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

NGUYỄN PHÚC BẢO DANH - 52000193
NGUYỄN THÀNH AN - 52000621

PHÁT HIỆN PHẦN MỀM ĐỘC HẠI SỬ DỤNG CÁC THUẬT TOÁN HỌC MÁY

CHUYÊN ĐỀ NGHIÊN CỨU 1

KHOA HỌC MÁY TÍNH

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

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

NGUYỄN PHÚC BẢO DANH - 52000193
NGUYỄN THÀNH AN - 52000621

PHÁT HIỆN PHẦN MỀM ĐỘC HẠI SỬ DỤNG CÁC THUẬT TOÁN HỌC MÁY

CHUYÊN ĐỀ NGHIÊN CỨU 1

KHOA HỌC MÁY TÍNH

Người hướng dẫn
TS. Trương Đình Tú

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

LỜI CẢM ƠN

Trong suốt thời gian qua, nhờ sự giảng dạy tận tâm của quý Thầy Cô Khoa Công nghệ thông tin, trường Đại học Tôn Đức Thắng, chúng em đã học hỏi được rất nhiều điều bổ ích và tích lũy cho mình một số kiến thức để hoàn thành bài báo cáo này. Chúng em xin chân thành cảm ơn.

Nhóm em xin chân thành cảm ơn thầy Trương Đình Tú đã tận tình chỉ bảo qua những buổi trao đổi trực tiếp, Thầy đã chỉ nhóm em cách thức làm bài, chỉ điểm những chỗ còn sai sót chưa phù hợp cũng như phải làm sao để trình bày bố cục đẹp. Nếu không có những lời hướng dẫn, dạy bảo của Thầy thì bài thu hoạch của nhóm em cũng rất khó để hoàn thiện. Một lần nữa chúng em xin chân thành cảm ơn Thầy.

Bước đầu đi vào thực tế với nền kiến thức mở rộng, kiến thức của nhóm em còn hạn chế và nhiều bỡ ngỡ. Vì thế, trong quá trình biên soạn khó tránh những sai sót, chúng em rất mong nhận được những ý kiến đóng góp quý báu của Thầy để bài báo cáo hoàn thiện hơn.

TP. Hồ Chí Minh, ngày 22 tháng 03 năm 2024.

Tác giả

(Ký tên và ghi rõ họ tên)

Danh

Nguyễn Phúc Bảo Danh

An

Nguyễn Thành An

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH
TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

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

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

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

TP. Hồ Chí Minh, ngày 22 tháng 03 năm 2024.

Tác giả

(Ký tên và ghi rõ họ tên)

Danh

Nguyễn Phúc Bảo Danh

An

Nguyễn Thành An

PHÁT HIỆN PHẦN MỀM ĐỘC HẠI SỬ DỤNG CÁC THUẬT TOÁN HỌC MÁY

TÓM TẮT

Phần mềm độc hại, hay còn được gọi Malware (Malicious Software), là một trong các hình thức tấn công mạng phức tạp cả về mức độ tinh vi và phổ biến trong những năm gần đây. Những Hacker sử dụng nhiều loại Malware khác nhau với mức độ nguy hiểm và cơ chế hoạt động khác nhau nhằm tấn công các thiết bị điện tử của cá nhân và tổ chức, gây ra những trận khủng hoảng không những về vật chất mà còn làm suy thoái kinh tế của một bộ phận cá nhân, tổ chức đó. May mắn thay, trong thời đại kỹ thuật số hiện nay, nhiều kỹ thuật trí tuệ nhân tạo được sử dụng để phát hiện các phần mềm độc hại và phân biệt chúng với các phần mềm sạch bao gồm các kỹ thuật học máy, học sâu...

Tuy nhiên, đi đôi với sự phát triển các phần mềm độc hại là sự phát triển và gia tăng không ngừng của nhiều biến thể Malware khác nhau, dữ liệu của chúng cũng dần được tăng lên cả về độ lớn và độ phức tạp mà các kỹ thuật trước đây vẫn chưa thể giải quyết triệt để. Trong bài nghiên cứu này, chúng tôi đã sử dụng một số thuật toán học máy với hiệu suất làm việc tương đối cao nhằm mang lại kết quả phát hiện phần mềm độc hại và phân biệt chúng với các phần mềm sạch một cách tối ưu và hiệu quả nhất.

Bộ dữ liệu nghiên cứu được lấy từ nền tảng Kaggle đã được chúng tôi thực hiện tiền xử lý và chọn lọc đặc trưng quan trọng dựa trên thuật toán Random Forest, chọn đặc trưng dựa trên trọng số quan trọng. Chúng tôi đã sử dụng kỹ thuật Grid Search cho một vài thuật toán mà chúng tôi đang nghiên cứu để tìm ra tham số, số đo phù hợp với mô hình. Kết quả đánh giá sẽ dựa trên 5 tiêu chí quan trọng đối với bài toán phân loại nhị phân (phần mềm độc hại hay phần mềm sạch) là Confusion Matrix, độ chính xác (Accuracy), Recall, Precision và F1-score.

Kết quả chúng tôi đạt được với hiệu suất của các mô hình Decision Tree và Random Forest đạt độ chính xác cao hơn 80%, các mô hinh Naive Bayes và Gradient Boosting đạt độ chính xác cao hơn 70%.

MALWARE DETECTION USING MACHINE LEARNING ALGORITHMS

ABSTRACT

Malicious software, also known as Malware (Malicious Software), is one of the most complex forms of cyber attacks in both sophistication and popularity in recent years. Hackers use many different types of malwares with different levels of danger and operating mechanisms to attack the electronic devices of individuals and organizations, causing not only physical but also physical crises. economic recession of a part of that individual or organization. Fortunately, in today's digital age, many artificial intelligence techniques are used to detect malware and distinguish them from clean software including machine learning and deep learning techniques. ...

However, along with the development of malware is the continuous development and increase of many different malware variants, whose data is also gradually increasing in both size and complexity. Previous techniques have not yet been able to completely solve the problem. In this research, we used a number of machine learning algorithms with relatively high performance to provide optimal results for detecting malware and distinguishing it from clean software. the most effective.

The research data set is taken from the Kaggle platform, and we have preprocessed and selected important features based on the Random Forest algorithm, selecting features based on important weights. We have used Grid Search technique for a few algorithms that we are researching to find parameters and measures suitable for the model. The evaluation results will be based on five important criteria for the binary classification problem (malware or benign) which are Confusion Matrix, Accuracy, Recall, Precision and F1-score.

The results we achieved with the performance of the Decision Tree and Random Forest models reached an accuracy higher than 80%, and the Naive Bayes and Gradient Boosting models achieved an accuracy higher than 70%.

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

DANH MỤC CÁC CHỮ VIẾT TẮT

ML	Machine Learning
DL	Deep Learning
DT	Decision Tree
RF	Random Forest
NB	Naïve Bayes
GB	Gradient Boosting
Malware	Malicious Software
LSTM	Long Short – Term Memory
CNN	Convolutional Neural Network
DNN	Deep Neural Network

CHƯƠNG 1. TỔNG QUAN

1.1 Lý do chọn đề tài

Trong thời đại kỷ nguyên số, khi mỗi phần của cuộc sống chúng ta đều gắn liền với không gian mạng, an ninh thông tin trở thành một trụ cột quan trọng đảm bảo cho sự vận hành liền mạch của xã hội. Song hành với sự tiến bộ này là sự gia tăng không ngừng của các cuộc tấn công mạng, điều này đã đặt ra nhiều thách thức cho việc bảo vệ thông tin cá nhân tổ chức và doanh nghiệp. Trong bối cảnh ngày nay, việc phát hiện sớm và chính xác phần mềm độc hại là không chỉ cần thiết mà còn cấp bách. Đề tài nghiên cứu "Phát hiện phần mềm độc hại sử dụng các thuật toán học máy" của chúng tôi đặt mục tiêu giải quyết vấn đề này bằng cách kết hợp nghiên cứu lý thuyết cùng với việc xử lý và phân tích dữ liệu được ghi lại thực tế thông qua áp dụng phương pháp học máy.

Bằng việc áp dụng các thuật toán học máy, chúng tôi hy vọng sẽ làm tăng hiệu quả trong việc phát hiện các loại phần mềm độc hại. Cách tiếp cận này giúp chúng tôi không chỉ nắm bắt và phân tích được các mẫu độc hại mới mà còn tối ưu hóa quá trình phát hiện để nhanh chóng và chính xác hơn. Nghiên cứu của chúng tôi tập trung vào việc kết hợp nhiều thuật toán khác nhau để có thể đạt được kết quả tốt nhất, giảm thiểu sai sót và tăng cường khả năng bảo vệ dữ liệu. Chính vì vậy thông qua nghiên cứu này, chúng tôi muốn xây dựng mô hình học máy phát hiện và phân loại phần mềm độc hại góp phần vào việc làm cho không gian mạng trở nên an toàn hơn cho mọi người và cộng đồng.

1.2 Mục tiêu đề tài

Trước tình hình mà các cuộc tấn công an ninh mạng ngày càng trở nên phức tạp và phổ biến, mỗi sự cố an ninh thông tin có thể dẫn đến hậu quả nghiêm trọng, ảnh hưởng đến từng cá nhân và toàn xã hội. Sự cần thiết của việc nâng cao lớp lá chắn phòng thủ mạng không chỉ là vấn đề cấp thiết của ngành Công nghệ thông tin mà còn là trách nhiệm xã hội. Chúng tôi chọn đề tài này không chỉ bởi ý thức về vai trò của

mình trong việc đối phó với các thách thức an ninh mạng hiện đại mà còn bởi sự quan tâm sâu sắc đối với việc ứng dụng công nghệ để tạo ra những thay đổi tích cực.

Cùng với sự lựa chọn áp dụng các thuật toán học máy khác nhau trong nghiên cứu này, chúng tôi mong muốn được tiếp cận vấn đề một cách khoa học và có hệ thống. Sự đa dạng trong việc sử dụng các thuật toán học máy sẽ làm cho quá trình phát hiện Malware trở nên linh hoạt và đa chiều hơn, đồng thời cũng là cách chúng tôi muốn kết hợp kiến thức đã học để giải quyết các vấn đề thực tiễn.

Mục tiêu cuối cùng của chúng tôi là không chỉ vận dụng những gì đã học để phát triển chuyên đề nghiên cứu, mà còn hướng đến mục đích góp phần xây dựng một xã hội số an toàn hơn và bền vững hơn.

1.3 Đối tượng nghiên cứu

Đối tượng nghiên cứu trong bài báo cáo này bao gồm:

- Bộ dữ liệu phần mềm độc hại được thu thập từ nền tảng Kaggle bao gồm các thông số của mỗi mẫu trong bộ dữ liệu như về tên file, trạng thái của tác vụ, kích thước tác vụ, số vùng nhớ, ... để từ đó chúng tôi sẽ tiến hành xây dựng các mô hình học máy phát hiện và phân biệt phần mềm độc hại hay phần mềm sạch cho bài toán phân loại nhị phân mà chúng tôi đang nghiên cứu.
- Kiến thức tổng quát về phần mềm độc hại và các phương pháp phát hiện chúng, bao gồm cách hoạt động của phần mềm độc hại, các biến thể phổ biến của Malware và các kỹ thuật phát hiện và phân loại phần mềm độc hại và phần mềm sạch.
- Các mô hình học máy được sử dụng trong bài nghiên cứu bao gồm Decision Trees (DT), Random Forests (RF), Naive Bayes (NB), và Gradient Boosting (GB). Các mô hình này sẽ được áp dụng để phát hiện và phân loại phần mềm độc hại từ bộ dữ liệu đã thu thập.

1.4 Phạm vi nghiên cứu

Trong bài nghiên cứu này, chúng tôi đã thu thập và phân tích dữ liệu về phần mềm độc hại từ nền tảng Kaggle cũng như phương pháp phát hiện các phần mềm độc hại

từ nhiều nguồn và các bài báo khoa học khác nhau. Các mô hình học máy phù hợp với bài toán phân loại nhị phân đảm bảo hiệu suất mô hình đạt hiệu quả tốt nhất.
Dựa vào các dữ liệu, thông số, mô hình đã thu thập và nghiên cứu để xây dựng mô hình phát hiện và phát hiện giữa phần mềm độc hại và phần mềm sạch. Đánh giá hiệu suất của các mô hình thông qua các tiêu chí cơ bản nhằm tìm ra thuật toán phù hợp nhất cho mô hình của chúng tôi.

1.5 Cấu trúc bài báo cáo

Bài báo cáo của chúng tôi bao gồm 6 chương như sau:

Chương 1 - Tổng quan đề tài: Trình bày sơ bộ tổng quan về bài nghiên cứu bao gồm lý do chọn đề tài, mục tiêu thực hiện đề tài, đối tượng nghiên cứu, phạm vi nghiên cứu.

Chương 2 - Các nghiên cứu liên quan: Giới thiệu bài toán, trình bày các hướng tiếp cận để giải quyết bài toán. Giới thiệu sơ lược các nghiên cứu có liên quan đến đề tài tìm hiểu.

Chương 3 - Cơ sở lý thuyết và các phương pháp phát hiện Malware: Trình bày lý thuyết về phần mềm độc hại bao gồm khái niệm, cơ chế hoạt động, những biến thể phổ biến của chúng, ... Trình bày các phương pháp khả quan trong việc phát hiện và phân loại phần mềm độc hại.

