ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

VÕ VĂN TRƯỜNG

NGHIÊN CỨU VÀ ỨNG DỤNG KỸ THUẬT HỌC MÁY VÀO BÀI TOÁN PHÁT HIỆN MÃ ĐỘC

Ngành: Công nghệ Thông tin
Chuyên ngành: Kỹ thuật phần mềm
Mã số: 60480103

TÓM TẮT LUẬN VĂN THẠC SĨ
NGÀNH CÔNG NGHỆ THÔNG TIN

Hà Nội - 2016

MỤC LỤC

MỞ ĐẦU

Ngày nay song song với sự bùng nổ mạnh mẽ của công nghệ thông tin và sự phát triển của Internet toàn cầu là các nguy cơ mất an toàn thông tin đang trở nên trầm trọng và nguy hiểm hơn, trong đó mã độc hại đang là các hiểm họa hàng đầu bởi khả năng có thể lây lan phát tán trên các hệ thống máy tính và thực hiện các hành vi tấn công bất hợp pháp. Mã độc đang ngày càng tiến hóa với những biết thể đa dạng, với những cách thức che dấu ngày càng tinh vi hơn. Có thể nói phát hiện và ngăn chặn mã độc đang là một thách thức được đặt ra trong lĩnh vực An toàn thông tin. Các phương pháp phát hiện mã độc thông thường chủ yếu sử dụng kỹ thuật so sánh mẫu dựa trên cơ sở dữ liệu mã độc được xây dựng và định nghĩa từ trước, tuy nhiên phương pháp này bộc lộ nhiều nhược điểm đó là không có khả năng phát hiện ra các mẫu mã độc mới, số lượng dữ liệu mã độc ngày càng gia tăng làm cho cơ sở dữ liệu mẫu trở nên ngày càng lớn. Hiện nay hướng nghiên cứu dựa vào các mô hình học máy để phân loại và phát hiện mã độc đang tỏ ra là phương pháp tiềm năng và hiệu quả khi có thể cải thiện được các nhược điểm đã nêu ở trên so với phương pháp truyền thống. Tuy nhiên, một trong những vấn đề được quan tâm là làm sao để xây dựng được mô hình học máy tốt nhất đạt hiệu quả chính xác và hiệu suất cao. Trong đó một yếu tố quan trọng được xem là quyết định chính là giải pháp trích chọn đặc trưng. Trong các phần nghiên cứu của luận văn này tôi trình bày về phương pháp ứng dụng học máy vào xây dựng các mô hình phát hiện mã độc trong đó các thực nghiệm dựa trên phương pháp phân tích tĩnh mã độc, tiền xử lý dữ liệu bằng kỹ thuật dịch ngược đưa các file dữ liệu mẫu về dạng mã hex và thực hiện khai phá dữ liệu text sử dụng các mã n-gram byte là các đặc trưng ban đầu .Sau đó các dữ liệu đặc trưng này sẽ được trích chọn ra một bộ dữ liệu đặc trưng tốt nhất để xây dựng mô hình trên cơ sở giải pháp trích chọn đặc trưng mà trong luận văn này tôi đã tập trung nghiên cứu và đề xuất. Các kết quả của luận văn được thực nghiệm trên khoảng 4698 file mẫu thực thi trên nền Windows trong đó 2373 file mã thông thường và 2325 file mẫu mã độc với nhiều thể loại đa dạng như Backdoor, Virus, Trojan, Worm...

Nội dung luận văn được chia ra làm 4 phần như sau:

Chương 1: Chương này nghiên cứu tổng quan về mã độc trình bày các kiến thức chung nhất liên quan đến mã độc, phân loại mã độc cũng như các

kỹ thuật phân tích và phát hiện mã độc hiện nay.

Chương 2: Nghiên cứu một số thuật toán phân lớp dữ liệu điển hình trong kỹ thuật học máy giám sát trong đó 2 thuật toán phân lớp dữ liệu tiêu biểu được trình bày là cây quyết định (DT) và máy véc tơ hỗ trợ (SVM)

Chương 3: Chương này trình bày giải pháp ứng dụng kỹ thuật học máy vào phát hiện mã độc bao gồm quá trình tiền xử lý dữ liệu, xây dựng các mô hình học máy để phát hiện mã độc. Trong đó trọng tâm là trình bày một đề xuất giải pháp chọn đặc trưng cải thiện và nâng cao hiệu quả cho các thuật toán phân lớp đối với bài toán phát hiện mã độc.

Chương 4: Trình bày về quá trình thực nghiệm và đánh giá, các kết quả được thực nghiệm và so sánh trên các tập đặc trưng được chọn dựa trên giải pháp đã đề xuất, các kết quả cũng được so sánh giữa 2 thuật toán phân lớp đã trình bày là cây quyết định và máy véc tơ hỗ trợ.

CHƯƠNG 1 TỔNG QUAN VỀ MÃ ĐỘC

1.1. Giới thiệu về mã độc máy tính

Mã độc hay phần mềm độc hại là các chương trình máy tính có chứa bên trong nó nội dung các mã độc hại được tạo ra với mục đích thực hiện các hành vi bất hợp pháp. Bất kỳ một phần mềm nào là lý do làm tổn thương, phá vỡ đến tính bí mật, tính toàn vẹn và tính sẵn sàng của dữ liệu người dùng, máy tính hoặc môi trường mạng đều có thể được xem như các mã độc.

1.2. Phân loại mã độc

Ban đầu các phần mềm độc hại được tạo ra bằng cách sống kí sinh và lây nhiễm trên các vật chủ là các chương trình có chứa các nội dung thực thi. Các dạng chương trình độc hại kiểu khác là các chương trình mã độc mà tự chúng có khả năng thực thi một các độc lập trên các phần mềm hệ thống mà không cần kí sinh trên các vật chủ là các chương trình ứng dụng hay tệp tin.

Một cách khác để phân biệt các loại mã độc khác nhau là dựa trên mục đích và các hành vi của chúng.

Có 3 đặc điểm thường liên quan với các loại phần mềm độc hại là:

- Mã độc tự nhân bản và nỗ lực lây nhiễm bằng việc tạo các bản sao chép mới hoặc các thể hiện của chính nó.
- Sự tăng trưởng về số lượng của phần mềm độc hại mô tả sự thay đổi tổng thể trong đó có số lượng lớn các trường hợp là do tự nhân bản
- Phần mềm độc hại ký sinh yêu cầu một số chương trình mã thực thi khác để tồn tại.

1.2.1. Virus máy tính

Virus là một loại mã độc có các đặc điểm như tự nhân bản, chúng ký sinh trên các vật chủ, virus là một chương trình độc hại mà khi được thực thi nó sẽ cố gắng sao chép chính nó vào bên trong một mã thực thi khác.

Một virus máy tính gồm 3 thành phần chính như sau:

Cơ chế lây nhiễm: Làm thế nào để virus có thể lây lan bằng cách sửa đổi mã khác để chứa một sao chép của virus, các cách chính xác thông qua đó một loại virus lây lan được gọi là vector lây nhiễm của nó.

Kích hoạt: Các cách quyết định xem có hay không để mở một payload

Payload: là mã lệnh hay cái mà virus thực hiện bên cạnh việc lây nhiễm

Virus có thể được phân loại bằng nhiều cách, dựa vào mục tiêu cố gắng

lây nhiễm của virus, và dựa vào các phương thức mà virus sử dụng để che giấu bản thân nó với các hệ thống phát hiện và các phần mềm chống virus.

1.2.1.1. Phân loại virus dựa vào các hình thức lây nhiễm:

1.2.1.1.1. Virus lây nhiễm Boot-Sector

Virus lây nhiễm Boot-Sector là một virus lây nhiễm bằng việc sao chép chính nó đến khối khởi động bằng cách như vậy sau khi hệ điều hành máy tính tiến hành khởi động nó sẽ thực thi các đoạn mã virus bị chèn vào chứ không phải là các đoạn mã khởi động thông thường.

1.2.1.1.2. Virus lây nhiễm tập tin:

Để tiến hành lây nhiễm virus thực hiện ác công việc như sau:

- ✓ Tìm kiếm đối tượng tệp thực thi để lây nhiễm
- ✓ Nạp các mã độc lây nhiễm vào các tập tin tìm được
- ✓ Lưu trú và đảm bảo nó tồn tại duy nhất trong tập tin bị lây nhiễm
- ✓ Tiếp tục tìm kiếm các tệp tin khác để tiến hành lẫy nhiễm mã độc

Mỗi một loại virus sẽ có những cách thức để thực hiện lây nhiễm khác nhau song chúng đều được trải qua các nguyên lý bao gồm 3 giai đoạn là:

- ✓ Lây nhiễm và chiếm quyền điều khiển,
- ✓ Chạy các chức năng của nó là các đoạn mã độc được gắn vào tệp tin lây nhiễm
- ✓ Trao trả quyền điều khiển cho tệp và đảm bảo tính toàn vẹn ban đầu của dữ liệu.

