**Trường Đại Học Xây Dựng Hà Nội**

**Khoa Công nghệ Thông tin**

**--o0o--**

Icon

Description automatically generated

**Báo cáo BTL môn học:**

**HỌC MÁY**

***Tên đề tài: Tiền xử lý, triển khai mô hình và đánh giá kết quả theo bộ dữ liệu TUANDROMD – Phân tích mã độc trên ứng dụng Android***

**Lớp: 66PM3**

**Nhóm thực hiện: Nhóm 8**

**Các thành viên:**

**1) Vũ Hoàng Giang - 0188166**

**2) Huỳnh Minh Hiếu - 0189466**

**3) Nguyễn Đình Tùng – 0208266**

**4) Nguyễn Phúc Hợp – 0191566**

**5) Nguyễn Gia Cường - 0182366**

**Giáo viên hướng dẫn : Thầy Đào Việt Cường**

**Hà Nội, 10/2024**

**MỤC LỤC**

[**MỤC LỤC** 1](#_Toc180359071)

[1. **Giới thiệu** 2](#_Toc180359072)

[2. **Phân tích mã độc** 3](#_Toc180359073)

[**2.1 Phân tích tĩnh** 3](#_Toc180359074)

[**2.2.Phân tích động** 5](#_Toc180359075)

[**3. Chuẩn bị dữ liệu và kỹ thuật tính năng** 7](#_Toc180359076)

[**3.1.Bộ dữ liệu TUANDROMD** 7](#_Toc180359077)

[**3.2.Tiền xử lý dữ liệu** 7](#_Toc180359078)

[**3.3 Lựa chọn tính năng** 8](#_Toc180359079)

[**3.4 Trích xuất vecto đặc trưng** 9](#_Toc180359080)

[**3.5 Tầm quan trọng của việc lựa chọn tính năng** 9](#_Toc180359081)

[**3.6 Quy trình thực hiện** 10](#_Toc180359082)

[**4. Đo lường hiệu suất** 19](#_Toc180359083)

[**5. Triển khai và đánh giá kết quả** 19](#_Toc180359084)

[**5.1 Phân tích hiệu suất của các thuật toán được đề xuất** 19](#_Toc180359085)

[**5.1.1 Thuật toán Support Vector Classification (SVC)** 19](#_Toc180359086)

[**5.1.2 Thuật toán K-Nearest Neighbors (KNN)**  27](#_Toc180359087)

[**5.1.3 Thuật toán Random Forest** 35](#_Toc180359088)

[**5.2 So sánh phân tích độ chính xác của thuật toán dựa trên ML được đề xuất** 43](#_Toc180359089)

[**6. Thảo luận** 44](#_Toc180359090)

# **Giới thiệu**

Ứng dụng di động đã trở thành công cụ thiết yếu cho nhiều chức năng của điện thoại thông minh như thanh toán, mua vé và chụp ảnh. Tính đến nay, có hơn năm triệu ứng dụng có thể truy cập trên cả nền tảng Android và iPhone. Trong số đó, Android là hệ điều hành (OS) chiếm ưu thế, chiếm khoảng 74% thị phần toàn cầu. Điều này đưa Android trở thành hệ điều hành được sử dụng rộng rãi nhất cho các thiết bị di động trên toàn thế giới. Ứng dụng di động có thể sử dụng nhiều cảm biến khác nhau trên thiết bị, chẳng hạn như GPS, camera và micrô, cũng như truy cập dữ liệu cá nhân được lưu trữ trên thiết bị, chẳng hạn như hình ảnh, tin nhắn và danh bạ. Tuy nhiên, truy cập trái phép vào dữ liệu của người dùng cuối là một vấn đề phổ biến, như nhiều nghiên cứu đã chỉ ra. Ngoài ra, truy cập trái phép vào dữ liệu của người dùng cuối là một vấn đề dai dẳng. Ngay cả các ứng dụng độc hại đã bị xóa khỏi Cửa hàng Google Play cũng có thể vượt qua các biện pháp bảo mật và xuất hiện trở lại, khiến thông tin cá nhân của người dùng gặp rủi ro. Điều cần thiết là phải phân biệt giữa các ứng dụng hợp pháp và độc hại để bảo vệ dữ liệu của người dùng cuối. Cửa hàng Google Play có thể vượt qua các biện pháp bảo mật và xuất hiện trở lại, khiến thông tin cá nhân của người dùng gặp rủi ro. Điều cần thiết là phải phân biệt giữa ứng dụng hợp pháp và ứng dụng độc hại để bảo vệ dữ liệu của người dùng cuối.

Cộng đồng nghiên cứu đã đề xuất nhiều giải pháp khác nhau để giảm thiểu rủi ro liên quan đến ứng dụng độc hại. Hai kỹ thuật được sử dụng rộng rãi đó là phân tích tĩnh và phân tích động. Phân tích tĩnh bao gồm việc kiểm tra kỹ lưỡng mã nguồn và tệp nhị phân để xác định các mẫu bất thường. Phân tích động đòi hỏi phải chạy phần mềm trong môi trường được kiểm soát và theo dõi hành vi và hành động của phần mềm. Các thuật toán học máy (ML) cung cấp phương pháp tiếp cận chủ động và hiệu quả hơn để phát hiện và chống lại phần mềm độc hại Android vì chúng liên tục học hỏi và thích ứng với các mối đe dọa mới và đang phát triển. Các thuật toán này có thể nhanh chóng phát hiện và phân loại phần mềm độc hại bằng cách xác định các mẫu và bất thường trong các tập dữ liệu lớn như lệnh gọi hệ thống, lưu lượng mạng, quyền API, ý định và hành vi của người dùng. Các mô hình dựa trên ML cải thiện độ chính xác theo thời gian, sử dụng các kỹ thuật học có giám sát và không giám sát, giúp chúng hiệu quả hơn trong việc xác định các loại phần mềm độc hại mới và chưa từng biết đến trước đây. Phương pháp tiếp cận chủ động này cho phép phát hiện sớm và phản hồi nhanh hơn đối với các mối đe dọa mới, cung cấp khả năng bảo vệ tốt hơn chống lại phần mềm độc hại Android. Ngoài ra, phương pháp tiếp cận dựa trên ML có thể tự động hóa quy trình phát hiện và loại bỏ phần mềm độc hại, giảm gánh nặng cho các chuyên gia bảo mật và tăng hiệu quả của toàn bộ hệ thống bảo mật. Mục tiêu chính của nghiên cứu này là thiết lập một khuôn khổ mạnh mẽ sử dụng các thuật toán dựa trên ML, cụ thể là Support Vector Machine (SVC), Random Forest (RF) và K-Nearest Neighbors (KNN), để xác định và phân loại phần mềm độc hại Android. Để thực hiện điều này, nhóm đã thu thập tập dữ liệu riêng biệt chứa các ứng dụng Android lành tính và độc hại trên TUANDROMD, để đánh giá thuật toán tốt nhất để phát hiện phần mềm độc hại Android một cách chính xác. Các tập dữ liệu này bao gồm nhiều loại và họ phần mềm độc hại khác nhau, cho phép nhóm đào tạo và thử nghiệm các mô hình đề xuất của họ trên một mẫu đại diện cho bối cảnh phần mềm độc hại Android. Nghiên cứu cũng triển khai một số kỹ thuật lựa chọn tính năng, chẳng hạn như lựa chọn tính năng Gain ratio và các bài kiểm tra chi bình phương, để xác định các tính năng có liên quan nhất để phát hiện phần mềm độc hại. Các tính năng này đại diện cho các đặc điểm của ứng dụng như quyền, lệnh gọi API và ý định. Sau đó, các thuật toán đề xuất được đào tạo và thử nghiệm bằng các tính năng đã chọn và được đánh giá bằng các số liệu nổi tiếng như độ chính xác, độ chính xác, khả năng thu hồi, điểm F1 và AUC. Kết quả của các thí nghiệm sẽ cung cấp những hiểu biết có giá trị về hiệu suất của các thuật toán này để phát hiện phần mềm độc hại trên Android, góp phần vào việc phát triển các giải pháp bảo mật hiệu quả hơn cho nền tảng Android. Hơn nữa, nghiên cứu này sẽ đóng vai trò là chuẩn mực cho các nghiên cứu trong tương lai trong lĩnh vực này, cung cấp cơ sở để so sánh và cải tiến hơn nữa

1. **Phân tích mã độc**

Việc sử dụng các kỹ thuật dựa trên ML để phát hiện phần mềm độc hại Android, tập trung vào các phương pháp phân tích tĩnh, động và lai. Mỗi phương pháp đều có những hạn chế riêng, với phân tích tĩnh ít tốn tài nguyên hơn nhưng dễ bị che giấu hơn, phân tích động tốn nhiều thời gian hơn nhưng chính xác hơn và phân tích kết hợp cả hai phương pháp để phân tích toàn diện. Hiểu được những hạn chế này là rất quan trọng để cải thiện hiệu quả phát hiện phần mềm độc hại dựa trên ML.

## **2.1 Phân tích tĩnh**

**2.1.1 Khái niệm phân tích tĩnh**

Phân tích tĩnh là quá trình kiểm tra mã nguồn hoặc mã nhị phân của một chương trình để tìm các dấu hiệu mà không cần chạy chương trình, giúp giảm thiểu nguy cơ lây nhiễm cho hệ thống thực tế. Phân tích tĩnh dựa trên các quy tắc xác định trước hoặc các mẫu đã biết (signature) của mã độc.

**2.1.2 Các bước chính trong phân tích tĩnh**

* Disassembly (Giải mã):
  + Giải mã là quá trình chuyển đổi mã nhị phân của chương trình (file thực thi) thành mã Assembly có thể đọc được. Các công cụ như IDA Pro hoặc Radare2 thường được sử dụng để thực hiện disassembly.
  + Mục đích của bước này là để kiểm tra các lệnh của chương trình và phân tích cách thức hoạt động của mã.
* Decompilation (Biên dịch ngược):
  + Biên dịch ngược là quá trình chuyển mã nhị phân về mã nguồn cấp cao hơn (C/C++ hoặc Java) để dễ dàng đọc và hiểu logic  của chương trình.
  + Công cụ: JD-GUI (Java), Ghidra (đa nền tảng).
  + Phương pháp này giúp phân tích mã độc dễ dàng hơn, vì mã nguồn cấp cao dễ hiểu hơn so với Assembly, nhưng quá trình  này có thể khống chính xác hoàn toàn và phụ thuộc vào kỹ thuật bảo vệ của mã độc
* Phân tích chữ ký (Signature Analysis):
  + Phân tích tĩnh thường được sử dụng cơ sở dữ liệu chứa các chữ ký (signature) của các mẫu mã độc đã biết. Signature là các chuỗi mã, lệnh hoặc hành vi cụ thể được phát hiện trong các mã độc đã từng bị phát hiện trước đây.
  + Nếu chương trình chứa một mẫu giống hoặc tương tự với chữ ký mã độc, nó sẽ bị phát hiện và được coi là nguy hiểm.
* Trích xuất và phân tích đặc trưng:
  + Strings: Tìm kiếm các chuỗi ký tự ASCII hoặc Unicode trong mã thực thi. Các chuỗi này có thể chứa thông tin nhạy cảm như địa chỉ URL, đường dẫn tệp, hoặc các thông điệp mà mã độc sử dụng.
  + API calls: Mã độc thường sử dụng các lời gọi API hệ thống (như đọc/ghi file, kết nối mạng, truy xuất registy) để thực hiện hành vi độc hại. Việc xác định và phân tích các API mà chương trình gọi giúp phát hiện các hành vi đáng ngờ.
  + Header file: Các tệp thực thi (như tệp PE trên Windows) có các trường tiêu đề chứa thông tin quan trọng như loại tệp, phiên bản, điểm vào (entry point), thư viện liên kết động. Phân tích các thông tin này có thể giúp xác định mã độc.
* Cross-reference Analysis:
  + Các công cụ phân tích tĩnh cho phép người dùng theo dõi các tham chiếu giữa các lệnh hoặc các hàm trong mã. Điều này giúp tìm hiểu cách mà một hàm có thể gọi từ đâu, cũng như những hành động mà nó thực hiện.

