|  |  |
| --- | --- |
| Giảng viên hướng dẫn: | **Nguyễn Tấn Cầm** |
| Sinh viên thực hiện: |  |
|  | Nguyễn Lý Đinh Nhì 18521205 |
|  | Trần Anh Vũ 18520401 |



**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**BÁO CÁO**

**MÔN HỌC: Kỹ thuật phân tích mã độc - NT137.M11.ATCL**

**ĐỀ TÀI:**

**PHÁT HIỆN ỨNG DỤNG ĐỘC HẠI DỰA TRÊN PHƯƠNG PHÁP TRÍCH XUẤT PERMISSION**

Contents

[I. Giới thiệu đề tài 3](#_Toc89280586)

[II. Phương pháp thực hiện 3](#_Toc89280587)

[III. Xử lý dữ liệu 6](#_Toc89280588)

[IV. Xây dựng model 7](#_Toc89280589)

[1. SVM 7](#_Toc89280590)

[2. Random Forest 8](#_Toc89280591)

[3. CNN 9](#_Toc89280592)

[V. Cách thức lựa chọn số lượng permission 11](#_Toc89280593)

[VI. Đánh giá hiệu xuất khi trích xuất top 20, 50 và 100 permission đối với bộ dataset AndMal2017 15](#_Toc89280594)

[VII. Đánh giá model khi trích xuất top 50 permision 18](#_Toc89280595)

[1. CICDataset: AndMal2017 18](#_Toc89280596)

[2. MalDroid-2020 19](#_Toc89280597)

[3. Kết luận 19](#_Toc89280598)

[VIII. References 20](#_Toc89280599)

# Giới thiệu đề tài

Ngày nay, các thiết bị smartphone được sử dụng rộng rãi và bởi khả năng truy cập thông tin bí mật và riêng tư quan trọng của chúng đã dẫn đến việc các thiết bị này trở thành mục tiêu của các nhà phát triển phần mềm độc hại. Việc phát hiện phần mềm độc hại là một vấn đề rất được quan tâm, đặc biệt là trên nền tảng di động Android do tính phổ biến và khả năng tiếp cận nhiều ứng dụng của bên thứ ba. Điều này còn trở nên tồi tệ hơn bởi các kỹ thuật tránh phát hiện ngày càng tinh vi được sử dụng bởi các dòng phần mềm độc hại mới nổi. Các kỹ thuật phân tích phần mềm độc hại Android hiện tại có thể được phân loại rộng rãi thành phân tích tĩnh và động. Trong đồ án này, nhóm xin trình bày phương pháp phân loại mã độc dựa trên các mô hình SVM, Random Forest và CNN.

# Phương pháp thực hiện

Đối với các ứng dụng Android, mọi hoạt động của chúng điều gắn liền với các permission mà chúng yêu cầu với hệ thống, điều này có nghĩa là ta hoàn toàn có khả năng dự đoán được các hành động của ứng dụng mà chỉ cần dựa vào danh sách các permission mà chúng yêu cầu. Nắm được điều này nhóm đã áp dụng phương pháp máy học để phát hiện các ứng dụng độc hại dựa tiêu chí đánh giá là các permission.

Tuy nhiên, áp dụng học máy dựa trên toàn bộ các permission thu được sẽ ảnh hưởng đến thời gian training, điều này sẽ là một điểm trừ khi train model với bộ dataset lớn, mặc dù nếu ta sử dụng toàn bộ danh sách permission thu được thì tỉ lệ dự đoán sẽ cao hơn nhưng chi phí (CPU, thời gian) bỏ ra là quá lớn. Để khắc phục điều này thì ta có thể lựa chọn những permission thể hiện đặc trưng của các loại mã độc như thế sẽ giúp tăng tốc độ xử lý nhưng cũng không gây sai số quá nhiều so với phương pháp ban đầu.

**\* Tại sao nhóm lại sử dụng phương pháp trích xuất permission mà không sử dụng các phương pháp khác.**

Đối với việc áp dụng machine learning vào phát hiện mã độc thì trên thực tế có rất nhiều phương pháp đã và đang được áp dụng, có thể kể đến như phương pháp phân tích Davik bytecode, sử dụng các API làm đặc trưng, xây dựng mô hình mạng lưới thể hiện tính liên kết giữa các API hay có thể là sự kết hợp của nhiều phương pháp khác lại với nhau. Trong đó một trong những phương pháp đơn giản và đem lại kết quả tương đối tốt đó là phương pháp trích xuất permission của ứng dụng, lý do có thể kể đến đó là việc trích xuất permission có thể thực hiện dễ dàng, và từng permission thể hiện một hoặc một nhóm các hoạt động có liên quan đến nhau, điều này sẽ giúp giảm thiểu lượng thông tin cần thu thập nhưng vẫn đảm bảo khái quát những hoạt động mà ứng dụng sẽ thực hiện.

Tuy nhiên, phương pháp này cũng tồn tại một thiếu sót khá lớn, điều đó xảy ra khi một ứng dụng độc hại che dấu nó bằng cách thêm một số lượng đáng kể các permission của ứng dụng thông thường.

**\* Các bước thực hiện:**

* B1: Trích xuất thông tin permission của các ứng dụng Android (gôm có các loại: Adware, Benign, Ransomeware, SMSmalware, Scareware).
* B2: Tìm đặc trưng, ta sẽ chọn ra 50 permission đặc trưng nhất của mỗi loại, bằng cách thống kê số lần xuất hiện.
* B3: Tổng hợp các permission đã lựa chọn.

Cụ thể các bước được thực hiện như sau:

**\* Trích xuất thông tin permission:**

Việc trích xuất được thực hiện dựa trên thư viện androguard, trong đó thông tin permission được lưu trong biến permission\_list, bên cạnh đó ta cũng sẽ lưu lại tên của ứng dụng được trích xuất, mặc dù việc lưu tên có thể không cần thiết nhưng nó sẽ giúp phân biệt các danh sách các permission thu được.



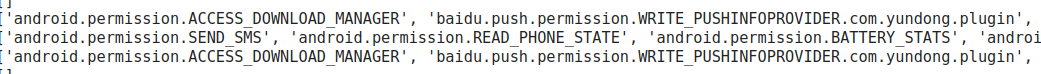
Hình Trích xuất permission từ app

**\* Chọn đặc trưng:**

Ta sẽ chọn ra đặc trưng của từng loại: Adware, BankingMalware, Benign, Riskware, SMSmalware.

Việc lựa chọn đặc trưng có nghĩa là ta sẽ chọn ra các permission được sử dụng nhiều nhất của từng loại ứng dụng, bằng cách thống kê các permssion đã thu được.

Các permission được trích xuất khi sử dụng androguard thường sẽ có dạng như sau:



Hình Ví dụ về permission sau trích xuất

Để thuận tiện cho quá trình thống kê ta sẽ chỉ lấy tên các permission và bỏ qua địa chỉ nguồn của chúng, vd: đối với android.permission.ACCESS\_DOWNLOAD\_MANAGER -> sẽ thành ACCESS\_DOWNLOAD\_MANAGER.

Đoạn code thực hiện chức năng này sẽ như sau:



Hình : Xóa các permission lỗi

Hàm remove\_source\_permission có tham số remove\_non, tham số này có ý nghĩa sẽ bỏ qua hoặc là chuyển các thông tin bị lỗi xuất hiện trong quá trình trích xuất sang 'null', thông thường các thông tin lỗi sẽ bị bỏ qua nên tham số remove\_non được mặc định là True.

Khi thực hiện thống kê các permission ta sẽ chọn ra 1 lượng permission đươc sử dụng nhiều nhất, dưới đây là ví dụ với tập dataset Benign:

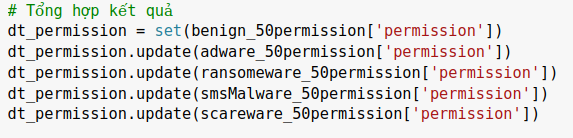


Hình : Lựa chọn các permisison xuất hiện nhiều nhất

Dataset của các loại còn lại thực hiện tương tự.

**\* Tổng hợp kết quả:**

Để phân loại được các ứng dụng khi sử dụng machine learning ta sẽ tổng hợp dặc trưng của các loại lại với nhau:



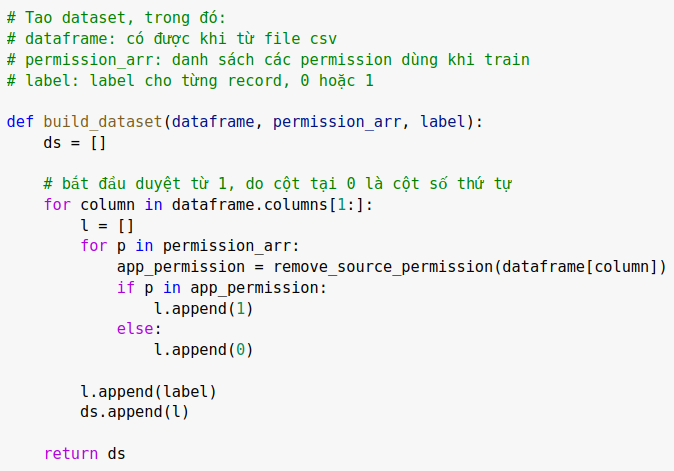
Hình : Tổng hợp các permission được sử dụng nhiều nhất

Như vậy lúc này ta đã hoàn thành tổng hợp lại kết quả, do đó tiếp theo ta sẽ xử lý dữ liệu dựa trên danh sách vừa thu được và xây dựng model đánh giá phương pháp này.