1.2.1.1.3. Virus Macro:

Một số ứng dụng cho phép chứa các tập tin dữ liệu, như các trình xử lý văn bản cho phép nhúng các hàm macro bên trong chúng, Macro là một đoạn mã nhỏ được viết bởi một ngôn ngữ chuẩn được biên dịch bởi ứng dụng, ngôn ngữ này cung cấp đầy đủ các hàm để viết một virus. Do vậy virus Macro chiếm được nhiều lợi thế hơn là virus lây nhiễm tập tin.

1.2.1.2. Phân loại virus dựa trên các chiến lược ẩn náu:

Một cách khác để phân loại virus là dựa vào các cách làm thế nào để chúng có thể ẩn náu và qua mặt được người dùng cũng như các phầm mềm

diệt virus.

1.2.1.2.1. Virus mã hóa

Đối với một virus được mã hóa ý tưởng là phần thân của virus bao gồm phần lây nhiễm, kích hoạt và payload sẽ được mã hóa thông qua một vài cách thức, mục đích là gây ra sự khó khăn để có thể phát hiện ra chúng, khi phần thân của virus ở dạng mã hóa nó sẽ không thực thi cho đến khi được giải mã, sau một vòng giải mã sẽ giải mã phần thân của virus và chuyển điều khiển trỏ đến nó, bằng cách này virus giải mã sẽ cung cấp ít thông tin về chúng hơn với các phần mềm phát hiện mã độc.

1.2.1.2.2. Virus tàng hình

Virus tàng hình là một virus nỗ lực tiến hành các bước để che dấu sự lây nhiễm của chính nó, không chỉ có phần thân virus, một virus tàng hình luôn cố gắng ẩn mọi thứ không chỉ với các phần mềm chống virus.

Virus có thể được lưu dữ hoặc có khả năng tái sinh tất cả những thông tin về tập tin trước khi lây nhiễm bao gồm nhãn thời gian của nó, kích thước tập tin, và nội dung của tập tin, sau đó các cuộc gọi vào/ra hệ thống có thể bị ngăn chặn, và virus sẽ phát lại các thông tin ban đầu đề đáp ứng bất kì hoạt động vào/ra nào trên các tập tin bị lây nhiễm, làm cho nó xuất hiện như là không bị lây nhiễm.

1.2.1.2.3. Virus Oligomorphism:

Một virus oligomorphic hay là virus bán đa hình là một virus được mã hóa trong đó có một hữu hạn nhỏ số lượng các vòng lặp giải mã khác nhau theo ý của nó. Vì vậy virus này nó có thể sinh ra được hữu hạn các biến thể khác nhau.

1.2.1.2.4. Virus đa hình:

Một virus đa hình bề ngoài thì giống như một virus Oligomorphism, cả hai cùng mã hóa virus, cả hai cùng thay đổi vòng lặp giải mã của chúng cho mỗi lần lây nhiễm, tuy nhiên một virus đa hình có một số lượng vô hạn các biến thể của vòng lặp giải mã. Tremor là một ví dụ nó có tới gần sáu tỷ khả năng lặp giải mã tức là có thể sinh ra 6 tỷ biến thể khác nhau. Có thể nói virus đa hình rõ ràng không thể bị phát hiện bởi liệt kê tất cả các kết hợp có thể của chúng.

1.2.1.2.5. Virus siêu đa hình:

Virus siêu đa hình là virus mà đa hình phần thân mã bên trong của nó. Chúng không thực hiện mã hóa do đó không cần các vòng giải mã tuy nhiên để tránh bị phát hiện bởi sự thay đổi, một phiên bản mới hay biến thể của phần thân virus được thực hiện cho mỗi lần lây nhiễm mới.

1.2.2. Logic Bomb

Logic bomb là loại mã độc không tự nhân bản chúng có thể ký sinh trên các vật chủ, một mã độc logic bomb bao gồm 2 phần chính là :

Một payload là các mã nạp có thể là bất cứ thứ gì nhưng có ý đồ xấu hay là các mục đích của mã độc như là phá hoại hay phá hủy các chức năng an toàn của hệ thống.

Một trigger (kích hoạt) là một điều kiện đúng hoặc sai được đánh giá và điều khiển khi một payload được thực thi

1.2.3. Trojan Horse:

Trojan là một loại mã độc mặc dù không có khả năng tự nhân bản tuy nhiên nó thường lây nhiễm vào hệ thống với những thể hiện rất bình thường dưới hình thức đóng giả như là các chương trình phần mềm hữu ích thực hiện những nhiệm vụ thông thường và hợp pháp, nhưng thực chất bên trong lại bí mật thực hiện một số nhiệm vụ nguy hiểm, ác tính mà kẻ có ý đồ xấu đã cài đặt vào nó.

1.2.4. Back Door

Back door là một mã độc không tự tái tạo và chúng thường kí sinh trên các vật chủ một back door là bất kì một cơ chế nào cái mà cho phép vượt qua các kiểm tra an ninh thông thường.

Back door tỏ ra rất nguy hiểm bởi khả năng lẩn trốn của nó đôi khi chúng được hẹn trước thời gian để kết nối ra ngoài vì thế trong thời gian lưu trú chúng không để lộ bất kỳ hành vi hay thông tin gì cho phép các phần mềm diệt mã độc phát hiện ra.

1.2.5. Sâu máy tính (Worm):

Sâu máy tính là loại mã độc có khả năng tự nhân bản tuy nhiên chúng thường xuất hiện như những chương trình độc lập mà không cần tập tin chủ để mang nó. Sâu máy tính hoàn toàn có khả năng sao chép và nhân bản chính nó mà không cần bất kỳ thao tác nào của người sử dụng đây là một đặc điểm

giúp chúng lan rộng với tốc độ chóng mặt.

Sâu dịch vụ mạng (Network Service Worm): chúng phát tán và lây lan thông qua việc khai thác các lỗ hổng bảo mật của các dịch vụ mạng, các ứng dụng hay hệ điều hành.

Sâu gửi điện thư hàng loạt (Mass Mailing Worm): Chúng là các sâu tấn công vào dịch vụ thư điện tử cách thức hoạt động của chúng là lây lan trên các hệ thống thư điện tử.

1.3. Các kỹ thuật phát hiện mã độc

Các kỹ thuật phát hiện mã độc là một quá trình tìm kiếm và thẩm định xem một chương trình phần mềm có thể đã bị lây nhiễm mã độc hay bên trong có chứa các đoạn mã được xem là mã độc hay không, thêm vào đó các hành vi của chúng cũng được phân tích và xem xét là nhóm các hành vi thông thường hay các hành vi thuộc về mã độc, dựa vào các kết quả đó để có thể chứng minh và phát hiện sự tồn tại của mã độc trên các hệ thống.

1.3.1. Các kỹ thuật phát hiện dựa trên phân tích tĩnh

Kỹ thuật phát hiện mã độc dựa trên phương pháp phân tích tĩnh có đặc điểm là phát hiện mã độc mà không cần phải chạy hay thực thi bất kỳ đoạn mã nào của nó gồm có 3 phương pháp chính là kỹ thuật dò quét, chẩn đoán dựa trên kinh nghiệm và kiểm tra tính toàn vẹn.

1.3.1.1. Kỹ thuật dò quét (scanner):

Thông thường mỗi một mã độc được biểu diễn bởi một hay nhiều mẫu, hoặc là các dấu hiệu (signatures), chuỗi tuần tự các byte là những cái được coi là đặc trưng duy nhất của mã độc. Các dấu hiệu này đôi khi còn được gọi là các chuỗi (scan strings) và chúng không cần bất kỳ một ràng buộc về chuỗi nào. Quá trình phát hiện mã độc bằng cách tìm kiếm thông qua một tập tin với các dấu hiệu của nó thì được gọi là scanning và các mã được tìm thấy được gọi là một scanner.

1.3.1.2. Kỹ thuật Static Heuristics

Kỹ thuật này được áp dụng để nhân lên khả năng chuyên gia trong các phần mềm chống virus, chẩn đoán dựa trên kinh nghiệm trong phương pháp phân tích tĩnh có thể tìm thấy các mã độc đã biết hoặc chưa biết bằng cách tìm kiếm một mẩu mã mà có những đặc điểm chung giống như là một mã độc thay vì scanning các dấu hiệu đặc biệt của mã độc

Một số phương pháp phức tạp trong các phân tích dữ liệu có thể sử dụng trí tuệ nhân tạo như là mạng neural, hệ chuyên gia, hay các kỹ thuật khai phá dữ liệu.