**2.1.3.Ưu điểm của phân tích tĩnh**

* Không cần thực thi mã độc: Không cần chạy mã độc trên hệ thống, giúp đảm bảo an toàn cho hệ thống phân tích. Phương pháp này hữu ích trong môi trường bảo mật cao hoặc khi không thể tạo môi trường cách ly (sandbox).
* Nhanh chóng phát hiện mã độc đã biết: Dựa vào cơ sở dữ liệu chữ ký và các mẫu hành vi đã biết, phân tích tĩnh có thể nhanh chóng xác định các mã độc quen thuộc.
* Dễ dàng phân tích logic mã: Khi mã được biên dịch ngược thành mã nguồn cấp cao, các nhà phân tích có thể hiểu rõ hơn về logic của chương trình, từ đó phát hiện các hành vi đáng ngờ mà không cần phải quan sát hành vi động.

**2.1.4.Hạn chế của phân tích tĩnh**

* Khó phát hiện mã độc mới: Phân tích phụ thuộc nhiều vào chữ ký và các mẫu đã biết. Nếu mã độc sử dụng các kỹ thuật mới chưa được phát hiện trước đó, phương pháp này có thể bỏ sót.
* Obfuscation(Làm rối mã):
  + Nhiều mã độc sử dụng kỹ thuật obfuscation (làm rối mã) để ẩn đi mục đích và hành vi thực sự của chúng, làm cho việc phân tích trở nên khó khắn. Chúng có thể mã hóa hoặc che giấu các đoạn mã quan trọng.
  + Ví dụ: Packers hoặc Crypters là các công cụ thường được sử dụng để nén hoặc mã hóa mã thực thi, làm cho quá trình disassembly hoặc decompilation trở nên khó khăn hơn.
* Phân tích tĩnh không phát hiện được hành vi động
  + Một số hành vi độc hại chỉ xuất hiện khi mã được thực thi, như khai thác tài nguyên hệ thống, kết nối mạng, hoặc tạo ra các tệp mới. Phân tích tĩnh không thể theo dõi những hoạt động này.
  + Kích thước và độ phức tạp của mã:
  + Các chương trình lớn hoặc phức tạp có thể chứa hàng triệu dòng mã, khiến quá trình phân tích thủ công trở nên cực kỳ khó khăn.

**2.1.5.Công cụ hỗ trợ phân tích tĩnh**

* IDA Pro: Một trong những công cụ disassembly mạnh nhất hiện nay, được sử dụng để phân tích mã Assembly và hỗ trợ nhiều định dạng tệp khác nhau.
* Radare2: Công cụ mã nguồn mở cho phép phân tích tĩnh và động.
* Ghidra: Một bộ công cụ decompilation mã nguồn mở do NSA phát triển, giúp phân tích và biên dịch ngược mã nhị phân.
* Binwalk: Công cụ phân tích firmware hoặc các tệp nhị phân lớn, thường được sử dụng trong phân tích phần cứng.

**2.1.6.Ứng dụng thực tế của phân tích tĩnh**

* Phân tích tĩnh thường được sử dụng trong các giai đoạn đầu của quy trình phát hiện mã độc, nơi mà tốc độ và độ an toàn là ưu tiên hàng đầu. Ngoài ra, nó cũng hữu ích cho:
* Kiểm tra mã nguồn của phần mềm: Phân tích tĩnh có thể được áp dụng để kiểm tra mã nguồn của các phần mềm để đảm bảo không có mã độc hoặc các lỗ hổng bảo mật trong quá trình phát triển phần mềm.
* Phân tích phần mềm của bên thứ ba: Khi nhận phần mềm từ các nguồn bên ngoài, phân tích tĩnh giúp đảm bảo rằng phần mềm không chứa mã độc.

## **2.2.Phân tích động**

**2.2.1. Khái niệm phân tích động**

Phân tích động là quá trình thực thi mã trong một môi trường kiểm soát để quan sát hành vi của mã. Phương pháp này được sử dụng để phát hiện mã độc khi mã nguồn hoặc mã nhị phân không thể cho biết đầy đủ thông tin về hành vi độc hại của chương trình. Phân tích động giúp phát hiện mã độc, đặc biệt khi mã được ngụy trang hoặc mã hóa để tránh phân tích tĩnh.

**2.2.2. Các bước chính trong phân tích động**

* Tạo môi trường sandbox:
  + Sandbox là một môi trường ảo an toàn, nơi các mã độc có thể được thực thi mà không gây ra tác hại cho hệ thống thực.
  + Công cụ sandbox phổ biến: Cuckoo Sandbox, VMware, VirtualBox.
  + Môi trường này được cấu hình sao cho giống hệ thống thực tế nhưng cách ly hoàn toàn khỏi hệ thống chính, nhằm đảm bảo an toàn khi phân tích các phần mềm nguy hiểm.
* Thực thi mã độc:
  + Mã độc được thực thi trong sandbox để quan sát các hành vi như: truy cập tệp, đọc/ghi dữ liệu, kết nối mạng, thay đổi registry, hoặc mở tệp khác.
  + Trong quá trình này, mọi hoạt động đều được ghi lại và phân tích để tìm ra các dấu hiệu đáng ngờ.
* Quan sát và ghi lại hành vi:
  + Hành vi mạng: Mã độc thường tạo các kết nối tới các máy chủ điều khiển (C2 servers), gửi thông tin nhạy cảm, hoặc tải thêm mã độc từ internet. Các công cụ phân tích động sẽ theo dõi các kết nối, tên miền và IP mà mã độc liên lạc.
  + Thay đổi hệ thống: Theo dõi các thay đổi trong hệ thống như việc tạo hoặc sửa đổi tệp, thư mục, hoặc thay đổi cấu hình hệ thống (như registry trong Windows).
  + Sử dụng tài nguyên: Kiểm tra các hành động của mã độc đối với tài nguyên hệ thống như CPU, bộ nhớ, tài nguyên mạng.
  + Chạy các tiến trình con: Một số mã độc có thể tạo ra các tiến trình con (sub-processes) để thực hiện các hoạt động độc hại hoặc phân phối tải trọng độc hại qua các chương trình khác.
* Phân tích hành vi:
  + Hành vi mà mã độc thể hiện trong quá trình thực thi sẽ được phân tích để so sánh với các hành vi điển hình của các mẫu mã độc đã biết.
  + Các hành vi thường bị giám sát bao gồm: kết nối tới máy chủ từ xa, thực hiện thay đổi tệp hệ thống, tiêm mã vào tiến trình khác, sử dụng kỹ thuật chống phân tích (anti-debugging, anti-virtualization).

**2.2.3. Ưu điểm của phân tích động**

* Phát hiện hành vi thực tế:
  + Phân tích động giúp phát hiện các hành vi độc hại thực tế của mã, như lây nhiễm, tấn công hoặc thực hiện các hành vi không mong muốn trên hệ thống.
  + Nó không chỉ dựa vào mã nguồn mà trực tiếp quan sát các tương tác với hệ thống, giúp phát hiện mã độc ngay cả khi mã đã bị obfuscation (mã hóa hoặc làm rối).
* Phát hiện mã độc mới và biến thể:
  + Vì tập trung vào hành vi, phân tích động có thể phát hiện các mẫu mã độc mới (zero-day malware) hoặc các biến thể của mã độc cũ, ngay cả khi không có chữ ký nhận dạng.
  + Điều này đặc biệt hữu ích với các loại mã độc sử dụng polymorphism (thay đổi hình thái) để trốn tránh phân tích tĩnh.
* Phát hiện kỹ thuật lẩn tránh:
  + Nhiều mã độc sử dụng các kỹ thuật để tránh bị phát hiện bởi phân tích tĩnh, nhưng những kỹ thuật này có thể được quan sát qua phân tích động. Ví dụ: mã độc chỉ thực hiện hành vi khi kết nối với internet, hoặc khi hệ thống đáp ứng một điều kiện cụ thể.

**2.2.4. Hạn chế của phân tích động**

* Hiệu suất thấp và tốn thời gian:
  + Việc thực thi mã độc trong môi trường kiểm soát cần thời gian và tài nguyên lớn, đặc biệt khi phân tích các mẫu mã độc phức tạp hoặc khi cần phân tích hàng loạt mẫu.
  + Cần phải theo dõi mã độc trong một thời gian đủ dài để quan sát được tất cả các hành vi, điều này có thể làm giảm hiệu suất.
* Khả năng nhận diện môi trường ảo:
  + Một số mã độc tiên tiến có khả năng phát hiện liệu chúng đang chạy trong môi trường ảo (sandbox), và nếu nhận diện được, chúng sẽ thay đổi hành vi hoặc không thực hiện các hành vi độc hại.
  + Các kỹ thuật như anti-debugging (chống gỡ lỗi) hoặc anti-virtualization (chống ảo hóa) thường được mã độc sử dụng để tránh bị phân tích.
* Không phát hiện được tất cả hành vi:
  + Một số mã độc chỉ kích hoạt hành vi độc hại trong các điều kiện cụ thể, ví dụ: vào một thời điểm nhất định, khi hệ thống đạt ngưỡng tải hoặc khi kết nối với một máy chủ từ xa cụ thể. Nếu các điều kiện này không xảy ra trong môi trường sandbox, hành vi độc hại có thể không được phát hiện.

**2.2.5. Các kỹ thuật trong phân tích động**

* Sandboxing:
  + Sandbox là môi trường ảo cách ly, nơi mã độc được thực thi và quan sát. Hệ thống sandbox ghi lại toàn bộ các hoạt động mà mã độc thực hiện như: thay đổi tệp, kết nối mạng, hành vi tiến trình.
  + Công cụ phổ biến: Cuckoo Sandbox, Any.Run, Joe Sandbox.
* API Monitoring:
  + Mã độc thường thực hiện các hành vi thông qua các lời gọi API của hệ điều hành, như tạo tệp, mở kết nối mạng, hoặc thay đổi registry.
  + Giúp hiểu rõ cách mã độc tương tác với các thành phần của hệ thống.
* Network Traffic Analysis:
  + Một trong những hành vi phổ biến của mã độc là gửi dữ liệu ra ngoài hoặc nhận lệnh từ máy chủ điều khiển (C&C). Phân tích lưu lượng mạng giúp xác định mã độc dựa trên các kết nối đáng ngờ, tên miền độc hại hoặc lưu lượng mạng bất thường.
* Behavioral Profiling:
  + Sau khi quan sát hành vi của mã độc, hệ thống sẽ so sánh với các hồ sơ hành vi đã biết của các loại mã độc phổ biến để phát hiện hành vi tương đồng.
  + Nếu mã độc có các hành vi tương tự với một loại mã độc đã biết, nó có thể được phân loại và phát hiện dựa trên hành vi thay vì chỉ dựa vào mẫu chữ ký (signature).

**2.2.6. Công cụ hỗ trợ phân tích động**

* Cuckoo Sandbox: Một trong những công cụ phân tích động mã nguồn mở phổ biến nhất. Nó có thể tự động phân tích các tệp độc hại và ghi lại các hành vi như thay đổi hệ thống, lưu lượng mạng, và các tiến trình được tạo.
* Any.Run: Dịch vụ phân tích động trực tuyến cho phép người dùng tải lên và thực thi mã độc trong một môi trường cách ly, đồng thời cung cấp các báo cáo chi tiết về hành vi.
* Detux: Công cụ tập trung vào phân tích mã độc Linux.
* Process Monitor (ProcMon): Công cụ của Microsoft giúp theo dõi và ghi lại toàn bộ các tiến trình, lời gọi API, và thay đổi hệ thống mà mã độc thực hiện.

