LỜI CAM ĐOAN

Tôi cam đoan đây là công trình nghiên cứu của riêng tôi.

Các số liệu, kết quả nêu trong luận văn là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác. Nếu không đúng như đã nêu trên, tôi xin hoàn toàn chịu trách nhiệm về đề tài của mình.

Tp. HCM, ngày 25 tháng 01 năm 2022

Học viên thực hiện luận văn

Huỳnh Vũ Trường Giang

LỜI CẢM ƠN

Trong thời gian thực hiện luận văn tốt nghiệp, được sự hướng dẫn tận tình của giáo viên hướng dẫn và được phía nhà trường tạo điều kiện thuận lợi, tôi đã có một quá trình nghiên cứu, tìm hiểu và học tập nghiêm túc để hoàn thành đề tài. Kết quả thu được không chỉ do nỗ lực của cá nhân tôi mà còn có sự giúp đỡ của quý thầy cô, gia đình và các bạn.

Tôi xin chân thành cảm ơn **PGS. TS. Trần Mạnh Hà**. Thầy đã hướng dẫn, hỗ trợ tôi hoàn thành tốt luận văn về phương pháp, lý luận và nội dung luận văn.

Cám ơn Bán Giám hiệu, Khoa Công Nghệ Thông Tin – Học Viện Công Nghệ Bưu Chính Viễn Thông cơ sở tại Tp. HCM đã quan tâm, tạo điều kiện giúp tôi hoàn thành luận văn tốt nghiệp.

Cám ơn Ban giám đốc và các đồng nghiệp tại Viễn thông Tây Ninh đã hỗ trợ, giúp đỡ tôi trong suốt quá trình thực hiện luận văn.

Trong quá trình thực hiện và trình bày không thể tránh khỏi những sai sót và hạn chế, do vậy tôi rất mong nhận được sự góp ý, nhận xét phê bình của quý thầy cô và các bạn để hoàn thiện kiến thức và bản thân.

Tp. HCM, ngày 25 tháng 01 năm 2022

Học viên thực hiện luận văn

Huỳnh Vũ Trường Giang

MỤC LỤC

DANH MỤC CÁC THUẬT NGỮ, CHỮ VIẾT TẮT

Viết tắt	Tiếng Anh	Tiếng Việt
AI	Artificial Intelligence	Trí tuệ nhân tạo
PDF	Portable Document Format	Định dạng văn bản đơn giản
RF	Random Forest	Rừng ngẫu nhiên
ANN	Artificial Neural Network	Mạng nơ-ron nhân tạo
CSDL	Database	Cơ sở dữ liệu
CNTT	Information Technology	Công nghệ thông tin
SVM	Support Vector Machines	Máy véc tơ hỗ trợ
BTS	Bug Tracking System	Hệ thống kiểm tra sự cố
CQĐ	Decision Tree	Cây quyết định
RF	Random Forest	Rừng ngẫu nhiên

DANH SÁCH HÌNH VẼ

DANH SÁCH BẢNG

MỞ ĐẦU

Lý do chọn đề tài

Ngày nay, với sự phát triển của hệ thống mạng và truyền thông cả về sự đa dạng, độ phức tạp và độ ổn định thì việc phát hiện lỗi trong mạng truyền thông và hệ thống phân tán thường yêu cầu sự tham gia của các công cụ hỗ trợ và chuyên môn của người vận hành hệ thống. Hệ thống giám sát đưa ra các sự kiện lỗi sau đó được chuyển tiếp cho người vận hành hệ thống để phân tích và tạo báo cáo lỗi. Việc xây dựng các chức năng phát hiện lỗi là một thách thức vì rất khó để có cách tiếp cận hiệu quả thay thế kiến thức và cơ chế suy luận của người vận hành hệ thống, đặc biệt là một số vấn đề liên quan đến tính khả dụng, khả năng chịu lỗi và khả năng dự đoán hiệu suất là rất khó phát hiện trên mạng truyền thông diện rộng và hệ thống phân tán với độ phức tạp, khả năng mở rộng và tầm quan trọng cao.

Quản lý lỗi thể hiện một trong những điều chắc chắn là trong bất kỳ loại mạng nào, bất kể quy mô, cho dù nó nhỏ và được sử dụng để mô phỏng trong phòng thí nghiệm hoặc một mạng lớn cung cấp truy cập internet và điện thoại phủ sóng cho toàn bộ quốc gia, có một chiến lược để phát hiện và xác định lỗi càng sớm càng tốt là điều quan trọng để duy trì hiệu suất và sự ổn định của mạng.

Phát hiện lỗi là một trong những chức năng chính của việc quản lý lỗi trong hệ thống mạng và phân tán. Chức năng này trở nên đòi hỏi cao do sự phát triển nhanh chóng của các hệ thống này ngày nay với độ phức tạp ngày càng cao, tính năng động và khả năng mở rộng. Tuy nhiên, việc tự động kiểm tra các lỗi này là một thách thức vì điều này phụ thuộc nhiều vào kiến thức chuyên môn và các công cụ hỗ trợ thường được sử dụng để tổ chức công việc và quy trình quản lý lỗi. Để vượt qua những trở ngại, một cách tiếp cận phổ biến đã áp dụng là các phương pháp thông minh để khai thác dữ liệu đầu ra của hệ thống và phát hiện các khiếm khuyết trên một hệ thống.

Fault detection [7] hay tiếng việt hiểu là phát hiện lỗi mạng, là một trong những vấn đề hiện nay trở lên rất quan trọng trong việc quản trị các hệ thống mạng.

Nó hạn chế tối đa việc hệ thống mạng và truyền thông bị gián đoạn trong quá trình hoạt động, đảm bảo an toàn và chất lượng dịch vụ cho hệ thống mạng và truyền thông. Với một hệ thống mạng và truyền thông có nhiều thiết bị, việc tìm hiểu nguyên nhân lỗi hệ thống để tìm cách khắc phục là vô cùng khó khăn với số lượng lỗi, cảnh báo vô cùng lớn. Vì vậy việc áp dụng các kỹ thuật học máy để phân loại và dự báo các cảnh báo/lỗi mạng thực sự là lỗi gì và nguyên nhân do đâu là vô cùng cần thiết. Như việc áp dụng Máy hỗ trợ vectơ cho việc chẩn đoán lỗi mạng của các tác giả [10]. Eslamloueyan trong nghiên cứu [11] đã xây dựng hệ thống mạng nơron phân cấp dựa trên thuật toán gom cụm mờ để chuẩn đoán lỗi mạng. Trong luận văn này sẽ tiến hành nghiên cứu việc áp dụng thuật toán "Rừng ngẫu nhiên" vào việc hỗ trợ xác định lỗi mạng và dự báo sự cố dựa trên việc học có giám sát. Các phương pháp học máy không chỉ giúp phân tích dữ liệu sự kiện chính xác hơn mà còn dự báo các sự kiện lỗi có thể xảy ra bằng cách học hỏi từ những lỗi hiện có. Vì vậy hoàn toàn có thể xây dựng một hệ thống phát hiện lỗi tự động hỗ trợ người vận hành hệ thống phát hiện và dự báo lỗi.

Mục đích nghiên cứu

Hiện tại không có cách thực tế nào để phân tích lỗi của một thành phần trong hệ thống mạng một cách tự động. Nó được để lại như một nhiệm vụ yêu cầu người vận hành thực hiện thủ công bằng cách sử dụng vô số công cụ để thu thập thông tin về hoạt động hiện tại của các thiết bị trên hệ thống. Các câu hỏi yêu cầu câu trả lời trong luận văn này là:

- Liệu có một công cụ tự động có thể hỗ trợ thực hiện quá trình trên?

- Chúng ta có thể xây dựng một mô hình có khả năng thu thập tất cả thông tin lỗi này không, hãy hiểu rõ về nó, và do đó tiết kiệm thời gian và tài nguyên cho người vận hành?