1.3.1.3. Kỹ thuật kiểm tra sự toàn vẹn (Integrity Checkers)

Kỹ thuật kiểm tra tính toàn vẹn nhằm khai thác các hành vi này để tìm ra mã độc bằng cách xem các thay đổi trái phép vào các tập tin. Một kiểm tra toàn vẹn được khởi đầu bằng việc tính và lưu dữ một checksum cho mỗi tập tin trong hệ thống được xem xét. Sau đó một checksum của tập tin cần kiểm tra sẽ được tính lại và so sánh với giá trị checksum gốc của nó. Nếu checksum khác nhau có nghĩa là đã có một sự thay đổi diễn ra.

1.3.2. Các kỹ thuật phát hiện dựa trên phân tích động

Kỹ thuật phát hiện mã độc dựa trên phân tích động là kỹ thuật quyết định một tập tin có bị lây nhiễm hay không thông qua việc thực thi các mã chương trình và quan sát các hành vi của nó.

1.3.2.1. Kỹ thuật Behavior Monitors/Blockers

Một behavior blocker là một kỹ thuật chống mã độc giám sát các hành vi thực thi của một chương trình trong thời gian thực, theo dõi các hành động, các khối lệnh khả nghi của nó.

1.3.2.2. Kỹ thuật Emulation

Giám sát các hành vi kỹ thuật Behavior blocking cho phép mã chương trình chạy trên máy thực. Ngược lại kỹ thuật phát hiện mã độc sử dụng mô phỏng giả lập (emulation) cho phép các mã được chạy và phân tích trong một môi trường mô phỏng giả lập.

Bao gồm 2 kỹ thuật chính là Dynamic heuristic và Generic decryption.

CHƯƠNG 2 MỘT SỐ THUẬT TOÁN PHÂN LỚP DỮ LIỆU ĐIỂN HÌNH TRONG KỸ THUẬT HỌC MÁY GIÁM SÁT

2.1. Thuật toán cây quyết định

2.1.1. Giới thiệu thuật toán

Học máy dựa trên cây quyết định là một phương pháp xấp xỉ với hàm mục tiêu có giá trị rời rạc trong đó các hàm học được biểu diễn bởi một cây quyết định. Cây quyết định cũng có thể được biểu diễn bởi một tập các luật if-then các luật này thì khá là gần gũi với tư duy con người.

Một cây quyết định trong học máy được mô tả bởi:

- Mỗi nút trong biểu diễn một thuộc tính hay một đặc trưng cần kiểm tra của dữ liệu mẫu
- Mỗi nhánh đi ra từ một nút thuộc tính biểu diễn giá trị có thể có của biến thuộc tính đó
- Mỗi nút lá biểu diễn cho một giá trị phân lớp (là giá trị các nhãn trong tập huấn luyện)

Cây quyết định được dùng để dự đoán các dữ liệu chưa biết bằng cách duyệt cây từ nút gốc đến một nút lá được gán nhãn.

2.1.2. Xây dựng cây quyết định dựa trên thuật toán ID3

Giải thuật ID3:

Function Build_Tree(tập_ví_dụ, thuộc_tính_mục_tiêu,tập_thuộc_tính)

Trong đó tập_ví_dụ là tập các trường hợp hay dữ liệu mẫu huấn luyện, thuộc_tính_mục_tiêu là thuộc tính mà giá trị của nó là giá trị đích được dự đoán bởi cây quyết định , tập_thuộc_tính là một danh sách thuộc tính khác có thể được kiểm tra để xây dựng cây quyết định được học .

Begin

If tất cả các ví dụ trong tập_ví_dụ thuộc vào lớp dương **then return** một nút gốc được gán nhãn là (+)

Else If tất cả các ví dụ trong tập_ví_dụ thuộc vào lớp âm **then return** một nút gốc được gán nhãn là (–)

Else If tập_thuộc_tính mà rỗng **then return** một nút gốc với nhãn được gán bằng giá trị phổ biến nhất của thuộc_tính_mục_tiêu trong tập_ví_dụ

Else

Chọn ra A là thuộc tính tốt nhất để phân lớp từ tập_thuộc_tính
Đặt A là nút gốc cho cây quyết định hiện tại
For each giá trị v_i có thể có của A
Begin
Thêm một nhánh cây mới bên dưới nút gốc, tương đương kiểm tra A= v_i
Đặt tập_ví_du$_{vi}$ là một tập con của tập_ví_dụ là các ví dụ có giá trị v_i cho A
If tập_ví_du$_{vi}$ là rỗng **then** thêm vào bên dưới nhánh mới này một nút lá với nhãn được gán bằng giá trị phổ biến nhất của thuộc_tính_mục_tiêu trong tập_ví_dụ
Else thêm vào bên dưới nhánh mới này một cây con và gọi hàm đệ quy với tập_thuộc_tính không chứa thuộc tính A.
Function Build_Tree(tập_ví_dụ, thuộc_tính_mục tiêu, tập_thuộc_tính - {A})
End
Return ***Nút_gốc***
End

Trong thuật toán ID3 thuộc tính nào nên được kiểm tra ở nút gốc của một cây ? để giải quyết điều này mỗi thuộc tính có trong dữ liệu mẫu sẽ được đánh giá thông qua việc thống kê các độ lợi về thông tin, độ lợi về thông tin của một thuộc tính được hiểu là khả năng phân tách dữ liệu huấn luyện của thuộc tính đó theo các lớp.

Độ đo Entropy: Độ đo entropy là một khái niệm được dùng trong lý thuyết thông tin nhằm đánh giá độ hỗn loạn của thông tin. Trong đó Entropy của một tập S đối với bộ phân lớp gồm C lớp được định nghĩa như sau:

$$Entropy(S) = \sum_{i=1}^{C} -p_i . \log_2 p_i$$

Trong đó: p_i là tỷ lệ các mẫu trong tập S thuộc vào lớp i

Trong công thức này nếu p_i =0 thì ta có thể coi $0.\log_2 0 = 0$

Độ lợi về thông tin (Information Gain):

Độ lợi về thông tin của thuộc tính A đối với tập mẫu S được tính như

sau:

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Trong đó Values(A) là tập các giá trị có thể của thuộc tính A và

$$S_v = \{x \mid x \in S, x_A = v\}$$

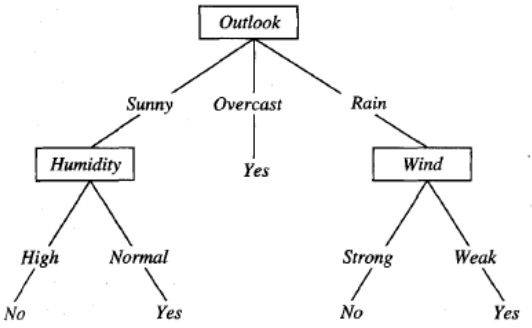
2.1.3. Ví dụ minh họa:

Xét tập dữ liệu mẫu những ngày mà một người đi chơi hay không đi chơi tennis dựa trên các điều kiện thời tiết như sau:

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Hình 2.2. Tập dữ liệu mẫu ví dụ cho xây dựng cây quyết định [18]

Một cây quyết định được sinh ra như sau:



Hình 2.4 Cây quyết định được sinh ra trong dữ liệu mẫu ví dụ chơi tennis [18]

2.1.4. Nhận xét:

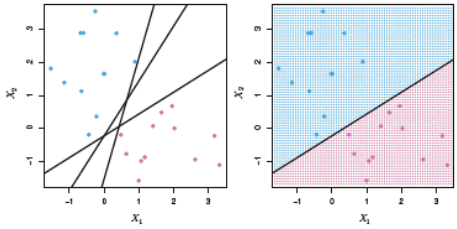
Cây quyết định là một phương pháp học máy mà dữ liệu mẫu được mô

tả thông qua cấu trúc cây nó tương đối dễ hiểu bởi khả năng sinh ra các luật gần gũi với tư duy con người phương pháp này không đòi hỏi cao trong chuẩn hóa dữ liệu đầu vào cũng như xử lý tốt lượng dữ liệu lớn trong thời gian ngắn. Tuy nhiên phương pháp này có nhược điểm là không thể giải quyết được khi nhãn phân lớp là liên tục, nó chỉ có thể thực hiện tốt các bài toán phân lớp nhị phân và đôi khi cắt tỉa một cây để tối ưu mô hình cũng là một vấn đề phức tạp.