**2.2.7. Ứng dụng thực tế của phân tích động**

* Phát hiện và phân loại mã độc zero-day: Phân tích động rất hữu ích trong việc phát hiện các mã độc mới chưa có trong cơ sở dữ liệu chữ ký của hệ thống phát hiện tĩnh.
* Phân tích hành vi mã độc phức tạp: Các mã độc có kỹ thuật ẩn dấu phức tạp thường không thể bị phát hiện qua phân tích tĩnh, nhưng chúng sẽ lộ diện khi thực thi trong môi trường động.
* Kiểm tra phần mềm của bên thứ ba: Phân tích động giúp đảm bảo rằng phần mềm của bên thứ ba không chứa mã độc hoặc các hành vi

# **3. Chuẩn bị dữ liệu và kỹ thuật tính năng**

## **3.1.Bộ dữ liệu TUANDROMD**

Bộ dữ liệu phần mềm độc hại Android của đại học Tezpur, còn được gọi là TUANDROMD, là một tập hợp các ứng dụng Android thường được sử dụng cho mục đích nghiên cứu trong lĩnh vực an ninh mạng. Các nhà nghiên cứu tại Đại học Tezpur ở Ấn Độ đã tạo ra một bộ dữ liệu có 4465 ứng dụng Android, bao gồm cả ứng dụng độc hại và lành tính. TUANDROMD, một công cụ phát hiện phần mềm độc hại, đã được sử dụng trong các nghiên cứu để đánh giá các kỹ thuật, phân tích các đặc điểm của phần mềm độc hại và hiểu hành vi của nó trên thiết bị Android

## **3.2.Tiền xử lý dữ liệu**

Tiền xử lý dữ liệu là một bước rất quan trọng trong việc giải quyết bất kỳ vấn đề nào trong lĩnh vực Học Máy. Hầu hết các bộ dữ liệu được sử dụng trong các vấn đề liên quan đến Học Máy cần được xử lý, làm sạch và biến đổi trước khi một thuật toán Học Máy có thể được huấn luyện trên những bộ dữ liệu này.

Việc xử lý dữ liệu trên tập dữ liệu phần mềm độc hại TUANDROMD tập trung vào việc rời rạc hóa, các giá trị bị thiếu được quy kết và dữ liệu không cân bằng, loại bỏ các biến chất lượng thấp và sử dụng các quy trình làm sạch dữ liệu để xác định và sửa chữa sự không nhất quán trong tập dữ liệu. Việc xác định và loại bỏ các mẫu trùng lặp khỏi tập dữ liệu là một bước quan trọng trong giai đoạn chuẩn bị dữ liệu, bằng cách sử dụng phương pháp sau. Giả sử P và H lần lượt là tên gói và giá trị băm SHA256 của mỗi mẫu. Tập hợp các mẫu duy nhất có thể thu được bằng cách chọn mẫu s thuộc S sao cho s có sự kết hợp duy nhất của P và H, như minh họa dưới đây:



Xử lý các giá trị đặc trưng bị thiếu bằng cách sử dụng phép quy ước trung bình hoặc trung vị dựa trên phân phối của các giá trị đặc trưng.

A close up of a number

Description automatically generated with medium confidence

Hoặc

A close up of a text

Description automatically generated

Các bước này giúp cải thiện chất lượng và độ tin cậy của tập dữ liệu TUANDROMD để đào tạo các mô hình dựa trên ML.

## **3.3 Lựa chọn tính năng**

Quy trình chọn lựa đặc trưng là một bước quan trọng trong các mô hình phân loại nhằm cải thiện độ chính xác và giảm gánh nặng tính toán. Bước này đặc biệt hữu ích đối với các tập dữ liệu lớn có nhiều đặc trưng và giá trị bị thiếu, giúp tăng hiệu quả và hiệu suất của các mô hình học máy. Nghiên cứu chỉ ra rằng các yếu tố như quyền truy cập, API và luồng dữ liệu đóng vai trò quan trọng trong việc phát hiện hành vi độc hại trên ứng dụng Android

Các quyền truy cập, API và *intent* là các thành phần chính của hệ điều hành Android thường được phân tích. Quyền truy cập giới hạn việc ứng dụng tiếp cận các tài nguyên cần thiết, intent hỗ trợ giao tiếp giữa các thành phần của ứng dụng, và các API cung cấp quyền truy cập vào các chức năng hệ thống và thư viện của bên thứ ba.

Sử dụng phương pháp chọn lựa đặc trưng dựa trên tỷ số Gain-ratio để xác định các đặc trưng mang tính thông tin cao nhất trong tập dữ liệu. Tỷ số Gain-ratio được tính bằng cách chia dữ liệu theo một đặc trưng với lượng thông tin nội tại của đặc trưng đó. Gain đo lường mức độ giảm entropy, trong khi thông tin nội tại đo lường lượng cần thiết để biểu diễn các giá trị của đặc trưng. Giá trị Gain khi chia dữ liệu dựa trên đặc trưng X được cho trong Phương trình



Trong đó Parent là tập dữ liệu gốc, X là tính năng để phân tách, values (X) là tập hợp tất cả các giá trị riêng biệt của X và |Xv| là số trường hợp trong Parent có giá trị v cho tính năng X. Entropy là thước đo độ không tinh khiết của một tập dữ liệu và được tính như sau:

A black and white math equation

Description automatically generated

Trong đó D là một tập dữ liệu, C là tập hợp các nhãn lớp có thể có và Pi là tỷ lệ các trường hợp trong D thuộc lớp i. Thông tin nội tại của một tính năng X được đưa ra trong công thức sau:

A black and white text

Description automatically generated

Tỷ lệ tăng trưởng của tính năng (X) được xác định bởi công thức:

A black text on a white background

Description automatically generated

Phương pháp lựa chọn tính năng Gain-ratio là phương pháp lựa chọn các tính năng có tỷ lệ cao hơn dựa trên thông tin nội tại của chúng, có hiệu quả trong việc xử lý cả kiểu dữ liệu rời rạc và liên tục, đồng thời tránh thiên vị các đặc điểm có nhiều giá trị khác biệt.

## **3.4 Trích xuất vecto đặc trưng**

Kiểm tra Chi-Squared được sử dụng để tính toán ý nghĩa thống kê của mối liên hệ giữa Quyền, lệnh gọi API, tính năng ý định, và biến mục tiêu, phát hiện phần mềm độc hại. Tính năng Chi-Squared được minh họa bởi công thức:

A black and white math symbol

Description automatically generated with medium confidence

Trong đó là thống kê kiểm tra Chi-Squared là tổng của tất cả các ô trong bảng dự phòng; O là tần suất quan sát được trong một ô; E là tần suất mong đợi trong một ô theo giả thuyết không. Để xác định các đặc trưng thiết yếu cho việc phát hiện malware trên Android và cải thiện độ chính xác của mô hình phát hiện, một bảng tần suất đã được tạo với các đặc trưng Permissions, API calls và Intents là các hàng, và biến mục tiêu (phát hiện malware) là các cột. Nghiên cứu đã sử dụng công thức kiểm định Chi-Squared để tính toán thống kê kiểm định Chi-Squared và giá trị p tương ứng cho việc phát hiện malware. Tần suất quan sát được xác định cho mỗi ô dựa trên các giá trị đặc trưng đã gán, và tần suất mong đợi được tính toán theo giả thuyết không. Các đặc trưng có thống kê Chi-Squared cao và giá trị p thấp đã được chọn để đưa vào tập đặc trưng cuối cùng, từ đó nâng cao độ chính xác của mô hình phát hiện malware trên Android.

## **3.5 Tầm quan trọng của việc lựa chọn tính năng**

Các kỹ thuật lựa chọn tính năng tiên tiến được sử dụng để tăng cường khả năng phát hiện phần mềm độc hại trên Android. Quá trình này bao gồm việc tinh chỉnh TUANDROMD, giải quyết tình trạng mất cân bằng dữ liệu và loại bỏ các biến chất lượng thấp. Trọng tâm là quyền, lệnh gọi API và ý định.

Lựa chọn tính năng Gain-ratio và kiểm tra Chi-Squared để xác định các tính năng thông tin, nâng cao hiệu quả phát hiện phần mềm độc hại và tăng cường độ chính xác và độ tin cậy của các mô hình dựa trên học máy trong việc xác định phần mềm độc hại Android, do đó làm giảm sự suy giảm entropy và ý nghĩa thống kê. Bản đồ nhiệt dưới đây minh họa mối tương quan giữa các tính năng khác nhau trong tập dữ liệu TUANDROMD.

A grid with red and blue lines

Description automatically generated

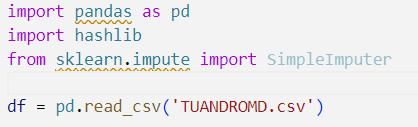
Các cụm đặc trưng: Có nhiều vùng đỏ, cho thấy một số nhóm đặc trưng có thể có mối tương quan cao với nhau. Ví dụ, các đặc trưng liên quan đến quyền truy cập (như ACCESS\_ALL\_DOWNLOADS, ACCESS\_FINE\_LOCATION,...) có thể liên kết mạnh mẽ với nhau.

Đặc trưng có mối quan hệ yếu: Một số ô có màu xanh nhạt hoặc gần như không có màu sắc, cho thấy những đặc trưng này không có mối quan hệ mạnh mẽ với nhau hoặc với các đặc trưng khác

## **3.6 Quy trình thực hiện**

**1.Tiền xử lý**

Bước 1: Đọc dữ liệu



* **pd.read\_csv('TUANDROMD.csv')**: Dùng Pandas để đọc tập dữ liệu từ file 'TUANDROMD.csv' và lưu vào biến df dưới dạng DataFrame. Đây là tập dữ liệu cần xử lý.

Bước 2: Loại bỏ các biến chất lượng thấp

A close-up of a number

Description automatically generated

* **threshold = len(df) \* 0.5**: Đặt ngưỡng để loại bỏ các cột có quá nhiều giá trị bị thiếu. Ở đây, threshold được đặt là 50% số lượng hàng. Nếu một cột có ít hơn 50% giá trị không bị thiếu, nó sẽ bị loại bỏ.
* **df.dropna(thresh=threshold, axis=1)**: Loại bỏ các cột (axis=1) có số giá trị khác null nhỏ hơn ngưỡng threshold. DataFrame sau khi loại bỏ các cột được lưu vào df\_cleaned.

Bước 3: Xử lý các giá trị bị thiếu

A close up of a text

Description automatically generated

* **df\_cleaned.fillna(df.median(), inplace=True)**: Thay thế tất cả các giá trị bị thiếu trong DataFrame df\_cleaned bằng trung vị của cột df.median(). Thao tác này sẽ được thực hiện trực tiếp trên DataFrame gốc (inplace=True), không tạo bản sao.

Bước 4: Tạo chuỗi đại diện cho mỗi hàng



* **df\_cleaned.astype(str)**: Chuyển tất cả các giá trị trong DataFrame df\_cleaned thành chuỗi để có thể kết hợp chúng.
* **.agg(' '.join, axis=1)**: Kết hợp các giá trị của từng hàng (axis=1) thành một chuỗi duy nhất, với các giá trị được ngăn cách bởi khoảng trắng ' '.
* **df\_cleaned['combined\_columns']**: Lưu chuỗi đã kết hợp của mỗi hàng vào cột mới có tên 'combined\_columns'.

Bước 5: Tính hàm băm SHA256 cho từng hàng



* **df\_cleaned['combined\_columns'].apply(lambda x: hashlib.sha256(x.encode()).hexdigest())**:
  + apply() áp dụng hàm lambda cho từng giá trị trong cột 'combined\_columns'.
  + hashlib.sha256(x.encode()).hexdigest(): Tính hàm băm SHA256 cho chuỗi đại diện của mỗi hàng. Hàm sha256() chuyển đổi chuỗi thành một giá trị băm duy nhất, sau đó hàm hexdigest() trả về chuỗi băm dưới dạng chuỗi ký tự thập lục phân (hexadecimal).
* **df\_cleaned['hash']**: Lưu kết quả băm SHA256 cho từng hàng vào cột mới có tên 'hash'.