Chương 4 - Phương pháp nghiên cứu: Trình bày các thuật toán đề xuất sẽ sử dụng để xây dựng các mô hình phát hiện. Trình bày các kỹ thuật hỗ trợ trong quá trình thực nghiệm và xử lý dữ liệu. Trình bày phương pháp các tiêu chí cơ sở để đánh giá kết quả và hiệu suất của các mô hình.

Chương 5 - Kết quả thực nghiệm: Trình bày các vấn đề liên quan đến dữ liệu chuẩn bị, cài đặt thực nghiệm (nền tảng, môi trường, quy trình huấn luyện mô hình, ...) và đánh giá kết quả và hiệu suất của các mô hình.

Chương 6 - Kết luận và hướng phát triển: Trình bày nội dung cũng như kết quả đã đạt được, đánh giá mức độ hoàn thành, ưu nhược điểm và phương hướng phát triển của mô hình nghiên cứu.

CHƯƠNG 2. CÁC NGHIÊN CỨU LIÊN QUAN

2.1 Giới thiệu bài toán và cách tiếp cận để giải quyết

2.1.1 Giới thiệu bài toán

Phát hiện phần mềm độc hại (hay còn gọi là Malware) là một trong những bước quan trọng nhất trong việc bảo vệ hệ thống máy tính và dữ liệu của người dùng trước các mối đe dọa an ninh mạng. Đó không chỉ là một phần trong chiến lược phòng ngừa mà còn là cơ sở để ngăn chặn các cuộc tấn công do phần mềm độc hại gây ra.

Malware không chỉ đơn thuần là những chương trình gây rối hoặc lừa đảo, mà còn là một loại công cụ có thể gây ra những thiệt hại nghiêm trọng đến tính toàn vẹn của hệ thống và thông tin cá nhân. Mức độ nguy hiểm của Malware không ngừng tăng lên qua thời gian, với sự phát triển không ngừng của công nghệ và kỹ thuật tấn công. Hiện nay, Malware đã trở nên cực kỳ phức tạp và tinh vi, thường sử dụng các kỹ thuật che giấu và tránh phát hiện từ các chương trình bảo mật thông thường. Nó có thể tự động lây nhiễm, tự động triển khai và tự động tận dụng các lỗ hổng bảo mật trong hệ thống mà không cần sự can thiệp của người dùng. Các loại Malware phổ biến như Virus, Worm, Trojan, Ransomware, Spyware và Adware không chỉ gây ra sự phiền toái cho người dùng thông qua việc hiển thị quảng cáo không mong muốn hay giả mạo thông tin cá nhân, mà còn có thể gây ra những hậu quả nghiêm trọng như mất dữ liệu, tài sản hoặc thậm chí là kiểm soát toàn bộ hệ thống.

Đối mặt với sự nguy hiểm và phức tạp của Malware, việc phát hiện và ngăn chặn trở thành một thách thức khó khăn đối với ngành Bảo mật thông tin. Để bảo vệ hệ thống máy tính, dữ liệu của cả cá nhân và tổ chức, cần triển khai các giải pháp phát hiện Malware tiên tiến, không thể không kể đến việc áp dụng các thuật toán học máy hoặc học sâu để phát hiện và phân loại phần mềm độc hại tự động một cách hiệu quả và chính xác.

2.1.2 Cách tiếp cận giải quyết

Đối với bài toán phát hiện phần mềm độc hại, có nhiều phương pháp để tiếp cận cũng như giải quyết bài toán, các hướng tiếp cận giải quyết trọng tâm là: tiếp cận thủ công, tiếp cận dựa trên con người và tiếp cận dựa trên máy tính.

2.1.2.1 Tiếp cận thủ công

Trong phương pháp tiếp cận thủ công, các chuyên gia an ninh mạng tập trung vào việc sử dụng sự hiểu biết và kinh nghiệm để phân tích và phát hiện phần mềm độc hại. Họ thường thực hiện kiểm tra thủ công các file, quét mã nguồn và xem xét các dấu hiệu bất thường trong hệ thống để xác định sự tồn tại của Malware. Mặc dù phương pháp này có thể cung cấp kết quả chính xác, nhưng nó đối mặt với nhược điểm khi cần xử lý lượng lớn dữ liệu hoặc khi phải đối mặt với các loại Malware mới và không xác định trước.

2.1.2.2 Tiếp cận dựa trên con người

Đối với phương pháp tiếp cận dựa trên con người, các biện pháp tập trung vào việc giáo dục và huấn luyện người dùng về cách nhận biết và phòng tránh phần mềm độc hại. Các biện pháp bao gồm cung cấp thông tin về các kỹ thuật xâm nhập phổ biến, cách phát hiện email lừa đảo và cách ứng phó với các trường hợp tấn công. Mặc dù giúp giảm thiểu một số loại tấn công, nhưng không thể ngăn chặn hoàn toàn các cuộc tấn công do sự phí phạm hoặc sơ xuất từ người dùng.

2.1.2.3 Tiếp cận dựa trên máy tính

Phương pháp dựa trên máy tính sử dụng các thuật toán học máy hoặc học sâu để phát hiện Malware. Các phương pháp bao gồm sử dụng kỹ thuật học máy để phân tích đặc trưng của phần mềm độc hại hoặc áp dụng học sâu để phân tích dữ liệu lớn và phức tạp. Đây sẽ là giải pháp tự động và hiệu quả nhưng yêu cầu dữ liệu huấn luyện phải đầy đủ và chất lượng, đôi khi độ chính xác chỉ có thể tính toán tương đương hoặc sấp xỉ.

2.2 Các nghiên cứu liên quan đến hướng tìm hiểu

2.2.1 Long Short - Term Memory (LSTM)

Trong nghiên cứu của Yazdinejad et al., họ đã sử dụng mô hình Long Short-Term Memory (LSTM) để xây dựng hệ thống phát hiện phần mềm độc hại. Dữ liệu của họ bao gồm các mã hoạt động cho cả phần mềm độc hại và hoạt động lành tính. Họ đã áp dụng phương pháp xác thực chéo 10 lần trên tập dữ liệu thu được và đạt được độ chính xác phát hiện là 98% [1].

2.2.2 Deep Neural Network (DNN)

Mạng nơ-ron sâu (DNN) là một dạng của mạng nơ-ron nhân tạo được cấu trúc với nhiều tầng ẩn giữa tầng đầu vào và đầu ra. DNN được thiết kế để tự động học cách biểu diễn dữ liệu thông qua nhiều tầng ẩn, mỗi tầng có thể gồm nhiều nơ-ron. Các tầng ẩn này giúp mô hình học được các đặc trưng phức tạp từ dữ liệu đầu vào và tạo ra một biểu diễn tốt hơn cho việc dự đoán hoặc phân loại.

Hwang và đồng nghiệp đã đề xuất một hệ thống phát hiện phần mềm độc hại sử dụng mạng nơ-ron sâu (DNN), được đào tạo trên một bộ dữ liệu gồm 10.000 bản ghi phần mềm độc hại và 10.000 tệp lành tính. Hệ thống này đã được huấn luyện trên 80% dữ liệu và kiểm tra trên 20% còn lại. Kết quả của họ cho thấy hệ thống đạt được độ chính xác 94% [2].

2.2.3 Convolutional Neural Network (CNN)

Mạng nơ-ron tích chập (CNN) là loại mạng nơ-ron nhân tạo thích hợp cho việc xử lý dữ liệu không gian như hình ảnh và video. Nó sử dụng các bộ lọc để tự động học các đặc trưng cục bộ từ dữ liệu đầu vào, giúp nhận diện các đặc trưng như cạnh, góc và mẫu phức tạp.

Ban và cộng sự đã áp dụng mạng nơ-ron tích chập (CNN) để phát hiện phần mềm độc hại trên nền tảng Android. Họ sử dụng tập dữ liệu bao gồm 28.179 bản ghi về các hoạt động phần mềm độc hại từ năm 2018 đến năm 2020. Kết quả thử nghiệm của họ cho thấy phương pháp này đạt độ chính xác 98% và điểm F1 là 0,82 [3].

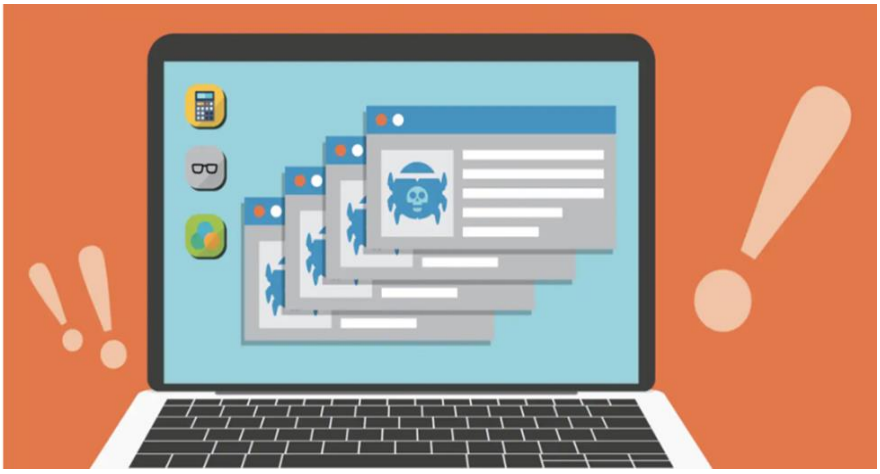
CHƯƠNG 3. CƠ SỞ LÝ THUYẾT

3.1 Lý thuyết về phần mềm độc hại

3.1.1 Định nghĩa cơ bản

Phần mềm độc hại (hay còn gọi là Malware) là một tệp hoặc mã nguồn, thường được lây lan qua mạng, lây nhiễm, nhằm đánh cắp hoặc thực hiện hầu như mọi hành vi mà kẻ tấn công mong muốn. Và vì Malware có rất nhiều biến thể nên có rất nhiều cách để lây nhiễm vào hệ thống máy tính [4]. Mặc dù đa dạng về loại hình và khả năng, Malware thường sẽ có các mục tiêu chính sau đây:

- Cung cấp quyền điều khiển từ xa cho kẻ tấn công sử dụng máy tính bị nhiễm Malware.
- Gửi các tệp thư rác tới các mục tiêu không đề phòng khác.
- Điều tra mạng cục bộ của người dùng.
- Lấy các dữ liệu nhạy cảm của doanh nghiệp.



Hình 3. 1 Malware [4]

3.1.2 Lịch sử

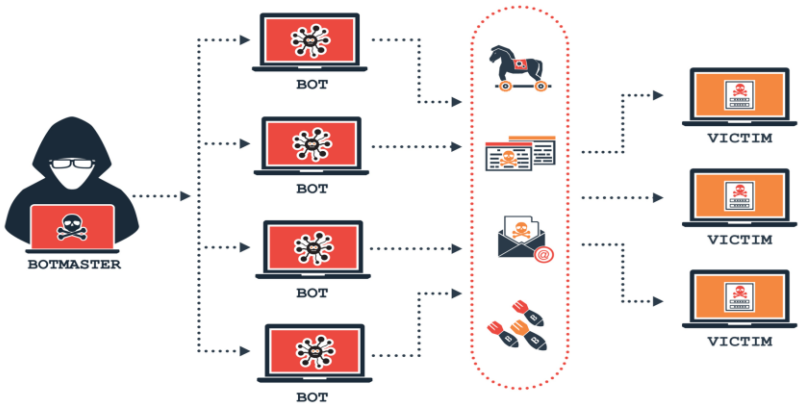
Lịch sử của phần mềm độc hại bắt đầu từ đầu những năm 1970 với một loại Worm có tên “Creeper”. Worm là một chương trình lây lan sang các máy tính khác bằng cách tự sao chép chính nó và lây lan giữa các máy tính khác nhau bằng ARPANET, một mạng kết nối các trường đại học và cơ sở nghiên cứu khác nhau ở Hoa Kỳ. Trên

thực tế, ARPANET về nhiều mặt là tiền thân của Internet. Creeper tự sao chép từ máy tính này sang máy tính tiếp theo, sau đó nó tự xóa khỏi máy tính đầu tiên cho phép nó di chuyển giữa các hệ thống. Sau đó vào năm 1972, "Reaper" được tạo ra với mục đích săn lùng và tiêu diệt Creeper, cũng sử dụng ARPANET. Creeper được tạo ra để xem liệu một chương trình có thể tự sao chép sang các hệ thống khác và không gây ra bất kỳ thiệt hại nào ngoại trừ việc hiển thị thông báo "I'M THE CREEPER. CATCH ME IF YOU CAN!" [5].

