**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

**AN TOÀN THÔNG TIN**

---------------o0o---------------

****

**Báo cáo đồ án môn học**

**Ứng dụng học máy vào bài toán nhận diện mã độc JavaScript**

**GVHD: Nguyễn Đức Toàn**

**SV: Trần Đức Sơn 20163571**

**HÀ NỘI, 4/2020**

Mục lục

[Đặt vấn đề 2](#_Toc40512405)

[Phân tích hành vi JavaScript malware 3](#_Toc40512406)

[Các hành vi cụ thể 3](#_Toc40512407)

[Lý thuyết cơ bản 7](#_Toc40512408)

[1. Xử lý dữ liệu 7](#_Toc40512409)

[2. Thuật toán Machine Learning 7](#_Toc40512410)

[Giải pháp đề xuất 11](#_Toc40512411)

[Công nghệ sử dụng 11](#_Toc40512412)

[Triển khai 14](#_Toc40512413)

[1. Sử dụng thuật toán Random Forests 14](#_Toc40512414)

[Kết luận 22](#_Toc40512415)

[Tài liệu tham khảo 22](#_Toc40512416)

# Đặt vấn đề

Công nghệ web hiện nay đang chiếm vị trí quan trọng trong tổng thể ngành công nghệ thông tin. Ở bất cứ đâu con người đều có thể truy cập vào web thông qua các ứng dụng web, làm mọi thứ từ truy cập thông tin đến giải trí, học tập và làm việc. Công cụ đóng vai trò quan trọng, giúp chúng ta tương tác với Web và Internet là Trình duyệt (browser). Ở bất cứ hệ điều hành nào, ta đều bắt gặp ít nhất một browser được cài sẵn giúp người dùng thuận tiện khi muốn truy cập vào website.

Tuy nhiên, song song với sự phát triển, đi lên của trình duyệt internet thì những hình thức tấn công vào máy tính người dùng thông qua nền tảng này cũng được phát triển, mở rộng và đang là mối nguy lớn. Chủ yếu, nó được lợi dụng bằng cách cài đặt một số những mã nguồn độc hại bằng JavaScript nhằm thực thi chuỗi lệnh từ đó thực hiện hành vi độc hại mà kẻ tấn công đã lên kịch bản trước.

# Phân tích hành vi JavaScript malware

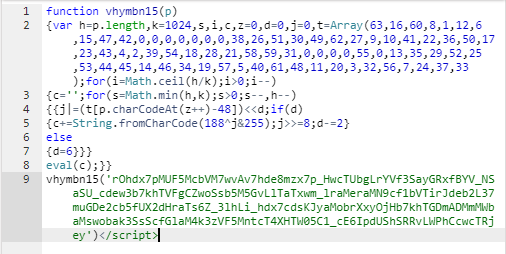
## Các hành vi cụ thể

1. **Chèn thuộc tính ẩn vào document**

Những mẫu JavaScript loại này được cài đặt trong những trang web không được xác thực, không an toàn hay sử dụng giao thức HTTP để giao tiếp. Nó có cấu trúc như một đoạn mã JavaScript thông thường nhưng thay vì thực hiện theo mục đích của lập trình viên website thì nó lại mang những hành động xấu. Cụ thể là chèn trực tiếp vào document HTML một thẻ <iframe> với attribute display:none hoặc cài đặt attribute.width, attribute.height tương đối nhỏ, làm cho người dùng không nhận ra.

Mục đích của hành vi này là đánh lừa người dùng click vào đối tượng, hoặc tương tác với đối tượng mà không chủ ý. Dẫn đến sinh lợi cho attacker (thông quan chạy quảng cáo) hoặc thực hiện những hành vi nguy hiểm hơn (chuyển hướng đến trang web độc hại hay tải mã độc về máy tính người dùng), …

Ví dụ một JavaScript malware chèn thẻ iframe ẩn vào document:



Hình minh họa: 1 Một mẫu JavaScript chèn thẻ iframe ẩn

Đoạn code sau khi giải mã (decode) là một câu lệnh document.write để ghi một thẻ iframe ẩn vào trong file HTML:



Hình minh họa: 2 Mã nguồn mẫu này sau khi giải mã

1. **Gọi hàm ẩn đối tượng trong document**

Những hàm ẩn này cũng có tác dụng tương tự như set lại thuộc tính ẩn. Tuy nhiên, do khả năng tương tác với mọi đối tượng của document nên có thể sử dụng để tương tác với tất cả các đối tượng trong document. Từ đó dễ dàng đặt lại thuộc tính cho đối tượng hơn.

1. **Thực hiện các câu lệnh jQuery**

Những câu lệnh JQuery có tính tương tác với các đối tượng trong document HTML, tác động đến nội dung file HTML, cấu trúc DOM, có thể lựa chọn những đối tượng trong DOM, trong CSS, hỗ trợ hiệu ứng, hoạt cảnh trong trang web, các phương thức HTML, chuyển đổi JSON, gọi hàm AJAX, các plug-ins, …

Những đoạn mã độc giả mạo những câu lệnh jQuery có tác động trực tiếp đến trang web mà người dùng truy cập đến, thường được inject vào các framework phổ biến để tạo trang web hiện nay như Wordpress, Joomla. Dưới đây là minh họa mã javascript JQuery trong các website Joomla và Wordpress



Hình minh họa: 3 Một mẫu giả dạng Jquery

Thường thì địa chỉ của những loại virus này thường có dạng:



Hình minh họa: 4 Đường dẫn trong biến base ở mẫu trên trỏ đến link mã độc JavaScript

Một đặc điểm nữa của những loại giả mạo Jquery là thường sẽ không sử dụng kèm kỹ thuật Obfuscated giống như những loại chèn thuộc tính ẩn hay chuyển hướng do mục đích là tối đa giả mạo câu lệnh JQuery và đánh lừa những chuyên viên phân tích mã độc.

1. **Chuyển hướng trang web hiện tại**

Một mã độc JavaScript chuyển hướng được chèn vào một website với mục đích chuyển hướng người dùng tới một website khác. Nó thường được tạo ra bởi các attacker nhằm đáp ứng một nội dung quảng cáo. Tuy nhiên, một vài mã độc chuyển hướng kiểu này có tác động nhiều hơn tới máy tính người dùng. Nó exploit (khai thác) một lỗ hổng trong máy tính của người dùng đang truy cập vào website, thông qua đoạn mã để cài những loại mã độc mà không được bảo vệ ở phía người dùng, hoặc lợi dụng để cài đặt mã độc cùng loại chuyển hướng mà rất khó có thể loại bỏ trực tiếp khỏi máy người dùng.

