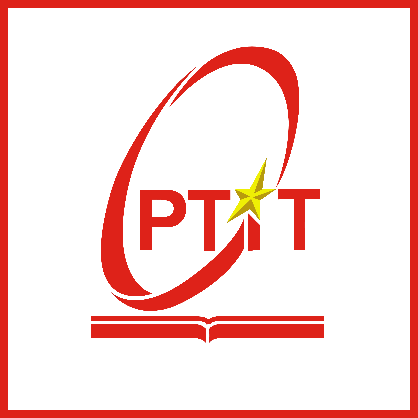
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA AN TOÀN THÔNG TIN**

**--------------------------------**

****

**BÁO CÁO BÀI TẬP LỚN PYTHON**

HỌC MÁY CHO PHÁT HIỆN MÃ ĐỘC ANDROID

|  |  |
| --- | --- |
| Giảng viên | : Vũ Minh Mạnh |
| Nhóm | : Lớp 01 Nhóm 02 |
| Đề tài | : 2 |

Thành viên:

|  |  |
| --- | --- |
| Nguyễn Duy Đạt | B21DCAT056 |

**Hà Nội – tháng 5 năm 2024**

Mục lục

[**I.** **Giới thiệu chung** 4](#_Toc176101220)

[**II.** **Cơ sở lý thuyết** 4](#_Toc176101221)

[1. Xây dựng FCG 5](#_Toc176101222)

[1.1. Tỉa đồ thị 6](#_Toc176101223)

[2. Mô hình Infercode 9](#_Toc176101224)

[2.1. Giới thiệu 9](#_Toc176101225)

[2.2. Infercode hoạt động như thế nào 9](#_Toc176101226)

[2.3. Ứng dụng mô hình sau đào tạo 11](#_Toc176101227)

[3. Graph Isomorphism Network 18](#_Toc176101228)

[3.1. Giới thiệu về bài toán đồ thị và GNN 18](#_Toc176101229)

[3.2. GIN- Graph Isomorphism Network 20](#_Toc176101230)

[3.3. GIN học biểu diễn đồ thị như thế nào 21](#_Toc176101231)

[3.4. GIN mang lại những gì mới 23](#_Toc176101232)

[4. Random Forest 25](#_Toc176101233)

[4.1. Random Forest là gì 25](#_Toc176101234)

[4.2. Đi sâu vào Random forest 29](#_Toc176101235)

[**III.** **Tổng kết** 32](#_Toc176101236)

[1. Đánh giá mô hình 32](#_Toc176101237)

[**IV.** **Danh mục tham khảo** 33](#_Toc176101238)

Nhận xét của giảng viên

Lời mở đầu

Kính gửi thầy Vũ Minh Mạnh,

Đầu tiên chúng em xin gửi lời cảm ơn chân thành tới thầy vì đã hướng dẫn, hỗ trợ và tạo điều kiện tốt nhất cho chúng em trong quá trình thực hiện đề tài nghiên cứu này.

Vì đây là một lĩnh vực rất mới và lần đầu tiên chúng em được tiếp xúc nên còn nhiều bỡ ngỡ, được thầy tạo điều kiện là một cơ hội rất quý giá để chúng em có thể tìm hiểu và áp dụng những kiến thức mới mẻ này vào đề tài, đồng thời cũng là dịp để chúng em rèn luyện kĩ năng làm việc nhóm và giải quyết vấn đề. Dù vậy thì đây vẫn là lần đầu chúng em tiếp xúc với đề tài mới lạ liên quan đến học máy, học sâu này nên trong quá trình thực hiện đề tài không thể tránh khỏi những sai sót, rất mong thầy thông cảm và đưa ra lời khuyên để chúng em có thể hoàn thiện kĩ năng cho những đề tài tiếp theo trong tương lai.

Dưới đây là báo cáo chi tiết về đề tài và kết quả nghiên cứu. Chúng em rất mong nhận được sự phản hồi và đánh giá của thầy, chúng em xin chân thành cảm ơn.

1. **Giới thiệu chung**
2. Về đề tài nghiên cứu

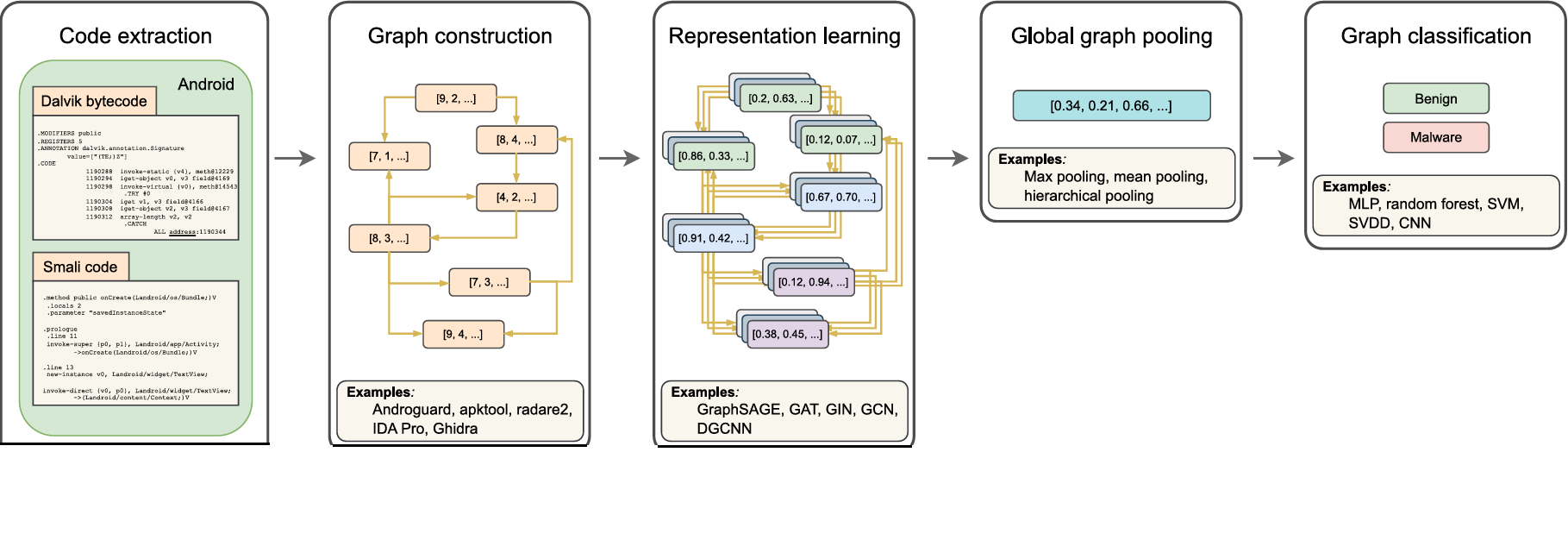
Các thiết bị điện thoại thông minh được kết nối Internet đóng một vai trò quan trọng trong miền ứng dụng Internet of Things. Các thiết bị này đang được sử dụng rộng rãi cho các hoạt động hàng ngày như điều khiển từ xa hệ thống chiếu sáng và sưởi ấm tại nhà, trả tiền đỗ xe và gần đây là thanh toán hàng hóa bằng thông tin thẻ tín dụng đã lưu bằng kết nối không dây trong phạm vi tầm gần. Android là nền tảng điện thoại thông minh phổ biến nhất hiện nay. Đây cũng là sự lựa chọn của các tác giả phần mềm độc hại để lấy dữ liệu an toàn và riêng tư.Phần mềm độc hại đặt ra mối đe dọa đầy thách thức đối với thế giới giao tiếp kỹ thuật số này vì nó có thể thao túng hoặc thực hiện bất kỳ hoạt động gây trở ngại nào bên trong máy tính bằng cách làm hỏng các tệp quan trọng và vô hiệu hóa hệ thống mạng bằng các cuộc tấn công độc hại. Vì những kẻ viết phần mềm độc hại đã phát triển kỹ thuật thiết kế phần mềm độc hại trong thập kỷ qua nên cơ chế phát hiện phải đủ mạnh để xác định phần mềm độc hại mới nhất và cũng sẽ hạn chế sự lây lan của nó trong điện thoại. Truyền thống. phương pháp phát hiện phần mềm độc hại dựa trên chữ ký. các phương pháp này không phát hiện được phần mềm độc hại phức tạp mới vì các nhà phát triển hiện đang phát triển các biến thể của chúng và ngụy trang phần mềm độc hại. Do đó, phần mềm độc hại không bị phát hiện bằng các kỹ thuật truyền thống. Vì vậy, rất cần một hệ thống phát hiện phần mềm độc hại hiệu quả và hiệu quả có thể xác định và phát hiện phần mềm độc hại bị che khuất và mã hóa với độ chính xác tối đa để doanh nghiệp lớn hoặc bất kỳ tổ chức nào khác dựa vào trao đổi thông tin kỹ thuật số không nên duy trì bất kỳ phần mềm độc hại nào. loại tổn thất về tài chính và thông tin. Việc sử dụng máy học trong việc phát hiện phần mềm độc hại đã được chứng minh là có hiệu quả trong việc phát hiện phần mềm độc hại ẩn và bị che khuất. Nghiên cứu này tập trung vào đánh giá nhanh chóng và tập trung khả năng phát hiện phần mềm độc hại bằng cách sử dụng một số phương pháp học máy.

Cụ thể về đề tài này, chúng ta sẽ sử dụng Androguard để dịch ngược mã nguồn APK, sau đó sẽ xây dựng đồ thị fcg, kế đến sẽ làm giàu thông tin cho đồ thị bằng Infercode, rồi cho vào mô hình GIN để học biểu diễn đồ thị, cuối cùng sẽ cần đến mô hình Random forest để phân loại mã độc

Để hiểu hơn về đề tài nghiên cứu này, ta sẽ cần đi qua cơ sở lý thuyết để hình dung ta cần làm những gì để có thế mô hình hóa bài toán này.

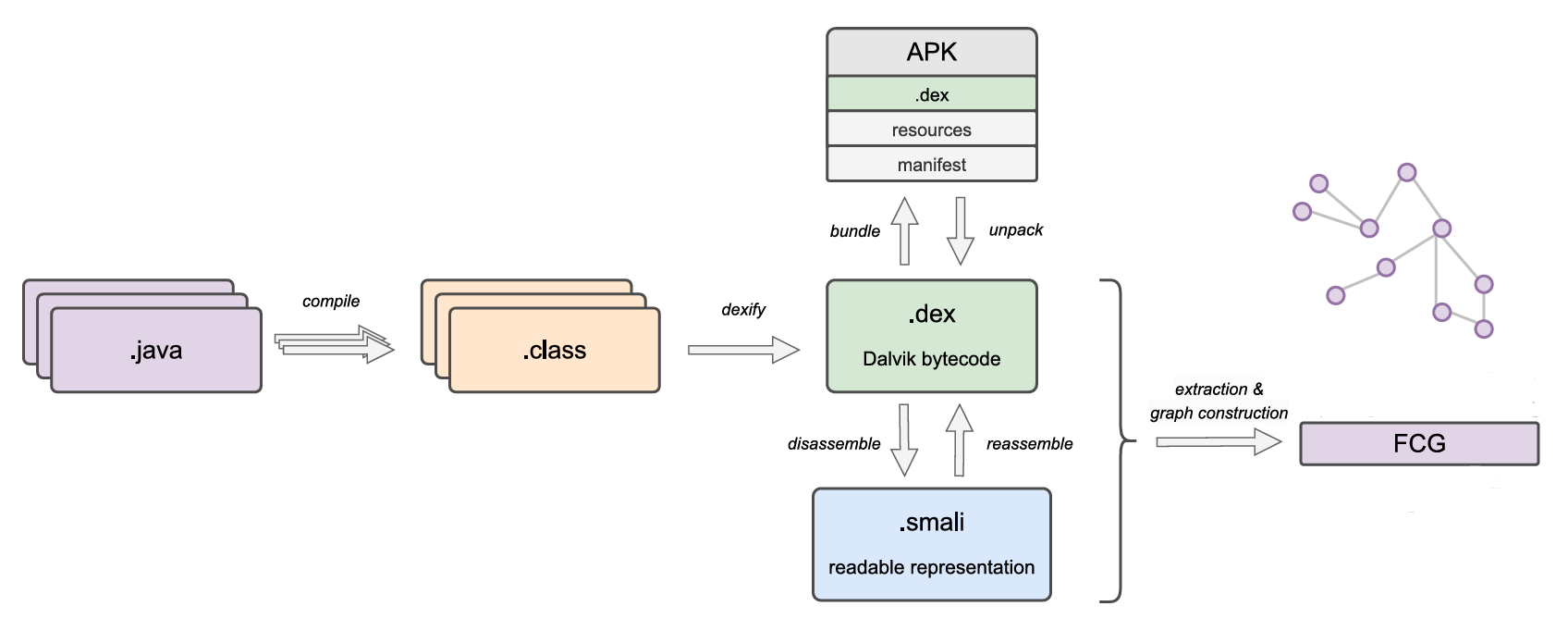
1. **Cơ sở lý thuyết**

Bước đầu tiên, xây dựng đồ thị gọi hàm FCG từ file APK được sử dụng nhằm duy trì ngữ nghĩa của chương trình. Bước đầu tiên này được thực hiện bằng cách sử dụng công cụ Androguard. Tiếp theo sẽ xây dựng đồ thị gọi hàm tăng cường Enhanced FCG bằng cách gán cho mỗi nút một vector đặc trưng. Ở đây, ta tạo ra vector đặc trưng cho các nút bằng cách nhúng hàm thành vector. Sau đó, kỹ thuật học biểu diễn đồ thị GIN được sử dụng để tạo ra một vectơ nhúng đồ thị có kích thước cố định đóng gói tất cả thông tin của đồ thị. Sau đó, vectơ cuối cùng được phân loại bằng phương pháp Random Forest