# Xử lý dữ liệu

Ta sẽ xây dựng bộ dataset dùng trong training dựa trên dữ liệu permission được trích xuất ban đầu và danh sách các thuộc tính thu được trong bước lựa chọn đặc trưng.

Do khi training các model chỉ tương tác với các kiểu dữ liệu dạng binary, nên việc xây dựng dataset sẽ dựa trên nguyên tắc là: giá trị ‘1’ có nghĩa permission của file apk có tồn tại trong danh sách thuộc tính, ngược lại ‘0’, bên cạnh đó ta sẽ thêm một cột label để đánh dấu file apk thuộc loại malware nào ( Adware, BankingMalware, Riskware, SMSmalware, …).



Hình : Hàm chuyển dataset sang dạng binary

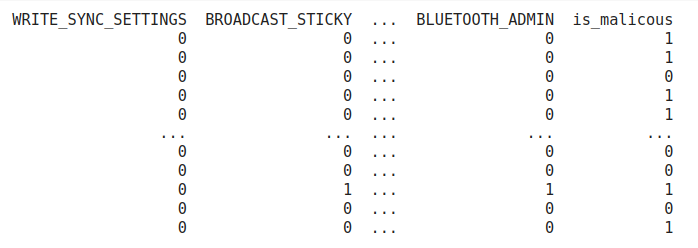
Khi này ta chỉ cần áp dụng hàm build\_dataset với các bộ dataset đã có. Do đây là bài toán về phát hiện mã độc nên số lượng label sẽ tương ứng với số loại malware mà ta xét. Sau khi xây dựng bộ dataset cho training thì ta sẽ xáo trộn bộ dataset để kết quả khi train có tính thực tế cao hơn.

Graphical user interface, text, application, email

Description automatically generated

Hình : Chuyển toàn bộ dataset sang binary

Kết quả sau khi xây dựng bộ dataset



Hình : Minh họa kết quả sau khi chuyển đổi sang binary

# Xây dựng model

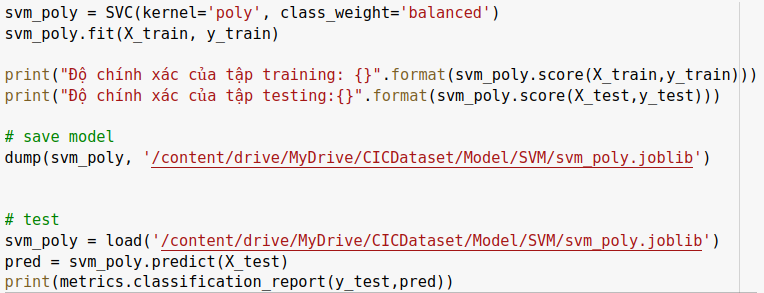
Do mỗi model đều có ưu và nhược điểm riêng, nên nhóm quyết định xây dựng mô hình training dựa trên 3 loại thuật toán: SV, RandomForest, CNN. Bằng cách sử dụng nhiều loại model điều này sẽ giúp xác định được đâu là mô hình tối ưu nhất.

## SVM

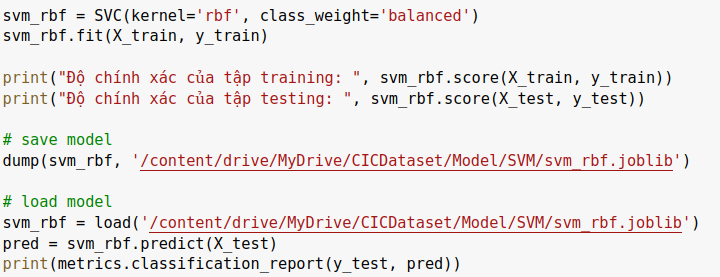
SVM hay Support Vector Machine là một thuật toán học có giám được áp dụng trong việc phân chia dữ liệu hành các nhóm riêng biệt. SVM hoạt động dựa trên việc xác định các ranh giới để phân chia các nhóm dữ liệu. Các đường ranh giới này có thể là đường thẳng, đường cong hoặc thậm chí là các siêu mặt phẳng. Hình dạng của các đường ranh giới phụ thuộc vào độ phức tạp của bộ dữ liệu và kernel mà chúng ta chọn.

Đối với đề tài này nhóm đã sử dụng 2 loại kernel được hỗ trơ bởi SVM đó là ‘poly’ và ‘rbf’, hai kernel này có thể tạo ra các đường ranh giới phức tạp phù hợp với độ phức tạp của bộ dữ liệu nên sẽ cho kết quả tốt hơn.

Việc sử dụng SVM rất đơn giản ta chỉ cần sử dụng thư viện sklearn.svm của python.

\* Kernel Poly:

Hình : thuật toán SVM - Poly kernel

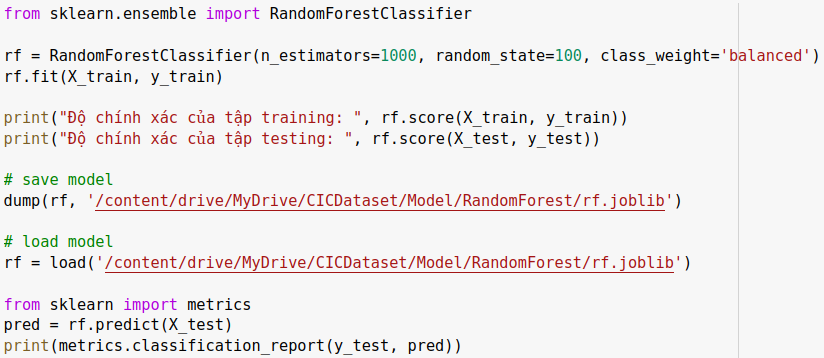
\* Kernel Rbf:

Hình : thuật toán SVM - Rbf kernel

## Random Forest

Random Forest là thuật toán dự đoán kết quả dựa trên việc xây dựng nhiều cây quyết định, trong đó mỗi cây quyết định dùng tập dữ liệu và thuộc tính khác nhau được trích xuất từ tập dữ liệu gốc ban đầu. Cuối cùng, kết quả của Random Forest sẽ là kết quả được tổng hợp từ đầu ra của các cấy quyết định.

Đối với bài toán phát hiện phần mềm độc hại, nhóm đã xử dụng RandomForestClassifier có trong thư viện sklearn.ensemble.



Hình : thuật toán Random Forest

## CNN

CNN là một mô hình phức tạp, được mô phỏng dựa trên mới lưới nerual của não bộ con người.Quá trình cho kết quả khi sử dụng CNN trải qua nhiều lớp, với mỗi lớp gồm nhiều node khác nhau, các node ở lớp sau sử dụng kết quả của lớp trước đó tạo nên một mạng lưới xử lý phức tạp. Độ phức tạp của thuật toán tùy thuộc vào số node và số lớp mà lập trình viên định nghĩa.

Đối với bài toán phân loại mã độc nhóm đã xây dựng 2 mô hình, với sô lớp lẫn lược là 1 và 4.

Thuật toán CNN là một mô hình chuyên về phân loại ảnh, do đó cần thực hiện việc chuyển các thông tin của từng record trong bộ dataset vừa được xây dựng sang dạng ảnh.

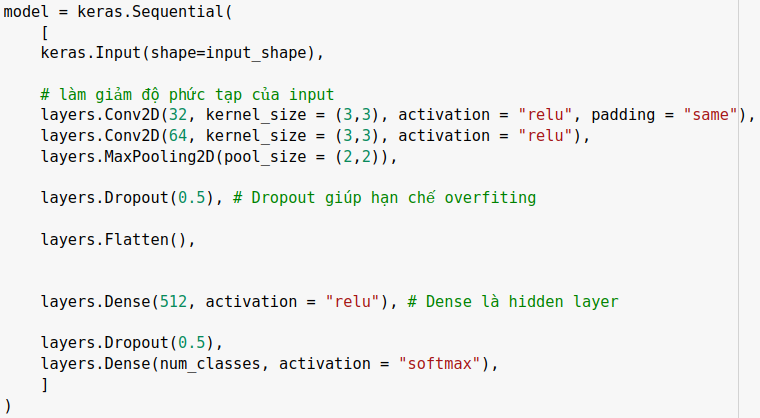
Text

Description automatically generated

Hình : chuyển dataset từ binary sang ảnh

**Xây dựng model**

Một trong những bước quan trong khi xây dựng model CNN là cần phải làm giảm độ phức tạp của đầu vào để làm giảm độ phức tạp cũng như là thời gian xử lý của mô hình. Do đó ta thường sẽ thêm các layer phụ để xử lý đầu vào.