Các phương thức tấn công của loại này:

* Chèn vào file trong trang web người dùng: ví dụ trong WordPress, nó được chèn vào các theme file. Thường được tìm thấy trong các theme header
* Chèn vào trang hoặc post
* Chèn vào các Widgets
* Chèn vào các file .htaccess dùng để cấu hình máy chủ web Apache, hoặc dùng quy định đường dẫn tới server mà tập tin của trang web được kết nối tới
* Ad networks: các mạng quảng cáo được chấp nhận trong mạng, kẻ tấn công có thể lợi dụng nó để giả mạo hoặc chèn vào mạng một loại quảng cáo độc hại do hệ thống không có (hoặc ít) cơ chế để kiểm tra và quét.

1. **Mã hóa file mã nguồn js hoặc sử dụng kỹ thuật Obfuscated javascript**

Với mục đích ẩn giấu mã nguồn, không cho những người không có kỹ năng về phân tích mã nguồn có thể đọc được, kỹ thuật obfuscated ra đời. Tuy nhiên, vì đặc tính che dấu mã nguồn này nên đa số các javascript malware lợi dụng để che dấu mã nguồn thật đằng sau một chuỗi mã hóa, khiến cho những người làm về phân tích mã nguồn không thể trực tiếp tìm ra được đoạn code mã độc

Các dạng Obfuscator trong javascript:

* **Randomization Obfuscation**: Ở dạng này, một số các element được chèn vào (ví dụ comment, những câu lệnh thực thi không liên quan, ..) mà không làm thay đổi ngữ nghĩa của đoạn mã để nhằm mục đích tạo nên sự ngẫu nhiên của các chữ cái trong mã nguồn được sử dụng, tính ngẫu nhiên trong comment và sử dụng các biến, đặt tên ngẫu nhiên
* **Data Obfuscation**: Ý tưởng chính là chuyển giá trị biến hoặc giá trị hằng thành kết quả có thể tính toán được của một hoặc nhiều biến khác. Có 2 kỹ thuật chính được sử dụng trong mục này là string splitting và keyword substitution. String splitting là chuyển đổi chuỗi thành kết quả của tổng nhiều chuỗi con. Keyword substitution là ánh xạ một biến đến một keyword từ đó thực thi bằng cách gọi biến đó như khi gọi đến keyword.
* **Encoding Obfuscation**: Có 3 cách để encode mã gốc: một là chuyển code thành ký tự ASCII, Unicode hoặc chuỗi hexadecimal được đã được escape. Phương thức thứ 2 là sử dụng hàm encode được định nghĩa, kẻ tấn công sử dụng hàm encoding này để tạo ra obfuscated code và khi thực thi thì sẽ gọi hàm decode trước, giải mã code sau đó mới triển khai. Ngoài ra, một vài chuẩn mã hóa và giải mã có thể được lợi dụng trong việc mã hóa code Javascript: ví dụ Jscript.Encode của Microsoft hỗ trợ mã hóa code JavaScript.
* **Logic Obfuscation**: Loại này lợi dụng luồng thực thi của JavaScript để thay đổi cấu trúc logic mà không ảnh hưởng đến ngữ nghĩa của code. Có 2 cách để thực hiện dạng này là chèn thêm một vài câu lệnh vớ vẩn hoặc không theo logic, không phụ thuộc vào hàm,. .. Một cách khác là thêm hoặc thay đổi một vài cấu trúc if…else, switch..case, for, while, …

Việc sử dụng kết hợp các phương pháp được nêu có thể gây nên khó khăn cho các nhà phân tích, điều này là một cách lý tưởng để kẻ tấn công thực hiện hành vi độc hại đồng thời cũng che dấu bản thân gọi sự phát hiện của hệ thống hoặc chuyên viên về an ninh an toàn.

Ví dụ về một dạng Obfuscator:



Hình minh họa: 5 Mã nguồn phức tạp sau khi đã được Obfuscator

1. **Theo dõi hành vi và vị trí con trỏ chuột trên trình duyệt**

Hành vi: Đoạn code JS tìm kiếm vị trí hiện tại của con trỏ chuột người dùng, sau đó tạo một vùng hoặc một thẻ tại ví trí đó, thẻ đó sẽ được tạo lại mỗi khi người dùng di chuyển con trỏ chuột

1. **Con đường lây lan**

Có 7 con đường lây lan chính cho Javascript malware

* Bị inject và các trang web hợp pháp
* Thông qua thẻ iframe ẩn
* Inject thông qua mạng lưới quảng cáo online
* Thông qua việc download bởi các drive độc hại: thường sử dụng file JavaScript để chạy malware infection
* Malicious JavaScript attachments: chạy thông qua chương trình windows và có thể kích hoạt lây lan bên ngoài trình duyệt
* Kích hoạt tải xuống thông qua việc inject một JavaScript độc hại khác
* Add-ons và plugins trên trình duyệt
* Giả mạo thông điệp pop-up của phần mềm

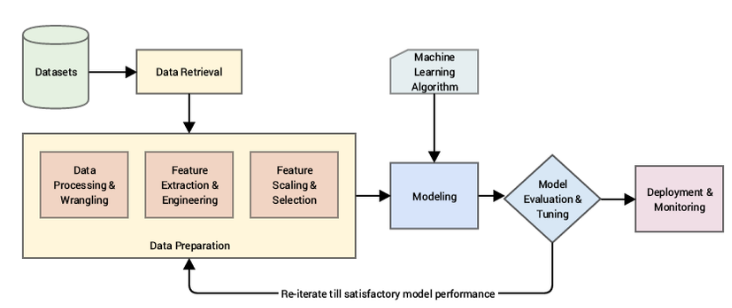
# Lý thuyết cơ bản

## Xử lý dữ liệu

Vai trò của xử lý dữ liệu:

Bất kỳ bài toán học máy nào đều phải trải qua các bước từ dữ liệu thô thu được, sử dụng các kỹ thuật để sắp xếp, xử lý, thiết kế các đặc trưng (feature) và thuộc tính có ý nghĩa từ dữ liệu này. Sau đó mới sử dụng các mô hình thống kê và học máy để xây dựng và giải quyết các yêu cầu đặt ra.

Mô hình xử lý chung cho các bài toán học máy dựa theo mô hình tiêu chuẩn công nghiệp CRISP-DM



Hình minh họa: 6 Mô hình cơ bản về quá trình làm một project học máy

Nếu chỉ giữ nguyên dữ liệu thô mà đưa luôn vào mô hình xử lý thì hiệu quả đạt được của mô hình sẽ không cao.

Với mỗi vấn đề cụ thể luôn có những đặc trưng quan trọng quyết định đến hiệu suất hệ thống. Feature Engineering luôn chiếm tới 70% thời gian trong quá trình xây dựng mô hình.

Xử lý dữ liệu hay cụ thể là trích chọn đặc trưng là quá trình biến đổi dữ liệu thành các đặc trưng đóng vai trò là đầu vào cho các mô hình học máy. Các đặc trưng xử lý tốt sẽ nâng cao hiệu suất mô hình. Tùy vào những dự án khác nhau mà những đặc trưng được lựa chọn cũng sẽ khác nhau.