Bước 6: Loại bỏ các hàng có hàm băm SHA256 trùng lặp



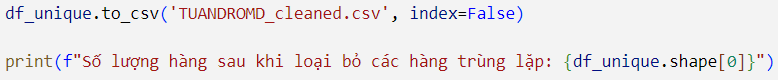
* **df\_cleaned.drop\_duplicates(subset=['hash'])**: Loại bỏ các hàng có giá trị hàm băm SHA256 trùng lặp trong cột 'hash'. Điều này giúp loại bỏ các hàng có dữ liệu giống nhau (sau khi chuyển đổi thành chuỗi và băm).
* **df\_unique**: Lưu kết quả sau khi loại bỏ các hàng trùng lặp vào DataFrame df\_unique.

Bước 7: Xóa các cột trung gian



* **df\_unique.drop(columns=['combined\_columns', 'hash'])**: Loại bỏ các cột trung gian combined\_columns (chuỗi kết hợp) và hash (giá trị băm) không cần thiết nữa từ DataFrame df\_unique.

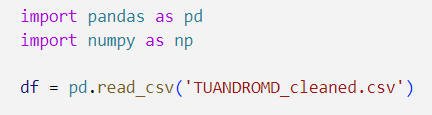
Bước 8: Lưu dữ liệu đã loại bỏ trùng lặp vào file CSV mới



* **df\_unique.to\_csv('TUANDROMD\_cleaned.csv', index=False)**: Lưu DataFrame df\_unique sau khi đã loại bỏ các hàng trùng lặp vào file CSV mới có tên 'TUANDROMD\_cleaned.csv'.
* **df\_unique.shape[0]**: Trả về số lượng hàng của DataFrame df\_unique (sau khi loại bỏ trùng lặp).
* **Kết quả:** Số lượng hàng sau khi loại bỏ các hàng trùng lặp: 663

**2.Lựa chọn tính năng**

Bước 1: Đọc dữ liệu



Bước 2: Phân loại các đặc trưng

A computer screen shot of a computer program

Description automatically generated

A screenshot of a computer screen

Description automatically generated

* Phân loại các đặc trưng thành các nhóm : Quyền truy cập, Gọi API và Intent thông qua các keyword

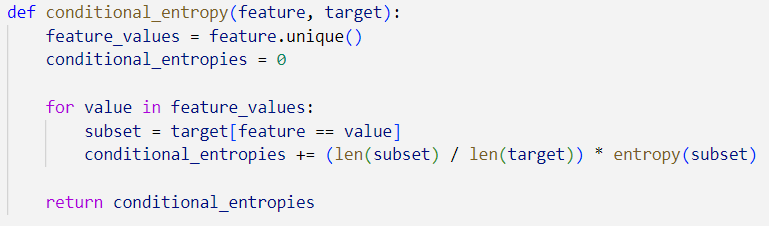
Bước 2: Tính Entropy của biến mục tiêu

A screenshot of a computer code

Description automatically generated

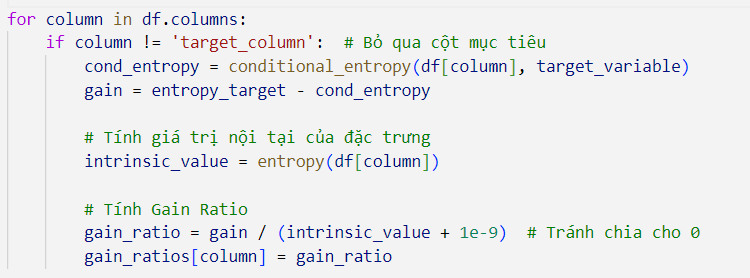
* **Hàm entropy(target)**: Hàm này tính entropy cho biến mục tiêu
  + value\_counts(): Đếm số lần xuất hiện của mỗi giá trị trong cột mục tiêu.
  + probabilities = value\_counts / len(target): Tính xác suất của từng giá trị.
  + np.log2(probabilities + 1e-9): Sử dụng log cơ số 2, thêm giá trị rất nhỏ (1e-9) để tránh trường hợp log(0).
  + Entropy được tính bằng công thức:  với  à xác suất của từng giá trị.
* **target\_variable = df['Label']**: Lấy cột 'Label' trong DataFrame df làm biến mục tiêu.
* **entropy\_target = entropy(target\_variable)**: Tính entropy của biến mục tiêu bằng cách gọi hàm entropy.

Bước 3: Tính Conditional Entropy cho mỗi đặc trưng



* **Hàm conditional\_entropy(feature, target)**: Hàm này tính entropy có điều kiện (Conditional Entropy) của biến mục tiêu dựa trên từng giá trị của một đặc trưng (feature).
  + feature\_values = feature.unique(): Lấy tất cả các giá trị khác nhau trong cột đặc trưng.
  + Vòng lặp for value in feature\_values: Duyệt qua từng giá trị khác nhau trong cột đặc trưng.
  + subset = target[feature == value]: Chọn tập con của biến mục tiêu ứng với giá trị cụ thể của đặc trưng.
  + conditional\_entropies: Tính entropy có điều kiện dựa trên từng giá trị của đặc trưng và tích lũy tổng entropy.

Bước 4: Tính Gain và Gain Ratio cho mỗi đặc trưng



* **Biến gain\_ratios = {}**: Tạo một từ điển rỗng để lưu trữ Gain Ratio của mỗi đặc trưng.
* **Vòng lặp for column in df.columns**: Duyệt qua tất cả các cột (đặc trưng) trong DataFrame df.
  + if column != 'target\_column': Đây là điều kiện để bỏ qua cột mục tiêu, nhưng do bạn đã sử dụng df['Label'] cho cột mục tiêu, điều kiện này cần sửa thành if column != 'Label'.
  + cond\_entropy = conditional\_entropy(df[column], target\_variable): Tính entropy có điều kiện của cột hiện tại.
  + **Gain**: Được tính bằng công thức: Gain=Entropy của mục tiêu - Conditional Entropy.
  + **Intrinsic Value**: Được tính bằng entropy của chính đặc trưng đó.
  + **Gain Ratio**: Được tính bằng: , thêm 1e-9 để tránh chia cho 0.
  + Sau đó, lưu giá trị Gain Ratio của từng cột vào từ điển gain\_ratios.

Bước 5: Lưu kết quả các đặc trưng Gain-ratio theo từng nhóm vào bộ dữ liệu mới

A screenshot of a computer

Description automatically generated

**Phân Tích Mã**

* **Lọc Các Đặc Trưng Cao Nhất:**
  + Biến top\_n được đặt là 20, điều này có nghĩa là bạn muốn chọn 20 đặc trưng hàng đầu cho mỗi nhóm (Permission, API calls, Intent).
  + Bạn sử dụng nlargest để chọn các đặc trưng có Gain-Ratio cao nhất từ gain\_ratios\_df cho từng nhóm.
* **Tạo Danh Sách Đặc Trưng Đã Chọn:**
  + Tạo danh sách selected\_features bao gồm các đặc trưng từ ba nhóm trên cùng với cột 'Label'.
* **Tạo DataFrame Mới:**
  + Sử dụng danh sách các đặc trưng đã chọn để tạo một DataFrame mới, final\_df, chỉ chứa các đặc trưng đã chọn.
* **Lưu DataFrame vào Tệp CSV:**
  + Cuối cùng, bạn lưu final\_df vào một tệp CSV có tên TUANDROMD\_gain.csv và in thông báo hoàn tất.
* **Kết quả:**

A screenshot of a computer program

Description automatically generated

* Thu được bộ dữ liệu gồm 60 đặc trưng và 1 cột nhãn

**3.Trích xuất vecto đặc trưng**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Phân Tích Mã**

* **Đọc Dữ Liệu:**
  + Bạn đọc dữ liệu từ tệp CSV TUANDROMD\_gain.csv vào DataFrame df.
* **Hàm Kiểm Định Chi-Squared:**
  + Hàm chi\_squared\_test nhận một đặc trưng và một biến mục tiêu (target) để tạo bảng tần suất và thực hiện kiểm định Chi-Squared, trả về thống kê Chi-Squared và p-value.
* **Kiểm Định Cho Permissions, API Calls, và Intents:**
  + Bạn kiểm tra các đặc trưng từ ba danh sách (permissions, API calls, intents) bằng cách duyệt qua từng đặc trưng và lưu kết quả vào các từ điển results\_permissions, results\_api\_calls, và results\_intents.
* **In Kết Quả:**
  + Kết quả của các kiểm định Chi-Squared cho mỗi nhóm được in ra.
* **Lọc Các Đặc Trưng Có p-value Nhỏ Hơn 0.05:**
  + Bạn lọc ra các đặc trưng có p-value nhỏ hơn 0.05 (có ý nghĩa thống kê) và lưu chúng vào các từ điển tương ứng.
* **Tạo Danh Sách Các Đặc Trưng Đã Chọn:**
  + Tạo danh sách final\_features chứa tất cả các đặc trưng đã chọn, bao gồm cột 'Label'.
* **Tạo DataFrame Mới:**
  + Tạo một DataFrame mới final\_df chỉ chứa các đặc trưng đã chọn.
* **Lưu Kết Quả vào Tệp CSV:**
  + Cuối cùng, bạn lưu final\_df vào tệp CSV TUANDROMD\_chi\_square.csv và in ra thông báo hoàn tất.
* **Kết quả**

A screenshot of a computer

Description automatically generated

* Thu được bộ dữ liệu gồm 51 đặc trưng và 1 nhãn label

**4.Tầm quan trọng của lựa chọn tính năng**

Bước 1: Đọc dữ liệu và hiển thị thông tin cơ bản

A white background with red text

Description automatically generated

* **pd.read\_csv('TUANDROMD\_cleaned.csv')**: Đọc file CSV 'TUANDROMD\_cleaned.csv' và lưu nó dưới dạng DataFrame df.

Bước 2: Kiểm tra các đặc trưng luôn bằng 0 và 1

A screenshot of a computer program

Description automatically generated

* **(df == 0).all(axis=0)**: Kiểm tra xem các cột có phải toàn bộ giá trị đều bằng 0 hay không. Kết quả là một mảng boolean, trong đó True chỉ ra rằng toàn bộ giá trị trong cột đó đều bằng 0.
* **.sum()**: Tính tổng số lượng cột mà tất cả các giá trị đều bằng 0. Kết quả được lưu vào biến always\_zero.
* **print("Number of features that are always zero:", always\_zero)**: In ra số lượng các cột mà tất cả các giá trị đều bằng 0.
* **(df == 1).all(axis=0)**: Kiểm tra xem có cột nào mà toàn bộ giá trị đều bằng 1 không, tương tự như cách làm với giá trị 0 ở trên.
* **.sum()**: Tính tổng số lượng cột mà tất cả các giá trị đều bằng 1, và kết quả được lưu vào biến always\_one.
* **print("Number of features that are always one:", always\_one)**: In ra số lượng cột có toàn bộ giá trị đều bằng 1.

Bước 3: Vẽ heatmap của ma trận tương quan

A close-up of a math equation

Description automatically generated

* **plt.figure(figsize=(20, 20))**: Tạo một biểu đồ với kích thước là 20x20 inch để đảm bảo tất cả các đặc trưng đều hiển thị rõ ràng.
* **sns.heatmap(corr\_matrix, annot=False, cmap='coolwarm')**: Vẽ heatmap cho ma trận tương quan. Mỗi ô trên heatmap đại diện cho hệ số tương quan giữa hai đặc trưng.
  + **annot=False**: Không hiển thị giá trị số lên các ô.
  + **cmap='coolwarm'**: Sử dụng bảng màu "coolwarm", với các giá trị âm được thể hiện bằng màu xanh lạnh và các giá trị dương được thể hiện bằng màu đỏ nóng.
* **plt.title('Correlation Matrix')**: Thêm tiêu đề cho biểu đồ.
* **plt.show()**: Hiển thị heatmap.