Mục đích của luận văn là xây dựng một mô hình có thể thực hiện tự động đánh giá mức độ nghiêm trọng của lỗi. Trước đây, khó có được thông tin các lỗi đã xảy ra

khi thu thập bằng cách thủ công. Luận văn muốn khai thác thông tin có sẵn tại các Bug Tracking System [7] và sử dụng nó một cách hiệu quả nhất có thể để tiết lộ nguyên nhân của lỗi và đánh giá mức độ nghiêm trọng của lỗi. Luận văn này nhằm mục đích vạch ra con đường hướng tới một cách tiếp cận tự chủ hơn để quản lý lỗi bằng cách phát triển một mô hình dựa trên việc phân lớp và dự đoán theo thuật toán Rừng ngẫu nhiên và phương pháp tf x idf [6]. Về cơ bản, công việc này có ý định đưa các cảnh báo mức độ nghiêm trọng của lỗi thay vì thực hiện thăm dò thủ công. Nhằm mục đích đưa ra các cảnh báo này một cách kịp thời, đáng tin cậy.

Đối tượng nghiên cứu

Nghiên cứu tìm hiểu xác định lỗi của các thiết bị trên hệ thống mạng.

Các kỹ thuật phân tích dữ liệu giúp phân lớp, gom cụm hoặc dự đoán nguyên nhân lỗi của thông tin lỗi từ hệ thống gửi về.

Kỹ thuật khai phá dữ liệu văn bản nhằm mục đích phân tích nội dung lỗi.

Phạm vi nghiên cứu

Sử dụng các cảnh báo lỗi như: warning, error… từ thiết bị gửi về để xác định lỗi và chẩn đoán nguyên nhân lỗi. Đánh giá lỗi đang gặp phải là lỗi nhỏ, lỗi bình thường hay lỗi nghiêm trọng cần xử lý ngay.

Luận văn này sẽ nhấn mạnh vào quá trình làm thế nào để tự động hóa việc phát hiện các lỗi nghiêm trọng trong vô số các lỗi mà các thiết bị đang gửi về tự hệ thống. Mục đích sử dụng công cụ phần mềm là thay thế thành phần thủ công trong việc phân loại các lỗi hiện hành. Chỉ những công cụ và thông tin hiện có, với sự xử lý của một trình phân loại có kinh nghiệm, sẽ được xem xét và đánh giá đầu vào của luận văn là các lỗi đang gửi về. Công cụ sẽ không cố gắng sửa chữa lỗi, mục tiêu là nghiên cứu và báo cáo thông tin có ảnh hưởng đáng kể đến hoạt động bình thường của hệ thống.

Phương pháp nghiên cứu

Quá trình khám phá kiến thức trong khai thác dữ liệu [4] chủ yếu bao gồm các quy trình sau: thu thập dữ liệu, chuẩn bị dữ liệu, xây dựng mô hình và kết quả phân

tích. Thu thập dữ liệu là bước xác định loại tập dữ liệu liên quan và thu thập chúng từ các nguồn đáng tin cậy. Chuẩn bị dữ liệu là quá trình định dạng dữ liệu theo các thông số kỹ thuật của thuật toán và làm sạch để tránh những ngoại lệ. Xây dựng mô hình bao gồm đào tạo mô hình cho các bộ dữ liệu và kiểm tra mô hình để có được tỉ lệ chính xác và các kết quả thống kê khác. Phân tích kết quả liên quan đến diễn giải kết quả mô hình được sử dụng để đưa ra kết luận cho người dùng. Phần lớn, nghiên cứu này tuân theo phương pháp luận này; tuy nhiên, những thay đổi nhỏ như chia nhỏ các tập dữ liệu và phân tích thống kê đã được thực hiện để phù hợp với công việc nghiên cứu.

Mô hình phân loại lỗi này được đặc trưng bởi khả năng khai thác tài nguyên kiến thức lỗi tại các kho lưu trữ trực tuyến khác nhau, các sự kiện nhật ký và các thông số trạng thái từ hệ thống được giám sát; và áp dụng phân tích lỗi cùng phương pháp lọc sự kiện để đánh giá sự kiện và dự báo lỗi. Hệ thống sử dụng một mô hình dữ liệu lỗi để thu thập các báo cáo lỗi, xây dựng một tính năng và phương pháp lọc ngữ nghĩa để tương quan các sự kiện và phương pháp học máy để đánh giá mức độ nghiêm trọng, mức độ ưu tiên và mối quan hệ của các sự kiện nhật ký và dự báo các lỗi nghiêm trọng sắp tới của hệ thống được giám sát.

Đề tài này sử dụng các phương pháp nghiên cứu lý thuyết để phân tích dữ liệu kết hợp với việc xây dựng một mô hình đưa ra phân loại sự cố nghiêm trọng và cảnh báo sự cố nghiêm trọng nếu có cho người quản trị mạng:

Phần nội dung của luận được chia thành 5 chương:

Chương 1: Nghiên cứu tổng quan, đưa ra lĩnh vực nghiên cứu cũng như mang lại cho người đọc kiến thức về các khái niệm được sử dụng trong luận văn.

Chương 2: Tìm hiểu cách phân loại lỗi mạng: nghiên cứu mô hình, thuộc tính của lỗi mạng và phương pháp khai phá nội dung của lỗi mạng.

Chương 3: Xây dựng mô hình phân loại lỗi mạng, mô tả công việc được thực hiện để trả lời các câu hỏi nghiên cứu.

Chương 4: Phân tích và đánh giá kết quả thực hiện

Chương 5: Kết luận

Chương 1 - NGHIÊN CỨU TỔNG QUAN

1.1. Bài toán phân lớp dữ liệu

1.1.1. Khái niệm về phân lớp dữ liệu và bài toán phân lớp dữ liệu

Khai phá dữ liệu: Khai phá dữ liệu nói chung có nghĩa là khai thác hoặc đào sâu vào dữ liệu ở các dạng khác nhau để có được các mẫu và để có được kiến

thức về mẫu đó. Trong quá trình khai thác dữ liệu, các tập dữ liệu lớn trước tiên được sắp xếp, sau đó các mẫu được xác định và các mối quan hệ được thiết lập để thực hiện phân tích dữ liệu và giải quyết vấn đề [28].

Phân lớp dữ liệu: Đây là một nhiệm vụ phân tích dữ liệu, tức là quá trình tìm kiếm một mô hình mô tả và phân biệt các lớp và khái niệm dữ liệu. Phân loại là vấn đề xác định một tập hợp các danh mục (quần thể con), một dữ liệu mới thuộc về loại nào, trên cơ sở một tập dữ liệu huấn luyện chứa các dữ liệu và các lớp của chúng đã được biết đến [28].

Phân lớp dữ liệu có thể chia làm các bước sau:

Bước học tập (Giai đoạn đào tạo): Xây dựng mô hình phân loại. Các thuật toán khác nhau được sử dụng để xây dựng mô hình phân loại bằng cách làm cho mô hình học bằng cách sử dụng tập huấn luyện có sẵn. Mô hình phải được đào tạo để dự đoán kết quả chính xác. Dữ liệu kiểm tra được sử dụng để ước tính độ chính xác của quy tắc phân loại.

Bước phân loại: Mô hình được sử dụng để dự đoán và thử nghiệm mô hình đã xây dựng trên dữ liệu thử nghiệm và sau đó ước tính độ chính xác của các quy tắc phân loại. Dữ liệu kiểm tra được sử dụng để ước tính độ chính xác của quy tắc phân loại.

Ta có thể phát biểu bài toán phân lớp dữ liệu như sau:

Đầu vào của bài toán phân lớp dữ liệu:

Cho tập dữ liệu ban đầu $D = \{(x_i, y_i) \mid i = 1, 2, \ldots, n\}$, trong đó, $x_i = (x_{i1}, x_{i2}, ..., x_{ik}) \in R^k$ là dữ liệu gồm k thuộc tính ứng với tập thuộc tính $A = \{A_1, A_2, \ldots, A_k\}$ và $y_i \in C = \{c_1, c_2, \ldots, c_m\}$ là tập nhãn của các lớp dữ liệu ban đầu.

Đầu ra của bài toán phân lớp dữ liệu:

Một mô hình phân lớp $F: R^k \rightarrow C$, tương ứng mỗi phần tử $x \in R^k$ là một nhãn lớp $F(x) \in C$, sao cho đối với tập mẫu đầu vào D là phù hợp nhất theo nghĩa sau đây:

$||F(x_i) - y_i|| \cong 0$, với mọi $(x_i, y_i) \in D$ và $||\ ||$ là một độ đo nào đó.

