

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



BÁO CÁO CUỐI KỲ
NHẬP MÔN BẢO MẬT THÔNG TIN

DETECT PHISHING WEBSITES USING MACHINE LEARNING

Người hướng dẫn: **TS HUỖNH NGỌC TÚ**

Người thực hiện: **LÝ TUẤN AN – 52000620**

LÊ PHẠM ANH TRÍ – 52000153

Lớp : **20050201**

Khoá : **24**

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**



BÁO CÁO CUỐI KỲ

NHẬP MÔN BẢO MẬT THÔNG TIN

DETECT PHISHING WEBSITES USING MACHINE LEARNING

Người hướng dẫn: **TS HUỖNH NGỌC TỨ**

Người thực hiện: **LÝ TUẦN AN – 52000620**

LÊ PHẠM ANH TRÍ – 52000153

Lớp : 20050201

Khoá : 24

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn sâu sắc đến cô Huỳnh Ngọc Tú đã nhiệt tình giúp đỡ và hướng dẫn chúng em thực hiện bài báo cáo này. Chúng em cũng gửi lời cảm ơn đến Ban giám hiệu và các giảng viên khác của trường đã cung cấp cho chúng em môi trường học tập và nghiên cứu tốt nhất. Chúng em tin rằng những kiến thức từ bài báo cáo này sẽ giúp chúng em phát triển sự nghiệp của mình trong tương lai.

Chúng em xin chân thành cảm ơn.

TP. Hồ Chí Minh, ngày 19 tháng 05 năm 2024

Tác giả

(Ký tên và ghi rõ họ tên)



Lý Tuấn An

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

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi và được sự hướng dẫn khoa học của TS. Huỳnh Ngọc 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 báo cáo 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 tôi xin hoàn toàn chịu trách nhiệm về nội dung báo cáo 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 19 tháng 05 năm 2024

Tác giả

(Ký tên và ghi rõ họ tên)



Lý Tuấn An

TÓM TẮT

Đây là bài báo cáo cuối kỳ môn Nhập môn Bảo mật thông tin với đề tài phát hiện trang web lừa đảo sử dụng Machine Learning. Trong đề tài này chúng em đã sử dụng các kiến thức về bảo mật thông tin và các thuật toán trong Machine Learning để thực hiện báo cáo.

MỤC LỤC

DANH MỤC HÌNH VẼ	vi
DANH MỤC BẢNG BIỂU	vii
DANH MỤC CÁC CHỮ VIẾT TẮT.....	viii
CHƯƠNG 1. CƠ SỞ LÝ THUYẾT.....	1
1.1 Phishing	1
1.1.1 Khái niệm Phishing.....	1
1.1.2 Các phương thức Phishing phổ biến.....	2
1.1.3 Mối đe dọa của Phishing Websites	3
1.1.4 Các cuộc tấn công Phishing Websites	3
1.2 Sử dụng Machine Learning để phát hiện trang web lừa đảo	4
1.2.1 Khái niệm Machine Learning	4
1.2.2 Quy trình hoạt động của Machine Learning	5
1.2.3 Các phương pháp và thuật toán phổ biến của Machine Learning	6
1.2.4 Hoạt động của Machine Learning để phát hiện trang web lừa đảo.....	8
1.2.5 Ưu điểm và hạn chế của việc áp dụng Machine Learning trong phát hiện trang web lừa đảo	8
1.2.6 Ứng dụng của Machine Learning trong việc phát hiện trang web lừa đảo	9
CHƯƠNG 2. DEMO	11
2.1 Xử lý dữ liệu	11
2.1.1 Thu thập dữ liệu	11
2.1.2 Làm sạch dữ liệu.....	12
2.1.3 Mức độ tương quan của các thuộc tính	13

2.1.4 Trích xuất đặc trưng	14
2.2 Mô hình học máy	16
2.2.2 Lựa chọn mô hình	17
2.2.3 Huấn luyện mô hình	18
2.2.4 Đánh giá mô hình.....	22
2.3 Xây dựng Ứng dụng dự đoán URLs	24
2.3.1 Xây dựng chương trình trích xuất thuộc tính.....	24
2.3.2 Xây dựng Website.....	27
TÀI LIỆU THAM KHẢO	30

DANH MỤC HÌNH VẼ

Hình 1.1 Phishing.....	1
Hình 1.2 Phương thức Phishing phổ biến	2
Hình 1.3 Phishing Websites Twitter	4
Hình 1.4 Machine Learning	5
Hình 1.5 Quy trình hoạt động của Machine Learning	6
Hình 1.6 Các phương pháp và thuật toán phổ biến của Machine Learning.....	6

DANH MỤC BẢNG BIỂU

DANH MỤC CÁC CHỮ VIẾT TẮT

CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

1.1 Phishing

1.1.1 Khái niệm Phishing

Phishing (Tấn công giả mạo) là hình thức tấn công mạng mà kẻ tấn công giả mạo thành một đơn vị uy tín để lừa đảo người dùng cung cấp thông tin cá nhân cho chúng hoặc cài đặt các phần mềm độc hại. Phishing là một trong những hình thức tấn công phổ biến nhất trên mạng và đã gây ra nhiều thiệt hại cho các tổ chức và cá nhân.



Hình 1.1 Phishing

Phương thức Phishing được biết đến lần đầu tiên vào năm 1987. Nguồn gốc của từ Phishing là sự kết hợp của 2 từ: *fishing for information* (câu thông tin) và *phreaking* (trò lừa đảo sử dụng điện thoại của người khác không trả phí). Do sự giống nhau giữa việc “câu cá” và “câu thông tin người dùng”, nên thuật ngữ *Phishing* ra đời.

1.1.2 Các phương thức Phishing phổ biến

Phishing email: Một trong những kỹ thuật cơ bản trong tấn công Phishing là giả mạo email. Tin tặc sẽ gửi email cho người dùng dưới danh nghĩa một đơn vị hoặc tổ chức uy tín, dụ người dùng click vào đường link dẫn tới một website giả mạo và “mắc câu”.

Phishing Website: Thực chất, việc giả mạo website trong tấn công Phishing chỉ là làm giả một landing page chứ không phải toàn bộ website. Trang được làm giả thường là trang đăng nhập để cướp thông tin của nạn nhân.

Smishing: Cách tấn công này sẽ tiếp cận người dùng thông qua tin nhắn SMS, kẻ tấn công sẽ gửi tin nhắn kèm theo link lừa đảo với nội dung đa dạng nhằm hấp dẫn người dùng click vào link để chiếm đoạt thông tin.

Vishing: Đây là hình thức tấn công qua cuộc gọi điện thoại, trong đó kẻ tấn công giả danh một đại diện của một tổ chức đáng tin cậy và yêu cầu người dùng cung cấp thông tin nhạy cảm như số thẻ tín dụng hoặc mật khẩu.

Spear Phishing: Đây là một loại hình phải hơn, trong đó kẻ tấn công tập trung vào một số nhỏ các cá nhân hoặc tổ chức cụ thể. Kẻ tấn công tìm kiếm thông tin chi tiết về nạn nhân trên mạng xã hội hoặc các trang web công cộng khác để tạo ra email hoặc tin nhắn giả mạo có thể gây ảnh hưởng tới nạn nhân.



Hình 1.2 Phương thức Phishing phổ biến

1.1.3 Môi đe dọa của Phishing Websites

Với sự phát triển của internet và các giao dịch trực tuyến, các kẻ tấn công ngày càng sử dụng các kỹ thuật tinh vi hơn để làm giả trang web, khiến cho các cuộc tấn công phishing trở nên khó phát hiện và khó phòng chống hơn. Các kẻ tấn công cũng có xu hướng tìm kiếm các mục tiêu mới, bao gồm cả các tổ chức và doanh nghiệp, thay vì chỉ tấn công người dùng cá nhân.

Tác động của Phishing Websites:

- Tổn thất tài chính, lấy mất thông tin tài khoản ngân hàng hoặc thẻ tín dụng của người dùng.
- Đánh cắp danh tính người dùng và sử dụng thông tin này để thực hiện các hành vi lừa đảo khác.
- Tổn hại danh tiếng.
- Dữ liệu nhạy cảm của người dùng bị xâm phạm.

1.1.4 Các cuộc tấn công Phishing Websites

Tấn công phishing Facebook: Cuộc tấn công này đã sử dụng một trang web giả mạo Facebook để lừa đảo người dùng cung cấp thông tin đăng nhập của họ.

Tấn công phishing PayPal: Cuộc tấn công này đã sử dụng một trang web giả mạo PayPal để lừa đảo người dùng cung cấp thông tin đăng nhập của họ.

