BÁO CÁO BÀI TẬP 1

**Môn học: Phương pháp học máy trong an toàn thông tin**

**Tên chủ đề: Feature Selection**

*GVHD: Phan Thế Duy*

1. **THÔNG TIN CHUNG:**

*(Liệt kê tất cả các thành viên trong nhóm)*

Lớp: ……….

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** | **Email** |
| 1 | Trần Hoàng Khang | 19521671 | [19521671@gm.uit.edu.vn](file:///D:\KHANG\DocumentForStudy\ML%20in%20Security\Excercise\Ex1\19521671@gm.uit.edu.vn) |
| 2 | Nguyễn Tú Ngọc | 20521665 | [20521665@gm.uit.edu.vn](mailto:1552yyyy@gm.uit.edu.vn) |
| 3 | Lê Hồng Bằng | 19520396 | [19520396@gm.uit.edu.vn](file:///D:\KHANG\DocumentForStudy\ML%20in%20Security\Excercise\Ex1\19520396@gm.uit.edu.vn)+ |

1. **NỘI DUNG THỰC HIỆN:[[1]](#footnote-1)**

|  |  |  |
| --- | --- | --- |
| **STT** | **Công việc** | **Kết quả tự đánh giá** |
| 1 | Phân tích chung về bộ dữ liệu | 100% |
| 1.1 | Tổng số nhãn, tổng số thuộc tính | 100% |
| 1.2 | Xem thông tin chung các giá trị của các thuộc tính, các giá trị **NA** | 100% |
| 1.3 | Thông tin về giá trị means, median, min, max, … (Làm thêm cho vui 🤞) | 100% |
| 2 | Tìm hiểu và thực hiện đầy đủ 03 nhóm phương pháp: **Wrapper, Filter, Intrinsic** trong *Feature Selection* | 100% |
| 2.1 | Dùng **Filter**, sử dụng các phương pháp loại bỏ feature bằng phương pháp truyền thống | 100% |
| a | Bỏ feature có nhiều giá trị ***NA (Not Available)*** | 100% |
| b | Bỏ feature bằng ***VarianceThreshold*** | 100% |
| c | Bỏ feature bằng các hệ số ước lượng bằng ***Pearson******Coefficient*** | 100% |
| 2.2 | Dùng **Wrapper**, sử dụng kỹ thuật ***Sequential Feature Selector*** | 100% |
| a | Chỉ thực hiện làm mẫu với phương pháp ***Sequential Forward Selection*** | 100% |
| 2.3 | Dùng **Intrinsic**, sử dụng các thuật toán phổ biến đối với phương pháp này | 100% |
| a | Dùng thuật toán ***Lasso***: tạo dataset, build model, train, xuất kết quả và tìm tham số tối ưu. Đồng thời so sánh mức độ hiệu quả với các thuật toán khác | 100% |
| b | Dùng thuật toán ***Ridge***: tạo dataset, build model, train, xuất kết quả và tìm tham số tối ưu. Đồng thời so sánh mức độ hiệu quả với các thuật toán khác | 100% |
| c | Dùng thuật toán ***ElasticNet***: tạo dataset, build model, train, xuất kết quả và tìm tham số tối ưu. Đồng thời so sánh mức độ hiệu quả với các thuật toán khác | 100% |
| 3 | Thực nghiệm trên model thực tế (một model khác) | 80% |
| 3.1 | Xây dựng model Machine Learning | 80% |
| a | Dùng Gradient Boosting | 80% |
| 3.2 | Xây dựng model Deep Learning | 80% |
| b | Dùng | 80% |

BÁO CÁO CHI TIẾT

**1.** Phân tích chung về bộ dữ liệu (tổng số nhãn, tổng số thuộc tính, phân phối của các  
nhãn dữ liệu…). Yêu cầu lập trình code để in các thông tin này ra màn hình quan sát.