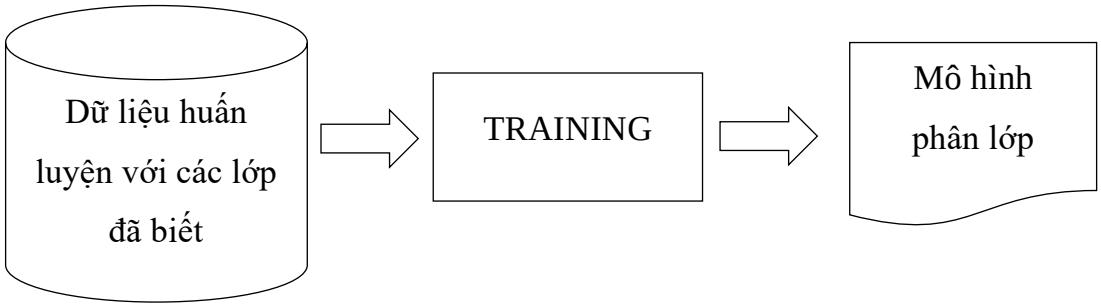
1.1.2. Các bước giải quyết bài toán phân lớp dữ liệu

Để giải quyết bài toán phân lớp dữ liệu ta tiến hành hai gian đoạn: giai đoạn đầu tiên ta xây dựng mô hình phân lớp (còn hay được gọi là *giai đoạn Huấn luyện*) và giai đoạn thứ hai là kiểm tra đánh giá mô hình phân lớp (còn được gọi là *giai đoạn Kiểm chứng*).

Giai đoạn huấn luyện

Quá trình này nhằm mục đích xây dựng ra một mô hình phân lớp dữ liệu dựa trên việc mô tả tập các lớp dữ liệu hoặc các khái niệm đã được xác định trước. Trong giai đoạn này, thuật toán phân lớp được sử dụng để xây dựng mô hình phân lớp bằng cách phân tích hay "học" từ một tập các dữ liệu huấn luyện (training set) và các nhãn tương ứng của chúng [4].

Quá trình thực hiện giai đoạn học được mô tả trong hình 1.1.



Hình 1.1: Giai đoạn xây dựng mô hình phân lớp dữ liệu

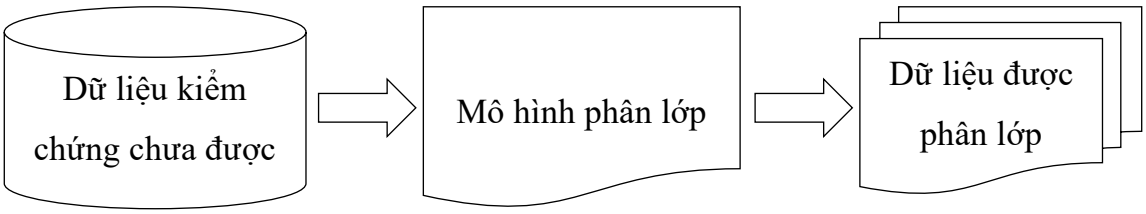
Kết quả sau khi kết thúc giai đoạn này là đưa ra một mô hình phân lớp dữ liệu. Mô hình phân lớp dữ liệu có thể là các công thức toán học, hoặc các luật quyết định, hoặc bộ các quy tắc để gán nhãn lớp cho mỗi dữ liệu trong tập các dữ liệu huấn luyện.

Giai đoạn kiểm chứng

Ở giai đoạn này, mô hình phân lớp ở bước đầu tiên sẽ được sử dụng để thực hiện phân lớp thử nghiệm và đánh giá mô hình phân lớp. Tập các dữ liệu test hay tập kiểm chứng được sử dụng trong giai đoạn. Do đó, tập dữ liệu kiểm chứng được sử

dụng trong giai đoạn này phải độc lập với tập dữ liệu huấn luyện ở giai đoạn huấn luyện [4].

Quá trình thực hiện giai đoạn phân lớp thử nghiệm được mô tả trong hình 1.2.



Hình 1.2: Quá trình kiểm tra đánh giá mô hình phân lớp dữ liệu

Các kết quả phân lớp trong quá trình phân lớp thử nghiệm lại có thể sử dụng trong quá trình học tiếp theo.

Sau khi thực hiện xong hai giai đoạn trên, một mô hình phân lớp phù hợp nhất theo một ý nghĩa nào đó (thông qua việc đánh giá các độ đo của mô hình) sẽ được lựa chọn để thực hiện việc phân lớp dữ liệu trong các bài toán ứng dụng khác nhau trong thực tế.

1.1.3. Các độ đo để đánh giá mô hình phân lớp dữ liệu

Sự phù hợp, mức độ hiệu quả của bất kỳ mô hình phân lớp dữ liệu nào cũng thường được xác định thông qua các độ đo được mô tả dưới đây.

Xét một lớp dữ liệu $c_i \in C = \{c_1, c_2, \ldots, c_m\}$ trong một bài toán phân lớp. Tập hợp các mẫu dữ liệu thuộc lớp c_i được gọi là các phần tử dương (positive). Tập hợp các mẫu dữ liệu không thuộc lớp c_i được gọi là các phần tử âm (negative). Kết quả phân lớp sau khi thực hiện phân lớp dữ liệu có thể xảy ra các trường hợp sau đây:

- True Positive (Trường hợp đúng dương): Phần tử dương được phân loại đúng là dương.
- False Positive (Trường hợp sai dương): Phần tử âm được phân loại sai thành dương.
- True Negative (Trường hợp đúng âm): Phần tử âm được phân loại đúng là âm.

- False Negative (Trường hợp sai âm): Phần tử dương được phân loại sai thành âm.

Ta gọi TP_i là số lượng các mẫu dữ liệu thuộc vào lớp c_i được phân loại đúng (chính xác) vào lớp c_i; gọi FP_i là số lượng các mẫu dữ liệu không thuộc lớp c_i nhưng bị phân loại sai vào lớp c_i; gọi TN_i là số lượng các mẫu dữ liệu không thuộc lớp c_i và được phân loại chính xác và gọi FN_i là số lượng các mẫu dữ liệu thuộc lớp c_i nhưng bị phân loại sai vào các lớp khác với lớp c_i.

Căn cứ vào các đại lượng trên, các khái niệm độ đo sau để đánh giá mức độ hiệu quả của mô hình phân lớp dữ liệu:

Độ đo Precision (Mức chính xác)

Định nghĩa: Precision = TP / (TP + FP).

Ý nghĩa: Giá trị Precision càng cao thể hiện khả năng để một kết quả phân lớp dữ liệu được đưa ra bởi bộ phân lớp là chính xác càng cao.

Độ đo Recall (Độ bao phủ, độ nhạy hoặc độ triệu hồi)

Định nghĩa: Recall = TP / (TP + FN).

Ý nghĩa: Giá trị Recall càng cao thể hiện khả năng kết quả đúng trong số các kết quả đưa ra của bộ phân lớp càng cao.

Độ đo Accuracy (Độ chính xác)

Định nghĩa: Accuracy = (TP + TN) / (TP + TN + FP + FN) * 100%.

Ý nghĩa: Accuracy phản ánh độ chính xác chung của bộ phân lớp dữ liệu.

Độ đo Specificity (Độ đặc hiệu)

Định nghĩa: Specificity = TN/(TN+FP).

Ý nghĩa: Độ đo Specitivity đánh giá khả năng một dữ liệu là phần tử âm được bộ phân lớp cho ra kết quả chính xác.