2.2. Thuật toán SVM

2.2.1. Giới thiệu thuật toán

Thuật toán này được đề xuất bởi Vladimir N. Vapnik [20] ý tưởng chính của nó là coi các dữ liệu đầu vào như là các điểm trong một không gian n chiều và từ các dữ liệu huấn luyện ban đầu được gán nhãn sẽ tìm ra được một siêu phẳng phân lớp chính xác các dữ liệu này, siêu phẳng sau đó được dùng để phân lớp các dữ liệu chưa biết cần tiên đoán.



Hình 2.5 Biểu diễn phân lớp dựa trên thuật toán SVM [8]

Xét một tập dữ liệu mẫu:

$$\mathcal{D} = \{(x1, y1), \ldots, (xl, yl)\}\ ,\ x \in \mathbb{R}^n\ ,\ y \in \{-1,1\} \qquad (2.1)$$

Trong đó x_i là một véc tơ đặc trưng hay một điểm (trong không gian n chiều $x_i \in \mathbb{R}^n$) biểu diễn tập mẫu d_i cặp (x_i, y_i) biểu diễn rằng với một véc tơ đặc trưng x_i thì được gán nhãn là y_i tương ứng trong đó $y \in \{-1,1\}$ hay nói cách khác với tập mẫu d_i sẽ được gán nhãn cho trước là y_i. Ta có phương trình một siêu phẳng được cho như sau :

$$f(x) = w \cdot x + b = 0 \qquad [17] \qquad (2.2)$$

Trong đó $w \cdot x$ là tích vô hướng giữa véc tơ x và véc tơ pháp tuyến $w \in \mathbb{R}^n$ được biểu diễn trong không gian n chiều, và $b \in \mathbb{R}$ là hệ số tự do.

2.2.2. Bài toán tìm siêu phẳng tối ưu cho dữ liệu tuyến tính và không có nhiễu:

Ta có phương trình siêu phẳng cần tìm có dạng: $w \cdot x + b = 0$

Với: $y = \text{sign}\{ w \cdot x_i + b \} = \begin{cases} 1 & y_i = 1 \\ -1 & y_i = -1 \end{cases} \quad \forall (x_i, y_i) \in D$

Xét các điều kiện ràng buộc với các tham số w và b như sau:

$$\min_i \left| (w \cdot x_i) + b \right| = 1 \text{ với } i = 1, ..., l \qquad (2.3)$$

Các ràng buộc này thì giúp đơn giản hóa vấn đề hơn nói cách khác chuẩn của một véc tơ trọng số nên được tính bằng nghịch đảo khoảng cách của điểm gần nhất trong dữ liệu đến siêu phẳng.

Một siêu phẳng phân chia cần phải thỏa mãn điều kiện ràng buộc sau đây:

$$y_i \left[w \cdot x_i + b \right] \geq 1 \text{ với } i = 1, ..., l \qquad (2.4)$$

Để tìm được một siêu phẳng tốt nhất trong trường hợp này chúng ta cần phải đi giải quyết bài toán tối ưu với các hàm mục tiêu và ràng buộc như sau:

$$\begin{cases} \underset{w}{Min}\ \Phi(w) = \frac{1}{2} \|w\|^2 \\ y_i (w \cdot x_i + b) \geq 1 \quad i = 1, .., l \end{cases} \qquad (2.7)$$

2.2.3. Bài toán tìm siêu phẳng tối ưu cho dữ liệu tuyến tính và có xảy ra nhiễu:

Điều này có nghĩa rằng không phải tất cả các điểm dữ liệu thuộc lớp dương (được gán nhãn y = 1) sẽ nằm về một phía của siêu phẳng và tất cả các điểm thuộc lớp âm (được gán nhãn y = -1) sẽ nằm về phía còn lại của siêu phẳng đó. Đôi khi có một vài điểm nhiễm thuộc lớp dương nhưng lại bị siêu phẳng phân loại nhầm vào lớp âm tức là nó nằm về phía bên âm của siêu phẳng đó. Tương tự trong trường hợp dữ liệu âm nằm vào vùng dương của siêu phẳng.

Trong trường hợp này các ràng buộc được linh động và thay đổi như sau:

$$y_i \left(w \cdot x_i + b \right) \geq 1 - \xi_i \quad \text{với } i = 1, ..., l \qquad (2.8)$$

$\xi_i \geq 0$ được gọi là biến số nới lỏng cho các điểm dữ liệu huấn luyện.

Như vậy lúc này để tìm siêu phẳng tốt nhất chính là đi giải quyết bài toán ưu với các hàm mục tiêu và ràng buộc như sau [4]:

$$\begin{cases} Min\ \Phi(\mathrm{w},\xi)=\frac{1}{2}\|\mathrm{w}\|^2+C\sum_{i=1}^{l}\xi_i \\ y_i(\mathrm{w}\cdot x_i+b) \geq 1-\xi_i \qquad i=1,..,l \\ \xi_i\geq 0 \quad i=1,..,l \end{cases}$$

Trong đó C là tham số được thêm vào với mục đích cân bằng lỗi và để điều chỉnh mức độ vi phạm đối với lỗi trong thực nghiệm.

2.2.4. Bài toán tìm siêu phẳng tối ưu cho dữ liệu không tuyến tính:

Trong thuật toán máy véc tơ hỗ trợ để giải quyết những bài toán với các dữ liệu không thể phân chia tuyến tính giải pháp được đưa ra là ánh xạ các điểm dữ liệu ban đầu sang một không gian mới nhiều chiều hơn, trong không gian mới này chúng ta sẽ dễ dàng tìm được các siêu phẳng để phân tách các dữ liệu này một cách tuyến tính. Việc ánh xạ các điểm dữ liệu x trong không gian n chiều vào một không gian m chiều lớn hơn , được mô tả với một ánh xạ phi tuyến tính ϕ như sau:

$$\phi: R^n \rightarrow R^m$$

Trong không gian mới việc tìm ra các siêu phẳng sẽ dựa trên các điểm ánh xạ $\phi(x)$ các siêu phẳng trong trường hợp này được xác định thông qua việc giải bài toán tối ưu với hàm mục tiêu và các ràng buộc như sau:

$$\begin{cases} Min\ \Phi(\mathrm{w},\xi)=\frac{1}{2}\|\mathrm{w}\|^2+C\sum_{i=1}^{l}\xi_i \\ y_i(\mathrm{w}\cdot\phi(x_i)+b) \geq 1-\xi_i \qquad i=1,..,l \\ \xi_i\geq 0 \quad i=1,..,l \end{cases}$$

2.2.5. Hàm nhân (Kernel)

Trên thực tế đối với thuật toán svm chúng ta không cần phải biết một cách quá rõ ràng các giá trị của một véc tơ dữ liệu trong không gian mới như thế nào, để tìm ra các siêu phẳng tối ưu giá trị cần quan tâm nhất chính là tích

vô hướng giữa các véc tơ đặc trưng là ảnh của véc tơ dữ liệu ban đầu trong không gian mới, một hàm cho phép tìm ra các tích vô hướng của các dữ liệu ảnh này được gọi hàm nhân Kernel đây là một hàm hết sức quan trọng trong thuật toán svm:

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

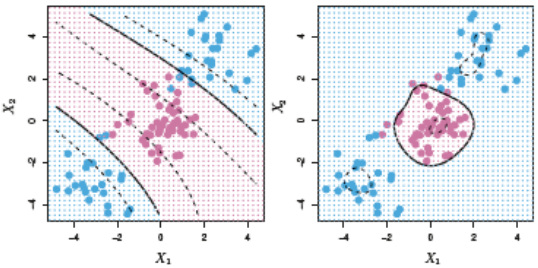
Một số hàm nhân (Kernel) cơ bản [4]:

Hàm nhân tuyến tính (linear): $K(x_i, x_j) = x_i \cdot x_j$

Hàm nhân đa thức (polynomial): $K(x_i, x_j) = (\gamma x_i \cdot x_j + r)^d, \gamma > 0$

Hàm RBF (radial basis funciont) $K(x_i, x_j) = \exp(-\gamma \left\| x_i - x_j \right\|^2), \quad \gamma > 0$

Hàm sigmoid: $K(x_i, x_j) = \tanh(\gamma x_i \cdot x_j + r)$



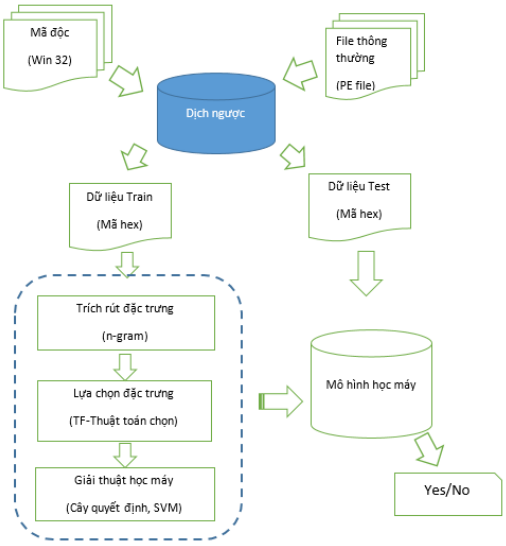
Hình 2.9 Dữ liệu được phân lớp bởi hàm nhân (kernel) [8]

CHƯƠNG 3 GIẢI PHÁP ỨNG DỤNG KỸ THUẬT HỌC MÁY VÀO PHÁT HIỆN MÃ ĐỘC

3.1. Tổng quan về phương pháp thực hiện

Trong luận văn này tôi trình bày một hướng nghiên cứu trong đó ứng dụng kỹ thuật học máy vào việc phân lớp và phát hiện mã độc. Luận văn tập trung nghiên cứu và đưa ra một đề xuất cũng như xây dựng một giải pháp trích chọn đặc trưng nâng cao hiệu quả và phù hợp cho các bài toán phân lớp dữ liệu. Quá trình được thực nghiệm và phân tích trên các bộ dữ liệu mã độc chỉ ra rằng phương pháp đề xuất cho kết quả phân lớp chính xác và hiệu suất tương đối tốt.