Thông tin bộ dữ liệu gốc, từ paper: [Identifying Useful Features for Malware Detection](https://ieeexplore.ieee.org/document/8951564)

Table

Description automatically generated

a. (Dataset gốc)Thông tin số lượng các nhãn:

Text

Description automatically generated

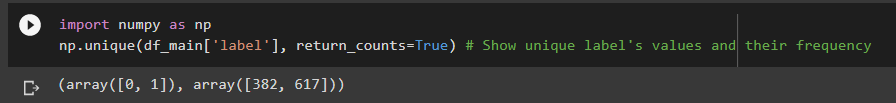
b. (Dataset khác) Thông tin chung:

Thừa nhận dataset của <https://ieee-dataport.org/authors/quynh-trinh> trong link sau: [Maliciou\_Benign\_PE\_file\_dataset](https://l.facebook.com/l.php?u=https%3A%2F%2Fdrive.google.com%2Fdrive%2Ffolders%2F1rXmo01fzWFgnUD0OF2qsphHWwV-1qtLg%3Fusp%3Dsharing%26fbclid%3DIwAR0EM2wxEJ8OAn9SCnPY2rppmU6nDbQZHvm8scXODBYms5En70hOfFVVF9E&h=AT1EoZprx4z879vFt5sUHcorjmaGs0uf9koP-mxH_3VZvY4RCVptCO5x4J1d_m60VftpjPwhOSMirbADezsPeRCzvlR9U9EdkzGgye7EgXk_ZZiwkBedui6O4vQcyeyNnpZYAdjEAQftCIh1QCMMYA).

**Note:***Việc trích xuất file Jsonl thành file CSV mình đã làm được nhưng dữ liệu đối với một số trường vẫn đang ở dạng Object, mình cần phải biến đổi thêm, tuy nhiên phần này mình chưa thực hiện nên để cho nhanh và tiện thì chúng ta công nhận một dataset khác*

Dataset này được trích xuất sẵn dựa vào dữ liệu 2017 (cũng thuộc bộ dữ liệu của EMBER) nên có ít thuộc tính hơn. Đồng thời, các features trong dataset chứa các dữ liệu và các thư viện được ***imported*** và ***exported***, được gán giá trị 0 (không) và 1(có). Song, mình thấy số lượng feature vẫn khá đầy đủ để phục vụ cho việc thực nghiệm Feature Selection

Tập dataset mới chỉ có 2 nhãn:



Sau đó ta loại bỏ các cột không cần thiết đi



Lực hiện lấy tập dữ liệu để rtain:



Thông tin về số lượng (mẫu, đặc trưng):

A screenshot of a computer

Description automatically generated with medium confidence

Số mẫu: 999

Số lượng feature: 1001

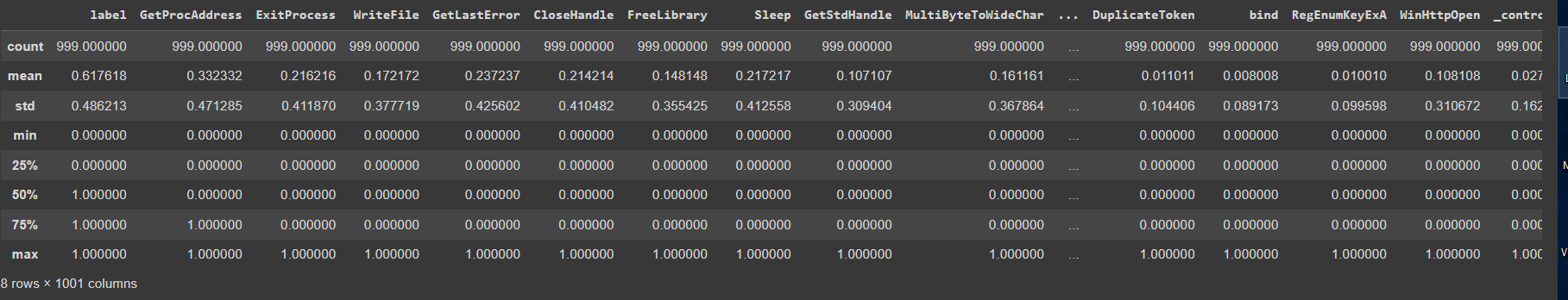
**Note:** *Mình thực hiện lấy 999 mẫu để demo chạy đỡ lâu. Nếu để > 10000 thì thời gian xử lý của các thuật toán ở các lần sau lên đến cả tiếng (tầm >5 tiếng)*

Xem tổng quát “hình dạng” của mẫu dữ liệu

Graphical user interface

Description automatically generated

Thông tin một số thông số đặc biệt *(count, means, average, min, max, …)*:



**2.** Feature Selection:

a. Sử dụng nhóm phương pháp Filter:

***Định nghĩa cơ bản***: Filter method là một trong 3 nhóm phương pháp chính cho Feature Selection và xếp hạng từng feature dựa trên một số chỉ số “đơn biến” (uni-variate) và sau đó chọn các feature có xếp hạng cao nhất. Cách tính toán chỉ số ranking này tùy thuộc vào thuật toán ta sử dụng

*Tham khảo*: <https://medium.com/mlearning-ai/feature-selection-using-filter-method-python-implementation-from-scratch-375d86389003>

i. Loại bỏ những feature có giá trị hằng (constant):

*Giải thích ngắn gọn*: Các feature không đổi là loại feature chỉ chứa một giá trị cho mọi đầu ra trong tập dữ liệu. Các feature cố định không cung cấp thông tin nào có thể giúp ích cho việc phân loại bản ghi hiện có. Do đó, nên xóa tất cả các feature này

Sau khi loại bỏ các feature nói trên:

Text

Description automatically generated

Số lượng feature hiện tại còn **956** (so với **1001**)

ii. Loại bỏ Quasi-Constant Features:

*Giải thích ngắn gọn* : Các feature *gần như không đổi (quasi),* như tên cho thấy, là các feature gần như không đổi. Các feature như vậy không hữu ích lắm để đưa ra dự đoán. *Không có quy tắc nào về ngưỡng cho phương sai của các quasi-feature.*

TIếp tục với số lượng feature trên, tạo một đối tượng VarianceThreshold với ngưỡng là 0.1

Text

Description automatically generated

Tuy nhiên, theo nguyên tắc chung, hãy loại bỏ các đặc trưng gần như không đổi có giá trị tương tự hơn 99% cho các quan sát đầu ra.

Sau khi loại bỏ các feature nói trên:

Text

Description automatically generated with low confidence

Số lượng feature hiện tại còn **612** (so với **956**)

iii. Loại bỏ Columns bị trùng nhau:

*Giải thích ngắn gọn* : Các feature trùng lặp là các feature có giá trị tương tự nhau. Các feature trùng lặp không bổ sung bất kỳ giá trị nào cho quá trình training thuật toán, thay vào đó chúng thêm chi phí hoạt động và độ trễ không cần thiết vào thời gian đào tạo. Do đó, luôn luôn nên xóa các feature trùng lặp khỏi tập dữ liệu trước khi đào tạo.

Tiếp tục với số lượng feature trên, ta loại bỏ các cột có giá trị trung nhau bằng cách xử lý dataframe như với dòng, rồi xóa bỏ các dòng có giá trị bị duplicated.

Hoán vị dataframe của chúng ta:

Text

Description automatically generated with medium confidence

Tìm những hàng bị trùng nhau và loại bỏ chúng

Graphical user interface, text, website

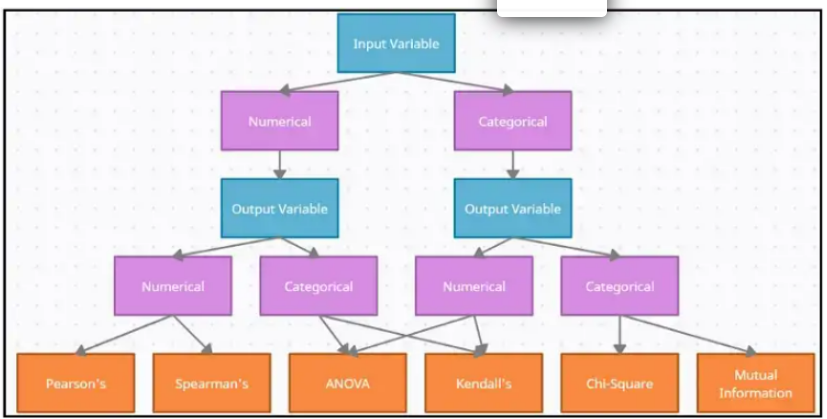
Description automatically generated

Số lượng feature hiện tại còn **581** (so với **612**)

iv. Tính hệ số tương quan (Correlation) với cột đích (label) và loại bỏ các hệ số thấp:

*Giải thích ngắn gọn:* Tính hệ số tương quan hay còn gọi mức độ tương quan (Correlation) với cột đích (label) bằng phương pháp Pearson và chọn các cột có hệ số tương quan cao sao cho “phù hợp”, sự “phù hợp” này tùy theo số lượng feature cuối cùng chúng ta chọn