## **3.7 Kết quả**

* Sau khi thực hiện tiền xử lý dữ liệu gồm làm sạch và thay các giá trị bị thiếu bằng giá trị trung bình, ta thu được bộ dữ liệu mới “TUANDROMD\_clean.csv” gồm 663 ứng dụng phần mềm.
* Sau khi thực hiện lựa chọn tính năng với Gain-ratio, ta thu được bộ dữ liệu gồm 51 đặc trưng có gain cao nhất theo từng nhóm: Quyền truy cập, Quyền API, Intent và lưu thành bộ dữ liệu mới “TUANDROMD\_gain.csv”,
* Sau khi thực hiện lựa chọn vecto đặc trưng với Chi-square, ta thu được bộ dữ liệu gồm 51 đặc trưng có chi\_square với tần suất <0.5 và lưu thành bộ dữ liệu mới “TUANDROMD\_chi\_square.csv”, các đặc trưng bao gồm:

# **4. Đo lường hiệu suất**

Nghiên cứu đánh giá hiệu quả của các thuật toán chúng tôi đề xuất trong việc phát hiện phần mềm độc hại Android bằng các số liệu chính như lỗi bình phương trung bình (MSE), hệ số tương quan Pearson và lỗi bình phương trung bình căn bậc hai (RMSE). Nghiên cứu cũng xem xét các số liệu phân loại thiết yếu như độ chính xác (Accuracy), độ chính xác (Precision), độ thu hồi (Recall) và AUC (Area Under the ROC Curve) để đánh giá tính đúng đắn của các mô hình, khả năng giảm thiểu các kết quả dương tính giả, kết quả âm tính giả và sức mạnh phân biệt.

# **5. Triển khai và đánh giá kết quả**

## **5.1 Phân tích hiệu suất của các thuật toán được đề xuất**

Đề xuất ba thuật toán dựa trên ML, cụ thể là SVC, KNN và RF, để phát hiện và phân loại các ứng dụng Android độc hại. Để kiểm tra hiệu quả của các thuật toán, nghiên cứu này đã sử dụng tập dữ liệu di động về phần mềm độc hại nổi tiếng. Tập dữ liệu TUANDROMD chứa 4.464 ứng dụng độc hại và lành tính để đánh giá khả năng phát hiện các cuộc tấn công phần mềm độc hại trên các thiết bị Android, tuy nhiên sau khi làm sạch dữ liệu, ta chỉ còn 662 ứng dụng .Ở đây ,ta lựa chọn bộ dữ liệu “TUANDROMD\_chi\_square.csv” để thực hiện việc huấn luyển và đánh giá. Các tập dữ liệu này đã cung cấp các điểm chuẩn quan trọng để đánh giá các thuật toán dựa trên ML. Tuy nhiên, chất lượng và tính đa dạng của dữ liệu đào tạo cũng như các tính năng và tham số của thuật toán có thể làm giảm khả năng xác định các ứng dụng độc hại của chúng. Các mô hình dựa trên ML được đề xuất đã chứng minh là một phương pháp hiệu quả để phát hiện các ứng dụng Android độc hại.

### **5.1.1 Thuật toán Support Vector Classification (SVC)**

#### 5.1.1.1 Lý thuyết Support Vector Classification (SVC)

**SVC (Support Vector Classification)** là một dạng cụ thể của SVM, chuyên dùng cho các bài toán phân loại. Dưới đây là một cái nhìn sâu sắc về SVC, từ định nghĩa, nguyên lý hoạt động, cho đến các ứng dụng và ưu nhược điểm của nó.

**1. Định nghĩa**

SVC là một thuật toán học máy dùng để phân loại các điểm dữ liệu trong không gian đặc trưng. SVC tìm ra siêu phẳng tối ưu (optimal hyperplane) để phân chia các lớp trong không gian này, với mục tiêu tối đa hóa khoảng cách giữa các lớp.

**2. Nguyên lý hoạt động**

* **Siêu phẳng (Hyperplane)**: SVC tìm siêu phẳng tốt nhất trong không gian n chiều, được định nghĩa bởi phương trình:

**A math symbols with numbers and symbols

Description automatically generated with medium confidence**

Trong đó w là vector trọng số, x là vector đặc trưng của điểm dữ liệu, và b là độ dịch (bias).

* **Support Vectors**: Đây là các điểm dữ liệu gần nhất với siêu phẳng. Chỉ những điểm này ảnh hưởng đến vị trí của siêu phẳng và được sử dụng để xác định khoảng cách.
* **Margin (Khoảng cách)**: SVC cố gắng tối đa hóa khoảng cách từ siêu phẳng đến các support vectors. Margin lớn hơn thường đồng nghĩa với mô hình có khả năng tổng quát tốt hơn.

**3. Hàm mục tiêu**

SVC tối ưu hóa hàm mục tiêu để tìm ra siêu phẳng:

* **Hàm mục tiêu cho phân loại**: Tối ưu hóa hàm dưới với ràng buộc về khoảng cách:

A white background with black text

Description automatically generated

**4. Soft Margin**

Để xử lý điểm dữ liệu không thể phân tách hoàn toàn, SVC sử dụng softmargin với tham số C:

A math equation with black text

Description automatically generated

**5. Kernel Trick**

Khi dữ liệu không thể phân tách tuyến tính trong không gian ban đầu, SVC sử dụng kernel để ánh xạ dữ liệu vào không gian cao hơn, nơi dữ liệu có thể phân tách tuyến tính. Một số loại kernel phổ biến:

* **Linear Kernel**:



* **Polynomial Kernel**:



* **RBF Kernel**:



**6. Cách hoạt động**

1. **Chuẩn bị dữ liệu**: Dữ liệu và nhãn được tập hợp.
2. **Chọn kernel**: Tùy thuộc vào tính chất của dữ liệu, lựa chọn kernel phù hợp.
3. **Huấn luyện mô hình**: Tìm siêu phẳng tốt nhất thông qua tối ưu hóa.
4. **Đánh giá mô hình**: Sử dụng dữ liệu kiểm thử để đánh giá hiệu suất mô hình.
5. **Dự đoán**: Sử dụng mô hình đã huấn luyện để dự đoán nhãn cho dữ liệu mới.

**7. Ưu và nhược điểm**

* **Ưu điểm**:
  + Hiệu quả cao trong các bài toán phân loại với dữ liệu lớn.
  + Khả năng làm việc tốt với dữ liệu không tuyến tính thông qua kernel.
  + Tổng quát tốt và ít bị quá khớp nếu được tối ưu hóa đúng cách.
* **Nhược điểm**:
  + Thời gian huấn luyện có thể lâu với dữ liệu lớn.
  + Cần chọn tham số đúng (kernel, giá trị CCC) để đạt hiệu suất tốt nhất.
  + Không hiệu quả với dữ liệu nhiều noise và không cân bằng.

**8. Ứng dụng**

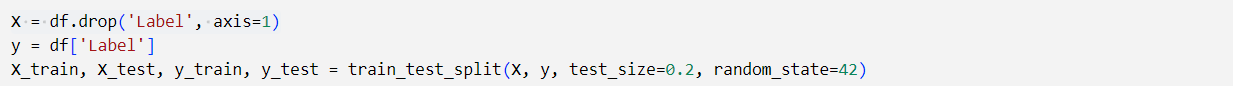
* Phân loại văn bản (như phân loại email spam).
* Nhận diện hình ảnh.
* Phân tích sinh học (phân loại gene).
* Dự đoán trong tài chính.

**Kết luận**

SVC là một thuật toán mạnh mẽ trong học máy, với khả năng phân loại hiệu quả trong nhiều tình huống. Bằng cách tìm kiếm siêu phẳng tối ưu, SVC có thể xử lý cả dữ liệu tuyến tính và không tuyến tính. Dù có một số nhược điểm, SVC vẫn là lựa chọn phổ biến trong nhiều ứng dụng thực tế.

#### **5.1.1.2 Triển khai**

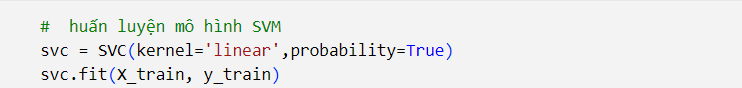
1. Chia dữ liệu thành tập huấn luyện và kiểm thử



**Giải thích:**

* Dữ liệu được chia thành hai phần: tập huấn luyện (X\_train, y\_train) và tập kiểm thử (X\_test, y\_test). Tỷ lệ chia là 80% cho tập huấn luyện và 20% cho tập kiểm thử, giúp mô hình học từ dữ liệu lớn hơn và đánh giá hiệu suất trên dữ liệu chưa thấy trước đó.
* random\_state=42 đảm bảo rằng việc chia dữ liệu là đồng nhất trong nhiều lần chạy khác nhau, giúp tái hiện kết quả trong các lần thử nghiệm khác nhau.

2. Huấn luyện mô hình SVC



**Giải thích:**

* Mô hình SVM được khởi tạo với kernel tuyến tính. Kernel tuyến tính giúp phân tách các lớp bằng một đường thẳng trong không gian đặc trưng.
* probability=True cho phép mô hình ước lượng xác suất cho mỗi dự đoán, điều này hữu ích cho các bài toán yêu cầu đánh giá độ tin cậy của dự đoán.
* Sau khi khởi tạo, mô hình được huấn luyện trên tập dữ liệu huấn luyện. Quá trình huấn luyện tìm ra các tham số tối ưu để phân tách các lớp khác nhau trong dữ liệu.

1. Tính ma trận nhầm lẫn và vẽ biểu đồ

A screen shot of a computer code

Description automatically generated

Giải thích:

* Ma trận nhầm lẫn cho biết số lượng dự đoán đúng và sai cho mỗi lớp. Mỗi hàng của ma trận thể hiện các mẫu thực tế, trong khi mỗi cột thể hiện các dự đoán của mô hình.
* Biểu đồ ma trận nhầm lẫn trực quan hóa số lượng các dự đoán chính xác và không chính xác, giúp đánh giá hiệu suất của mô hình theo cách trực quan. Mỗi ô trong ma trận chứa số lượng mẫu cho từng loại dự đoán, từ đó dễ dàng nhận biết những điểm yếu của mô hình.

1. Tính và vẽ đường cong ROC

A computer code with text

Description automatically generated with medium confidence

Giải thích:

* Đường cong ROC là một công cụ đánh giá mô hình phân loại, thể hiện mối quan hệ giữa tỷ lệ dương tính giả (False Positive Rate) và tỷ lệ dương tính thật (True Positive Rate) cho nhiều ngưỡng khác nhau. Đường cong ROC càng gần với góc trên bên trái của biểu đồ, mô hình càng tốt.
* AUC (Area Under Curve) là chỉ số tổng quát về hiệu suất phân loại của mô hình, với giá trị càng gần 1 cho thấy mô hình càng hiệu quả. Đường cong ROC giúp xác định ngưỡng tối ưu cho mô hình.

1. Tính và vẽ đường cong Precision-Recall

A screen shot of a computer code

Description automatically generated

Giải thích:

* Đường cong Precision-Recall giúp đánh giá khả năng của mô hình trong việc nhận diện các trường hợp dương tính, đặc biệt hữu ích trong các bài toán không cân bằng giữa các lớp.
* Average Precision (AP) là chỉ số thống kê cho biết độ chính xác của mô hình khi so sánh với recall, cung cấp cái nhìn sâu sắc hơn về chất lượng của các dự đoán dương tính. Đường cong này cho phép đánh giá khả năng phát hiện các trường hợp dương tính trong các bài toán phân loại.