Đối với bài toán JavaScript malware, đặc trưng được trích xuất dựa vào các hành động thực thi của mã nguồn, những đặc điểm thu được khi phân tích tĩnh mã nguồn và phân tích động hành vi thông qua quá trình giả lập và thực thi

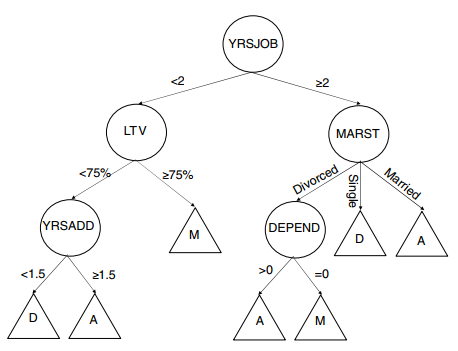
## Thuật toán Machine Learning

1. **Thuật toán Decision Trees**

Cây quyết định là thuật toán cơ bản trong những thuật toán học máy, được sử dụng trong cả 2 mô hình phân loại và hồi quy. Nó chủ yếu đề cập đến mô hình hình phân cấp của các quyết định và kết quả của chúng. Người xây dựng dựa vào kết quả của cây quyết định để ra quyết định hoặc xác định chiến lược để đạt được kết quả hay mục tiêu.

Khi cây quyết định được sử dụng cho hoạt động phân loại thì được gọi là Classification Trees. Nó được dùng rất nhiều trong hoạt động phân lớp, phân loại trong mô hình học có giám sát của học máy. Nó khá hữu dụng trong kỹ thuật thăm dò (exploratory technique) tuy nhiên không thay thế được các phương pháp thống kê truyền thống, hoặc các phương pháp khác sử dụng trong thống kê hoặc dự đoán.

Hình minh họa cây quyết định:



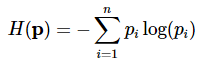
Hình minh họa: 7 Cây quyết định

Hình trên mô tả một cây quyết định cơ bản dùng trong việc tính toán khách hàng tiềm năng trong lĩnh vực vay thế chấp của ngân hàng. Một người muốn vay thế chấp cần điền đầy đủ thông tin về (DEPEND – người phụ thuộc), thuế (LTV), tình trạng hôn nhân (MARST), thu nhập bình quân đầu người (PAYINC), chỉ số tín nhiệm (interest rate – RATE), thời gian (số năm) ở tại địa chỉ hiện tại (YRSADD), số năm làm công việc hiện tại (YRSJOB). Cây quyết định dựa vào các chỉ số đó để phân loại thành 3 kết quả: Approved (A) – được phê duyệt, Denied (D) – bị từ chối, Manual underwriting (M) – Khai báo thêm.

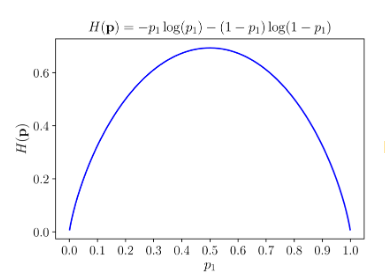
**Một số thuật toán sinh cây quyết định thường gặp**

**Thuật toán ID3**

* Là thuật toán cây quyết định đơn giản, trong đó sử dụng các thông tin input đầu vào như những tiêu chí phân chia, thứ tự các tiêu chí này được xem xét thông qua các công thức số học hoặc theo một tiêu chuẩn nào đó. Tại mỗi bước chọn tương ứng với một tiêu chí, sẽ chọn ra một thuộc tính. Với mỗi thuộc tính được chọn, ta chia dữ liệu vào các child node tương ứng với giá trị thuộc tính đó, sau đó tiếp tục áp dụng phương pháp này cho mỗi child node tiếp theo. Các bước phân chia diễn ra cho đến khi tại các node lá của cây quyết định, tập dữ liệu tương ứng chỉ cho ra kết quả đầu ra là một output xác định. Hoặc dựa vào điều kiện dừng để đặt giới hạn phân chia tránh trường hợp overfitting hoặc cây thu được quá dài.
* Để phân chia dữ liệu vào các node hiệu quả, ta lợi dụng đặc tính của hàm entropy biểu diễn độ phân tán của dữ liệu. Trong trường hợp dữ liệu tại một node trong cây quyết định là vẩn đục (impurity – từ tập dữ liệu con phân chia này có nhiều hơn 1 output thu được) thì giá trị entropy sẽ gần bằng 0.5, còn lại nếu node là tinh khiết (purity – dữ liệu con được phân chia trong node có chỉ 1 output), hàm entropy sẽ cho ra kết quả gần với 0 hoặc 1.



Hình minh họa: 8 Hàm entropy



Hình minh họa: 9 Đồ thị của hàm entropy với n = 2

**Điều kiện dừng của thuật toán**

* Nếu không có điều kiện dừng, tree có thể sẽ rất phức tạp với không gian tập mẫu lớn. Vì theo yêu cầu của thuật toán này thì phải phân chia tất cả các node chưa tinh khiết thành các node tinh khiết.
* Điều này dẫn đến mô hình có thể bị overfitting, hoặc có thể rất phức tạp để tính toán dẫn đến thời gian tính toán lớn. Do đó người ta quy định một số luật hoặc điều kiện để không tiếp tục phân chia các node nữa và coi nó như một leaf node.
  + nếu node đó có entropy bằng 0, tức mọi điểm trong node đều thuộc một class.
  + nếu node đó có số phần tử nhỏ hơn một ngưỡng nào đó. Trong trường hợp này, ta chấp nhận có một số điểm bị phân lớp sai để tránh overfitting. Class cho leaf node này có thể được xác định dựa trên class chiếm đa số trong node.
  + nếu khoảng cách từ node đó đến root node đạt tới một giá trị nào đó. Việc hạn chế chiều sâu của tree này làm giảm độ phức tạp của tree và phần nào giúp tránh overfitting.
  + nếu tổng số leaf node vượt quá một ngưỡng nào đó.
  + nếu việc phân chia node đó không làm giảm entropy quá nhiều (information gain nhỏ hơn một ngưỡng nào đó).

1. **Thuật toán random forests**

Random Forests được sáng tạo dựa trên thuật toán Cây quyết định đã được trình bày ở trên, trong đó có cải tiến về hiệu quả ra quyết định và khắc phục được những hạn chế của Cây quyết định như Overfitting.

Random forest là một thuật toán học có giám sát, được sử dụng trong cả 2 lĩnh vực là phân lớp và hồi quy. Nó khá linh hoạt và dễ sử dụng. Một forest là một tập bao gồm nhiều cây. Random forest được tạo thành từ những cây quyết định được lựa chọn ngẫu nhiên trong tập dữ liệu mẫu và ra quyết định từ một cây, giải pháp hợp lý nhất được lựa chọn từ các cây đó. Độ chính xác của thuật toán này phụ thuộc rất nhiều vào feature mà ta lựa chọn.