\* Model có 1 hidden layer

Hình : CNN - 1 hidden layer

Graphical user interface, text

Description automatically generated\* Model có 4 hidden layer

Hình : CNN - 4 hidden layer

Có thể thấy ở 2 mô hình CNN trên tại các layer có sử dụng các tham số activation, đây là một tham số vô cùng quan trọng, nó chỉ ra các hàm được sử dụng trong quá trình tính toán để đưa ra output. Trong đó 2 hàm thông dụng nhất là ‘relu’ và ‘softmax’, đối với ‘relu’ thì thường nó được sử dụng tại các hidden layer trong CNN, có kết quả tính toán nhanh và ít sai sót từ đó giúp model học chính xác hơn, còn đối với ‘softmax’, đây là hàm thường được dùng tại layer output trong trường hợp multi-classification, nó giúp xác định tỉ lệ phù hợp đối với từng label nếu output của model có nhiều label khác nhau.

Đối với bài toán phát hiện mã độc của, layer ouput có thể sử dụng hàm actication ‘relu’ vì nó chỉ cho 1 trong 2 kết quả 0 hoặc 1 (là malware hoặc không), tuy nhiên nhóm đã sử dụng hàm ‘softmax’ để có cái nhìn tốt hơn.

# Cách thức lựa chọn số lượng permission

Việc xác định malware của nhóm được thực hiện dựa trên việc xác định các permission mà ứng dụng đó yêu cầu, vì lý do đó việc xây dựng một bộ dataset phù hợp là vô cùng quan trọng, cụ thể là ta cần phải xác định được bộ dataset cần chứa những permission nào. Có hai cách để giải quyết vấn đề này đó là chúng ta sẽ sư dụng toàn bộ permission có trong ứng dụng, tuy nhiên cách này sẽ khiến cho bộ dataset trở nên rất lớn gây ảnh hưởng đến thời gian train, đồng thời có khả năng không đạt được kết quả mong muốn do data về malware cao hơn rất nhiều so với benign. Phương pháp còn lại mà nhóm đã sử dụng là lựa chọn ra các permission được sử dụng nhiều nhất đối với từng loại ứng dụng (benign, malware: adware, sms malware, …) rồi tổng hợp tất cả lại với nhau. Khi sử dụng phương pháp này thì vấn đề khó khăn sẽ gặp phải đó là nên chọn ‘top’ bao nhiêu permission là tốt nhất, đây là một câu hỏi rất khó giải quyết, bởi vì đối với mỗi bộ dữ liệu mà ta xem xét thì có thể tần xuất xuất hiện của các permission sẽ khác nhau, chưa kể đến việc dữ liệu khi chạy thực tế sẽ khác so với bộ dữ liệu training. Do đó việc xác định các permission để xây dựng bộ dữ liệu training chỉ mang tính chất tương đối và có thể khác nếu sử dụng bộ dataset khác.

Nhóm đã sử dụng 2 bộ dataset AndMal2017 và MalDroid2020 trong quá trình train. Tuy nhiên, nhóm sẽ sử dụng bộ dataset AndMal2017 để nói về phương pháp lựa danh sách permission mà nhóm đã thực hiện.

AndMal2017 bao gồm các loại ứng dụng sau: Adware, Ransomeware, Scareware, SMS Malware, Benign 2017. Tổng cộng số record có được sau khi trích xuất bao gồm 1021 record. Sau đây là sơ đồ thể hiện số lượng permission của các ứng dụng:

Chart

Description automatically generated with medium confidence  
Rectangle

Description automatically generated with medium confidence

Hình : thống kê số lượng permission của Adware

Hình : thống kê số lượng permission của Benign

A picture containing rectangle

Description automatically generatedChart, line chart

Description automatically generated

Hình : thống kê số lượng permission của Ransomeware

Hình : Thống kê số lượng permission của SMSmalware

Chart

Description automatically generated

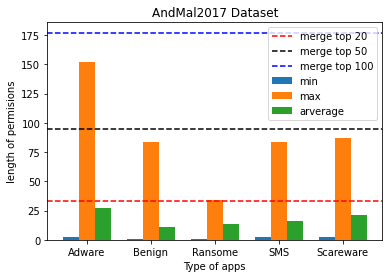
Hình : thống kế số lượng permission của Scareware

Dựa theo các sơ đồ biểu diễn ta có thể thấy rằng hầu hết các ứng dụng đều có dưới 50 permission (~ 95%) ngoại trừ một số trường hợp cá biệt. Do đó việc chọn top 50 permission của từng loại có thể xem là một sự lựa chọn hợp lý. Tuy nhiên có thể thấy rằng trên 50% ứng dụng của từng loại đều có dưới 20 permission, thậm chí đối với 2 loại Benign và Ransomeware con số này lên đến khoảng 80%.

Từ những thông tin đã phân tích có thể kết luận rằng không có một giá trị chính xác để lựa chọn số lượng permission, đều này phụ thuộc và đánh giá của từng người và từng bộ dataset mà họ sử dụng.

Để có được cái nhìn tổng quan hơn về phương pháp này, nhóm đã quyết định tạo bộ dataset dựa trên 3 mức độ gồm: 20, 50 và 100 permissions, sau đó thực hiện so sánh về độ chính xác cũng như là thời gian train giữa chúng.

Sau đây là sơ đồ đánh giá sơ bộ về bộ dataset AndMal2017, bao gồm các thông tin sô lượng permission min, max và trung bình của từng loại, bên cạnh đó sơ đồ cũng thể hiện tổng số lượng permission sao khi lấy top 20, 50 và 100 để thể hiện mức độ bao phủ của chúng.



Hình 20: so sánh số lượng permission của các loại app

# Đánh giá hiệu xuất khi trích xuất top 20, 50 và 100 permission đối với bộ dataset AndMal2017

## Các thuộc tính xây dựng dataset ở vị trí ngẫu nhiên

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Top 20 | Total: 1021 records | | | | | | |
|  | Train: 816 records | |  | Test: 205 records | | | |
|  | Total train time | accuracy training | Precision | Recall | F1 score | Accuracy |
| SVM Poly | ~0.04s | 0.94 | 0: 0.87  1: 0.69  2: 0.89  3: 0.89  4: 0.92 | 0: 0.96  1: 0.87  2: 0.85  3: 0.73  4: 0.46 | 0: 0.91  1: 0.77  2: 0.87  3: 0.80  4: 0.61 | 0.85 |
| SVM Rbf | ~0.05s | 0.92 | 0: 0.96  1: 0.68  2: 0.83  3: 0.78  4: 0.75 | 0: 0.92  1: 0.91  2: 0.95  3: 0.82  4: 0.50 | 0: 0.94  1: 0.78  2: 0.88  3: 0.80  4: 0.60 | 0.92 |
| Random Forest | ~2.04s | 0.98 | 0: 0.95  1: 0.71  2: 0.83  3: 0.86  4: 1.00 | 0: 0.97  1: 0.87  2: 0.95  3: 0.82  4: 0.62 | 0: 0.96  1: 0.78  2: 0.88  3: 0.84  4: 0.77 | 0.90 |
| CNN 1 layer | ~17.43s | 0.93 |  | 0: 0.96  1: 1.00  2: 0.89  3: 1.00  4: 1.00 | 0: 0.99  1: 0.96  2: 1.00  3: 0.91  4: 0.83 | 0: 0.98  1: 0.98  2: 0.94  3: 0.95  4: 0.91 | 0.97 |
| CNN 4 layer | ~20.77s | 0.96 |  | 0: 1.00  1: 1.00  2: 0.89  3: 0.92  4: 1.00 | 0: 0.98  1: 1.00  2: 1.00  3: 1.00  4: 0.94 | 0: 0.99  1: 1.00  2: 0.94  3: 0.96  4: 0.97 | 0.98 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Top 50 | Total: 1021 records | | | | | | |
|  | Train: 816 records | |  | Test: 205 records | | | |
|  | Total train time | accuracy training | Precision | Recall | F1 score | Accuracy |
| SVM Poly | ~0.04s | 0.94 | 0: 0.81  1: 0.67  2: 1.00  3: 0.94  4: 0.71 | 0: 0.96  1: 0.63  2: 0.63  3: 0.68  4: 0.50 | 0: 0.88  1: 0.65  2: 0.77  3: 0.79  4: 0.59 | 0.81 |
| SVM Rbf | ~0.06s | 0.94 | 0: 0.87  1: 0.71  2: 0.75  3: 0.86  4: 0.75 | 0: 0.92  1: 0.63  2: 0.95  3: 0.76  4: 0.50 | 0: 0.89  1: 0.67  2: 0.84  3: 0.81  4: 0.60 | 0.83 |
| Random Forest | ~2.07s | 0.98 | 0: 0.86  1: 0.91  2: 0.89  3: 0.83  4: 0.83 | 0: 0.97  1: 0.53  2: 0.89  3: 0.80  4: 0.62 | 0: 0.91  1: 0.67  2: 0.89  3: 0.82  4: 0.71 | 0.86 |
| CNN 1 layer | ~45.34s | 0.97 |  | 0: 0.98  1: 1.00  2: 1.00  3: 1.00  4: 1.00 | 0: 1.00  1: 1.00  2: 1.00  3: 1.00  4: 0.92 | 0: 0.99  1: 1.00  2: 1.00  3: 1.00  4: 0.96 | 0.99 |
| CNN 4 layer | ~27.08s | 0.96 |  | 0: 0.98  1: 1.00  2: 1.00  3: 1.00  4: 1.00 | 0: 1.00  1: 1.00  2: 1.00  3: 1.00  4: 0.92 | 0: 0.99  1: 1.00  2: 1.00  3: 1.00  4: 0.96 | 0.99 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Top 100 | Total: 1021 records | | | | | | |
|  | Train: 816 records | |  | Test: 205 records | | | |
|  | Total train time | accuracy training | Precision | Recall | F1 score | Accuracy |
| SVM Poly | ~0.08s | 0.91 | 0: 0.88  1: 0.80  2: 0.87  3: 0.73  4: 0.76 | 0: 0.94  1: 0.77  2: 0.72  3: 0.69  4: 0.65 | 0: 0.91  1: 0.78  2: 0.79  3: 0.71  4: 0.79 | 0.85 |
| SVM Rbf | ~0.09s | 0.93 | 0: 0.93  1: 0.77  2: 0.80  3: 0.87  4: 0.78 | 0: 0.94  1: 0.77  2: 0.89  3: 0.81  4: 0.70 | 0: 0.93  1: 0.77  2: 0.84  3: 0.84  4: 0.74 | 0.88 |
| Random Forest | ~2.31s | 0.98 | 0: 0.92  1: 0.86  2: 0.84  3: 0.88  4: 0.88 | 0: 0.98  1: 0.69  2: 0.89  3: 0.94  4: 0.70 | 0: 0.95  1: 0.77  2: 0.86  3: 0.91  4: 0.78 | 0.90 |
| CNN 1 layer | ~73.35s | 0.97 |  | 0: 1.00  1: 1.00  2: 0.87  3: 1.00  4: 1.00 | 0: 0.98  1: 1.00  2: 1.00  3: 1.00  4: 0.94 | 0: 0.99  1: 1.00  2: 0.93  3: 1.00  4: 0.97 | 0.99 |
| CNN 4 layer | ~63.74s | 0.97 |  | 0: 0.99  1: 1.00  2: 1.00  3: 1.00  4: 0.78 | 0: 0.98  1: 0.94  2: 0.85  3: 1.00  4: 1.00 | 0: 0.99  1: 0.97  2: 0.92  3: 1.00  4: 0.88 | 0.97 |

