}$  là giá trị lớn nhất của cột/ tính năng.
- $X_{\min}$  là giá trị nhỏ nhất của cột/ tính năng.

Khi  $X$  bằng với giá trị nhỏ nhất trong cột, tử số sẽ là 0 và  $X'$  sẽ bằng 0. Khi  $X$  bằng với giá trị lớn nhất trong cột thì tử số bằng mẫu số và  $X'$  sẽ bằng 1. Nếu giá trị của  $X$  nằm trong khoảng giá trị nhỏ nhất và giá trị lớn nhất trong cột/ tính năng đó thì giá trị của  $X'$  sẽ nằm trong khoảng từ 0 đến 1 [28]. Do đó, từ một cột/ tính năng với các giá trị khác nhau thì ta luôn có thể thay đổi chúng về khoảng giá trị từ 0 đến 1.



Hình 3. 17: Ví dụ khi scale dữ liệu về khoảng (0, 1) [43]

### 3.3.5.2. Standardization

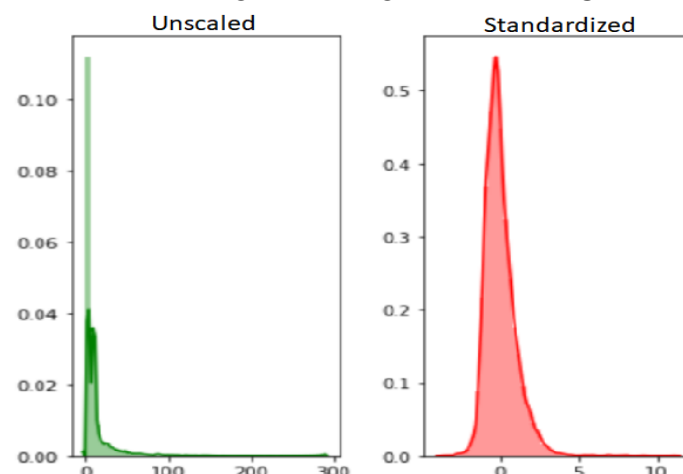
Standardization là một kỹ thuật chia tỷ lệ (scaling) khác trong đó các giá trị được tập trung xung quanh giá trị trung bình với độ lệch chuẩn đơn vị. Điều này có nghĩa là giá trị trung bình của thuộc tính trở thành 0 và phân phối kết quả có độ lệch chuẩn đơn vị [28].

Công thức của kỹ thuật standardization:

$$X' = \frac{X - \mu}{\sigma}$$

Trong đó:

- $\mu$  là giá trị trung bình của các giá trị trong cột/ tính năng.
- $\sigma$  là độ lệch chuẩn của các giá trị trong cột/ tính năng.



Hình 3. 18: Ví dụ khi thay đổi hình dạng dữ liệu với Standardization [43]

### 3.3.5.3. Khi nào nên sử dụng kỹ thuật Normalization hay Standardization

Đối với các thuật toán Tree-Based (dựa trên cây) như Decision Tree, Random Forest, XGBoost thì chúng khá nhạy cảm với việc chia tỷ lệ (scaling)

nên thường bất biến hoặc xảy ra thay đổi rất nhỏ khi thực hiện 2 kỹ thuật trên [28]. Nhưng bù lại nó có thể giúp giảm thời gian đào tạo các mô hình máy học.

Còn đối với các thuật toán dựa trên Gradient Descent như Linear Regression, Logistic Regression, Neural Network,... hay những thuật toán dựa trên khoảng cách như K-Nearest Neighbors, K-means và SVM thì việc chuẩn hoá dữ liệu bằng kỹ thuật normalization là một ý tưởng tốt nếu phân phối dữ liệu không tuân theo phân phối Gaussian. Mặt khác, kỹ thuật standardization có thể hữu ích trong trường hợp dữ liệu tuân theo phân phối Gaussian. Tuy nhiên, chúng ta có thể thử nghiệm xây dựng các mô hình máy học trên tập dữ liệu thô trước, sau đó dùng tới 2 kỹ thuật normalization và standardization để chọn ra trường hợp tối ưu nhất [28].

### **3.4. Công cụ, thư viện, ngôn ngữ, môi trường cài đặt hỗ trợ trong quá trình xây dựng ứng dụng**

#### **3.4.1. Công cụ Tcpdump**

Tcpdump [29] là công cụ hỗ trợ người dùng trong việc bắt gói tin và lưu trữ chúng dưới định dạng file pcap. Tcpdump sử dụng giao diện dòng lệnh để xuất ra các thông tin cần thiết.

Sau khi quá trình bắt gói tin hoàn tất hay bị buộc dừng bởi người dùng Tcpdump hiển thị các thông tin sau:

- Packet capture: số lượng gói tin bắt được và xử lý.
- Packet received by filter: số lượng gói tin được nhận bởi bộ lọc (phụ thuộc vào hệ điều hành đang chạy Tcpdump).
- Packet dropped by kernel: số lượng packet đã bị dropped bởi cơ chế bắt gói tin của hệ điều hành.

#### **3.4.2. Công cụ Argus**

Argus [30] là một công cụ kiểm tra giao dịch mạng dữ liệu mà nó phân loại và theo dõi các gói mạng khớp với biểu thức bộ lọc libpcap thành một mô hình giao dịch luồng mạng theo giao thức cụ thể. Các tính năng có thể được cấu hình thông qua tập tin argus.conf. Sử dụng cùng với Argus là tool Argus clients ‘ra’ giúp đọc các file từ Argus và chuyển chúng thành định dạng file NetFlow. Cấu hình Argus clients ‘ra’ thông qua tập tin ra.conf.

#### **3.4.3. Python sys module**

Module sys [31] giúp cung cấp quyền truy cập vào một số biến được trình thông dịch sử dụng hoặc duy trì và đến các hàm tương tác mạnh với trình thông dịch. Trong đó, hàm sys.argv trả về danh sách các đối số dòng lệnh được



chuyển đến một tập lệnh Python, argv[0] là tên tập lệnh luôn là mục ở chỉ số 0 và phần còn lại của các đối số được lưu trữ tại các chỉ mục tiếp theo.

#### **3.4.4. Thư viện subprocess**

Subprocess [32] là module cho phép tạo ra các process mới, kết nối với các input/output/error pipes của chúng và lấy mã return của chúng. Nói một cách đơn giản, subprocess giúp thực thi các lệnh shell hoặc chạy các ứng dụng khác từ một chương trình Python và thu về kết quả thực thi. Ví dụ, chúng ta có thể chạy công cụ tcpdump thông qua một đoạn mã Python sau:

```
tcpdump = subprocess.run(["sudo tcpdump -c 100 -i enth-0"], shell=True)
```

#### **3.4.5. Ngôn ngữ lập trình Python**

Ngôn ngữ lập trình Python [33] là ngôn ngữ lập trình hướng đối tượng được tạo ra bởi Guido van Rossum. Python giúp thực thi nhanh các đoạn mã đơn giản với cú pháp dễ hiểu. Không chỉ vậy, nó còn giúp tự động hoá các tác vụ hàng ngày và linh hoạt trong nhiều công việc, lĩnh vực khác nhau như phát triển web, xây dựng phần mềm ứng dụng/ game di động và đặc biệt là được sử dụng nhiều trong khoa học phân tích dữ liệu, Machine Learning hay thậm chí là trí tuệ nhân tạo (AI).

#### **3.4.6. Môi trường cài đặt**

VMware Workstation Pro [34] là phần mềm để chạy các tính máy ảo hoạt động như một máy chủ vật lý bình thường với tài nguyên và hệ điều hành riêng. Để hệ điều hành và ứng dụng của máy ảo có thể truy cập nhóm tài nguyên vật lý được chia sẻ, lúc này trình quản lý máy ảo Hypervisor sẽ lập lịch yêu cầu tới tài nguyên của hệ thống vật lý. Với phần mềm này chúng ta có thể tạo ra các máy ảo với nhiều hệ điều hành khác nhau hoặc tạo ra một mô hình mạng LAN để phục vụ cho các mục đích học tập cũng như nghiên cứu.

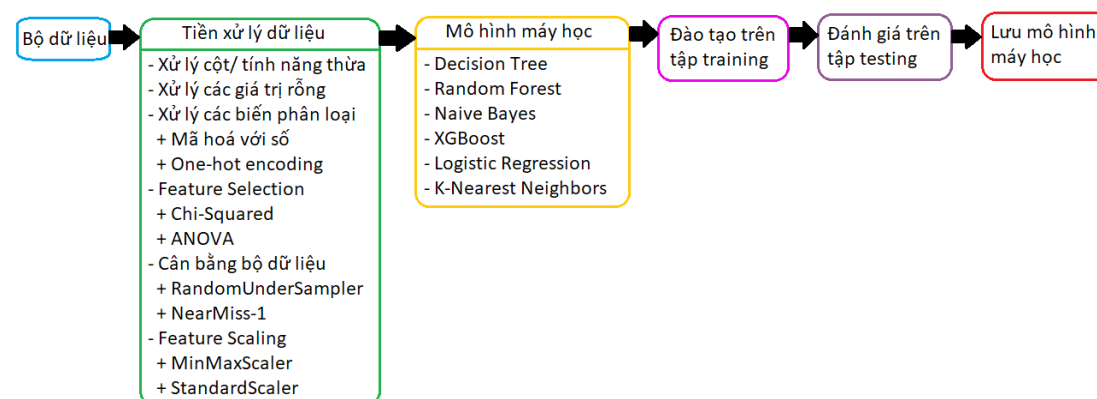
## CHƯƠNG 4

# XÂY DỰNG ỨNG DỤNG VÀ MÔ HÌNH MÁY HỌC

### 4.1. Xây dựng mô hình máy học

Hiện nay, có rất nhiều thư viện hỗ trợ các mô hình máy học như thư viện sklearn cho các thuật toán như Decision Tree, Random Forest,... và thư viện xgboost cho thuật toán XGBoost. Điều này giúp cho mọi người dễ dàng tiếp cận với Machine Learning hơn và thuận lợi trong việc xây dựng các mô hình máy học để phục vụ mục đích riêng của mỗi cá nhân.

Nhóm chúng tôi cũng áp dụng các thư viện có sẵn này để xây dựng các mô hình máy học phục vụ cho mục đích nghiên cứu này. Sơ đồ chung của quá trình xây dựng các mô hình máy học được thể hiện trong hình bên dưới.



Hình 4. 1: Sơ đồ chung của quá trình xây dựng các mô hình máy học

Ban đầu, bộ dữ liệu được sử dụng sẽ qua quá trình tiền xử lý để có được bộ dữ liệu chất lượng nhất cho các mô hình máy học. Sau đó sẽ đưa vào các thuật toán máy học để đào tạo trên tập training và đánh giá hiệu suất dựa trên tập testing. Tập training và tập testing là 2 tập dữ liệu đã được chia theo tỉ lệ (training: 80% và testing: 20%) trước khi đào tạo trên các mô hình máy học. Cuối cùng là chọn ra mô hình máy học đạt hiệu suất tối ưu nhất và lưu lại mô hình để sử dụng.

Trong bước chọn các mô hình máy học, chúng tôi đề xuất sử dụng 6 mô hình máy học như ví dụ trong hình trên. Các mô hình này được sử dụng thông qua 2 thư viện là sklearn và xgboost. Ở đây chúng tôi sử dụng các tham số mặc định cho các thuật toán máy học. Cụ thể, như sau:

- Decision Tree sử dụng lớp DecisionTreeClassifier(criterion='gini', splitter='best', max\_depth=None, min\_samples\_split=2, min\_samples\_leaf=1, min\_weight\_fraction\_leaf=0.0, max\_features=None,

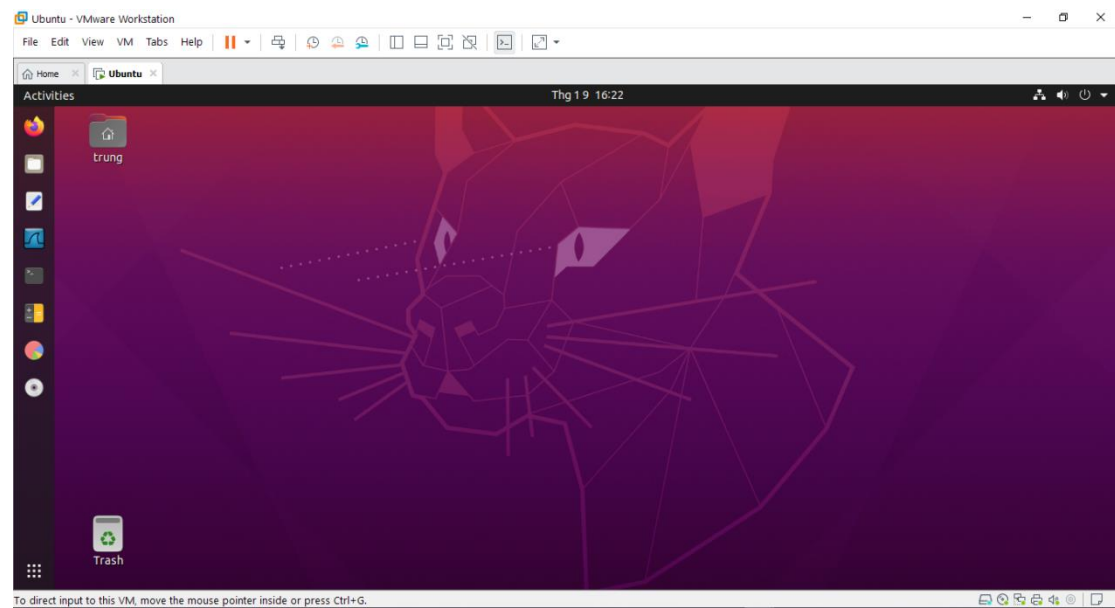
```
random_state=0, max_leaf_nodes=None, min_impurity_decrease=0.0,  
class_weight=None, ccp_alpha=0.0)
```

- Random Forest sử dụng lớp RandomForestClassifier(n\_estimators=100, \*, criterion='gini', max\_depth=None, min\_samples\_split=2, min\_samples\_leaf=1, min\_weight\_fraction\_leaf=0.0, max\_features='auto', max\_leaf\_nodes=None, min\_impurity\_decrease=0.0, bootstrap=True, oob\_score=False, n\_jobs=None, random\_state=0, verbose=0, warm\_start=False, class\_weight=None, ccp\_alpha=0.0, max\_samples=None)
- Naive Bayes sử dụng lớp GaussianNB(priors=None, var\_smoothing=1e-09)
- XGBoost sử dụng lớp XGBClassifier(base\_score=0.5, colsample\_bylevel=1, colsample\_bytree=1, gamma=0, learning\_rate=0.1, max\_delta\_step=0, max\_depth=10, min\_child\_weight=1, missing=None, n\_estimators=100, nthread=-1, objective='binary:logistic', reg\_alpha=0, reg\_lambda=1, scale\_pos\_weight=1, seed=0, silent=True, subsample=1)
- Logistic Regression sử dụng lớp LogisticRegression(penalty='l2', \*, dual=False, tol=0.0001, C=1.0, fit\_intercept=True, intercept\_scaling=1, class\_weight=None, random\_state=0, solver='lbfgs', max\_iter=100, multi\_class='auto', verbose=0, warm\_start=False, n\_jobs=None, l1\_ratio=None)
- K-Nearest Neighbors sử dụng lớp KNeighborsClassifier(n\_neighbors=5, \*, weights='uniform', algorithm='auto', leaf\_size=30, p=2, metric='minkowski', metric\_params=None, n\_jobs=None)

Vậy là chúng tôi đã trình bày xong phần xây dựng các mô hình máy học và các bước xây dựng này được thực hiện thông qua Google Colab. Nhưng để có thể sử dụng được mô hình máy học thì phải triển khai nó trên 1 ứng dụng hoặc công cụ, trang web nào đó,... Điều này sẽ được thực hiện trong phần sau. Lưu ý: các phần nhỏ trong bước tiền xử lý dữ liệu sẽ được trình bày rõ hơn trong chương 5.