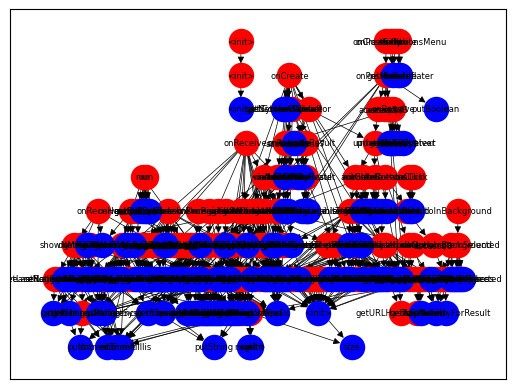
1. Xây dựng FCG

Androguard là một công cụ Python hoàn chỉnh được sử dụng để phân tích với các tệp Android và được sử dụng để trích xuất FCG trong bài này. Nó có thể dịch ngược tệp APK và lấy các hàm trong lớp. Bằng cách phân tích các lệnh được gọi trong mỗi hàm, nó lấy các hàm gọi và được gọi làm nút, thêm các cạnh có hướng theo mối quan hệ gọi và sau đó xây dựng đồ thị gọi hàm có hướng. Đồ thị gọi hàm có thể tự động nắm bắt các đặc điểm hành vi, khám phá mối quan hệ của chúng thông qua mối quan hệ call-callee giữa các hàm khác nhau.



Trong FCG, mỗi nút đại diện cho một hàm và mỗi cạnh biểu thị một lệnh gọi hàm. Các nút của FCG được chia thành các nút internal và các nút external. Một hàm internal có khả năng triển khai bên trong ứng dụng và có thể được dịch ngược để truy cập các lệnh code. Hàm external là một phương thức có khai báo nhưng không có mã cụ thể trong ứng dụng Android. Tên của các hàm bên ngoài thường không thay đổi và được sử dụng nhiều lần trong các ứng dụng khác nhau.

Khi xây dựng đồ thị gọi hàm tăng cường Enhanced FCG gặp phải một vấn đề là các hàm external không có code để nhúng thành vector. Trong bài này, ta giải quyết vấn đề này bằng cách coi tên của hàm external là code và vẫn nhúng thành vector như bình thường.



Hình vẽ mô tả fcg của file apk trước khi tỉa các node

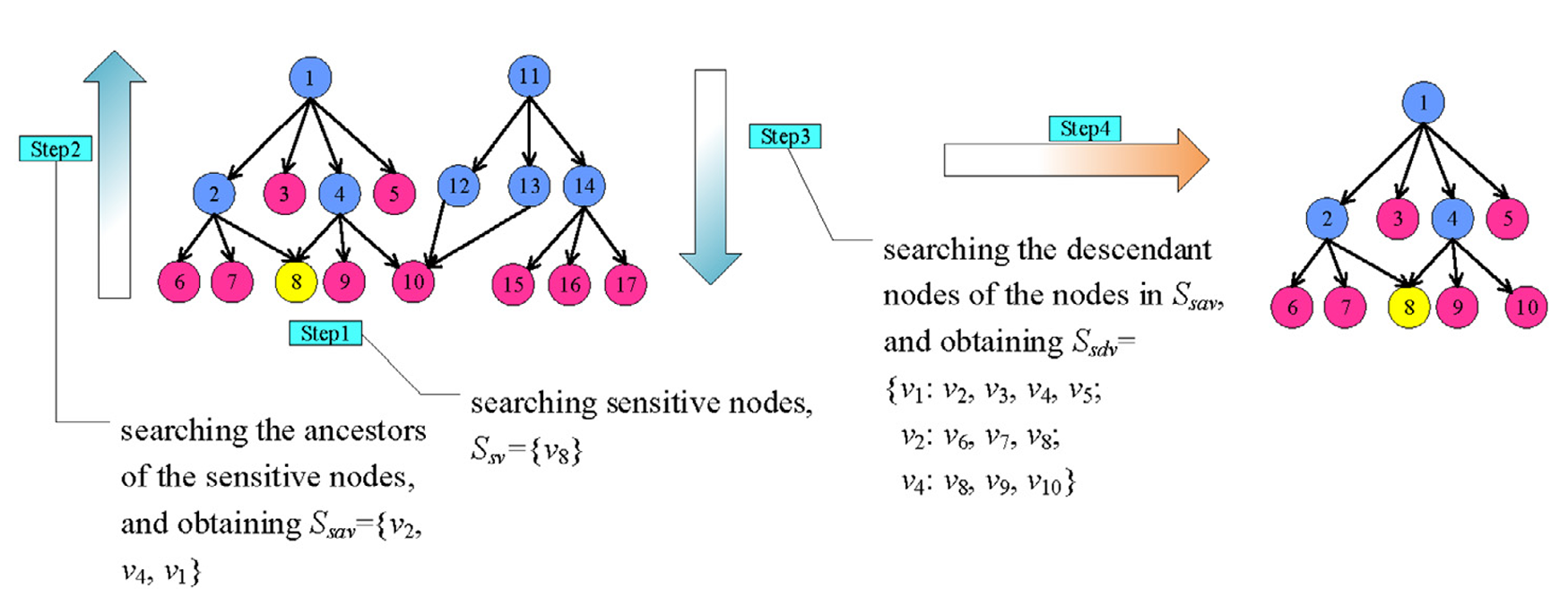
* 1. Tỉa đồ thị

FCG của ứng dụng Android thường có hàng trăm nghìn nút, điều này gây khó khăn cho việc đánh giá mức độ tương tự của FCG. Làm thế nào để tận dụng tối đa thông tin cấu trúc để tạo ra vectơ đặc trưng đại diện trong thời gian hợp lý là một vấn đề cần phải giải quyết.

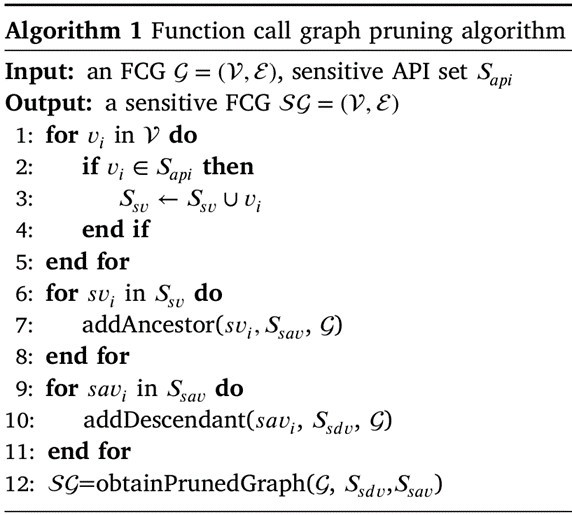
Chúng ta đề xuất một phương pháp cắt tỉa đồ thị để đơn giản hóa FCG đồng thời vẫn trì bối cảnh của các lệnh gọi API nhạy cảm (tức là các lệnh gọi API liên quan đến bảo mật yêu cầu các quyền cụ thể) trong khi loại bỏ các nút không cần thiết. đồ thị sau khi tỉa khiến mô hình chú ý hơn đến hành vi độc hại có liên quan đến các API nhạy cảm, giảm mức tiêu thụ tài nguyên đào tạo .

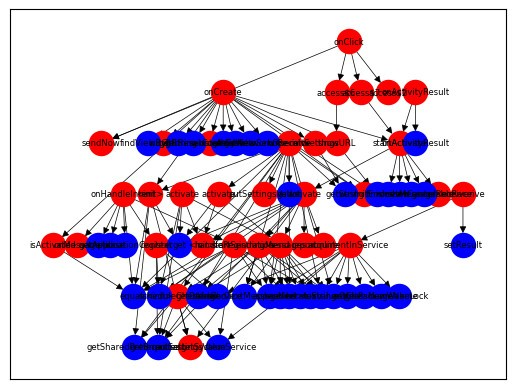
Trong bài này thực hiện tỉa đồ thị sử dụng thuật toán trong bài báo:

[“SeGDroid:An Android malware detection method based on sensitive function call graph learning”](#tỉađồthị)



Các API nhạy cảm được sử dụng để xây dựng FCG nhạy cảm dựa trên ánh xạ của API và quyền.





Đồ thị fcg ta thu được sau khi tỉa đồ thị

1. Mô hình Infercode
   1. Giới thiệu

Infercode là một mô hình học máy tự giám sát(Self-Supervised), được dùng để học cách biểu diễn mã nguồn bằng cách dự đoán qua các nhánh con của cú pháp cây trừu tượng AST

Điểm mới lạ nằm ở việc đào tạo các biểu diễn mã bằng cách dự đoán các cây con được xác định tự động từ ngữ cảnh của AST. Với Mã suy luận, các cây con trong AST được coi là nhãn để đào tạo các biểu diễn mã mà không cần bất kỳ nỗ lực gắn nhãn nào của con người hoặc chi phí xây dựng biểu đồ đắt tiền và các biểu diễn được đào tạo không còn bị ràng buộc với bất kỳ nhiệm vụ hoặc đơn vị mã cụ thể nào nữa.

Các nhà nghiên cứu đã đào tạo một phiên bản của mô hình InferCode bằng cách sử dụng Mạng thần kinh chuyển đổi dựa trên cây (TBCNN) làm bộ mã hóa của một bộ mã Java lớn. Sau đó, mô hình được đào tạo trước này có thể được áp dụng cho các tác vụ không được giám sát xuôi dòng như phân cụm mã, phát hiện bản sao mã, tìm kiếm mã đa ngôn ngữ hoặc được sử dụng lại theo sơ đồ học chuyển giao để tiếp tục đào tạo trọng số mô hình cho các tác vụ được giám sát như phân loại mã và dự đoán tên phương thức So sánh với các kỹ thuật trước đây được áp dụng cho cùng các tác vụ xuôi dòng, chẳng hạn như code2vec, code2seq, ASTNN, việc sử dụng mô hình InferCode được đào tạo trước đạt được kết quả hiệu suất cao hơn với biên độ đáng kể cho hầu hết các tác vụ, bao gồm cả các tác vụ liên quan đến các ngôn ngữ lập trình khác nhau.

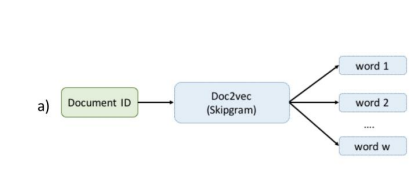
* 1. Infercode hoạt động như thế nào

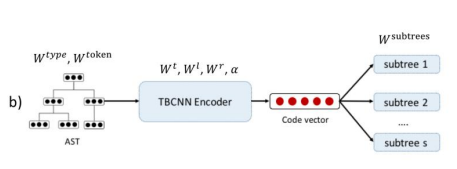
Mục đích của Infercode là phát triển một kĩ thuật mới để học biểu diễn mã và nó có thể huấn luyện được mà không cần bất kỳ nhãn nào của con người, linh hoạt trong việc tạo các phần nhúng cho bất kỳ đơn vị mã nào có thể được phân tích cú pháp thành các cây cú pháp và đủ tổng quát để các biểu diễn mã có thể hoạt động tốt cho các tác vụ tiếp theo

Dựa trên ý tưởng đó, Infercode được sinh ra là một kỹ thuật học tự giám sát cho mã nguồn bằng cách dự đoán các cây con cú pháp. Để các cách biểu diễn được đào tạo như vậy nắm bắt được ý nghĩa mã, là các đoạn mã có cùng ngữ nghĩa phải bao gồm một số thành phần mã tương tự về mặt cú pháp. Mặc dù hai đoạn mã triển khai cùng một chức năng có thể rất khác nhau về mặt cú pháp, nhưng vẫn có thể có một số phần tử chi tiết trong mã hoặc các đoạn mã khác sử dụng hai đoạn mã này có cú pháp tương tự nhau khi cơ sở mã lớn.