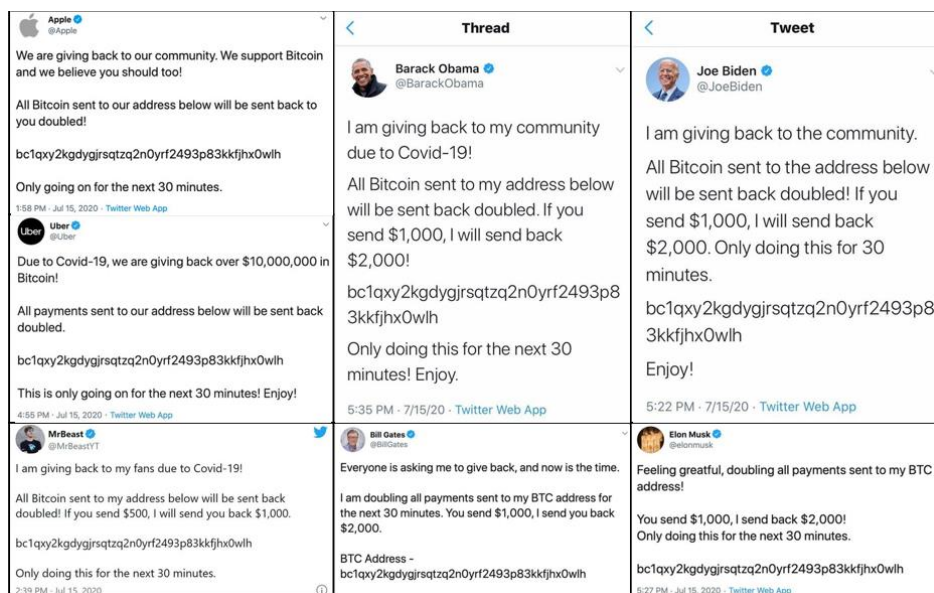
Năm 1990: America Online (AOL).

Những năm 2000: The Love Bug (2000), PayPal (2003).

Những năm 2010: RSA (2011), Google Docs (2017).

Năm 2020: Twitter: Một hacker 17 tuổi và đồng phạm đã thiết lập một trang web giả mạo giống như nhà cung cấp VPN nội bộ của Twitter được các nhân viên làm việc từ xa sử dụng. Giả làm nhân viên trợ giúp, họ gọi điện cho nhiều nhân viên Twitter, hướng dẫn họ gửi thông tin xác thực đến trang web VPN giả mạo. Sử dụng thông tin chi tiết do các nhân viên không nghi ngờ cung cấp, họ có thể giành quyền kiểm soát một số tài khoản người dùng cao cấp, bao gồm cả tài khoản của Barack Obama, Elon Musk, Joe Biden và tài khoản công ty của Apple Inc. Sau đó, các tin

tặc đã gửi tin nhắn tới những người theo dõi trên Twitter để chào mời Bitcoin, hứa hẹn sẽ tăng gấp đôi giá trị giao dịch. Các tin tặc đã thu thập được 12,86 BTC (khoảng 117.000 USD vào thời điểm đó).



Hình 1.3 Phishing Websites Twitter

1.2 Sử dụng Machine Learning để phát hiện trang web lừa đảo

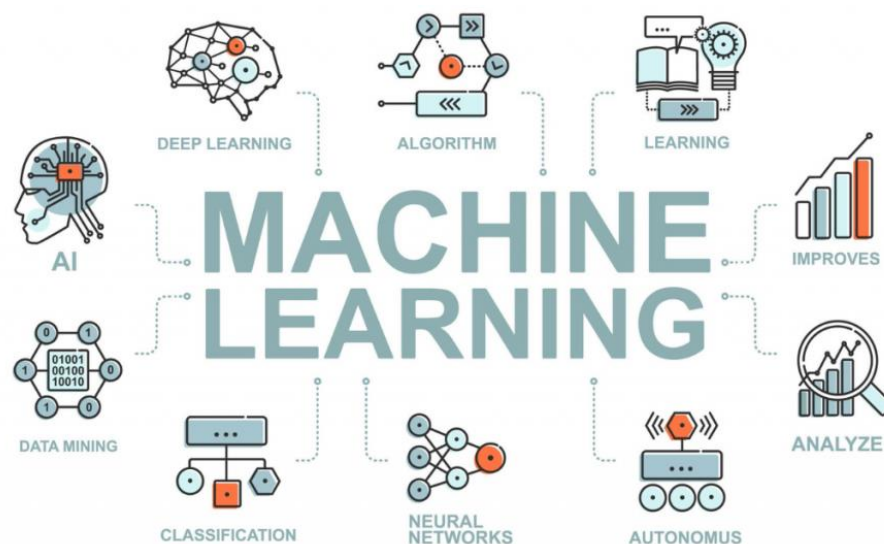
1.2.1 Khái niệm Machine Learning

Machine Learning (máy học) là một trường con của trí tuệ nhân tạo (AI) sử dụng các thuật toán được đào tạo trên các bộ dữ liệu để tạo ra các mô hình tự học có khả năng dự đoán kết quả và phân loại thông tin mà không cần sự can thiệp của con người.

Machine Learning tập trung vào việc xây dựng các thuật toán và mô hình thống kê cho phép hệ thống máy tính học từ dữ liệu mà không cần lập trình rõ ràng. Trong lĩnh vực an ninh mạng, Machine Learning đóng vai trò quan trọng trong việc nâng cao khả năng phát hiện và phân tích mối đe dọa.

Các thuật toán của Machine Learning có thể được huấn luyện trên các tập dữ liệu lớn để nhận ra các mô hình bình thường và bất thường trong lưu lượng mạng, hành vi của người dùng hoặc hoạt động của hệ thống. Khả năng học tập thích ứng này cho phép các hệ thống Machine Learning phát triển và tinh chỉnh độ chính xác

của chúng theo thời gian, cung cấp một phương thức hiệu quả để xác định các mối đe dọa mạng mới và tinh vi.



Hình 1.4 Machine Learning

1.2.2 Quy trình hoạt động của *Machine Learning*

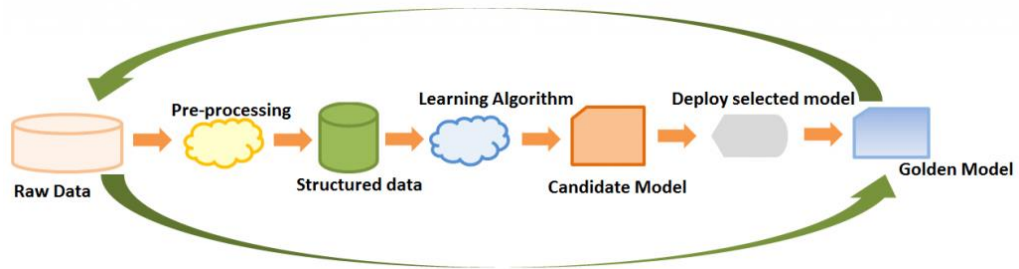
Data collection – thu thập dữ liệu: để máy tính học được cần có một bộ dữ liệu, có thể tự thu thập hoặc lấy các bộ dữ liệu đã công bố trước đó. Lưu ý là phải thu thập từ nguồn chính thức thì dữ liệu mới chính xác và máy học đúng cách, đạt hiệu quả cao hơn.

Preprocessing - tiền xử lý: bước này dùng để chuẩn hóa dữ liệu, loại bỏ các thuộc tính không cần thiết, gán nhãn dữ liệu, mã hóa một số đặc trưng, trích xuất, thu gọn dữ liệu nhưng vẫn đảm bảo kết quả...

Training model – huấn luyện mô hình: bước này là huấn luyện mô hình hoặc để nó học trên dữ liệu bạn đã thu thập và xử lý ở hai bước đầu tiên.

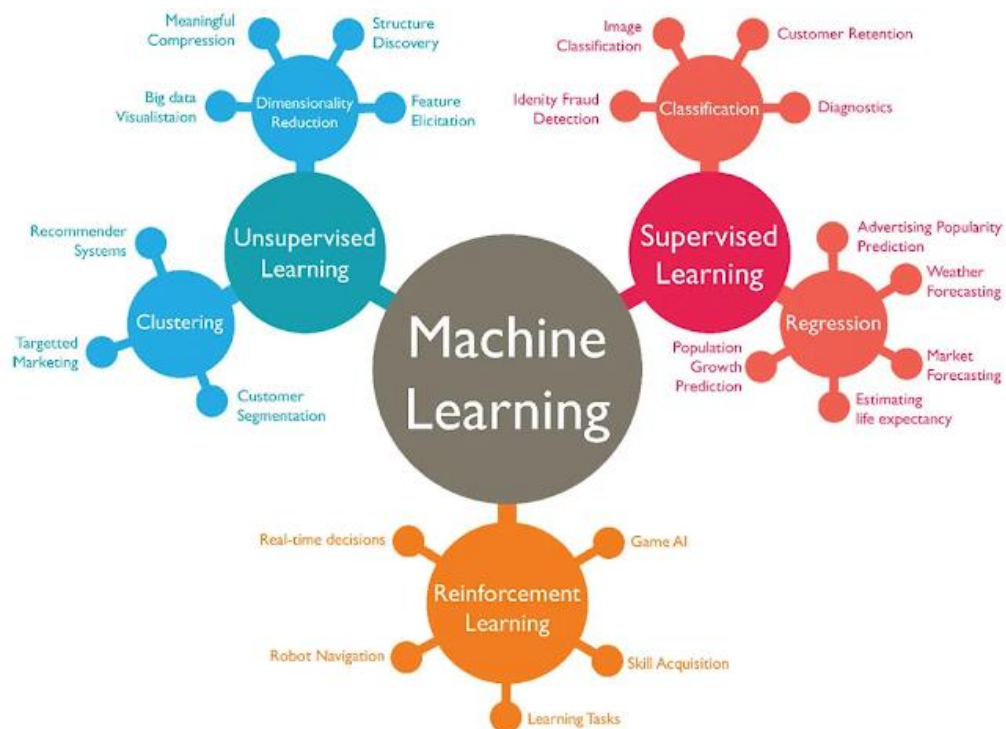
Evaluating model – đánh giá mô hình: sau khi huấn luyện mô hình, cần sử dụng các công cụ để đánh giá mô hình. Độ chính xác của mô hình trên 80% được coi là tốt.