## 4.2. Hiện thực cài đặt các công cụ, thư viện hỗ trợ

Trước khi đến với phần xây dựng ứng dụng, chúng tôi sẽ trình bày việc cài đặt các công cụ hỗ trợ cho quá trình xây dựng ứng dụng. Đầu tiên, chúng tôi cài đặt một máy ảo bằng phần mềm VMware Workstation Pro 15.5. Máy ảo được cấu hình sử dụng hệ điều hành Ubuntu 20.04.3 LTS với bộ xử lý Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz 2.30 GHz, 2 core, 2.8 GB RAM, ổ cứng 50 GB, card mạng NAT.



Hình 4. 2: Cài đặt hoàn tất máy ảo Ubuntu

Tiếp theo là cài đặt các công cụ như tcpdump và argus. Lệnh để cài đặt 2 công cụ này là: `sudo apt-get install tcpdump`, `sudo apt-get install argus-server` và `sudo apt-get install argus-client`.

Cuối cùng là import các thư viện và module:

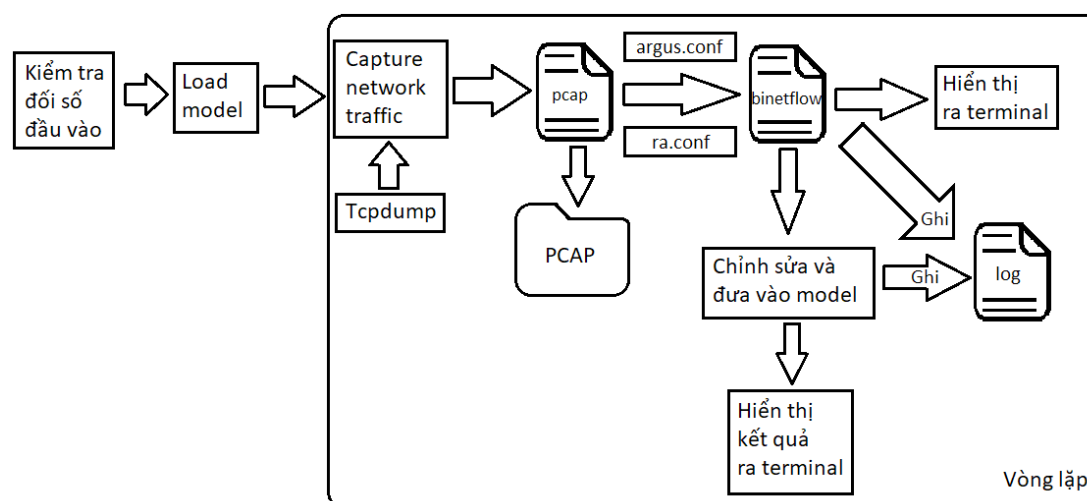
- Subprocess: `pip install subprocess.run`
- Pandas: `sudo apt install python3-pandas`
- Joblib: `pip install joblib`

### 4.3. Xây dựng ứng dụng

Sau khi đã xây dựng xong mô hình máy học và cài đặt các công cụ, thư viện cần thiết. Chúng tôi sẽ thực hiện xây dựng ứng dụng bằng ngôn ngữ lập trình Python3. Ứng dụng này chạy và sử dụng terminal của Ubuntu để hiển thị kết quả.

#### 4.3.1. Sơ đồ ứng dụng

Sơ đồ ứng dụng sẽ trình bày nội dung các bước thực hiện của mô hình.



Hình 4. 3: Sơ đồ ứng dụng

Các bước thực hiện của mô hình trên bao gồm:

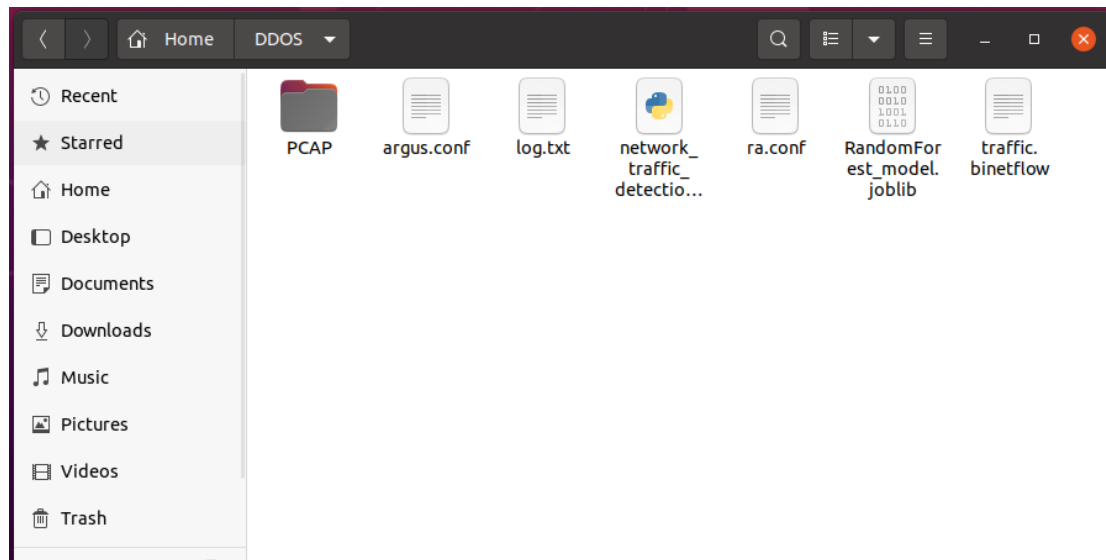
- Bước 1: Thực hiện kiểm tra đối số đầu vào, đối số là 2 giá trị gồm network interface card (card mạng) và số lượng gói tin trong 1 lần bắt.
- Bước 2: Khởi chạy/ tải model RandomForest\_model.joblib đã được đào tạo (sẽ trình bày trong chương 5).
- Bước 3: Bắt gói tin với công cụ Tcpdump, lưu về định dạng tập tin pcap và đồng thời lưu vào thư mục PCAP.
- Bước 4: Tập tin pcap sau đó sẽ được chuyển đổi thành tập tin binetflow thông qua công cụ Argus với 2 tập tin argus.conf và ra.conf.
- Bước 5: Khi đã có được tập tin binetflow, ta sẽ hiển thị thông tin này ra terminal và đồng thời ghi vào tập tin log.
- Bước 6: Đọc tập tin binetflow và chỉnh sửa cho phù hợp để đưa vào model.
- Bước 7: Hiển thị kết quả dự đoán từ model ra terminal và đồng thời ghi vào tập tin log.

Lưu ý nhỏ: bước 4 được thực hiện giống với cách mà nhóm tác giả trong bài báo [2] đã tạo ra bộ dữ liệu CTU-13 (bộ dữ liệu cùng quy trình tạo ra bộ dữ liệu sẽ được trình bày trong chương 5). Do đó, 2 tập tin cấu hình argus.conf và ra.conf sẽ được sử dụng lại từ trang web [35].

Các tập tin, thư mục sử dụng trong quá trình xây dựng ứng dụng bao gồm:

- Tập tin mã Python3 network\_traffic\_detection.py là chương trình thực thi chính của ứng dụng trên terminal.
- Hai tập tin argus.conf và ra.conf.
- Thư mục PCAP lưu trữ các tập tin pcap bắt được từ Tcpdump.

- Tập tin traffic.binnetflow được dùng để chứa tạm thông tin chuyển từ tập tin pcap sang định dạng netflow.
- Tập tin RandomForest\_model.joblib là model đã được đào tạo.
- Tập tin log.txt chứa các thông tin về tập tin traffic.bitnetflow cùng kết quả dự đoán tương ứng từ model máy học.



Hình 4. 4: Các tập tin, thư mục sử dụng trong quá trình xây dựng ứng dụng

### 4.3.2. Mã Python3

Tiếp theo là mã Python3 để xây dựng ứng dụng.

*Import các thư viện và module cần thiết*

Các thư viện được import vào phục vụ cho mục đích của ứng dụng này.

```
import subprocess as sp
import sys, os
import pandas as pd
import joblib
import warnings
warnings.filterwarnings('ignore')
```

*Hàm kiểm tra đối số đầu vào*

Đối số đầu vào ở đây bao gồm 2 giá trị, giá trị thứ nhất là card mạng (NIC) để bắt các gói tin, lưu lượng mạng đi qua và giá trị thứ hai là số lượng gói tin trong 1 lần bắt để đưa vào mô hình. Đồng thời, hàm cũng kiểm tra giá trị card mạng (NIC) có tồn tại hay không.

```
def check_input():
    if len(sys.argv) != 3:
        print("USE: python3 " + sys.argv[0] + " <NIC> <NUMBER_PACKET_CAPTURING>")
        print("USE: CTRL + C to exit")
        return 0
    elif len(sys.argv) == 3:
        if sys.argv[1] not in os.listdir('/sys/class/net/'):
            print(f"ERROR: Interface \"{sys.argv[1]}\" not found!!!")
            return 0
        else:
            return 1
```

### *Hàm chỉnh sửa tập tin traffic.binetflow*

Sau khi tệp pcap được chuyển đổi thành tệp traffic.binetflow, hàm bên dưới sẽ định dạng lại bằng cách trích xuất các cột tính năng và thay đổi các giá trị sao cho dữ liệu đầu vào sẽ tương ứng với các giá trị đầu vào của mô hình. Kết quả của hàm sẽ trả về một mảng 2 chiều (ma trận) cho đầu vào của mô hình học máy.

```
def edit_data(df):
    lst, lst1 = [], []
    for index, row in df.iterrows():
        lst2 = [row['Dur'], row['TotPkts'], row['TotBytes'], row['SrcBytes']]
        lst.extend(lst2)
        if row['Dir'] != '    ->':
            lst.extend([1,0])
        elif row['Dir'] != '    <->':
            lst.extend([0,1])
        else:
            lst.extend([0,0])
        if row['Proto'] != 'icmp':
            lst.extend([1,0,0])
        elif row['Proto'] == 'tcp':
            lst.extend([0,1,0])
        elif row['Proto'] == 'udp':
            lst.extend([0,0,1])
        else:
            lst.extend([0,0,0])
        if str(row['dTos']) == 'nan':
            lst.extend([1])
        else:
            lst.extend([0])
        lst3 = lst.copy()
        lst1.append(lst3)
        lst.clear()
    return lst1
```

### *Hàm main thực hiện các bước chính*

Cuối cùng là hàm main có nhiệm vụ gọi lại hai hàm trên và thực hiện theo các bước trong sơ đồ ứng dụng. Cụ thể, hàm main này sẽ gọi lại hàm check\_input(), nếu 2 đối số phù hợp thì sẽ thực hiện bước tiếp theo là load model máy học đã được xây dựng từ trước thông qua thư viện joblib.

Tiếp theo, một vòng lặp được thực hiện bao gồm các công việc theo thứ tự như bắt và lưu lại lưu lượng mạng vào tệp pcap, chuyển tệp pcap sang tệp

binetflow, ghi tệp binetflow vào tệp log, đọc tệp binetflow thành một dataframe thông qua thư viện pandas, gọi lại hàm edit\_data() để chỉnh sửa giá trị đầu vào cho tệp binetflow, đưa kết quả từ hàm edit\_data() vào mô hình để dự đoán bằng function predict() của mô hình, lấy kết quả từ function predict() là 1 mảng chứa các giá trị dự đoán gồm 0, 1, 2 cho ba lớp lưu lượng khác nhau, cuối cùng là đưa ra dự đoán, nếu tổng số lượng các dự đoán là 1 lớn hơn hoặc bằng độ dài của mảng đó thì kết luận là lưu lượng Botnet-DDoS, ngược lại thì kết luận là lưu lượng Background-Normal cho các trường hợp dự đoán là 0 và 2.

```
def main():
    if check_input():
        print('Loading model...')
        model_LR = joblib.load('RandomForest_model.joblib')
        print('Done!!!')
        i = 0
        while(True):
            nic = sys.argv[1]
            num_cap = sys.argv[2]
            pcap_file = 'capture_' + str(i) + '.pcap'

            # Use tcpdump to capture traffic and save to capture.pcap
            command1 = "sudo tcpdump -c " + num_cap + " -i " + nic + " -w PCAP/" + pcap_file
            tcpdump = sp.run([command1], shell=True)

            # Use argus.conf and ra.conf to convert capture.pcap to traffic.binetflow
            command2 = "argus -r PCAP/" + pcap_file + " -w - -F argus.conf | ra -F ra.conf -L0 -c , -n > traffic.binetflow"
            argus_ra = sp.run([command2], shell=True)

            # Read traffic.binetflow to show on terminal
            cat_binetflow = sp.run(["cat traffic.binetflow"], shell=True)

            # Read and write to log file
            f = open('traffic.binetflow', 'r')
            chuoi = f.read()
            ff = open('log.txt', 'a')
            ff.write(chuoi)

            # Read file by pd.read_csv()
            df = pd.read_csv('traffic.binetflow')

            # Edit features
            df = df.drop(['StartTime', 'SrcAddr', 'Sport', 'DstAddr', 'Dport', 'State', 'sTos'], axis=1)
            df_new = edit_data(df)

            # Use model to decet
            y_predict = model_LR.predict(df_new)
            print(y_predict)

            s1 = 0
            for j in y_predict:
                if j == 1:
                    s1 = s1 + 1

            if s1 >= (len(y_predict) - s1):
                print('||=====||')
                print('||>>>>>>>>> BOTNET-DDOS TRAFFIC <<<<<<<<<||')
                print('||=====||\n')
                ff.write('INFORMATION: BOTNET-DDOS TRAFFIC \n\n')
            else:
                print('||=====||')
                print('||>>>>>>>>> BACKGROUND-NORMAL TRAFFIC <<<<<<<<<||')
                print('||=====||\n')
                ff.write('INFORMATION: BACKGROUND-NORMAL TRAFFIC \n\n')

            i = i + 1
            print('=>', i)

if __name__ == "__main__":
    main()
```

Kết luận chương: tóm lại, nhóm chúng tôi đã xây dựng được các mô hình máy học và áp dụng chúng vào trong ứng dụng được tạo ra bằng ngôn ngữ Python3. Các kết quả của ứng dụng cũng như các đánh giá về các mô hình máy học sẽ được trình bày trong chương tiếp theo.



## CHƯƠNG 5

### THỰC NGHIỆM VÀ ỨNG DỤNG

#### 5.1. Môi trường cài đặt

Máy tính xách tay HP

- Bộ xử lý: Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz 2.30 GHz
- RAM: 8.00 GB
- Hệ điều hành: Windows 10 Pro, 64-bit

Các mô hình máy học được đào tạo và đánh giá trên Google Colaboratory “Colab” với:

- Bộ xử lý: Python 3 phần phụ trợ Google Compute Engine
- RAM: 25.46 GB
- Ổ đĩa: 107.72 GB

#### 5.2. Bộ dữ liệu

Như đã trình bày trong chương 1, nhóm chúng tôi đã đề xuất hướng tiếp cận phát hiện tấn công từ chối dịch vụ phân tán (DDoS) thông qua việc phát hiện lưu lượng mạng Botnet. Do đó, chúng tôi sử dụng bộ dữ liệu CTU-13 [2] để đào tạo các mô hình máy học.

##### 5.2.1. Giới thiệu bộ dữ liệu CTU-13

Bộ dữ liệu CTU-13 là bộ dữ liệu lưu lượng mạng botnet đã được thu thập vào năm 2011 tại Đại học CTU ở Cộng hòa Séc. Nó là một bộ dữ liệu được gán nhãn bao gồm các nhãn là lưu lượng botnet, normal và background. Mục tiêu của tập dữ liệu là có được một lượng lớn lưu lượng truy cập botnet thực kết hợp với lưu lượng truy cập thông thường (normal) và lưu lượng truy cập nền (background).

Bộ dữ liệu bao gồm 13 kịch bản trong đó mỗi kịch bản được tạo ra bằng cách thực thi một phần mềm độc hại cụ thể như được hiển thị trong bảng bên dưới. Trong đó có ba kịch bản 4, 10 và 11 là kịch bản triển khai tấn công DDoS.