## Các thuộc tính xây dựng dataset được giữ theo thứ tự giảm dần tần suất xuất hiện

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Top 20 | Total: 1021 records | | | | | | |
|  | Train: 816 records | |  | Test: 205 records | | | |
|  | Total train time | accuracy training | Precision | Recall | F1 score | Accuracy |
| SVM Poly | ~0.03s | 0.94 | 0: 0.87  1: 0.86  2: 0.64  3: 0.84  4: 0.87 | 0: 0.93  1: 0.78  2: 0.88  3: 0.64  4: 0.59 | 0: 0.90  1: 0.82  2: 0.74  3: 0.73  4: 0.70 | 0.84 |
| SVM Rbf | ~0.04s | 0.93 | 0: 0.92  1: 0.82  2: 0.75  3: 0.82  4: 0.78 | 0: 0.92  1: 0.78  2: 0.94  3: 0.72  4: 0.82 | 0: 0.92  1: 0.80  2: 0.83  3: 0.77  4: 0.80 | 0.87 |
| Random Forest | ~1.85s | 0.98 | 0: 0.93  1: 0.85  2: 0.71  3: 0.81  4: 0.94 | 0: 0.94  1: 0.74  2: 0.94  3: 0.84  4: 0.73 | 0: 0.93  1: 0.79  2: 0.81  3: 0.82  4: 0.82 | 0.88 |
| CNN 1 layer | ~18.35s | 0.94 |  | 0: 0.99  1: 0.83  2: 0.79  3: 1.00  4: 1.00 | 0: 0.98  1: 0.94  2: 1.00  3: 1.00  4: 0.83 | 0: 0.98  1: 0.88  2: 0.88  3: 1.00  4: 0.91 | 0.96 |
| CNN 4 layer | ~18.82s | 0.95 |  | 0: 0.99  1: 0.94  2: 0.79  3: 1.00  4: 1.00 | 0: 0.98  1: 0.94  2: 100  3: 100  4: 0.88 | 0: 0.99  1: 0.94  2: 0.88  3: 1.00  4: 0.93 | 0.97 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Top 50 | Total: 1021 records | | | | | | |
|  | Train: 816 records | |  | Test: 205 records | | | |
|  | Total train time | accuracy training | Precision | Recall | F1 score | Accuracy |
| SVM Poly | ~0.06s | 0.93 | 0: 0.90  1: 0.77  2: 0.95  3: 0.71  4: 0.83 | 0: 0.94  1: 0.92  2: 0.82  3: 0.68  4: 0.62 | 0: 0.92  1: 0.84  2: 0.88  3: 0.70  4: 0.71 | 0.86 |
| SVM Rbf | ~0.08s | 0.93 | 0: 0.96  1: 0.77  2: 0.81  3: 0.81  4: 0.77 | 0: 0.89  1: 0.96  2: 1.00  3: 0.77  4: 0.71 | 0: 0.93  1: 0.86  2: 0.90  3: 0.79  4: 0.74 | 0.88 |
| Random Forest | ~2.04s | 0.98 | 0: 0.95  1: 0.86  2: 0.91  3: 0.80  4: 0.94 | 0: 0.96  1: 0.96  2: 0.95  3: 0.91  4: 0.62 | 0: 0.95  1: 0.91  2: 0.93  3: 0.85  4: 0.75 | 0.91 |
| CNN 1 layer | ~35.36s | 0.97 |  | 0: 0.99  1: 0.96  2: 1.00  3: 0.94  4: 1.00 | 0: 0.98  1: 1.00  2: 0.95  3: 1.00  4: 1.00 | 0: 0.99  1: 0.98  2: 0.97  3: 0.97  4: 1.00 | 0.99 |
| CNN 4 layer | ~41.24s | 0.97 |  | 0: 0.99  1: 0.96  2: 1.00  3: 0.94  4: 1.00 | 0: 0.98  1: 1.00  2: 0.95  3: 1.00  4: 1.00 | 0: 0.99  1: 0.98  2: 0.97  3: 0.97  4: 1.00 | 0.99 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Top 100 | Total: 1021 records | | | | | | |
|  | Train: 816 records | |  | Test: 205 records | | | |
|  | Total train time | accuracy training | Precision | Recall | F1 score | Accuracy |
| SVM Poly | ~0.06s | 0.92 | 0: 0.90  1: 0.83  2: 0.95  3: 0.93  4: 0.78 | 0: 0.98  1: 0.79  2: 0.72  3: 0.78  4: 0.74 | 0: 0.94  1: 0.81  2: 0.82  3: 0.85  4: 0.76 | 0.89 |
| SVM Rbf | ~0.06s | 0.92 | 0: 0.97  1: 0.75  2: 0.89  3: 0.82  4: 0.79 | 0: 0.93  1: 0.88  2: 0.96  3: 0.78  4: 0.79 | 0: 0.95  1: 0.81  2: 0.92  3: 0.80  4: 0.79 | 0.90 |
| Random Forest | ~1.85s | 0.99 | 0: 0.95  1: 0.88  2: 1.00  3: 0.76  4: 0.70 | 0: 0.96  1: 0.88  2: 0.80  3: 0.89  4: 0.74 | 0: 0.95  1: 0.88  2: 0.89  3: 0.82  4: 0.72 | 0.90 |
| CNN 1 layer | ~36.22s | 0.97 |  | 0: 0.96  1: 1.00  2: 1.00  3: 0.95  4: 1.00 | 0: 0.99  1: 0.93  2: 0.97  3: 1.00  4: 0.88 | 0: 0.97  1: 0.97  2: 0.98  3: 0.97  4: 0.94 | 0.97 |
| CNN 4 layer | ~38.42s | 0.98 |  | 0: 0.97  1: 0.93  2: 1.00  3: 0.95  4: 1.00 | 0: 0.99  1: 0.93  2: 0.93  3: 1.00  4: 0.92 | 0: 0.98  1: 0.93  2: 0.96  3: 0.97  4: 0.96 | 0.97 |

Icon

Description automatically generatedChart, bar chart

Description automatically generatedCó thể xem sơ đồ sau để có thể nắm được các thông tin chính

Hình : So sánh thời gian thực thi của top 20, 50 và 100 permission

Hình : So sánh độ chính xác của top 20, 50 và 100 permission

Thông qua sơ đồ có thể thấy rằng, trong các mô hình thì CNN cho kết quả tốt hơn, kể cả về độ chính xác mà còn vệ sự chênh lệch giữa khi train và test.  
Bên cạnh đó quan sát từ đồ thị, ta có thể thấy rằng top 20 permission có kết quả về độ chính xác gần như tương đương với top 50 permission, và thậm chí có kết quả tốt hơn top 100 đối với 2 thuật toán SVM và Random Forest, tuy nhiên điểm nổi bật là top 20 có thời gian train thấp hơn so với top 50 và 100, đặc biệt là đối với mô hình CNN, cụ thể khi so sánh với top 100 thì độ chênh lệch là ~72s đối với CNN 1 layer và 44s đối với CNN 4 layer. Khi so sánh giữa top 50 và 100 thì tỉ lệ chính xác khá tương đồng, ngoài độ chênh lệch thời gian là rất lớn.