1.2. Tổng quan về học máy

Trong các lĩnh vực khoa học, công nghệ và nhân văn khác nhau, cũng như trong sinh học, khí tượng, y học hoặc tài chính, để trích dẫn một số, các chuyên gia nhắm vào dự đoán một hiện tượng dựa trên các quan sát hoặc đo lường trong quá khứ. Ví dụ, các nhà khí tượng học cố gắng dự báo thời tiết cho những ngày tiếp theo từ điều kiện khí hậu của những ngày trước đó. Trong y học, luyện tập thu thập các phép đo và thông tin như huyết áp, tuổi hoặc tiền sử chẩn đoán tình trạng của bệnh nhân. Ban đầu, trong hóa học, các hợp chất được phân tích bằng cách sử dụng khối phổ thử các phép đo để xác định xem chúng có chứa một loại phân tử hoặc nguyên tử. Trong tất cả các trường hợp này, mục tiêu là sự thay đổi của một biến phản hồi dựa trên một tập hợp các yếu tố dự đoán được quan sát. Trong nhiều thế kỉ, các nhà khoa học đã giải quyết những vấn đề như vậy bằng cách dẫn xuất theo khuôn khổ lý thuyết từ các nguyên tắc đầu tiên hoặc đã tích lũy kiến thức để mô hình hóa, phân tích và hiểu các vấn đề đang nghiên cứu. Ví dụ, các học viên biết từ những bệnh nhân cũ trong quá khứ, bệnh nhân cao tuổi bị đau tim với huyết áp thấp nói chung là rủi ro cao. Tương tự, các nhà khí tượng học biết từ lớp học các mô hình khí hậu mà một ngày nắng nóng, ô nhiễm cao có khả năng xảy ra tiếp theo là các diễn biến khác. Tuy nhiên, đối với một số vấn đề ngày càng tăng về số lượng, các phương pháp tiếp cận tiêu chuẩn bắt đầu chỉ ra các giới hạn của nó. Ví dụ, xác định thâm nhập các yếu tố nguy cơ di truyền đối với bệnh tim, nơi mà kiến thức vẫn còn rất thưa thớt, gần như không thực tế đối với khả năng nhận thức của con người do sự phức tạp cao và phức tạp của các tương tác tồn tại trong gen di truyền. Tương tự như vậy, đối với các dự báo khí tượng chi tiết, một số lượng lớn các biến cần phải được tính đến, nhanh chóng vượt ra ngoài khả năng của các chuyên gia để đưa tất cả họ vào một hệ phương trình. Để phá vỡ rào cản nhận thức này, máy móc với tốc độ và công suất ngày càng tăng đã được xây dựng và thiết kế từ giữa thế kỷ XX để hỗ trợ con người trong tính toán của họ. Tuy nhiên, thật đáng ngạc nhiên, cùng với sự tiến bộ này về phần cứng, sự phát triển trong khoa học máy tính lý thuyết, trí thông minh nhân tạo và số liệu thống kê nhanh chóng đã chứng minh máy móc trở nên vượt trội hơn máy tính. Những tiến bộ gần đây đã khiến họ trở thành chuyên gia trong lĩnh vực riêng, có khả năng học

hỏi từ dữ liệu và tự khám phá cấu trúc dự đoán của các vấn đề. Các kỹ thuật và thuật toán bắt nguồn từ lĩnh vực máy học đã thực sự trở thành một công cụ mạnh mẽ để phân tích dữ liệu lớn và phức tạp, hỗ trợ thành công các nhà khoa học trong nhiều bước đột phá của các biến thể trong lĩnh vực khoa học và công nghệ. Ví dụ công khai và nổi tiếng bao gồm việc sử dụng cây quyết định tăng cường trong phân tích thống kê dẫn đến việc phát hiện Higgs boson tại CERN [25], việc sử dụng các rừng ngẫu nhiên để phát hiện tư thế con người ở Microsoft Kinect [26] hoặc bộ phận tổng hợp các kỹ thuật học máy khác nhau để xây dựng hệ thống IBM tại Watson [27], có khả năng cạnh tranh với người đàn ông vô địch trên chương trình đố vui truyền hình Jeopardy của Mỹ. Về mặt hình thức, học máy có thể được định nghĩa là nghiên cứu các hệ thống có thể học từ dữ liệu mà không cần được lập trình rõ ràng. Một chương trình máy tính được cho là học từ dữ liệu và đo lường hiệu suất nếu hiệu suất của nó ở những tác vụ đó được cải thiện cùng với dữ liệu. Đặc biệt, học máy cung cấp các thuật toán có thể giải quyết các nhiệm vụ hồi quy, do đó mang đến các quy trình tự động để dự đoán một hiện tượng dựa trên những quan sát trong quá khứ. Tuy nhiên, từ trước đến nay, mục tiêu của học máy không chỉ là tạo ra các thuật toán đưa ra dự đoán chính xác, nó cũng là để cung cấp thông tin chi tiết về cấu trúc của dữ liệu. Đối với các học viên, không phải là chuyên gia trong lĩnh vực máy học, nó cung cấp các diễn giải thực sự quan trọng như độ chính xác của dự đoán. Nó cho phép hiểu rõ hơn trong việc tìm hiểu hiện tượng đang nghiên cứu, khám phá dữ liệu tốt hơn và tự đạt kết quả dễ dàng hơn.

1.2.1. Khái niệm

Học máy là một những lĩnh vực của trí tuệ nhân tạo, học máy liên quan đến quá trình nghiên cứu và xây dựng các kĩ thuật giúp các hệ thống máy tính học tự động từ dữ liệu ban đầu để giải quyết một số vấn đề cụ thể nào đó.

Học máy là một quá trình tự động của các quá trình học và việc học thì tương đương với quá trình xây dựng các tập luật trên cơ sở quan sát các trạng thái của cơ sở dữ liệu và những sự thay đổi của chúng. Học máy là lĩnh vực rộng lớn và nó không chỉ bao gồm việc học từ các mẫu, mà còn là học tăng cường. Các thuật toán học máy

dựa trên tập dữ liệu mẫu và các thông tin liên quan để làm đầu vào và trả về kết quả đầu ra là một mô hình diễn tả những kết quả học được.

Nhìn chung, học máy sẽ sử dụng một tập hữu hạn các dữ liệu được gọi là tập huấn luyện. Tập này sẽ chứa các mẫu dữ liệu mà nó được chuẩn hóa bằng mã theo một cách nào đó để máy có thể đọc và hiểu được. Tuy nhiên có một sự thật là tập huấn luyện bao giờ cũng có hữu hạn các phần tử, vì vậy không phải toàn bộ dữ liệu sẽ được học một cách chính xác.

1.2.2. Phân loại các kỹ thuật học máy

Các thuật toán học máy được chia làm 3 loại chính: học có giám sát, học không giám sát và học bán giám sát.

Học có giám sát

Học có giám sát là phương pháp học từ những dữ liệu mà trong quá trình học các kỹ thuật học máy sẽ giúp hệ thống xây dựng cách xác định những lớp dữ liệu. Hệ nthống bắt buộc phải tìm ra một sự mô tả cho từng lớp dữ liệu. Sau đó người ta có thể sử dụng các luật phân loại được hình thành trong quá trình học và phân lớp nó để có thể sử dụng cho việc dự báo các lớp dữ liệu sau này.

Học không giám sát

Học không giám sát là hệ thống khai thác dữ liệu ứng dụng với những dữ liệu không có lớp được định nghĩa cụ thể từ trước, mà để máy học phải tự hệ thống quan sát các mẫu và nhận ra mẫu. Hệ thống này sẽ dẫn đến một tập lớp, mỗi lớp có một tập mẫu riêng được khám phá từ trong tập dữ liệu. Học không giám sát hay còn gọi là học từ quan sát và khám phá.

Học bán giám sát

Đây là các thuật toán học tích hợp từ việc học giám sát và việc học không giám sát. Học bán giám sát sẽ sử dụng cả dữ liệu đã gán nhãn và chưa gán nhãn để huấn luyện – điển hình là một số ít dữ liệu có gán nhãn cùng với lượng lớn dữ liệu chưa gán nhãn ban đầu.