*Bảng 5. 1: 13 kịch bản của bộ dữ liệu CTU-13*

Id	IRC	SPAM	CF	PS	DDoS	FF	P2P	US	HTTP	Note
1	✓	✓	✓							
2	✓	✓	✓							
3	✓			✓				✓		
4	✓				✓			✓		UDP and ICMP DDoS.
5		✓		✓					✓	Scan web proxies.
6				✓						Proprietary C&C. RDP.
7									✓	Chinese hosts.
8				✓						Proprietary C&C. Net-BIOS, STUN.
9	✓	✓	✓	✓						
10	✓				✓			✓		UDP DDoS.
11	✓				✓			✓		ICMP DDoS.
12							✓			Synchronization.
13		✓		✓					✓	Captcha. Web mail.

Mỗi kịch bản được ghi lại trong một tệp pcap chứa tất cả các gói của ba loại lưu lượng. Các tệp pcap này được xử lý để thu được các loại thông tin khác, chẳng hạn như NetFlows, WebLogs, v.v. Phân tích đầu tiên của bộ dữ liệu CTU-13, đã được mô tả và xuất bản trong bài báo [2] đã sử dụng unidirectional NetFlows (NetFlows một chiều) để biểu diễn lưu lượng và gán nhãn. Nhưng không nên sử dụng các unidirectional NetFlows vì chúng hoạt động khác so với phân tích thứ hai của bộ dữ liệu khi sử dụng bidirectional NetFlows (NetFlows hai chiều). NetFlows hai chiều có một số ưu điểm so với các NetFlows một chiều. Đầu tiên, chúng giải quyết vấn đề phân biệt giữa máy khách và máy chủ, thứ hai, chúng bao gồm nhiều thông tin hơn và thứ ba là chúng bao gồm các nhãn chi tiết hơn nhiều.

Mối quan hệ giữa thời lượng của từng kịch bản, số lượng gói, số lượng NetFlows và kích thước của tệp pcap được hiển thị trong bảng bên dưới. Bảng này cũng cho thấy phần mềm độc hại được sử dụng để tạo bản chụp (capture) và số lượng máy tính bị nhiễm trên mỗi kịch bản.

*Bảng 5. 2: Lượng dữ liệu trên mỗi kịch bản mạng botnet*

Id	Duration(hrs)	# Packets	#NetFlows	Size	Bot	#Bots
1	6.15	71,971,482	2,824,637	52GB	Neris	1
2	4.21	71,851,300	1,808,123	60GB	Neris	1
3	66.85	167,730,395	4,710,639	121GB	Rbot	1
4	4.21	62,089,135	1,121,077	53GB	Rbot	1
5	11.63	4,481,167	129,833	37.6GB	Virut	1
6	2.18	38,764,357	558,920	30GB	Menti	1
7	0.38	7,467,139	114,078	5.8GB	Sogou	1
8	19.5	155,207,799	2,954,231	123GB	Murlo	1
9	5.18	115,415,321	2,753,885	94GB	Neris	10
10	4.75	90,389,782	1,309,792	73GB	Rbot	10
11	0.26	6,337,202	107,252	5.2GB	Rbot	3
12	1.21	13,212,268	325,472	8.3GB	NSIS.ay	3
13	16.36	50,888,256	1,925,150	34GB	Virut	1

Đặc điểm khác biệt của tập dữ liệu CTU-13 là được các tác giả từ bài báo [2] phân tích và gán nhãn cho từng kịch bản theo cách thủ công. Quá trình gán nhãn đã được thực hiện bên trong các tệp NetFlows. Bảng bên dưới cho thấy mối quan hệ giữa số lượng nhãn cho Background, Botnet, Kênh C&C và Normal trên mỗi kịch bản.

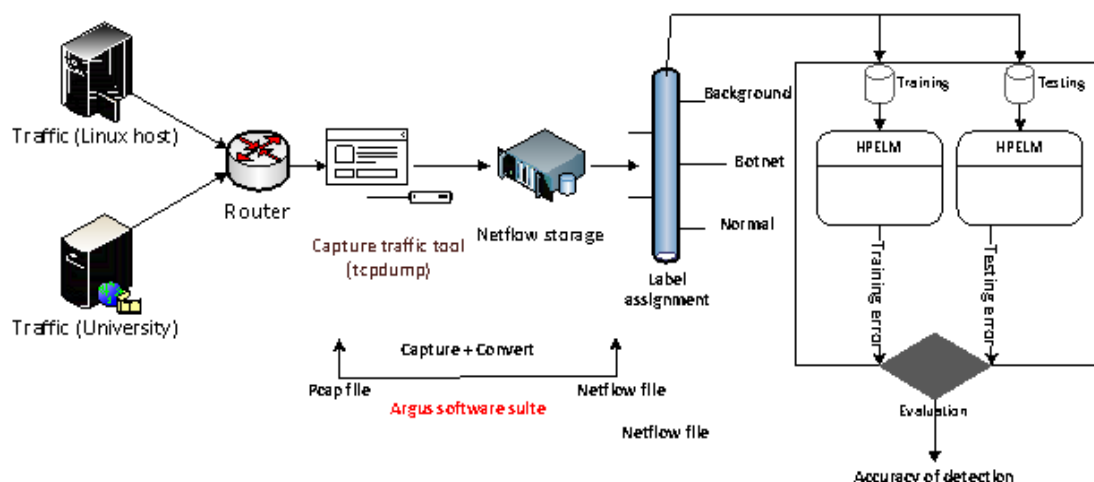
Bảng 5. 3: Phân phối nhãn trong NetFlows cho mỗi kịch bản của bộ dữ liệu

Scen.	Total Flows	Botnet Flows	Normal Flows	C&C Flows	Background Flows
1	2,824,636	39,933(1.41%)	30,387(1.07%)	1,026(0.03%)	2,753,290(97.47%)
2	1,808,122	18,839(1.04%)	9,120(0.5%)	2,102(0.11%)	1,778,061(98.33%)
3	4,710,638	26,759(0.56%)	116,887(2.48%)	63(0.001%)	4,566,929(96.94%)
4	1,121,076	1,719(0.15%)	25,268(2.25%)	49(0.004%)	1,094,040(97.58%)
5	129,832	695(0.53%)	4,679(3.6%)	206(1.15%)	124,252(95.7%)
6	558,919	4,431(0.79%)	7,494(1.34%)	199(0.03%)	546,795(97.83%)
7	114,077	37(0.03%)	1,677(1.47%)	26(0.02%)	112,337(98.47%)
8	2,954,230	5,052(0.17%)	72,822(2.46%)	1,074(2.4%)	2,875,282(97.32%)
9	2,753,884	179,880(6.5%)	43,340(1.57%)	5,099(0.18%)	2,525,565(91.7%)
10	1,309,791	106,315(8.11%)	15,847(1.2%)	37(0.002%)	1,187,592(90.67%)
11	107,251	8,161(7.6%)	2,718(2.53%)	3(0.002%)	96,369(89.85%)
12	325,471	2,143(0.65%)	7,628(2.34%)	25(0.007%)	315,675(96.99%)
13	1,925,149	38,791(2.01%)	31,939(1.65%)	1,202(0.06%)	1,853,217(96.26%)

Từ bảng trên chúng ta có thể thấy được một sự chênh lệch rất lớn giữa các luồng Background so với các luồng còn lại. Điều này gây ra sự mất cân lớn trên toàn bộ tập dữ liệu nên cần phải xử lý thật tốt khi đưa vào các mô hình máy học.

### 5.2.2. Quy trình tạo ra bộ dữ liệu CTU-13

Bộ dữ liệu CTU-13 cơ bản đã được tạo ra từ mô hình mạng bên dưới.



Hình 5. 1: Mô hình thu thập dữ liệu cho bộ dữ liệu CTU-13

Đầu tiên, lưu lượng mạng từ bên ngoài sẽ đi qua router và bị bắt lại với công cụ Tcpdump và tạo thành tệp pcap. Sau đó, các tệp pcap được chuyển đổi sang tiêu chuẩn tệp NetFlow bằng cách sử dụng bộ công cụ Argus trong hai bước. Bước đầu tiên sẽ chuyển đổi các tệp pcap thành bộ lưu trữ nhị phân Argus hai chiều thông qua tệp cấu hình argus.conf. Bước thứ hai là chuyển đổi bộ lưu trữ nhị phân Argus hai chiều thành tệp NetFlow thông qua tệp cấu hình ra.conf. Kết quả của các bước này là tệp NetFlow đầu ra cuối cùng.

Tiếp theo là giai đoạn gán nhãn cho dữ liệu NetFlow theo thứ tự ưu tiên như sau:

- Gán nhãn Background cho toàn bộ lưu lượng truy cập.
- Gán nhãn Normal cho lưu lượng phù hợp với các bộ lọc nhất định.
- Gán nhãn Botnet cho tất cả lưu lượng truy cập đến từ hoặc đến bất kỳ địa chỉ IP nào đã biết bị nhiễm.

Cuối cùng, tệp NetFlow sẽ bao gồm các trường sau: StartTime, Dur, Proto, SrcAddr, Sport, Dir, DstAddr, Dport, State, sTos, dTos, TotPkts, TotBytes, SrcBytes [2]. Hình bên dưới minh họa cho một tệp NetFlow được chuyển đổi từ tệp pcap.

```
StartTime,Dur,Proto,SrcAddr,Sport,Dir,DstAddr,Dport,State,sTos,dTos,TotPkts,TotBytes,SrcBytes,Label
2011/08/15 10:42:52.613616,1065.731934,udp,188.114.23.248,14268, <->,147.32.84.118,1150,CON,0,0,2,252,145,flow=Background-UDP-Established
2011/08/15 10:42:52.676517,1471.787109,udp,2.224.28.134,48279, <->,147.32.84.118,1150,CON,0,0,2,252,145,flow=Background-UDP-Established
2011/08/15 10:43:34.544316,1095.947266,udp,41.145.83.148,40812, <->,147.32.84.118,1150,CON,0,0,2,288,145,flow=Background-UDP-Established
2011/08/15 10:58:26.370803,2896.377197,udp,217.42.33.104,55066, <->,147.32.84.118,1150,CON,0,0,3,433,290,flow=Background-UDP-Established
2011/08/15 10:58:27.208147,1384.708008,udp,41.107.0.220,11066, <->,147.32.84.118,1150,CON,0,0,2,288,145,flow=Background-UDP-Established
2011/08/15 10:59:02.040572,1071.322388,udp,213.44.235.37,9200, <->,147.32.84.118,1150,CON,0,0,2,288,145,flow=Background-UDP-Established
2011/08/15 11:00:05.704408,3582.270264,tcp,147.32.84.134,45266, <->,205.188.10.203,443,PA_PA,0,0,662,70674,25955,flow=From-Normal-V45-Jist
2011/08/15 11:00:05.704760,44.728664,tcp,62.44.1.18,80, <->,147.32.84.118,51463,PA_FRA,0,0,768,756975,741555,flow=Background
2011/08/15 11:00:05.707670,0.000000,udp,147.32.80.9,53, ->,147.32.85.103,18006,INT,0,,1,175,175,flow=From-Normal-V45-UDP-CVUT-DNS-Server
2011/08/15 11:00:05.709591,0.003098,udp,147.32.85.103,41094, <->,147.32.80.9,53,CON,0,0,2,223,71,flow=To-Background-UDP-CVUT-DNS-Server
2011/08/15 11:00:05.715153,0.000609,udp,147.32.85.103,27423, <->,147.32.80.9,53,CON,0,0,2,212,73,flow=To-Background-UDP-CVUT-DNS-Server
2011/08/15 11:00:05.716016,35.265488,tcp,212.194.183.145,49584, <->,147.32.85.56,44076,FPA_FPA,0,0,43,3147,2061,flow=Background
```

Hình 5. 2: Ví dụ tệp NetFlow

Tiếp theo, chúng ta sẽ đi khảo sát các giá trị tính năng trích xuất từ tệp NetFlow trên.

### 5.2.3. Các cột/ tính năng của bộ dữ liệu

Các kịch bản trong bộ dữ liệu CTU-13 có tới 15 cột/ tính năng được trình bày trong bảng dưới.

Bảng 5. 4: Mô tả 15 cột/ tính năng của bộ dữ liệu

Cột/ tính năng	Mô tả
StartTime	Nó đại diện cho thời gian bắt đầu của cuộc tấn công và mỗi bản ghi có một dấu thời gian khác nhau ở định dạng sau là 2011/08/10 12:30:13.516854
Dur	Nó đại diện cho thời gian của cuộc tấn công và được chỉ định bằng giây
Proto	Nó bao gồm tổng cộng 15 giao thức cụ thể là ‘tcp’, ‘udp’, ‘rtp’, ‘pim’, ‘icmp’, ‘arp’, ‘ipx/spx’, ‘rtcp’, ‘igmp’, ‘ipv6- icmp’, ‘ipv6’, ‘udt’, ‘esp’, ‘unas’, ‘rarp’
SrcAddr	Địa chỉ IP nguồn

Sport	Port nguồn
Dir	Hướng của traffic được biểu thị dưới dạng ‘->’, ‘?>’, ‘<->’, ‘<?>’, ‘who’, ‘<’, ‘<?’
DstAddr	Địa chỉ IP đích
Dport	Port đích
State	Trạng thái của giao dịch theo giao thức và có tổng cộng 231 giá trị duy nhất khác nhau.
sTos	Trường source type of service có các giá trị là 0, 1, 2, 3, 192, NaN
dTos	Trường destination type of service có các giá trị là 0, 1, 2, 3, NaN
TotPkts	Tổng số gói tin có giá trị trong khoảng từ 1 đến 2686731
TotBytes	Tổng số bytes có giá trị trong khoảng từ 60 đến 2689640464
SrcBytes	Tổng số bytes bắt nguồn từ source có giá trị trong khoảng từ 0 đến 2635366235
Label	Ba nhãn lưu lượng truy cập gồm Background, Botnet và Normal

#### 5.2.4. Tổng quát hoá bộ dữ liệu

Bộ dữ liệu CTU-13 bao gồm 13 kịch bản sẽ chứa 19976683 mẫu dữ liệu và 15 cột/ tính năng bao gồm cả cột mục tiêu với nhãn thô.

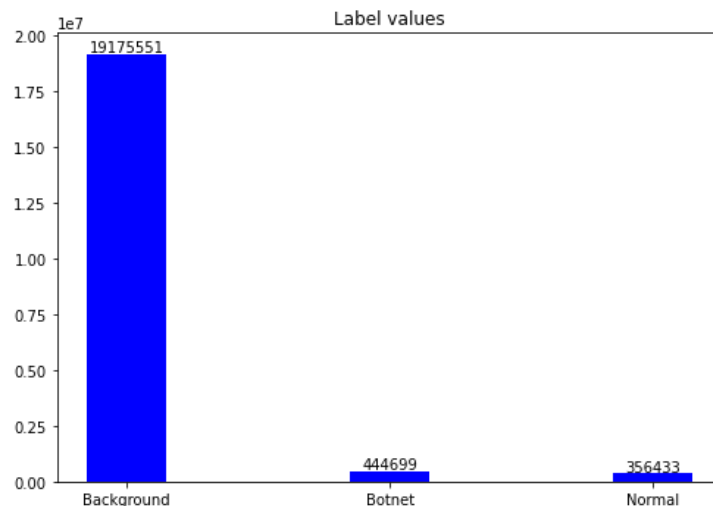
	StartTime	Dur	Proto	SrcAddr	Sport	Dir	DstAddr	Dport	State	sTos	dTos	TotPkts	TotBytes	SrcBytes	Label
0	2011/08/10 09:46:59.607825	1.026539	tcp	94.44.127.113	1577	->	147.32.84.59	6881	S_RA	0.0	0.0	4	276	156	flow=Background-Established-cmpgw-CVUT
1	2011/08/10 09:47:00.634364	1.009595	tcp	94.44.127.113	1577	->	147.32.84.59	6881	S_RA	0.0	0.0	4	276	156	flow=Background-Established-cmpgw-CVUT
2	2011/08/10 09:47:48.185538	3.056586	tcp	147.32.86.89	4768	->	77.75.73.33	80	SR_A	0.0	0.0	3	182	122	flow=Background-TCP-Attempt
3	2011/08/10 09:47:48.230897	3.111769	tcp	147.32.86.89	4788	->	77.75.73.33	80	SR_A	0.0	0.0	3	182	122	flow=Background-TCP-Attempt
4	2011/08/10 09:47:48.963351	3.083411	tcp	147.32.86.89	4850	->	77.75.73.33	80	SR_A	0.0	0.0	3	182	122	flow=Background-TCP-Attempt
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
19976695	2011/08/16 09:36:00.710103	0.000000	udp	147.32.84.229	13363	->	125.14.162.10	27460	INT	0.0	NaN	1	476	476	flow=Background-UDP-Attempt
19976696	2011/08/16 09:36:00.777160	0.000427	udp	213.146.167.54	20856	<->	147.32.84.229	13363	CON	0.0	0.0	2	135	75	flow=Background-UDP-Established
19976697	2011/08/16 09:36:00.784094	0.000000	udp	147.32.84.229	13363	->	122.121.0.92	12923	INT	0.0	NaN	1	476	476	flow=Background-UDP-Attempt
19976698	2011/08/16 09:36:00.784160	0.000000	udp	147.32.84.229	13363	->	110.233.175.133	37690	INT	0.0	NaN	1	476	476	flow=Background-UDP-Attempt
19976699	2011/08/16 09:36:00.806547	0.000008	tcp	147.32.87.51	12489	->	147.32.86.96	49773	SA_	0.0	NaN	2	140	140	flow=Background-TCP-Established

19976683 rows x 15 columns

Hình 5. 3: Bộ dữ liệu CTU-13 trong khi đọc trong DataFrame

Trong đó, số lượng mẫu trong 3 lớp Background, Botnet và Normal có sự chênh lệch lớn được thể hiện trong hình bên dưới.