Từ những thông tin trên ta có thể kết luận rằng việc lựa chọn nhiều permission để xây dựng bộ dataset chưa chắc sẽ cho kết quả tốt hơn, mặc khác nó còn có thể gây ảnh hưởng xấu đến hiệu suất của mô hình, đặc biệt là đối với cnn. Tuy nhiên cần nhấn mạnh việc top 20 permissons dường như có hiệu xuất tốt nhất đối với bộ dataset AndMal2017 không có nghĩa nó cũng như vậy đối với các bộ dataset khác, bởi vì thông qua các biểu đồ từ Figure1-6 có thể thấy mặc dù phần lớn các ứng dụng có số lượng permission dưới 20, tuy nhiên vẫn tồn tại một số lượng tương đối cao các ứng dụng có nhiều hơn 20 permission, bên cạnh đó thông qua figure6 có thể thể tổng số permission khi lấy top 20, vẫn còn khá nhỏ so với số lượng permisions mà 1 ứng dụng có thể có, do đó chọn top 50 permission vẫn là lựa chọn an toàn nhất bởi qua các biểu đồ từ figure1-6 có thể thấy lựa chọn này có thể lấy được số lượng trung bình các permision có thể có của ứng dụng và nó cũng không quá cao như khi chọn top 100 nên sẽ không ảnh hưởng nhiều đến hiệu suất khi train.

# Đánh giá hiệu xuất khi trích xuất top 20, 50 và 100 permission đối với bộ dataset MalDroid2020

## Các thuộc tính xây dựng dataset ở vị trí ngẫu nhiên

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Top 20 | Total: 1021 records | | | | | | |
|  | Train: 816 records | |  | Test: 205 records | | | |
|  | Total train time | accuracy training | Precision | Recall | F1 score | Accuracy |
| SVM Poly | ~0.04s | 0.94 | 0: 0.87  1: 0.69  2: 0.89  3: 0.89  4: 0.92 | 0: 0.96  1: 0.87  2: 0.85  3: 0.73  4: 0.46 | 0: 0.91  1: 0.77  2: 0.87  3: 0.80  4: 0.61 | 0.85 |
| SVM Rbf | ~0.05s | 0.92 | 0: 0.96  1: 0.68  2: 0.83  3: 0.78  4: 0.75 | 0: 0.92  1: 0.91  2: 0.95  3: 0.82  4: 0.50 | 0: 0.94  1: 0.78  2: 0.88  3: 0.80  4: 0.60 | 0.92 |
| Random Forest | ~2.04s | 0.98 | 0: 0.95  1: 0.71  2: 0.83  3: 0.86  4: 1.00 | 0: 0.97  1: 0.87  2: 0.95  3: 0.82  4: 0.62 | 0: 0.96  1: 0.78  2: 0.88  3: 0.84  4: 0.77 | 0.90 |
| CNN 1 layer | ~17.43s | 0.93 |  | 0: 0.96  1: 1.00  2: 0.89  3: 1.00  4: 1.00 | 0: 0.99  1: 0.96  2: 1.00  3: 0.91  4: 0.83 | 0: 0.98  1: 0.98  2: 0.94  3: 0.95  4: 0.91 | 0.97 |
| CNN 4 layer | ~20.77s | 0.96 |  | 0: 1.00  1: 1.00  2: 0.89  3: 0.92  4: 1.00 | 0: 0.98  1: 1.00  2: 1.00  3: 1.00  4: 0.94 | 0: 0.99  1: 1.00  2: 0.94  3: 0.96  4: 0.97 | 0.98 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Top 50 | Total: 1021 records | | | | | | |
|  | Train: 816 records | |  | Test: 205 records | | | |
|  | Total train time | accuracy training | Precision | Recall | F1 score | Accuracy |
| SVM Poly | ~0.04s | 0.94 | 0: 0.81  1: 0.67  2: 1.00  3: 0.94  4: 0.71 | 0: 0.96  1: 0.63  2: 0.63  3: 0.68  4: 0.50 | 0: 0.88  1: 0.65  2: 0.77  3: 0.79  4: 0.59 | 0.81 |
| SVM Rbf | ~0.06s | 0.94 | 0: 0.87  1: 0.71  2: 0.75  3: 0.86  4: 0.75 | 0: 0.92  1: 0.63  2: 0.95  3: 0.76  4: 0.50 | 0: 0.89  1: 0.67  2: 0.84  3: 0.81  4: 0.60 | 0.83 |
| Random Forest | ~2.07s | 0.98 | 0: 0.86  1: 0.91  2: 0.89  3: 0.83  4: 0.83 | 0: 0.97  1: 0.53  2: 0.89  3: 0.80  4: 0.62 | 0: 0.91  1: 0.67  2: 0.89  3: 0.82  4: 0.71 | 0.86 |
| CNN 1 layer | ~45.34s | 0.97 |  | 0: 0.98  1: 1.00  2: 1.00  3: 1.00  4: 1.00 | 0: 1.00  1: 1.00  2: 1.00  3: 1.00  4: 0.92 | 0: 0.99  1: 1.00  2: 1.00  3: 1.00  4: 0.96 | 0.99 |
| CNN 4 layer | ~27.08s | 0.96 |  | 0: 0.98  1: 1.00  2: 1.00  3: 1.00  4: 1.00 | 0: 1.00  1: 1.00  2: 1.00  3: 1.00  4: 0.92 | 0: 0.99  1: 1.00  2: 1.00  3: 1.00  4: 0.96 | 0.99 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Top 100 | Total: 1021 records | | | | | | |
|  | Train: 816 records | |  | Test: 205 records | | | |
|  | Total train time | accuracy training | Precision | Recall | F1 score | Accuracy |
| SVM Poly | ~0.08s | 0.91 | 0: 0.88  1: 0.80  2: 0.87  3: 0.73  4: 0.76 | 0: 0.94  1: 0.77  2: 0.72  3: 0.69  4: 0.65 | 0: 0.91  1: 0.78  2: 0.79  3: 0.71  4: 0.79 | 0.85 |
| SVM Rbf | ~0.09s | 0.93 | 0: 0.93  1: 0.77  2: 0.80  3: 0.87  4: 0.78 | 0: 0.94  1: 0.77  2: 0.89  3: 0.81  4: 0.70 | 0: 0.93  1: 0.77  2: 0.84  3: 0.84  4: 0.74 | 0.88 |
| Random Forest | ~2.31s | 0.98 | 0: 0.92  1: 0.86  2: 0.84  3: 0.88  4: 0.88 | 0: 0.98  1: 0.69  2: 0.89  3: 0.94  4: 0.70 | 0: 0.95  1: 0.77  2: 0.86  3: 0.91  4: 0.78 | 0.90 |
| CNN 1 layer | ~73.35s | 0.97 |  | 0: 1.00  1: 1.00  2: 0.87  3: 1.00  4: 1.00 | 0: 0.98  1: 1.00  2: 1.00  3: 1.00  4: 0.94 | 0: 0.99  1: 1.00  2: 0.93  3: 1.00  4: 0.97 | 0.99 |
| CNN 4 layer | ~63.74s | 0.97 |  | 0: 0.99  1: 1.00  2: 1.00  3: 1.00  4: 0.78 | 0: 0.98  1: 0.94  2: 0.85  3: 1.00  4: 1.00 | 0: 0.99  1: 0.97  2: 0.92  3: 1.00  4: 0.88 | 0.97 |

Bảng : Kết quả khi sử dụng top 20 permission

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Top 20 | Total: 12245 records | | | | | | |
|  | Train: 816 records | |  | Test: 205 records | | | |
|  | Total train time | accuracy training | Precision | Recall | F1 score | Accuracy |
| SVM Poly | ~1.53s | 0.93 | 0: 0.70  1: 0.81  2: 0.88  3: 0.99  4: 0.93 | 0: 0.87  1: 0.96  2: 0.86  3: 0.96  4: 0.86 | 0: 0.78 1: 0.88  2: 0.87  3: 0.97  4: 0.89 | 0.91 |
| SVM Rbf | ~1.7s | 0.94 | 0: 0.80  1: 0.79  2: 0.94  3: 0.98  4: 0.93 | 0: 0.90  1: 0.97  2: 0.87  3: 0.97  4: 0.88 | 0: 0.85  1: 0.87  2: 0.91  3: 0.97  4: 0.91 | 0.92 |
| Random Forest | ~6.84s | 0.98 | 0: 0.86  1: 0.87  2: 0.94  3: 0.99  4: 0.95 | 0: 0.88  1: 0.96  2: 0.92  3: 0.99  4: 0.93 | 0: 0.87  1: 0.91  2: 0.93  3: 0.99  4: 0.94 | 0.95 |
| CNN 1 layer | ~227.15s | 0.96 |  | 0: 0.94  1: 0.95  2: 0.98  3: 0.99  4: 0.98 | 0: 0.98  1: 0.99  2: 0.97  3: 0.99  4: 0.96 | 0: 0.96  1: 0.97  2: 0.97  3: 0.99  4: 0.97 | 0.98 |
| CNN 4 layer | ~238.47s | 0.96 |  | 0: 0.93  1: 0.96  2: 0.99  3: 1.00  4: 0.98 | 0: 0.98  1: 0.99  2: 0.97  3: 0.99  4: 0.97 | 0: 0.95  1: 0.98  2: 0.98  3: 0.99  4: 0.98 | 0.98 |