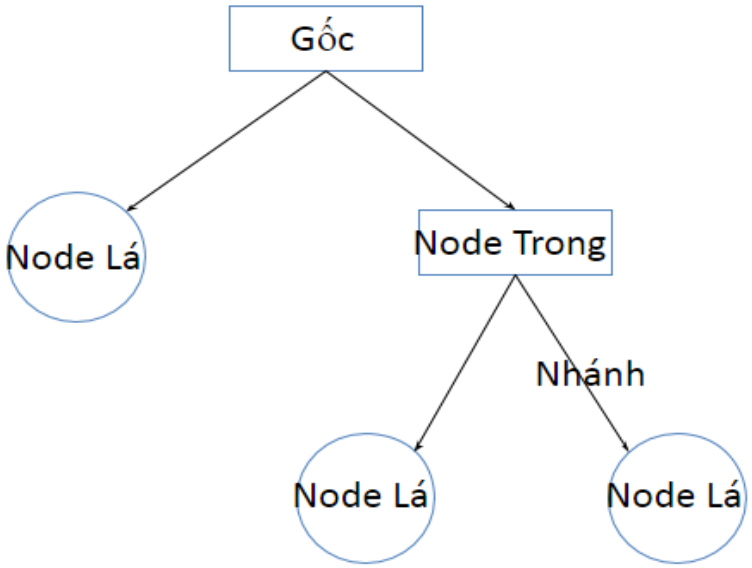
Học bán giám sát là quá trình học đứng giữa học không giám sát (không có bất kì dữ liệu đã được nhãn nào) và có giám sát (toàn bộ dữ liệu đều được gán nhãn).

Việc học bán giám sát tận dụng những ưu điểm của việc học giám sát và học không giám sát và loại bỏ những khuyết điểm thường gặp trên hai kiểu học này.

1.3. Thuật toán Cây quyết định

1.3.1. Giới thiệu phương pháp

Cây quyết định [24] là một mô hình cấu trúc cây giống như một lưu đồ mà trong đó mỗi nút bên trong cây diễn tả cho việc kiểm tra một thuộc tính, mỗi nhánh trên cây sẽ đại diện cho một kết quả của quá trình kiểm tra và các nút lá sẽ đại diện cho các lớp hoặc phân phối lớp. Nút trên cùng sẽ là nút gốc. Quá trình xây dựng cây quyết định được thực hiện bằng việc phân tách các dữ liệu trong một nút, chia chúng thành các nút con. Quá trình tương tự được áp dụng cho từng các nút con một cách đệ quy cho đến khi không còn nút con nào có thể được tách ra nữa. Các nút không thể được chia nhỏ hơn nữa sẽ được phát triển thành các nút lá. Cây quyết định được biểu diễn dưới dạng một cấu trúc cây như trong hình 1.3 dưới đây.



(Nguồn: Internet)

Hình 1.3: Mô hình cây quyết định

Trong cây mô hình quyết định, mỗi nút trung gian [5], tức là nút khác với nút lá và nút gốc, sẽ tương ứng với một phép kiểm tra một thuộc tính. Mỗi nhánh phía dưới của nút đó sẽ tương ứng cho một giá trị của thuộc tính hay còn gọi là kết quả

của phép thử. Khác với các nút trung gian, nút lá [5] không chứa thuộc tính cụ thể mà sẽ chứa các nhãn phân lớp. Để xác định nhãn phân lớp cho một dữ liệu mẫu bất kỳ, ta cho dữ liệu mẫu di chuyển từ gốc cây về phía nút lá. Tại mỗi nút trung gian, thuộc tính tương ứng với nút đó được kiểm tra, tùy vào giá trị của thuộc tính đó mà dữ liệu mẫu sẽ được chuyển xuống nhánh bên dưới tương ứng. Quá trình di chuyển này lặp lại cho đến khi dữ liệu mẫu đó tới được nút lá và được gán nhãn phân lớp là nhãn của nút lá tương ứng.

Quá trình xây dựng một cây quyết định thường được thực hiện như sau:

(1) Bắt đầu từ nút gốc nơi biểu diễn tất cả các mẫu của tập dữ liệu.

(2) Nếu tất cả các mẫu thuộc về cùng một lớp, nút đang xét sẽ trở thành nút lá và được gán nhãn chính bằng lớp đó.

(3) Ngược lại, dùng độ đo thuộc tính nào đó để chọn thuộc tính sẽ phân tách các mẫu tốt nhất vào các lớp tương ứng.

(4) Một nhánh được tạo ra cho từng giá trị của thuộc tính được chọn.

(5) Lặp lại quá trình trên để tạo cây quyết định.

(6) Tiến trình kết thúc chỉ khi bất kỳ điều kiện nào sau đây là đúng:

- Tất cả các mẫu của một nút cho trước đều thuộc về cùng một lớp.
- Không còn thuộc tính nào mà mẫu có thể dựa vào để phân hoạch xa hơn.
- Không còn mẫu nào cho nhánh.

Tuy nhiên, nếu chúng ta không lựa chọn được thuộc tính nào để phân loại hợp lý tại mỗi nút, cây quyết định sau khi xây dựng có thể rất phức tạp. Vì thế người ta thường sử dụng hai cách sau để xây dựng cây quyết định phù hợp:

- Dừng việc phát triển cây sớm hơn bình thường trước khi phân lớp hoàn toàn tập dữ liệu huấn luyện.
- Sử dụng một số kỹ thuật "cắt", "tỉa" cây phù hợp.

Xây dựng Cây quyết định dựa trên Entropy

Khái niệm Entropy [5] của một tập S được định nghĩa trong lý thuyết thông tin là số lượng mong đợi các bit cần thiết để mã hóa thông tin một thành phần rút ra một cách ngẫu nhiên từ tập S về lớp của nó. Đối với trường hợp tối ưu, mã sẽ có độ dài ngắn nhất. Theo lý thuyết thông tin, một mã có độ dài tối ưu sẽ được gán $-\log_2 p$ bits cho một thông điệp có xác suất là p.

Đối với trường hợp tập S là tập mẫu thì mỗi thành phần của tập S là một mẫu. Mỗi mẫu thuộc một lớp nào đó hay nói cách khác là có một giá trị phân loại. Giả sử các mẫu trong tập S thuộc về một lớp trong c lớp, trong đó lớp thứ i ($1 \le i \le c$) có tỉ lệ là p_i.

Độ đo Entropy của tập mẫu S được định nghĩa bởi công thức sau:

$$\text{Entropy(S)} = \sum_{i=1}^{C} - p_i \log_2 p_i$$

Về bản chất, độ đo Entropy sẽ phản ánh mức độ không đồng nhất của tập mẫu S. Entropy là một độ đo để đo độ pha trộn dữ liệu của một tập mẫu, Entropy càng nhỏ thì tập mẫu càng đồng nhất.

Từ đó, ta định nghĩa lượng thông tin thu thêm và ký hiệu là Gain cho một phép đo hiệu suất phân loại các mẫu của một thuộc tính. Cụ thể hơn, Gain(S, A) của thuộc tính A, trên tập S, được định nghĩa như sau:

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Trong đó Values(A) là tập hợp các giá trị có thể có của thuộc tính A, và S_v là tập chứa các mẫu có thuộc tính A mang giá trị v trong tập S. Giá trị Gain(S, A) được sử dụng vào mục đích lựa chọn thuộc tính phân lớp dữ liệu tại mỗi nút trung gian và nút gốc trong quá trình xây dựng cây quyết định. Thuộc tính cho lượng thông tin thu thêm lớn nhất sẽ là thuộc tính được chọn.

Các thuật toán xây dựng cây quyết định dựa trên Entropy có thể tóm tắt như sau:

- Với mỗi thuộc tính bất kỳ A chưa được sử dụng trong quá trình xây dựng cây quyết định, tính Gain (S, A) theo công thức bên trên.
- Chọn một thuộc tính P sao cho giá trị Gain(S, P) có giá trị lớn nhất trong các thuộc tính A kể trên.
- Gán nút tương ứng với thuộc tính P có giá trị Gain lớn nhất.

Xây dựng cây quyết định dựa trên Gini index

Công thức Gini index thường được sử dụng phổ biến hơn Goodness of Split, là phương pháp hướng đến đo lường tần suất một đối tượng dữ liệu ngẫu nhiên trong tập dữ liệu ban đầu được phân loại không chính xác, trên cơ sở đối tượng dữ liệu đã nằm trong một tập con đã được phân ra từ dữ liệu ban đầu, có dán nhãn để thể hiện thuộc tính chung bất kỳ của các đối tường còn lại trong tập con này, giá trị phân loại chính là nhãn của tập con.

Gini index cũng chính là chỉ số đo lường mức độ đồng nhất hay mức độ nhiễu loạn của thông tin. Công thức Gini có thể áp dụng cho cả biến định tính và biến định lượng.