Hình 5. 4: Số lượng mẫu trong mỗi 3 lớp

Thống kê mô tả cho các cột/ tính năng về count (số lượng), mean (giá trị trung bình), std (độ lệch chuẩn), giá trị nhỏ nhất (min), giá trị lớn nhất (max),...

	Dur	sTos	dTos	TotPkts	TotBytes	SrcBytes
count	1.997668e+07	1.975616e+07	1.825869e+07	1.997668e+07	1.997668e+07	1.997668e+07
mean	2.879466e+02	8.158967e-02	4.074773e-04	4.139078e+01	3.232714e+04	6.435327e+03
std	8.318068e+02	3.910226e+00	3.245888e-02	5.545727e+03	3.983039e+06	1.667901e+06
min	0.000000e+00	0.000000e+00	0.000000e+00	1.000000e+00	6.000000e+01	0.000000e+00
25%	2.750000e-04	0.000000e+00	0.000000e+00	2.000000e+00	2.140000e+02	7.800000e+01
50%	7.270000e-04	0.000000e+00	0.000000e+00	2.000000e+00	2.640000e+02	8.300000e+01
75%	1.966016e+00	0.000000e+00	0.000000e+00	4.000000e+00	6.190000e+02	3.040000e+02
max	3.657061e+03	1.920000e+02	3.000000e+00	1.658064e+07	4.376239e+09	3.423408e+09

Hình 5. 5: Thống kê cho các cột/ tính năng

Thống kê thông tin về dtype của từng cột/ tính năng và mức sử dụng bộ nhớ. Trong đó có 3 cột/ tính năng là kiểu float64, 3 cột/ tính năng là kiểu int64 và 9 cột/ tính năng là kiểu object.

```

Int64Index: 19976683 entries, 0 to 19976699
Data columns (total 15 columns):
#   Column      Dtype
---  ---
0   StartTime   object
1   Dur         float64
2   Proto       object
3   SrcAddr     object
4   Sport       object
5   Dir         object
6   DstAddr     object
7   Dport       object
8   State       object
9   sTos        float64
10  dTos        float64
11  TotPkts     int64
12  TotBytes    int64
13  SrcBytes    int64
14  Label       object
dtypes: float64(3), int64(3), object(9)
memory usage: 2.4+ GB
    
```

Hình 5. 6: Thống kê cho từng cột/ tính năng

### 5.3. Quá trình xử lý bộ dữ liệu

Bộ dữ liệu CTU-13 bao gồm 13 kịch bản với số lượng khoảng 20 triệu mẫu dữ liệu cho thấy được quy mô rất lớn của bộ dữ liệu này. Bên cạnh đó, bộ dữ liệu còn có hiện tượng mất cân bằng các lớp lưu lượng trong cột/ tính năng Label, thiếu sót các giá trị hay tồn tại các giá trị với nhiều kiểu dữ liệu khác nhau và các giá trị chênh lệch nhau về khoảng giá trị. Điều này đòi hỏi cần phải có một quá trình xử lý dữ liệu thật tốt để mang lại hiệu quả cho các mô hình học máy.

#### 5.3.1. Quá trình tiền xử lý

##### 5.3.1.1. Loại bỏ các cột/ tính năng không cần thiết

Chúng tôi sẽ loại bỏ các cột/ tính năng không cần thiết như 'StartTime', 'SrcAddr', 'Sport', 'DstAddr', 'Dport', 'State'. Lý do, loại bỏ 'StartTime' vì đây chỉ là giá trị ngày giờ mang lại thông tin mô tả cho từng mẫu dữ liệu chứ không mang lại nhiều giá trị cho các mô hình máy học. Bên cạnh đó việc loại bỏ 5 cột/ tính năng 'SrcAddr', 'Sport', 'DstAddr', 'Dport', 'State' cũng giúp giảm đi số lượng dữ liệu và giảm thời gian cần đạo tạo mô hình máy học.

##### 5.3.1.2. Xử lý các giá trị rỗng

Sau khi loại bỏ các tính năng không cần thiết thì chúng tôi xử lý tiếp các giá trị rỗng. Các giá trị rỗng tồn tại trong 2 cột/ tính năng là 'sTos' và 'dTos'.

Cách đơn giản để giải quyết vấn đề này là xóa đi 2 cột/ tính năng 'sTos' và 'dTos'. Nhưng điều này có thể làm giảm hiệu suất của mô hình máy học. Vì thế chúng tôi chọn giải pháp thay thế các giá trị rỗng bằng giá trị khác, cụ thể chúng tôi sẽ thay thế các giá trị rỗng bằng một số nguyên là 9 để tránh bị trùng với các giá trị của 'sTos' và 'dTos'.

##### 5.3.1.3. Xử lý dữ liệu nhãn thô (raw label)

Cột/ tính năng 'Label' (cột lớp mục tiêu) trong bộ dữ liệu CTU-13 chứa các nhãn được gán dưới dạng dữ liệu thô mô tả đặc tính của từng luồng tương ứng như ví dụ trong hình bên dưới.

```
0      flow=Background-Established-cmpgw-CVUT
1      flow=Background-Established-cmpgw-CVUT
2      flow=Background-TCP-Attempt
3      flow=Background-TCP-Attempt
4      flow=Background-TCP-Attempt
...
19976695      flow=Background-UDP-Attempt
19976696      flow=Background-UDP-Established
19976697      flow=Background-UDP-Attempt
19976698      flow=Background-UDP-Attempt
19976699      flow=Background-TCP-Established
Name: Label, Length: 19976683, dtype: object
```

Hình 5. 7: Cột/ tính năng 'Label' với dữ liệu thô

Để các mô hình máy học có thể sử dụng được cột/ tính năng này thì chúng ta phải chuyển đổi các giá trị thô trên thành các giá trị số nguyên. Các giá trị này đại diện cho các lớp mà mô hình máy học sẽ phân loại ra dựa trên các cột/ tính năng còn lại. Chúng tôi sẽ gán 3 giá trị là 0, 1, 2 cho các nhãn thô tương ứng. Trong đó, các giá trị thuộc về lưu lượng Background sẽ là 0, lưu lượng Botnet sẽ là 1 và lưu lượng Normal sẽ là 2.

```
0      0
1      0
2      0
3      0
4      0
..
19976695  0
19976696  0
19976697  0
19976698  0
19976699  0
Name: Label, Length: 19976683, dtype: int64
```

Hình 5. 8: Cột/ tính năng 'Label' sau khi gán lại nhãn

Lưu ý: nhãn của các luồng do phần mềm độc hại tạo ra sẽ bắt đầu bằng "From-Botnet". Các nhãn "To-Botnet" là các luồng được gửi đến botnet bởi các máy tính không xác định, vì vậy chúng không được xem là lưu lượng độc hại.

#### 5.3.1.4. Xử lý các biến/ giá trị phân loại

Bước tiếp theo của quá trình tiền xử lý là xử lý các biến/ giá trị phân loại. Các biến/ giá trị phân loại này tồn tại trong 4 cột/ tính năng bao gồm 'Dir', 'Proto', 'sTos' và 'dTos'.

Chúng tôi sẽ sử dụng 2 kỹ thuật đã trình bày trong chương 3 phần 3.3.2 để thực hiện bước này. Lưu ý: việc chúng tôi sử dụng 2 kỹ thuật là do chúng tôi sẽ chia ra thành 2 trường hợp để chuẩn bị cho các kịch bản sẽ trình bày trong phần sau.

##### ***Trường hợp 1: Sử dụng kỹ thuật mã hoá với giá trị số***

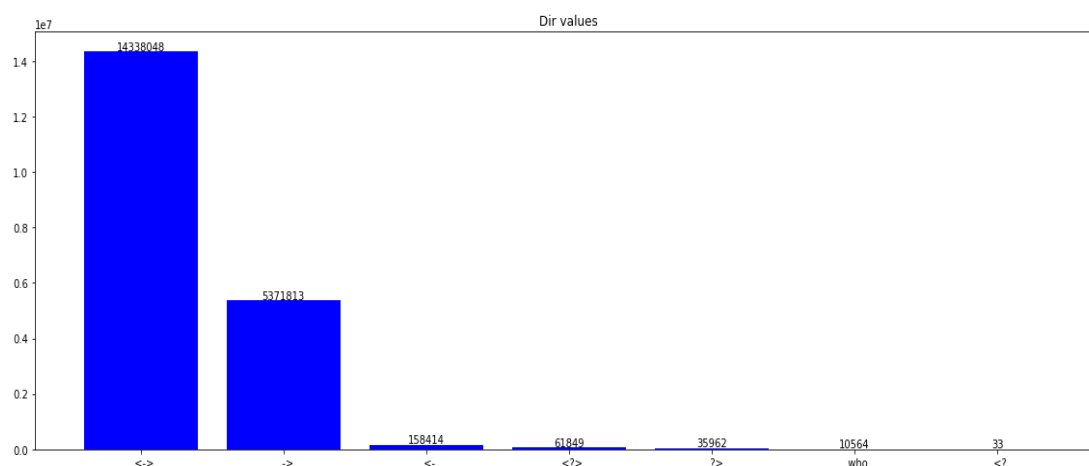
Hai cột/ tính năng 'sTos' và 'dTos' cơ bản đã chứa các giá trị số (xem như các biến phân loại) nên sẽ bỏ qua. Với cột/ tính năng 'Dir' có chứa tổng cộng 7 biến giá trị phân loại bao gồm '<->', '<?>', '<->', '<?>', 'who', '<', '<?'. Trong đó giá trị '<->' chiếm khoảng 71.77% và giá trị '<->' chiếm khoảng 26.89% so với các giá trị còn lại. Do đó, chúng tôi sẽ mã hoá riêng cho giá trị '<->' bằng 1 và giá trị '<->' bằng 2, còn tất cả các giá trị còn lại sẽ được mã hoá bằng 0 do tỉ lệ phần trăm thấp so với tổng số 2 giá trị kia. Tương tự với cột/ tính năng 'Proto' có chứa tổng cộng 15 biến giá trị phân loại bao gồm 'tcp', 'udp', 'rtcp', 'pim', 'icmp', 'arp', 'ipx/spx', 'rtcp', 'igmp', 'ipv6- icmp', 'ipv6', 'udt',



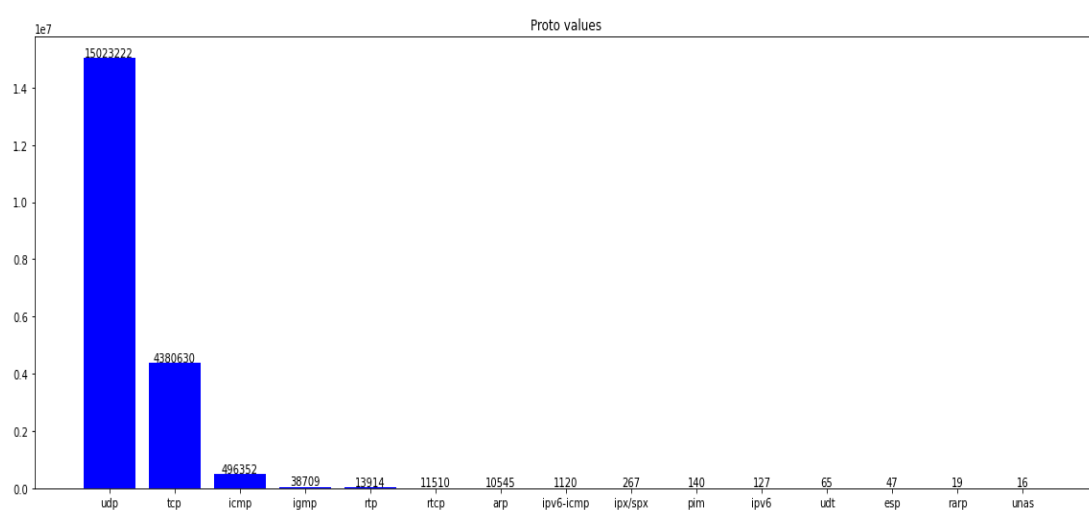
‘esp’, ‘unas’, ‘rarp’. Trong đó giá trị ‘udp’ chiếm khoảng 75.20%, giá trị ‘tcp’ chiếm khoảng 21.92% và giá trị ‘icmp’ chiếm khoảng 2.48% so với các giá trị còn lại. Lúc này chúng tôi cũng thực hiện giống với cột/ tính năng ‘Dir’. Đó là mã hoá riêng cho giá trị ‘udp’ bằng 17, giá trị ‘tcp’ bằng 6, giá trị ‘icmp’ bằng 1 và tất cả các giá trị còn lại sẽ được mã hoá bằng 0.

### ***Trường hợp 2: Sử dụng kỹ thuật mã hoá One-hot Encoding***

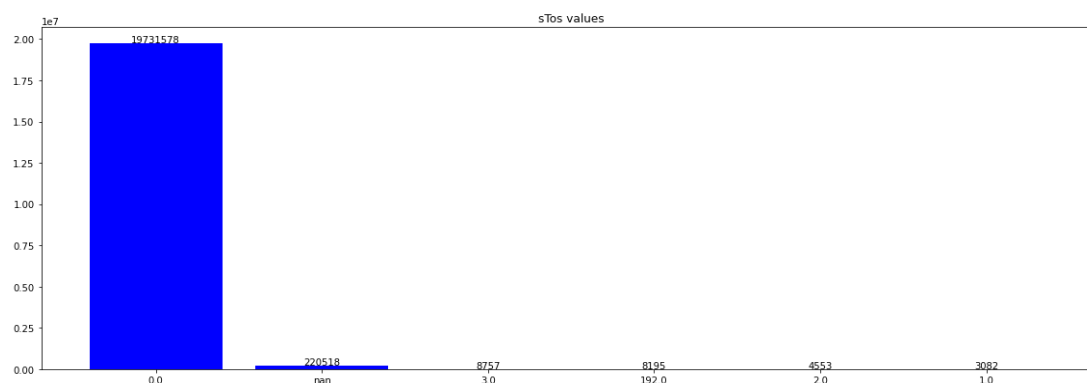
Với trường hợp này chúng tôi sử dụng kỹ thuật mã hoá One-hot Encoding đã trình bày trong chương 3 phần 3.3.2. Vẫn là 4 cột/ tính năng ‘Dir’, ‘Proto’, ‘sTos’ và ‘dTos’. Trong đó ‘Dir’ bao gồm 7 biến/ giá trị phân loại, ‘Proto’ bao gồm 15 biến/ giá trị phân loại, ‘sTos’ bao gồm 6 biến/ giá trị phân loại và cuối cùng là ‘dTos’ bao gồm 5 biến/ giá trị phân loại. Nếu ta mã hoá hết 4 cột/ tính năng này bằng One-hot Encoding thì sẽ tạo ra 33 cột/ tính năng mới chứa 2 giá trị phân loại 0, 1 trên bộ dữ liệu.