3.1.3 Một số loại Malware phổ biến

Trong thực tế có rất nhiều loại phần mềm độc hại với các chức năng và hành vi khác nhau trong đa dạng các tình huống khác nhau. Một số loại phần mềm độc hại phổ biến và nguy hiểm có thể kể đến như:

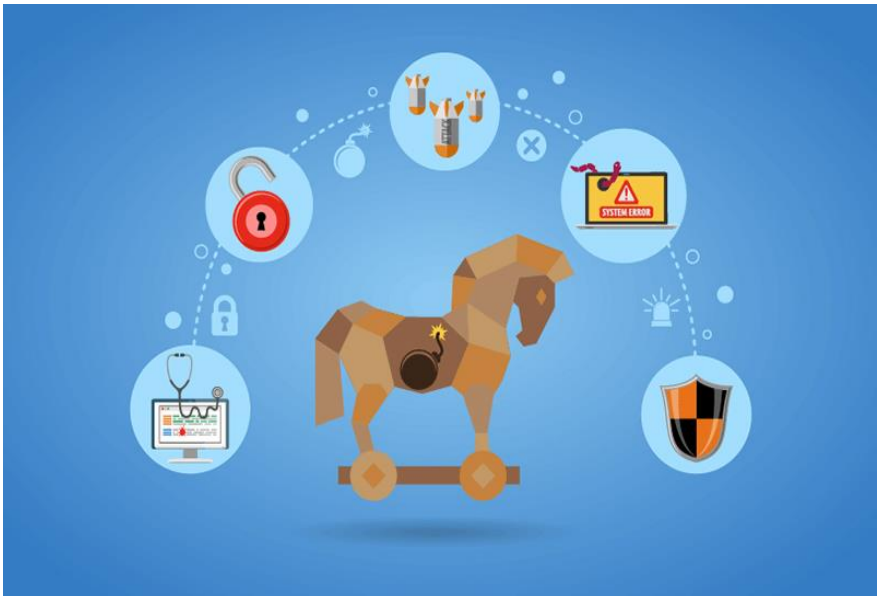
- Botnets (Robot network): Đây là một dạng Malware liên quan đến mạng của các máy tính bị nhiễm dưới sự kiểm soát của các bên tấn công bằng cách sử dụng máy chủ để ra lệnh. Đây thường là nguyên nhân dẫn đến các cuộc tấn công Distributed Denial of Service (DDoS).



Hình 3. 2 Botnets Malware [11]

- Cryptojacking: Là hoạt động khai thác tiền điện tử, đây là một quá trình tính toán một cách thông minh để xác định đúng các giao dịch ở trên Blockchain và kiếm tiền từ đó.

- Malvertising: Là từ ghép giữa “Malware + Advertising", mô tả hoạt động tim mã độc vào các quảng cáo trực tuyến trên các trang web và mạng quảng các trực tuyến hợp pháp.
- Virus: Cho phép các chương trình tự sao chép tất cả mọi thứ trên toàn bộ máy tính và Virus độc hại bám trên các chương trình hiện có để lây lan và thường chỉ được kích hoạt khi người đùng mở chương trình chứa Virus.
- Worm: Worm tự nhân bản khai thác lỗ hổng bảo mật để tự động lây lan trên máy tính và mạng. Không giống như Virus, Worm không đính kèm vào các chương trình hiện có hoặc thay đổi tệp. Chúng thường không được chú ý cho đến khi quá trình sao chép đạt đến quy mô tiêu tốn đáng kể tài nguyên hệ thống hoặc băng thông mạng.
- Trojan Malware: Phần mềm độc hại được ngụy trang dưới dạng phần mềm hợp pháp. Sau khi được kích hoạt, Trojan phần mềm độc hại sẽ thực hiện bất kỳ hành động nào mà chúng đã được lập trình để thực hiện. Không giống như virus và sâu, Trojan không sao chép hoặc sinh sản thông qua lây nhiễm. “Trojan” ám chỉ câu chuyện thần thoại về những người lính Hy Lạp ẩn giấu bên trong một con ngựa gỗ được trao cho thành Troy của kẻ thù.



Hình 3. 3 Trojan Malware [12]

- Ransomware: Là một mô hình mội phạm sử dụng Malware để lưu giữ các tệp, dữ liệu hoặc thông tin có giá trị để đòi tiền chuộc, nạn nhân của những cuộc tấn công Ransomware thường bị thiệt hại rất nghiêm trọng.
- Remote Administration Tools (RATs): Phần mềm cho phép người vận hành từ xa điều khiển hệ thống. Những công cụ này ban đầu được xây dựng để sử dụng hợp pháp nhưng hiện được các tác nhân đe dọa sử dụng. RAT cho phép kiểm soát quản trị, cho phép kẻ tấn công thực hiện hầu hết mọi thứ trên máy tính bị nhiễm [4].

3.1.4 Cơ chế hoạt động

Malware thường lây nhiễm vào máy bằng cách lừa người dùng nhấp vào link bất kì, thường là vào một liên kết hoặc cửa sổ bật lên. Mô tả có thể chứa nội dung khiêu khích như "Nhận giải thưởng của bạn" hoặc "Tài khoản của bạn đã bị xâm phạm. Vui lòng đăng nhập và xác minh các khoản phí gần đây" [6].

Ngoài ra, Malware có thể từ việc cài đặt một chương trình mà dùng không nên cài đặt từ Internet. Khi nhấp chuột hoặc cài đặt xảy ra, mã độc sẽ thực thi các hành động mà người dùng không lường trước hoặc không có ý định, có thể bao gồm:

- Tự sao chép trong các phần khác nhau của hệ thống tập tin.
- Cài đặt các ứng dụng chiếm đoạt tài nguyên hệ thống, thường chạy ngầm mà người dùng không hề hay biết, đồng thời làm chậm đáng kể hệ thống
- Chặn quyền truy cập vào tệp, chương trình hoặc thậm chí chính hệ thống, đôi khi buộc người dùng phải thanh toán để lấy lại quyền truy cập
- Phá vỡ các thành phần thiết yếu của hệ thống và khiến thiết bị không thể hoạt động.

3.2 Phương pháp phát hiện Malware

Hiểu được mức độ nguy hiểm và những thiệt hại do phần mềm độc hại (Malware) mang lại, nhiều phương pháp phát hiện phần mềm độc hại được đề xuất, những phương pháp phổ biến như:

- Phân tích chữ ký (Signature Analysis): Một trong những cách cổ điển để phát hiện phần mềm độc hại. Tập trung vào việc so sánh các chữ ký của các phần mềm độc hại với các chữ ký đã biết từ trước. Các chữ ký này thường được tạo ra từ các biểu diễn của mã máy hoặc các đặc điểm độc đáo của phần mềm độc hại. Khi một tệp tin mới xuất hiện, nó được so sánh với cơ sở dữ liệu các chữ ký đã biết. Nếu phần lớn các byte hoặc bit trong chữ ký của tệp tin mới trùng khớp với một trong số các chữ ký đã biết, thì tệp tin đó có thể được phân loại là phần mềm độc hại.
- Phân tích hành vi (Behavior Analysis): Tập trung vào việc quan sát và phân tích các hành vi của chương trình để xác định xem chúng có hành vi độc hại hay không. Thay vì dựa vào cấu trúc hoặc nội dung của tệp tin, phương pháp này quan tâm đến cách chương trình thực sự hoạt động. Các hành vi như tạo ra và sao chép các tệp, kết nối mạng không xác định, hoặc thay đổi các cài đặt hệ thống có thể là dấu hiệu của phần mềm độc hại.
- Phân tích đặc trưng (Feature Analysis): Tập trung vào việc phân tích các đặc trưng của phần mềm độc hại để xác định các mẫu hoặc đặc điểm độc hại. Các đặc trưng này có thể bao gồm các hàm băm, chuỗi ký tự độc hại, hoặc các quy trình hệ thống không bình thường. Bằng cách xác định và phân tích các đặc trưng này, các nhà nghiên cứu có thể tạo ra các mô hình hoặc bộ phân loại để nhận diện các trường hợp của phần mềm độc hại.
- Sử dụng học máy và học sâu: Các phương pháp này sử dụng thuật toán máy tính để tự động học và nhận diện các mẫu từ dữ liệu. Học máy và học sâu có thể phân loại các tệp tin hoặc quyết định xem một chương trình có tính độc hại hay không dựa trên các đặc trưng và dữ liệu huấn luyện. Cách tiếp cận này đã và đang trở nên phổ biến hơn trong việc phát hiện phần mềm độc hại do khả năng của chúng trong việc tự động hóa và cập nhật liên tục để chống lại các mối đe dọa mới.

Trong bài nghiên cứu cũng như dựa vào bài toán, chúng tôi sẽ sử dụng bốn thuật toán học máy (Machine Learning) khác nhau để xây dựng các mô hình phát hiện phần mềm độc hại và đánh giá kết quả, hiệu suất của những mô hình đó.

CHƯƠNG 4. PHƯƠNG PHÁP NGHIÊN CỨU

4.1 Mô hình đề xuất

Với bài toán phân loại nhị phân "Phát hiện phần mềm độc hại", ta sẽ có rất nhiều thuật toán khác nhau để phục vụ trong việc huấn luyện mô hình. Quá trình lựa chọn mô hình phù hợp phải dựa vào các tiêu chí như bộ dữ liệu và loại bài toán (phân loại, hồi quy, ...). Trong nghiên cứu của chúng tôi, chúng tôi đã quyết định chọn lọc các mô hình có hiệu suất cao với những tính chất khác nhau nhằm so sánh độ chính xác, phục vụ cho bài toán nghiên cứu. Các mô hình nghiên cứu được sử dụng: Decision Tree, Random Forest, Naive Bayes và Gradient Boosting.

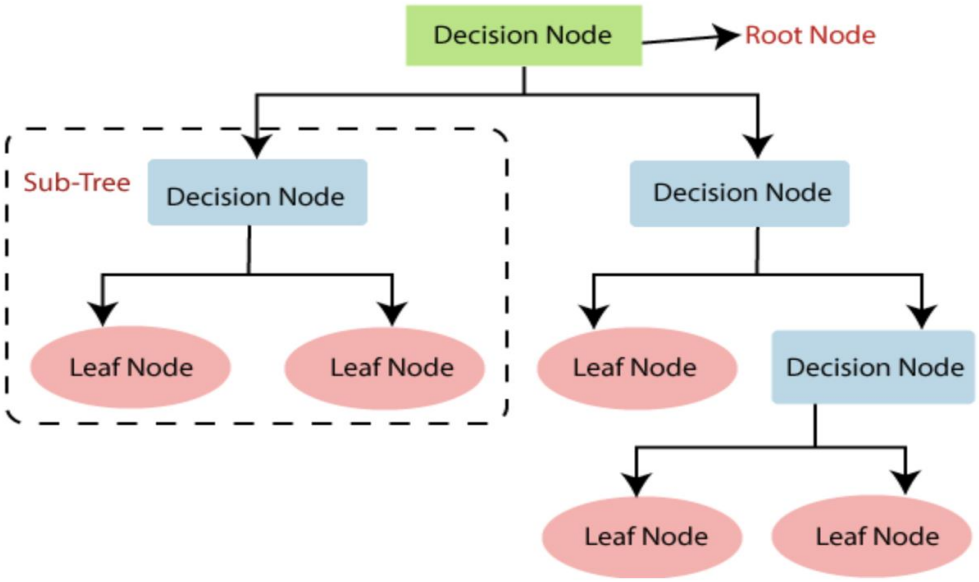
4.1.1 Thuật toán Decision Tree

4.1.1.1 Khái niệm

Decision Tree là một thuật toán trong Machine Learning (ML) phổ biến thuộc kỹ thuật học có giám sát. Decision Tree có thể được sử dụng cho cả các vấn đề phân loại và hồi quy, nhưng thường được ưa chuộng hơn cho việc giải quyết các vấn đề phân loại. Nó là một bộ phân loại được cấu trúc theo dạng cây, trong đó các Decision Node đại diện quyết định và các Leaf Node đại diện cho kết quả. Trong một cây quyết định, có hai loại node chính, là Decision Node và Leaf Node. Decision Node được sử dụng để ra quyết định và có nhiều nhánh, trong khi các Leaf Node là kết quả của các quyết định đó và không chứa thêm bất kỳ nhánh nào nữa [7].

4.1.1.2 Cơ chế hoạt động

Các quyết định được thực hiện dựa trên các đặc trưng của tập dữ liệu được cung cấp (Input). Đây là một biểu đồ biểu diễn minh hoạ để lấy tất cả các giải pháp có thể có cho một vấn đề hoặc quyết định dựa trên các dữ liệu đầu vào. Nó được gọi là Decision Tree (Cây quyết định) bởi vì, tương tự như một cây, nó bắt đầu với Root (nút gốc), mở rộng ra các nhánh khác và xây dựng một cấu trúc giống như cây. Một cây quyết định đơn giản chỉ đặt một câu hỏi và dựa vào câu trả lời (Có hoặc Không), nó tiếp tục phân chia cây thành các cây con.



Hình 4. 1 Cơ chế hoạt động của Decision Tree [7]

4.1.1.3 Ưu điểm và nhược điểm

Ưu điểm: Decision Tree có cấu trúc dễ hiểu và dễ giải thích giúp cho người dùng dễ dàng diễn giải quá trình ra quyết định. Bên cạnh đó, Decision Tree có khả năng xử lý cả dữ liệu số và dữ liệu phân loại mà không cần phải thực hiện các biến đổi phức tạp tạp bên ngoài.

Nhược điểm: Decision có thể dễ bị overfitting trên tập dữ liệu huấn luyện, đặc biệt là khi cây quá sâu hoặc khi có quá nhiều biến. Ngoài ra, những thay đổi nhỏ trong dữ liệu có thể dẫn đến sự thay đổi lớn trong cấu trúc của cây, làm cho cây quyết định không ổn định. Hơn nữa là Decision Tree có thể dễ bị ảnh hưởng bởi các biến không quan trọng trong quá trình xây dựng cây, làm cho cây không hiệu quả.