1. Đánh giá hiệu năng

A screenshot of a computer program

Description automatically generated

* **Precision**: Tỷ lệ mẫu dự đoán đúng là dương so với tổng số mẫu được dự đoán là dương.
* **Recall**: Tỷ lệ mẫu dương được dự đoán đúng so với tổng số mẫu dương trong thực tế.
* **F1-score**: Trung bình điều hòa giữa Precision và Recall, giúp cân bằng giữa hai chỉ số này.
* **Accuracy**: Tỷ lệ mẫu được dự đoán đúng so với tổng số mẫu.

A screenshot of a computer error

Description automatically generated

* **MAE (Mean Absolute Error)**: Sai số trung bình tuyệt đối giữa giá trị dự đoán và giá trị thực tế.
* **MSE (Mean Squared Error)**: Sai số trung bình bình phương, nhấn mạnh hơn vào các sai số lớn.
* **R2-score**: Đo lường mức độ giải thích của mô hình trên biến phụ thuộc (giá trị càng gần 1 càng tốt).

#### **5.1.1.3 Kết quả**

A yellow and purple squares with black text

Description automatically generated A graph with a line and a blue line

Description automatically generatedA graph of a line

Description automatically generated

1. Biểu đồ ma trận nhầm lẫn
2. ROC
3. Precision-Recall Curve

Hiệu suất phân loại của SVC trên tập dữ liệu TUNADROMD

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-score | Accuracy |
| TUNADROMD | 0.862 | 0.758 | 0.806 | 0.910 |

**1. Ma trận nhầm lẫn (Confusion Matrix):**

* Ma trận nhầm lẫn thể hiện rằng mô hình phân loại đúng 96 mẫu thuộc lớp 0 và 25 mẫu thuộc lớp 1.
* Mô hình đã nhầm lẫn 4 mẫu lớp 0 thành lớp 1 và 8 mẫu lớp 1 thành lớp 0.
* Tỷ lệ nhầm lẫn của lớp 1 cao hơn so với lớp 0, cho thấy mô hình gặp khó khăn hơn khi phân biệt các mẫu thuộc lớp 1.
* Điều này phù hợp với việc Recall của lớp 1 thấp hơn, nghĩa là một số mẫu thuộc lớp 1 đã bị bỏ sót và phân loại nhầm vào lớp 0.

**2. Đường cong ROC (ROC Curve) và AUC:**

* Đường cong ROC (Receiver Operating Characteristic) cho thấy sự tương quan giữa Tỷ lệ Dương tính thật (True Positive Rate) và Tỷ lệ Dương tính giả (False Positive Rate) ở các ngưỡng khác nhau.
* Đường ROC của mô hình SVC tiến gần đến góc trên bên trái của biểu đồ, điều này phản ánh khả năng phân biệt tốt giữa hai lớp.
* Diện tích dưới đường cong ROC (AUC) là 0.93, cho thấy mô hình có khả năng phân biệt hai lớp khá tốt. Một giá trị AUC gần 1 thường được coi là tốt, và với giá trị 0.93, mô hình này có thể được coi là hiệu quả.
* Tuy nhiên, dù AUC cao, điều này không nhất thiết phản ánh hiệu quả tốt cho lớp thiểu số (lớp 1). Điều này cần được kiểm tra thêm qua đường cong Precision-Recall.

**3. Đường cong Precision-Recall:**

* Đường cong Precision-Recall giúp đánh giá hiệu suất của mô hình đối với lớp 1, đặc biệt khi dữ liệu mất cân bằng.
* Từ đường cong Precision-Recall, ta thấy rằng Precision có xu hướng giảm khi Recall tăng lên. Điều này phản ánh rằng khi mô hình cố gắng không bỏ sót (tăng Recall), thì tỷ lệ dương tính giả (False Positive) cũng tăng lên, làm giảm Precision.
* Giá trị Average Precision (AP) là 0.84, cho thấy hiệu suất tổng thể của mô hình đối với lớp 1 là khá tốt, nhưng không cao. Giá trị AP này phù hợp với nhận định rằng mô hình có thể bỏ lỡ một số mẫu thuộc lớp 1.
* Độ giảm mạnh của Precision khi Recall tăng cho thấy mô hình khó khăn trong việc duy trì độ chính xác cao cho lớp 1 khi cố gắng bao quát nhiều mẫu lớp 1 hơn. Điều này cũng được phản ánh qua Recall khá thấp (0.758).

**4. Bảng kết quả:**

* Precision (0.862) cao hơn Recall, nghĩa là khi mô hình dự đoán một mẫu là lớp 1, thì xác suất mẫu đó thực sự thuộc lớp 1 là khá cao. Tuy nhiên, khả năng bao quát các mẫu lớp 1 vẫn còn hạn chế.
* Recall (0.758) thấp hơn, chỉ ra rằng mô hình đã bỏ qua một số lượng đáng kể các mẫu thuộc lớp 1.
* F1-score (0.806) là trung bình điều hòa giữa Precision và Recall, cho thấy hiệu suất tổng thể của mô hình đối với lớp 1 là khá ổn nhưng chưa thực sự tối ưu.
* Accuracy (0.910) cao, nhưng điều này chủ yếu do mô hình phân loại tốt lớp 0, chiếm tỷ lệ lớn hơn. Accuracy không phản ánh đầy đủ khả năng của mô hình trong việc phân loại lớp 1 khi dữ liệu mất cân bằng.

**Kết luận:**

* Mô hình SVC đạt hiệu suất phân loại cao cho lớp 0, nhưng hiệu suất cho lớp 1 chưa thực sự tốt, đặc biệt là về Recall.
* Với giá trị AUC cao và đường cong ROC khá tốt, mô hình có khả năng phân biệt giữa hai lớp. Tuy nhiên, đường cong Precision-Recall cho thấy rằng mô hình gặp khó khăn trong việc duy trì Precision khi tăng Recall cho lớp 1.
* Để cải thiện mô hình, có thể xem xét các kỹ thuật như điều chỉnh ngưỡng quyết định, sử dụng phương pháp cân bằng dữ liệu, hoặc tối ưu hóa siêu tham số để tăng Recall cho lớp 1 mà không làm giảm Precision quá nhiều.

### **5.1.2 Thuật toán K-Nearest Neighbors (KNN)**

#### **5.1.2.1.  Lý thuyết thuật toán K-Nearest Neighbors (KNN)**

* Định nghĩa:  
  KNN là thuật toán phân loại hoặc hồi quy dựa trên láng giềng gần nhất. Một điểm dữ liệu mới được gán nhãn dựa trên nhãn của K điểm gần nhất trong không gian đặc trưng.
* Hàm mục tiêu:
  + Phân loại: Tìm nhãn phổ biến nhất (mode) trong số K láng giềng gần nhất.
  + Hồi quy: Tính trung bình giá trị của K láng giềng gần nhất.
* Hàm khoảng cách:  
  KNN sử dụng các hàm để đo khoảng cách giữa điểm mới và các điểm dữ liệu khác, phổ biến nhất là:
  + Khoảng cách Euclid:
* A mathematical equation with numbers and symbols

  Description automatically generated
  + Ngoài ra, có thể dùng khoảng cách Manhattan, Minkowski, hoặc Hamming.
* Hàm mất mát:  
  Đối với phân loại, sử dụng Hàm mất mát 0-1 (Zero-One Loss):A black text with white text

  Description automatically generated with medium confidence  
  Đối với hồi quy, thường sử dụng Mean Squared Error (MSE):  
  A mathematical equation with numbers and symbols

  Description automatically generated
* Cách hoạt động:
  + Tìm K láng giềng gần nhất của điểm cần dự đoán.
  + Bỏ phiếu đa số (phân loại) hoặc tính trung bình (hồi quy) từ K láng giềng.
* Ưu và nhược điểm:
  + Ưu điểm: Dễ hiểu, không cần huấn luyện mô hình phức tạp, hiệu quả với dữ liệu nhỏ.
  + Nhược điểm: Chậm với dữ liệu lớn, nhạy cảm với giá trị ngoại lai, cần chọn K tối ưu.
* Tóm lại, KNN là thuật toán đơn giản nhưng hiệu quả cho các bài toán phân loại và hồi quy khi kích thước dữ liệu không quá lớn.

#### **5.1.2.2. Triển khai**

Các bước làm:

1. **Chia dữ liệu:** [Tương tự SVC](#_5.1.1.2_Triển_khai)
2. **Khởi tạo và huấn luyện mô hình:** Sử dụng KNeighborsClassifier từ sklearn.neighbors để khởi tạo mô hình và huấn luyện với dữ liệu huấn luyện

A screenshot of a computer

Description automatically generated

Giải thích:

1. Khởi tạo mô hình KNN:

    KNeighborsClassifier(n\_neighbors=5):

* Tạo một đối tượng mô hình KNN với 5 láng giềng gần nhất.
* n\_neighbors=5 là tham số cho biết mô hình sẽ xem xét 5 điểm dữ liệu gần nhất để đưa ra dự đoán cho một mẫu mới.

2. Huấn luyện mô hình:

      knn.fit(X\_train, y\_train):

* Dùng tập dữ liệu huấn luyện X\_train (đặc trưng) và y\_train (nhãn) để mô hình học cách phân loại.
* Sau khi huấn luyện, mô hình có thể dự đoán nhãn cho dữ liệu chưa biết dựa vào các điểm lân cận trong không gian đặc trưng.

1. [Biểu đồ triển khai tương tư như SVC](#_5.1.1.2_Triển_khai)
2. **Đánh giá hiệu năng:** [Tương tự triển khai SVC](#_5.1.1.2_Triển_khai)

#### **5.1.2.3. Kết quả**

A yellow and purple squares with black numbers

Description automatically generated A graph with a line

Description automatically generated

A graph of a line

Description automatically generated

1. Biểu đồ ma trận nhầm lẫn
2. ROC
3. Precision-Recall Curve

Hiệu suất phân loại của KNN trên tập dữ liệu TUNADROMD

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-score | Accuracy |
| TUNADROMD | 0.92 | 0.697 | 0.763 | 0.91 |

**1. Ma trận nhầm lẫn (Confusion Matrix):**

* Mô hình đã dự đoán đúng 98 mẫu thuộc lớp 0 và 23 mẫu thuộc lớp 1.
* Có 2 mẫu thuộc lớp 0 bị nhầm thành lớp 1, và 10 mẫu thuộc lớp 1 bị nhầm thành lớp 0.
* Điều này cho thấy mô hình KNN phân loại chính xác phần lớn các mẫu lớp 0, nhưng có tỷ lệ nhầm lẫn khá cao cho lớp 1. Nhìn chung, mô hình gặp khó khăn hơn trong việc nhận diện lớp 1.

**2. Đường cong ROC (ROC Curve) và AUC:**

* Đường cong ROC của mô hình cho thấy mô hình có khả năng phân biệt giữa hai lớp khá tốt.
* Diện tích dưới đường cong ROC (AUC) là 0.90, chỉ ra rằng mô hình có hiệu suất phân loại tương đối cao. Tuy nhiên, AUC của mô hình KNN thấp hơn một chút so với mô hình SVC (AUC = 0.93).
* Đường cong ROC không tiến quá gần góc trên bên trái, cho thấy rằng độ chính xác trong việc phân biệt giữa lớp 0 và lớp 1 chưa hoàn hảo.

**3. Đường cong Precision-Recall:**

* Đường cong Precision-Recall giúp đánh giá chính xác hiệu suất của mô hình đối với lớp 1, đặc biệt là khi dữ liệu mất cân bằng.
* Precision có xu hướng giảm mạnh khi Recall tăng. Điều này cho thấy rằng khi cố gắng tăng Recall để bao phủ các mẫu lớp 1, độ chính xác của mô hình (Precision) giảm đáng kể.
* Giá trị **Average Precision (AP)** là 0.81, thấp hơn một chút so với AP của mô hình SVC (0.84). Điều này cho thấy mô hình KNN có hiệu suất thấp hơn khi phân loại lớp 1.
* Sự sụt giảm nhanh của Precision khi tăng Recall cho thấy rằng mô hình KNN gặp khó khăn trong việc duy trì độ chính xác cao cho lớp 1, đặc biệt khi cố gắng bao quát nhiều mẫu lớp 1 hơn.