InferCode có thể hoạt động như một bộ mã hóa ánh xạ bất kỳ đoạn mã có thể phân tích cú pháp nào vào một biểu diễn vectơ (nhúng) và vectơ này có thể được sử dụng cho nhiều tác vụ tiếp theo khác nhau, chẳng hạn như phân cụm mã, phát hiện bản sao và tìm kiếm mã. InferCode có thể phục vụ như một mô hình được đào tạo trước và các trọng số của nó có thể được sử dụng lại trong quá trình đào tạo mô hình ở cấp độ tiếp theo cho các nhiệm vụ học có giám sát, điều này có thể tăng tốc quá trình đào tạo và giảm bớt vấn đề thiếu dữ liệu cho một nhiệm vụ cụ thể. Các nhà khoa học triển khai InferCode dựa trên AST do SrcML tạo ra. Nó cung cấp từ vựng kết hợp của các loại nút AST cho nhiều ngôn ngữ (ví dụ: Java, C, C++, C#), ngụ ý rằng InferCode có thể là polyglot, tạo ra các biểu diễn mã phù hợp cho các tác vụ liên quan đến các ngôn ngữ khác nhau, chẳng hạn như tìm kiếm mã đa ngôn ngữ , miễn là SrcML có thể nhận dạng được AST của đoạn mã

Mục tiêu của việc học tự giám sát là huấn luyện bộ mã hóa E sao cho E có thể ánh xạ một đối tượng thành biểu diễn vectơ (nhúng). Trong trường hợp của chúng tôi, phần nhúng v dành cho biểu diễn AST T của đoạn mã C. Huấn luyện bộ mã hóa E là tìm hiểu các tham số (hoặc trọng số) của nó để E có thể tạo ra các phần nhúng cho đoạn mã sao cho các vectơ cho các đoạn mã có thông tin cú pháp và ngữ nghĩa tương tự nhau sẽ nằm gần nhau trong không gian vectơ.





Hình trên trình bày về phương pháp tiếp cận InferCode tương tự như Doc2vec bằng cách coi toàn bộ AST là một tài liệu và coi các cây con là các từ. Nhưng khác với doc2vec, InferCode không truy vấn trực tiếp các vectơ nhúng từ ma trận nhúng cho toàn bộ tài liệu; thay vào đó, trước tiên cần mã hóa toàn bộ AST để lấy vector , sau đó sử dụng nó để dự đoán các cây con. Tổng thể sẽ được chia làm 3 bước:

Bước 1: Với mỗi AST trích xuất được từ code đầu vào chúng ta sẽ trích suất các cây con và cho chúng vào một từ điển

Bước 2: Sử dụng TBCNN để nhúng các AST thành vector

Bước 3: Sử dụng các vector để dự đoán các cây con trong từ điển

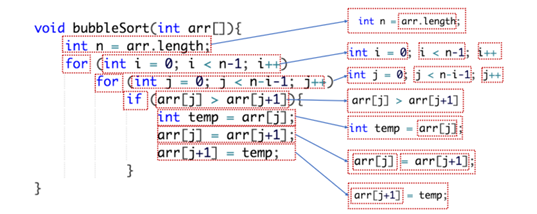
Sau khi trải qua 3 bước thì mô hình có thể chuyển đổi các AST ứng với đoạn code thành các vector với ngữ nghĩa gần nhất, và chúng ta có thể dùng nó cho các nhiệm vụ khác tiếp theo, cụ thể:

Trích xuất các subtrees và thêm vào tập từ điển các cây con

Bằng cách duyệt qua các nodes của cây các cây AST, với mỗi cây AST đi qua: mỗi node nếu thõa mãn điều kiện được đặt ra thì tại node dó sẽ tạo ra một cây con với gốc là chính tại node đó

Với các từ đơn như là if, for, while chúng ta sẽ xem xét chúng như một cây con với kích thước là 1 chứ không lấy toàn bộ cây con trong if, for, while đó vì những cây con đó có kích thước quá lớn khiến cho các cây con không đủ tinh giản để thể hiện về mặt ngữ nghĩa và đồng thời khiến cho encoder rất khó để nhận biết sự tương đồng về mặt ngữ nghĩa của nó

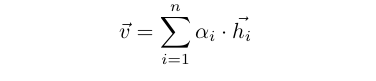
Cứ lần lượt như vậy cho đến hết các AST ta sẽ thu được một từ điển chứa các toàn bộ các cây con được trích từ các cây AST của tập dữ liệu



Chuyển AST vào TBCNN để tạo ra vector v của AST đó

Học biểu diễn các nodes: từ cây AST, mô hình sẽ duyệt qua các nodes từ dưới lên trên, và với mỗi node cha sẽ tích lũy các thông tin từ các node hậu duệ của nó. Sau khi đi qua hết các node, mỗi nodes sẽ chứa thông tin của các node hậu duệ

Tổng hợp biểu diễn các node: Vì mục tiêu là muốn ánh xã đoạn code thành một vector biểu diễn nên ta cần phải tổng hợp các vector node thành một vector biểu diễn duy nhất cho đoạn code. Trong bước này ta sẽ sử dụng thêm trọng số để tập trung vào các node quan trọng tránh cho mất mát thông tin, vector tổng hợp cuối cùng được tính bằng công thức dưới



Ai: là trọng số được khởi tạo ngẫu nhiên ban đầu và được học dần theo mô hình

Hi: là các vector trạng thái của mỗi node

* 1. Ứng dụng mô hình sau đào tạo

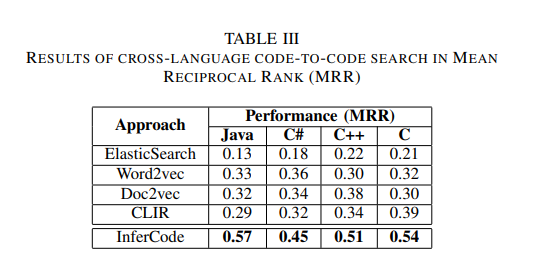
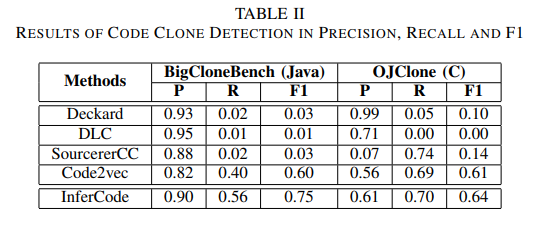
Các nhà nghiên cứu đã trình bày quy trình đào tạo InferCode bằng cách dự đoán các cây con làm nhãn. Lưu ý rằng trong việc học tự giám sát, người ta thường không quan tâm đến việc thực hiện nhiệm vụ giả định. Thay vào đó họ quan tâm đến các trọng số đã được học và khả năng của mô hình trong việc tạo ra các phần nhúng. Bộ mã hóa TBCNN đã được huấn luyện của InferCode có thể được sử dụng để tạo vectơ nhúng v cho bất kỳ đoạn mã có thể phân tích cú pháp nào bằng cách phân tích mã thành AST và cung cấp AST thông qua bước mã hóa để lấy vectơ. Các trọng số trong mô hình được đào tạo cũng có thể được sử dụng cho các mô hình dự đoán trong các nhiệm vụ học có giám sát ở hạ nguồn để tiết kiệm chi phí đào tạo và có khả năng cải thiện độ chính xác dự đoán của chúng.

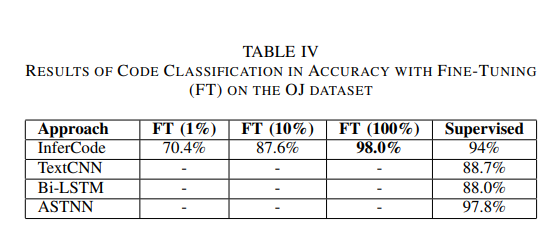
Infercode có thể áp dụng vào việc vector nhứng mã cho các tác vụ không giám sát: phân cụm mã, phát hiện bản sao mã, tìm kiếm mã theo mã qua ngôn ngữ; áp dụng vào tinh chỉnh các nhiệm vụ học tập có giám sát.

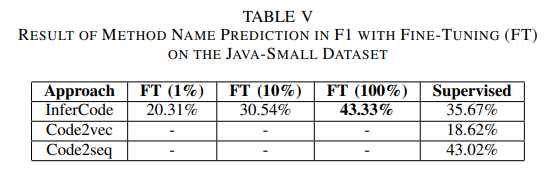
Chúng ta sẽ không đi quá sâu vào việc tìm hiểu quy trình mà tới ngay với đánh giá thực nghiệm để so sánh mô hình Infercode với các model học máy khác. Để huấn luyện mô hình của mình, các nhà nghiên cứu đã sử dụng lại tập dữ liệu Java-Large đã được sử dụng trong Code2vec và Code2seq. Tập dữ liệu này chứa một số lượng lớn các dự án Java được thu thập từ Github (4 triệu tệp). Chúng tôi phân tích tất cả các tệp thành AST bằng SrcML. Sau đó, chúng ta xác định tất cả các cây con để hình thành từ vựng về các cây con. Có AST và các cây con làm nhãn giả, chúng tôi huấn luyện mô hình InferCode bằng cách sử dụng entropy chéo softmax làm hàm mất mục tiêu và chọn Adam làm trình tối ưu hóa với tốc độ học ban đầu là 0,001 trên Nvidia Tesla P100 GPU.

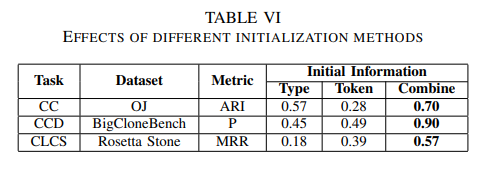
Các bảng tổng hợp hiệu suất dưới đây là minh chứng rõ ràng nhất cho mô hình Infercode.

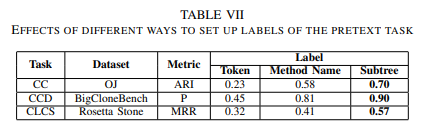




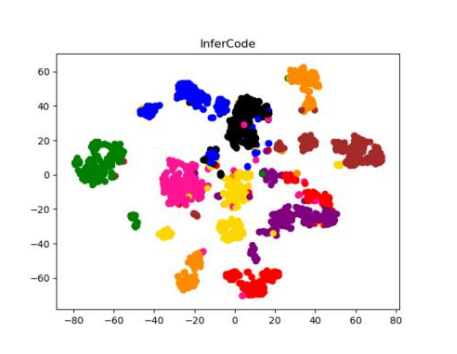


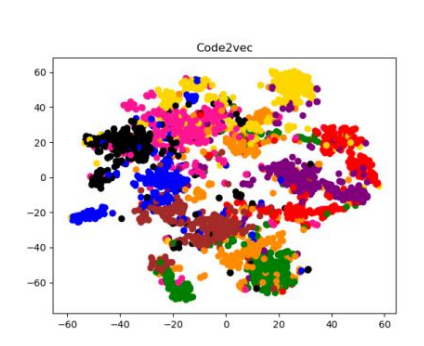


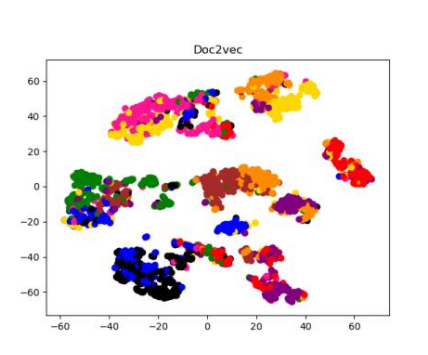




Có thể thấy rất rõ hiệu suất của infercode so với các mô hình học máy khác. Còn dưới đây là trực quan hóa các vector mã của các chương trình từ 9 lớp trong bộ liệu OJ của Infercode, code2vec và doc2vec.