Gini index cho phép chúng ta đánh giá sự tối ưu của từng các phân nhánh thông qua xác định mức độ thuần khiết của từng node trong mô hình cây quyết định. Nếu tất cả các điểm dữ liệu nằm về cũng một lớp thì thể hiện sự đồng nhất không có nhiễu loạn ứng với Gini bằng 0, và sẽ càng lớn nếu các điểm dữ liệu khác biệt nhau và lớn nhất bằng 1.

$$GINI(t) = 1 - \sum_{j} [p(j \mid t)]^2$$

Do hệ số Gini dùng cho thuật toán CART mà CART chỉ giới hạn mỗi lần phân chỉ được 2 nhánh nên giả sử một biến có n thuộc tính thì sẽ có 2^n-2 tập con các cách phân nhánh trên cây quyết định.

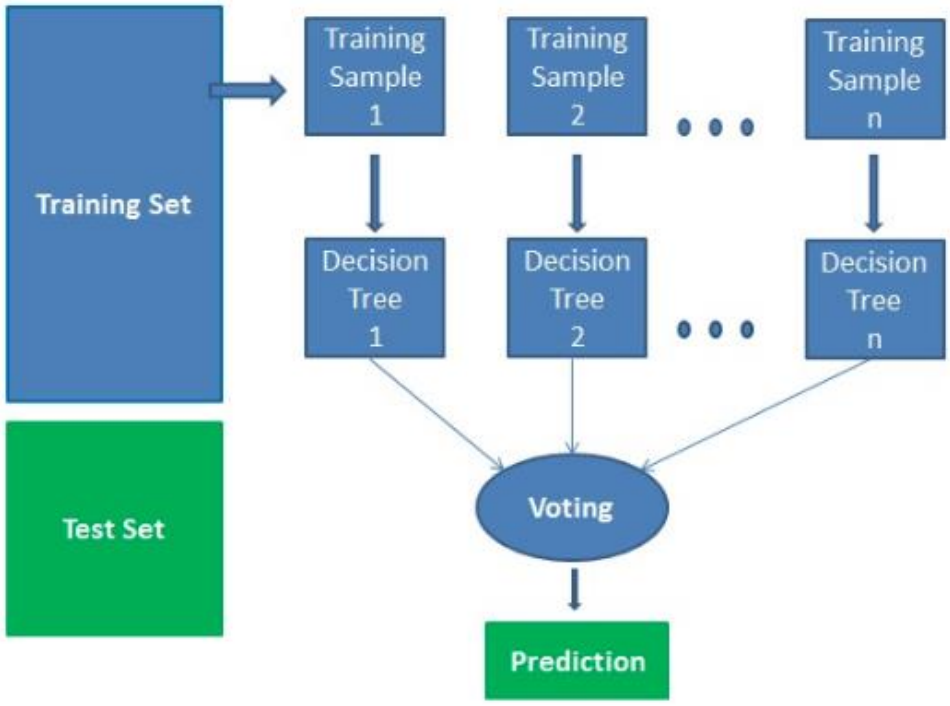
Công thức Gini để tính độ đồng nhất của một nút, vậy khi chúng ta có nhiều cách phân nhánh, mỗi cách có thể phân ra một số nút nhất định, tức có thể chia tập dữ liệu thành các tập con khác nhau theo các giá trị của biến dữ liệu, lúc này chúng ta có thêm công thức thứ 2 để tìm ra cách chia tối ưu nhất.

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

Với n_i là số dữ liệu có trung nút con, n là số dữ liệu trong nút cha. Hệ số $GINI_{split}$ càng nhỏ tức cách phân nhánh càng tốt.

1.3.2. Thuật toán Rừng ngẫu nhiên

Rừng ngẫu nhiên [24] là một thuật toán học có giám sát. Như bạn có thể thấy từ tên của nó, nó tạo ra một khu rừng một cách ngẫu nhiên. "Khu rừng" mà ta tạo ra là một tập hợp các cây quyết định. Ý tưởng chính của phương pháp là sự kết hợp của các mô hình học tập làm tăng kết quả chung.



(Nguồn: Internet)

Hình 1.4: Thuật toán rừng ngẫu nhiên

Rừng ngẫu nhiên được đề xuất vào năm 2001 [2]. Đây là thuật toán phân loại có kiểm định dựa trên cây quyết định và kỹ thuật Bagging and Bootstrapping đã được cải tiến. Bootstrapping là một phương pháp rất nổi tiếng trong thống kê được giới thiệu bởi Efron vào năm 1979. Phương pháp này được thực hiện như sau: từ một quần thể ban đầu lấy ra một mẫu L = (x_1, x_2,..,x_n) gồm n thành phần để tính toán các tham số mong muốn. Trong các bước tiếp theo lặp lại b lần tạo ra mẫu L_b cũng gồm n phần bằng cách lấy lại mẫu với sự thay thế các thành phần trong mẫu ban đầu sau đó tính toán các tham số mong muốn. Phương pháp Bagging được xem như là một phương pháp tổng hợp kết quả có được từ các bootstraping sau đó huấn luyện mô hình từ các mẫu ngẫu nhiên này và cuối cùng đưa ra dự đoán phân loại dựa vào số phiếu bầu cao nhất của lớp phân loại. Cây quyết định là một sơ đồ phát triển có cấu trúc dạng cây phân nhánh đi từ gốc cho đến lá, giá trị các lớp phân loại của mẫu được đưa vào kiểm tra trên cây quyết định. Mỗi mẫu tương ứng có một đường đi từ gốc (tức là dữ liệu đầu vào) đến lá (tức là các kết quả phân loại dự đoán đầu ra), đường đi này biểu diễn sự phân lớp của mẫu đó. Mỗi sơ đồ cây trong tập mẫu được tạo thành từ tập hợp các dữ liệu huấn luyện được lựa chọn ngẫu nhiên để huấn luyện mô hình phân loại Rừng ngẫu nhiên (mỗi tập mẫu bootstrap sẽ cho ra một cây và n cây tương ứng với n bootstrap). Khi một tập mẫu được rút ra từ tập huấn luyện (bootstrap) với sự thay thế có hoàn lại, thì thông thường có khoảng 1/3 các phần tử không nằm trong mẫu này và vì thế chúng không tham gia vào quá trình huấn. Điều này có nghĩa là chỉ có khoảng 2/3 các phần tử trong tập huấn luyện tham gia vào trong các tính toán để phân loại và 1/3 các phần tử này dùng để kiểm tra sai số. Dữ liệu kiểm tra được sử dụng để ước lượng sai số tạo ra từ việc kết hợp các kết quả phân loại riêng lẻ sau đó được tổng hợp trong mô hình Rừng ngẫu nhiên cũng như dùng để ước tính các biến quan trọng.

Rừng ngẫu nhiên chứa một lượng lớn các cây, mỗi cây được phát triển từ các tập huấn luyện được lựa chọn ngẫu nhiên. Hai tham số cần được xác định

trong thuật toán phân loại này là ntree (số lượng cây được phát triển) và mtry (số lượng biến để phân chia tại mỗi node). Số ntree được lựa chọn phụ thuộc vào khoảng thời gian xử lý ngắn nhất để kết quả đạt được độ sai số thấp nhất và mtry biến động từ số biến độc lập tối thiểu (bằng 1) đến số biến độc lập tối đa được sử dụng trong phân loại.

Sau khi mô hình Random Forest được tạo thành, mỗi kết quả của các boostrap trong tập hợp sẽ bỏ phiếu cho lớp phổ biến nhất và cho ra một kết quả phân loại. Mô hình được tạo thành dựa vào phân loại có số phiếu bầu nhiều nhất của mỗi sơ đồ cây quyết định ntree.

1.4. Bug Tracking System

Theo dõi lỗi [7] là một quá trình được sử dụng bởi các nhân viên đảm bảo chất lượng và lập trình viên để theo dõi các vấn đề phần mềm và phần cứng. Một hệ thống theo dõi lỗi thường đưa ra để lưu trữ thông tin về lỗi đã thông báo.

Hệ thống theo dõi lỗi (BTS) là một hệ thống kiểm tra sự cố đặc biệt được sử dụng để theo dõi các lỗi phần mềm. Các trường (thuộc tính) được sử dụng để theo dõi trạng thái của vấn đề trong khi mô tả bằng văn bản được sử dụng để mô tả vấn đề. Các BTS nói chung nhằm mục đích nâng cao chất lượng của các sản phẩm phần mềm.