**4. Bảng kết quả:**

* **Precision** (0.92) khá cao, cho thấy khi mô hình dự đoán một mẫu thuộc lớp 1, xác suất mẫu đó thực sự thuộc lớp 1 là khá cao. Điều này nghĩa là mô hình ít tạo ra dương tính giả (false positives) khi phân loại lớp 1.
* **Recall** (0.697) lại thấp hơn, chỉ ra rằng mô hình bỏ sót nhiều mẫu thuộc lớp 1. Mức Recall này cho thấy mô hình KNN không bao phủ được toàn bộ các mẫu lớp 1, khiến khả năng phát hiện các mẫu lớp 1 bị hạn chế.
* **F1-score** (0.763) là trung bình điều hòa giữa Precision và Recall, và giá trị này cho thấy hiệu suất tổng thể của mô hình đối với lớp 1 ở mức khá, nhưng chưa tốt.
* **Accuracy** đạt 0.91, cao và gần giống với kết quả của mô hình SVC. Tuy nhiên, như đã đề cập, độ chính xác tổng thể (Accuracy) bị ảnh hưởng bởi số lượng lớn các mẫu lớp 0 và không phản ánh đầy đủ hiệu suất phân loại cho lớp 1.

**Kết luận:**

* Mô hình KNN có Precision cao, nhưng Recall thấp cho lớp 1, cho thấy mô hình có xu hướng dự đoán an toàn cho lớp 0 và dễ bỏ sót các mẫu thuộc lớp 1.
* Mặc dù AUC đạt 0.90 và Accuracy đạt 0.91, hiệu suất đối với lớp 1 chưa thực sự tốt, đặc biệt là với Recall thấp.
* Để cải thiện mô hình KNN, có thể thử điều chỉnh số lượng láng giềng (k) hoặc sử dụng các phương pháp cân bằng dữ liệu như SMOTE để cải thiện khả năng phát hiện lớp 1.

### **5.1.3 Thuật toán Random Forest (RF)**

#### **5.1.3.1. Lý thuyết thuật toán Random Forest (RF)**

**Định nghĩa:**

Random Forest là một thuật toán học máy được sử dụng cho các bài toán phân loại và hồi quy. Thuật toán này xây dựng một tập hợp các cây quyết định (decision trees) và kết hợp các kết quả của chúng để đưa ra dự đoán chính xác hơn.

**Cách hoạt động:**

1. Xây dựng cây quyết định:
   * Tạo ra nhiều cây quyết định bằng cách lấy mẫu ngẫu nhiên từ tập dữ liệu (bagging).
   * Trong quá trình xây dựng mỗi cây, chỉ một tập hợp con các đặc trưng được chọn ngẫu nhiên để quyết định điểm chia tách.
2. Dự đoán:
   * Đối với bài toán phân loại: Mỗi cây trong rừng sẽ đưa ra một dự đoán và kết quả cuối cùng sẽ là nhãn được chọn nhiều nhất (bỏ phiếu đa số).
   * Đối với bài toán hồi quy: Kết quả cuối cùng là trung bình giá trị dự đoán của tất cả các cây.

**Hàm mục tiêu:**

* Phân loại: Tìm nhãn phổ biến nhất (mode) trong số các cây quyết định.

A mathematical equation with a square and a triangle

Description automatically generated with medium confidence

A mathematical equation with numbers and symbols

Description automatically generated

* Hồi quy: Tính trung bình giá trị dự đoán từ các cây quyết định.

A black text on a white background

Description automatically generated

**Hàm mất mát:**

* Đối với phân loại: Sử dụng độ chính xác (accuracy) hoặc hàm mất mát entropy.

A black text with black letters

Description automatically generated with medium confidence

* Đối với hồi quy: Sử dụng Mean Squared Error (MSE).

A black text on a white background

Description automatically generated

**Ưu và nhược điểm:**

* Ưu điểm:
  + Chống overfitting: Random Forest ít nhạy cảm với overfitting hơn so với cây quyết định đơn lẻ.
  + Khả năng xử lý dữ liệu lớn: Thích hợp cho các tập dữ liệu lớn và phức tạp.
  + Tính chính xác cao: Đem lại độ chính xác tốt trong nhiều bài toán phân loại và hồi quy.
  + Phát hiện đặc trưng quan trọng: Có khả năng đánh giá độ quan trọng của các đặc trưng trong dữ liệu.
* Nhược điểm:
  + Khó giải thích: So với cây quyết định đơn lẻ, kết quả của Random Forest khó hiểu hơn.
  + Tốc độ: Thời gian dự đoán có thể lâu hơn, nhất là với một số lượng cây lớn.
  + Bộ nhớ: Yêu cầu bộ nhớ lớn để lưu trữ nhiều cây quyết định.

#### **5.1.3.2 Triển khai**

1. **Chia dữ liệu:** [Tương tự triển khai SVC](#_5.1.1.2_Triển_khai)
2. **Khởi tạo và huấn luyện mô hình:** Sử dụng RandomForestClassifier từ sklearn.ensemble để khởi tạo mô hình và huấn luyện với dữ liệu huấn luyện.

A screenshot of a computer program

Description automatically generated

**Giải thích:**

Tạo một đối tượng mô hình Random Forest (model)

- **n\_estimators=100:** Chỉ định số lượng cây quyết định trong mô hình. 100 cây sẽ được sử dụng trong quá trình huấn luyện.

- **random\_state=42**: Thiết lập giá trị ngẫu nhiên để đảm bảo kết quả mô hình có thể tái tạo được

3. **Biểu đồ:** [Tương tự triển khai SVC](#_5.1.1.2_Triển_khai)

4. **Đánh giá hiệu năng:** [Tương tự triển khai SVC](#_5.1.1.2_Triển_khai)

#### **5.1.3.3. Kết quả**

A graph of a line

Description automatically generatedA yellow and purple squares with black numbers

Description automatically generatedA graph with a line and a blue line

Description automatically generated

1. Biểu đồ ma trận nhầm lẫn
2. ROC with UAC
3. Precision-Recall Curve

Hiệu suất phân loại của RF trên tập dữ liệu TUNADROMD

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-score | Accuracy |
| TUNADROMD | 0.962 | 0.758 | 0.847 | 0.932 |

**1. Ma trận nhầm lẫn (Confusion Matrix):**

* Mô hình đã dự đoán chính xác 99 mẫu thuộc lớp 0 và 25 mẫu thuộc lớp 1.
* Chỉ có 1 mẫu lớp 0 bị nhầm thành lớp 1 và 8 mẫu lớp 1 bị nhầm thành lớp 0.
* Điều này cho thấy mô hình RF rất tốt trong việc phân loại chính xác lớp 0, đồng thời cũng có độ chính xác khá cao đối với lớp 1. Tuy nhiên, một số mẫu lớp 1 vẫn bị bỏ sót (10 mẫu nhầm thành lớp 0), cho thấy vẫn còn một mức độ khó khăn trong việc phân loại lớp này.

**2. Đường cong ROC (ROC Curve) và AUC:**

* Đường cong ROC của mô hình RF cho thấy tỷ lệ True Positive Rate cao với False Positive Rate thấp, điều này thể hiện khả năng phân biệt giữa hai lớp khá tốt.
* Diện tích dưới đường cong ROC (AUC) đạt 0.94, đây là một con số rất cao, phản ánh hiệu suất phân loại tổng thể của mô hình RF rất tốt. AUC của mô hình RF cao hơn so với các mô hình SVC (AUC = 0.93) và KNN (AUC = 0.90).
* Đường ROC của mô hình gần sát góc trên bên trái, cho thấy độ chính xác trong việc phân biệt giữa lớp 0 và lớp 1 rất cao.

**3. Đường cong Precision-Recall:**

* Đường cong Precision-Recall cho thấy khi Recall tăng lên thì Precision giảm đi, nhưng nhìn chung mức giảm không quá nhanh, điều này cho thấy mô hình RF vẫn duy trì được một mức độ chính xác nhất định khi cố gắng bao phủ nhiều mẫu lớp 1 hơn.
* Giá trị **Average Precision (AP)** là 0.89, cao hơn so với KNN (AP = 0.81) và chỉ thấp hơn một chút so với SVC (AP = 0.84). Điều này cho thấy mô hình RF có hiệu suất tốt trong việc phân loại lớp 1, dù vẫn còn bỏ sót một số mẫu.
* Độ giảm dần của Precision khi Recall tăng cho thấy rằng mô hình RF gặp ít khó khăn hơn so với KNN và có thể duy trì mức độ chính xác tốt hơn cho lớp 1.

**4. Bảng kết quả:**

* **Precision (0.962)** cao, cho thấy khi mô hình dự đoán một mẫu là lớp 1, xác suất mẫu đó thực sự thuộc lớp 1 là rất cao, tức là mô hình tạo ra rất ít dương tính giả (false positives).
* **Recall (0.758)** cao hơn so với KNN nhưng tương đương với SVC, cho thấy mô hình RF bao phủ khá tốt các mẫu thuộc lớp 1, mặc dù một số mẫu vẫn bị bỏ sót.
* **F1-score (0.847)** là trung bình điều hòa giữa Precision và Recall, với giá trị cao hơn hẳn so với các mô hình KNN (0.763) và SVC (0.806), cho thấy mô hình RF có hiệu suất tổng thể tốt hơn đối với lớp 1.
* **Accuracy (0.932)** cũng đạt mức cao nhất so với các mô hình khác, cho thấy RF có độ chính xác tổng thể rất tốt.

**Kết luận:**

* Mô hình RF có khả năng phân loại tốt cả hai lớp, với AUC, Precision, Recall, và F1-score đều cao, phản ánh khả năng phân biệt và dự đoán chính xác cao.
* Mặc dù mô hình RF vẫn bỏ sót một số mẫu lớp 1, nhưng nhìn chung, nó hoạt động hiệu quả hơn so với các mô hình SVC và KNN trên tập dữ liệu này.
* Với hiệu suất tổng thể tốt nhất, mô hình RF có thể là lựa chọn ưu tiên cho tập dữ liệu TUNADROMD.

## **5.2** **So sánh phân tích độ chính xác của thuật toán dựa trên ML được đề xuất**

Để đánh giá độ chính xác dự đoán của các thuật toán dựa trên ML, báo cáo này đã sử dụng một số số liệu thống kê, bao gồm lỗi tuyệt đối trung bình (MAE), lỗi bình phương trung bình (MSE) và R bình phương (R2). Các lỗi dự đoán thu được giữa lớp mục tiêu và các giá trị dự đoán được trình bày trong bảng dưới đây. Phân tích được trình bày cung cấp những hiểu biết có giá trị về hiệu suất của các mô hình dựa trên ML, cho phép chúng tôi xác định và giải quyết mọi lỗi dự đoán.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TUNADROMD | Thuật toán | MAE | MSE | R2-score |
| RF | 0.068 | 0.068 | 0.637 |
| SVC | 0.09 | 0.09 | 0.516 |
| KNN | 0.09 | 0.09 | 0.516 |

* Phân tích từ bảng kết quả
* MAE (Mean Absolute Error):
  + Mô hình Random Forest (RF) có MAE thấp nhất (0.068), cho thấy nó có độ chính xác tốt nhất trong việc dự đoán giá trị thực tế.
  + Cả Support Vector Classifier (SVC) và K-Nearest Neighbors (KNN) đều có MAE cao hơn (0.09), cho thấy các mô hình này có độ sai lệch lớn hơn trong dự đoán, nghĩa là chúng không chính xác bằng RF.
* MSE (Mean Squared Error):
  + Tương tự như MAE, MSE của RF cũng thấp nhất (0.068), cho thấy mô hình này không chỉ có sai số tuyệt đối thấp mà còn có sai số bình phương cũng thấp hơn nhiều so với SVC và KNN.
  + SVC và KNN có MSE bằng nhau (0.09), điều này cho thấy chúng có cùng mức độ sai lệch nhưng không đạt được hiệu suất tốt như RF.
* R²-score:
  + R²-score của RF là 0.637, cho thấy mô hình này giải thích được 63.7% sự biến động của dữ liệu. Điều này là rất tốt và chứng tỏ RF có khả năng nắm bắt các mối quan hệ trong dữ liệu tốt hơn.
  + SVC và KNN đều có R²-score là 0.516, nghĩa là chúng chỉ giải thích được 51.6% sự biến động của dữ liệu. Kết quả này cho thấy cả hai mô hình này không hoạt động hiệu quả trong việc dự đoán, với khả năng giải thích biến động kém hơn nhiều so với RF.
* Kết luận
  + Mô hình Random Forest (RF) là lựa chọn tốt nhất trong ba mô hình dựa trên các chỉ số MAE, MSE và R²-score. RF không chỉ có độ chính xác cao mà còn giải thích tốt nhất sự biến động trong dữ liệu.
  + Support Vector Classifier (SVC) và K-Nearest Neighbors (KNN) có hiệu suất tương đương nhưng kém hơn nhiều so với RF. Cả hai mô hình này có mức độ sai lệch lớn hơn và không giải thích được nhiều sự biến động trong dữ liệu.
  + KNN có thể xem là lựa chọn kém hiệu quả nhất trong số ba mô hình này, cần cân nhắc điều chỉnh tham số hoặc sử dụng mô hình khác để cải thiện hiệu suất dự đoán.