Cách chọn thuật toán phù hợp nhất để xác định mối tương quan của tính năng



Trong bài tập này, mình sẽ thử phương pháp được đề xuất đầu tiên: *Hệ số Pearson*

Nguồn: <https://medium.com/analytics-vidhya/feature-selection-in-machine-learning-ec1f5d053007>

TIếp tục sử dụng các feature đã được trích xuất trên, ta tính toán các hệ số Pearson

Graphical user interface, text

Description automatically generated

Sắp xếp lại cho dễ nhìn bằng numpy.argsort(array) và in ra màn hình:

Graphical user interface, text

Description automatically generated

Vẽ biểu đồ để xem thông tin vùng giá trị:

A picture containing graphical user interface

Description automatically generated

Sau khi thử một số threshold (ngưỡng), mình chọn 0,1 để thu hẹp số lượng feature

A screenshot of a computer

Description automatically generated with medium confidence

Số lượng feature hiện tại còn **245** (so với **581**)

Và đó là một chuỗi các chiến lược khi dùng phương pháp Filter, từ **1001** thành **245** feature

b. Sử dụng nhóm phương pháp Wrapper:

***Định nghĩa cơ bản*** Wrapper method đánh giá nhiều mô hình bằng cách sử dụng các quy trình thêm và/hoặc loại bỏ các yếu tố dự đoán để tìm ra sự kết hợp tối ưu giúp tối đa hóa hiệu suất của mô hình. Các quy trình này thường được xây dựng sau khi khái niệm về thuật toán *“Tìm kiếm tham lam.”* ra đời. Thuật toán tham lam là bất kỳ thuật toán nào tuân theo kinh nghiệm giải quyết vấn đề để đưa ra lựa chọn tối ưu cục bộ ở mỗi giai đoạn. Và những thuật toán này sẽ cố overfitting dữ liệu của chúng ta.

**Sequential Feature Selector** là một chiến lược để chọn một tập hợp con các tính năng có hiệu suất tốt nhất. Điều này bao gồm 4 chiến lược chính:

* Sequential Forward Selection (SFS)
* Sequential Backward Selection (SBS)
* Sequential Backward Floating Selection (SBFS)
* Sequential Forward Floating Selection (SFFS) Chúng tôi sử dụng sklearn để biến các thử nghiệm thành hành động

Tham khảo (Giải thích đầy đủ):

<http://rasbt.github.io/mlxtend/user_guide/feature_selection/SequentialFeatureSelector/>

Các thông số đánh giá: <https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SequentialFeatureSelector.html>

Ta lấy lại dataset ban đầu với số chiều là (999, 1001). Ở đây, mình sẽ sử dụng chiến lược đầu tiên: **Sequential Forward Selection**

Tạo một model với thuật toán **Kneighbor**:

A screenshot of a computer

Description automatically generated with medium confidence

Tạo bộ chọn Wrapper với chiến lược nói trên và đưa dữ liệu vào tập train.

**Note:***Việc train bằng phương pháp Wrapper hay các thuật toán tham lam nói chung đều rất lâu, do vậy bước này sẽ khá take time* 💖

Text

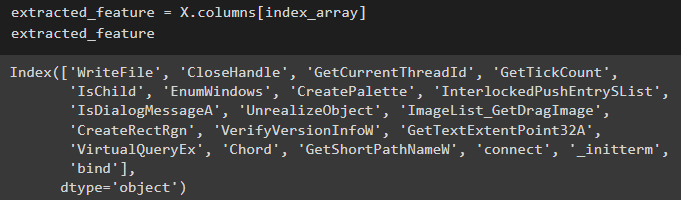
Description automatically generated

Lấy ra những index của các feature quan trọng sau khi train:

Graphical user interface, text

Description automatically generated

Với index tìm được, xem thông tin tên feature.