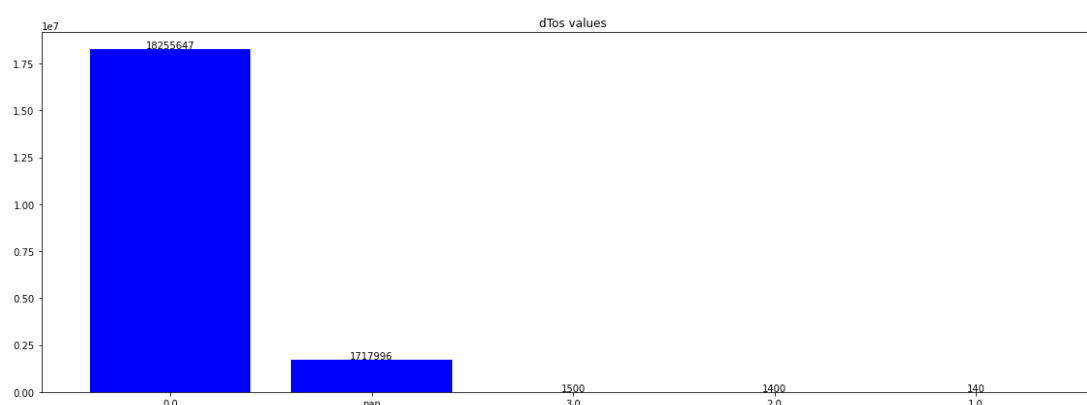
***Hình 5. 9: Số lượng giá trị mỗi loại trong cột/ tính năng ‘Dir’***



***Hình 5. 10: Số lượng giá trị mỗi loại trong cột/ tính năng ‘Proto’***



Hình 5. 11: Số lượng giá trị mỗi loại trong cột/ tính năng 'sTos'



Hình 5. 12: Số lượng giá trị mỗi loại trong cột/ tính năng 'dTos'

Việc tạo ra 33 cột/ tính năng mới sẽ phát sinh vấn đề đa chiều trên bộ dữ liệu đó, đồng thời với số lượng dữ liệu nhiều như vậy sẽ làm tăng thời gian đào tạo trên các mô hình máy học và tiêu tốn thời gian khi mô hình đưa ra các dự đoán về dữ liệu mới. Vấn đề này đã được tác giả của bài báo [12] giải quyết bằng cách mã hoá One-hot Encoding cho cột/ tính năng 'Proto' thành 4 cột 'Proto\_udp', 'Proto\_tcp', 'Proto\_icmp' và 'Proto\_others', cột/ tính năng 'Dir' thành 3 cột 'Dir\_<->', 'Dir\_->' và 'Dir\_others', cột/ tính năng 'sTos' thành 2 cột 'sTos\_0.0' và 'sTos\_others', cột/ tính năng 'dTos' đã bị tác giả loại bỏ do có ngưỡng phương sai thấp. Trong đó, các cột 'Proto\_others', 'Dir\_others' và 'sTos\_others' là các cột được tạo ra từ các biến/ giá trị phân loại với số lượng rất thấp so với các biến/ giá trị phân loại khác trong mỗi 3 cột 'Proto', 'Dir', 'sTos'. Không chỉ vậy, để giảm thêm các cột/ tính năng, bài báo [12] còn bỏ đi bớt một cột từ các cột đã được mã hoá, ví dụ như cột 'Dir' mã hoá One-hot Encoding thành 3 cột 'Dir\_<->', 'Dir\_->', 'Dir\_others' nhưng chỉ giữ lại 2 cột 'Dir\_<->', 'Dir\_->' và cũng xử lý tương tự với 2 cột 'Proto' và 'Dir'.

Nhóm chúng tôi cũng sẽ thực hiện theo cách này nhưng có thay đổi khác một chút. Cụ thể là chúng tôi vẫn sử dụng mã hoá One-hot Encoding cho cột

‘Proto’ thành 4 cột ‘Proto\_udp’, ‘Proto\_tcp’, ‘Proto\_icmp’ và ‘Proto\_others’, cột ‘Dir’ thành 3 cột ‘Dir\_ <->’, ‘Dir\_ ->’ và ‘Dir\_others’. theo như bài báo. Vấn đề mã hoá như vậy là do sự chênh lệch các giá trị trong 2 cột trên nên những cột có giá trị với số lượng lớn hơn các giá trị khác thì sẽ được mã hoá One-hot thành 1 cột riêng, còn những giá trị với số lượng nhỏ còn lại sẽ được gom lại để mã hoá thành cột “others”. Còn 2 cột/ tính năng ‘sTos’ và ‘dTos’ thì với ‘sTos’ mã hoá thành 3 cột ‘sTos\_0.0’, ‘sTos\_9.0’, ‘sTos\_others’, tương tự với ‘dTos’ sẽ được mã hoá thành 3 cột ‘dTos\_0.0’, ‘dTos\_9.0’, ‘dTos\_others’. Trong đó, 2 cột ‘sTos\_9.0’ và ‘dTos\_9.0’ là 2 cột được mã hoá từ giá trị rỗng đã được thay thế bằng số nguyên là 9 trong bước xử lý các giá trị rỗng. Khác biệt ở đây là bài báo [12], tác giả đã chuyển các giá trị rỗng thành các giá trị bằng 0 trùng với giá trị đã tồn tại trong cột ‘sTos’ và ‘dTos’, điều này vô tình chỉ ra rằng các mẫu có giá trị ‘sTos’ là rỗng hay ‘dTos’ là rỗng cũng sẽ “giống” với các mẫu có giá trị ‘sTos’ là 0 hay ‘dTos’ là 0. Còn chúng tôi thì mã hoá One-hot Encoding cho các giá trị rỗng bằng cách chuyển chúng sang 1 giá trị mới khác các giá trị đã tồn tại sau đó mới thực hiện mã hoá One-hot Encoding.

	Dur	TotPkts	TotBytes	SrcBytes	Label	Dir_others	Dir_<->	Dir_->	Proto_others	Proto_icmp	Proto_tcp	Proto_udp	sTos_others	sTos_0.0	sTos_9.0	dTos_others	dTos_0.0	dTos_9.0
0	1.026539	4	276	156	0	0	1	0	0	0	1	0	0	1	0	0	1	0
1	1.009595	4	276	156	0	0	1	0	0	0	1	0	0	1	0	0	1	0
2	3.056586	3	182	122	0	0	1	0	0	0	1	0	0	1	0	0	1	0
3	3.111769	3	182	122	0	0	1	0	0	0	1	0	0	1	0	0	1	0
4	3.083411	3	182	122	0	0	1	0	0	0	1	0	0	1	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
19976695	0.000000	1	476	476	0	0	1	0	0	0	0	1	0	1	0	0	0	1
19976696	0.000427	2	135	75	0	0	0	1	0	0	0	1	0	1	0	0	1	0
19976697	0.000000	1	476	476	0	0	1	0	0	0	0	1	0	1	0	0	0	1
19976698	0.000000	1	476	476	0	0	1	0	0	0	0	1	0	1	0	0	0	1
19976699	0.000008	2	140	140	0	0	1	0	0	0	1	0	0	1	0	0	0	1

19976683 rows × 18 columns

Hình 5. 13: Bộ dữ liệu sau khi mã hoá One-hot Encoding

### 5.3.1.5. Sử dụng kỹ thuật Feature Selection

Chúng tôi chỉ sử dụng kỹ thuật này trên bộ dữ liệu trong trường hợp có sử dụng mã hoá One-hot Encoding vì nó sẽ giúp giảm đi số lượng các cột/ tính năng cho bộ dữ liệu. Bên cạnh đó, chúng tôi cũng chia ra làm 2 trường hợp sử dụng phương pháp thống kê Chi-Squared và ANOVA thông qua lớp SelectKBest từ thư viện sklearn.feature\_selection.

#### Trường hợp 1: Phương pháp Chi-Squared

Bộ dữ liệu sau khi được mã hoá One-hot Encoding sẽ bao gồm 17 cột/ tính năng sau: ‘Dur’, ‘TotPkts’, ‘TotBytes’, ‘SrcBytes’, ‘Dir\_ <->’, ‘Dir\_ ->’, ‘Dir\_others’, ‘Proto\_udp’, ‘Proto\_tcp’, ‘Proto\_icmp’, ‘Proto\_others’, ‘sTos\_0.0’, ‘sTos\_9.0’, ‘sTos\_others’, ‘dTos\_0.0’, ‘dTos\_9.0’, ‘dTos\_others’.

Sau bước cân bằng tập dữ liệu thì chúng tôi sẽ sử dụng lớp SelectKBest với tham số chi2 đại diện cho phương pháp Chi-Squared và chọn ra 10 cột/ tính năng tốt nhất với tham số k=10. Cuối cùng, 10 cột tính năng đã được chọn là 'Dur', 'TotPkts', 'TotBytes', 'SrcBytes', 'Dir\_->', 'Dir\_-<->', 'Proto\_icmp', 'Proto\_tcp', 'Proto\_udp', 'dTos\_9.0'.

### ***Trường hợp 2: Phương pháp ANOVA***

Tương tự với trường hợp phương pháp Chi-Squared, chúng tôi cũng sẽ dùng lại lớp SelectKBest với tham số f\_classif đại diện cho phương pháp ANOVA và chọn ra 10 cột/ tính năng tốt nhất với tham số k=10. Cuối cùng, 10 cột tính năng đã được chọn là 'Dur', 'Dir\_others', 'Dir\_->', 'Dir\_-<->', 'Proto\_icmp', 'Proto\_tcp', 'Proto\_udp', 'sTos\_0.0', 'dTos\_0.0', 'dTos\_9.0'.

#### **5.3.1.6. Xử lý sự mất cân bằng trên bộ dữ liệu**

Như đã trình bày, bộ dữ liệu CTU-13 xảy ra vấn đề mất cân bằng rất lớn trên lưu lượng của nhãn lớp Background so với phần còn lại. Sau khi trải qua bước xử lý dữ liệu nhãn thô để gán lại nhãn cho bộ dữ liệu, lúc này ta quan sát thấy được số lượng mẫu Background là 19175551 (95.98%), mẫu Botnet là 444699 (2.22%) và mẫu Normal là 356433 (1.78%). Sự chênh lệch là rất nhiều, vì vậy chúng tôi cần thực hiện các kỹ thuật khác nhau để cân bằng bộ dữ liệu này.

### ***Trường hợp 1: Sử dụng kỹ thuật RandomUnderSampler***

Lớp RandomUnderSampler từ thư viện imblearn.under\_sampling cho phép ta lấy mẫu ngẫu nhiên từ các lớp đa số hay lớp thiểu số theo một chiến lược tùy chọn bất kỳ. Ở đây chúng tôi, thực hiện việc lấy mẫu theo chiến lược sau: vẫn giữ nguyên số lượng mẫu của 2 lớp thiểu số Botnet là 444699 mẫu và Normal là 356433 mẫu, còn với lớp đa số là Background thì giảm số lượng mẫu từ 19175551 xuống còn 801132 mẫu bằng với tổng số lớp Botnet và lớp Normal.

### ***Trường hợp 2: Sử dụng kỹ thuật NearMiss-1***

Kỹ thuật NearMiss lấy mẫu dựa theo thuật toán nearest neighbors và có 3 phiên bản riêng. Ở đây chúng tôi sử dụng kỹ thuật NearMiss-1 để lấy mẫu với số lượng mẫu như trường hợp sử dụng kỹ thuật RandomUnderSampler.

#### **5.3.1.7. Sử dụng kỹ thuật Feature Scaling**

Bước cuối cùng của quá trình tiền xử lý dữ liệu là chia tỷ lệ (scaling) lại bộ dữ liệu để nó hoạt động tốt hơn. Với kỹ thuật này thì chúng tôi sẽ chỉ áp dụng cho 2 mô hình máy học là sử dụng thuật toán K-Nearest Neighbors và Logistic Regression.

### ***Trường hợp 1: Sử dụng kỹ thuật Normalization***

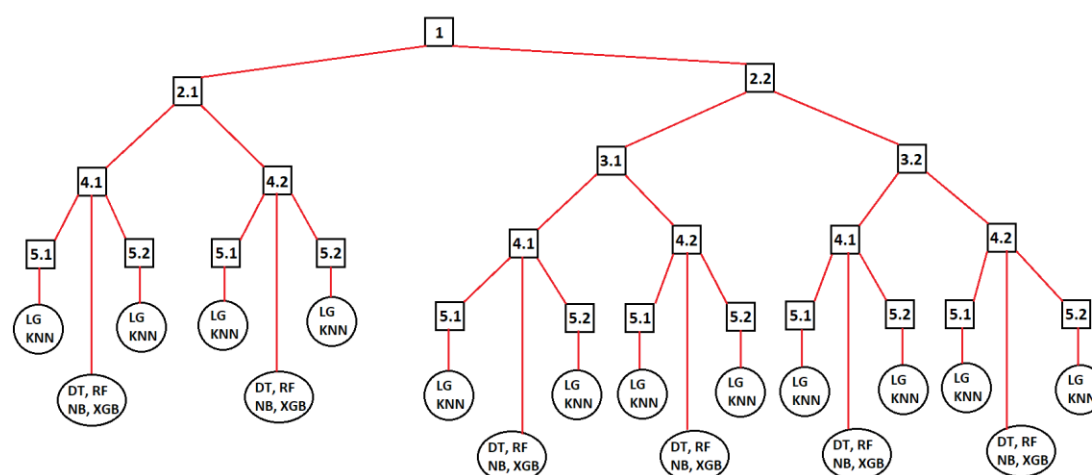
Với kỹ thuật này chúng tôi sử dụng lớp MinMaxScaler từ thư viện sklearn.preprocessing. Nó sẽ tìm ra giá trị nhỏ nhất và lớn nhất của từng cột dữ liệu sau đó dùng 2 giá trị này để chia tỷ lệ cho từng giá trị trên cột dữ liệu về khoảng (0, 1).

### ***Trường hợp 2: Sử dụng kỹ thuật Standardization***

Tương tự với kỹ thuật trên, kỹ thuật Standardization cũng sử dụng lớp StandardScaler thông qua thư viện sklearn.preprocessing. Điều này sẽ thay đổi hình dạng phân bố của dữ liệu giúp mạng lại hiệu quả cho 2 thuật toán Logistic Regression và K-Nearest Neighbors.

### **5.3.2. Tạo các kịch bản đào tạo mô hình máy học**

Chúng tôi sẽ tạo các kịch bản để áp dụng vào các mô hình máy học và đưa ra đánh giá cho chúng. Cụ thể, các kịch bản sẽ sử dụng các phương pháp và kỹ thuật đã trình bày ở trên để thực hiện. Sơ đồ của các kịch bản được thể hiện trong hình bên dưới.



*Hình 5. 14: Sơ đồ kịch bản*

Bằng cách kết hợp các kỹ thuật trong quá trình tiền xử lý, chúng tôi chia ra được 18 kịch bản như trên. Trong đó:

- (1): là mô hình baseline, sau khi đã xoá các cột/ tính năng không cần thiết, mã hoá lại nhãn cho cột 'Label' và xử lý các giá trị rỗng.
- (2.1): Mã hoá các biến/ giá trị phân loại với giá trị số.
- (2.2): Mã hoá các biến/ giá trị phân loại với One-hot Encoding.
- (3.1): Lựa chọn cột/ tính năng với Chi-Squared.
- (3.2): Lựa chọn cột/ tính năng với ANOVA.
- (4.1): Cân bằng bộ dữ liệu với RandomUnderSampler.
- (4.2): Cân bằng bộ dữ liệu với NearMiss-1.