4.1.1.4 Một số ứng dụng của Decision Tree

Hệ thống khuyễn nghị: Decision Tree có thể được sử dụng để xây dựng các hệ thống khuyến nghị, giúp dự đoán các sản phẩm hoặc nội dung mà người dùng có thể quan tâm dựa trên lịch sử hoặc hành vi trước đó.

Chuẩn đoán y tế: Mô hình Decision Tree có thể giúp trong việc chẩn đoán các bệnh lý y tế dựa trên các dữ liệu như triệu chứng, kết quả xét nghiệm và hồ sơ bệnh án.

Dự đoán giá cổ phiếu: Cây quyết định có thể được sử dụng để dự đoán giá cổ phiếu hoặc thị trường tài chính dựa trên các yếu tố kinh tế và tài chính.

4.1.2 Thuật toán Random Forest

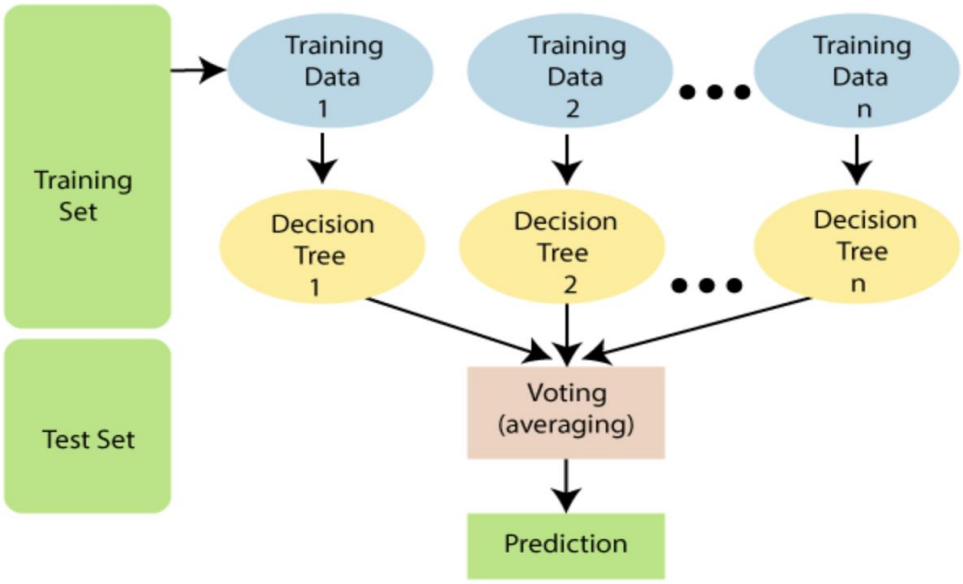
4.1.2.1 Khái niệm

Random Forest là một thuật toán Machine Learning (ML) phổ biến thuộc kỹ thuật học có giám sát. Nó có thể được sử dụng cho cả các vấn đề phân loại và hồi quy. Random Forest kết hợp nhiều bộ phân loại để giải quyết các vấn đề phức tạp và cải thiện hiệu suất của mô hình. Random Forest là một bộ phân loại bao gồm nhiều cây quyết định được xây dựng trên các tập con của dữ liệu đầu vào và kết hợp kết quả dự đoán từ mỗi tập con đó bằng Decision Tree để tạo ra dự đoán cuối cùng. Số lượng cây càng lớn thì sẽ dẫn đến độ chính xác cao hơn và ngăn chặn vấn đề overfitting [8].

4.1.2.2 Cơ chế hoạt động

Từ một bộ dữ liệu ban đầu sẽ được chia ra làm các tập dữ liệu con và mỗi tập dữ liệu con này sẽ được đưa qua một Decision Tree để thực hiện Training. Sau khi giai đoạn Training kết thúc thì mỗi Decision Tree sẽ cho ra một kết quả dự đoán và khi đó ta sẽ được một tập dữ liệu mới chứa các kết quả ta đã dự đoán trước đó.

Dự đoán cuối cùng được xác định bằng cách chọn ra kết quả từ tập dữ liệu mới, đối với việc phân loại thì dự đoán cuối cùng sẽ là dự đoán có kết quả giống nhau nhiều nhất, còn đối với hồi quy thì dự đoán cuối cùng sẽ là giá trị trung bình từ kết quả của các Decision Tree.



Hình 4. 2 Cơ chế hoạt động của Random Forest [8]

4.1.2.3 Ưu điểm và nhược điểm

Ưu điểm: Random Forest mất ít thời gian hơn để Training dữ liệu so với các thuật toán khác, bên cạnh đó còn dự đoán đầu ra với độ chính xác cao kể cả đối với tập dữ liệu lớn và vẫn duy trì độ chính xác mặc dù bị thiếu hụt một lượng lớn dữ liệu.

Nhược điểm: Mặc đù Random Forest có khả năng giảm overfitting so với Decision Tree, nhưng nó vẫn sẽ xảy ra overfitting khi tập dữ liệu quá nhỏ hoặc có quá nhiều đặc trưng. Ngoài ra, Random Forest cũng yêu cầu một lượng lớn bộ nhớ để lưu trữ các quyết định và phải thực hiện một số tính toán phức tạp khi xây dựng mô hình và dự đoán. Do việc tính toán dự đoán từ nhiều cây quyết định khác nhau, điều này có thể làm cho việc diễn giải dự đoán của mô hình trở nên khó khăn, đặc biệt đối với người không chuyên về machine learning.

4.1.2.4 Một số ứng dụng của Random Forest

Phân loại ảnh và nhận dạng: Random Forest có thể được sử dụng để phân loại ảnh và nhận dạng đối tượng trong các ứng dụng như nhận dạng khuôn mặt, phát hiện vật thể, hay phân loại hình ảnh y tế.

Dự đoán nhu cầu và nguồn cung: Trong ngành sản xuất và quản lý chuỗi cung ứng, Random Forest có thể được sử dụng để dự đoán nhu cầu và nguồn cung, giúp tối ưu hóa quản lý hàng tồn kho và lịch trình sản xuất.

Phát hiện gian lận: Random Forest có thể được áp dụng để phát hiện gian lận trong các giao dịch tài chính hoặc giao dịch trực tuyến bằng cách phân loại các hành vi không bình thường.

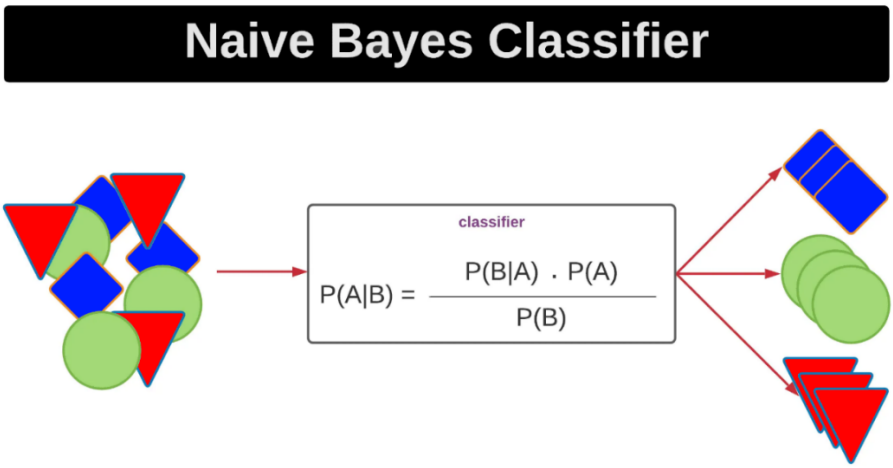
4.1.3 Thuật toán Naïve Bayes

4.1.3.1 Khái niệm

Naive Bayes là một thuật toán trong Machine Learning (ML) phổ biến thuộc kỹ thuật học có giám sát, dựa trên định lý Bayes và được sử dụng để giải các bài toán phân loại. Naive Bayes chủ yếu được sử dụng trong phân loại văn bản trong một tập dữ liệu đa chiều, đây là một trong những thuật toán phân loại đơn giản và hiệu quả nhất giúp xây dựng các mô hình học máy nhanh để đưa ra dự đoán nhanh chóng. Naive Bayes là một thuật toán phân loại dự đoán dựa trên xác suất của một đối tượng và một số ví dụ về thuật toán Naive Bayes có thể kể đến như lọc thư rác, phân loại bài viết, phân loại email spam [9].

4.1.3.2 Cơ chế hoạt động

Xây dựng mô hình: Naive Bayes giả định rằng các đặc trưng của mỗi mẫu dữ liệu là độc lập với nhau, nghĩa là giá trị của mỗi đặc trưng không phụ thuộc vào giá trị của các đặc trưng khác trong mẫu. Dựa trên giả định này, Naive Bayes sử dụng định lý Bayes để tính xác suất một mẫu thuộc về mỗi lớp dựa trên xác suất của các đặc trưng trong mẫu sau đó chọn lớp có xác suất cao nhất làm dự đoán cho mẫu dữ liệu mới.



Hình 4. 3 Cơ chế hoạt động của Naïve Bayes [13]

4.1.3.3 Công thức nguyên lý Bayes

Naive Bayes tính xác suất mẫu dữ liệu mới thuộc về một lớp bằng cách sử dụng định lý Bayes:

$$P(C_k|x) = \frac{P(x|C_k) \cdot P(C_k)}{P(x)}$$

Trong đó:

- $P(C_k|x)$: Là xác suất mẫu thuộc về lớp C_k dựa trên các đặc trưng x.
- $P(x|C_k)$: Là xác suất của mẫu dữ liệu x nếu biết rằng nó thuộc về lớp C_k.
- $P(C_k)$: Là xác suất tiền nghiệm của lớp C_k, thường được tính toán từ tần suất của lớp C_k trong tập dữ liệu được Training.
- P(x): Là xác suất của mẫu dữ liệu x trên tất cả các lớp.

4.1.3.4 Ưu điểm và nhược điểm

Ưu điểm: Là một trong những thuật toán Machine Learning (ML) nhanh chóng và dễ dàng triển khai để dự đoán một lớp tập dữ liệu lớn với hiệu suất tốt. Naive Bayes có thể sử dụng cho phân loại nhị phân và phân loại đa lớp và có thể xử lý các giá trị bị thiếu trong dữ liệu mà không cần phải loại bỏ hoặc điền vào các giá trị bị thiếu.

Nhược điểm: Naive Bayes cho rằng tất cả các đặc trưng độc lập với nhau, điều này thường không phù hợp với thực tế và có thể dẫn đến các dự đoán không chính xác. Ngoài ra, nó cũng gặp khó khăn trong việc xử lý các biến liên tục hoặc có phân phối không chuẩn và dễ bị ảnh hưởng bởi dữ liệu không cân bằng khi một lớp có số lượng mẫu lớn hơn so với lớp khác.

4.1.3.5 Một số ứng dụng của Naïve Bayes

Phân loại văn bản: Naive Bayes được sử dụng để phân loại các tài liệu văn bản vào các chủ đề khác nhau như thể thao, chính trị, khoa học, văn hóa. Ví dụ điển hình là phân loại email là spam hay không spam.

Nhận diện khuôn mặt: Naive Bayes là một trong những phương pháp phổ biến nhất được sử dụng để nhận diện khuôn mặt bằng cách phân loại các gương mặt đã được quét trước đó.

Phân loại y tế: Naive Bayes có thể được sử dụng để phân loại dữ liệu y tế, ví dụ như dự đoán liệu một bệnh nhân có mắc bệnh hay không dựa trên các chỉ số y tế và triệu chứng.

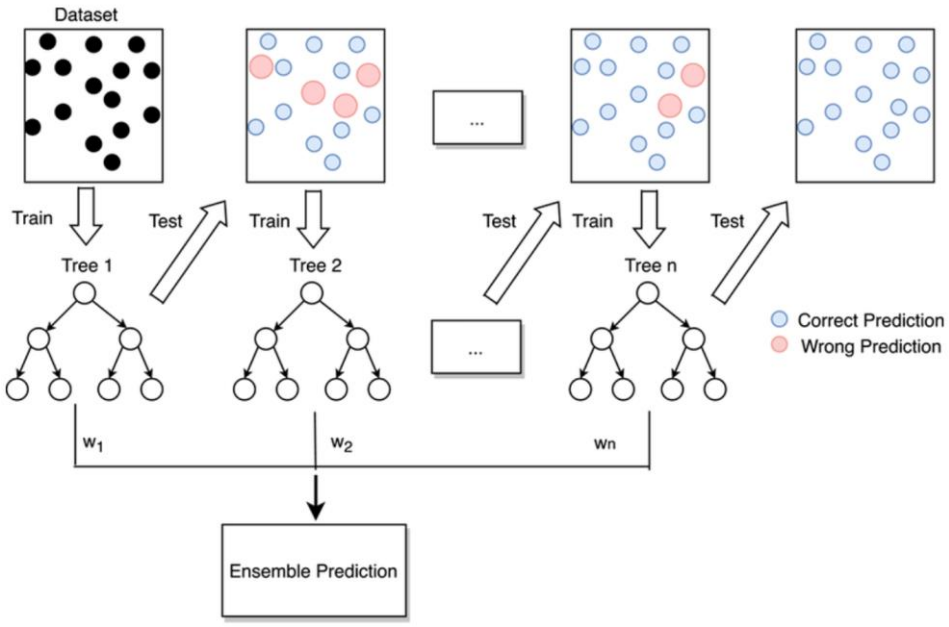
4.1.4 Thuật toán Gradient Boosting

4.1.4.1 Khái niệm

Gradient Boosting là một thuật toán máy học trong đó các mô hình dự đoán được xây dựng tuần tự, mỗi mô hình cố gắng cải thiện và điều chỉnh dự đoán của mô hình trước đó. Ý tưởng cơ bản của Gradient Boosting là sử dụng một chuỗi các mô hình yếu (weak learners) - thường là cây quyết định đơn giản - và tối ưu hóa các dự đoán tuần tự dựa trên các sai số của các mô hình trước đó. Quá trình này được thực hiện bằng cách sử dụng Gradient Descent để điều chỉnh các tham số của mô hình để giảm bias. Gradient Boosting thường được sử dụng cho cả các bài toán phân loại và hồi quy và là một trong những phương pháp Ensemble Learning phổ biến nhất trong machine learning. Thường được áp dụng cho các model có variance thấp và bias cao [10] [15].