Bảng : Kết quả khi sử dụng top 50 permissions

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Top 50 | Total: 12245 records | | | | | | |
|  | Train: 816 records | |  | Test: 205 records | | | |
|  | Total train time | accuracy training | Precision | Recall | F1 score | Accuracy |
| SVM Poly | ~2.91s | 0.93 | 0: 0.67  1: 0.82  2: 0.88  3: 0.99  4: 0.94 | 0: 0.89  1: 0.98  2: 0.85  3: 0.94  4: 0.86 | 0: 0.77  1: 0.90  2: 0.86  3: 0.96  4: 0.90 | 0.91 |
| SVM Rbf | ~2.69s | 0.94 | 0: 0.69  1: 0.83  2: 0.95  3: 0.98  4: 0.92 | 0: 0.89  1: 0.98  2: 0.85  3: 0.97  4: 0.88 | 0: 0.78  1: 0.90  2: 0.90  3: 0.98  4: 0.90 | 0.92 |
| Random Forest | ~9.56s | 0.98 | 0: 0.82  1: 0.87  2: 0.94  3: 0.99  4: 0.94 | 0: 0.86  1: 0.97  2: 0.90  3: 0.98  4: 0.93 | 0: 0.84  1: 0.92  2: 0.92  3: 0.99  4: 0.93 | 0.94 |
| CNN 1 layer | ~415.73s | 0.97 |  | 0: 0.99  1: 0.95  2: 0.98  3: 0.99  4: 0.98 | 0: 0.94  1: 0.99  2: 0.98  3: 0.99  4: 0.98 | 0: 0.96  1: 0.97  2: 0.98  3: 0.99  4: 0.98 | 0.98 |
| CNN 4 layer | ~443.53s | 0.98 |  | 0: 0.98  1: 0.97  2: 0.99  3: 0.99  4: 0.98 | 0: 0.94  1: 0.99  2: 0.98  3: 0.99  4: 0.98 | 0: 0.96  1: 0.98  2: 0.98  3: 0.99  4: 0.98 | 0.98 |

Bảng : Kết quả khi sử dụng top 100 permissions

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Top 100 | Total: 12245 records | | | | | | |
|  | Train: 816 records | |  | Test: 205 records | | | |
|  | Total train time | accuracy training | Precision | Recall | F1 score | Accuracy |
| SVM Poly | ~4.17s | 0.92 | 0: 0.63  1: 0.84  2: 0.86  3: 0.99  4: 0.95 | 0: 0.89  1: 0.95  2: 0.86  3: 0.95  4: 0.86 | 0: 0.74  1: 0.89  2: 0.86  3: 0.97  4: 0.90 | 0.91 |
| SVM Rbf | ~1.7s | 0.94 | 0: 0.80  1: 0.83  2: 0.95  3: 0.98  4: 0.95 | 0: 0.90  1: 0.97  2: 0.89  3: 0.99  4: 0.90 | 0: 0.85  1: 0.89  2: 0.92  3: 0.98  4: 0.92 | 0.94 |
| Random Forest | ~10.72s | 0.98 | 0: 0.88  1: 0.87  2: 0.94  3: 0.99  4: 0.95 | 0: 0.87  1: 0.95  2: 0.92  3: 0.99  4: 0.93 | 0: 0.88  1: 0.91  2: 0.93  3: 0.99  4: 0.94 | 0.95 |
| CNN 1 layer | ~899.00s | 0.97 |  | 0: 1.00  1: 0.97  2: 0.98  3: 0.99  4: 0.98 | 0: 092.  1: 0.99  2: 0.98  3: 0.99  4: 0.98 | 0: 0.96  1: 0.98  2: 0.98  3: 0.99  4: 0.98 | 0.98 |
| CNN 4 layer | ~889.85s | 0.98 |  | 0: 1.00  1: 0.98  2: 0.98  3: 0.99  4: 0.97 | 0: 0.93  1: 0.99  2: 0.99  3: 0.99  4: 0.98 | 0: 0.96  1: 0.99  2: 0.99  3: 0.99  4: 0.98 | 0.99 |

Bảng : Kết quả khi sử dụng top 20 permissions

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Top 20 | Total: 12245 records | | | | | | |
|  | Train: 816 records | |  | Test: 205 records | | | |
|  | Total train time | accuracy training | Precision | Recall | F1 score | Accuracy |
| SVM Poly | ~1.56s | 0.93 | 0: 0.65  1: 0.82  2: 0.90  3: 0.99  4: 0.94 | 0: 0.90  1: 0.96  2: 0.85  3: 0.97  4: 0.86 | 0: 0.76  1: 0.88  2: 0.87  3: 0.98  4: 0.89 | 0.92 |
| SVM Rbf | ~1.74s | 0.94 | 0: 0.73  1: 0.80  2: 0.94  3: 0.99  4: 0.93 | 0: 0.92  1: 0.94  2: 0.88  3: 0.98  4: 0.87 | 0: 0.81  1: 0.87  2: 0.91  3: 0.98  4: 0.90 | 0.93 |
| Random Forest | ~6.58s | 0.98 | 0: 0.82  1: 0.87  2: 0.94  3: 1.00  4: 0.93 | 0: 0.90  1: 0.94  2: 0.90  3: 0.99  4: 0.91 | 0: 0.85  1: 0.90  2: 0.92  3: 0.99  4: 0.92 | 0.94 |
| CNN 1 layer | ~223.88s | 0.96 |  | 0: 0.95  1: 0.95  2: 0.98  3: 0.99  4: 0.98 | 0: 0.94  1: 0.99  2: 0.97  3: 0.99  4: 0.97 | 0: 0.94  1: 0.97  2: 0.97  3: 0.99  4: 0.97 | 0.98 |
| CNN 4 layer | ~316.42s | 0.96 |  | 0: 0.98  1: 0.95  2: 0.98  3: 1.00  4: 0.96 | 0: 0.92  1: 0.99  2: 0.97  3: 0.99  4: 0.98 | 0: 0.95  1: 0.97  2: 0.97  3: 0.99  4: 0.97 | 0.98 |

Bảng : Kết quả khi sử dụng top 50 permissions

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Top 50 | Total: 12245 records | | | | | | |
|  | Train: 816 records | |  | Test: 205 records | | | |
|  | Total train time | accuracy training | Precision | Recall | F1 score | Accuracy |
| SVM Poly | ~2.75s | 0.92 | 0: 0.67  1: 0.82  2: 0.88  3: 0.99  4: 0.94 | 0: 0.89  1: 0.99  2: 0.84  3: 0.96  4: 0.84 | 0: 0.76  1: 0.90  2: 0.86  3: 0.98  4: 0.89 | 0.91 |
| SVM Rbf | ~2.34s | 0.95 | 0: 0.76  1: 0.83  2: 0.95  3: 0.98  4: 0.94 | 0: 0.88  1: 0.99  2: 0.89  3: 0.98  4: 0.87 | 0: 0.81  1: 0.90  2: 0.92  3: 0.98  4: 0.91 | 0.93 |
| Random Forest | ~8.93s | 0.98 | 0: 0.84  1: 0.88  2: 0.94  3: 0.99  4: 0.95 | 0: 0.84  1: 0.97  2: 0.91  3: 0.99  4: 0.93 | 0: 0.84  1: 0.92  2: 0.93  3: 0.99  4: 0.94 | 0.95 |
| CNN 1 layer | ~289.85s | 0.97 |  | 0: 0.98  1: 0.96  2: 0.98  3: 0.99  4: 0.98 | 0: 0.94  1: 1.00  2: 0.96  3: 0.99  4: 0.98 | 0: 0.96  1: 0.98  2: 0.97  3: 0.99  4: 0.98 | 0.98 |
| CNN 4 layer | ~498.56s | 0.99 |  | 0: 0.99  1: 0.97  2: 0.98  3: 0.99  4: 0.98 | 0: 0.94  1: 0.99  2: 0.97  3: 0.99  4: 0.98 | 0: 0.97  1: 0.98  2: 0.97  3: 0.99  4: 0.98 | 0.98 |

Bảng 9: Kết quả khi sử dụng top 100 permissions

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Top 100 | Total: 12245 records | | | | | | |
|  | Train: 816 records | |  | Test: 205 records | | | |
|  | Total train time | accuracy training | Precision | Recall | F1 score | Accuracy |
| SVM Poly | ~4.15s | 0.92 | 0: 0.64  1: 0.82  2: 0.87  3: 0.99  4: 0.94 | 0: 0.84  1: 0.97  2: 0.84  3: 0.95  4: 0.87 | 0: 0.73  1: 0.89  2: 0.86  3: 0.97  4: 0.90 | 0.91 |
| SVM Rbf | ~3.20s | 0.94 | 0: 0.69  1: 0.83  2: 0.94  3: 0.98  4: 0.93 | 0: 0.87  1: 0.96  2: 0.84  3: 0.98  4: 0.90 | 0: 0.77  1: 0.89  2: 0.89  3: 0.98  4: 0.92 | 0.92 |
| Random Forest | ~11.36s | 0.98 | 0: 0.88  1: 0.89  2: 0.94  3: 1.00  4: 0.95 | 0: 0.87  1: 0.97  2: 0.92  3: 0.99  4: 0.94 | 0: 0.88  1: 0.93  2: 0.93  3: 0.99  4: 0.94 | 0.95 |
| CNN 1 layer | ~716.95s | 0.97 |  | 0: 0.97  1: 0.96  2: 0.99  3: 0.99  4: 0.99 | 0: 0.96  1: 1.00  2: 0.96  3: 1.00  4: 0.97 | 0: 0.96  1: 0.98  2: 0.98  3: 0.99  4: 0.98 | 0.98 |
| CNN 4 layer | ~671.44s | 0.98 |  | 0: 0.97  1: 0.96  2: 0.98  3: 0.99  4: 0.98 | 0: 0.96  1: 0.98  2: 0.97  3: 1.00  4: 0.97 | 0: 0.96  1: 0.97  2: 0.97  3: 0.99  4: 0.98 | 0.98 |