- (5.1): Scaling cột/ tính năng với Normalization.
- (5.2): Scaling cột/ tính năng với Standardization.
- DT là Decision Tree, RF là Random Forest, NB là Naive Bayes, XGB là XGBoost, LG là Logistic Regression và KNN là K-Nearest Neighbors.

#### 5.4. Phương thức đánh giá

Chúng tôi sẽ sử dụng các metric phổ biến trong Machine Learning để đánh giá các mô hình máy học. Các metric được sử dụng bao gồm Accuracy, Recall, Precision, F1-score. Ngoài ra chúng tôi còn sử dụng thêm Confusion Matrix và ROC-AUC để đánh giá.

Accuracy là độ chính xác của các trường hợp dự đoán đúng so với tất cả các trường hợp dự đoán. Công thức là:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Recall là tỉ lệ số dự đoán True Positive của một lớp trong số những sự đoán thực sự là Positive cho riêng lớp đó. Công thức là:

$$Recall = \frac{TP}{TP + FN}$$

Precision là tỉ lệ số dự đoán True Positive của một lớp trong số những dự đoán được phân loại là Positive cho riêng lớp đó. Công thức là:

$$Precision = \frac{TP}{TP + FP}$$

F1-score là sự kết hợp của Recall và Precision để đưa ra đánh giá chung cho mô hình máy học. Công thức là:

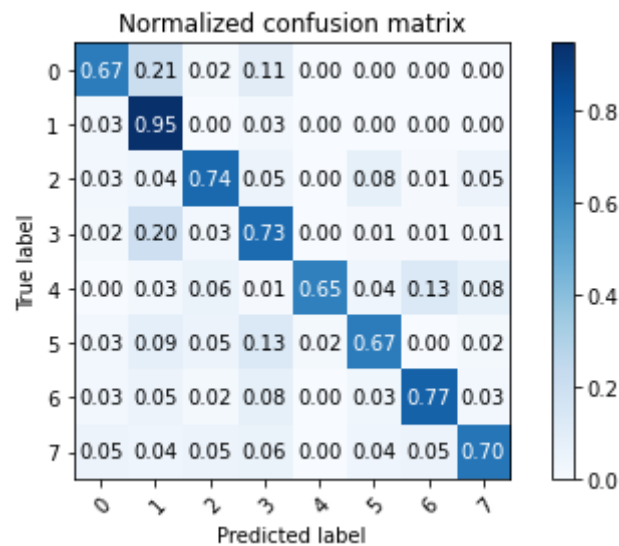
$$F1 - Score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Trong đó:

- True Positive (TP) là trường hợp dự đoán đúng nhãn là True và nhãn thực tế là True.
- True Negative (TN) là trường hợp dự đoán đúng nhãn là False và nhãn thực tế là False.
- False Positive (FP) là trường hợp dự đoán sai nhãn là True và nhãn thực tế là False.
- False Negative (FN) là trường hợp dự đoán sai nhãn là False và nhãn thực tế là True.

Confusion Matrix là một ma trận thể hiện mối quan hệ giữa số lượng trường hợp trong thực tế so với số lượng trường hợp trong thực tế đã được mô hình đưa ra dự đoán. Nó cho chúng ta cái nhìn trực quan về số trường hợp nào

là dự đoán chính xác và trường hợp nào là dự đoán không chính xác cho mỗi lớp phân loại.

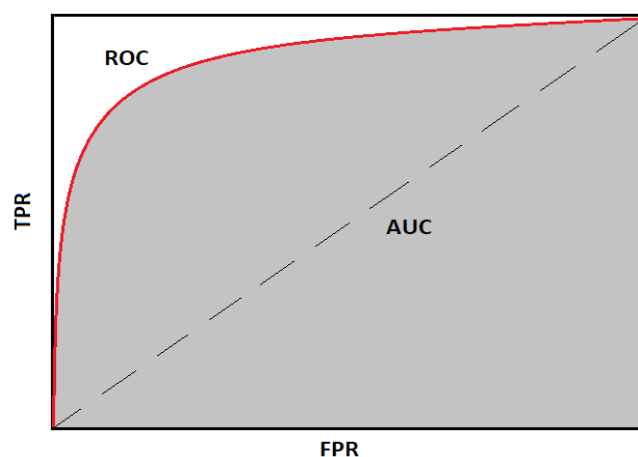


Hình 5. 15: Ví dụ về Confusion Matrix

ROC là đường cong biểu diễn khả năng phân loại của một mô hình tại các ngưỡng threshold [36]. Đường cong này dựa trên hai chỉ số là: True Positive Rate (TPR) và False Positive Rate (FPR) được tính theo 2 công thức sau:

$$TPR = \frac{TP}{TP + FP}$$

$$FPR = \frac{FP}{TN + FN}$$



Hình 5. 16: Ví dụ về ROC và AUC

Đúng với cái tên của nó AUC (area under curve) là phần diện tích nằm dưới đường cong ROC và trên trục hoành x có giá trị nằm trong khoảng [0, 1]. Khi diện tích này càng lớn thì đường cong ROC có xu hướng tiệm cận đường

thẳng  $y = 1$  thì khả năng phân loại của mô hình càng tốt. Khi đường cong ROC nằm sát với đường chéo đi qua hai điểm (0, 0) và (1, 1), lúc này mô hình sẽ tương đương với một phân loại ngẫu nhiên [36].

### 5.5. Kết quả thực nghiệm

Các kịch bản đã được chia thành 2 phần là tập dữ liệu đào tạo (80%) và tập dữ liệu để đánh giá (20%). Cụ thể là sau khi đào tạo các mô hình máy học trên tập dữ liệu đào tạo, chúng tôi sẽ đánh giá chúng bằng tập dữ liệu đánh giá đã chia sẵn từ trước.

#### 5.5.1. Kết quả đánh giá trường hợp kịch bản sử dụng mã hoá với giá trị số

Trong trường hợp này có tổng cộng 16 kết quả cho 16 kịch bản được trình bày trong bảng sau:

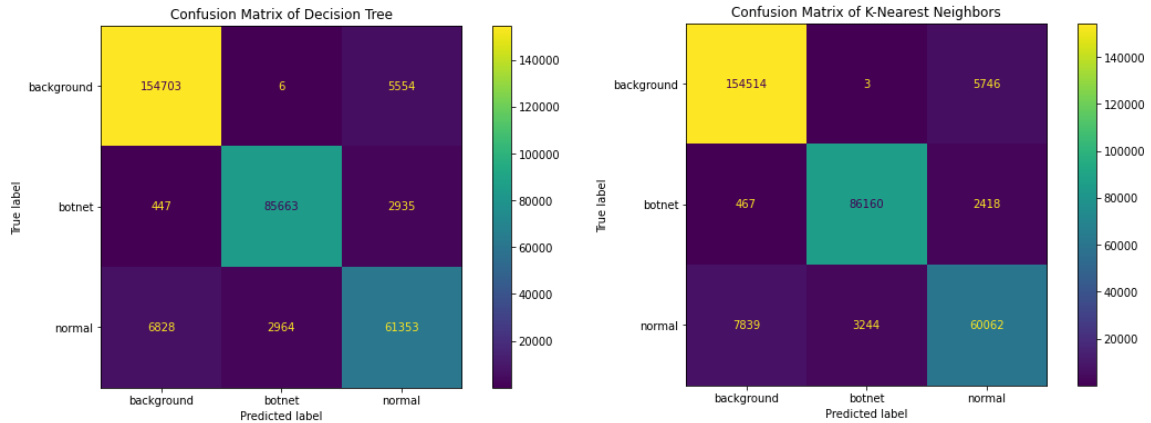
*Bảng 5. 5: Trường hợp kịch bản sử dụng mã hoá với giá trị số*

Metrics Kịch bản			Accuracy	Recall label 1	Precision label 1	F1 score label 1	ROC AUC label 0	ROC AUC label 1	ROC AUC label 2
4.1	None	DT	90.71%	0.930	0.924	0.927	0.92	0.95	0.91
		RF	92.04%	0.947	0.934	0.940	0.98	0.99	0.98
		NB	27.41%	5.6e-5	0.277	0.0001	0.75	0.80	0.68
		XGB	84.24%	0.834	0.880	0.856	0.95	0.97	0.95
	5.1	LG	62.27%	0.512	0.865	0.643	0.66	0.71	0.71
		KNN	87.88%	0.898	0.877	0.888	0.96	0.97	0.95
	5.2	LG	62.27%	0.512	0.865	0.643	0.64	0.72	0.76
		KNN	91.15%	0.941	0.917	0.929	0.97	0.98	0.97
4.2	None	DT	94.15%	0.962	0.966	0.964	0.98	0.98	0.93
		RF	92.04%	0.947	0.934	0.940	0.98	0.99	0.98
		NB	57.61%	0.0008	0.459	0.001	0.83	0.81	0.63
		XGB	90.29%	0.902	0.938	0.919	0.98	0.99	0.95
	5.1	LG	70.18%	0.511	0.879	0.646	0.91	0.90	0.81
		KNN	93.21%	0.958	0.951	0.955	0.98	0.99	0.95
	5.2	LG	70.19%	0.511	0.879	0.646	0.85	0.83	0.81
		KNN	93.84%	0.967	0.963	0.965	0.98	0.99	0.95

Đánh giá: có 6 mô hình từ 6 kịch bản đã đạt hiệu suất tốt nhất. Trong đó, 3 mô hình là K-Nearest Neighbors, 2 mô hình là Random Forest và 1 mô hình là Decision Tree. Mô hình có độ chính xác (Accuracy) cao nhất với 94.15% là



Decision Tree trong kịch bản (4.2) và mô hình có độ chính xác (Accuracy) thấp nhất với 27.41% là Naive Bayes trong kịch bản (4.1). Tuy có độ chính xác cao nhất nhưng mô hình Decision Tree trong kịch bản (4.2) lại có ROC-AUC label 2 thấp nhất trong 6 mô hình với 0.93. Để cân nhắc, một mô hình ổn định nhất chúng tôi sẽ xem xét đến confusion matrix của chúng. Và sau khi xem xét, chúng tôi chọn ra 2 mô hình là Decision Tree trong kịch bản (4.2) và K-Nearest Neighbors trong kịch bản (4.2 - 5.2).



Hình 5. 17: So sánh 2 confusion matrix của 2 mô hình

Nhìn vào hình ta thấy được mô hình Decision Tree có dự đoán trên 2 lớp Background và Normal tốt hơn mô hình K-Nearest Neighbors nhưng ngược lại dự đoán của nó trên lớp Botnet là ít chính xác hơn. Nhưng để chọn ra mô hình ổn định hơn trong cả 2 thì ta ưu tiên mô hình Decision Tree hơn vì tốc độ dự đoán của nó nhanh hơn và thời gian đào tạo cũng nhanh hơn so với K-Nearest Neighbors. Do đó, phải đánh đổi một chút độ chính xác trên lớp Botnet.

### 5.5.2. Kết quả đánh giá trường hợp kịch bản sử dụng mã hoá One-hot Encoding

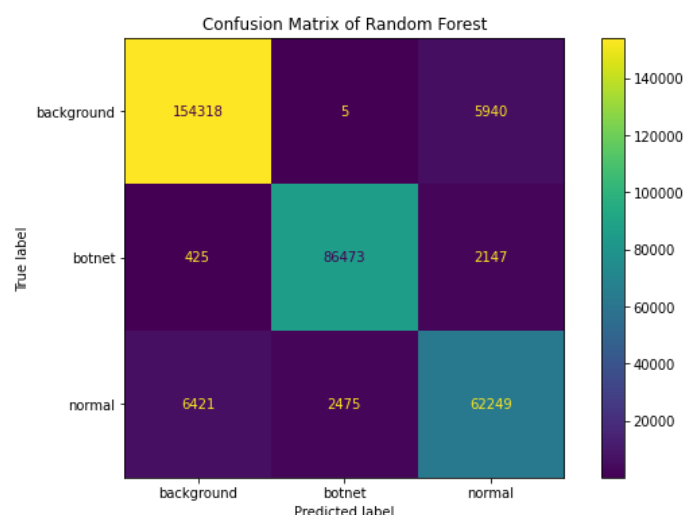
*Khi sử dụng kỹ thuật lựa chọn tính năng với Chi-Squared*

Bảng 5. 6: Trường hợp sử dụng kỹ thuật lựa chọn tính năng với Chi-Squared

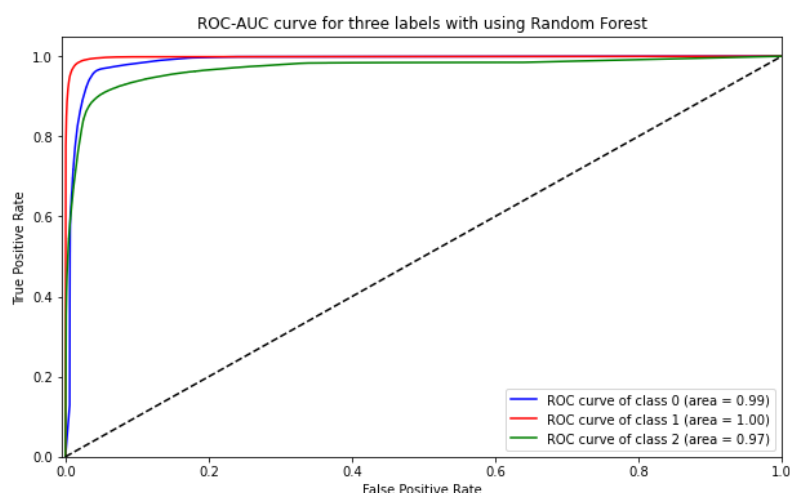
Metrics Kịch bản			Accuracy	Recall label 1	Precision label 1	F1 score label 1	ROC AUC label 0	ROC AUC label 1	ROC AUC label 2
4.1	None	DT	90.72%	0.930	0.924	0.927	0.92	0.95	0.91
		RF	92.04%	0.947	0.934	0.940	0.98	0.99	0.98
		NB	27.41%	5.6e-5	0.277	0.0001	0.73	0.78	0.68
		XGB	84.70%	0.832	0.880	0.856	0.95	0.97	0.95
	5.1	LG	62.55%	0.512	0.881	0.648	0.68	0.71	0.71

	5.2	KNN	87.87%	0.898	0.877	0.888	0.96	0.97	0.95
		LG	62.54%	0.512	0.881	0.648	0.64	0.72	0.76
		KNN	91.14%	0.940	0.918	0.929	0.97	0.98	0.97
4.2	None	DT	94.12%	0.961	0.966	0.963	0.98	0.98	0.93
		RF	94.56%	0.971	0.972	0.971	0.99	1.00	0.97
		NB	57.60%	0.0008	0.459	0.001	0.83	0.81	0.62
		XGB	90.21%	0.905	0.931	0.918	0.98	0.99	0.95
	5.1	LG	70.19%	0.513	0.877	0.647	0.91	0.91	0.81
		KNN	93.29%	0.958	0.952	0.955	0.99	0.98	0.95
	5.2	LG	70.20%	0.513	0.877	0.647	0.89	0.88	0.81
		KNN	93.91%	0.968	0.963	0.966	0.98	0.99	0.95

Đánh giá: trong 16 kịch bản trong bảng trên, chúng tôi thấy có 4 mô hình tốt nhất bao gồm 1 mô hình Decision Tree, 1 mô hình Random Forest và 2 mô hình K-Nearest Neighbors. Mô hình có độ chính xác cao nhất là Random Forest trong kịch bản (4.2) với 94.56% và mô hình có độ chính xác thấp nhất là Naive Bayes trong kịch bản (4.1) với 27.41%. Điều đặc biệt ở đây là cả 4 mô hình tốt nhất đều nằm trong phần kịch bản (4.2) có sử dụng kỹ thuật cân bằng bộ dữ liệu với NearMiss-1. Bên cạnh đó, ta còn thấy được hiệu quả của việc sử dụng 2 kỹ thuật chia tỷ lệ Normalization (5.1) và Standardization (5.2) trên 2 mô hình Logistic Regression và K-Nearest Neighbors khi nó giúp cải thiện độ chính xác cho mô hình. Cuối cùng, chúng tôi chọn ra mô hình Random Forest trong kịch bản (4.2) là mô hình tốt nhất vì có độ chính xác cao nhất và hơn thế là ROC-AUC label 1 bằng 1.00.