Random forests được ứng dụng trong nhiều lĩnh vực, như máy gợi ý, phân loại ảnh và lựa chọn đặc trưng. Nó được sử dụng để phân loại người đi vay tiềm năng, xác định hoạt động phạm tội và phán đoán bệnh tật. Nó thuộc họ thuật toán Boruta, dùng để lựa chọn những đặc trưng quan trọng trong một tập dữ liệu.

Để hiểu thuật toán, ta nghiên cứu một nhu cầu của con người. Giả sử như bạn muốn đi du lịch đến những nơi mà bạn thích. Vậy bạn làm gì để tìm được những nơi đó? Bạn có thể tìm kiếm online, đọc các bài review trên các blog du lịch hoặc hỏi bạn bè.

Giả sử bạn quyết định đi hỏi những người bạn của mình và nói với họ về những trải nghiệm trong những lần đi du lịch trước đây đến nhiều nơi. Bạn sẽ có được một vài lời khuyên từ bạn bè. Giờ bạn phải lập danh sách các nơi tham khảo. Sau đó yêu cầu họ chọn ra một nơi tốt nhất trong số chúng. Nơi có số lượt bình chọn nhiều nhất thì sẽ là lựa chọn cuối cùng để đi du lịch.

Trong quá trình quyết định, có 2 nội dung. Một là khi hỏi bạn bè của bạn về những trải nghiệm du lịch cá nhân của họ và muốn họ cho lời khuyên về những nơi mà họ đã từng đến. Phần này giống như sử dụng thuật toán cây quyết định. Mỗi một người bạn có một lựa chọn về những nơi mà họ đến trước đó.

Phần 2, sau khi đã thu được một số gợi ý từ bạn bè, bạn sẽ sử dụng phương pháp bầu chọn để tìm ra nơi được vote nhiều nhất, đây được coi như thuật toán random forests

Về mặt kỹ thuật nó là một phương pháp tập hợp (bao gồm chia để trị) của những cây quyết định được tạo thành từ tập mẫu được chia ra ngẫu nhiên. Tập những cây quyết định dùng cho phân loại này được biết đến như một forest (rừng cây). Những cây riêng lẻ được tạo thành từ các thuộc tính như thông tin đạt được, tần suất đạt được, giá trị Gini cho mỗi thuộc tính. Mỗi cây tương ứng một mẫu ngẫu nhiên. Trong một bài toán phân lớp, mỗi cây bình chọn và lớp phổ biến nhất được lựa chọn sẽ là kết quả cuối cùng. Nó đơn giản và mạnh mẽ nếu so với các thuật toán phân lớp phi tuyến tính khác.

Các bước hoạt động:

- Lựa chọn mẫu ngẫu nghiên từ tập mẫu đã cho

- Xây dựng cây quyết định cho mỗi mẫu và tìm kết quả đầu ra từ mỗi decision tree

- Tiến hành một hoạt động bầu chọn với mỗi một cây quyết định

- Lựa chọn kết quả bầu chọn với số vote cao nhất làm giá trị đầu ra

# Giải pháp đề xuất

1. Thử nghiệm với thuật toán Machine Learning

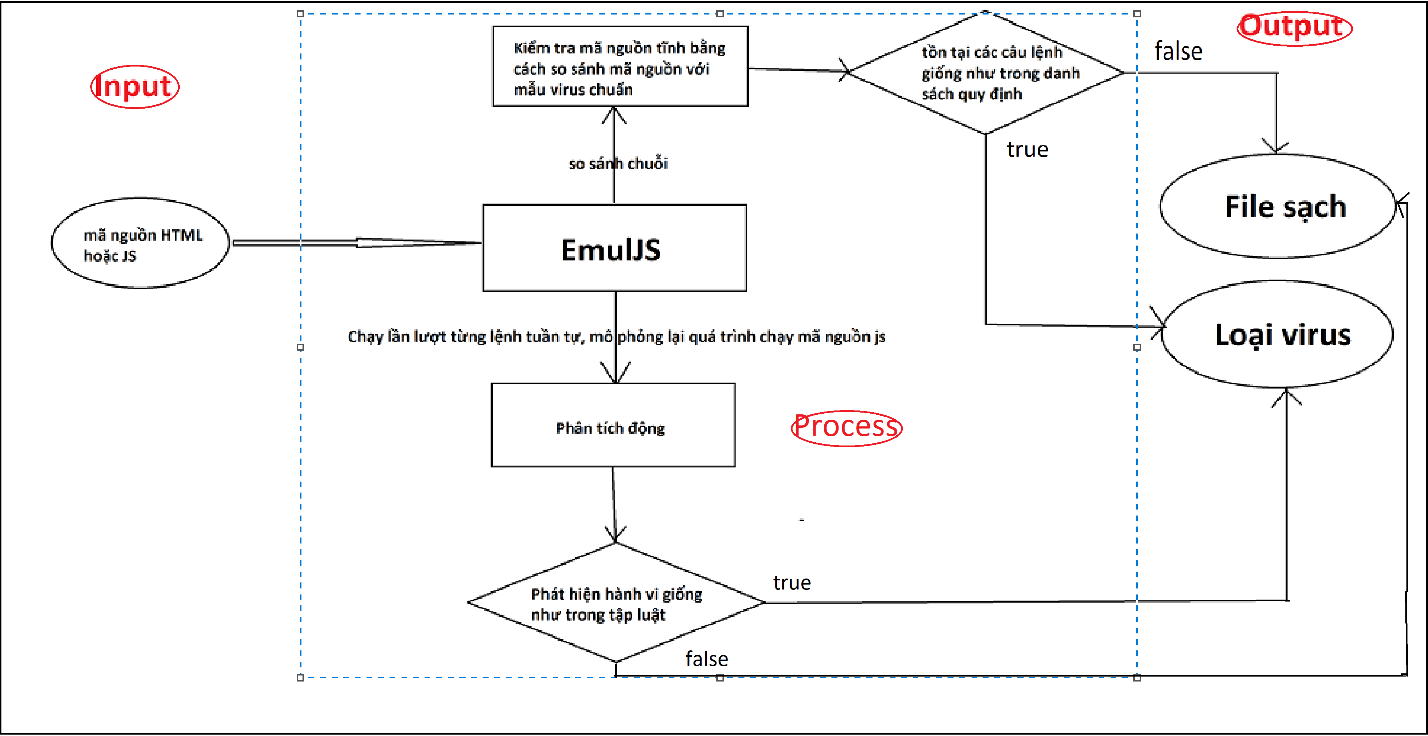
# Công nghệ sử dụng

1. **Giới thiệu về Engine EmulJS**

Engine EmulJS hiện đang được triển khai để nhận diện mẫu virus trên VirusTotal của Bkav. Công cụ này sử dụng kết hợp 2 phương pháp phát hiện code tĩnh và phân tích code động để tìm hành vi độc hại, từ đó sẽ trả về kết quả loại virus tùy theo hành vi phân tích được