4.1.4.2 Cơ chế hoạt động

Gradient Boosting là một mô hình Machien Learning (ML) mà trong đó bao gồm một lượng lớn các mô hình khác (thường là những mô hình có cùng cấu trúc). Mỗi mô hình sau sẽ học cách sửa những errors (dữ liệu mà mô hình trước dự đoán sai) của mô hình trước đó. Từ đó tạo thành một chuỗi các mô hình mà mô hình sau sẽ tốt hơn mô hình trước vì trọng số qua mỗi mô hình sẽ được update và kết quả của mô hình cuối cùng trong chuỗi này sẽ là kết quả trả về.



Hình 4. 4 Cơ chế hoạt động của Gradient Boosting [14]

4.1.4.3 Ưu điểm và nhược điểm

Ưu điểm: Grandient Boosting tuân theo phương pháp tổng hợp cho phép mô hình đưa ra dự đoán chính xác nhất mà không có kết quả nào khác. Không yêu cầu xử lý trước dữ liệu vì Gradient Boosting phù hợp cho cả biến số, biến phân loại và nó tự động xử lý các dữ liệu bị thiếu.

Nhược điểm: Do quá trình tuần tự tối ưu các mô hình yếu, Gradient Boosting có nguy cơ cao bị overfitting đối với các tập dữ liệu nhỏ hoặc có nhiễu. Còn đối với những tập dữ liệu lớn thì thường sẽ tốn nhiều tài nguyên tính toán vì yêu cầu khá nhiều câu quyết định và thời gian Training cũng lâu hơn so với các thuật toán khác.

4.1.4.4 Một số ứng dụng của Gradient Boosting

Xử lý dữ liệu lớn: Gradient Boosting cũng được sử dụng để xử lý dữ liệu lớn trong các hệ thống Big Data, nơi mà việc dự đoán và phân tích dữ liệu là quan trọng, bao gồm việc phát hiện và dự đoán thói quen người dùng, quản lý rủi ro tài chính, hoặc dự đoán nhu cầu người dùng.

Dự đoán và xử lý tín hiệu: Trong các ứng dụng xử lý tín hiệu và thị giác máy tính, Gradient Boosting có thể được sử dụng để dự đoán và phân loại tín hiệu âm thanh, hình ảnh, hoặc video.

Dự đoán chuỗi thời gian: Trong các bài toán dự đoán chuỗi thời gian, như dự báo doanh số bán hàng hoặc lưu lượng truy cập trang web, Gradient Boosting có thể được sử dụng để tạo ra dự đoán chính xác.

4.2 Kỹ thuật đề xuất

4.2.1 Kỹ thuật Scale

Trong tiền xử lý dữ liệu, Scaling là một kỹ thuật được sử dụng để điều chỉnh các giá trị của các đặc trưng sao cho chúng nằm trong một phạm vi nhất định. Việc này giúp tối ưu hóa hiệu suất của các mô hình học máy bằng cách đồng bộ hóa các đặc trưng và giảm biến động của dữ liệu.

MinMax Scaling là một phương pháp đặc biệt của scaling, trong đó các giá trị của các đặc trưng được chuyển đổi sao cho chúng thuộc vào một phạm vi cố định, thường là từ 0 đến 1. Quá trình này làm cho các đặc trưng có cùng phạm vi giá trị, giúp làm giảm biến động và đồng bộ hóa dữ liệu, tạo điều kiện thuận lợi cho quá trình huấn luyện mô hình.

4.2.2 Kỹ thuật Grid Search

Kỹ thuật Grid Search là một phương pháp quan trọng trong điều chỉnh các siêu tham số của các mô hình học máy. Nó hoạt động bằng cách xác định một lưới các giá trị cho mỗi siêu tham số và thử nghiệm tất cả các tổ hợp có thể từ các giá trị này. Quá

trình này giúp tìm ra tổ hợp siêu tham số tối ưu nhất cho mô hình, dựa trên một phép đo hiệu suất nhất định như độ chính xác (Accuracy) hoặc F1-score.

Công thức được sử dụng để tính kết quả trong GridSearchCV phụ thuộc vào phương pháp cross-validation. Trong đó, GridSearchCV sử dụng phương pháp cross-validation k-fold, tập dữ liệu được chia thành k phần (folds), mỗi phần sẽ lần lượt được sử dụng làm tập kiểm tra trong khi các phần còn lại được sử dụng làm tập huấn luyện. Kết quả cuối cùng là trung bình của các điểm số đánh giá trên các fold kiểm tra.

Trong bài nghiên cứu này, giá trị tham số của thuật toán sẽ được chọn dựa trên điểm số đánh giá (Scoring) được chỉ định trong GridSearchCV (trong nghiên cứu chúng tôi sử dụng 'Accuracy' làm Scoring) nhằm đánh giá hiệu suất của mô hình trên mỗi fold kiểm tra.

Mặt khác, kỹ thuật Grid Search sẽ tốn rất nhiều thời gian tính toán, đặc biệt là với các mô hình phức tạp và tập dữ liệu lớn và khả năng cho ra tham số phù hợp với mô hình có thể không được tối ưu. Điều này là do Grid Search phải thử nghiệm mọi tổ hợp có thể từ các giá trị được chỉ định, và số lượng tổ hợp có thể tăng lên một cách nhanh chóng với số lượng siêu tham số và các giá trị thử nghiệm.

4.3 Phương pháp đánh giá

Trong bài nghiên cứu này, chúng tôi quyết định thực hiện bài toán phân loại là dựa trên mục tiêu của nghiên cứu và tính chất của dữ liệu. Việc lựa chọn bài toán phân loại được đặt ra nhằm mục đích phân biệt và phân loại các mẫu dữ liệu thành hai nhóm: phần mềm độc hại và phần mềm không độc hại [16].

Và khi thực hiện bài toán phân loại, có 4 trường hợp của dự đoán có thể xảy ra [17]:

- True Positive (TP): đối tượng ở lớp Positive, mô hình phân đối tượng vào lớp Positive (dự đoán đúng).
- True Negative (TN): đối tượng ở lớp Negative, mô hình phân đối tượng vào lớp Negative (dự đoán đúng).
- False Positive (FP): đối tượng ở lớp Negative, mô hình phân đối tượng vào lớp Positive (dự đoán sai) – Type I Error.

- False Negative (FN): đối tượng ở lớp Positive, mô hình phân đối tượng vào lớp Negative (dự đoán sai) – Type II Error.

Trong bài nghiên cứu này, chúng tôi đã sử dụng 5 tiêu chí để đánh giá hiệu suất của mô hình như sau:

- Confusion matrix (Ma trận nhầm lẫn): Là một công cụ được sử dụng để đánh giá hiệu suất, thể hiện độ chính xác của các mô hình phân loại và được biểu thị dưới dạng ma trận. Confusion Matrix giúp phân tích các loại lỗi mà mô hình thực hiện và đánh giá hiệu suất của nó dựa trên các tiêu chí như Accuracy, Precision, Recall và F1-score.
- Accuracy: Là tỷ lệ phần trăm dự đoán đúng cho dữ liệu thử nghiệm. Nó có thể được tính toán dễ dàng bằng cách chia số lần dự đoán đúng cho tổng số lần dự đoán.

$$\text{Accuracy} = \frac{\text{Correct Prediction}}{\text{All Prediction}}$$

- Precision: Là tiêu chi đo lường tỷ lệ các dự đoán True Positives so với tỷ lệ tổng số dự đoán thuộc nhóm positives.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- Recall: Là tiêu chí đo lường tỷ lệ của các trường hợp True Positives thực sự mà mô hình phân loại đúng so với tổng số các trường hợp True Positives có thực sự tồn tại trong dữ liệu.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

- F1-Score: F1-score là một tiêu chí đánh giá hiệu suất của mô hình phân loại trong machine learning, tổng hợp giữa precision và recall, thể hiện mức độ cân bằng giữa precision và recall. F1-score cao khi cả precision và recall đều cao, và nó đạt giá trị cao nhất khi precision và recall đều đạt giá trị tối đa.

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

CHƯƠNG 5. THỰC NGHIỆM

5.1 Dữ liệu thực nghiệm

Bộ dữ liệu mà chúng tôi nghiên cứu đã được thu thập trên nền tảng Kaggle cũng như các bài báo khoa học. Bộ dữ liệu bao gồm 35 đặc trưng (đã bao gồm cột mục tiêu) cùng với 100.000 bộ mẫu (trong đó có 50.000 bộ mẫu là phần mềm độc hại và 50.000 bộ mẫu là phần mềm sạch) hỗ trợ cho việc huấn luyện mô hình. Bộ dữ liệu được thu thập và xây dựng thông quan lưu lượng mạng của máy ảo trên nền tảng Unix/Linux. Bộ dữ liệu được tạo ra nhằm mục đích phát hiện và phân loại phần mềm độc hại. Bảng thông tin sau bao gồm tên các đặc trưng cũng như mô tả và kiểu dữ liệu của đặc trưng đó.

Bảng 5. 1 Bảng đặc tả dữ liệu thực nghiệm

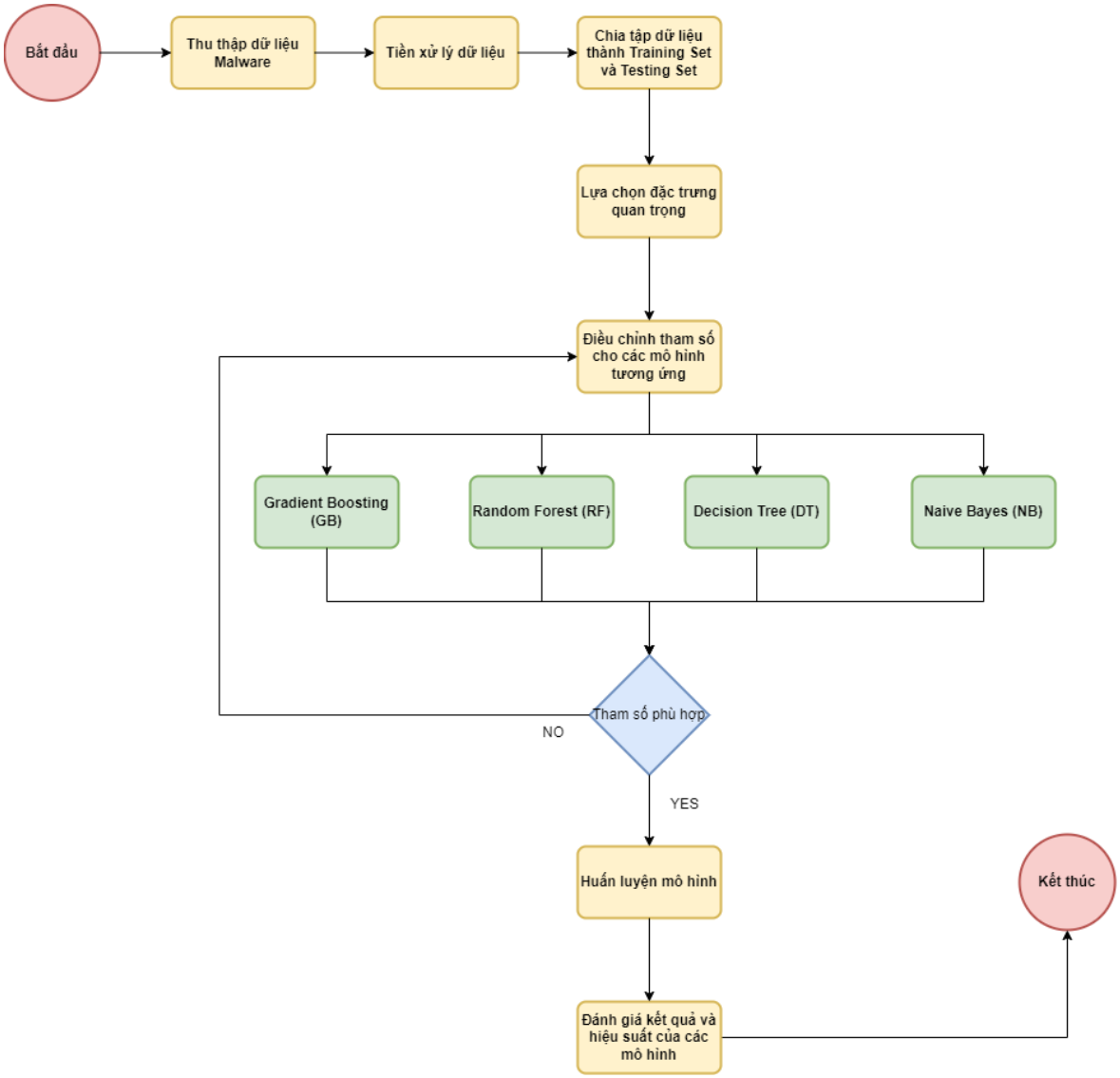
STT	Tên đặc trưng	Kiểu dữ liệu	Mô tả
1	hash	Text	APK / SHA256 định dạng file cụ thể, tên của một file.
2	Millisecond/ number	Number	Thời gian khi mẫu được thu thập hoặc phân tích.
3	Classification	Text	Phân loại của mẫu là malware hay benign.
4	State	Number	Trạng thái của các tác vụ: 0 - Không thể chạy được 1 - Có thể chạy được 2 - Đã dừng
5	usage_counter	Number	Bộ đếm sử dụng của cấu trúc nhiệm vụ, đếm số lần một cấu trúc nhiệm vụ cụ thể được sử dụng.
6	prio	Number	Mức độ ưu tiên của một tác vụ.
7	Static_prio	Number	Mức độ ưu tiên tĩnh của một tác vụ.