Tổng quan về phương pháp được mô tả qua 6 bước sau:



Hình 3.1 Tổng quan về phương pháp thực hiện

Bước 1: Thu thập dữ liệu các file mã độc và các file PE thông thường.

Bước 2: Các file này sẽ được dịch ngược về mã hex thông qua một chương trình được viết bằng ngôn ngữ Python.

Bước 3: Sau khi dịch ngược thì các dữ liệu huấn luyện sẽ được trích rút ra các đặc trưng là các mã hex dựa vào phương pháp n-gram. Trong khuôn khổ nghiên cứu này kích thước sử dụng là 2-gram.

Bước 4: Từ các đặc trưng là các mã n-gram byte thực hiện tính tần số

xuất hiện (Term Frequency) của các mã n-gram này trên mỗi tập dữ liệu. Sau đó áp dụng thuật toán được tôi đề xuất (được trình bày ở phần sau) để chọn ra được một bộ đặc trưng tốt nhất.

Bước 5: Từ bộ đặc trưng có được ta đưa chúng vào xây dựng các mô hình học máy ở đây tôi sử dụng hai giải thuật để thực nghiệm và so sánh là cây quyết định và svm.

Bước 6: Sau khi xây dựng xong mô hình thì đưa các dữ liệu test vào để đánh giá kết quả.

3.2. Tiền xử lý dữ liệu

3.2.1. Sử dụng các kỹ thuật phân tích mã độc

Trong nội dung nghiên cứu và thử nghiệm này tôi sử dụng phương pháp phân tích tĩnh, ở giai đoạn tiền xử lý này các file thông thường và các file mã độc là các file dạng PE sẽ được dịch ngược về các mã hex có rất nhiều công cụ cho phép thực hiện điều này có thể kể ra như: IDA, OLLYDBG… Tuy nhiên để đồng bộ và phát triển các hệ thống tự động sau này, tôi đã xây dựng một chương trình dựa trên cấu trúc của một file PE cho phép dịch ngược các file này về mã hex. Chương trình được viết bằng ngôn ngữ Python và sử dụng thư viện Pefile. Các file dữ liệu mẫu sau khi được dịch ngược sẽ được xử lý để lấy các mã hex quan trọng chủ yếu chúng nằm ở các phần PE header và Section nơi chứa các mã chương trình (Executable Code Section), các dữ liệu (Data Section), các tài nguyên (Resources Section), các thư viện (Import Data ,Export Data) …

Các file thông thường và mã độc sau khi được dịch ngược sẽ lấy nội dung các mã hex của từng file này và lưu lại thành các file text tương đương phục vụ cho quá trình trích chọn đặc trưng và xây dựng mô hình dự đoán.

3.2.2. Phương pháp n-gram

Trong nghiên cứu này tôi sử dụng chuỗi các byte là các đặc trưng đầu vào trong đó ở giai đoạn tiền xử lý các file dữ liệu mẫu được trích xuất dựa vào việc tính tần số xuất hiện của các n-gram byte. N-gram là một dãy các byte liên tiếp có độ dài N được mô tả như sau:

Với một dãy các mã hex sau khi dịch ngược giả sử là "AB C0 EF 12" thì dãy các n-gram byte thu được là:

Bảng 3.1 Mô tả dãy các n-gram byte

1-gram	2-gram	3-gram	4-gram
AB	AB C0	AB C0 EF	AB C0 EF 12
C0	C0 EF	C0 EF 12	
EF	EF 12		
12			

Có thể nhận thấy rằng với độ dài n càng cao thì kích thước đặc trưng càng lớn. Đối với mã hex có 16 giá trị khác nhau như vậy không gian đặc trưng của 1-gram sẽ là 16^2=256 với 2-gram là 16^4=65536. Trong nội dung luận văn này tôi chủ yếu tập trung vào phương pháp chọn đặc trưng vì vậy các kết quả được thử nghiệm trên dãy 2-gram. Như vậy ở giai đoạn tiền xử lý các file dữ liệu mẫu được dịch ngược sang các mã hex và sau đó tiếp tục được trích rút ra các đặc trưng dựa vào phương pháp n-gram.

3.2.3. Tính tần số xuất hiện (Term Frequency)

Ở bước này tôi thực hiện tính toán tần số xuất hiện của mỗi n-gram byte khác nhau trên từng file dữ liệu mẫu. Các kết quả này được lưu như mỗi véc tơ đặc trưng và trước khi được đưa vào mô hình học sẽ được xử lý để trích ra một bộ đặc trưng tốt nhất.

Công thức để tính các giá trị TF được cho như sau:

$$\mathrm{tf}(t,d) = \frac{\mathrm{f}(t,d)}{\max\{\mathrm{f}(w,d) : w \in d\}}$$

Tần số xuất hiện của một mã n-gram byte trong một tập mẫu (file dữ liệu mẫu đã được đưa về dạng text) được tính bằng thương của số lần xuất hiện n-gram byte đó trong tập mẫu và số lần xuất hiện nhiều nhất của một n-gram byte bất kỳ trong tập mẫu đó (giá trị sẽ thuộc khoảng [0, 1])

- **f(t,d)** - số lần xuất hiện của một n-gram byte t trong tập mẫu d.
- **max{f(w,d):w∈d}** - số lần xuất hiện nhiều nhất của một n-gram byte bất kỳ trong tập mẫu.

3.3. Đề xuất giải pháp chọn đặc trưng cho thuật toán phân lớp

Trong phương pháp học máy có thể thấy rằng khi số lượng đặc trưng lớn sẽ làm giảm hiệu suất và đôi khi là chất lượng của mô hình học. Lượng

đặc trưng quá nhiều sẽ khiến cho quá trình huấn luyện và phân lớp dữ liệu tốn kém về mặt tài nguyên cũng như thời gian xử lý thậm chí nếu nhiều đặc trưng phổ biến sẽ đẫn đến dư thừa gây nhiễu và ảnh hướng đến chất lượng khi xây dựng mô hình, chính vì vậy bài toán đặt ra và cần thiết là làm sao loại bỏ được các đặc trưng gây nhiễu và chọn được một tập đặc trưng đại diện tốt nhất mà vẫn đảm bảo độ chính xác hiệu quả của mô hình dự đoán. Trong phương pháp n-gram nêu trên ta thấy sẽ có các đặc trưng mà tần số xuất hiện của chúng tương tự nhau trên 2 lớp vì vậy khi đưa vào mô hình học máy sẽ không đạt kết quả cao.

Phương pháp trích chọn đặc trưng mà tôi đề xuất mục đích là tìm ra tập các đặc trưng mà giá trị tần số xuất hiện trung bình của chúng trên 2 lớp cần phân chia có độ lệch lớn nhất. Cụ thể là các mã n-gram byte mà có tần số xuất hiện trên các tập của lớp này khác nhất với chính nó trên các tập của lớp còn lại.

3.3.1. Mô tả giải pháp

Gọi D là tập các đặc trưng có độ dài "d" phần tử là các mã n-gram byte. 2 lớp cần phân chia lớp thứ 1 có độ dài là "n" tập mẫu. Lớp thứ 2 có độ dài là "m" tập mẫu.

Gọi TF1[i] là tập chứa các giá trị tần số xuất hiện của đặc trưng D[i] $\in$ D; i ϵ [0,d] trên các mẫu dữ liệu thuộc lớp thứ 1. Mỗi TF1[i] với i ϵ [0,d] là một mảng chứa 'n' phần tử .