Hiện tại em đang chỉnh sửa công cụ này để thay vì trả về kết quả virus như bình thường thì sẽ chỉ có tác dụng chạy động virus để thu thập những hành vi đặc trưng của virus từ đó lấy được dữ liệu output làm đầu vào cho mô hình học máy

Luồng xử lý của EmulJS:



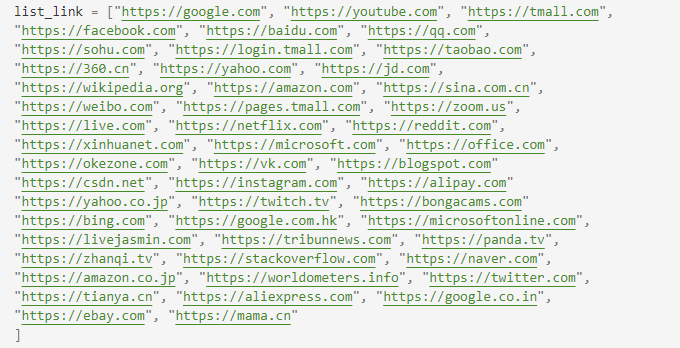
Có 2 hướng đi cho chỉnh sửa công cụ này

* **Trả về những nội dung làm đầu vào cho học máy – Trình bày trong báo cáo này**
* Xử lý và chuẩn hóa mã nguồn dạng text để làm đầu vào cho các mạng nơ ron.

1. **Giới thiệu về Tool Crawl mẫu HTML sạch**

Công cụ được viết bằng ngôn ngữ Python và sử dụng các thư viện làm việc với trang web và hỗ trợ crawl dữ liệu về như: urllib (download các trang web theo định dạng HTML), BeautifulSoup, time, os

Nguồn link ban đầu bao gồm:



Trong quá trình duyệt trang web, nếu thấy link Website khác với danh sách hoặc không nằm trong những link đã download thì sẽ thêm vào list để download tiếp. Tạo một list lưu danh sách những trang web đã crawl dữ liệu, danh sách sẽ tự động refresh lại về rỗng khi kích thước lớn hơn 2000 link. Có thể điều chỉnh kích thước nếu thực tế không hiệu quả.

Sau khi cho chạy lệnh này bằng python, mã nguồn HTML của các trang web được duyệt sẽ được lưu vào một thư mục làm đầu vào cho Tool Heuristic FS để trích xuất thông tin cần thiết ra file text

1. **Giới thiệu về công cụ Heuristic FS**

Công cụ sử dụng dll được tạo ra từ ***emulJS*** để quét offline một tập mẫu JavaScript đầu vào. Có thể tùy chỉnh đầu ra khi chạy công cụ bằng cách sửa mã nguồn của dll EmulJS

1. **Giới thiệu về nguồn lấy mẫu JavaScript và công cụ xử lý liên quan**

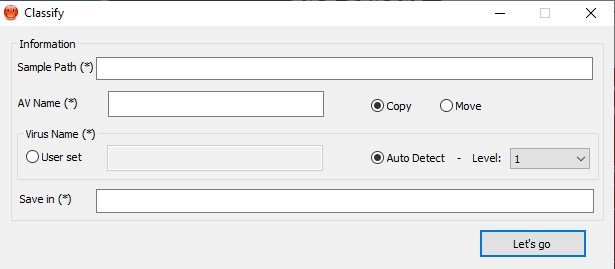
Nguồn lấy mẫu: VirusTotal

* VirusTotal là một dịch vụ miễn phí phân tích tập tin và URL nghi ngờ, nó có chứa một kho dữ liệu bao gồm rất nhiều mẫu JavaScript malware. Mỗi engine dùng để phân tích được hỗ trợ bởi cộng đồng các công cụ phân tích và diệt mã độc nổi tiếng trên thế giới. Mỗi một công ty cung cấp dịch vụ như vậy đều được cấp một tài khoản có thể truy cập và tải về kho tập mẫu của VirusTotal chứa nội dung Virus và một file log chứa thông tin các engine phát hiện.
* Các trang web được truy cập nhiều

Quy trình thu thập và phân loại mẫu:

* Sử dụng công cụ FileZila Client có cung cấp tài khoản, mật khẩu và địa chỉ host để lấy tập mẫu. Tập mẫu thu được sẽ bao gồm tổng hợp của file html + javascript + (có thể có) VBScript.
* Ở đây chỉ xét mã javascript do vậy sẽ loại bỏ đi những file chỉ thuần HTML và các file chứa mã VBScript. Để làm được điều này ta sử dụng công cụ Find VirusTotal Sample có chứa năng phân loại các mẫu Virus theo các từ khóa: ví dụ Microsoft thì sẽ phân loại theo tên virus được nhận diện dựa theo engine của Microsoft trên VirusTotal.

Giao diện của Tools:



Kết quả trả về là tập mẫu đã được phân loại và file log chứa thông tin về số lượng các loại virus liệt kê theo tên của Microsoft.

* Tiếp theo sẽ sử dụng Tool: ***scanfolder.py*** để lọc ra những loại virus nào là Javascript, VB và html sau đó chỉ lấy Javascript làm mẫu.
* Tên của Virus được lấy về thường sẽ đặt theo mã hash sha256 của file virus.

Nguồn lấy mẫu sạch Javascript:

* Sử dụng công cụ ***crawler\_ver1.py*** để download các file html với đường link cho trước để làm mẫu. Những file HTML này sau đó sẽ đưa vào engine EmulJS để lọc lấy code JavaScript, loại bỏ các đoạn mã HTML dư thừa
* Để chuẩn hóa, sử dụng công cụ ***find\_sample.py*** để tìm kiếm trong một đường dẫn tên file có mã sha256 được chỉ định.