8	normal_prio	Number	Mức độ ưu tiên thông thường (không tính đến tính kế thừa RT).
9	Policy	Number	Chính sách lập kế hoạch nhiệm vụ, xác định cách nhiệm vụ được lập kế hoạch để thực thi.
10	vm_pgoff	Number	Độ lệch của vùng trong tệp (tính bằng trang).
11	vm_truncate_count	Number	Được sử dụng để đánh dấu một VMA đã được xử lý.
12	task_size	Number	Kích thước hiện tại của nhiệm vụ được tính bằng byte.
13	cached_hole_size	Number	Kích thước của các khoảng trống (holes) trong không gian địa chỉ tự do (free address space) của hệ thống.
14	free_area_cache	Number	Địa chỉ đầu tiên của khoảng trống bộ nhớ.
15	mm_users	Number	Phân loại người dùng hoặc không gian dữ liệu trong môi trường hệ thống để phát hiện ra các hoạt động đáng ngờ hoặc malware.
16	map_count	Number	Số vùng nhớ.
17	hiwater_rss	Number	Độ lớn nhất của tập bộ nhớ đang được sử dụng.
18	total_vm	Number	Tổng số trang trong không gian ảo.
19	shared_vm	Number	Số trang được chia sẻ trong không gian ảo.
20	exec_vm	Number	Số trang thực thi trong không gian.
21	reserved_vm	Number	Số trang dùng riêng.
22	nr_ptes	Number	Số mục trong bảng trang.
23	end_data	Number	Địa chỉ cuối của thành phần mã.

24	last_interval	Number	Thời gian giữa các sự kiện cuối cùng trước khi hệ thống bắt đầu bị "thrashing" (hiện tượng quá tải và không hiệu quả).
25	Nvcsw	Number	Đo lường số lượng các chuyển đổi bối cảnh (context switches) tự nguyện trong hệ thống, đánh giá sự hoạt động bất thường của một tiến trình hoặc ứng dụng.
26	Nivcsw	Number	Đo lường số lượng các chuyển đổi bối cảnh (context switches) không tự nguyện trong hệ thống.
27	min_flt	Number	Lỗi nhỏ (lỗi trang).
28	maj_flt	Number	Lỗi chính (lỗi trang).
29	fs_excl_counter	Number	Số lượng tài nguyên độc quyền của hệ thống tập tin.
30	Lock	Number	Khóa đồng bộ hóa đọc-ghi được sử dụng để truy cập hệ thống tệp.
31	Utime	Number	Đo lường thời gian mà người dùng (user) đã truy cập hoặc sử dụng trong quá trình thực hiện một hoạt động liên quan đến việc phát hiện malware.
32	Stime	Number	Thời gian hệ thống.
33	Gtime	Number	Thời gian mà một "guest" (một người dùng không được phép hoặc không được xác nhận) tiêu thụ hoặc tương tác với một hệ thống hay dịch vụ cụ thể.
34	Cgtime	Number	Thời gian tích lũy của nhóm tiến trình.
35	signal_nvcsw	Number	Chỉ số theo dõi sự tiêu tốn tài nguyên của một chương trình độc hại trên hệ thống.

5.2 Cài đặt thực nghiệm

Chúng tôi đã đề xuất quy trình xây dựng mô hình phát hiện phần mềm độc hại như sau:



Hình 5. 1 Quy trình xây dựng mô hình phát hiện Malware

5.2.1 Cài đặt môi trường

Mô hình phát hiện phần mềm độc hại được chúng tôi huấn luyện trên ngôn ngữ Python, hiển thị kết quả và source code ở định dạng Notebook (Jupyter) và file đuôi .py, sử dụng cấu hình máy với các thông số quan trọng như sau:

- RAM: 16GB.
- GPU: NVIDIA Geforce RTX 3050.
- CPU: 11th Gen Intel(R) Core(TM) i5-11400H @ 2.70GHz.

5.2.2 Tiền xử lý

Trong bài nghiên cứu này, các bước tiền xử lý cho tập dữ liệu Malware_dataset.csv được thực hiện như sau:

- Loại bỏ cột 'hash' từ tập dữ liệu Malware_dataset.csv (đây là cột chứa đường dẫn hoặc tên file mẫu).
- Chuẩn hóa các giá trị trong cột mục tiêu 'classification' với hai giá trị: 1 cho giá trị 'malware' (phần mềm độc hại) và 0 cho giá trị 'benign' (phần mềm sạch).
- Lưu vào tập X bộ dữ liệu gốc (trừ cột 'hash' đã được loại bỏ và cột 'classification' là cột mục tiêu).
- Lưu vào tập Y cột giá trị mục tiêu 'classification'.
- Thực hiện scaling dữ liệu theo chuẩn MinMaxScaler() sau khi chia tập dữ liệu thành Training set và Testing set.

5.2.3 Chia tập dữ liệu

- Thực hiện chia tập dữ liệu thành 2 tập khác nhau: Training Set và Tesing Set theo tỉ lệ 7:3 (70% để huấn luyện và 30% để kiểm tra).
- Gọi tập huấn luyện (training set) là X_train, Y_train. Tập kiểm tra (testing set) là X_test, Y_test.
- Tiến hành scaling các tập dữ liệu X_train và X_test.

5.2.4 Chọn đặc trưng

Đối với bộ dữ liệu phần mềm độc hại được sử dụng để nghiên cứu, có tổng cộng 35 đặc trưng (bao gồm đặc trưng mục tiêu). Chúng tôi đã sử dụng thuật toán Random Forest để tìm đặc trưng quan trọng dựa trên trọng số quan trọng của từng đặc trưng được tính bằng 'rf_feature.feature_importances_', một thuộc tính của mô hình Random Forest sau khi huấn luyện.

Trọng số quan trọng của mỗi đặc trưng để tính bằng công thức sau:

$$\text{Trọng số quan trọng của đặc trưng } X_i = \frac{\text{số lần đặc trưng } X_i \text{ được sử dụng để chia nhánh}}{\text{Tổng số lần chia nhánh trong mô hình}}$$

Trong đó:

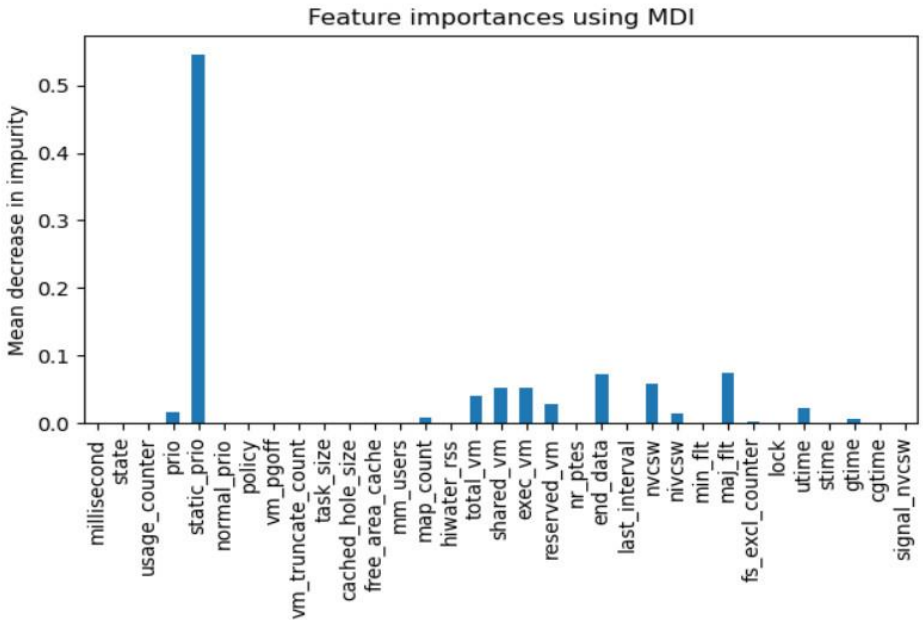
- Xi là đặc trưng thứ i.
- 'Số lần đặc trưng Xi được sử dụng để chia nhánh' là số lần mà nó được chọn để phân tách các nút trong cây quyết định.
- 'Tổng số lần chia nhánh trong mô hình' là tổng số lần chia nhánh trên tất cả các cây trong rừng ngẫu nhiên.

Chúng tôi đã trực quan hóa các trọng số quan trọng của từng đặc trưng thông qua Bảng 5.2 và Hình 5.2 như sau:

Bảng 5. 2 Trọng số quan trọng của các đặc trưng

Tên đặc trưng	**Trọng số quan trọng**
static_prio	0.5462092434063818
maj_flt	0.08359324907273627
nvcsw	0.05922532604739088
shared_vm	0.05791160620873195
end_data	0.05773571084245897
exec_vm	0.05220151808368746
total_vm	0.04028252646692509
reserved_vm	0.028336923122481466
utime	0.02293546135564136
prio	0.01710251249911049
nivcsw	0.013743951707752845
map_count	0.008131606856473975
gtime	0.007129926573336206
fs_excl_counter	0.0026202860571698824
mm_users	0.0009089972662497548
vm_truncate_count	0.0008556507363987487

free_area_cache	0.0007089530087697356
millisecond	0.00019603233998661098
stime	0.00013433857888790925
last_interval	3.191926361842293e-05
min_flt	4.260505810303697e-06
state	0.0
usage_counter	0.0
normal_prio	0.0
policy	0.0
vm_pgoff	0.0
task_size	0.0
cached_hole_size	0.0
hiwater_rss	0.0
nr_ptes	0.0
lock	0.0
cgtime	0.0
signal_nvcsw	0.0



Hình 5. 2 Biểu đồ trực quan các trọng số quan trọng của các đặc trưng

Các đặc trưng sẽ được so sánh trên thang đo của Mean Decrease Impurity, đo lường mức độ giảm sự đa dạng (Impurity) trung bình của dữ liệu khi mỗi đặc trưng được sử dụng để phân chia nút trong các cây quyết định. Có nghĩa là mỗi khi một đặc trưng được chọn để phân chia dữ liệu, sự đa dạng (Impurity) của dữ liệu giảm đi một lượng nhất định, và Mean Decrease Impurity đo lường mức độ giảm này trung bình qua tất cả các phân chia trong tất cả các cây quyết định.

Hay nói cách khác các đặc trưng với giá trị Mean Decrease Impurity cao hơn đóng góp nhiều hơn vào quá trình giảm sự đa dạng của dữ liệu khi huấn luyện mô hình Random Forest.

Những đặc trưng có giá trị Mean decrease impurity cao được coi là quan trọng hơn trong việc phân loại bộ dữ liệu.

Trong bài nghiên cứu, chúng tôi sử dụng kỹ thuật non-zero để trích xuất đặc trưng từ trọng số quan trọng của chúng, phương pháp này sẽ chọn lọc các đặc trưng có trọng số quan trọng khác 0, phương pháp được sử dụng khi trong bộ dữ liệu có nhiều đặc trưng có trọng số thấp. Các đặc trưng mà chúng tôi chọn lọc để nghiên cứu dựa trên kết quả tính toán là: 'static_prio', 'end_data', 'shared_vm', 'nvcsw', 'exec_vm', 'maj_flt',

'total_vm', 'reserved_vm', 'utime', 'prio', 'nivcsw', 'map_count', 'gtime', 'fs_excl_counter', 'mm_users', 'free_area_cache', 'vm_truncate_count', 'millisecond', 'stime', 'last_interval', 'min_flt'.

5.2.5 Huấn luyện mô hình

Trong nghiên cứu này, chúng tôi đã áp dụng kỹ thuật Grid Search cho thuật toán Gradient Boosting nhằm tối ưu hóa tham số cho mô hình này, với mục tiêu cải thiện hiệu suất và giảm thiểu nguy cơ under-fitting và over-fitting. Những mô hình còn lại sẽ được điều chỉnh tham số thông qua kết quả phát hiện trong quá trình thực nghiệm để tìm ra tham số tốt nhất.