Gọi TF2[i] là tập chứa các giá trị tần số xuất hiện của đặc trưng D[i] $\in$ D; i ϵ [0,d] trên các mẫu dữ liệu thuộc lớp thứ 2. Mỗi TF2[i] với i ϵ [0,d] là một mảng chứa 'm' phần tử.

Bước 1: Với mỗi giá trị i ϵ [0,d] sắp xếp các phần tử trong TF1[i] và TF2[i] theo chiều giảm hoặc tăng dần

Bước 2: Sau khi thực hiện sắp xếp ở bước 1:

Với mỗi i $\in$ [0,d] ta thực hiện chia các phần tử trong TF1[i] tương ứng thành "k" đoạn liên tiếp bắt đầu từ phần tử đầu tiên, mỗi đoạn chứa C1 phần tử (số phần tử trong các đoạn có thể không bằng nhau).

Tương tự với mỗi i $\in$ [0,d] ta cũng thực hiện chia các phần tử trong TF2[i] tương ứng thành "k" đoạn liên tiếp bắt đầu từ phần tử đầu tiên, mỗi đoạn chứa C2 phần tử (số phần tử trong các đoạn có thể không bằng nhau)

Bước 3: Tính trung bình cộng tần số xuất hiện trên từng khoảng đã chia

trên mỗi TF1[i]; i ∈ [0,d] với mỗi TF1[i] ta thu được "k" giá trị TB1[i][j] với i ϵ [0,d]; j∈[0,k] là giá trị tần số xuất hiện trung bình của D[i] trên đoạn j của TF1[i]

Tương tự thực hiện tính trung bình cộng tần số xuất hiện trên từng khoảng đã chia trên mỗi TF2[i]; i ∈ [0,d] với mỗi TF2[i] ta thu được "k" giá trị TB2[i][j] với i ϵ [0,d]; j∈[0,k] là giá trị tần số xuất hiện trung bình của D[i] trên đoạn j của TF2[i]

Bước 4: Tính độ lệch tần số xuất hiện trung bình trên "k" đoạn đã chia của mỗi TF1[i] và TF2[i].

Gọi TB[i][j] là độ lệch trung bình của đặc trưng D[i] trên đoạn j của 2 tập TF1[i] và TF2[i] thì TB[i][j] được tính bằng trị tuyệt đối của phép trừ giữa TB1[i][j] và TB2[i][j] ta có:

$$TB[i][j] = | TB1[i][j] - TB2[i][j] |$$

(i ϵ [0,d]; j ϵ [0,k])

Với mỗi i trện đoạn [0,d] thực hiện tính các TB[i][j] của nó với mỗi j ∈ [0,k] trên 2 tập TF1[i] và TF2[i] tương ứng.

Bước 5: Với mỗi i ϵ [0,d] ta thực hiện tính độ lệch trung bình giá trị tần số của đặc trưng D[i] tương ứng trên toàn tập bằng cách tính sau:

$$DL[i] = \frac{\sum_{j=0}^{k} TB[i][j]}{k}$$

Trong đó DL[i] với i ∈[0,d] là độ lệch trung bình của đặc trưng D[i] trên 2 lớp cần phân chia.

Bước 6: Kết thúc bước 5 ta sẽ thu được kết quả độ lệch trung bình tần số xuất hiện của "d" đặc trưng ban đầu từ "d" đặc trưng này thực hiện chọn ra một bộ đặc trưng có giá trị độ lệch cao nhất để đưa vào xây dựng mô hình.

3.3.2. Ví dụ:

Bài toán: Giả sử cho 2 lớp với các đặc trưng là các mã hex 2-gram có số tập mẫu khác nhau và tần số xuất hiện như sau:

Bảng 3.2 Ví dụ tập tần số xuất hiện ban đầu của các đặc trưng thuộc lớp 1

Lớp 1	Đặc trưng	AA AA	BB BB	CC CC	DD DD
	Tập 1	0.5	0.7	1.0	0.2
	Tập 2	0.5	0.9	0	0.1
	Tập 3	0.6	0.5	1.0	0.2
	Tập 4	0.6	0.6	0	0.1

Bảng 3.3 Ví dụ tập tần số xuất hiện ban đầu của các đặc trưng thuộc lớp 2

Lớp 2	Đặc trưng	AA AA	BB BB	CC CC	DD DD
	Tập 1	0.6	0.2	0.5	0.1
	Tập 2	0.5	0.3	0.5	0.2
	Tập 3	0.6	0.3	0.5	0.7
	Tập 4	0.5	0.2	0.5	0.8
	Tập 5	0.6	0.3	0.5	0.3
	Tập 6	0.5	0.2	0.5	0.2

Bước 1: sau khi đã tính tần số xuất hiện của các đặc trưng trên mỗi tập mẫu thuộc 2 lớp thực hiện sắp xếp các giá trị theo chiều giảm dần ở mỗi đặc trưng trong cả 2 lớp như sau:

Bảng 3.4 Kết quả tập dữ liệu lớp 1 sau khi thực hiện sắp xếp

Lớp 1	Đặc trưng	AA AA	BB BB	CC CC	DD DD
		0.6	0.9	1.0	0.2
		0.6	0.7	1.0	0.2
		0.5	0.6	0	0.1
		0.5	0.5	0	0.1

Bảng 3.5 Kết quả tập dữ liệu lớp 2 sau khi thực hiện sắp xếp

Lớp 2	Đặc trưng	AA AA	BB BB	CC CC	DD DD
		0.6	0.3	0.5	0.8
		0.6	0.3	0.5	0.7
		0.6	0.3	0.5	0.3
		0.5	0.2	0.5	0.2
		0.5	0.2	0.5	0.2
		0.5	0.2	0.5	0.1

Bước 2: Tiến hành chia lớp 1 và lớp 2 mỗi lớp thành k đoạn. Giả sử ta chọn k=2 như vậy với lớp 1 mỗi đoạn sẽ có 2 giá trị tần số liên tiếp cạnh nhau, với lớp 2 mỗi đoạn có 3 giá trị tần số liên tiếp cạnh nhau.

Bảng 3.6 Kết quả tập dữ liệu lớp 1 sau khi thực hiện phân đoạn

Lớp 1	Đặc trưng	AA AA	BB BB	CC CC	DD DD
	Đoạn 1	0.6	0.9	1.0	0.2
		0.6	0.7	1.0	0.2
	Đoạn 2	0.5	0.6	0	0.1
		0.5	0.5	0	0.1

Bảng 3.7 Kết quả tập dữ liệu lớp 2 sau khi thực hiện phân đoạn

Lớp 2	Đặc trưng	AA AA	BB BB	CC CC	DD DD
	Đoạn 1	0.6	0.3	0.5	0.8
		0.6	0.3	0.5	0.7
		0.6	0.3	0.5	0.3
	Đoạn 2	0.5	0.2	0.5	0.2
		0.5	0.2	0.5	0.2
		0.5	0.2	0.5	0.1

Bước 3: Trên mỗi đoạn thực hiện tính ra giá trị trung bình cộng tần số xuất hiện của mỗi đặc trưng :

Bảng 3.10 giá trị trung bình cộng tần số xuất hiện của mỗi đặc trưng trên từng đoạn thuộc lớp 1

Lớp 1	Đặc trưng	AA AA	BB BB	CC CC	DD DD
TF trung bình trên từng đoạn	Đoạn 1	0.6	0.8	1.0	0.2
	Đoạn 2	0.5	0.55	0	0.1

Bảng 3.11 giá trị trung bình cộng tần số xuất hiện của mỗi đặc trưng trên từng đoạn thuộc lớp 2

Lớp 2	Đặc trưng	AA AA	BB BB	CC CC	DD DD
TF trung bình trên từng đoạn	Đoạn 1	0.6	0.3	0.5	0.6
	Đoạn 2	0.5	0.2	0.5	0.16

Bước 4: Thực hiện tính độ lệch của tần số xuất hiện giữa các đoạn của từng đặc trưng trên 2 lớp, bằng cách trừ lấy trị tuyệt đối 2 tần số trên cùng một đoạn của cùng 1 đặc trưng trên 2 lớp dữ liệu.

Bảng 3.12 Kết quả độ lệch tần số xuất hiện trên từng đoạn

Độ lệch TF trên từng đoạn	Đặc trưng	AA AA	BB BB	CC CC	DD DD
	Đoạn 1	0	0.5	0.5	0.4
	Đoạn 2	0	0.35	0.5	0.06

Bước 5: Từ kết quả độ lệch tần số xuất hiện có được trên các đoạn ở bước 4 tiến hành tính trung bình độ lệch tần số xuất hiện của mỗi đặc trưng trên toàn tập, bằng cách tính trung bình cộng độ lệch tần số xuất hiện trên các đoạn của mỗi đặc trưng để ra kết quả cuối cùng.