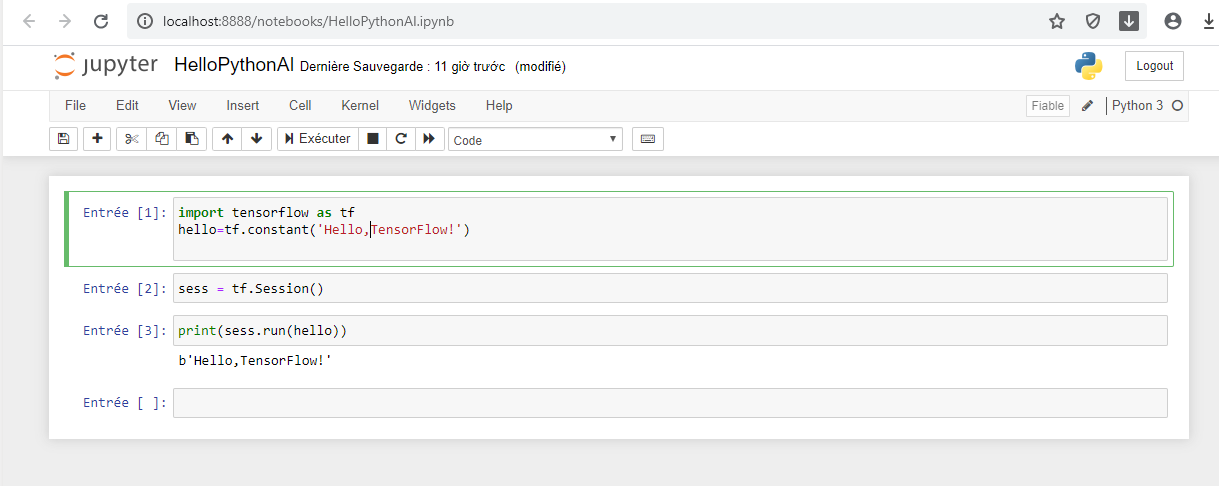
1. **Giới thiệu về Jupyter Notebook**

Jupyter Notebook được cài sẵn trong Anaconda cung cấp một giao diện sử dụng ngôn ngữ python được cài sẵn trong máy để chạy code. Nó có ưu điểm là giao diện trực quan, chạy code theo từng dòng lệnh, có thể lưu lại các checkpoint, …

Jupyter Notebook được sử dụng là bản được cài kèm cùng với bản Anaconda 3 mới nhất.

Sử dụng Jupyter Notebook trong triển khai mô hình học máy hoặc xây dựng mạng neural giúp tiết kiệm thời gian, linh hoạt do có khả năng sử dụng lại câu lệnh đã viết từ trước, trực quan và dễ dàng quan sát kết quả với giao diện thân thiện.

Giao diện trực quan của Jupyter Notebook

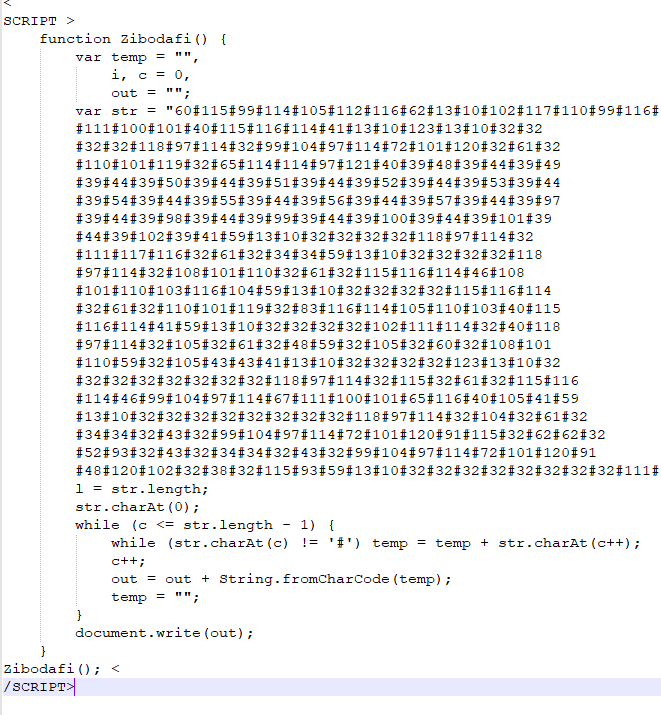


# Triển khai

## Sử dụng thuật toán Random Forests

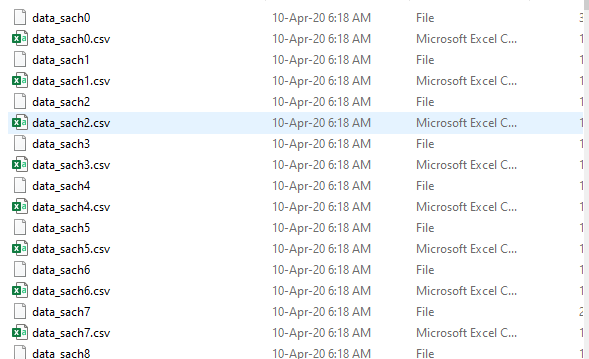
**Các đặc trưng ban đầu lựa chọn:**

* Dựa vào quá trình rà soát, phân tích mẫu virus js lựa chọn một số giá trị thuộc tính liên quan đến code js:
  + Entropy của mã nguồn js: Tính entropy của đoạn code ban đầu
  + Ký tự xuất hiện nhiều nhất trong code: Dựa theo thực tế những virus thường hay sử dụng mã hóa, và trong đoạn code mã hóa thì có tồn tại ký tự đặc biệt xuất hiện nhiều hơn các ký tự khác. Ví dụ:

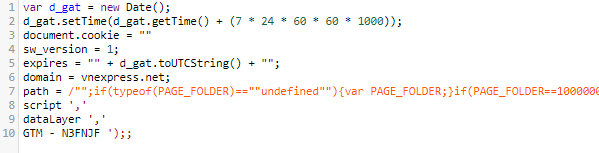


Đoạn mã sử dụng ký tự ‘#’ nhiều hơn các ký tự khác do chuỗi str chứa đoạn được encode, sau đó sẽ được giải mã bằng hàm fromCharCode()

* + Comment style dạng ‘<!--‘ , ‘/-->’ : Dạng comment style này hay xuất hiện ở những đoạn code js cũ, trình duyệt dễ dàng tưởng nhầm mà thực thi câu lệnh bên trong đoạn comment. Đếm giá trị số lần xuất hiện comment style dạng này
  + Một số hàm có thể sử dụng để viết js virus: Đếm giá trị số lần gọi của hàm và lưu lại giá trị đó
    - Document.Write()
    - Eval()
    - String.fromCharCode()
    - getUserAgent()
    - indexOf()
    - assign()
    - split()
    - replace()
    - concat()
    - unescape()
    - Window.setTimeOut()
    - Window.setInterval()
    - setCookie()
    - getCookie()
    - Element.replaceChild()
    - Element.changeAttribute()
    - Document.addEventListener()
    - createXMLHTTPRequest()
    - Date. ObjectToGMTString()
    - Element.appendChild()
    - Element.innerHTML()
* Sử dụng tool EmulJS: tool này hỗ trợ chạy giả lập động code JS. Em đã sửa phần code của tool để mỗi mẫu js quét được thì nó sẽ trả về các thuộc tính kể trên vào một file csv. Đối với những file sạch, đính kèm thêm một cột có giá trị là 0 để biểu thị không phải là virus, là 1 đối với những file virus (nguồn github, chắc chắn là virus)
* Kết quả sau khi xử lý:



* Các data được export ra file csv, tương ứng với một mẫu data là một mẫu js đi kèm tương ứng, để rà soát sau khi có kết quả train.