Improve - Cải thiện: Sau khi đánh giá mô hình, những mô hình có độ chính xác kém cần đào tạo lại. Sẽ lặp lại từ bước 3, cho đến khi đạt độ chính xác mong muốn. Tổng thời gian của 3 bước cuối chiếm khoảng 30% tổng thời gian thực hiện.



Hình 1.5 Quy trình hoạt động của Machine Learning

1.2.3 Các phương pháp và thuật toán phổ biến của Machine Learning



Hình 1.6 Các phương pháp và thuật toán phổ biến của Machine Learning

Các phương pháp Machine Learning phổ biến:

Phương pháp Machine Learning được giám sát (Supervised machine learning): Trong phương pháp máy học có giám sát, các thuật toán được đào tạo trên các tập dữ liệu được gắn nhãn với các thẻ mô tả từng mục dữ liệu. Nói cách khác, các thuật toán nhận dữ liệu đi kèm với “answer key” cung cấp hướng dẫn về cách diễn giải dữ liệu. Các mô hình Machine Learning để phân loại và dự đoán thường được phát triển thông qua học máy có giám sát.

Phương pháp Machine Learning không được giám sát (Unsupervised machine learning): Trong phương pháp này, các thuật toán được đào tạo bằng cách sử dụng các tập dữ liệu chưa được gán nhãn. Chương trình lập trình phải tự tìm các mẫu mà không cần sự trợ giúp từ bên ngoài vì nó được cung cấp dữ liệu mà không có thể trong quy trình này. Các nhà nghiên cứu và nhà khoa học dữ liệu thường xuyên sử dụng công nghệ học máy không giám sát để tìm ra các mẫu trong các tập dữ liệu khổng lồ, không được gán nhãn một cách nhanh chóng và hiệu quả.

Phương pháp Machine Learning bán giám sát (Semi-supervised machine learning): Với phương pháp Machine Learning bán giám sát, các thuật toán được đào tạo bằng cách sử dụng cả tập dữ liệu được gán nhãn và không được gán nhãn. Trong học máy bán giám sát, các thuật toán thường được cung cấp lượng dữ liệu chưa được gán nhãn lớn hơn đáng kể để hoàn thiện mô hình sau lần đầu tiên nhận được một lượng nhỏ dữ liệu được gán nhãn để hướng dẫn sự phát triển của hệ thống. Vì không có sẵn một lượng đáng kể dữ liệu được gán nhãn, học máy bán giám sát thường được sử dụng để huấn luyện các thuật toán cho các ứng dụng phân loại và dự đoán.

Các thuật toán phổ biến của Machine Learning:

Neural networks: Mô phỏng cách thức hoạt động của bộ não con người, với một số lượng khổng lồ các nút xử lý được liên kết. Neural networks là thuật toán được dùng trong việc nhận dạng các mẫu và đóng một vai trò quan trọng trong các ứng dụng bao gồm dịch ngôn ngữ tự nhiên, nhận dạng hình ảnh, nhận dạng giọng nói và tạo hình ảnh.

Linear regression: Thuật toán này được sử dụng để dự đoán các giá trị số, dựa trên mối quan hệ tuyến tính giữa các giá trị khác nhau.

Logistic regression: Thuật toán giúp đưa ra dự đoán cho các biến phản hồi phân loại, chẳng hạn như câu trả lời “có/không” cho các câu hỏi. Nó có thể được sử dụng cho các ứng dụng như phân loại thư rác và kiểm soát chất lượng trên dây chuyền sản xuất.

Clustering: Các thuật toán phân cụm có thể xác định các mẫu trong dữ liệu để nó có thể được nhóm lại. Máy tính có thể giúp các nhà khoa học dữ liệu bằng cách xác định sự khác biệt giữa các mục dữ liệu mà con người đã bỏ qua.

Decision trees: Là thuật toán được sử dụng để dự đoán giá trị số (hồi quy) và phân loại dữ liệu. Decision trees sử dụng một chuỗi phân nhánh của các quyết định được liên kết có thể được biểu diễn bằng sơ đồ cây. Một trong những ưu điểm của decision trees là chúng dễ xác thực và kiểm tra, không giống thuật toán Neural networks.

Random forests: Trong một khu rừng ngẫu nhiên, thuật toán máy học dự đoán một giá trị hoặc danh mục bằng cách kết hợp các kết quả từ một số cây quyết định.

1.2.4 Hoạt động của Machine Learning để phát hiện trang web lừa đảo

Sử dụng các tính năng phổ biến để đào tạo mô hình.

Thuật toán đào tạo trên bộ dữ liệu lớn về các tính năng được trích xuất từ các trang web lừa đảo đã biết và các trang web hợp pháp.

Khi được giới thiệu một trang web mới, mô hình được đào tạo sẽ đánh giá các tính năng của trang web đó và dự đoán liệu đó có phải là trang web lừa đảo hay không.

1.2.5 Ưu điểm và hạn chế của việc áp dụng Machine Learning trong phát hiện trang web lừa đảo

Ưu điểm:

- Có thể phân tích lượng dữ liệu khổng lồ để xác định các mẫu và điểm bất thường liên quan đến các trang web lừa đảo.
- Phát hiện chính xác hơn các trang web lừa đảo so với các phương pháp dựa trên quy tắc truyền thống.
- Thời gian phát hiện và phản hồi nhanh hơn.

Hạn chế:

- Phải đảm bảo rằng các thuật toán có thể phát hiện các loại tấn công lừa đảo mới và đang phát triển.

- Yêu cầu cập nhật liên tục dữ liệu đào tạo và các tính năng được thuật toán sử dụng.
- Dễ bị tấn công bất lợi, kẻ tấn công có thể thao túng các tính năng của trang web lừa đảo để tránh bị phát hiện bởi các thuật toán Machine Learning. Họ có thể tinh chỉnh hoặc thay đổi các đặc trưng và hành vi để tránh bị phát hiện.

1.2.6 Ứng dụng của Machine Learning trong việc phát hiện trang web lừa đảo

Bảo mật email

- Để tự động phân tích các email đến để tìm dấu hiệu lừa đảo.
- Phân loại email dựa trên nội dung, danh tiếng của người gửi và các tính năng khác để xác định các thư đáng ngờ.
- Quét thời gian thực các tệp đính kèm email và liên kết nhúng để phát hiện tải trọng độc hại hoặc URL lừa đảo.

Trình duyệt web

- Tiện ích mở rộng hoặc plugin của trình duyệt.
- Để cung cấp các cảnh báo và cảnh báo theo thời gian thực khi người dùng cố gắng truy cập các trang web độc hại tiềm ẩn.
- Phân tích nội dung và hành vi của trang web để xác định các dấu hiệu lừa đảo như thiết kế lừa đảo, chuyển hướng đáng ngờ và các hình thức lừa đảo.

Giám sát phương tiện truyền thông xã hội

- Để giám sát các nền tảng truyền thông xã hội nhằm phát hiện các chiến dịch lừa đảo, tài khoản giả mạo và các chương trình khuyến mãi gian lận.
- Tự động báo cáo và gỡ bỏ nội dung liên quan đến lừa đảo dựa trên đánh giá rủi ro dựa trên Machine Learning.

An ninh mạng

- Phát hiện các hành vi bất thường như đăng ký miền bất thường, truy vấn DNS đáng ngờ và truyền dữ liệu không mong muốn liên quan đến các cuộc tấn công lừa đảo.
- Tích hợp với tường lửa và bộ lọc mạng để chặn quyền truy cập vào các miền và địa chỉ IP lừa đảo đã biết.