Có nhiều hệ thống BTS khác nhau, mỗi hệ thống sẽ tập trung vào một hoặc nhiều thuộc tính đặc trưng của dữ liệu mà hệ thống đó hướng đến [19], [20], [21], [22]. Một trong những tính năng quan trọng nhất của BTS là khả năng cho phép truy xuất dữ liệu. Tất cả BTS tối thiểu đều phải có giao diện Web thông qua HTML, nhưng việc hỗ trợ truy xuất tự động sẽ thuận tiện hơn nhiều. Thông tin cung cấp từ BTS thường dưới dạng text, trong khi một số BTS cung cấp thông tin dưới dạng đồ thị thông qua một số dạng ký tự đặc biệt. Một số BTS không cung cấp thông tin với mô tả cụ thể mà dưới dạng data model, đòi hỏi người dùng phải chuyển đổi sang dạng có thể sử dụng.

Việc theo dõi mối quan hệ phụ thuộc giữa các báo cáo lỗi rất quan trọng, đôi khi một lỗi liên quan đến lỗi khác không thể giải quyết nếu mối liên hệ với lỗi đó chưa được giải quyết hoặc tác động. Một số hệ thống BTS cho phép tìm kiếm theo các từ khóa, trong khi một số khác phải sử dụng bộ lọc có sẵn để tìm kiếm dữ liệu lỗi. Hệ thống cho phép tìm kiếm theo từ khóa thì thuận tiện hơn cho cho một hệ thống tự động.

Các cách truy xuất dữ liệu từ BTS:

- Dùng API.
- Truy xuất thông qua công cụ để lấy nội dung từ HTML. Truy xuất từ HTML có một vấn đề cần giải quyết đó là có thể với cùng một cấu trúc nhưng dữ liệu có thể được thể hiện với các giao diện khác nhau. Tuy nhiên tất cả báo cáo lỗi từ một nguồn BTS sẽ có cấu trúc giống nhau, nếu một thuật toán truy xuất được 1 báo cáo lỗi thì có thể truy xuất tất cả các báo cáo lỗi còn lại.

1.5. Thư viện Scikit-learn

Là một thư viện mạnh mẽ có thể mang các thuật toán học máy (machine learning) vào trong một hệ thống thích hợp nhất. Thư viện này tích hợp rất nhiều thuật toán hiện đại và cố điển hỗ trợ việc học và tiến hành đưa ra các giải pháp hữu ích cho bài toán học máy một cách đơn giản [27].

Scikit-learn (Sklearn) [30] là thư viện mạnh mẽ nhất dành cho các thuật toán học máy được viết trên ngôn ngữ Python. Thư viện cung cấp một tập các công cụ xử lý các bài toán machine learning và statistical modeling gồm: classification, regression, clustering, và dimensionality reduction.

Thư viện được cấp phép bản quyền chuẩn FreeBSD và chạy được trên nhiều nền tảng Linux. Scikit-learn được sử dụng như một tài liệu để học tập.

Để cài đặt scikit-learn trước tiên phải cài thư viện SciPy (Scientific Python). Những thành phần gồm:

- Numpy: Gói thư viện xử lý dãy số và ma trận nhiều chiều
- SciPy: Gói các hàm tính toán logic khoa học
- Matplotlib: Biểu diễn dữ liệu dưới dạng đồ thị 2 chiều, 3 chiều
- IPython: Notebook dùng để tương tác trực quan với Python
- SymPy: Gói thư viện các kí tự toán học
- Pandas: Xử lý, phân tích dữ liệu dưới dạng bảng

Những thư viện mở rộng của SciPy thường được đặt tên dạng SciKits. Như thư viện này là gói các lớp, hàm sử dụng trong thuật toán học máy thì được đặt tên là scikit-learn.

Scikit-learn hỗ trợ mạnh mẽ trong việc xây dựng các sản phẩm. Nghĩa là thư viện này tập trung sâu trong việc xây dựng các yếu tố: dễ sử dụng, dễ code, dễ tham khảo, dễ làm việc, hiệu quả cao.

Mặc dù được viết cho Python nhưng thực ra các thư viện nền tảng của scikit-learn lại được viết dưới các thư viện của C để tăng hiệu suất làm việc.

1.6. Các công trình nghiên cứu trong nước

1.6.1. Two-Phase Defect Detection Using Clustering and Classification Methods

Quản lý lỗi tự chủ của hệ thống mạng và hệ thống phân tán là một vấn đề nghiên cứu đầy thách thức và thu hút nhiều hoạt động nghiên cứu. Giải quyết vấn đề này phụ thuộc nhiều vào kiến thức chuyên môn và các công cụ hỗ trợ để giám sát và phát hiện các khuyết tật trong hệ thống mạng một cách tự động. Các hoạt động nghiên cứu gần đây đã tập trung vào các kỹ thuật máy học để xem xét kỹ lưỡng dữ liệu đầu ra của hệ thống và khai thác các sự kiện bất thường cũng như phát hiện các khiếm khuyết trong hệ thống. Bài báo này đề xuất phát hiện lỗi qua hai bước cho mạng và hệ thống phân tán sử dụng phương pháp phân nhóm và phân loại dữ liệu log. Cách tiếp cận tận dụng lợi thế của phương pháp phân nhóm K-mean để thu được các thông báo bất thường và phương pháp rừng ngẫu nhiên để phát hiện mối quan hệ của thông

báo bất thường và các khiếm khuyết hiện có. Một số thử nghiệm đã đánh giá hiệu suất của phương pháp này bằng cách sử dụng dữ liệu log nhật ký của Hệ thống tệp phân tán Hadoop (HDFS) và dữ liệu báo cáo lỗi của Hệ thống theo dõi lỗi (BTS) [6].

1.6.2. An automated fault detection system for communication networks and distributed systems

Mô hình này được đặc trưng bởi khả năng khai thác tài nguyên kiến thức lỗi tại các kho lưu trữ trực tuyến khác nhau, các sự kiện nhật ký và các thông số trạng thái từ hệ thống được giám sát, áp dụng phân tích lỗi và sự kiện bằng các phương pháp lọc để đánh giá sự kiện và dự báo lỗi. Hệ thống chứa một mô hình dữ liệu lỗi để thu thập các báo cáo lỗi, một tính năng và phương pháp lọc ngữ nghĩa để tương quan các nhật ký lỗi và phương pháp học máy để đánh giá mức độ nghiêm trọng, mức độ ưu tiên và mối quan hệ của các nhật ký lỗi và dự báo các lỗi nghiêm trọng sắp tới của hệ thống được giám sát [7].

1.6.3. Áp dụng thuật toán phân loại Random Forest để xây dựng bản đồ sử dụng đất/thảm phủ tỉnh Đắk Lắk dựa vào ảnh vệ tinh Landsat 8 OLI

Nghiên cứu nay đã sử dụng phương pháp Random Forest chạy trong môi trường phần mềm mã nguồn mở R để xây dựng bản đồ thảm phủ/sử dụng đất (LULC) khu vực tỉnh Đắk Lắk dựa vào ảnh vệ tinh đa phổ Landsat 8 OLI (Operational Land Imager). Hai tham số quan trọng được sử dụng trong phương pháp Random Forest la số cây quyết đinh (ntree) và số lượng biến dùng để phân chia tại mỗi nút (mtry) đã được phát hiện dựa trên các thử nghiệm với các giá tri khác nhau để tìm ra giá trị phù hợp nhất. Kết quả đã xác định được số lượng cây quyết định phù hợp tham gia vào quá trình phân loại là 250 và số lượng biến dùng để phân chia tại mỗi nút là 4 biến. Sử dụng 2 tham số nay để phân loại ảnh Landsat 8 OLI thành 10 loại thảm phủ/sử dụng đất của tỉnh Đắk Lắk bằng thuật toán Random Forest, kết quả phân loại đạt độ chính xác tổng thể là 90.36 % với hệ số Kappa là 0.8873 [2].