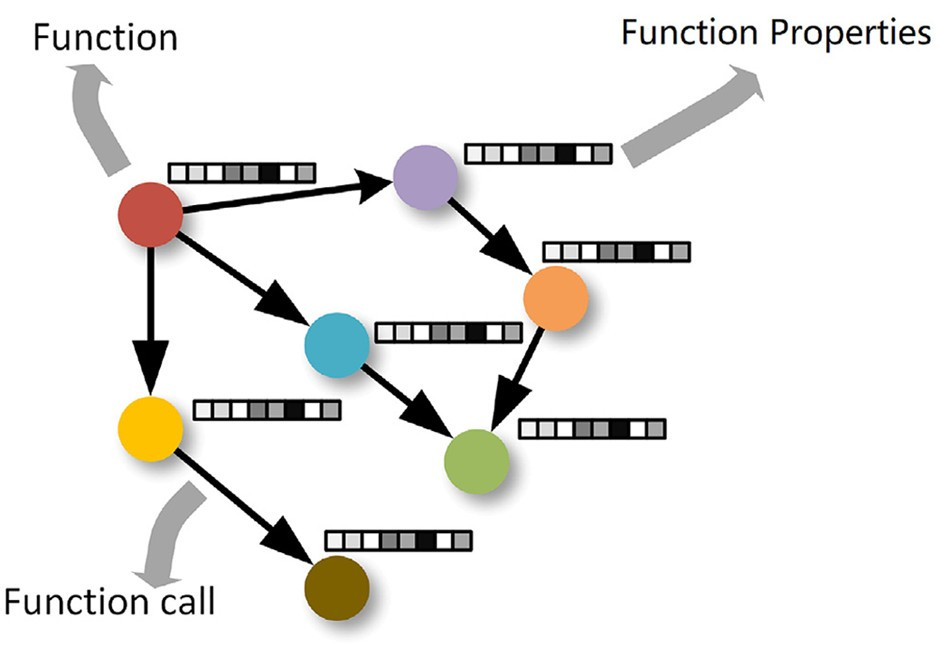




Các nhà nghiên cứu chọn Mạng thần kinh chuyển đổi dựa trên cây (TBCNN) vì khả năng nắm bắt cấu trúc các tính năng của mã nằm trong AST và sự sửa đổi mà chúng tôi

TBCNN được thiết kế để nhận AST trực tiếp với nỗ lực kỹ thuật tối thiểu để xử lý nó. AST tương đối dễ dàng được sản xuất chính xác cho hầu hết các ngôn ngữ lập trình dựa trên ngữ pháp của chúng, do đó, việc xây dựng mô hình học tập dựa trên cây dựa trên AST ngụ ý rằng chúng ta có thể có một mô hình dễ khái quát hóa hơn trên các ngôn ngữ, đó là lợi thế khi chọn cây- dựa trên các mô hình khác. Nhưng nói thế không có nghĩa là các mô hình khác không thực hiện tốt tất cả các nhiệm vụ học mã; chúng vẫn có thể hoạt động tốt khi dữ liệu và thời gian đào tạo được sử dụng đặc biệt và chúng có thể được sử dụng cùng nhau làm bộ mã hóa trong khung học tập tự giám sát để cải thiện hiệu suất cho các nhiệm vụ khác nhau hơn nữa. Chúng ta sẽ cần thời gian để khám phá những thú vị trong tương lai.

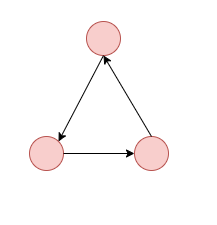
Sau khi nhúng code từng hàm cho từng node của đồ thị fcg thì ta sẽ thu được 1 đồ thị fcg và chứa các vector đặc trưng cho từng node



1. Graph Isomorphism Network
   1. Giới thiệu về bài toán đồ thị và GNN

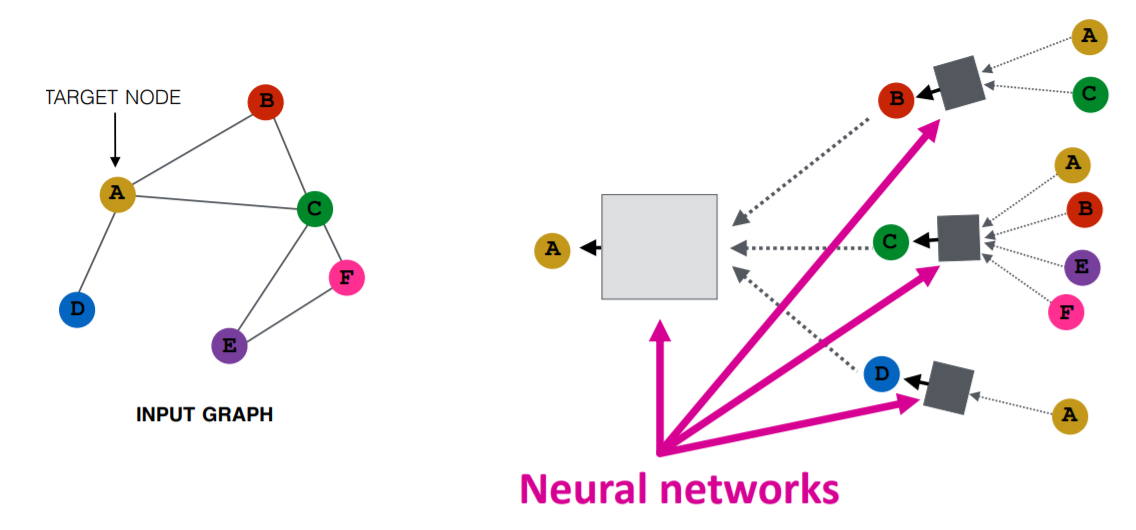
Các bài toán và dữ liệu về đồ thị là những khái niệm mà ai đã và đang muốn tìm hiểu về học sâu và học máy cũng đã từng nghe qua. Nhưng để thực sự hiểu và triển khai được các bài toán trên dữ liệu đồ thị thì rất khó khăn. Vậy nên việc hiểu về cách mà các mô hình học sâu được xây dựng trên dữ liệu đồ thị hoạt động như thế nào và triển khai một bài toán đơn giản trên dữ liệu đồ thị sao cho hiệu quả lần rất quan trọng và cần thiết. Dữ liệu đồ thị đã trở thành một phần quan trọng trong lĩnh vực máy học và khai phá dữ liệu, với khả năng biểu diễn và phân tích các mối quan hệ phức tạp giữa các đối tượng. Đồ thị có thể mô tả mạng xã hội, mạng lưới giao thông, sự tương tác giữa các phân tử trong hóa học, và nhiều hình thái khác. Bằng cách sử dụng các đỉnh (nút) và các cạnh (liên kết), đồ thị cung cấp khung cho việc phân tích mô hình, dự đoán và hiểu các quy luật ẩn trong dữ liệu. Trong quá trình phát triển AI hiện nay, việc hiểu và làm việc với dữ liệu đồ thị trở thành một kỹ năng quan trọng cho các nhà nghiên cứu và nhà phát triển

Graph hay đồ thị là những cái tên thị là loại cấu trúc dữ liệu chứa các nút và cạnh. Một nút có thể là một người, địa điểm hoặc vật thể, và các cạnh xác định mối quan hệ giữa các nút. Các cạnh có thể có hướng và không hướng dựa trên các phụ thuộc tương ứng. Trong ví dụ dưới đây, các hình tròn màu đỏ là các nút và các mũi tên là các cạnh. Hướng của các cạnh xác định các phụ thuộc giữa hai nút.

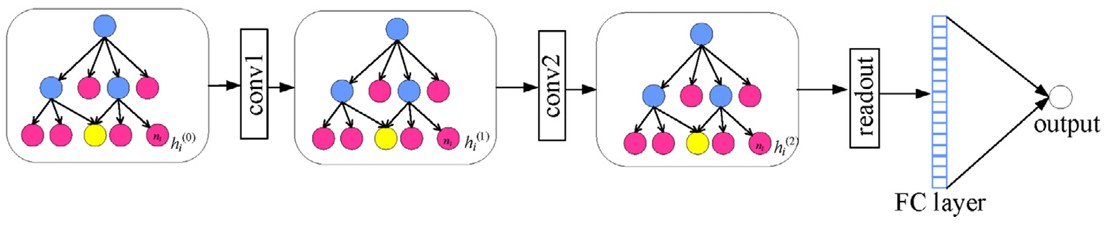


Mạng Nơ-ron Đồ Thị (Graph Neural Network - GNN) là một loại mô hình học máy được thiết kế đặc biệt để làm việc với dữ liệu đồ thị. GNN có khả năng mở rộng và áp dụng trên các đồ thị có cấu trúc phức tạp, như mạng xã hội, mạng lưới giao thông, hay bất kỳ hệ thống nào có mối quan hệ giữa các đối tượng. GNN hoạt động bằng cách truyền thông tin qua các đỉnh và cạnh trong đồ thị. Mô hình học thông qua việc cập nhật và kết hợp thông tin từ các hàng xóm của mỗi đỉnh, cho phép nắm bắt thông tin cấu trúc và tương tác giữa các đối tượng trong đồ thị. Một trong những đặc điểm đáng chú ý của GNN là khả năng tích hợp thông tin từ cả đặc trưng của các đỉnh và cấu trúc đồ thị. Điều này cho phép GNN học mô hình phức tạp và biểu diễn các mối quan hệ phức tạp giữa các đối tượng trong đồ thị. GNN đã chứng tỏ được hiệu quả trong nhiều nhiệm vụ, bao gồm phân loại đồ thị, phân loại nút, dự đoán liên kết và nhúng đồ thị. Các ứng dụng của GNN rất đa dạng, từ phân tích mạng xã hội, gợi ý người dùng, cho đến phát hiện và kiểm soát các hiện tượng trong các hệ thống phức tạp.

Đồ thị đầu vào được đi qua một loạt mạng neural. Cấu trúc đồ thị đầu vào được chuyển đổi thành nhúng đồ thị, cho phép chúng ta duy trì thông tin về các nút, cạnh và ngữ cảnh toàn cục. Sau đó, vectơ đặc trưng của các nút A và C được thông qua lớp mạng neural. Nó tổng hợp những đặc trưng này và truyền chúng vào lớp tiếp theo.



Sau một số k lần lặp thì một nút được biểu thị bằng 1 vector đặc trưng đã biến đổi của nó, vector này thu thập thông tin cấu trúc trong vùng lân cận. Sau đó có thể thu được biểu diễn của toàn bộ biểu đồ thông qua việc gộp chung, chẳng hạn như tính tổng các vector biểu diễn của tất cả các nút trong biểu đồ.

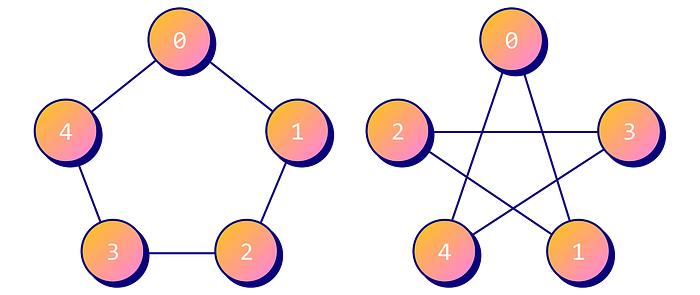


Có nhiều biến thể của GNN với các sơ đồ tổng hợp vùng lân cận và tổng hợp cấp đồ thị khác nhau đã được phát triển để làm việc với dữ liệu đồ thị như GCN, GAT, GraphSAGE, GAE, GRNN…

Tuy nhiên, thiết kế của GNN mới chủ yếu dựa trên trực giác thực nghiệm, phương pháp phỏng đoán và thử nghiệm và sai sót. Có rất ít hiểu biết về mặt lý thuyết về các đặc tính và hạn chế của GNN và phân tích chính thức về khả năng biểu diễn của GNN còn hạn chế

* 1. GIN- Graph Isomorphism Network

Mạng đẳng cấu đồ thị GIN là một thuật toán học máy mới được thiết kế cho các nhà khoa học dữ liệu, nhà phát triển hệ thống trí tuệ nhân tạo và nhà nghiên cứu. GIN hoạt động bằng cách sử dụng một khái niệm toán học được gọi là đẳng cấu đồ thị. Về cơ bản, đó là thuật toán so sánh cấu trúc của các biểu đồ khác nhau để xem chúng giống hay khác nhau. Khi làm như vậy, thuật toán có thể giúp xác định các yếu tố và mẫu khác nhau trong các tập dữ liệu lớn.