[illegible]

2. Data Preprocessing & EDA

[37] `raw_dataset.head()`

	url	length_url	length_hostname	ip	nb_dots	nb_hyphens	nb_at	nb_qm	nb_and	nb_or	...	domain_in_title	dom
0	http://www.crestonwood.com/router.php	37	19	0	3	0	0	0	0	0	...	0	
1	http://shadetretechnology.com/V4/validation/a...	77	23	1	1	0	0	0	0	0	...	1	
2	https://support-appleld.com.secureupdate.duila...	126	50	1	4	1	0	1	2	0	...	1	
3	http://rgipt.ac.in	18	11	0	2	0	0	0	0	0	...	1	
4	http://www.iracing.com/tracks/gateway-motorspo...	55	15	0	2	2	0	0	0	0	...	0	

5 rows x 89 columns

[38] `raw_dataset.shape`

(11430, 89)

2.1.2 Làm sạch dữ liệu

Dữ liệu thu thập được thường có chứa các giá trị bị thiếu hoặc không chính xác. Các bước làm sạch dữ liệu bao gồm:

- Loại bỏ các giá trị trùng lặp: Đảm bảo mỗi trang web chỉ xuất hiện một lần trong tập dữ liệu.
- Xử lý các giá trị bị thiếu: Có thể sử dụng các phương pháp như loại bỏ các hàng có giá trị bị thiếu hoặc sử dụng các kỹ thuật ước lượng để điền vào các giá trị này.

```

pd.set_option('display.max_rows', 500)
raw_dataset.isna().sum()
# no missing values

```

url	0
length_url	0
length_hostname	0
ip	0
nb_dots	0
nb_hyphens	0
nb_at	0
nb_qm	0
nb_and	0
nb_or	0
nb_eq	0
nb_underscore	0
nb_tilde	0
nb_percent	0
nb_slash	0
nb_star	0
nb_colon	0
nb_comma	0
nb_semicolon	0
nb_dollar	0
nb_space	0
nb_www	0
nb_com	0
nb_dslash	0
http_in_path	0
...	0
dns_record	0
google_index	0

Hiện tại bộ dữ liệu không có giá trị bị trùng lặp

2.1.3 Mức độ tương quan của các thuộc tính

Trong quá trình xây dựng mô hình phát hiện trang web lừa đảo, việc đánh giá mức độ tương quan giữa các thuộc tính là rất quan trọng để hiểu sâu hơn về dữ liệu và lựa chọn các đặc trưng phù hợp cho mô hình học máy. Mức độ tương quan giữa các thuộc tính có thể được đo bằng các phương pháp như:

2.1.3.1 Ma trận tương quan

Ma trận tương quan là một cách để đo mức độ tương quan giữa các cặp thuộc tính trong tập dữ liệu. Giá trị của ma trận tương quan nằm trong khoảng từ -1 đến 1, trong đó:

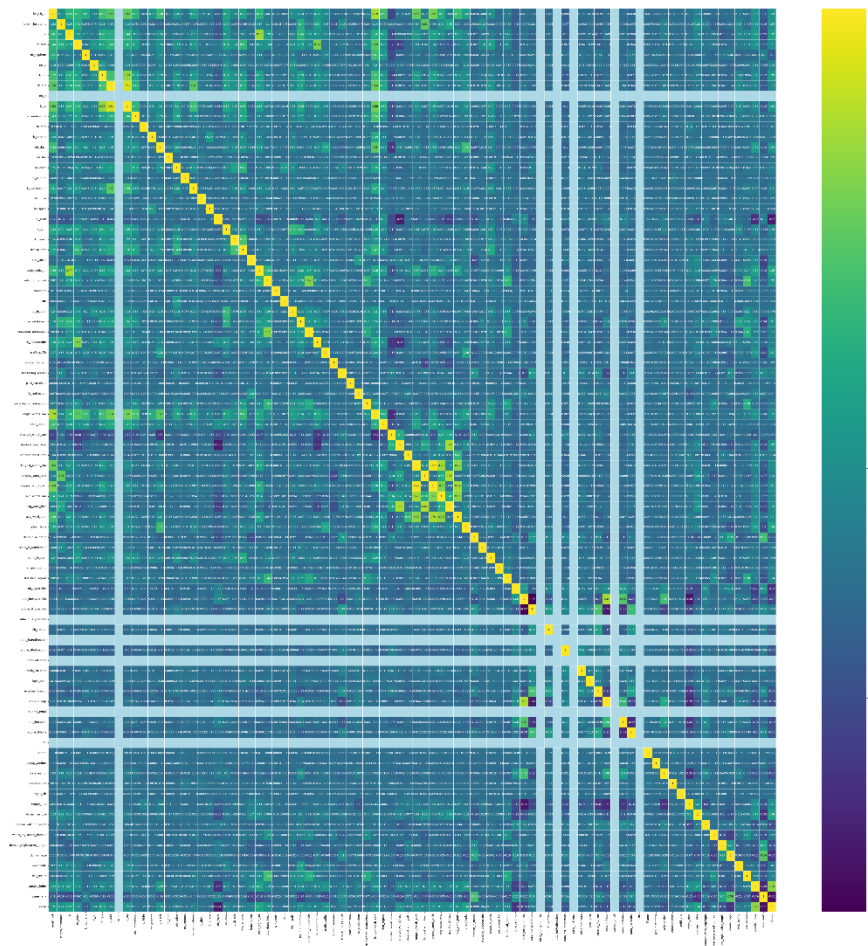
- Giá trị gần 1: Tương quan tích cực hoàn toàn, tức là khi một thuộc tính tăng, thuộc tính kia cũng tăng theo.
- Giá trị gần -1: Tương quan âm hoàn toàn, tức là khi một thuộc tính tăng, thuộc tính kia giảm.
- Giá trị gần 0: Không có tương quan tuyến tính giữa các thuộc tính.

2.1.3.2 Biểu đồ phân phối và scatter plot

Biểu đồ phân phối cho thấy phân phối của từng thuộc tính riêng lẻ, trong khi scatter plot thể hiện mối quan hệ giữa các cặp thuộc tính. Từ các biểu đồ này, có thể quan sát mức độ tương quan giữa các thuộc tính.

2.1.3.3 Sử dụng biểu đồ nhiệt (Heatmap)

Trong phần tìm kiếm độ tương quan của các thuộc tính, chúng ta sẽ sử dụng heatmap để trực quan hóa



2.1.4 Trích xuất đặc trưng

Thông qua việc đánh giá mức độ tương quan giữa các thuộc tính, chúng ta có thể loại bỏ hoặc kết hợp các thuộc tính có mức độ tương quan cao để giảm chiều dữ liệu và cải thiện hiệu suất của mô hình học máy.

```

return result

features_slt = feature_selector_correlation(status_corr, 0.2)
features_slt

...
[('length_url', ['0.248580']),
 ('length_hostname', ['0.238322']),
 ('ip', ['0.321698']),
 ('nb_dots', ['0.207029']),
 ('nb_qm', ['0.294319']),
 ('nb_eq', ['0.233386']),
 ('nb_slash', ['0.242270']),
 ('nb_www', ['0.443466']),
 ('ratio_digits_url', ['0.356395']),
 ('ratio_digits_host', ['0.224335']),
 ('tld_in_subdomain', ['0.208884']),
 ('prefix_suffix', ['0.214681']),
 ('shortest_word_host', ['0.223084']),
 ('longest_words_raw', ['0.200147']),
 ('longest_word_path', ['0.212709']),
 ('phish_hints', ['0.335393']),
 ('nb_hyperlinks', ['0.342628']),
 ('ratio_inhyperlinks', ['0.243982']),
 ('empty_title', ['0.207043']),
 ('domain_in_title', ['0.342807']),
 ('domain_age', ['0.331889']),
 ('google_index', ['0.731171']),
 ('page_rank', ['0.511137']),
 ('status', ['1.000000'])]

```

Trong phần này, chúng ta chỉ quan tâm mức độ tương quan với thuộc tính status, là một biến phụ thuộc. Giữ lại những thuộc tính có điểm số từ khoảng 0.2 đến 1 hoặc -0.2 đến -1.

Do đó tập dữ liệu chỉ còn lại 23 biến độc lập sau khi lược bỏ đi các thuộc tính ít có độ tương quan đến biến status.