Hình 5. 18: Confusion matrix của Random Forest trong kịch bản (4.2)



Hình 5. 19: ROC-AUC của Random Forest trong kịch bản (4.2)

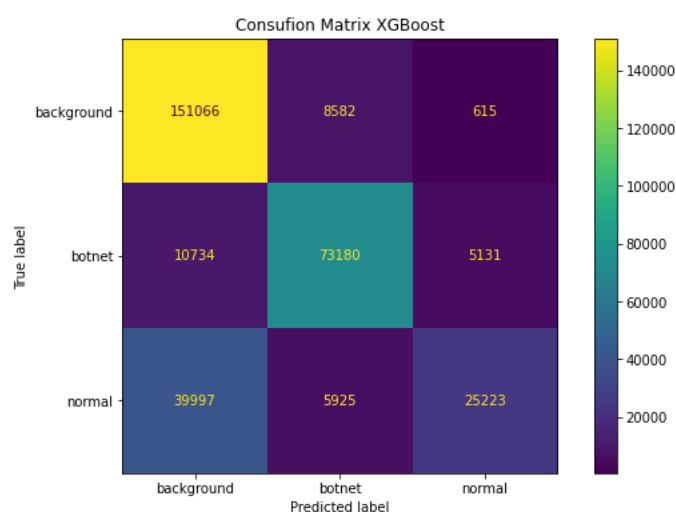
### Khi sử dụng kỹ thuật lựa chọn tính năng với ANOVA

Bảng 5. 7: Trường hợp sử dụng kỹ thuật lựa chọn tính năng với ANOVA

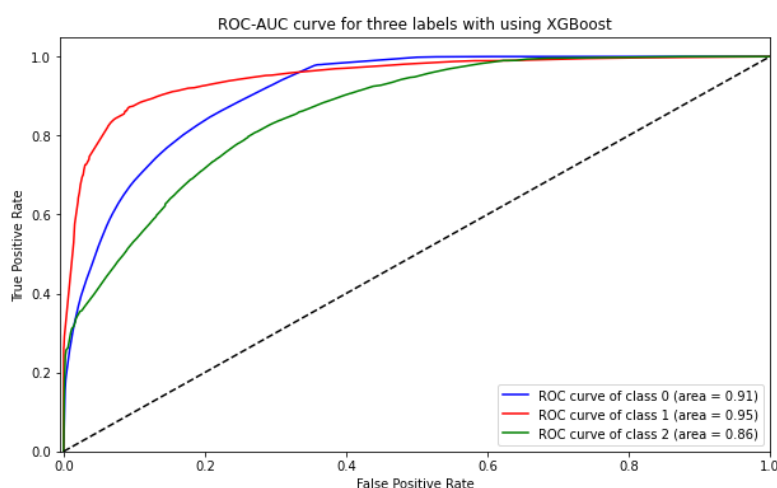
Metrics  Kịch bản			Accuracy	Recall label 1	Precision label 1	F1 score label 1	ROC AUC label 0	ROC AUC label 1	ROC AUC label 2
4.1	None	DT	69.73%	0.715	0.787	0.749	0.77	0.82	0.79
		RF	69.87%	0.722	0.786	0.752	0.82	0.92	0.82
		NB	42.43%	0.522	0.797	0.631	0.74	0.84	0.61
		XGB	70.70%	0.776	0.774	0.775	0.83	0.92	0.84
	5.1	LG	62.51%	0.512	0.881	0.648	0.68	0.71	0.71
		KNN	69.85%	0.744	0.771	0.757	0.81	0.89	0.79
	5.2	LG	62.54%	0.512	0.881	0.648	0.68	0.71	0.71
		KNN	69.89%	0.744	0.773	0.758	0.80	0.89	0.79
4.2	None	DT	77.66%	0.817	0.832	0.825	0.91	0.90	0.82
		RF	77.68%	0.815	0.834	0.824	0.91	0.94	0.85
		NB	65.81%	0.670	0.634	0.651	0.82	0.88	0.73
		XGB	77.84%	0.821	0.834	0.828	0.91	0.95	0.86
	5.1	LG	69.23%	0.513	0.830	0.635	0.83	0.89	0.73
		KNN	76.52%	0.769	0.861	0.812	0.88	0.92	0.80
	5.2	LG	69.31%	0.513	0.834	0.635	0.83	0.89	0.72
		KNN	76.42%	0.768	0.869	0.815	0.87	0.92	0.80

Đánh giá: trong trường hợp sử dụng kỹ thuật lựa chọn tính năng với ANOVA, ta thấy chất lượng của các mô hình bị giảm sút nhiều so với kỹ thuật

lựa chọn tính năng với Chi-Squared ở phần trên. Điều này có thể là do việc mất đi 3 cột/ tính năng ‘TotPkts’, ‘TotBytes’, ‘SrcBytes’ khi sử dụng ANOVA khiến mô hình học tập không tốt. Chúng tôi xem xét 3 mô hình cho kết quả tốt nhất là Decision Tree, Random Forest và XGBoost, cả 3 mô hình đều thuộc trong kịch bản (4.2). XGBoost trong kịch bản (4.2) là mô hình có độ chính xác cao nhất với 77.84% và vẫn là Naive Bayes trong kịch bản (4.1) có độ chính xác thấp nhất chỉ 42.43%. Điều này chỉ ra thuật toán Naive Bayes không phù hợp với bài toán này hoặc cần chỉnh sửa thêm các tham số cho nó để cải thiện hơn chất lượng mô hình.



Hình 5. 20: Confusion matrix của XGBoost trong kịch bản (4.2)



Hình 5. 21: ROC-AUC của XGBoost trong kịch bản (4.2)

Kết luận: sau khi đánh giá qua 18 kịch bản trên, chúng tôi đã tìm ra được mô hình tốt nhất là mô hình Random Forest trong kịch bản có sử dụng mã hoá One-hot Encoding, lựa chọn tính năng với Chi-Squared và cân bằng tập dữ liệu với NearMiss-1. Để đánh giá thêm về chất lượng của mô hình thì chúng tôi

sẽ triển khai mô hình vào ứng dụng đã được trình bày trong chương số 4. Các kết quả của ứng dụng sẽ nằm trong phần tiếp theo.

## 5.6. Kết quả ứng dụng

Khởi chạy ứng dụng.

Phần load model có thể lên tới 20 hoặc 30 giây do dung lượng của mô hình cao.

```
trung@Ubuntu:~/DDoS$ sudo python3 network_traffic_detection.py ens33 20
Loading model...
Done!!!
tcpdump: listening on ens33, link-type EN10MB (Ethernet), capture size 262144 bytes
```

Hình 5. 22: Ví dụ chạy ứng dụng

Phần nội dung được hiển thị với 4 phần chính:

```
=> 1 ➡
tcpdump: listening on ens33, link-type EN10MB (Ethernet), capture size 262144 bytes
10 packets captured
10 packets received by filter
0 packets dropped by kernel
StartTime,Dur,Proto,SrcAddr,Sport,Dir,DstAddr,Dport,State,sTos,dTos,TotPkts,TotBytes,SrcBytes
2022/01/10 11:23:23.739752,0.000030,icmp,192.168.1.2,0x00008, <->,192.168.1.108,0x0100,ECO,0,0,2,148,74
2022/01/10 11:23:24.745962,0.000025,icmp,192.168.1.2,0x00008, <->,192.168.1.108,0x0100,ECO,0,0,2,148,74
2022/01/10 11:23:25.485920,0.000012,arp,192.168.1.2,, who,192.168.1.108,,CON,,2,102,60
2022/01/10 11:23:25.749323,0.000024,icmp,192.168.1.2,0x00008, <->,192.168.1.108,0x0100,ECO,0,0,2,148,74
2022/01/10 11:23:26.754453,0.000028,icmp,192.168.1.2,0x00008, <->,192.168.1.108,0x0100,ECO,0,0,2,148,74
2022/01/10 11:23:28.599489,1.010336,man,0,0,,28,1,STP,,,0,11071768,0
[2 2 2 2 2] ➡ 3
||=====||
||=>>>>>> BACKGROUND-NORMAL TRAFFIC <<<<<<=|| ➡ 4
||=====||
```

Hình 5. 23: 4 thành phần chính trong kết quả hiển thị

Trong đó:

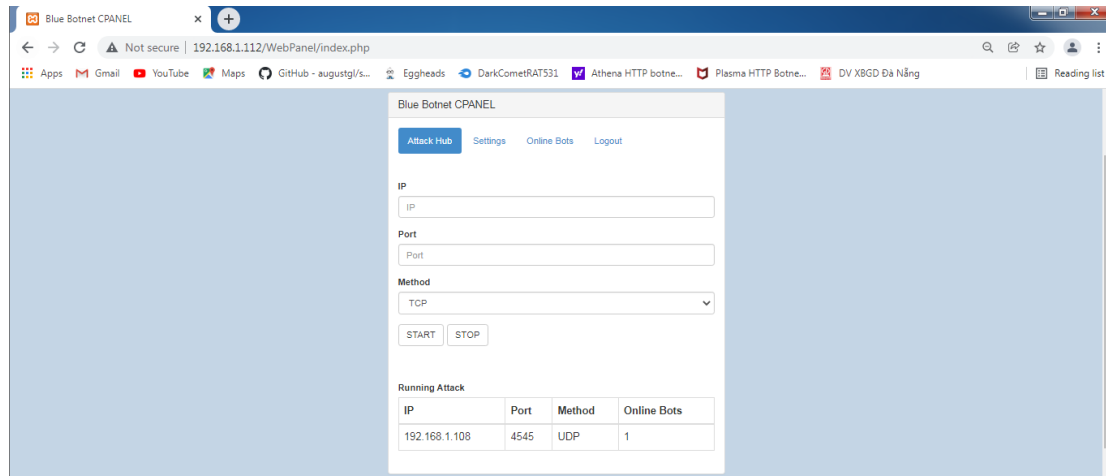
- 1: số thứ tự của mỗi lần bắt gói tin.
- 2: hiển thị lưu lượng bắt được theo định dạng NetFlow từ tệp bitnetflow.
- 3: mảng giá trị luồng NetFlow mà mô hình dự đoán với 3 giá trị 0, 1, 2.
- 4: hiển thị kết quả tổng hợp cho các dự đoán là lưu lượng nào.

Trường hợp lưu lượng Background-Normal được tạo ra với lệnh ping từ máy có IP: 192.168.1.2 đến máy mục tiêu đang chạy ứng dụng là máy có IP: 192.168.1.108.

```
trung@Ubuntu:~/DDoS$ sudo python3 network_traffic_detection.py ens33 20
Loading model...
Done!!!
tcpdump: listening on ens33, link-type EN10MB (Ethernet), capture size 262144 bytes
20 packets captured
21 packets received by filter
0 packets dropped by kernel
4294967292 packets dropped by interface
StartTime,Dur,Proto,SrcAddr,Sport,Dir,DstAddr,Dport,State,sTos,dTos,TotPkts,TotBytes,SrcBytes
2022/01/10 10:31:18.744465,0.000000,tcp,192.168.1.112,49169, <?,20.42.73.29,443,RST,,0,1,60,0
2022/01/10 10:31:24.971868,0.000020,arp,192.168.1.2,, who,192.168.1.108,,CON,,2,102,60
2022/01/10 10:31:24.972004,0.000023,icmp,192.168.1.2,0x00008, <->,192.168.1.108,0x0100,ECO,0,0,2,148,74
2022/01/10 10:31:25.987803,0.000050,icmp,192.168.1.2,0x00008, <->,192.168.1.108,0x0100,ECO,0,0,2,148,74
2022/01/10 10:31:26.996713,0.000078,icmp,192.168.1.2,0x00008, <->,192.168.1.108,0x0100,ECO,0,0,2,148,74
2022/01/10 10:31:28.021415,0.000032,icmp,192.168.1.2,0x00008, <->,192.168.1.108,0x0100,ECO,0,0,2,148,74
2022/01/10 10:31:29.033124,0.000100,icmp,192.168.1.2,0x00008, <->,192.168.1.108,0x0100,ECO,0,0,2,148,74
2022/01/10 10:31:30.051295,0.000032,icmp,192.168.1.2,0x00008, <->,192.168.1.108,0x0100,ECO,0,0,2,148,74
2022/01/10 10:31:30.087522,0.000744,arp,192.168.1.108,, who,192.168.1.2,,CON,,2,102,42
2022/01/10 10:31:31.058886,0.000031,icmp,192.168.1.2,0x00008, <->,192.168.1.108,0x0100,ECO,0,0,2,148,74
2022/01/10 10:31:32.063763,0.000000,icmp,192.168.1.2,0x00008, <->,192.168.1.108,0x0100,ECO,0,0,1,74,74
2022/01/10 10:31:35.161141,1.393635,man,0,0,,22,1,STP,,,0,11215388,0
[2 2 2 2 2 2 2 2 2 2]
||=====||
||=>>>>>> BACKGROUND-NORMAL TRAFFIC <<<<<<=||
||=====||
```

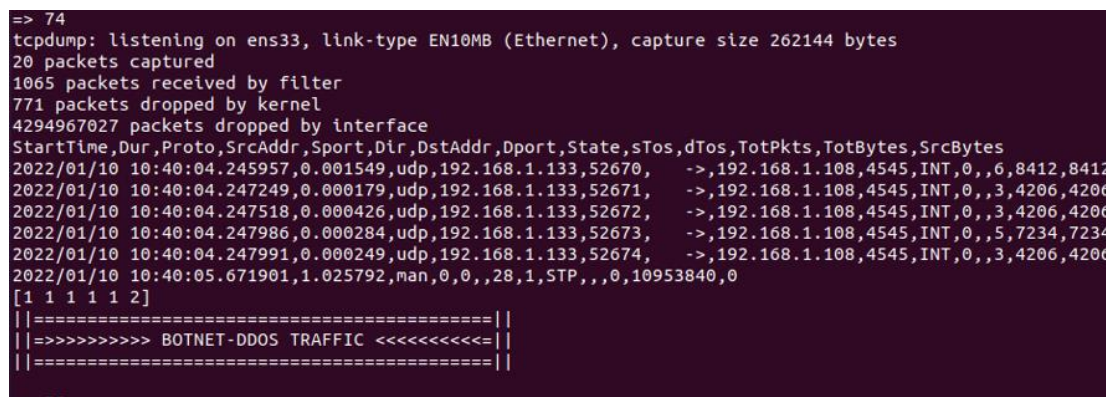
Hình 5. 24: Kết quả cho trường hợp phát hiện lưu lượng Background-Normal

Trường hợp lưu lượng Botnet-DDoS được tạo ra từ BlueBotnet. Một máy chủ với địa chỉ IP: 192.168.1.112 gửi lệnh cho máy botnet client (IP: 192.168.1.133) để thực hiện tấn công UDP Flood đến máy mục tiêu đang chạy ứng dụng là máy có IP: 192.168.1.108.



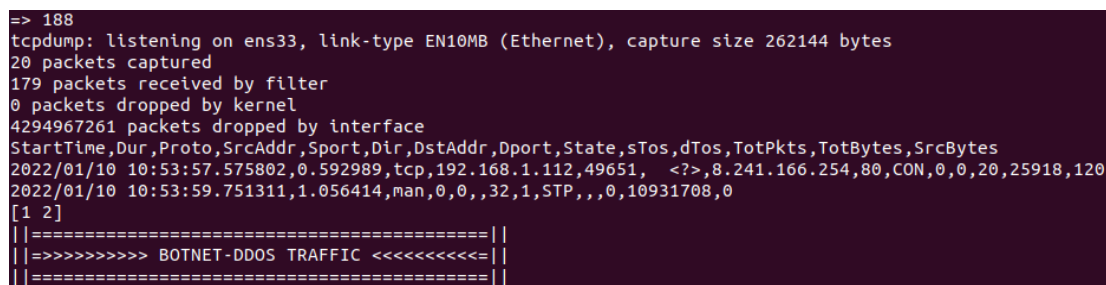
Hình 5. 25: Máy chủ gửi lệnh hướng dẫn tấn công cho botnet client

Lúc này trên máy mục tiêu chạy ứng dụng sẽ phát hiện ra lưu lượng Botnet-DDoS được gửi đến.