GIN đã được áp dụng rất nhiều trong thế giới khoa học dữ liệu và trí tuệ nhân tạo. Thuật toán này được ca ngợi là một trong những GNN có tính phân biệt cao nhất hiện có vì nó sử dụng một quy trình được gọi là thí nghiệm đẳng cấu đồ thị WL (Weisfeiler-Lehman)

Một trong những tính năng chính của GIN là khả năng khái quát hóa bài kiểm tra WL. Đây là một thuật toán phổ biến được sử dụng để phân tích dữ liệu đồ thị nhưng nó có một số hạn chế. GIN lấy khái niệm về WL và làm cho nó trở nên mạnh mẽ hơn bằng cách sử dụng nó để phân tích các tập dữ liệu lớn hơn, phức tạp hơn. Khi làm như vậy, thuật toán có thể xác định các mẫu và mối tương quan phức tạp hơn các thuật toán khác.

Thử nghiệm đẳng cấu đồ thị Weisfeiler-Lehman là một phương pháp quan trọng trong lĩnh vực xử lý dữ liệu đồ thị và nhận dạng mẫu. Phương pháp này được đặt tên theo hai nhà toán học: B. Weisfeiler và A. Lehman.

Cơ bản, thử nghiệm WL là một kỹ thuật nhóm các đỉnh trong đồ thị dựa trên các nhãn của các đỉnh và cạnh xung quanh của chúng. Quá trình này được thực hiện qua nhiều vòng lặp, trong mỗi vòng lặp, các nhãn của các đỉnh và cạnh được cập nhật dựa trên thông tin từ các hàng xóm của chúng. Cụ thể, trong mỗi vòng lặp của thử nghiệm WL, các đỉnh được nhóm lại dựa trên các chuỗi nhãn của các đỉnh và cạnh xung quanh của chúng. Sau đó, các chuỗi nhãn mới được tạo ra bằng cách kết hợp các chuỗi nhãn của các đỉnh trong cùng một nhóm. Quá trình này tiếp tục cho đến khi không có sự thay đổi nào trong việc nhóm các đỉnh.

Thử nghiệm đẳng cấu đồ thị Weisfeiler-Lehman thường được sử dụng để so sánh tính đồng đẳng của các đồ thị. Đặc biệt, nó là một công cụ quan trọng trong việc nhận biết đồ thị không đồng đẳng và phát hiện ra các mẫu cấu trúc trong dữ liệu đồ thị.

Đây là một thuật toán phổ biến được sử dụng trong các mô hình học máy dựa trên đồ thị và nó đã được sử dụng cho nhiều mục đích khác nhau. Tuy nhiên, GIN được thiết kế để đưa mọi thứ lên một tầm cao mới bằng cách cung cấp khả năng phân biệt đối xử tối đa giữa các GNN. Nói cách khác, thuật toán có thể được sử dụng để xác định các mẫu và yếu tố phức tạp hơn các thuật toán khác

* 1. GIN học biểu diễn đồ thị như thế nào

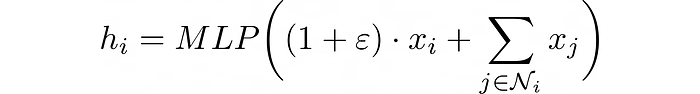
Vấn đề với các thuật toán học máy truyền thống là chúng không phù hợp để phân tích dữ liệu đồ thị. Điều này là do các biểu đồ không phù hợp với mô hình đầu vào/đầu ra tiêu chuẩn mà nhiều thuật toán trong số này được thiết kế để hoạt động. Mặt khác, GIN được thiết kế đặc biệt để làm việc với dữ liệu biểu đồ.

1. Quá trình Message passing

GIN tổng hợp vector node thông qua một quy trình truyền thông điệp gọi là message passing.

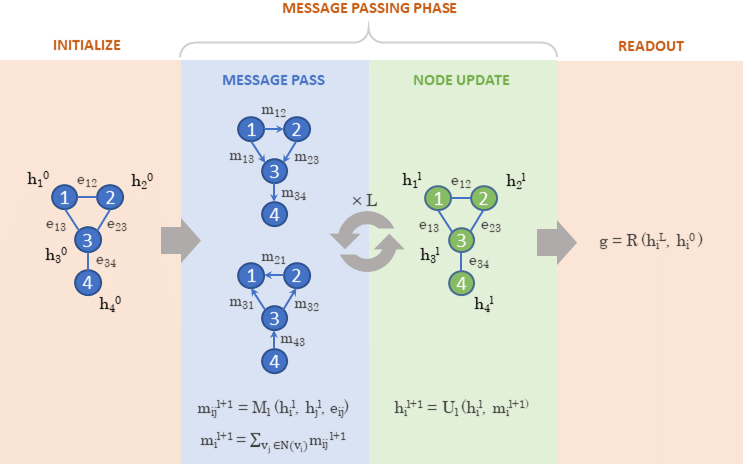
1. Đầu tiên, mỗi node trong đồ thị sẽ được khởi tạo với 1 vector đặc trưng
2. Message passing: trong mỗi vòng lặp, mỗi node sẽ gửi thông điệp đến các node hàng xóm. Thông điệp này sẽ thường là vector đặc trưng của node đó
3. Tổng hợp thông điệp: mỗi node sau đó sẽ tổng hợp các tin nhắn mà nó nhận được từ các node lân cận. Thường thì các node sẽ lấy tổng hoặc trung bình các vector thông điệp
4. Cập nhật vector đặc trưng: mỗi node sau đó sẽ cập nhật các vector đặc trưng của nó dựa trên các thông điệp đã tổng hợp. 1 hàm phi tuyến (như ReLU) thường sẽ được sử dụng để kết hợp vector đặc trưng hiện tại của node và các thông điệp tổng hợp

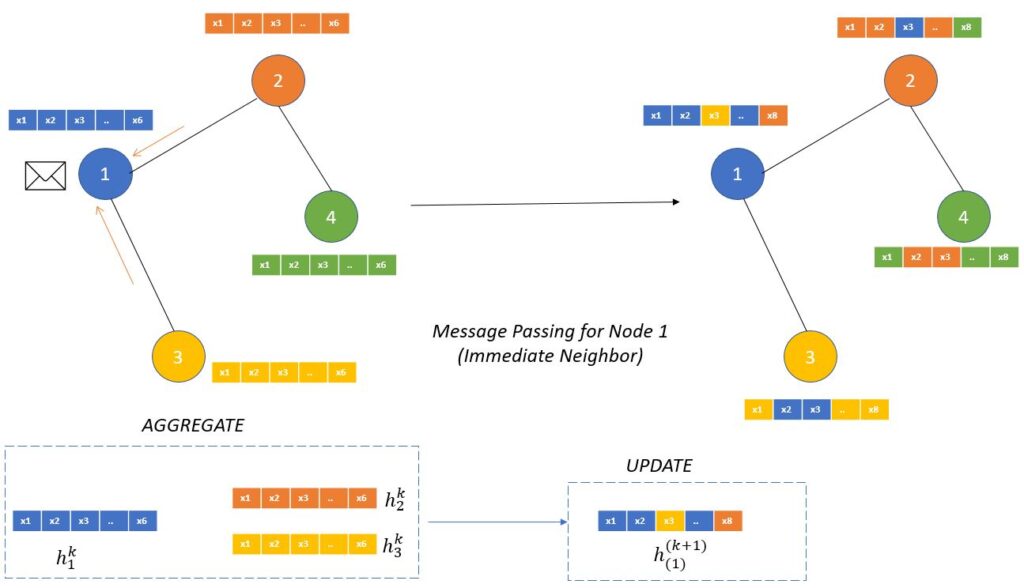
Cụ thể GIN sử dụng hàm MLP để tính vector cho mỗi node i:



MLP là Muliti-Layer Perceptron, được xây dựng từ nhiều lớp. Trong đó thì các thông tin di chuyển từ lớp này sang lớp khác theo 1 hướng duy nhất từ lớp đầu vào, qua các lớp ẩn và đến lớp đầu ra. Trong công thức này, ɛ xác định tầm quan trọng của nút mục tiêu so với các nút lân cận (nó có cùng tầm quan trọng nếu ɛ = 0). Nó có thể là một tham số có thể học được hoặc một đại lượng vô hướng cố định.

1. Quy trình trên sẽ được lặp 1 số k vòng nhất định để thu được vector đặc trưng cho node có giá trị nhất

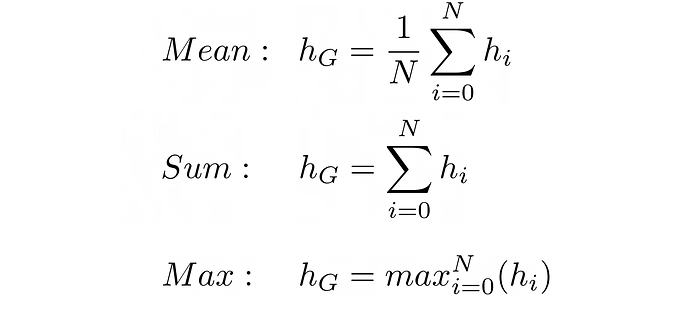




1. Quá trình Global pooling

Global pooling là 1 bước quan trọng trong quá trình tổng hợp thông tin từ nhiều node trong đồ thị để tạo ra 1 đại diện duy nhất tổng quát cho toàn bộ đồ thị

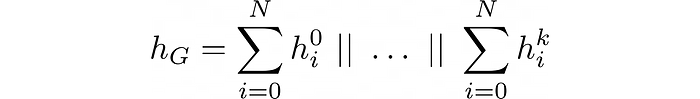
Cách đơn giản là sử dụng giá trị trung bình, tính tổng hoặc tìm giá trị lớn nhất của toàn bộ vector node



Các nhà nghiên cứu đã đưa ra hai điểm quan trọng về khả năng đọc ở cấp độ biểu đồ:

1. Để xem xét tất cả thông tin về cấu trúc, cần giữ lại các phần nhúng từ các lớp trước đó
2. Toán tử tổng có giá trị ấn tượng hơn nhiều so với giá trị trung bình và giá trị tối đa.

Những quan sát này khiến họ đề xuất phương pháp tổng hợp global pooling sau:



Đối với mỗi lớp, các node được tính tổng và kết quả được nối. Giải pháp này kết hợp giá trị đặc trưng của toán tử tổng với bộ nhớ của các lần lặp trước đó từ phép nối

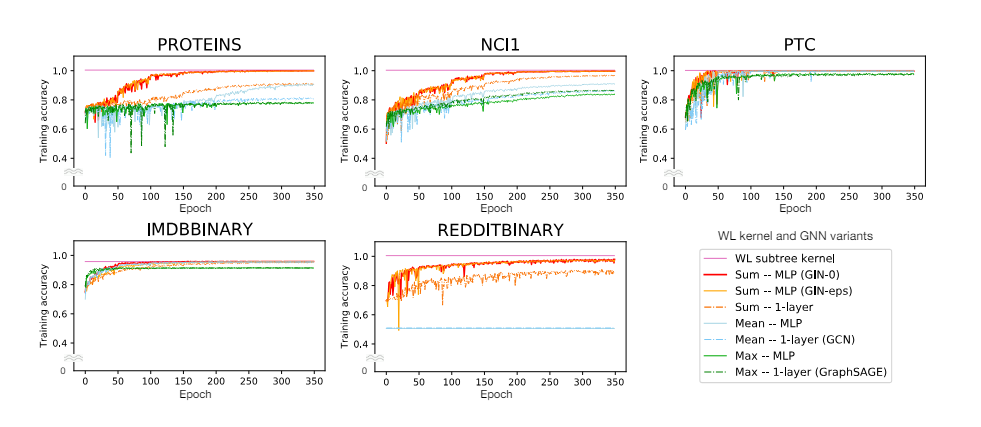
* 1. GIN mang lại những gì mới

GIN là một mạng đồ thị mới rất khả quan vì nó cung cấp một cách mới để phân tích và giải thích các tập dữ liệu. Các thuật toán học máy truyền thống bị hạn chế về khả năng phân tích dữ liệu biểu đồ, điều này có thể gây ra vấn đề khi xử lý các tập dữ liệu phức tạp. Gin đưa ra một cách tiếp cận mạnh mẽ hơn để phân tích biểu đồ bằng cách xác định các mẫu và mối tương quan phức tạp mà các thuật toán khác có thể bỏ sót.

GIN có thể được sử dụng để phân tích mạng xã hội nhằm xác định các nhóm người dùng có mối liên hệ chặt chẽ hơn với nhau. Điều này có thể cung cấp cái nhìn sâu sắc về cách thông tin lan truyền qua mạng xã hội và cách người dùng tương tác với nhau. Thuật toán cũng có thể được sử dụng để phân tích các loại biểu đồ khác, chẳng hạn như cấu trúc phân tử hoặc dữ liệu kết nối não.