3. Splitting the Data

3.1 setup predictors and targets

```

cleaned_dataset = pd.read_csv('datasets/cleaned_dataset.csv')
X_selected = cleaned_dataset.iloc[:, 1:-1]
X_selected

```

	length_url	length_hostname	ip	nb_dots	nb_qm	nb_eq	nb_slash	nb_www	ratio_digits_url	ratio_digits_host	...	longest_words_raw	longest_word_path	phish_hints	nb_hyperlinks	ratio_ir
0	37	19	0	3	0	0	3	1	0.000000	0.0	...	11	6	0	17	
1	77	23	1	1	0	0	5	0	0.220779	0.0	...	32	32	0	30	
2	126	50	1	4	1	3	5	0	0.150794	0.0	...	17	17	0	4	
3	18	11	0	2	0	0	2	0	0.000000	0.0	...	5	0	0	149	
4	55	15	0	2	0	0	5	1	0.000000	0.0	...	11	11	0	102	
...	
11436	84	22	0	3	0	0	7	0	0.000000	0.0	...	9	9	0	126	
11437	39	13	0	2	0	0	4	1	0.000000	0.0	...	9	9	2	88	
11438	56	16	0	1	0	0	4	0	0.000000	0.0	...	12	8	1	95	
11439	90	21	0	6	1	1	4	0	0.044444	0.0	...	11	11	1	39	
11440	37	21	0	3	0	0	4	0	0.000000	0.0	...	9	6	0	12	

11441 rows x 23 columns

2.2 Mô hình học máy

Sau khi dữ liệu đã được xử lý và chuẩn bị, chúng ta sẽ sử dụng nó để huấn luyện các mô hình học máy nhằm phát hiện các trang web lừa đảo.

2.2.1.1 Chia tập dữ liệu



```

3.2 train test split
[97] from sklearn.model_selection import train_test_split
ImportError: Import "sklearn.model_selection" could not be resolved from source

Python

# Splitting the dataset into train and test sets: 80-20 split
X_train, X_test, y_train, y_test = train_test_split(X_selected, y,
                                                    test_size=0.2,
                                                    random_state=42,
                                                    shuffle = True)
X_train.shape, X_test.shape
[98]
... ((9152, 23), (2289, 23))
Python

```

Ta thực hiện việc chia tập dữ liệu thành hai phần: một phần dành cho việc huấn luyện mô hình (train set) và một phần dành cho việc kiểm tra hiệu suất của mô hình (test set). Cụ thể:

- `X_selected`: là ma trận chứa các đặc trưng đã được chọn hoặc được tạo ra từ dữ liệu.
- `y`: là vector chứa nhãn tương ứng với mỗi mẫu trong tập dữ liệu.
- `test_size=0.2`: đây là tỷ lệ phần trăm của dữ liệu sẽ được chia cho tập kiểm tra, tức là 20% của dữ liệu sẽ được sử dụng để kiểm tra mô hình.
- `random_state=42`: đây là một số nguyên được sử dụng để tạo ra một trạng thái ngẫu nhiên cố định, đảm bảo rằng kết quả của quá trình chia dữ liệu này sẽ không thay đổi nếu bạn chạy code này nhiều lần.
- `shuffle=True`: đảm bảo rằng dữ liệu sẽ được xáo trộn trước khi chia thành các tập train và test, giúp loại bỏ bất kỳ sự phụ thuộc nào giữa các mẫu liên tiếp.

Sau khi chạy dòng code này, biến `X_train` sẽ chứa các đặc trưng của tập huấn luyện, `y_train` sẽ chứa nhãn tương ứng của tập huấn luyện, `X_test` sẽ chứa các đặc trưng của tập kiểm tra, và `y_test` sẽ chứa nhãn tương ứng của tập kiểm tra.

Kết quả của phép chia dữ liệu này là thông tin về hình dạng (shape) của `X_train` và `X_test`, được trả về bởi lời gọi `X_train.shape` và `X_test.shape`. Điều này giúp đảm

bảo rằng việc chia dữ liệu đã được thực hiện đúng cách và mỗi tập dữ liệu có kích thước phù hợp.

2.2.2 Lựa chọn mô hình

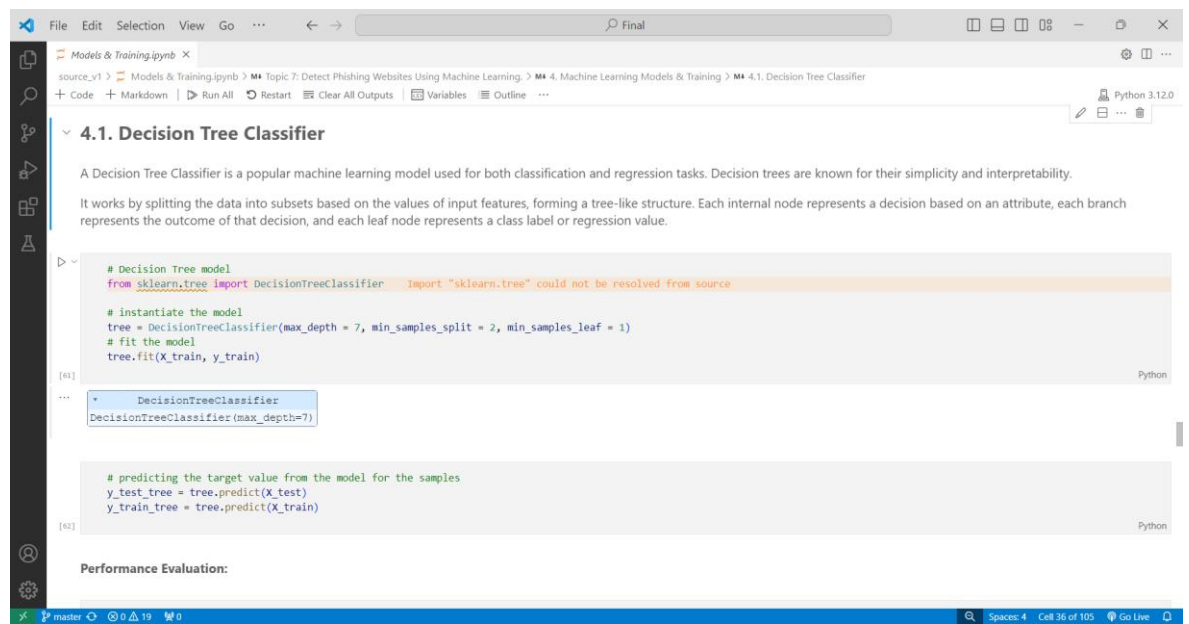
Dựa trên tập dữ liệu được cung cấp, chúng ta đang đối mặt với một bài toán học máy có giám sát. Các bài toán học máy có giám sát thường được chia thành hai loại chính: phân loại và hồi quy. Tập dữ liệu cụ thể này là cho một bài toán phân loại, nơi mục tiêu là phân loại các URL thành hai nhóm: lừa đảo (1) hoặc hợp pháp (0). Các mô hình phân loại được sử dụng để huấn luyện trên tập dữ liệu trong sổ tay này bao gồm:

- Decision Tree (Cây Quyết Định): Xây dựng một cây quyết định dựa trên các thuộc tính của dữ liệu để phân loại các mẫu.
- Random Forest (Rừng Ngẫu Nhiên): Kết hợp nhiều cây quyết định để tạo ra một mô hình phân loại mạnh mẽ hơn.
- Multilayer Perceptrons (Đa Tầng Perceptron): Một dạng của mạng nơ-ron sâu (deep neural network) được sử dụng trong các bài toán phân loại.
- XGBoost: Một thuật toán tăng cường gradient (gradient boosting) được sử dụng cho các bài toán phân loại và hồi quy.
- Support Vector Machines (Máy Vector Hỗ Trợ): Tìm ra ranh giới phân loại tốt nhất giữa các lớp bằng cách tìm ra siêu phẳng tốt nhất.
- Naive Bayes: Dựa trên giả định đơn giản rằng các đặc trưng độc lập với nhau để xác định xác suất phân loại của một mẫu.
- Logistic Regression (Hồi Quy Logistic): Một mô hình tuyến tính được sử dụng cho các bài toán phân loại nhị phân.
- k-Nearest Neighbors (k-Gần Nhất): Phân loại một mẫu bằng cách xem xét nhãn của các mẫu láng giềng gần nhất.
- Gradient Boosting Machines: Một phương pháp tăng cường gradient khác được sử dụng để xây dựng mô hình phân loại.

- Các mô hình này sẽ được huấn luyện trên tập dữ liệu để xây dựng các mô hình phân loại, và sau đó sẽ được đánh giá dựa trên hiệu suất của chúng trên tập kiểm tra.