Icon

Description automatically generatedChart, bar chart

Description automatically generatedCó thể xem sơ đồ sau để có thể nắm được các thông tin chính

Hình : So sánh thời gian thực thi của top 20, 50 và 100 permission

Hình : So sánh độ chính xác của top 20, 50 và 100 permission

Thông qua sơ đồ có thể thấy rằng, trong các mô hình thì CNN cho kết quả tốt hơn, kể cả về độ chính xác mà còn vệ sự chênh lệch giữa khi train và test.  
Bên cạnh đó quan sát từ đồ thị, ta có thể thấy rằng top 20 permission có kết quả về độ chính xác gần như tương đương với top 50 permission, và thậm chí có kết quả tốt hơn top 100 đối với 2 thuật toán SVM và Random Forest, tuy nhiên điểm nổi bật là top 20 có thời gian train thấp hơn so với top 50 và 100, đặc biệt là đối với mô hình CNN, cụ thể khi so sánh với top 100 thì độ chênh lệch là ~72s đối với CNN 1 layer và 44s đối với CNN 4 layer. Khi so sánh giữa top 50 và 100 thì tỉ lệ chính xác khá tương đồng, ngoài độ chênh lệch thời gian là rất lớn.

Từ những thông tin trên ta có thể kết luận rằng việc lựa chọn nhiều permission để xây dựng bộ dataset chưa chắc sẽ cho kết quả tốt hơn, mặc khác nó còn có thể gây ảnh hưởng xấu đến hiệu suất của mô hình, đặc biệt là đối với cnn. Tuy nhiên cần nhấn mạnh việc top 20 permissons dường như có hiệu xuất tốt nhất đối với bộ dataset AndMal2017 không có nghĩa nó cũng như vậy đối với các bộ dataset khác, bởi vì thông qua các biểu đồ từ Figure1-6 có thể thấy mặc dù phần lớn các ứng dụng có số lượng permission dưới 20, tuy nhiên vẫn tồn tại một số lượng tương đối cao các ứng dụng có nhiều hơn 20 permission, bên cạnh đó thông qua figure6 có thể thể tổng số permission khi lấy top 20, vẫn còn khá nhỏ so với số lượng permisions mà 1 ứng dụng có thể có, do đó chọn top 50 permission vẫn là lựa chọn an toàn nhất bởi qua các biểu đồ từ figure1-6 có thể thấy lựa chọn này có thể lấy được số lượng trung bình các permision có thể có của ứng dụng và nó cũng không quá cao như khi chọn top 100 nên sẽ không ảnh hưởng nhiều đến hiệu suất khi train.

## Các thuộc tính xây dựng dataset được giữ theo thứ tự giảm dần tần suất xuất hiện

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Top 20 | Total: 1021 records | | | | | | |
|  | Train: 816 records | |  | Test: 205 records | | | |
|  | Total train time | accuracy training | Precision | Recall | F1 score | Accuracy |
| SVM Poly | ~0.03s | 0.94 | 0: 0.87  1: 0.86  2: 0.64  3: 0.84  4: 0.87 | 0: 0.93  1: 0.78  2: 0.88  3: 0.64  4: 0.59 | 0: 0.90  1: 0.82  2: 0.74  3: 0.73  4: 0.70 | 0.84 |
| SVM Rbf | ~0.04s | 0.93 | 0: 0.92  1: 0.82  2: 0.75  3: 0.82  4: 0.78 | 0: 0.92  1: 0.78  2: 0.94  3: 0.72  4: 0.82 | 0: 0.92  1: 0.80  2: 0.83  3: 0.77  4: 0.80 | 0.87 |
| Random Forest | ~1.85s | 0.98 | 0: 0.93  1: 0.85  2: 0.71  3: 0.81  4: 0.94 | 0: 0.94  1: 0.74  2: 0.94  3: 0.84  4: 0.73 | 0: 0.93  1: 0.79  2: 0.81  3: 0.82  4: 0.82 | 0.88 |
| CNN 1 layer | ~18.35s | 0.94 |  | 0: 0.99  1: 0.83  2: 0.79  3: 1.00  4: 1.00 | 0: 0.98  1: 0.94  2: 1.00  3: 1.00  4: 0.83 | 0: 0.98  1: 0.88  2: 0.88  3: 1.00  4: 0.91 | 0.96 |
| CNN 4 layer | ~18.82s | 0.95 |  | 0: 0.99  1: 0.94  2: 0.79  3: 1.00  4: 1.00 | 0: 0.98  1: 0.94  2: 100  3: 100  4: 0.88 | 0: 0.99  1: 0.94  2: 0.88  3: 1.00  4: 0.93 | 0.97 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Top 50 | Total: 1021 records | | | | | | |
|  | Train: 816 records | |  | Test: 205 records | | | |
|  | Total train time | accuracy training | Precision | Recall | F1 score | Accuracy |
| SVM Poly | ~0.06s | 0.93 | 0: 0.90  1: 0.77  2: 0.95  3: 0.71  4: 0.83 | 0: 0.94  1: 0.92  2: 0.82  3: 0.68  4: 0.62 | 0: 0.92  1: 0.84  2: 0.88  3: 0.70  4: 0.71 | 0.86 |
| SVM Rbf | ~0.08s | 0.93 | 0: 0.96  1: 0.77  2: 0.81  3: 0.81  4: 0.77 | 0: 0.89  1: 0.96  2: 1.00  3: 0.77  4: 0.71 | 0: 0.93  1: 0.86  2: 0.90  3: 0.79  4: 0.74 | 0.88 |
| Random Forest | ~2.04s | 0.98 | 0: 0.95  1: 0.86  2: 0.91  3: 0.80  4: 0.94 | 0: 0.96  1: 0.96  2: 0.95  3: 0.91  4: 0.62 | 0: 0.95  1: 0.91  2: 0.93  3: 0.85  4: 0.75 | 0.91 |
| CNN 1 layer | ~35.36s | 0.97 |  | 0: 0.99  1: 0.96  2: 1.00  3: 0.94  4: 1.00 | 0: 0.98  1: 1.00  2: 0.95  3: 1.00  4: 1.00 | 0: 0.99  1: 0.98  2: 0.97  3: 0.97  4: 1.00 | 0.99 |
| CNN 4 layer | ~41.24s | 0.97 |  | 0: 0.99  1: 0.96  2: 1.00  3: 0.94  4: 1.00 | 0: 0.98  1: 1.00  2: 0.95  3: 1.00  4: 1.00 | 0: 0.99  1: 0.98  2: 0.97  3: 0.97  4: 1.00 | 0.99 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Top 100 | Total: 1021 records | | | | | | |
|  | Train: 816 records | |  | Test: 205 records | | | |
|  | Total train time | accuracy training | Precision | Recall | F1 score | Accuracy |
| SVM Poly | ~0.06s | 0.92 | 0: 0.90  1: 0.83  2: 0.95  3: 0.93  4: 0.78 | 0: 0.98  1: 0.79  2: 0.72  3: 0.78  4: 0.74 | 0: 0.94  1: 0.81  2: 0.82  3: 0.85  4: 0.76 | 0.89 |
| SVM Rbf | ~0.06s | 0.92 | 0: 0.97  1: 0.75  2: 0.89  3: 0.82  4: 0.79 | 0: 0.93  1: 0.88  2: 0.96  3: 0.78  4: 0.79 | 0: 0.95  1: 0.81  2: 0.92  3: 0.80  4: 0.79 | 0.90 |
| Random Forest | ~1.85s | 0.99 | 0: 0.95  1: 0.88  2: 1.00  3: 0.76  4: 0.70 | 0: 0.96  1: 0.88  2: 0.80  3: 0.89  4: 0.74 | 0: 0.95  1: 0.88  2: 0.89  3: 0.82  4: 0.72 | 0.90 |
| CNN 1 layer | ~36.22s | 0.97 |  | 0: 0.96  1: 1.00  2: 1.00  3: 0.95  4: 1.00 | 0: 0.99  1: 0.93  2: 0.97  3: 1.00  4: 0.88 | 0: 0.97  1: 0.97  2: 0.98  3: 0.97  4: 0.94 | 0.97 |
| CNN 4 layer | ~38.42s | 0.98 |  | 0: 0.97  1: 0.93  2: 1.00  3: 0.95  4: 1.00 | 0: 0.99  1: 0.93  2: 0.93  3: 1.00  4: 0.92 | 0: 0.98  1: 0.93  2: 0.96  3: 0.97  4: 0.96 | 0.97 |

Icon

Description automatically generatedChart, bar chart

Description automatically generatedCó thể xem sơ đồ sau để có thể nắm được các thông tin chính

Hình 20: So sánh thời gian thực thi của top 20, 50 và 100 permission

Hình 21: So sánh độ chính xác của top 20, 50 và 100 permission

Thông qua sơ đồ có thể thấy rằng, trong các mô hình thì CNN cho kết quả tốt hơn, kể cả về độ chính xác mà còn vệ sự chênh lệch giữa khi train và test.  
Bên cạnh đó quan sát từ đồ thị, ta có thể thấy rằng top 20 permission có kết quả về độ chính xác gần như tương đương với top 50 permission, và thậm chí có kết quả tốt hơn top 100 đối với 2 thuật toán SVM và Random Forest, tuy nhiên điểm nổi bật là top 20 có thời gian train thấp hơn so với top 50 và 100, đặc biệt là đối với mô hình CNN, cụ thể khi so sánh với top 100 thì độ chênh lệch là ~72s đối với CNN 1 layer và 44s đối với CNN 4 layer. Khi so sánh giữa top 50 và 100 thì tỉ lệ chính xác khá tương đồng, ngoài độ chênh lệch thời gian là rất lớn.