Một lý do khác khiến GIN quan trọng là vì nó là một phần của xu hướng hướng tới phát triển các thuật toán học máy tiên tiến hơn. Khi có nhiều dữ liệu hơn, việc có các thuật toán có thể phân tích và giải thích dữ liệu này ngày càng trở nên quan trọng. Gin chỉ là một ví dụ trong số nhiều thuật toán mới đang được phát triển để đáp ứng nhu cầu này.

Các nhà nghiên cứu đã đánh giá và so sánh hiệu suất huấn luyện và kiểm tra của GIN với các biến thể GNN khác. Ở đây có 9 bộ dữ liệu đã được sử dụng gồm 4 bộ dữ liệu tin sinh học (MUTAG, PTC, NCI1, PROTEINS) và 5 bộ dữ liệu xã hội (COLLAB, IMDB\_BINARY, IMDB\_MULTI, REDDIT\_BINARY)



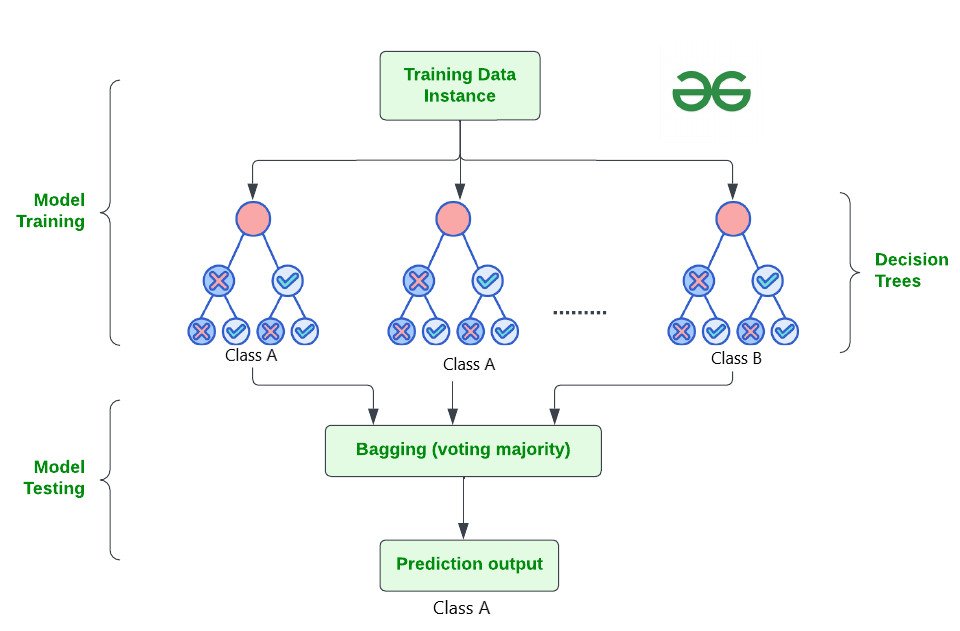
Kết quả nhìn chung cho thấy mô hình GIN có thể phù hợp gần như hoàn hảo với tất cả các tập huấn luyện. Các biển thể GNN với MLP có xu hướng hiệu suất cao các mô hình có perceptron 1 lớp. GIN hoạt động tương đương hoặc tốt hơn các biến thể GNN trên cả 9 bộ dữ liệu. Đặc biệt thì GIN tỏa sáng trên các bộ dữ liệu mạng xã hội, nơi chưa số lượng biểu đồ huấn luyện tương đối lớn.

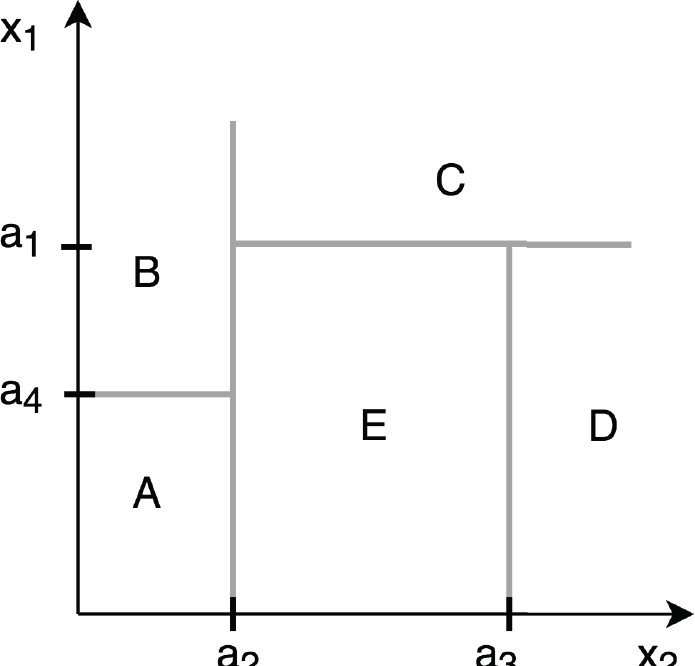
Nhìn chung, GIN là một mạng nơ-ron mới trong lĩnh vực trí tuệ nhân tạo, vì nó cung cấp một cách mới, mạnh mẽ hơn để phân tích và giải thích dữ liệu. Khi các nhà nghiên cứu tiếp tục khám phá các ứng dụng tiềm năng của thuật toán này, có khả năng GIN sẽ góp một phần quan trọng trong

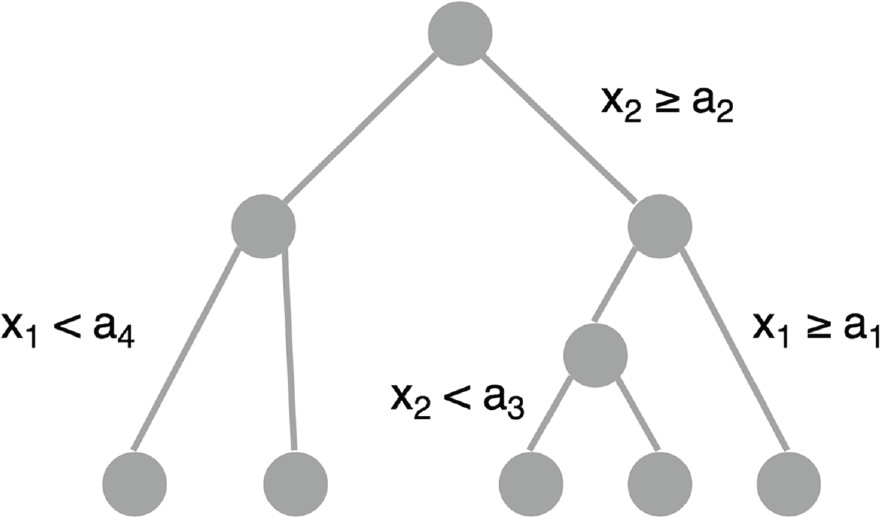
1. Random Forest

Học máy, sự kết hợp hấp dẫn giữa khoa học máy tính và thống kê, đã chứng kiến ​​sự tiến bộ đáng kinh ngạc, với một thuật toán nổi bật là Random forest. Random forest hay Cây quyết định ngẫu nhiên là một nhóm gồm các cây quyết định hợp tác làm việc cùng nhau để cung cấp một đầu ra duy nhất. Bắt nguồn từ năm 2001 thông qua Leo Breiman, Random Forest đã trở thành nền tảng cho những người đam mê học máy

* 1. Random Forest là gì

Random forest là một kỹ thuật học máy dựa vào cây trong Machine Learning. Nó hoạt động bằng cách tạo ra một số Cây quyết định trong giai đoạn đào tạo. Mỗi cây được xây dựng bằng cách sử dụng một tập hợp con ngẫu nhiên của tập dữ liệu để đo lường một tập hợp con ngẫu nhiên các tính năng trong mỗi phân vùng. Tính ngẫu nhiên này tạo ra sự biến đổi giữa các cây riêng lẻ, giảm nguy cơ trang bị quá mức và cải thiện hiệu suất dự đoán tổng thể. Trong dự đoán, thuật toán tổng hợp kết quả của tất cả các cây, bằng cách bỏ phiếu (đối với nhiệm vụ phân loại) hoặc bằng cách tính trung bình (đối với nhiệm vụ hồi quy). Quá trình ra quyết định hợp tác này, được hỗ trợ bởi nhiều cây với thông tin chuyên sâu của chúng, cung cấp một ví dụ về kết quả ổn định và chính xác . Rừng ngẫu nhiên được sử dụng rộng rãi cho các chức năng phân loại và hồi quy, được biết đến với khả năng xử lý dữ liệu phức tạp, giảm tình trạng trang bị quá mức và cung cấp dự báo đáng tin cậy trong các môi trường khác nhau

Đầu tiên chúng ta thảo luận về các mô hình dựa trên cây vì chúng tạo thành các khối xây dựng của thuật toán random forest. Mô hình dựa trên cây liên quan đến việc phân vùng đệ quy tập dữ liệu đã cho thành hai nhóm dựa trên một tiêu chí nhất định cho đến khi đáp ứng điều kiện dừng xác định trước. Ở dưới cùng của cây quyết định có cái gọi là nút lá hoặc lá.

Hình trên minh họa phân vùng đệ quy của không gian đầu vào hai chiều với các ranh giới thẳng hàng theo trục; nghĩa là, mỗi lần không gian đầu vào được phân vùng theo hướng song song với một trong các trục. Ở đây sự phân chia đầu tiên xảy ra trên *x*2*≥ a*2 . Sau đó, hai không gian con lại được phân chia: Nhánh trái được chia theo *x*1*≥ a*4 . Nhánh bên phải lần đầu tiên được phân chia trên *x*1*≥ a*1 và một trong các nhánh con của nó được phân chia trên *x*2*> a*3

Hình 2: biểu diễn đồ hoạ của các không gian con được phân vùng trong hình

Tùy thuộc vào cách thiết lập phân vùng và tiêu chí dừng, cây quyết định có thể được thiết kế cho cả nhiệm vụ phân loại (kết quả phân loại, ví dụ: hồi quy logistic) và nhiệm vụ hồi quy (kết quả liên tục).

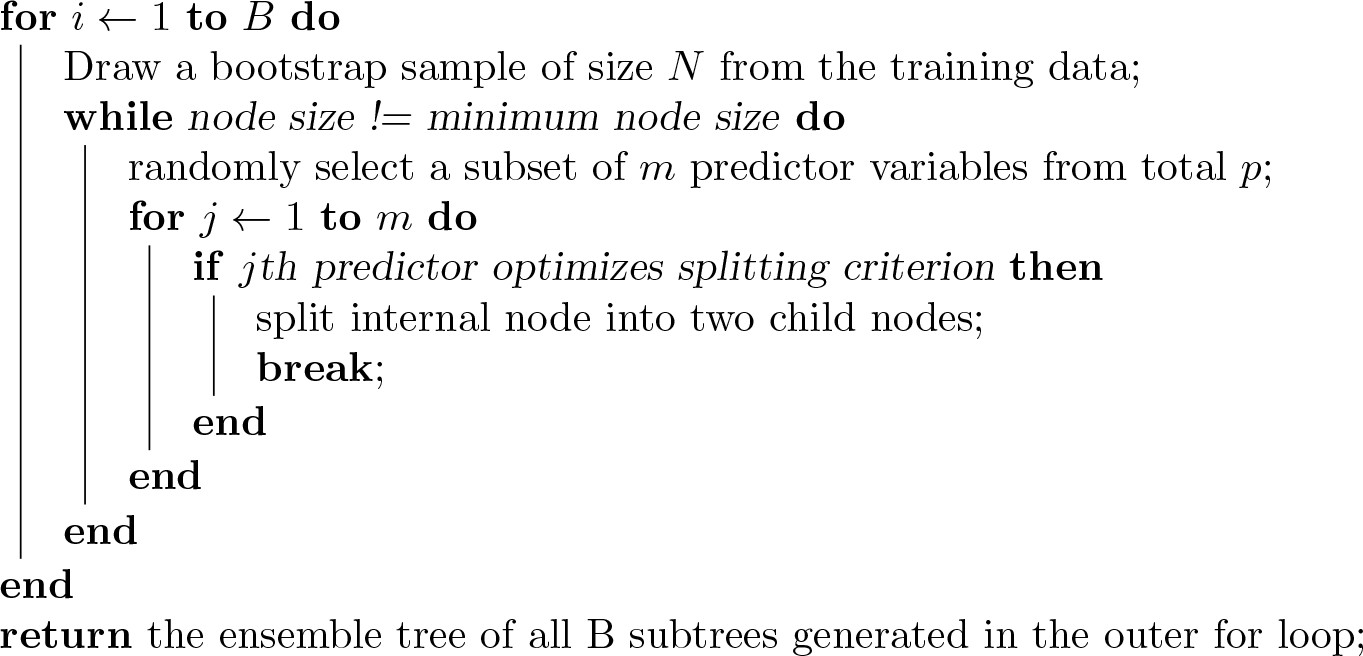
Đối với cả bài toán phân loại và hồi quy, tập hợp con của các biến dự đoán được chọn để phân chia một nút bên trong phụ thuộc vào tiêu chí phân tách được xác định trước được xây dựng dưới dạng bài toán tối ưu hóa. Một tiêu chí phân chia phổ biến trong các bài toán phân loại là entropy, đây là ứng dụng thực tế của định lý mã hóa nguồn [của Shannon (2001)](https://journals.sagepub.com/doi/full/10.1177/1536867X20909688#bibr10-1536867X20909688) xác định giới hạn dưới của độ dài biểu diễn bit của biến ngẫu nhiên. Tại mỗi nút bên trong của cây quyết định, entropy được tính theo công thức

1. 

trong đó *c* là số lớp duy nhất và p i là xác suất trước của mỗi lớp nhất định. Giá trị này được tối đa hóa để thu được nhiều thông tin nhất ở mỗi phần của cây quyết định. Đối với các bài toán hồi quy, tiêu chí phân tách thường được sử dụng là sai số bình phương trung bình tại mỗi nút bên trong.