Hình 5. 26: Kết quả cho trường hợp phát hiện lưu lượng Botnet-DDoS

Tuy nhiên kết quả có thể không chính xác trong 1 số trường hợp.



Hình 5. 27: Kết quả dự đoán bị sai cho lưu lượng bình thường

Các kết quả hiển thị trên terminal liên tục được lưu lại trong tệp log.txt với thông tin dự đoán tương ứng.



```
StartTime,Dur,Proto,SrcAddr,Sport,Dir,DstAddr,Dport,State,sTos,dTos,TotPkts,TotBytes,SrcBytes
2022/01/10 10:45:26.623655,0.000005,udp,192.168.1.133,0,->,192.168.1.108,0,INT,0,,2,2692,2692
2022/01/10 10:45:26.623697,0.000000,icmp,192.168.1.108,0x0303,->,192.168.1.133,0xc111,URP,192,,1,590,590
2022/01/10 10:45:26.624062,0.000008,udp,192.168.1.133,61832,->,192.168.1.108,4545,INT,0,,3,4206,4206
2022/01/10 10:45:26.624084,0.000004,udp,192.168.1.133,61867,->,192.168.1.108,4545,INT,0,,3,4206,4206
2022/01/10 10:45:26.624095,0.000017,udp,192.168.1.133,61868,->,192.168.1.108,4545,INT,0,,6,8412,8412
2022/01/10 10:45:26.624117,0.000040,udp,192.168.1.133,61869,->,192.168.1.108,4545,INT,0,,5,7234,7234
2022/01/10 10:45:28.399960,1.016288,man,0,0,,27,1,STP,,,0,10953992,0
INFORMATION: BOTNET-DDOS TRAFFIC

StartTime,Dur,Proto,SrcAddr,Sport,Dir,DstAddr,Dport,State,sTos,dTos,TotPkts,TotBytes,SrcBytes
2022/01/10 10:45:29.134184,0.000077,udp,192.168.1.133,64065,->,192.168.1.108,4545,INT,0,,6,8412,8412
2022/01/10 10:45:29.134228,0.000019,udp,192.168.1.133,64066,->,192.168.1.108,4545,INT,0,,6,8412,8412
2022/01/10 10:45:29.134275,0.000013,udp,192.168.1.133,64067,->,192.168.1.108,4545,INT,0,,3,4206,4206
2022/01/10 10:45:29.134301,0.000156,udp,192.168.1.133,64068,->,192.168.1.108,4545,INT,0,,5,7234,7234
2022/01/10 10:45:30.553065,1.021328,man,0,0,,29,1,STP,,,0,10951828,0
INFORMATION: BOTNET-DDOS TRAFFIC

StartTime,Dur,Proto,SrcAddr,Sport,Dir,DstAddr,Dport,State,sTos,dTos,TotPkts,TotBytes,SrcBytes
2022/01/10 10:45:31.303387,6.144287,icmp,192.168.1.108,0x010b,->,192.168.1.133,0x0000,TXD,192,,6,3540,3540
2022/01/10 10:45:32.273466,2.995244,tcp,192.168.1.112,49479,->,93.31.48.164,80,REQ,0,,2,132,132
2022/01/10 10:45:32.305108,0.000000,tcp,192.168.1.112,49471,<?,93.31.48.164,80,RST,,0,1,60,0
2022/01/10 10:45:33.471122,3.029589,udp,192.168.1.112,50390,->,239.255.255.250,1900,INT,0,,4,860,860
2022/01/10 10:45:34.174422,0.000016,arp,192.168.1.133,,who,192.168.1.108,,CON,,,2,102,60
2022/01/10 10:45:34.970044,0.000009,arp,192.168.1.112,,who,192.168.1.100,,CON,,,2,120,60
2022/01/10 10:45:35.410680,0.000188,tcp,192.168.1.133,49190,<?>,192.168.1.112,80,FIN,0,0,2,120,60
2022/01/10 10:45:36.719271,0.000000,tcp,192.168.1.112,49474,->,194.63.140.240,8080,REQ,0,,1,62,62
2022/01/10 10:45:39.598458,1.023966,man,0,0,,31,1,STP,,,0,11206424,0
INFORMATION: BACKGROUND-NORMAL TRAFFIC
```

Hình 5. 28: Kết quả trong tệp log.txt

Đồng thời các tệp pcap cũng được lưu lại liên tục trong thư mục PCAP khi chạy ứng dụng.

```
trung@ubuntu:~/ddos$ cd PCAP/
trung@ubuntu:~/ddos/PCAP$ ls
capture_0.pcap  capture_114.pcap  capture_19.pcap  capture_33.pcap  capture_48.pcap  capture_62.pcap  capture_77.pcap  capture_91.pcap
capture_100.pcap capture_115.pcap  capture_34.pcap  capture_49.pcap  capture_63.pcap  capture_78.pcap  capture_92.pcap
capture_101.pcap capture_116.pcap  capture_20.pcap  capture_35.pcap  capture_4.pcap  capture_64.pcap  capture_79.pcap  capture_93.pcap
capture_102.pcap capture_117.pcap  capture_21.pcap  capture_36.pcap  capture_50.pcap  capture_65.pcap  capture_7.pcap  capture_94.pcap
capture_103.pcap capture_118.pcap  capture_22.pcap  capture_37.pcap  capture_51.pcap  capture_66.pcap  capture_80.pcap  capture_95.pcap
capture_104.pcap capture_119.pcap  capture_23.pcap  capture_38.pcap  capture_52.pcap  capture_67.pcap  capture_81.pcap  capture_96.pcap
capture_105.pcap capture_120.pcap  capture_24.pcap  capture_39.pcap  capture_53.pcap  capture_68.pcap  capture_82.pcap  capture_97.pcap
capture_106.pcap capture_121.pcap  capture_25.pcap  capture_40.pcap  capture_54.pcap  capture_69.pcap  capture_83.pcap  capture_98.pcap
capture_107.pcap capture_122.pcap  capture_26.pcap  capture_41.pcap  capture_55.pcap  capture_70.pcap  capture_84.pcap  capture_99.pcap
capture_108.pcap capture_123.pcap  capture_27.pcap  capture_42.pcap  capture_56.pcap  capture_71.pcap  capture_85.pcap  capture_9.pcap
capture_109.pcap capture_124.pcap  capture_28.pcap  capture_43.pcap  capture_57.pcap  capture_72.pcap  capture_86.pcap
capture_1.pcap  capture_125.pcap  capture_29.pcap  capture_44.pcap  capture_58.pcap  capture_73.pcap  capture_87.pcap
capture_10.pcap capture_126.pcap  capture_30.pcap  capture_45.pcap  capture_59.pcap  capture_74.pcap  capture_88.pcap
capture_110.pcap capture_127.pcap  capture_31.pcap  capture_46.pcap  capture_60.pcap  capture_75.pcap  capture_89.pcap
capture_111.pcap capture_128.pcap  capture_32.pcap  capture_47.pcap  capture_61.pcap  capture_76.pcap  capture_90.pcap
```

Hình 5. 29: Kết quả trong thư mục PCAP

## 5.7. Tổng kết và hướng phát triển

Tóm lại, thông qua khoá luận này, nhóm chúng tôi đã có cơ hội tìm hiểu về các thuật toán máy học, bên cạnh đó là thực hiện thêm được các kỹ thuật tiền xử lý cho tập dữ liệu đầu vào sao cho tốt nhất khi đào tạo trên các mô hình máy học. Đồng thời cũng đã xây dựng, đào tạo thành công các mô hình máy học và triển khai nó trên một ứng dụng cụ thể. Cuối cùng, chúng tôi đã đưa ra các đánh giá cho các mô hình dựa trên 18 trường hợp kịch bản được chia ra từ bộ dữ liệu ban đầu. Trong đó, mô hình đạt hiệu suất cao nhất và tối ưu nhất là mô hình Random Forest có sử dụng mã hoá One-hot Encoding, lựa chọn tính năng với Chi-Squared và cân bằng tập dữ liệu với NearMiss-1. Mô hình đã được triển khai vào ứng dụng và mang lại một kết quả tương đối tốt.

Hướng phát triển của khoá luận này mà chúng tôi muốn nhắm đến là xây dựng thêm các mô hình học sâu (Deep Learning) và mở rộng thêm tập dữ liệu để tăng hiệu suất trên các mô hình học sâu. Không chỉ vậy, chúng tôi còn muốn

triển khai các mô hình này trên các nền tảng đám mây hoặc trên các website với giao diện thân thiện để các quản trị viên có thể thao tác tốt hơn. Cuối cùng là xây dựng một hệ cơ sở dữ liệu hoàn thiện để lưu trữ các thông tin về lưu lượng mạng nhằm mục đích phục vụ các nhà phân tích, bảo mật mạng có thể dùng chúng làm cơ sở để phát triển các công cụ, phần mềm chống lại việc tấn công DDoS.



## Tài liệu tham khảo

### *Tiếng Việt*

[1] Nguyễn Thành Hữu, “Nghiên cứu về DDOS và giải pháp ngăn chặn”, Hà Nội, 2014.

### *Tiếng Anh*

[2] Sebastian Garcia, Martin Grill, Jan Stiborek and Alejandro Zunino, “An empirical comparison of botnet detection methods”, Computers and Security Journal, Elsevier. 2014. Vol 45, pp 100-123. <http://dx.doi.org/10.1016/j.cose.2014.05.011>.

[3] D.Vanitha, Mr.R.Chandrasekar, “Detection of flooding ddos attack using anomaly and signature based intrusion detection system”, International Journal Of Engineering Research and Technology(IJERT), ICSEM-2013 Conference Proceedings.

[4] Li, Liying & Zhou, Jianying & Xiao, Ning. (2007). “DDoS Attack Detection Algorithms Based on Entropy Computing”. 4861. 452-466. 10.1007/978-3-540-77048-0\_35.

[5] Navale, G. S., Kasbekar, V., Ganjepatil, V., & Bugade, S. (2014), “Detecting and analyzing DDos attack using Map Reduce in Hadoop”, Sinhgad Institute of Technology and Science, Narhe, Pune 41 University of Pune, India.

[6] Kok, S. H., Azween Bin Abdullah, Mahadevan Supramaniam, Thulasyammal Ramiah Pillai and Ibrahim Abaker Targio Hashem. “A Comparison of Various Machine Learning Algorithms in a Distributed Denial of Service Intrusion.” (2019).

[7] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization”, 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, January 2018.

[8] Sambangi, Swathi, and Lakshmeeswari Gondi. 2020. “A Machine Learning Approach for DDoS (Distributed Denial of Service) Attack Detection Using Multiple Linear Regression”, Proceedings 63, no. 1: 51. <https://doi.org/10.3390/proceedings2020063051>.

[9] T. -T. Nguyen, C. -S. Shieh, C. -H. Chen and D. Miu, “Detection of Unknown DDoS Attacks with Deep Learning and Gaussian Mixture Model”, 2021 4th International Conference on Information and Computer Technologies (ICICT), 2021, pp. 27-32, doi: 10.1109/ICICT52872.2021.00012.

[10] Alghazzawi, Daniyal & Bamasaq, Omaira & Ullah, Hayat & Asghar, Dr. Muhammad. (2021). “Efficient Detection of DDoS Attacks Using a Hybrid Deep Learning Model with Improved Feature Selection”. Applied Sciences. 11. 11634. 10.3390/app112411634.

- [11] Gunter Ollmann, VP of Research, Damballa, Inc., “Botnet Communication Topology, Understanding the intricacies of botnet Command-and-Control”, 2009.
- [12] Jehad Ali, Rehanullah Khan, Nasir Ahmad, “Random Forests and Decision Trees”, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 5, No 3, September 2012.
- [13] Bentéjac, Candice & Csörgő, Anna & Martínez-Muñoz, Gonzalo. (2019). “A Comparative Analysis of XGBoost”.
- [14] Max Kuhn · Kjell Johnson, “Applied Predictive Modeling”, page 490, 2013.
- [15] Saeys Y, Inza I, Larranaga P (2007). “A Review of Feature Selection Techniques in Bioinformatics.” Bioinformatics, 23(19), 2507–2517.
- [16] Inderjeet Mani and I Zhang. “Knn approach to unbalanced data distributions: a case study involving information extraction”. In Proceedings of workshop on learning from imbalanced datasets, volume 126. 2003.
- [17] Vishwakarma, Anand Ravindra, “Network Traffic Based Botnet Detection Using Machine Learning” (2020), Master's Projects. 917.

### ***Trang Web***

- [18] Vũ Hữu Tiệp, “Bài 2: Phân nhóm các thuật toán Machine Learning”, <https://machinelearningcoban.com/2016/12/27/categories/#supervised-learning-hoc-co-giam-sat>, (2016), 25/12/2021 19:53
- [19] Hem Bahadur Gurung, “Decision-Trees-and-XGBoost-Algorithm-Documentation”, <https://github.com/Hem7513/Decision-Trees-and-XGBoost-Algorithm-Documentation>, (2021), 02/01/2022 15:30
- [20] <https://www.javatpoint.com/machine-learning-naive-bayes-classifier>, 25/12/2021 18:16
- [21] <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>, 02/01/2022 16:18
- [22] <https://www.javatpoint.com/logistic-regression-in-machine-learning>, 02/01/2022 16:45
- [23] Jason Brownlee, “How to Choose a Feature Selection Method For Machine Learning”, <https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/>, (2019), 28/12/2021 8:23
- [24] [https://en.wikipedia.org/wiki/Chi-squared\\_test](https://en.wikipedia.org/wiki/Chi-squared_test), 01/01/2022 21:04
- [25] <https://www.cuemath.com/chi-square-formula/>, 01/01/2022 20:56
- [26] <https://byjus.com/anova-formula/>, 01/01/2022 15:45
- [27] Jason Brownlee, “Undersampling Algorithms for Imbalanced Classification”, <https://machinelearningmastery.com/undersampling-algorithms-for-imbalanced-classification/>, (2021), 27/12/2021 00:05

- [28] Aniruddha Bhandari, “Feature Scaling for Machine Learning: Understanding the Difference Between Normalization vs. Standardization”, <https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/#>, (2020), 27/12/2021 17:57
- [29] <https://www.tcpdump.org/manpages/tcpdump.1.html>, 31/12/2021 20:15
- [30] <https://qosient.com/argus/manuals.shtml>, 31/12/2021 22:33
- [31] <https://docs.python.org/3/library/sys.html>, 04/01/2022 23:05
- [32] <https://docs.python.org/3/library/subprocess.html>, 31/12/2021 22:51
- [33] <https://www.python.org/>, 31/12/2021 23:02
- [34] <https://www.vmware.com/products/workstation-pro/workstation-pro-evaluation.html>, 31/12/2021 23:36
- [35] <https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-45/detailed-bidirectional-flow-labels/>
- [36] Phạm Đình Khánh, “Bài 46 - Đánh giá mô hình phân loại trong ML”, <https://phamdinhhkhanh.github.io/2020/08/13/ModelMetric.html>, (2020), 31/12/2021 19:50

### **Hình ảnh**

- [37] <https://codelungtung.wordpress.com/2018/09/12/image-classification-decision-trees/> 25/12/2021 18:05
- [38] <https://ai-pool.com/a/s/random-forests-understanding> 25/12/2021 18:12
- [39] <https://www.javatpoint.com/machine-learning-naive-bayes-classifier> 25/12/2021 18:16
- [40] <https://vietnambiz.vn/thuat-toan-k-lang-gieng-gan-nhat-k-nearest-neighbor-knn-la-gi-2020022911113334.htm> 25/12/2021 18:35
- [41] <https://www.tibco.com/reference-center/what-is-logistic-regression> 25/12/2021 18:42
- [42] [https://imbalanced-learn.org/stable/under\\_sampling.html#controlled-under-sampling](https://imbalanced-learn.org/stable/under_sampling.html#controlled-under-sampling) 01/01/2022 22:00
- [43] <https://pianalytix.com/feature-scaling-in-machine-learning/> 01/01/2022 14:46