Từ những thông tin trên ta có thể kết luận rằng việc lựa chọn nhiều permission để xây dựng bộ dataset chưa chắc sẽ cho kết quả tốt hơn, mặc khác nó còn có thể gây ảnh hưởng xấu đến hiệu suất của mô hình, đặc biệt là đối với cnn. Tuy nhiên cần nhấn mạnh việc top 20 permissons dường như có hiệu xuất tốt nhất đối với bộ dataset AndMal2017 không có nghĩa nó cũng như vậy đối với các bộ dataset khác, bởi vì thông qua các biểu đồ từ Figure1-6 có thể thấy mặc dù phần lớn các ứng dụng có số lượng permission dưới 20, tuy nhiên vẫn tồn tại một số lượng tương đối cao các ứng dụng có nhiều hơn 20 permission, bên cạnh đó thông qua figure6 có thể thể tổng số permission khi lấy top 20, vẫn còn khá nhỏ so với số lượng permisions mà 1 ứng dụng có thể có, do đó chọn top 50 permission vẫn là lựa chọn an toàn nhất bởi qua các biểu đồ từ figure1-6 có thể thấy lựa chọn này có thể lấy được số lượng trung bình các permision có thể có của ứng dụng và nó cũng không quá cao như khi chọn top 100 nên sẽ không ảnh hưởng nhiều đến hiệu suất khi train.

# Đánh giá model khi trích xuất top 50 permision

Để đánh giá model nhóm sẽ sử dụng 2 bộ dataset đó là CICDataset 2018

## CICDataset: AndMal2017

Đối với bộ dataset này ta có môt số thông tin như sau: Adware, Ransomeware, Scareware, SMS Malware, Benign 2017. Tổng cộng số record có được sau khi trích xuất bao gồm 1021 record. Trong 80% dùng cho traini và validate, 20% còn lại dùng để test.

Kết quả của các thuật toán khi sử dụng dụng bộ dataset AndMal2017 như sau (sử dụng lại kết quả ở Bảng 2):

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Total: 1021 records | | | | | | |
|  | Train: 816 records | |  | Test: 205 records | | | |
|  | Total train time | accuracy training | Precision | Recall | F1 score | Accuracy |
| SVM Poly | ~0.05s | 0.92 | 0: 0.84 1: 0.91 | 0: 0.95 1: 0.75 | 0: 0.89 1: 0.83 | 0.87 |
| SVM Rbf | ~0.06s | 0.95 | 0: 0.93 1: 0.93 | 0: 0.95 1: 0.91 | 0: 0.94 1: 0.92 | 0.93 |
| RandomForest | ~1.92s | 0.98 | 0: 0.97 1: 0.92 | 0: 0.94 1: 0.95 | 0: 0.95 1: 0.94 | 0.95 |
| CNN 1 layer | ~15.56s | 0.96 |  | 0: 0.98 1: 0.94 | 0: 0.96 1: 0.98 | 0: 0.97 1: 0.96 | 0.97 |
| CNN 4 layer | ~26.15s | 0.96 |  | 0: 0.98 1: 0.95 | 0: 0.97 1: 0.96 | 0: 0.97 1: 0.96 | 0.97 |

Chỉ cần dụa vào thông sô Accuracy ta có thể thấy tỉ lệ chính xác khi test tăng dần từ SVM Poly < SVM Rbf < RandomForest < CNN 1 layer ~ CNN 4 layer.

Nhận xét về các thuật toán đã sử dụng thì CNN mất nhiều thời gian train nhất, có thể nói khi sử dụng 4 layer thì cũng không cải thiện nhiều so với khi sử dụng 1 layer, tuy nhiên thời gian train lại có cách biệt khá lớn, nên có tể đánh giá CNN sử dụng 1 layer hiệu quả hơn.

Khi đối chiếu giá trị accuracy giữa khi train và test, có thể thấy độ chênh lệnh không quá lớn nên ta có thể đánh giá các model không bị overfitting và từ đó có thể nói phương pháp phát hiện mã độc dựa vào cách trích xuất permission đã hoạt động hiệu quả.

## MalDroid-2020

Bộ dataset này có một chút khác biệt so với AndMal2017, bao gồm một sô loại app như sau: Adware, Banking, Benign, Riskware và SMSmalware. Tuy nhiên do Benign của MalDroid-2020 có kích thước quá lơn(~41GB) nên nhóm đã sử dụng Benign-2016 thay vào đó.

Số lượng app của từng loại như sau: Benign – 600 apps, Adware – 1514 apps, Riskware – 2810 app, SMSmalware – 4821 app, Banking – 2505 apps.

Bảng : Kết quả khi sử dụng top 20 permission và MalDroid-2020 dataset

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Total: 12245 records | | | | | | |
|  | Train: 9796 records | |  | Test: 2449 records | | | |
|  | Total train time | accuracy training | Precision | Recall | F1 score | Accuracy |
| SVM Poly | ~2.74s | 0.96 | 0: 0.51 1: 1.00 | 0: 0.95 1: 0.96 | 0: 0.67 1: 0.98 | 0.96 |
| SVM Rbf | ~2.60s | 0.98 | 0: 0.67 1: 1.00 | 0: 0.94 1: 0.98 | 0: 0.78 1: 0.99 | 0.98 |
| RandomForest | ~6.88s | 0.99 | 0: 0.86 1: 0.99 | 0: 0.89 1: 0.99 | 0: 0.87 1: 0.99 | 0.99 |
| CNN 1 layer | ~501.74s | 0.9967 |  | 0: 0.99 1: 1.00 | 0: 0.94 1: 1.00 | 0: 0.96 1: 1.00 | 1.00 |
| CNN 4 layer | ~581.07s | 0.9959 |  | 0: 0.93 1: 1.00 | 0: 0.98 1: 1.00 | 0: 0.96 1: 1.00 | 1.00 |

Hai thuật toán SVM và RandomForest có tỉ lệ precision đối với label 0 có chênh lệch khá lớn so với label 1, điều này có thể là do số lượng record của label 1(malware) lớn hơn nhiều so với label 0(benign), tuy nhiên độ chính xác của các thuật toán vẫn rất cao.

## Kết luận

Từ kết quả train và test của 2 bộ dataset, có thể đánh giá rằng phương pháp trích xuất permission hoạt động khá tốt khi không dẫn đến các vấn đề như over-fitting hay under-fitting. Có thể thấy khả năng dự đoán của các model hoạt động tương đối ổn định và cho kết quả tốt ngay cả khi chỉ sử dụng một lượng nhỏ các mẫu để train (chỉ 1021 record đối với AndMal-2017). Tuy nhiên để đạt được kết quả tốt hơn cần sử dụng kết hợp các phương pháp như phân tích tĩnh (Davik bytecode), phân tích các API,… hoặc có thể sử dụng phương pháp GAN để phát sinh thêm số lượng mẫu thử. Sử dụng GAN là định hướng phát triển tiếp theo của nhóm và sẽ cập nhật khi đạt được những kết quả tốt hơn.

# Thông tin các file Colab đã thực hiện

Link Gdrive: <https://drive.google.com/drive/folders/17EzPsK5GurajkSb2t_MAGljPKcBQcS6p?usp=sharing>

Nội dung bao gồm các file:

* malware\_classification - use top 20 - AndMal2017.ipynb (1)
* malware\_classification – use top 50 – AndMal2017.ipynb (2)
* malware\_classification - use top 100 - AndMal2017.ipynb (3)
* malware\_classification - use top 50 - MalDroid\_2020.ipynb (4)
* Demo - detect apps.ipynb (5)

Trong đó các file từ 1->3: được thực hiện với mục đích đánh giá nên lựa chọn sử dụng bao nhiêu permission, file 2 và 4 được sử dụng để đánh giá mức độ hiệu quả của thuật toán khi sử dụng top 50 permission, file 5 thực hiện demo về phát hiện ứng dụng độc hại bằng cách sử dụng các model đã train được ở file thứ 2.

# References

|  |  |
| --- | --- |
| [1] | len008. [Trực tuyến]. Available: https://github.com/leen8908/Android\_Malware\_Classification. [Đã truy cập 1 12 2021]. |