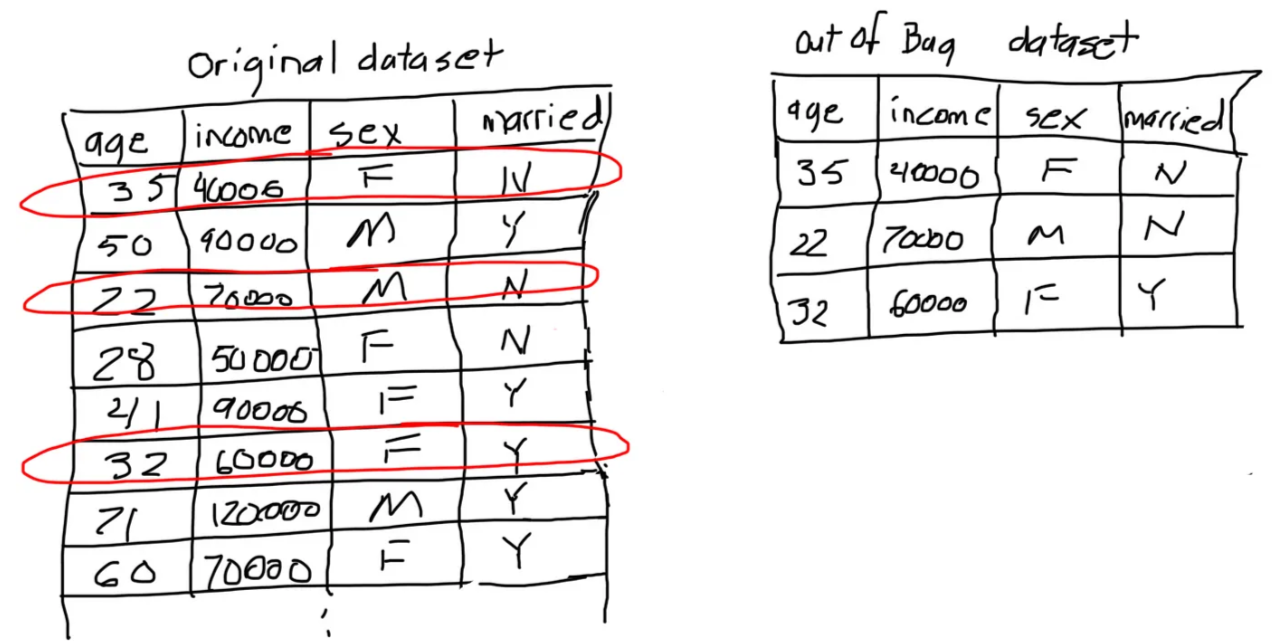
Một nhược điểm của cây quyết định là chúng dễ bị trang bị quá mức, điều đó có nghĩa là mô hình tuân theo các đặc điểm riêng của tập dữ liệu thử nghiệm quá chặt chẽ và hoạt động kém trên tập dữ liệu mới—tức là dữ liệu thử nghiệm. Cây quyết định trang bị quá mức sẽ dẫn đến độ chính xác dự đoán chung thấp, còn được gọi là độ chính xác tổng quát hóa.

Một cách để tăng độ chính xác của việc khái quát hóa là chỉ xem xét một tập hợp con của các quan sát và xây dựng nhiều cây riêng lẻ. Được giới thiệu lần đầu tiên bởi [Ho (1995)](https://journals.sagepub.com/doi/full/10.1177/1536867X20909688#bibr6-1536867X20909688) , ý tưởng về phương pháp không gian con ngẫu nhiên này sau đó đã được mở rộng và chính thức trình bày dưới dạng random forest bởi [Breiman (2001)](https://journals.sagepub.com/doi/full/10.1177/1536867X20909688#bibr2-1536867X20909688) . Mô hình random forest là một thuật toán học tập dựa trên cây tổng hợp; nghĩa là thuật toán tính trung bình các dự đoán trên nhiều cây riêng lẻ. Các cây riêng lẻ được xây dựng trên các mẫu bootstrap chứ không phải trên mẫu ban đầu. Điều này được gọi là tổng hợp bootstrap hoặc đơn giản là đóng bao, và nó làm giảm việc trang bị quá mức. Thuật toán như sau:



Các cây quyết định riêng lẻ có thể dễ dàng diễn giải được, nhưng khả năng diễn giải này bị mất trong các khu rừng ngẫu nhiên vì nhiều cây quyết định được tổng hợp lại. Tuy nhiên, đổi lại, các khu rừng ngẫu nhiên thường thực hiện nhiệm vụ dự đoán tốt hơn nhiều.

Thuật toán random forest ước tính chính xác hơn tỷ lệ lỗi so với cây quyết định. Cụ thể hơn, tỷ lệ lỗi đã được chứng minh về mặt toán học là luôn hội tụ khi số lượng cây tăng lên ( [Breiman 2001](https://journals.sagepub.com/doi/full/10.1177/1536867X20909688" \l "bibr2-1536867X20909688) ).



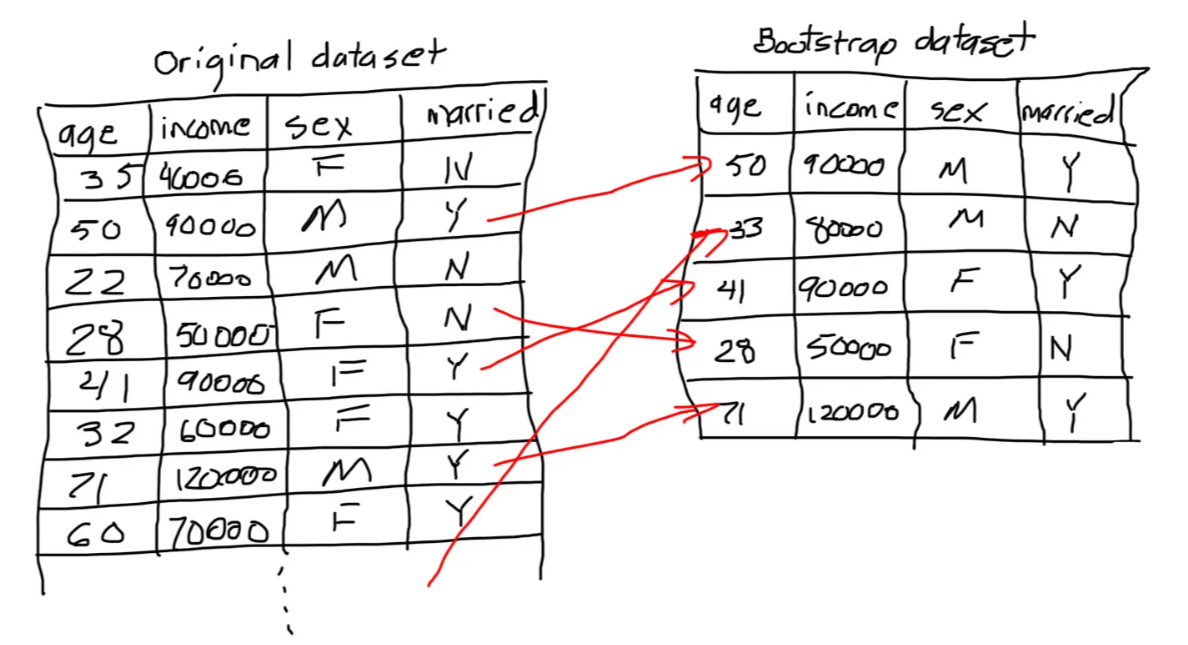
Lỗi của random forest được tính gần đúng bằng lỗi out-of-bag ( oob ) trong quá trình huấn luyện. Mỗi cây được xây dựng trên một mẫu bootstrap khác nhau. Mỗi mẫu bootstrap ngẫu nhiên bỏ đi khoảng một phần ba số quan sát. Những quan sát bị bỏ sót này đối với một cây nhất định được gọi là mẫu oob . Việc tìm kiếm các tham số có thể tạo ra lỗi oob thấp thường là vấn đề quan trọng cần cân nhắc khi lựa chọn mô hình và điều chỉnh tham số. Lưu ý rằng trong thuật toán rừng ngẫu nhiên, kích thước của tập hợp con các biến dự đoán, m , là rất quan trọng để kiểm soát độ sâu cuối cùng của cây. Do đó, đây là một tham số cần được điều chỉnh trong quá trình lựa chọn mô hình.

* 1. Đi sâu vào Random forest

a) Vậy Random forest hoạt động như thế nào ?

Tập hợp các Cây quyết định: Random forest tận dụng sức mạnh của việc học tập tổng hợp bằng cách xây dựng một đội quân Cây quyết định. Những cây này giống như các chuyên gia riêng lẻ, mỗi chuyên gia chuyên về một khía cạnh cụ thể của dữ liệu. Điều quan trọng là chúng hoạt động độc lập, giảm thiểu rủi ro mô hình bị ảnh hưởng quá mức bởi các sắc thái của một cây duy nhất.

Lựa chọn tính năng ngẫu nhiên: Để đảm bảo rằng mỗi cây quyết định trong quần thể mang đến một góc nhìn độc đáo, Random forest sử dụng lựa chọn tính năng ngẫu nhiên. Trong quá trình huấn luyện mỗi cây, một tập hợp con các đặc tính ngẫu nhiên sẽ được chọn. Tính ngẫu nhiên này đảm bảo rằng mỗi cây tập trung vào các khía cạnh khác nhau của dữ liệu, thúc đẩy một tập hợp các yếu tố dự đoán đa dạng trong quần thể.



Tổng hợp hoặc đóng bao Bootstrap: Kỹ thuật đóng bao là nền tảng trong chiến lược đào tạo của Random Forest, bao gồm việc tạo nhiều mẫu bootstrap từ tập dữ liệu gốc, cho phép lấy mẫu các phiên bản thay thế. Điều này dẫn đến các tập hợp dữ liệu khác nhau cho mỗi cây quyết định, tạo ra tính biến đổi trong quá trình đào tạo và làm cho mô hình trở nên mạnh mẽ hơn.

Ra quyết định và biểu quyết: Khi đưa ra dự đoán, mỗi cây quyết định trong Random forest sẽ bỏ phiếu. Đối với các nhiệm vụ phân loại, dự đoán cuối cùng được xác định theo chế độ (dự đoán thường xuyên nhất) trên tất cả các cây. Trong các nhiệm vụ hồi quy, giá trị trung bình của các dự đoán cây riêng lẻ được lấy. Cơ chế bỏ phiếu nội bộ này đảm bảo quá trình ra quyết định tập thể và cân bằng.

1. Một số đặc điểm chính của Random forest

Độ chính xác dự đoán cao: Hãy tưởng tượng Random forest như một nhóm gồm những người hướng dẫn đưa ra quyết định. Mỗi thuật sĩ (cây quyết định) xem xét một phần của vấn đề và cùng nhau dệt nên những hiểu biết sâu sắc của mình thành một tấm thảm dự đoán mạnh mẽ. Việc làm việc nhóm này thường tạo ra một mô hình chính xác hơn những gì một thuật sĩ đơn lẻ có thể đạt được.