Bảng 3.13 Kết quả độ lệch tần số xuất hiện trên toàn tập dữ liệu

Độ lệch TF trên toàn dữ liệu	Đặc trưng	AA AA	BB BB	CC CC	DD DD
	Đoạn 1	0	0.425	0.5	0.23

Như vậy dễ thấy các đặc trưng có độ lệch cao nhất là "CC CC" và "BB BB" sau đó là "DD DD" ta có thể xác định được các đặc trưng này có sự sai khác lớn về tần số xuất hiện của 2 lớp đúng theo thứ tự.

Bước 6: Từ các kết quả có được thực hiện chọn ra T đặc trưng có độ lệch về giá trị tần số xuất hiện lớn nhất.

3.4. Xây dựng mô hình dự đoán dựa trên các thuật toán phân lớp

Sau khi chọn được tập đặc trưng dựa vào phương pháp đã trình bày phía trên, bước tiếp theo ở giai đoạn này là tiến hành đưa các giá trị tần số xuất hiện của các đặc trưng n-gram byte đã được chọn trên các tập mẫu ban đầu vào các thuật toán trong học máy để xây dựng mô hình dự đoán., các dữ liệu mẫu được gán nhãn thành 2 lớp mã độc và file bình thường. Giai đoạn này sử dụng các véc tơ đặc trưng như là dữ liệu huấn luyện đầu vào cho các thuật toán phân lớp dữ liệu [13,19] có thể áp dụng như: k láng giềng (KNN), Naive Bayes (NB), cây quyết định (DT), Rừng ngẫu nhiên (RF), máy véc tơ hỗ trợ (SVM)..Trong nghiên cứu này tôi lựa chọn 2 thuật toán tiêu biểu là cây quyết định và máy véc tơ hỗ trợ để xây dựng mô hình đự đoán phát hiện mã độc đồng thời các kết quả của 2 phương pháp sẽ được so sánh và đánh giá thông qua quá trình thực nghiệm.

CHƯƠNG 4 THỰC NGHIỆM VÀ ĐÁNH GIÁ

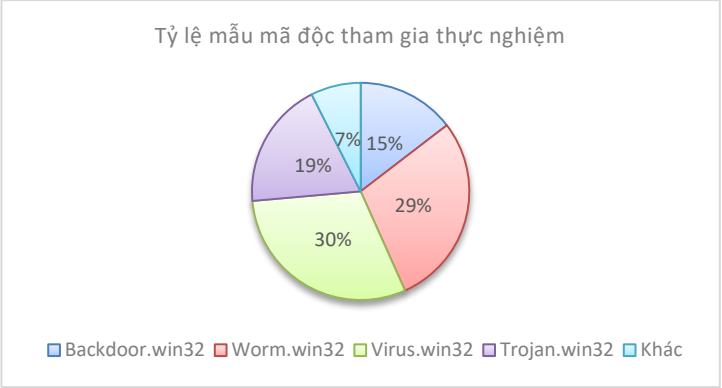
4.1. Dữ liệu thực nghiệm

Các kết quả của luận văn được thực nghiệm với đầu vào là 4698 file dữ liệu mẫu trong đó 2373 file mã thông thường và 2325 file mã độc, file mã độc được thu thập từ trang chủ "Vxheaven" [21] và các file mã thông thường là các file hệ thống thuộc kiểu file "PE" chủ yếu là các file ".exe" chạy trên trên hệ điều hành Windows số lượng các mẫu mã độc được mô tả như sau:

Bảng 4.1 Tỷ lệ số lượng từng mẫu mã độc tham gia thực nghiệm

Kiểu mã độc	Backdoor.win32	Worm.win32	Virus.win32	Trojan.win32	Khác
Số lượng mẫu	339	668	704	441	173

Trong đó tỷ lệ các mẫu mã độc được biểu diễn trực quan trong biểu đồ sau:



Hình 4.1 Biểu đồ mô tả tỷ lệ mẫu mã độc tham gia thực nghiệm

4.2. Chương trình thực nghiệm

Bảng 4.2 Thông tin về chương trình thực nghiệm

Môi trường thực nghiệm	- Processor: Intel(R) Core i5 – 2450M CPU @ 2.50GHz - Memory (Ram): 8.00 GB - System type: 64-bit Operating System, x64-based processor - Windows 10 Pro -2016

Ngôn ngữ sử dụng	Python
Công cụ lập trình (IDE)	PyCharm – JetBrains 5.0 Pro
Thư viện và các gói cài đặt chính	Scikit-learn, scipy, numpy,matplotlib,ipython, Pefile

4.3. Đánh giá dựa trên phương pháp ma trận nhầm lẫn

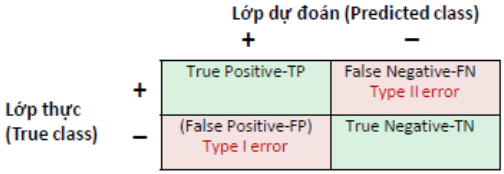
Một mô hình được đánh giá tốt nếu như tỉ lệ: TP (True Positive) và TN (True Negative) lớn đồng thời tỉ lệ FP (False Positive) và FN (False Negative) là nhỏ trong đó:

TP: Số lượng các mẫu thuộc lớp dương được phân loại chính xác vào lớp dương.

FP: Số lượng các mẫu không thuộc lớp dương bị phân loại nhầm vào lớp dương

TN: Số lượng các mẫu không thuộc lớp dương được phân loại đúng

FN: Số lượng các mẫu thuộc lớp dương bị phân loại nhầm vào các lớp không phải lớp dương.



Độ chính xác của mô hình được tính bằng công thức sau:

$$Acc=(TP+TN)/(TP+TN+FP+FN)$$

TPR (True positive rate) được tính bằng công thức sau:

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

FPR (False positive rate) được tính bằng công thức sau:

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

4.4. Kết quả thực nghiệm

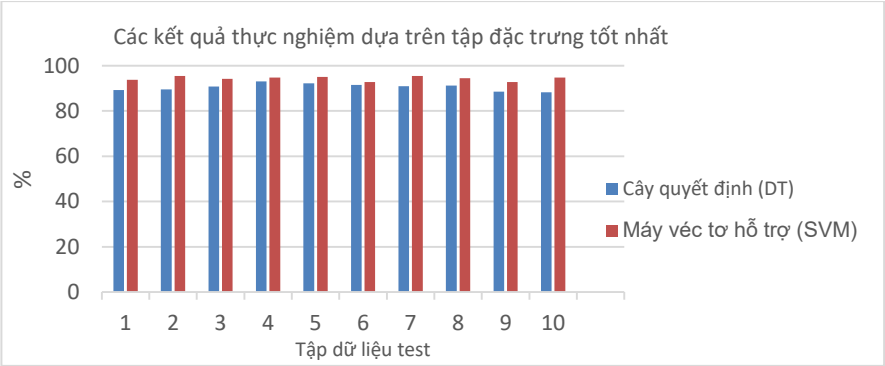
Tiến hành trích rút các đặc trưng 2-gram byte như vậy toàn bộ không gian đặc trưng sẽ là 16^4=65536 đặc trưng, sau đó áp dụng thuật toán đã đề xuất tôi thực hiện thu gọn và trích chọn ra 800 đặc trưng tốt nhất. Trong tổng số 4698

file thực hiện chọn ra 400 file test (trong đó 200 file mã độc và 200 file thông thường) các file còn lại độc lập với dữ liệu test được dùng làm dữ liệu huấn luyện. Tiến hành kiểm tra chéo trên các tập dữ liệu bằng cách chọn ngẫu nhiên 400 file test khác nhau với các file huấn luyện độc lập còn lại và thực hiện kiểm tra nhiều lần, các kết quả được đáng giá dựa trên phương pháp ma trận nhầm lẫn trong đó coi lớp dương (Positive) là mã độc , lớp âm (Negative) là file thông thường, các kết quả thu được mô tả trong bảng sau:

Bảng 4.3 Kết quả thực nghiệm trên các mô hình sử dụng tập đặc trưng tốt nhất

n-gram	Số đặc trưng	Mô hình	Độ chính xác	TPR	FPR
2	800	DT	90.75%	0.90	0.085
2	800	SVM	94.50%	0.945	0.055

Trong đó các kết quả chi tiết thực nghiệm trên các tập dữ liệu test khác nhau như sau:



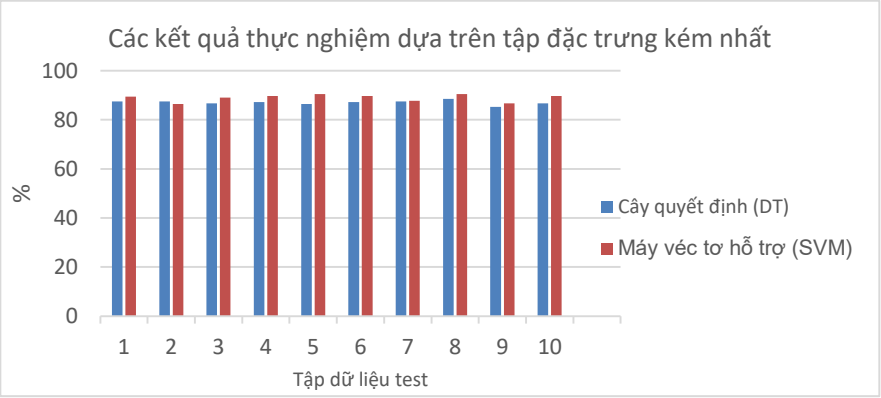
Hình 4.9 Kết quả độ chính xác thực nghiệm trên các tập dữ liệu test khác nhau sử dụng các đặc trưng được đánh giá tốt nhất

Để có thể so sánh được hiệu quả của giải pháp chọn đặc trưng đã đề xuất. Tôi tiến hành kiểm tra trên cùng tập dữ liệu và sử dụng 800 đặc trưng khác bị đánh giá kém nhất (có độ lệch tần số xuất hiện thấp nhất) để xây dựng mô hình học, các kết quả kiểm tra thu được như sau:

Bảng 4.4 Kết quả thực nghiệm trên các mô hình sử dụng tập đặc trưng kém nhất

n-gram	Số đặc trưng	Mô hình	Độ chính xác	TPR	FPR
2	800	DT	87.3%	0.855	0.11
2	800	SVM	88.5%	0.93	0.16

Trong đó các kết quả chi tiết thực nghiệm trên các tập dữ liệu test khác nhau như sau:



Hình 4.10 Kết quả độ chính xác thực nghiệm trên các tập dữ liệu test khác nhau sử dụng các đặc trưng bị đánh giá kém nhất

Nhận xét: Như vậy dựa vào các kết quả thực nghiệm có thể thấy tập đặc trưng được đánh giá cao mà thuật toán đã tìm ra cho kết quả tốt hơn rất nhiều so với tập đặc trưng bị đánh giá thấp.

KẾT LUẬN

Luận văn đã trình bày một hướng tiếp cận có hiệu quả trong đó ứng dụng kỹ thuật học máy vào việc nhận diện và phát hiện mã độc. Đồng thời đề xuất một giải pháp chọn đặc trưng để có thể gia tăng hiệu suất mà vẫn đảm bảo độ chính xác và hiệu quả của mô hình dự đoán, các kết quả được tiến hành thực nghiệm và đánh giá trên các tệp dữ liệu chạy trên nền Windows cho thấy với kỹ thuật chọn đặc trưng tôi đã đề xuất thì các đặc trưng tìm được cho kết quả mô hình dự đoán tốt hơn, đồng thời trong thực nghiệm này thuật toán học máy véc tơ hỗ trợ (SVM) tỏ ra hiệu quả hơn so với thuật toán học cây quyết định (DT). Với kết quả độ chính xác là 94.5% khi sử dụng thuật toán học SVM đây là một kết quả khá khả quan đối với yêu cầu cần giải quyết các bài toán phát hiện mã độc hiện nay.

❖ *Các kết quả chính đạt được của luận văn:*

- Tìm hiểu về mã độc các phương pháp phân tích, phân loại và phát hiện mã độc
- Trình bày cơ sở lý thuyết và thực nghiệm các thuật toán phân lớp trong kỹ thuật học máy giám sát bao gồm 2 thuật toán là cây quyết định và SVM
- Nghiên cứu và thực hiện việc ứng dụng kỹ thuật học máy vào giải quyết bài toán phát hiện mã độc.
- Đề xuất được giải pháp lựa chọn đặc trưng tốt nhất đảm bảo hiệu quả, hiệu suất của mô hình học máy trong bài toán phát hiện mã độc.
- Tiến hành thực nghiệm và đánh giá, so sánh các kết quả.

❖ *Hướng phát triển tiếp theo của nghiên cứu:*

Mở rộng phương pháp với nhiều hướng tiếp cận phân tích mã độc hơn như phân tích động, phân tích dựa trên kinh nghiệm từ đó phối hợp với các thuật toán học máy vào quá trình xây dựng các hệ thống có khả năng tự động nhận diện với đa dạng mã độc một cách chính xác, đồng thời tiếp tục nghiên cứu, phối hợp phát triển và áp dụng nhiều hơn các kỹ thuật trích chọn đặc trưng nhằm nâng cao hiệu quả hiệu suất của mô hình dự đoán.

DANH MỤC CÔNG TRÌNH KHOA HỌC CỦA TÁC GIẢ LIÊN QUAN ĐẾN LUẬN VĂN

1. Văn Trường Võ, Văn Vinh Nguyễn, Minh Đức Trịnh and Khánh Dương Lê. "*Đề xuất giải pháp trích chọn đặc trưng cho các thuật toán phân lớp dữ liệu trong kỹ thuật học máy giám sát và ứng dụng hiệu quả vào bài toán phát hiện mã độc".* Hội thảo lần thứ I, Một số vấn đề chọn lọc về an toàn an ninh thông tin (*1st Symposium on Information Security* - SoIS 2016). Đã được chấp nhận đăng trong kỷ yếu và trình bày tại hội thảo, pp. 35 -42

TÀI LIỆU THAM KHẢO

[1] Asaf Shabtai, Robert Moskovitch, Clint Feher, Shlomi Dolev and Yuval Elovici "Detecting unknown malicious code by applying classification techniques on OpCode patterns," Security Informatics 2012 1:1. doi:10.1186/2190-8532-1-1

[2] A. Shabtai, R. Moskovitch, Y. Elovici, C.Glezer,: "Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey," Information Security Technical Report 2009.

[3] Cai DM, Gokhale M, Theiler J "Comparison of feature selection and classification algorithms in identifying malicious executables,". Computational Statistics and Data Analysis 2007.

[4] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin "A Practical Guide to Support Vector Classification", Department of Computer Science National Taiwan University- Taipei 106- Taiwan, Last updated: May 19, 2016

[5] Digit Oktavianto, Iqbal Muhardianto,"Cuckoo Malware Analysis", Packt Publishing, 2013.

[6] D Krishna Sandeep Reddy - Arun K Pujari "N-gram analysis for computer virus detection," Springer-Verlag France 2006, doi 10.1007/s11416-006-0027-8

[7] Eldad Eilam, "Reversing-Secrets of Reverse Engineering", Wiley; 1 edition (April15, 2005)

[8] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani "An Introduction to Statistical Learning with Applications in R (Springer Texts in Statistics) ", by Springer Publishing, 2013

[9] Igor Santos, Yoseba K. Penya, Jaime Devesa and Pablo G. Bringas "n-grams-based file signatures for malware detection," Deusto Technological Foundation, Bilbao, Basque Country

[10] John Aycock, "Computer viruses and Malware (Advances in Information Security)", by Springer Publishing, 2006

[11] Madhu K. Shankarapani - Subbu Ramamoorthy - Ram S. Movva - Srinivas Mukkamala “Malware detection using assembly and API call sequences,” Springer-Verlag France 2010, doi 10.1007/s11416-010-0141-5

[12] Michael Sikorski and Andrew Honig, “Practical Malware Analysis”, No Starch Press, 1 edition (March 3, 2012)

[13] Peter Harrington, “Machine Learning in Action,” in Part 1 Classification, by Manning Publications, 2012 , pp. 1–129.

[14] Peter Szor, “The Art of Computer Virus Reseach and Defense”, Addison Wesley Professional, 2005.

[15] Schultz M, Eskin E, Zadok E, Stolfo S “Data mining methods for detection of new malicious executables,” Proc of the IEEE Symposium on Security and Privacy, IEEE Computer Society 2001.

[16] Smita Ranveer, Swapnaja Hiray, “Comparative Analysis of Feature Extraction Methods of Malware Detection,” International Journal of Computer Applications (0975 8887), Volume 120 - No. 5, June 2015

[17] Steve R. Gunn ,“Support Vector Machines for Classification and Regression”, Technical Report, Faculty of Engineering, Science and Mathematics School of Electronics and Computer Science, 1998

[18] Tom M. Mitchell, “Machine Learning” , by McGraw-Hill Science, 1997

[19] Trent Hauck, “scikit-learn Cookbook”, in Chapter 4 Classifying Data with scikit-learn, by Packt Publishing, 2014, pp. 119-157

[20] V. Vapnik. The Nature of Statistical Learning Theory. Springer-Verlag, New York, NY, 1995.

[21] VXheavens Website:url:http://vx.netlux.org.