# **6. Thảo luận**

Việc sử dụng điện thoại thông minh với các chức năng mới và các ứng dụng Android liên quan đã gia tăng đáng kể do sự phát triển nhanh chóng của công nghệ. Theo báo cáo của Statista, sẽ có 7.074 triệu điện thoại thông minh được sử dụng vào năm 2024. Điều này cũng tạo ra thách thức cho các nhà nghiên cứu và nhà phát triển cơ chế bảo mật cho các ứng dụng này, xuất phát từ sự phức tạp và lỗ hổng mới của các ứng dụng Android mà tin tặc có thể dễ dàng khai thác. Việc đánh giá đúng dữ liệu ứng dụng liên quan đến bảo mật là rất quan trọng, đặc biệt là đối với các ứng dụng Android trong thương mại điện tử, kinh doanh điện tử, tiết kiệm và ngân hàng trực tuyến. Điều này là vì các ứng dụng như vậy liên quan đến việc truyền tải thông tin bí mật và có giá trị qua các mạng di động.

Để ngăn chặn các lỗ hổng bảo mật trong mạng, các thuật toán dựa trên học máy (ML) được sử dụng để giám sát và phát hiện các cuộc tấn công độc hại trên ứng dụng Android. Nghiên cứu này đóng góp đáng kể vào an ninh mạng bằng cách phát triển một khung ML để xác định các bất thường trong cơ sở dữ liệu chữ ký. Do đó, hệ thống có khả năng phát hiện các ứng dụng độc hại chưa được biết đến. Độ chính xác của phương pháp ML trong việc phát hiện phần mềm độc hại Android bị ảnh hưởng bởi một số yếu tố chính. Chất lượng và sự đa dạng của bộ dữ liệu TUANDROMD cung cấp một cơ sở đào tạo toàn diện. Các thuật toán được chọn, bao gồm Random Forest (RF), SVC và KNN, có thế mạnh trong việc xử lý tính năng hiệu quả và khả năng phân loại, đóng góp đáng kể vào độ chính xác. Các kỹ thuật lựa chọn đặc trưng tiên tiến đảm bảo tính liên quan và tính thông tin của các tính năng được sử dụng. Tối ưu hóa siêu tham số của từng thuật toán tối ưu hóa hiệu suất, ngăn chặn hiện tượng quá khớp hoặc khớp dưới mức. Cuối cùng, khả năng thích ứng của các mô hình với tính chất thay đổi của phần mềm độc hại Android là yếu tố thiết yếu để duy trì độ chính xác theo thời gian. Những yếu tố này cộng lại giúp tăng cường khả năng của các mô hình trong việc phát hiện chính xác phần mềm độc hại.

Dưới đây là kết quả so sánh hiệu suất giữa các mô hình:

| Chỉ số | KNN | SVC | RF |
| --- | --- | --- | --- |
| Precision | 0.920 | 0.862 | 0.962 |
| Recall | 0.697 | 0.758 | 0.758 |
| F1-score | 0.763 | 0.806 | 0.847 |
| Accuracy | 0.910 | 0.910 | 0.932 |
| AUC (ROC) | 0.90 | 0.93 | 0.94 |
| Average Precision (AP) | 0.81 | 0.84 | 0.89 |
| Dương tính giả (False Positives) | 2 | 4 | 1 |
| Dương tính thật (True Positives) | 23 | 25 | 25 |
| Âm tính giả (False Negatives) | 10 | 8 | 8 |
| Âm tính thật (True Negatives) | 98 | 96 | 99 |

Bảng so sánh các chỉ số

**1. Precision:**

* KNN (0.920): Mô hình KNN đạt Precision cao, cho thấy khi KNN dự đoán một mẫu thuộc lớp 1, khả năng mẫu đó thực sự thuộc lớp 1 là cao. Tuy nhiên, Precision này chưa đạt mức cao nhất.
* SVC (0.862): Mô hình SVC có Precision thấp nhất trong ba mô hình, điều này có nghĩa là SVC tạo ra nhiều dương tính giả hơn. Precision thấp cho thấy khi SVC dự đoán một mẫu thuộc lớp 1, xác suất dự đoán đó chính xác là không cao bằng các mô hình khác.
* RF (0.962): Mô hình RF có Precision cao nhất, nghĩa là khi RF dự đoán một mẫu là lớp 1, độ tin cậy của dự đoán đó rất cao. Đây là mô hình có khả năng phân loại dương tính giả (false positive) tốt nhất trong ba mô hình.

**2. Recall:**

* KNN (0.697): KNN có Recall thấp nhất trong ba mô hình, nghĩa là KNN bỏ sót khá nhiều mẫu thuộc lớp 1, khiến khả năng bao phủ các mẫu thực sự thuộc lớp 1 không tốt. Điều này cho thấy mô hình có xu hướng dự đoán các mẫu về lớp 0 hơn, dẫn đến bỏ sót các mẫu lớp 1.
* SVC (0.758): SVC có Recall cao hơn KNN, đồng nghĩa với việc SVC nhận diện các mẫu lớp 1 tốt hơn KNN. Tuy nhiên, mức Recall này vẫn chưa đạt tối ưu so với mô hình RF.
* RF (0.758): Mặc dù RF có Precision cao nhất, Recall của RF tương đương với SVC. Điều này cho thấy RF có khả năng cân bằng tốt giữa việc nhận diện đúng các mẫu lớp 1 và tránh các dương tính giả.

**3. F1-score:**

* KNN (0.763): F1-score của KNN thấp nhất, cho thấy KNN gặp khó khăn trong việc cân bằng giữa Precision và Recall cho lớp 1. Điều này phản ánh khả năng phân loại mẫu lớp 1 của KNN là yếu nhất trong ba mô hình.
* SVC (0.806): F1-score của SVC cao hơn KNN, cho thấy SVC hoạt động hiệu quả hơn trong việc phân loại lớp 1, mặc dù vẫn chưa đạt mức tốt nhất.
* RF (0.847): RF đạt F1-score cao nhất, phản ánh hiệu suất tổng thể tốt trong việc phân loại mẫu lớp 1, với khả năng duy trì độ chính xác và độ bao phủ cao. Đây là mô hình tốt nhất trong ba mô hình về mặt hiệu suất tổng thể.

**4. Accuracy:**

* KNN (0.910) và SVC (0.910): Cả hai mô hình này đều có Accuracy bằng nhau, đạt 91%. Điều này phản ánh độ chính xác tổng thể của mô hình trong việc phân loại các mẫu thuộc cả lớp 0 và lớp 1.
* RF (0.932): RF có Accuracy cao nhất, đạt 93.2%. Đây là mô hình chính xác nhất khi phân loại cả hai lớp, đặc biệt là lớp 0. Tuy nhiên, do tập dữ liệu có thể mất cân bằng với số lượng mẫu lớp 0 nhiều hơn, Accuracy cao không phản ánh hoàn toàn hiệu suất của mô hình đối với lớp 1.

**5. AUC (ROC Curve):**

* KNN (0.90): KNN có AUC là 0.90, cho thấy mô hình có khả năng phân biệt giữa lớp 0 và lớp 1 khá tốt. Tuy nhiên, đây là giá trị thấp nhất trong ba mô hình, phản ánh việc KNN gặp khó khăn hơn trong việc tách biệt hai lớp.
* SVC (0.93): SVC có AUC là 0.93, cao hơn KNN. Điều này cho thấy SVC có khả năng phân biệt hai lớp tốt hơn và hiệu suất cao hơn KNN trong việc giảm thiểu lỗi phân loại.
* RF (0.94): RF có AUC cao nhất, đạt 0.94. Điều này phản ánh RF có khả năng phân biệt tốt nhất giữa lớp 0 và lớp 1, giúp tối ưu hóa hiệu suất phân loại tổng thể.

**6. Average Precision (AP):**

* KNN (0.81): AP của KNN là 0.81, cho thấy khả năng phân loại lớp 1 của mô hình ở mức khá. Tuy nhiên, KNN không duy trì được mức Precision ổn định khi Recall tăng lên.
* SVC (0.84): AP của SVC là 0.84, cao hơn KNN, cho thấy khả năng nhận diện lớp 1 tốt hơn và duy trì Precision cao khi tăng Recall.
* RF (0.89): AP của RF đạt 0.89, cao nhất trong ba mô hình, thể hiện hiệu suất ổn định nhất khi phân loại lớp 1. Điều này cho thấy RF không chỉ đạt Precision cao mà còn giữ được Precision ổn định khi Recall tăng.

**7. Chi tiết ma trận nhầm lẫn:**

* KNN: KNN dự đoán nhầm 10 mẫu thuộc lớp 1 thành lớp 0, cho thấy khả năng phân biệt lớp 1 của KNN chưa tốt. Mặc dù mô hình dự đoán rất chính xác lớp 0 với chỉ 2 mẫu bị nhầm sang lớp 1, nhưng vẫn còn bỏ sót nhiều mẫu lớp 1.
* SVC: SVC dự đoán nhầm 8 mẫu lớp 1 thành lớp 0 và 4 mẫu lớp 0 thành lớp 1. SVC có độ cân bằng tốt hơn giữa hai loại sai số so với KNN.
* RF: RF chỉ nhầm 1 mẫu lớp 0 thành lớp 1 và 8 mẫu lớp 1 thành lớp 0, là mô hình có độ chính xác cao nhất đối với lớp 0 và cũng duy trì độ chính xác khá tốt với lớp 1.

**Kết luận:**

* RF Classifier là mô hình có hiệu suất tổng thể tốt nhất với Precision, F1-score, Accuracy, AUC, và AP cao nhất, đặc biệt là khả năng phân biệt giữa hai lớp rất tốt. Đây là mô hình lý tưởng để áp dụng trên tập dữ liệu TUNADROMD.
* SVC Classifier cũng hoạt động hiệu quả với chỉ số Recall và AUC cao, cho thấy mô hình có khả năng phát hiện các mẫu lớp 1 tốt. Tuy nhiên, Precision thấp hơn RF khiến cho SVC có độ tin cậy thấp hơn khi dự đoán dương tính.
* KNN Classifier có độ chính xác cao trong phân loại lớp 0, nhưng bỏ sót nhiều mẫu thuộc lớp 1, dẫn đến Recall và F1-score thấp hơn so với hai mô hình còn lại.
* Do đó, RF Classifier là lựa chọn ưu tiên nhất cho tập dữ liệu này nhờ khả năng phân loại vượt trội giữa các lớp và duy trì các chỉ số hiệu suất tốt.