Các tham số của các mô hình mà chúng tôi đã điều chỉnh bằng việc áp dụng kỹ thuật Grid Search được thể hiện tốt ở thuật toán Gradient Boosting như sau:

- Gradient Boosting (GB): Mô hình Gradient Boosting được cấu hình với learning_rate là 0.1, độ sâu tối đa là 3 (max_depth=3), số lượng mẫu tối thiểu cần thiết để phân chia một nút là 2 (min_samples_split=2), và n_estimators là 150. Điều này cung cấp một mô hình mạnh mẽ với khả năng thích ứng cao và hiệu suất dự đoán ổn định.

Tuy nhiên, một vài mô hình mà kỹ thuật Grid Search sẽ không mang lại hiệu quả tốt nhất. Vì thế chúng ta sẽ điều chỉnh các tham số cho các thuật toán Decision Tree, Random Forests và Naive Bayes như sau:

- Decision Trees (DT): Mô hình cây quyết định được đặt độ sâu tối đa là 2 (max_depth=2), với số lượng mẫu tối thiểu tại mỗi nút lá là 1 (min_samples_leaf=1) và số lượng mẫu tối thiểu cần thiết để phân chia một nút là 2 (min_samples_split=2). Các tham số được điều chỉnh sẽ có cấu trúc đơn giản nhưng vẫn hiệu quả, tránh hiện tượng over-fitting và tối ưu hóa khả năng dự đoán của mô hình.
- Random Forests (RF): Mô hình Random Forests được điều chỉnh với độ sâu tối đa là 2 (max_depth=2), số lượng mẫu tối thiểu cần thiết để phân chia một nút là 2 (min_samples_split=2), số lượng cây trong rừng được thiết lập là 10

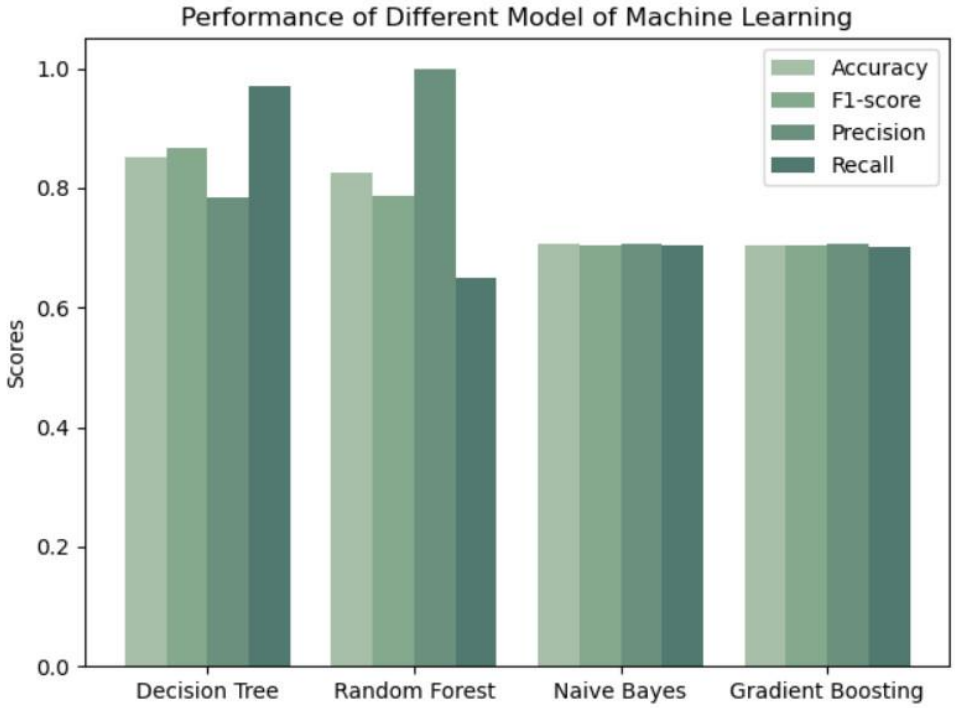
(n_estimators=10), và random_state là 42 để đảm bảo tính nhất quán trong kết quả. Các tham số được điều chỉnh sẽ làm tối ưu hóa hiệu suất của mô hình Random Forests.

- Naive Bayes (NB): Tham số alpha được điều chỉnh ở mức 0.1, áp dụng làm mịn Laplace/Lidstone để giảm thiểu rủi ro về hiện tượng under-fitting, và fit_prior được thiết lập là False (mô hình không sử dụng xác suất trước của các lớp để dự đoán).

Dựa vào các tham số đã được điều chỉnh của từng mô hình cũng như kết quả của các tham số trong quá trình thực nghiệm, những tham số này sẽ phù hợp với bài toán phân loại mà chúng tôi đang nghiên cứu, giúp cho các mô hình tuy có cấu trúc đơn giản nhưng vẫn đạt được hiệu quả cao, tối ưu và hạn chế các hiện tượng over-fitting và under-fitting. Từ đây, chúng tôi sẽ tiến hành huấn luyện mô hình và dự đoán biến mục tiêu cho dữ liệu kiểm thử bằng cách sử dụng các tham số đã được điều chỉnh. Dựa vào kết quả đó để đánh giá và so sánh từng mô hình dựa trên các tiêu chí Confusion Matrix, Accuracy, Precision, Recall và F1-score.

5.3 Kết quả thực nghiệm và đánh giá

5.3.1 Trực quan hóa kết quả thực nghiệm



Hình 5. 3 Biểu đồ trực quan kết quả thực nghiệm

5.3.2 Đánh giá hiệu suất

Bảng 5. 3 Kết quả thực nghiệm tổng quát

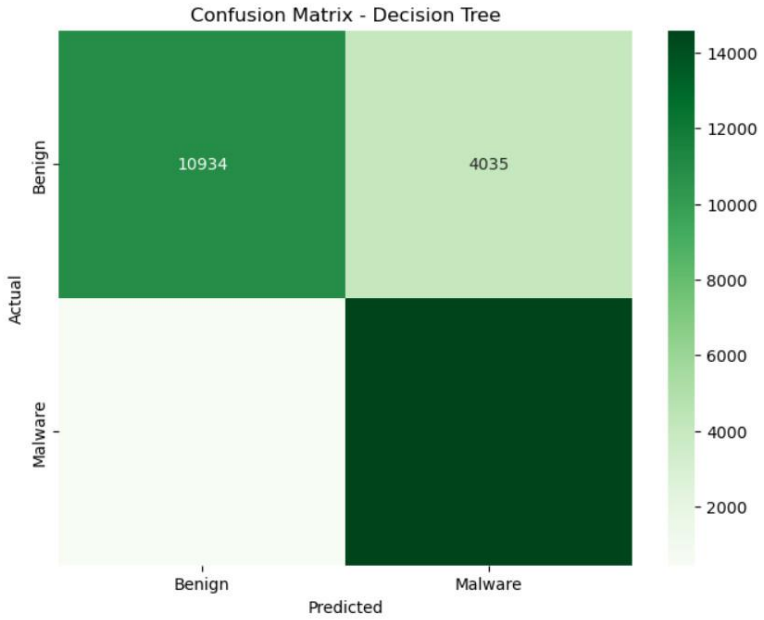
	Accuracy	Precision	Recall	F1-score
Decision Tree	85.07%	78.33%	97.04%	86.69%
Random Forest	82.43%	99.95%	64.95%	78.74%
Naïve Bayes	70.51%	70.70%	70.24%	70.47%
Gradient Boostring	70.53%	70.71%	70.28%	70.49%

Bảng 5. 4 Confusion Matrix của các mô hình

	True Negative (TN)	False Positive (FP)	False Negative (FN)	True Positive (TP)
Decision Tree	10934	4035	444	14587
Random Forest	14965	4	5267	9764
Naïve Bayes	10595	4374	4473	10558

Gradient Boosting	10595	4374	4467	10564

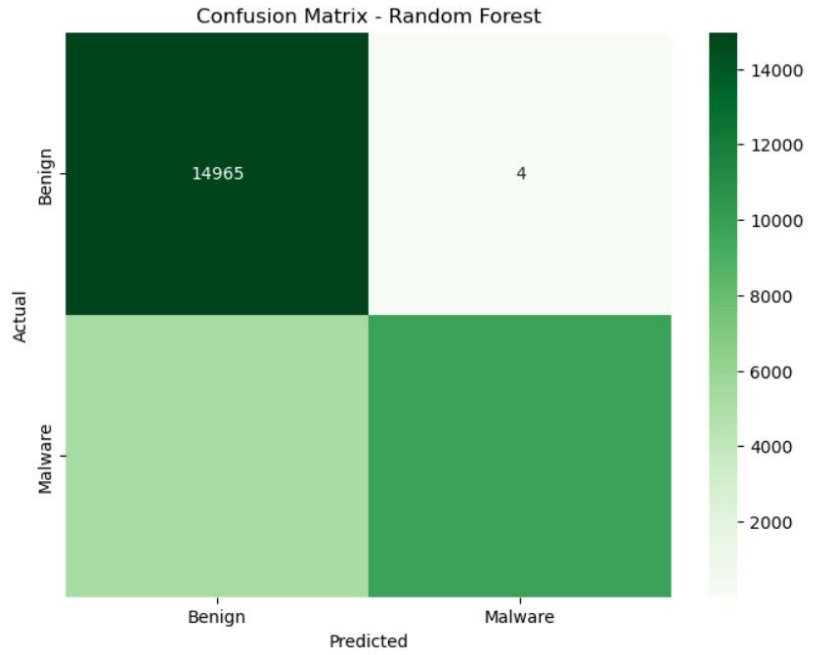
Dựa trên kết quả huấn luyện của các mô hình: Decision Tree, Random Forest, Naïve Bayes và Gradient Boosting, chúng tôi đã đánh giá hiệu suất của các mô hình như sau:



Hình 5. 4 Confusion Matrix của mô hình Decision Tree

- Mô hình Decision Tree có một lượng lớn True Positive (TP - 14587) và True Negative (TN - 10934), đồng thời có số lượng False Positive (FP - 4035) và False Negative (FN - 444) thấp. Điều này cho thấy mô hình Decision Tree có khả năng phân loại chính xác các mẫu Malware và Benign, đồng thời giảm thiểu được số lượng dự đoán sai.
- Mô hình Decision Tree (DT) có kết quả vượt trội hơn so với các mô hình khác với độ chính xác (Accuracy) 85.07%, điều này phản ánh khả năng tổng thể của mô hình này là tốt nhất trong việc phân loại và phát hiện các mẫu là malware hay benign (phần mềm sạch). Ngoài ra, hiệu suất của thuật toán Decision Tree còn được thể hiện qua giá trị F1-score cao nhất là 86.69%, cho thấy Decision Tree cân bằng tốt giữa việc phát hiện đúng malware và khả năng phát hiện một cách toàn diện các mẫu malware. Với các giá trị Precision 78.33% và Recall 97.04%,

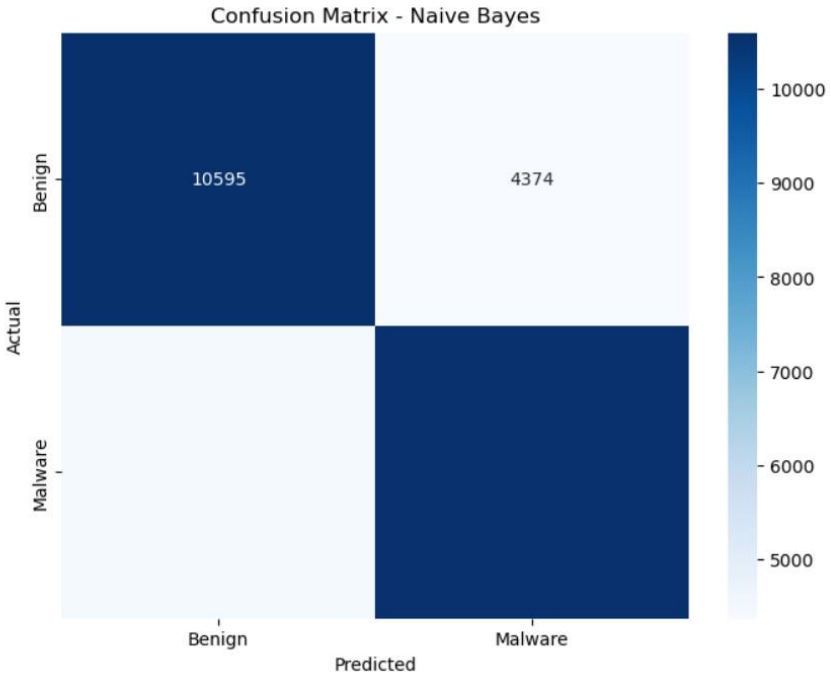
mô hình Decision Tree đã cho ta thấy hiệu suất tốt nhất cùng với khả năng linh hoạt trong việc phân loại và phát hiện malware mà không bỏ sót nhiều mẫu cũng như không phát hiện nhầm với các phần mềm benign.