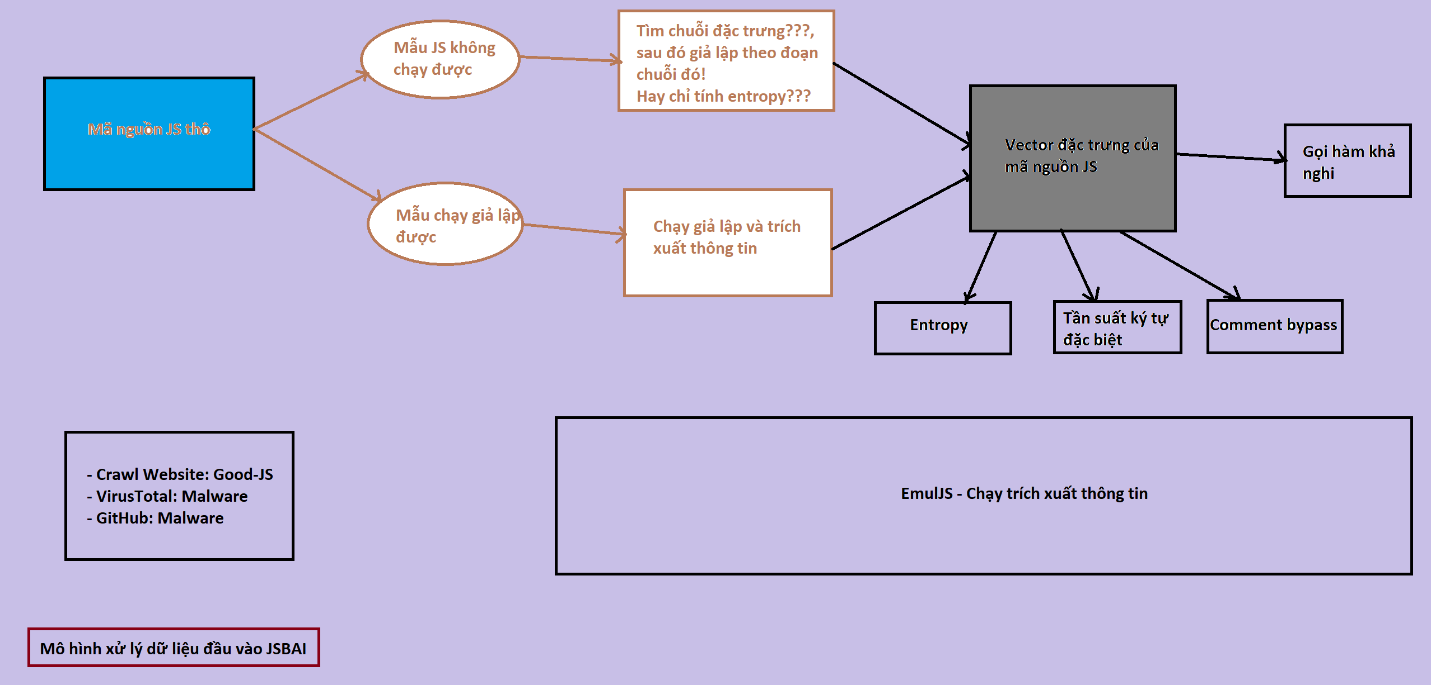


* Ví dụ một mẫu data(có 30 cột lưu giá trị thu được và 1 cột output:

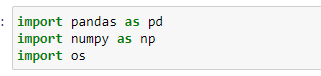


**Sử dụng Thuật toán Random Forests tìm hiểu phần trước để train thử và xem xét độ chính xác**

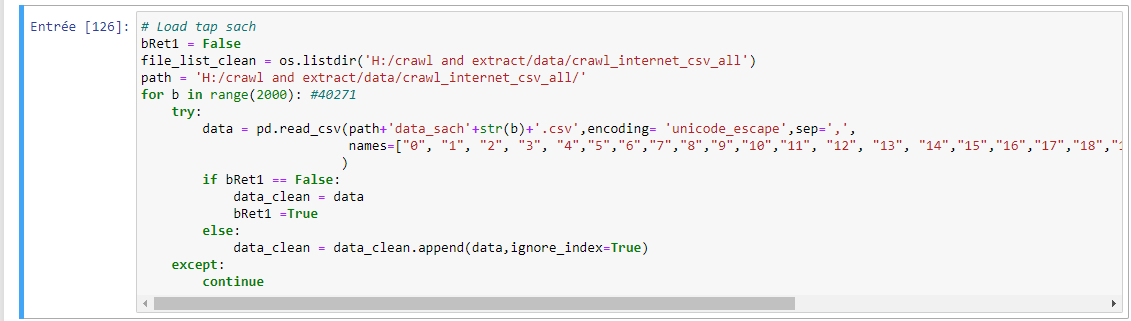
* Input gồm 8000 mẫu js sạch, crawl từ trang web thông dụng (nguồn Alexa), 8000 mẫu js virus, lấy từ nguồn github
* Tập dữ liệu train gồm (8000 mẫu sạch, 8000 mẫu virus)
* Quy trình xử lý mẫu:



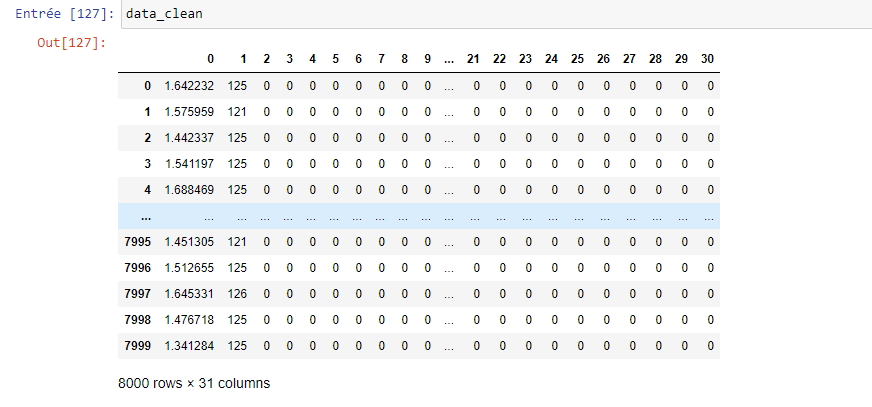
* Dữ liệu thu trích xuất ra file csv được sử dụng làm đầu vào cho thuật toán Random Forests
* Mã nguồn:
  + Import thư viện:



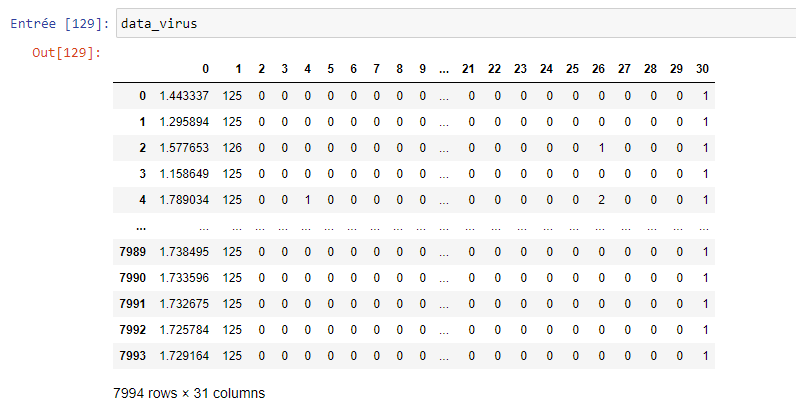
* + Load dữ liệu



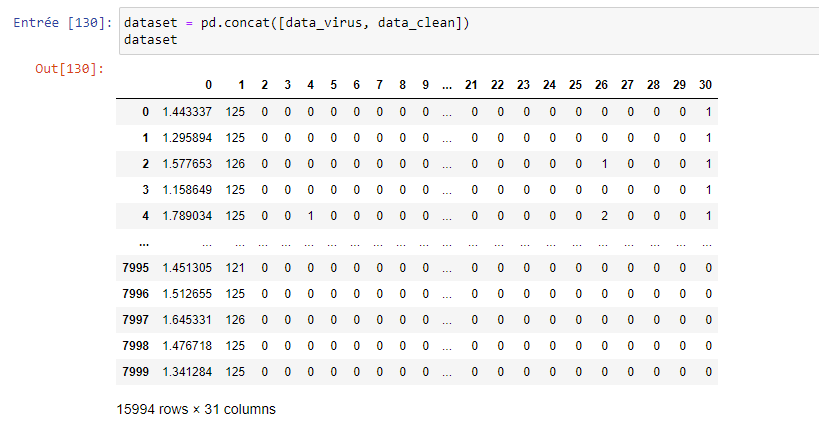
* + Tập sạch:



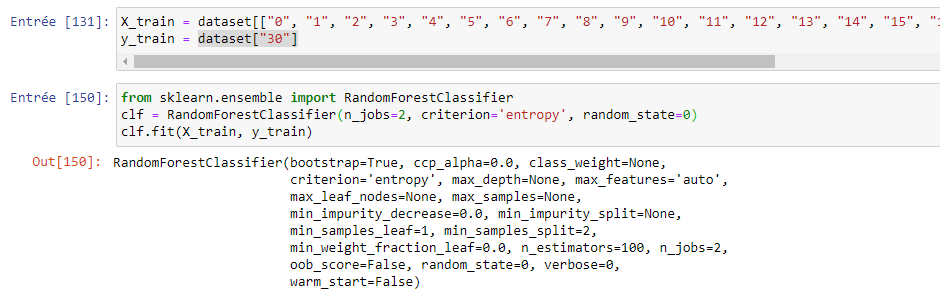
* Tập virus:



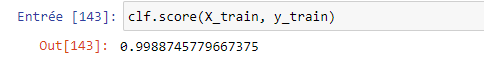
* Nối 2 dataset dùng làm tập train



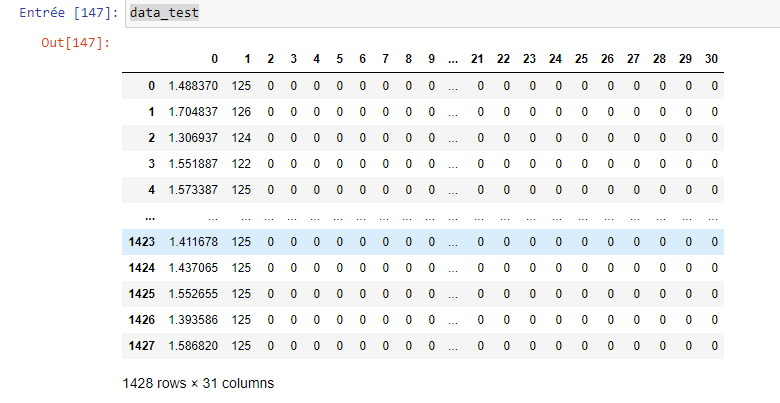
* Gán giá trị cho X\_train và y\_train làm đầu vào huấn luyện Random Forests. Huấn luyện mô hình:

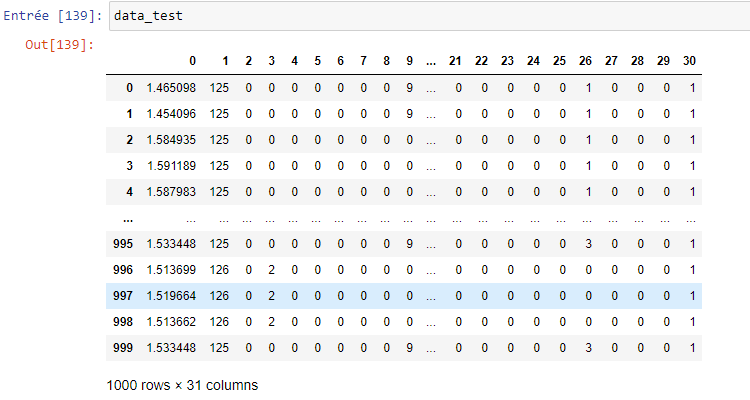


* Thử kiểm tra độ chính xác thuật toán so với tập đã train ban đầu:

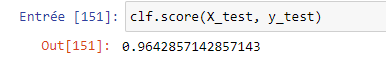


* Tập test gồm 2500 mẫu trong đó 1000 mẫu virus nguồn github (khác với tập train, và 1500 mẫu sạch. Cũng cho chạy qua tool EmulJS để lấy được dữ liệu dataset.

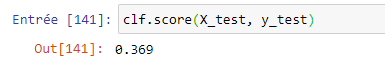




* Đánh giá độ chính xác với tập test 1 gồm 1500 mẫu sạch riêng biệt:



* Độ chính xác: 96,4%
  + Đánh giá độ chính xác với tập test gồm 1000 mẫu virus riêng biệt



* Độ chính xác chỉ khoảng 36.9%
* Qua quá trình tìm hiểu và thử nghiệm thì nhận thấy mô hình thuật toán Random Forests có độ chính xác phụ thuộc nhiều vào quá trình chọn đặc trưng do được phát triển từ thuật toán Cây quyết định. Muốn cải thiện độ chính xác của mô hình phải phân tích nhiều hành vi của Virus từ đó trích chọn được những đặc trưng phù hợp cho quá trình phân loại

# Kết luận

Sử dụng thuật toán học máy Random Forest có thể đem đến một số kết quả nhất định. Tuy nhiên, thực tế cho thấy thuật toán này có nhiều điểm hạn chế. Hạn chế đầu tiên là dựa vào quan sát của phân tích viên trong việc lựa chọn những dữ liệu để tạo các node trên cây quyết định. Trong trường hợp phát sinh một mẫu virus mới với những dữ liệu khác, ví dụ như: gọi hàm mới không nằm trong danh sách những hàm đã cho, cố gắng sử dụng đa dạng các từ ngữ để làm nhiễu chỉ số entropy, .. thì mô hình tính toán này trở nên vô dụng

# Tài liệu tham khảo

* <https://heimdalsecurity.com/blog/javascript-malware-explained>
* Data Mining with Decision Trees – Theory and Applications 2nd Edition – Lior Rokach – Oded Maimon