2.2.3 Huấn luyện mô hình

2.2.3.1 Decision Tree



4.1. Decision Tree Classifier

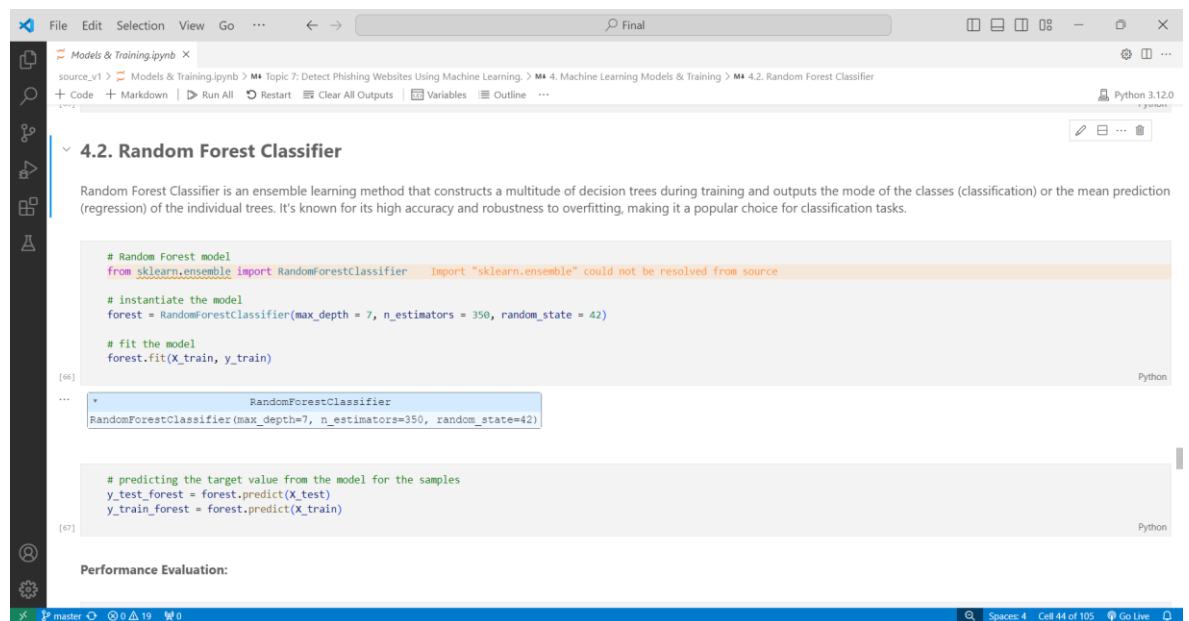
A Decision Tree Classifier is a popular machine learning model used for both classification and regression tasks. Decision trees are known for their simplicity and interpretability. It works by splitting the data into subsets based on the values of input features, forming a tree-like structure. Each internal node represents a decision based on an attribute, each branch represents the outcome of that decision, and each leaf node represents a class label or regression value.

```
[61] # Decision Tree model
from sklearn.tree import DecisionTreeClassifier
# instantiate the model
tree = DecisionTreeClassifier(max_depth = 7, min_samples_split = 2, min_samples_leaf = 1)
# fit the model
tree.fit(X_train, y_train)
```

```
[62] # predicting the target value from the model for the samples
y_test_tree = tree.predict(X_test)
y_train_tree = tree.predict(X_train)
```

Performance Evaluation:

2.2.3.2 Random Forest



4.2. Random Forest Classifier

Random Forest Classifier is an ensemble learning method that constructs a multitude of decision trees during training and outputs the mode of the classes (classification) or the mean prediction (regression) of the individual trees. It's known for its high accuracy and robustness to overfitting, making it a popular choice for classification tasks.

```
[66] # Random Forest model
from sklearn.ensemble import RandomForestClassifier
# instantiate the model
forest = RandomForestClassifier(max_depth = 7, n_estimators = 350, random_state = 42)
# fit the model
forest.fit(X_train, y_train)
```

```
[67] # predicting the target value from the model for the samples
y_test_forest = forest.predict(X_test)
y_train_forest = forest.predict(X_train)
```

Performance Evaluation:

2.2.3.3 Multilayer Perceptrons (MLPs): Deep Learning

4.3. Multilayer Perceptrons (MLPs): Deep Learning

Multilayer Perceptrons (MLPs) are a fundamental type of artificial neural network commonly used in deep learning. They consist of multiple layers of interconnected neurons, including an input layer, one or more hidden layers, and an output layer. MLPs are known for their capability to learn complex patterns in data and have been successfully applied to various tasks, including classification, regression, and pattern recognition.

```
# Multilayer Perceptrons model
from sklearn.neural_network import MLPClassifier  # Import "sklearn.neural_network" could not be resolved from source

# instantiate the model
mlp = MLPClassifier(alpha = 0.001, hidden_layer_sizes = ([100,100,100]))

# fit the model
mlp.fit(X_train, y_train)
```

```
[71]
...
MLPClassifier(alpha=0.001, hidden_layer_sizes=(100, 100, 100))
```

```
# predicting the target value from the model for the samples
y_test_mlp = mlp.predict(X_test)
y_train_mlp = mlp.predict(X_train)
```

[72]

Performance Evaluation:

2.2.3.4 XGBoost Classifier

4.4. XGBoost Classifier

XGBoost (Extreme Gradient Boosting) Classifier is an optimized implementation of gradient boosting designed for speed and performance. It is based on the gradient boosting framework and is widely used for supervised learning tasks, particularly in classification and regression problems. XGBoost has gained popularity for its efficiency, scalability, and effectiveness in producing highly accurate models.

```
# XGBoost Classification model
from xgboost import XGBClassifier  # Import "xgboost" could not be resolved

# instantiate the model
xgb = XGBClassifier(max_depth = 7, n_estimators = 350, learning_rate = 0.4)
# fit the model
xgb.fit(X_train, y_train)
```

```
[79]
...
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.4, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=7, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=350, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)
```

```
# predicting the target value from the model for the samples
y_test_xgb = xgb.predict(X_test)
y_train_xgb = xgb.predict(X_train)
```

2.2.3.5 Support Vector Machines

4.5. Support Vector Machines

Support Vector Machines (SVM) is a powerful supervised learning algorithm used for classification and regression tasks. It works by finding the hyperplane that best separates the data into different classes while maximizing the margin between the classes. SVMs are effective in high-dimensional spaces and are particularly useful when the number of features exceeds the number of samples.

```
# Support vector machine model
from sklearn.svm import SVC
import "sklearn.svm" could not be resolved from source

# instantiate the model
svm = SVC(C=1.0, random_state=12)
# fit the model
svm.fit(X_train, y_train)
```

[79] SVC

```
SVC(random_state=12)
```

```
# predicting the target value from the model for the samples
y_test_svm = svm.predict(X_test)
y_train_svm = svm.predict(X_train)
```

[80]

Performance Evaluation:

```
#computing the accuracy of the model performance
```

2.2.3.6 Naive Bayes

4.6. Naive Bayes Classifier

Naive Bayes is a simple yet effective probabilistic classifier based on Bayes' theorem with an assumption of independence between features. Despite its naive assumption, Naive Bayes often performs well in practice and is widely used in various classification tasks.

```
# Support vector machine model
from sklearn.naive_bayes import GaussianNB
import "sklearn.naive_bayes" could not be resolved from source

# instantiate the model
bayes = GaussianNB()
# fit the model
bayes.fit(X_train, y_train)
```

[83] GaussianNB

```
GaussianNB()
```

```
# predicting the target value from the model for the samples
y_test_bayes = bayes.predict(X_test)
y_train_bayes = bayes.predict(X_train)
```

[84]

Performance Evaluation:

```
#computing the accuracy of the model performance
acc_train_bayes = accuracy_score(y_train, y_train_bayes)
```

2.2.3.7 Logistic Regression

The screenshot shows a Jupyter Notebook titled "4.7. Logistic Regression". The code defines a Logistic Regression model, instantiates it, fits it to training data, and predicts on test data. A warning message is displayed: "ConvergenceWarning: lbfgs failed to converge (status=1): STOP: TOTAL NO. of ITERATIONS REACHED LIMIT." The warning suggests increasing the number of iterations or scaling the data, with links to scikit-learn documentation.

```

# Logistic Regression model
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
# fit the model
logreg.fit(X_train, y_train)

# predicting the target value from the model for the samples
y_test_logreg = logreg.predict(X_test)
y_train_logreg = logreg.predict(X_train)

```

Warning: ConvergenceWarning: lbfgs failed to converge (status=1): STOP: TOTAL NO. of ITERATIONS REACHED LIMIT. Increase the number of iterations (max_iter) or scale the data as shown in: <https://scikit-learn.org/stable/modules/preprocessing.html>. Please also refer to the documentation for alternative solver options: https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

2.2.3.8 k-Nearest Neighbors (kNN)

The screenshot shows a Jupyter Notebook titled "4.8. k-Nearest Neighbors (kNN)". The code instantiates a KNeighborsClassifier, fits it to training data, and predicts on test data. The notebook also includes a section for "Performance Evaluation".

```

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
# fit the model
knn.fit(X_train, y_train)

# predicting the target value from the model for the samples
y_test_knn = knn.predict(X_test)
y_train_knn = knn.predict(X_train)

```

Performance Evaluation:

2.2.3.9 Gradient Boosting Machines (GBM)

4.9. Gradient Boosting Machines (GBM)

Gradient Boosting Machines (GBM) is a powerful ensemble learning technique used for both regression and classification tasks in machine learning. It builds a strong predictive model by combining the predictions of multiple weak learners (typically decision trees) sequentially.

```

from sklearn.ensemble import GradientBoostingClassifier
import "sklearn.ensemble" could not be resolved from source

# instantiate the model
gbm = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1, max_depth=3, random_state=42)
# fit the model
gbm.fit(X_train, y_train)

[99]

...
GradientBoostingClassifier
GradientBoostingClassifier(random_state=42)

# predicting the target value from the model for the samples
y_test_gbm = gbm.predict(X_test)
y_train_gbm = gbm.predict(X_train)

[96]

Performance Evaluation:

# computing the accuracy of the model performance
acc_train_gbm = accuracy_score(y_train, y_train_gbm)
acc_test_gbm = accuracy_score(y_test, y_test_gbm)

```

2.2.4 Đánh giá mô hình

Train Accuracy và Test Accuracy đều là các chỉ số quan trọng trong quá trình đánh giá hiệu suất của một mô hình học máy. Tuy nhiên, mỗi chỉ số có vai trò và ý nghĩa riêng trong quá trình phát triển và đánh giá mô hình:

- Train Accuracy:

- Đo lường độ chính xác của mô hình trên dữ liệu huấn luyện.
- Train Accuracy cao có thể cho thấy mô hình đã học được dữ liệu huấn luyện tốt và có khả năng khớp dữ liệu mẫu.
- Tuy nhiên, một train accuracy quá cao có thể là dấu hiệu của overfitting, tức là mô hình quá tinh chỉnh để khớp với dữ liệu huấn luyện nhưng không thể tổng quát hóa tốt trên dữ liệu mới.

- Test Accuracy:

- Đo lường độ chính xác của mô hình trên dữ liệu kiểm tra, tức là dữ liệu mà mô hình chưa từng thấy trong quá trình huấn luyện.
- Test Accuracy là một chỉ số quan trọng hơn vì nó đo lường khả năng tổng quát hóa của mô hình trên dữ liệu mới.

- Nếu Test Accuracy cao và gần bằng Train Accuracy, điều này có nghĩa là mô hình đã học được các mẫu một cách hiệu quả và có khả năng tổng quát hóa tốt trên dữ liệu mới.

Trong tổng quan, Test Accuracy được coi là một chỉ số quan trọng hơn, bởi vì mục tiêu của chúng ta khi xây dựng mô hình là để có khả năng dự đoán tốt trên dữ liệu mới mà mô hình chưa từng thấy trước đó. Do đó, chúng ta thường quan tâm nhiều hơn đến Test Accuracy và cần đảm bảo rằng mô hình của chúng ta không chỉ khớp tốt với dữ liệu huấn luyện mà còn tổng quát hóa tốt trên dữ liệu mới.

Do đó để đánh giá hiệu suất của các mô hình học máy được huấn luyện trên tập dữ liệu, chúng ta sử dụng hai phép đo chính là Train Accuracy và Test Accuracy. Dưới đây là bảng tổng hợp kết quả của các mô hình:

	ML Model	Train Accuracy	Test Accuracy
0	Decision Tree	0.9506	0.9406
1	Random Forest	0.9552	0.9554
2	Multilayer Perceptrons	0.8905	0.8973
3	XGBoost	1.0000	0.9699
4	SVM	0.7206	0.7208
5	Bayesian Naive	0.8032	0.8038
6	Logistic Regression	0.8633	0.8707
7	k-Nearest Neighbors (kNN)	0.9016	0.8694
8	Gradient Boosting Machines (GBM)	0.9590	0.9576

Sau đó ta thực hiện sắp xếp giảm dần. Dưới đây là bảng tổng hợp kết quả của các mô hình theo sắp xếp giảm dần:

	ML Model	Train Accuracy	Test Accuracy
3	XGBoost	1.0000	0.9699
8	Gradient Boosting Machines (GBM)	0.9590	0.9576
1	Random Forest	0.9552	0.9554
0	Decision Tree	0.9506	0.9406
2	Multilayer Perceptrons	0.8905	0.8973
6	Logistic Regression	0.8633	0.8707
7	k-Nearest Neighbors (kNN)	0.9016	0.8694
5	Bayesian Naive	0.8032	0.8038
4	SVM	0.7206	0.7208

2.2.4.1 Nhận xét

Các mô hình Decision Tree và Random Forest đều cho thấy hiệu suất tốt trên cả tập huấn luyện và tập kiểm tra, với độ chính xác gần nhau.

Mô hình XGBoost đạt được độ chính xác 100% trên tập huấn luyện, nhưng vẫn giữ được hiệu suất tốt trên tập kiểm tra, chỉ mất một ít điểm so với tập huấn luyện.

Mô hình SVM và k-Nearest Neighbors cho thấy hiệu suất kém hơn so với các mô hình khác, có thể do các mô hình này không phản ánh tốt sự phức tạp của dữ liệu.

Logistic Regression và Naive Bayes cũng cho thấy hiệu suất tốt nhưng không bằng các mô hình cây và tăng cường gradient.

Gradient Boosting Machines cũng có hiệu suất rất tốt, tương đương với các mô hình Random Forest và Decision Tree.

Dựa trên kết quả trên, chúng ta có thể chọn một hoặc một số mô hình tốt nhất để triển khai trong ứng dụng thực tế dựa trên yêu cầu cụ thể và yêu cầu về hiệu suất của dự án.

2.3 Xây dựng Ứng dụng dự đoán URLs

2.3.1 Xây dựng chương trình trích xuất thuộc tính

Việc trích xuất đầy đủ các thuộc tính từ một URLs tương ứng với các thuộc tính mà ta đã quyết định cho việc huấn luyện mô hình trước đó là hết sức quan trọng. Vì chỉ khi trích xuất được đầy đủ thì ta mới có thể tiến hành đưa ra dự đoán một cách chính xác và rõ ràng.

Đối với 23 thuộc tính (biến độc lập) cần trích xuất, thì trong phần này ta sẽ tập trung về phần mã của việc trích xuất domain_age, google_index, page_rank. Vì 3 thuộc tính này cũng khá quan trọng và phức tạp trong việc trích xuất hơn các thuộc tính còn lại.

2.3.1.1 Thuộc tính domain_age

Thuộc tính này biểu thị tuổi đời của tên miền (domain). Một tên miền có tuổi đời dài thường được coi là đáng tin cậy hơn một tên miền mới đăng ký.

```

9  def domain_age(key, domain):
10     try:
11         r = 'whois'
12         domain = domain.split("//")[-1].split("/")[0].split('?')[0]
13         res = requests.get('https://api.whoapi.com', dict(domain=domain, r=r, apikey=key))
14         if res.status_code == 200:
15             data = res.json()
16             if int(data['status']) == 0:
17                 date_created = datetime.strptime(data['date_created'], "%Y-%m-%d %H:%M:%S")
18                 today = datetime.today()
19                 domain_age = today - date_created
20                 return domain_age.days
21             else:
22                 return -2
23         else:
24             return -1
25     except:
26         return -1

```

Mã nguồn trích xuất domain_age

Mô tả: Chúng ta sử dụng API whois để lấy thông tin về ngày tạo tên miền, sau đó tính toán số ngày kể từ ngày tạo đó đến ngày hiện tại.

Lỗi xử lý: Trả về -2 nếu không thể truy xuất dữ liệu domain, và -1 nếu không thể kết nối với API hoặc ngoại lệ xảy ra.

2.3.1.2 Thuộc tính google_index

Thuộc tính này kiểm tra xem URL có được lập chỉ mục bởi Google hay không. Một trang web được Google lập chỉ mục thường có mức độ uy tín cao hơn.

```

34 def google_index(url):
35     user_agent = 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/48.0.2564.116 Safari/537.36'
36     headers = {'User-Agent' : user_agent}
37     query = {'q': 'site:' + url}
38     google = "https://www.google.com/search?" + urlencode(query)
39     data = requests.get(google, headers=headers)
40     data.encoding = 'ISO-8859-1'
41     soup = BeautifulSoup(str(data.content), "html.parser")
42     try:
43         if 'Our systems have detected unusual traffic from your computer network.' in str(soup):
44             return -1
45         check = soup.find(id="rso").find("div").find("div").find("a")
46
47         if check and check['href']:
48             return 0
49         else:
50             return 1
51     except AttributeError:
52         return 1
53 
```

Mã nguồn trích xuất google_index

Mô tả: Sử dụng một truy vấn Google để kiểm tra xem URL có được lập chỉ mục hay không.

Lỗi xử lý: Trả về -1 nếu Google phát hiện lưu lượng truy cập không bình thường, và 1 nếu không tìm thấy kết quả.