Chống lại việc trang bị quá mức: Random Forest giống như một người cố vấn điềm tĩnh hướng dẫn những người học việc (cây quyết định). Thay vì để mỗi người học việc ghi nhớ từng chi tiết trong quá trình đào tạo của họ, nó khuyến khích sự hiểu biết toàn diện hơn. Cách tiếp cận này giúp tránh việc quá chú ý đến dữ liệu huấn luyện, khiến mô hình ít bị trang bị quá mức.

Xử lý tập dữ liệu lớn: Random forest giải quyết vấn đề này giống như một nhà thám hiểm dày dặn kinh nghiệm với một nhóm trợ giúp (cây quyết định). Mỗi người trợ giúp đảm nhận một phần của tập dữ liệu, đảm bảo rằng cuộc thám hiểm không chỉ diễn ra kỹ lưỡng mà còn nhanh chóng đến mức đáng ngạc nhiên.

Đánh giá tầm quan trọng có thể thay đổi: Hãy coi Random Forest như một thám tử tại hiện trường vụ án, tìm ra manh mối (đặc điểm) nào quan trọng nhất. Nó đánh giá tầm quan trọng của từng manh mối trong việc giải quyết vụ án, giúp bạn tập trung vào các yếu tố chính thúc đẩy dự đoán.

Xác thực chéo tích hợp: Random forest giống như có một huấn luyện viên cá nhân giúp bạn kiểm soát. Khi đào tạo từng cây quyết định, nó cũng dành ra một nhóm trường hợp bí mật (có sẵn) để thử nghiệm. Quá trình xác thực tích hợp này đảm bảo mô hình của bạn không chỉ đạt thành tích tốt trong quá trình đào tạo mà còn hoạt động tốt trước những thử thách mới.

Xử lý các giá trị bị thiếu: Cuộc sống chứa đầy những điều không chắc chắn, giống như những tập dữ liệu bị thiếu các giá trị. Random Forest là người bạn thích nghi với hoàn cảnh, đưa ra dự đoán dựa trên những thông tin có sẵn. Nó không bối rối vì những mảnh ghép bị thiếu; thay vào đó, nó tập trung vào những gì nó có thể tự tin cho chúng ta biết.

Song song hóa tốc độ: Random forest là người bạn tiết kiệm thời gian của bạn. Hãy tưởng tượng mỗi cây quyết định giống như một công nhân đang giải quyết một mảnh ghép cùng một lúc. Cách tiếp cận song song này khai thác sức mạnh của công nghệ hiện đại, giúp toàn bộ quá trình xử lý các dự án quy mô lớn nhanh hơn và hiệu quả hơn.

1. Chuẩn bị dữ liệu

Việc thu thâp dữ liệu là bước đầu tiên. Các tệp phần mềm độc hại và lành tính được thu thấp để cung cấp dữ liệu cho việc đào tạo mô hình bằng kỹ thuật học máy. Cơ sở dữ liệu về các tệp nguy hiểm và lành tính được kết hợp để tạo thành một tệp dữ liệu khổng lồ. Tập đữ liệu tệp lành tính có khoảng 15000 tệp, trong khi tập dữ liệu ứng dụng có hại chứa khoảng 3000 tệp.

Huấn luyện tập dữ liệu

Tập dữ liệu đã tạo được sử dụng để huấn luyện mô hình bằng kĩ thuật random forest với các tham số được cung cấp. Dữ liệu phải được sắp xếp, do đó việc chuẩn bị từ trước dữ liệu là một yếu tố quan trọng của quy trình. Để trích xuất các tính năng, mối quan hệ giữa các bộ dữ liệu này cũng phải được phát hiện, Việc tìm kiếm dữ liệu có các đặc điểm giá trị còn thiếu cũng là một phần quan trọng của quy trình. Do nhiều tệp hoặc phần mềm độc hại được thu thập từ Androzoo không thể được giải mã đầy đủ.

Khai thác tính năng

Mục đích chính là khám phá các mẫu quyền độc hại bằng cách sử dụng kết hợp các quyền làm bộ dữ liệu huấn luyện đầu vào ML của các tệp lành tính và độc hại. Các tính năng dựa trên quyền đã được trích xuát và những đặc điểm này đã được sử dụng để phân biệt giữa các yêu cầu cấp phép độc hại và lành tính. Việc trích xuất các tính năng rất là quan trọng để dự đoán. Chúng ta càng có nhiều tính năng thì tốc độ tính toán sẽ càng nhanh cũng như sẽ sử dụng ít bộ nhớ hơn. Điều này được sử dụng để duyệt qua mã nguồn đã được dịch ngược và truy xuất các API liên quan đến quyền được trích xuất dưới dạng các tính năngg từ cả tập dữ liệu tệp độc hại và lành tính, đồng thời mô hình này có thể được sử dụng để dự báo bất kỳ tệp nào từ tập dữ liệu thử nghiệm.

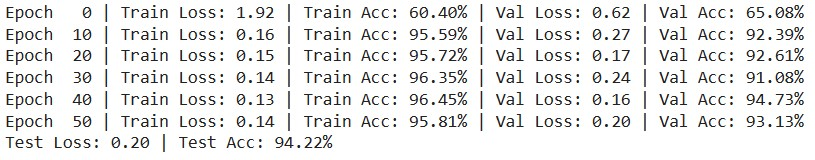
Phân loại

Thuật toán random forest được sử dụng để phân loại các đặc điểm sau khi chúng được trích xuất. Nếu chúng ta chia nhỏ từ này ra thì nó bao gồm forest, là tập hợp các cây quyết định và ngẫu nhiên, ám chỉ thực tế là chúng ta đang lấy mẫu ngẫu nhiên. Khi phương pháp này được áp dụng cho tập dữ liệu, một phần dữ liệu sẽ được sử dụng làm tập huấn luyện và dữ liệu được phân cụm thành các nhóm và nhóm con. Cây quyết định là một cấu trúc trông giống như một cái cây và được tạo bằng cách kết nối các điểm dữ liệu với các nhóm và nhóm con. Chương trình sau đó sẽ tạo ra một khu rừng từ một số cây. Tuy nhiên, mỗi cây là duy nhất vì các biến được chọn ngẫu nhiên cho mỗi phần trong cây.

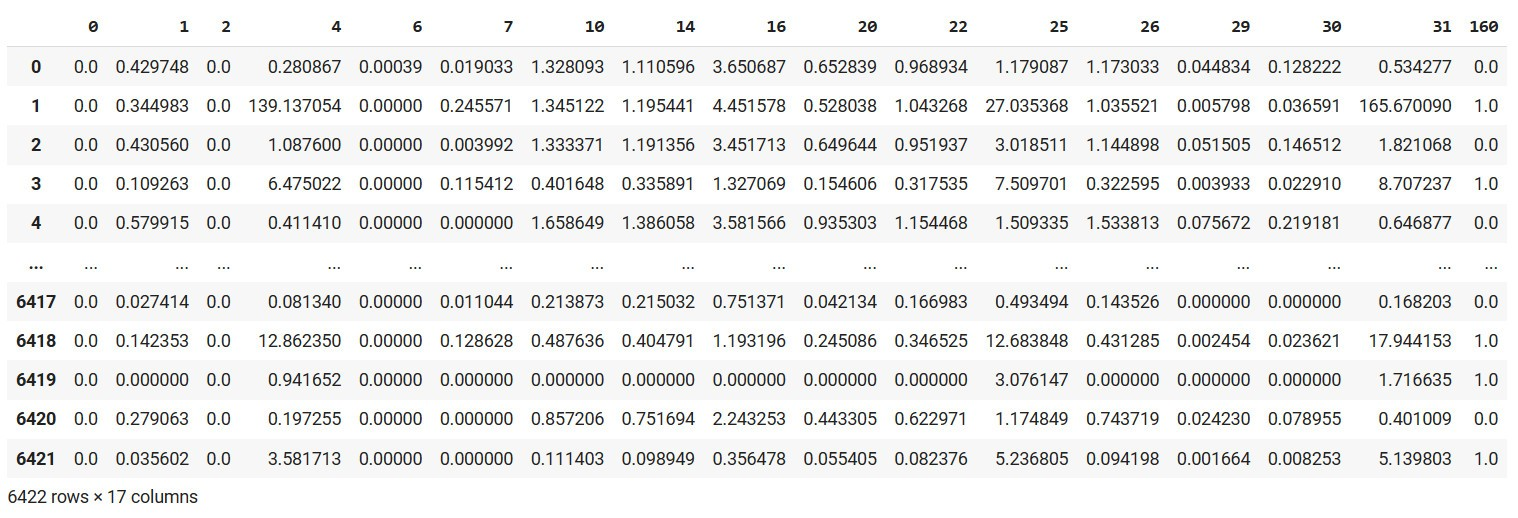
Ngoài tập huấn luyện, dữ liệu còn lại được sử dụng để dự báo cây nào trong rừng tạo ra sự phân loại điểm dữ liệu tốt nhất và cây có khả năng dự đoán cao nhất sẽ được hiển thị làm đầu ra. Sau đó, loại của mỗi chương trình được xác định bằng cách sử dụng một bộ nhãn, trong đó 1 biểu thị phần mềm độc hại và 0 biểu thị các tệp lành tính. Bằng cách giảm thiểu độ không chắc chắn của các nhãn lớp, cây quyết định chia tập huấn luyện thành hai tập con với các nhãn riêng biệt ở mỗi nút\

Chúng ta đã hiểu về bản chất của random forest là sự kết hợp giữa nhiều mô hình cây quyết định được huấn luyện trên các tập dữ liệu khác nhau được rút ra từ tập huấn luyện. Random forest có ưu điểm đó là giảm thiểu được hiện tượng quá khớp do có phương sai thấp và ít bị ảnh hưởng bởi nhiễu như mô hình cây quyết định. Khi huấn luyện mô hình, mô hình random forest cũng giúp chúng ta đánh giá nhanh tầm quan trọng của các biến đối với việc phân loại. Điều này cực kì hữu ích đối với những bộ dữ liệu có số chiều lớn.

1. **Tổng kết**
2. Đánh giá mô hình
3. Kết quả training mô hình GIN để học biểu diễn đồ thị



1. Vector tổng hợp của từng đồ thị thu được sau khi cho qua mô hình GIN



Kết quả thử nghiệm sau khi cho dữ liệu qua mô hình Random forest đạt độ chính xác lên tới 99%

1. **Danh mục tham khảo**
2. Fcg

[A Survey on Malware Detection with Graph Representation Learning - Archive ouverte HAL](https://hal.science/hal-04099618)

[Microsoft Word - RYFAVIJCCI (iop.org)](https://iopscience.iop.org/article/10.1088/1742-6596/1693/1/012080/pdf)

<https://www.usenix.org/conference/usenixsecurity23/presentation/li-heng>

["NSDroid: Efficient Multi-classification of Android Malware using Neigh" by Pengfei Liu, Weiping Wang et al. (csuohio.edu)](https://engagedscholarship.csuohio.edu/enece_facpub/460/?utm_source=engagedscholarship.csuohio.edu%2Fenece_facpub%2F460&utm_medium=PDF&utm_campaign=PDFCoverPages)

<https://doi.org/10.1007/s10207-023-00679-x>

[SeGDroid: An Android malware detection method based on sensitive function call graph learning - ScienceDirect](https://www.sciencedirect.com/science/article/abs/pii/S0957417423016275?via%3Dihub)

1. Infercode

<https://github.com/bdqnghi/infercode>

<https://arxiv.org/abs/2012.07023>

1. GIN

<https://mlabonne.github.io/blog/posts/2022-04-25-Graph_Isomorphism_Network.html>

<https://wandb.ai/graph-neural-networks/GIN/reports/What-are-Graph-Isomorphism-Networks---Vmlldzo1MTExMTg5>

[How to Design the Most Powerful Graph Neural Network | Towards Data Science](https://towardsdatascience.com/how-to-design-the-most-powerful-graph-neural-network-3d18b07a6e66)

[[1810.00826v3] How Powerful are Graph Neural Networks? (arxiv.org)](https://arxiv.org/abs/1810.00826v3)

1. Random forest

<https://arxiv.org/pdf/1609.07770>

https://www.researchgate.net/publication/308646416\_Random\_Forest\_for\_Malware\_Classification

<https://link.springer.com/chapter/10.1007/978-981-99-0601-7_56>