A picture containing graphical user interface

Description automatically generated

Vậy số lượng feature cuối cùng là **20** (so với **1001**)

c. Sử dụng nhóm phương pháp Intrinsic(Embedded) Method:

Các phương thức **Embedded** kết hợp các phẩm chất của **Filter** và **Wrapper**. Nó được triển khai bởi các thuật toán có các phương thức lựa chọn tính năng tích hợp sẵn của riêng chúng. Bản chất, **Embedded** là phương pháp bỏ dữ liệu của chúng ta vào một model thực và liên tục đánh giá và tối ưu nó

Một số ví dụ phổ biến nhất của các phương pháp này là hồi quy LASSO và RIDGE có các chức năng xử phạt sẵn có để giảm tình trạng thừa. Ngoài ra, còn có thuật toán ElasticNet được đề xuất là sự kết hợp của cả hai.

**Note:***Trong file .ipynb, mình triển khai cả ba thuật toán nói trên để so sánh mức độ hiệu quả*

Ta lấy lại dataset ban đầu với số chiều là (999, 1001).

i. Lasso algorithm:

Tạo model cho thuật toán và fit() dữ liệu của chúng ta vào model

*Text

Description automatically generated*

**Note** *Khác với Lasso, thư viện LassoCV của sklearn cho phép tìm các tham số tối ưu nhất khi chạy model*

*Text

Description automatically generated*

Đo lường các thông tin sai số:

<Bổ sung>

*Text

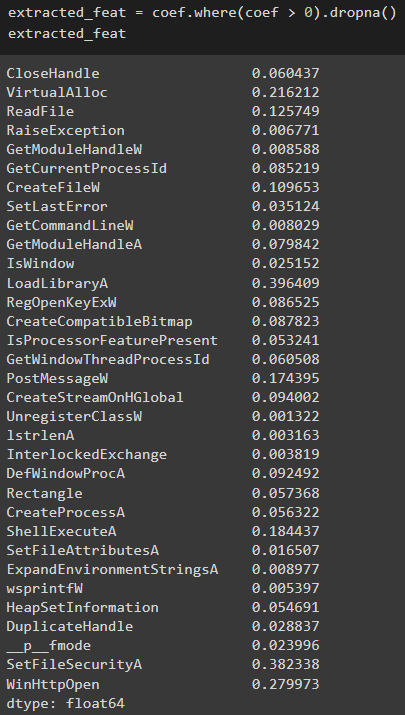
Description automatically generated*

Phần trên là giai đoạn training model, quá trình diễn ra khá nhanh, tương tự như **Wrapper** method. Tiếp theo đến phần trích xuất ra các hệ số đánh giá theo tư duy của **Filter** method

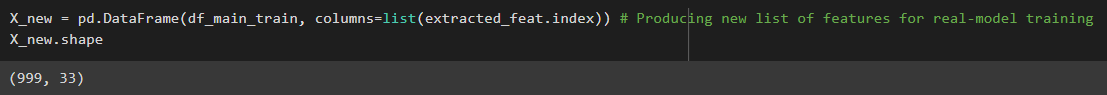
*Graphical user interface, text

Description automatically generated*

Chọn các đặc trưng theo một “ngưỡng” (threhold) xác định. Ở đây mình chọn **0**

**

Số feature được trích xuất là **33** (so với **1001** ban đầu)



Thực hiện tương tự với 2 thuật toán **Ridge** và **ElasticNet**. Ta có bảng so sánh sau

3. Áp dụng trên mô hình thực tế với ML và DL:

a. Machine Learning:

* Tách bộ train\_test\_plit từ sklearn.model\_selection
* Do dữ liệu vào model để huấn luyện với các tham số mặc định

Text

Description automatically generated

Kết quả các thông số cơ bản: **Accuracy, Recall, Precision và F1-score**

Text

Description automatically generated

b. Deep Learning:

Thực hiện với model Deep learning với thông tin được sử dụng sau:

|  |  |
| --- | --- |
| **Tên loại** | **Giá trị** |
| Loại model | Sequential() |
| Số lượng hidden Layer | 3 |
| Layer 1 (lớp input): số lượng node | 12 |
| Activation function | Relu |
| Layer 2: số lượng node | 8 |
| Activation function | Relu |
| Layer 3: số lượng node | 1 |
| Activation function | Sigmoid |
| Hàm loss | binary\_crossentropy |
| Hàm optimizer | Adam |
| Metric để đánh giá | Accuracy |
| Epoch | 150 |
| Batch size | 10 |

Text

Description automatically generated

Kết quả khi train xong:

Text

Description automatically generated

1. Ghi nội dung công việc, các kịch bản trong bài Thực hành [↑](#footnote-ref-1)