2.3.1.3 Thuộc tính page_rank

Thuộc tính này đại diện cho xếp hạng trang (PageRank) của URL. Xếp hạng trang cao cho thấy trang web có mức độ uy tín và phổ biến cao hơn.

```

60 def page_rank(key, domain):
61     url = 'https://openpagerank.com/api/v1.0/getPageRank?domains[0]=' + domain
62     try:
63         res = requests.get(url, headers={'API-OPR':key})
64         data = res.json()
65         data = data['response'][0]['page_rank_integer']
66         if data:
67             return data
68         else:
69             return 0
70     except:
71         return -1

```

Mã nguồn trích xuất page_rank

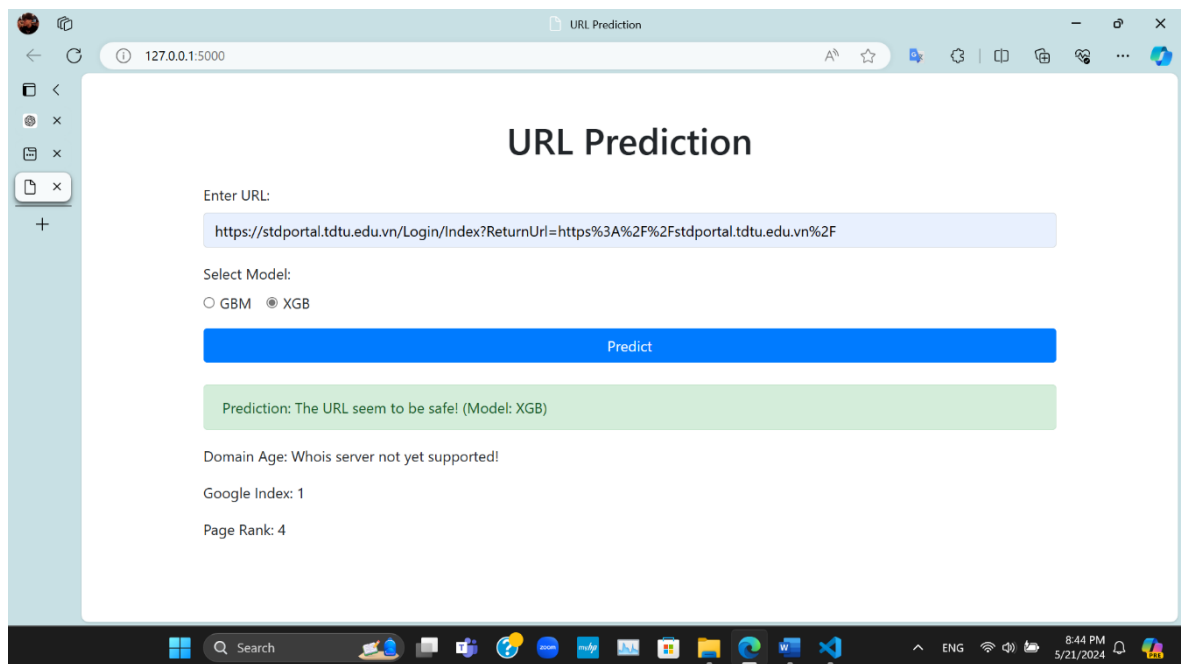
Mô tả: Sử dụng API của OpenPageRank để lấy xếp hạng trang của tên miền.

Lỗi xử lý: Trả về 0 nếu không có dữ liệu PageRank và -1 nếu ngoại lệ xảy ra.

2.3.2 Xây dựng Website

Cuối cùng, chúng ta sẽ xây dựng một website dự đoán để người dùng có thể nhập URL của một trang web và nhận lại dự đoán về tính đáng tin cậy của trang web đó. Các công việc cần thực hiện bao gồm:

2.3.2.1 Thiết kế giao diện người dùng thân thiện và dễ sử dụng.



Vì phạm vi của đề tài dừng lại ở việc thể hiện khả năng dự đoán của các mô hình học máy nên giao diện trang web sẽ được thiết kế ở mức đơn giản kèm với đó là có thể chọn mô hình theo ý muốn để thực hiện dự đoán kết quả.

2.3.2.2 Kết nối website với mô hình học máy đã huấn luyện để nhận dự đoán

Trong phần huấn luyện mô hình và đánh giá, ta chọn ra 2 mô hình có điểm số Test Accuracy cao nhất là XGB và GBM để phục vụ cho việc dự đoán.

```

40 def predict_phishing_gbm(features):
41     # Load the model
42     with open('../models/GBM.pickle.dat', 'rb') as f:
43         gbm_model = pickle.load(f)
44     # Make predictions
45     prediction = gbm_model.predict([features])
46     return prediction[0]
47
48 def predict_phishing_xgb(features):
49     # Load the model
50     with open('../models/XGB.pickle.dat', 'rb') as f:
51         xgb_model = pickle.load(f)
52     # Make predictions
53     prediction = xgb_model.predict([features])
54     return prediction[0]

```

Tải mô hình: Mô hình Gradient Boosting Machine (GBM) hoặc XGBoost được lưu trữ dưới dạng file pickle.

Dự đoán: Hàm predict của mô hình được gọi với danh sách các đặc trưng (features) của URL cần dự đoán. Vì hàm predict yêu cầu đầu vào là một danh sách các mẫu, chúng ta đặt features trong một danh sách.

Trả về kết quả: Hàm predict trả về một danh sách các dự đoán cho mỗi mẫu đầu vào. Vì ở đây chỉ có một mẫu đầu vào, chúng ta trả về phần tử đầu tiên của danh sách này (prediction[0]).

```

11 @app.route('/predict', methods=['POST'])
12 def predict():
13     data = request.get_json()
14     url = data['url']
15     model = data['model']
16
17     features = fe.extract_features(url)
18
19     if features is None:
20         return jsonify({'error': 'Failed to extract features from the URL', 'model': model}), 400
21
22     features = features[1:-1]
23
24     prediction = None
25     msg = None
26     model_type = model
27
28     if model == 'GBM':
29         prediction = predict_phishing_gbm(features)
30     elif model == 'XGB':
31         prediction = predict_phishing_xgb(features)
32
33     if prediction == 0:
34         msg = 'The URL seem to be safe!'
35     else:
36         msg = 'The URL may be a phishing site!'
37
38     return jsonify({'prediction': int(prediction), 'msg': msg, 'model': model_type, 'domain_age': features[-3], 'google_index': features[-2], 'page_rank': features[-1]})
39

```

Trong đoạn mã trên:

`@app.route('/predict', methods=['POST'])` xác định một endpoint `/predict` cho phương thức POST.

Dữ liệu được gửi lên qua JSON và trích xuất thông tin về URL và loại mô hình.

Tiến hành trích xuất các đặc trưng từ URL.

Dự đoán xem URL có phải là trang web lừa đảo hay không dựa trên mô hình đã chọn.

Trả về kết quả dự đoán cùng với thông tin về tuổi miền, chỉ số Google và PageRank.

TÀI LIỆU THAM KHẢO

Tiếng Việt

- [1] Minh T. P. *Từ A-Z về Phishing: Hình thức tấn công mạng cực kỳ nguy hiểm*. Link: <https://cloud.z.com/vn/news/phishing/>
- [2] Wikipedia contributors. (n.d.). *Tấn công giả mạo*.
Link: <https://vi.wikipedia.org/w/index.php>
- [3] (N.d.). Glints.com. Link: <https://glints.com/vn/blog/machine-learning-la-gi/>
- [4] Ait S. *Machine Learning là gì? Khái niệm, phân loại và ứng dụng của Machine Learning*. SOM AIT.
Link: <https://som.edu.vn/machine-learning-la-gi-phan-loai-quy-trinh-ung-dung/>
- [5] T.-Q. lý N. (2023, February 2). *Machine Learning là gì? Tìm hiểu về cách hoạt động và ứng dụng*.
Link: <https://tanca.io/blog/machine-learning-la-gi-tim-hieu-ve-cach-hoat-dong-va-ung-dung>

Tiếng Anh

- [1] "Phishing", Wikipedia. [Online].
Link: <https://en.wikipedia.org/wiki/Phishing>
- [2] "Social_engineering_(security)", Wikipedia. [Online].
Link: [https://en.wikipedia.org/wiki/Social_engineering_\(security\)](https://en.wikipedia.org/wiki/Social_engineering_(security))
- [3] "2020 Twitter account hijacking", Wikipedia. [Online].
Link: https://en.wikipedia.org/wiki/2020_Twitter_account_hijacking
- [4] "History of Phishing", Phishing.org. [Online].
Link: <https://www.phishing.org/history-of-phishing>
- [5] "Phishing URL Detection with Python and ML", ActiveState Blog. [Online].

Link: <https://www.activestate.com/blog/phishing-url-detection-with-python-and-ml/>

[6] "Detecting Phishing Websites Using Machine Learning", JavaTpoint. [Online].

Link: <https://www.javatpoint.com/detecting-phishing-websites-using-machine-learning>

[7] "OpenPageRank REST APIs Document", OpenPageRank. [Online].

Link: <https://www.domcop.com/openpagerank/documentation>

[8] "WHOAPI REST APIs Document", WHOAPI. [Online].

Link: <https://whoapi.com/how-to-use-whois-api/>