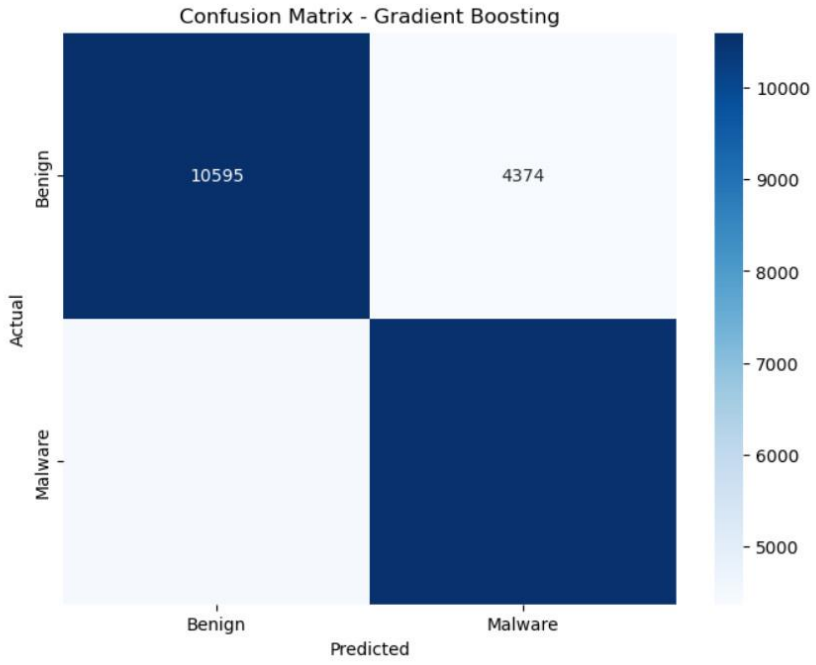
Hình 5. 5 Confusion Matrix của mô hình Random Forest

- Đứng sau mô hình Decision Tree, mô hình Random Forest có độ chính xác (accuracy) khá cao là 82.43%. Tuy nhiên, trái ngược với DT, mô hình Random Forest (RF) có một lượng lớn True Negative (TN - 14965) và True Positive (TP - 9764), đồng thời chỉ có một số ít False Positive (FP - 4) và False Negative (FN - 5267). Điều này cho thấy mô hình có khả năng phân loại chính xác các mẫu Malware và Benign, nhưng tỷ lệ False Negative (FN) khá cao so với mô hình Decision Tree, có thể thấy Random Forest đã bỏ sót một lượng các mẫu Malware.
- Mô hình Random Forest có Precision cao nhất là 99.95% trong số các mô hình được huấn luyện, gần như tuyệt đối, nhưng lại gặp phải vấn đề với Recall thấp là 64.95%. Điều này cho thấy mô hình Random Forest rất chính xác khi nó dự đoán một mẫu là Malware, nhưng lại bỏ sót một lượng các mẫu malware thực sự, giá trị F1-score của mô hình Random Forest (78.74%) cũng từ đó mà thấp hơn so với F1-score của mô hình Decision Tree (86.69%). Sự mất cân bằng

giữa Precision và Recall đã làm giảm hiệu suất của Random Forest đáng kể trong việc phân loại và phát hiện các mẫu malware.



Hình 5. 6 Confusion Matrix của mô hình Naïve Bayes



Hình 5. 7 Confusion Matrix của mô hình Gradient Boosting

- Cả hai mô hình Naïve Bayes và Gradient Boosting đều có số lượng lớn True Negative (TN) và True Positive (TP), cùng với một số ít False Positive (FP) và False Negative (FN). Tuy nhiên, tỷ lệ False Negative (FN) của cả hai mô hình đều cao, chỉ ra rằng chúng đều gặp khó khăn trong việc phát hiện tất cả các mẫu Malware có trong tập dữ liệu.
- Mô hình Naive Bayes (NB) và Gradient Boosting (GB) đạt được độ chính xác (accuracy) tương đối thấp hơn so với các mô hình Decision Tree và Random Forest, chỉ lần lượt là 70.51% và 70.53%. Precision của Naive Bayes và Gradient Boosting đều ở mức 70.70% và 70.71%, chỉ số này thể hiện khả năng của hai mô hình này trong việc dự đoán các mẫu malware và benign khá chính xác. Tuy nhiên, điểm yếu nằm ở tiêu chí Recall, tỷ lệ dự đoán đúng so với tổng số mẫu thực sự của Naive Bayes và Gradient Boosting chỉ là 70.24% và 70.28%, điều này cho thấy chúng không thể phát hiện và phân loại đúng một số lượng các mẫu malware. Giá trị gần nhau giữa Precision và Recall này dẫn đến giá trị F1-score không chênh lệch quá nhiều, chỉ là 70.47% và 70.49% cho Naive Bayes và Gradient Boosting. Mặc dù có khả năng dự đoán chính xác một số lượng mẫu đáng kể, Naive Bayes và Gradient Boosting không thể hiện sự linh hoạt và hiệu suất như Decision Tree và Random Forest trong việc phân loại và phát hiện các mẫu malware.

Từ những phân tích và đánh giá trên, ta có thể thấy rằng mô hình Decision Tree (DT) đã thể hiện sự vượt trội về các mặt tiêu chí đánh giá: độ chính xác cao nhất, F1-score cân bằng, và khả năng linh hoạt trong việc phân loại và phát hiện các mẫu malware. Mặc dù mô hình Random Forest (RF) có tỉ lệ Precision cao, nhưng tỉ lệ Recall đã làm giảm hiệu suất của mô hình này. Các mô hình Naive Bayes (NB) và Gradient Boosting (GB) đều có khả năng dự đoán tương đối, nhưng không đạt được hiệu suất cao như DT và RF, đặc biệt trong việc phát hiện các mẫu Malware. Do đó, mô hình Decision Tree (DT) là lựa chọn tốt nhất cho bộ dữ liệu nghiên cứu này. Với độ chính xác cao, kết hợp với mức độ cân bằng giữa Precision và Recall, điều này cho thấy

Decision Tree có khả năng phát hiện Malware một cách chính xác và đồng thời giảm thiểu tỉ lệ phát hiện nhầm giữa malware và benign.

CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1 Kết luận

Báo cáo của chúng tôi đã đề xuất bốn mô hình học máy phù học với bài toán phân loại nhị phân, cụ thể là phát hiện phần mềm độc hại, đó là mô hình Decision Tree, Random Forest, Naive Bayes và Gradient Boosting. Nghiên cứu đã đạt được kết quả tốt như mục tiêu đã đề ra với mô hình Decision Tree đạt hiệu suất cao nhất, các mô hình còn lại tuy đạt hiệu suất thấp hơn nhưng đây sẽ là cơ sở cũng như tiềm năng phát triển tốt hơn cho những bài nghiên cứu sắp tới. Việc sử dụng nhiều mô hình học máy khác nhau sẽ là tiền đề để so sánh hiệu suất và đưa ra lựa chọn mô hình tốt nhất cho bài toán.

Ưu điểm của các mô hình mà chúng tôi rút ra được trong quá trình nghiên cứu là kết quả đã đạt được như mong đợi, hạn chế được tình trạng over-fitting và under-fitting. Mô hình Decision Tree với hiệu suất cao cũng như linh hoạt, tránh bỏ sót nhiều mẫu trong quá trình phát hiện và phân loại phần mềm độc hại.

Mặt khác, nghiên cứu của chúng tôi cũng có một vài nhược điểm, quá trình tìm kiếm tham số phù hợp cho mô hình là một thách thức lớn, việc áp dụng một vài kỹ thuật nhằm tối ưu quá trình này mang lại hiệu quả tốt hơn nhưng lại tốn nhiều thời gian cho quá trình thực thi tìm kiếm tham số cũng như huấn luyện mô hình, điều này gây ra một vài trở ngại đối với một vài thiết bị hạn chế về mặt tài nguyên cũng như những tác vụ cần được xem xét trong thời gian thực.

6.2 Hướng phát triển

Trong những nghiên cứu sắp tới, chúng tôi sẽ nỗ lực mở rộng phạm vi của mô hình nhằm phát hiện và phòng chống các biến thể mới của phần mềm độc hại. Đồng thời, chúng tôi sẽ tiếp tục tối ưu hóa các thuật toán học máy đã đề xuất Decision Tree, Random Forest, Naive Bayes, Gradient Boosting và các mô hình học máy khác để cải thiện hiệu suất và độ chính xác của mô hình. Việc nghiên cứu thêm các mô hình học sâu (Deep Learning) và học tăng cường (Reinforcement Learning) cũng sẽ được tích hợp nhằm tìm ra mô hình phát hiện phần mềm độc hại mạnh mẽ và hiệu quả hơn.

Ngoài ra, chúng tôi sẽ đa dạng hóa bộ dữ liệu bằng cách tìm kiếm thêm nhiều bộ dữ liệu phần mềm độc hại khác nhau nhằm nâng cao hiệu suất khi huấn luyện mô hình, tiếp tục nghiên cứu và phát triển các kỹ thuật xử lý dữ liệu để cải thiện khả năng tổng quát hóa mô hình. Việc nghiên cứu thêm các kỹ thuật tìm kiếm tham số hay phương pháp làm tăng hiệu suất mô hình sẽ làm giảm thiểu thời gian tính toán, đảm bảo các mô hình vẫn có thể hoạt động một cách hiệu quả trên các thiết bị có tài nguyên hạn chế, cũng như nâng cao hiệu quả xem xét các tác vụ trong thời gian thực.

TÀI LIỆU THAM KHẢO

Tiếng Anh

[1]: Yazdinejad, A.; HaddadPajouh, H.; Dehghantanha, A.; Parizi, R.; Srivastava, G.; Chen, M.-Y. Cryptocurrency malware hunting: A deep recurrent neural network approach. Appl. Soft Comput. 2020, 96, 106630. [CrossRef]

[2]: Hwang, C.; Hwang, J.; Kwak, J.; Lee, T. Platform-independent malware analysis applicable to windows and linux environments. Electronics 2020, 9, 793.[CrossRef]

[3]: Ban, Y.; Lee, S.; Song, D.; Cho, H.; Yi, J.H. FAM: Featuring Android Malware for Deep Learning-Based Familial Analysis. IEEE Access 2022, 10, 20008–20018.[CrossRef]

[4]: Paloalto (2023). What is Malware & How to Stay Protected from Malware Attacks. Retrieved December 3, 2023, from Paloalto website: https://www.paloaltonetworks.com/cyberpedia/what-is-malware

[5]: Advenica. History of Malware. Retrieved December 3, 2023, from advenica website: https://advenica.com/learning-center/blog/the-history-of-malware/

[6]: Nationwide (2023). Understanding, recognizing and preventing malware. Retrieved December 3, 2023, from Nationwide website: https://www.nationwide.com/lc/resources/personal-finance/articles/how-malware-works

[7]: Free Learning Platform For Better Future (2024). Decision Tree Classification Algorithm. Retrieved January 7, 2024, from Free Learning Platform For Better Future website: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

[8]: Free Learning Platform For Better Future (2024). Random Forest Classification Algorithm. Retrieved January 14, 2024, from Free Learning

Platform For Better Future website: https://www.javatpoint.com/machine-learning-random-forest-algorithm

[9]: Free Learning Platform For Better Future (2024). Naive Bayes Classification Algorithm. Retrieved January 21, 2024, from Free Learning Platform For Better Future website: https://www.javatpoint.com/machine-learning-naive-bayes-classifier

[10]: Free Learning Platform For Better Future (2024). Gradient Boosting Classification Algorithm. Retrieved January 28, 2024, from Free Learning Platform For Better Future website: https://www.javatpoint.com/gbm-in-machine-learning

[11]: Thompson (2020). What Is a DDoS Attack. Retrieved December 3, 2023, from hashedout website: https://www.thesslstore.com/blog/what-is-a-ddos-attack/

[12]: The Insights Now Staff (2022). How Trojan Horse Works – Explained In Detail. Retrieved December 3, 2023, from the insightsnow website: https://theinsightsnow.com/how-trojan-horse-works/

[13]: Elzeiny (2023). The ultimate guide to Naive Bayes. Retrieved January 21, 2024, from mlarchive website: https://mlarchive.com/machine-learning/the-ultimate-guide-to-naive-bayes/

[14]: Zhang (2021). Improving Convection Trigger Functions in Deep Convective Parameterization Schemes Using Machine Learning. Retrieved January 28, 2024, from RearchGate website: https://www.researchgate.net/publication/351542039_Improving_Convection_Trigger_Functions_in_Deep_Convective_Parameterization_Schemes_Using_Machine_Learning

Tiếng Việt

[15]: Tung (2021). Gradient Boosting - Tất tần tật về thuật toán mạnh mẽ nhất trong Machine Learning. Retrieved January 28, 2024, from Viblo website:

https://viblo.asia/p/gradient-boosting-tat-tan-tat-ve-thuat-toan-manh-me-nhat-trong-machine-learning-YWOZrN7vZQ0

[16]: Rabiloo (2021). Các phương pháp đánh giá mô hình học máy, học sâu (Machine learning & Deep learning). Retrieved March 10, 2024, from Rabiloo website: https://rabiloo.com/vi/blog/cac-phuong-phap-danh-gia-mo-hinh-machine-learning-va-deep-learning?fbclid=IwAR30WymHpwPOshAeBLX3tt_wXjq0CbjHkqTHLAGZ1DTWgk6VA1Ufe3HX4y4

[17]: Đức (2021). Đánh giá các mô hình học máy. Retrieved March 10, 2024, from Viblo website: https://viblo.asia/p/danh-gia-cac-mo-hinh-hoc-may-RnB5pp4D5PG?fbclid=IwAR0OujlmWkdrInTMe5kDsd2MxAFL5IY6RgT7K9CjHqL-rgvvI7lZBYbstpE