1.6.4. Nghiên cứu một số thuật toán học máy để phân lớp dữ liệu và thử nghiệm

Tác giả Đỗ Thị Lương đã nghiên cứu các kỹ thuật học máy để giải quyết bài toán phân lớp dữ liệu nói chung và thử nghiệm đánh giá hiệu năng của chúng trên bộ dữ liệu KDD cup 99 [4].

1.7. Các công trình nghiên cứu ngoài nước

1.7.1. VAX/VMS Event monitoring and analysis

Nghiên cứu này sử dụng một trong những bộ dữ liệu lớn nhất và là cuộc điều tra chuyên sâu nhất về quá trình giám sát được thực hiện cho đến nay, để kiểm tra việc theo dõi và phân tích sự kiện. 2,35 triệu sự kiện từ 193 hệ thống VAX/VMS bao gồm 335 năm máy đã được sử dụng. Các ví dụ được trình bày cho thấy các thiếu sót trong việc giám sát làm phức tạp các phân tích, tiêu tốn thêm thời gian và có nhiều khả năng đưa ra kết luận không chính xác. Ví dụ: việc xử lý không chính xác thời gian không có thật sẽ thay đổi thời gian trung bình giữa các nhóm sự kiện theo một thứ tự độ lớn. Một quy trình phân tích để xác định các lỗi được cung cấp, cùng với các quy tắc thiết kế để tạo ra các bản ghi lỗi với chất lượng tốt hơn [8].

1.7.2. Using Secondary Knowledge to Support Decision Tree Classification of Retrospective Clinical Data

Trong nghiên cứu này, đề tài mô tả việc áp dụng phương pháp khai thác dữ liệu để xây dựng mô hình dự đoán về mức độ trầm trọng của cơn hen cho bệnh nhi tại khoa cấp cứu. Những khó khăn trong việc xây dựng một mô hình như vậy buộc tác giả phải nghiên cứu các chiến lược thay thế để phân tích và xử lý dữ liệu hồi cứu. Bài báo này mô tả quá trình này cùng với cách tiếp cận để khai thác dữ liệu lâm sàng hồi cứu bằng cách kết hợp kiến thức chuyên gia bên ngoài được chính thức hóa (nguồn kiến thức thứ cấp) vào nhiệm vụ phân loại. Kiến thức này được sử dụng để phân vùng dữ liệu thành một số tập hợp nhất quán, trong đó mỗi tập hợp được mô tả rõ ràng theo nguồn kiến thức thứ cấp. Các cá thể từ mỗi tập hợp sau đó được phân

loại theo cách phù hợp với các đặc điểm của tập hợp cụ thể. Tác giả trình bày phương pháp luận của mình và phác thảo một tập hợp các kết quả kinh nghiệm chứng minh một số ưu điểm và một số hạn chế trong cách tiếp cận của đề tài [9].

1.7.3. Designing a hierarchical neural network based on fuzzy clustering for fault diagnosis of the Tennessee–Eastman process

Bài báo này đề xuất một mạng nơ-ron nhân tạo phân cấp (HANN) để cô lập các lỗi của quy trình Tennessee – Eastman (TEP). Quy trình TEP là mô phỏng của một nhà máy hóa chất do Công ty hóa chất Eastman tạo ra để cung cấp một quy trình công nghiệp thực tế để đánh giá các phương pháp giám sát và kiểm soát quá trình. Bước đầu tiên trong việc thiết kế HANN là chia không gian các mẫu lỗi thành một vài không gian con thông qua sử dụng thuật toán phân cụm C-mean mờ. Đối với mỗi không gian con của các mẫu lỗi, một mạng nơ-ron đặc biệt đã được huấn luyện để chẩn đoán các lỗi của không gian con đó. Một mạng giám sát đã được phát triển để quyết định một trong các mạng nơ-ron đặc biệt được kích hoạt. Về vấn đề này, mỗi mạng nơ-ron trong HANN đề xuất đã được giao một nhiệm vụ cụ thể, do đó, thủ tục được đề xuất có thể được gọi là HANN hướng theo nhiệm vụ (DOHANN). Cấu trúc của mạng dựa trên mạng đa lớp (MLP). Mô phỏng quá trình Tennessee – Eastman (TE) đã được sử dụng để tạo dữ liệu đào tạo và kiểm tra cần thiết. Hiệu suất của phương pháp đã phát triển đã được đánh giá và so sánh với hiệu suất của mạng nơ-ron đơn thông thường (SNN) cũng như kỹ thuật phân tích thành phần chính động (DPCA). Kết quả mô phỏng chỉ ra rằng DOHANN chẩn đoán lỗi TEP tốt hơn đáng kể so với các phương pháp SNN và DPCA. Việc đào tạo từng mạng MLP cho mô hình DOHANN cần ít thời gian sử dụng máy tính hơn so với mô hình SNN. Điều này là do các MLP có cấu trúc đơn giản hơn được sử dụng bởi phương pháp DOHANN đã phát triển [11].

1.7.4 Fault Detection and Diagnosis for Solar-Powered Wireless Mesh Networks Using Machine Learning

Sự phức tạp vốn có của Mạng lưới không dây (WMN) khiến các nhiệm vụ quản lý và cấu hình trở nên khó khăn, đặc biệt là để phát hiện và chẩn đoán lỗi. Ngoài ra, việc kiểm tra thủ công cực kỳ tốn kém và đòi hỏi lực lượng lao động có tay nghề cao, do đó trở nên không thực tế khi vấn đề quy mô. Để giải quyết vấn đề này, bài báo này đề xuất một giải pháp sử dụng các kỹ thuật máy học để phát hiện và chẩn đoán lỗi tự động (FDD) trên Mạng lưới không dây (WMN) chạy bằng năng lượng mặt trời. Tác giả đã sử dụng phương pháp Khám phá tri thức trong Cơ sở dữ liệu (KDD) và một từ điển được xác định trước về các lỗi dựa trên kinh nghiệm trước đây của các tác giả về việc triển khai WMN. Sau đó, vấn đề đã được giải quyết như một bài toán phân loại mẫu. Một số thuật toán phân loại đã được đánh giá, chẳng hạn như Naive Bayes, Máy hỗ trợ vectơ (SVM), Cây quyết định, k-Nearest Neighbors (k-NN) và SVM đã đưa ra kết quả tốt nhất, đạt độ chính xác tổng thể 90,59% trong quá trình đào tạo và hơn 85% trong các bài kiểm tra xác nhận [17].

1.7.5 Fault Detection for Cloud Computing Systems with Correlation Analysis

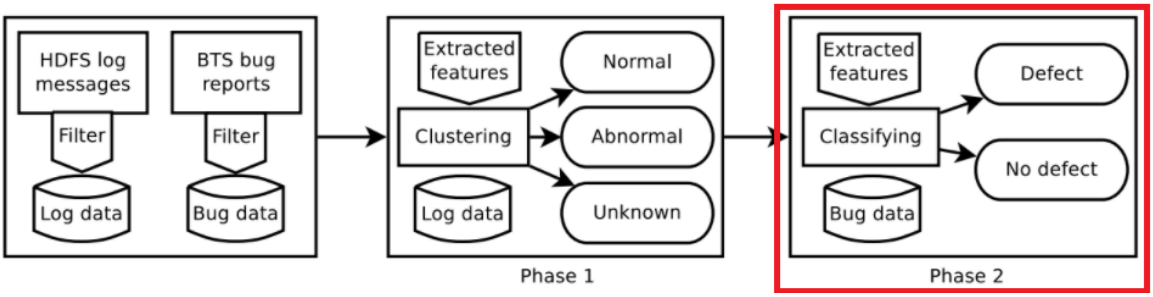
Môi trường điện toán đám mây động quy mô lớn đã đặt ra những thách thức lớn đối với việc chẩn đoán lỗi trong các ứng dụng Web. Thứ nhất, khối lượng công việc biến động khiến các mô hình ứng dụng truyền thống thay đổi theo thời gian. Hơn nữa, việc mô hình hóa các hành vi của các ứng dụng phức tạp luôn đòi hỏi kiến thức miền khó có được. Cuối cùng, việc quản lý các ứng dụng quy mô lớn theo cách thủ công là không thực tế đối với các nhà khai thác. Bài báo này giải quyết những vấn đề trên và đề xuất một phương pháp chẩn đoán lỗi tự động cho các ứng dụng Web trong điện toán đám mây. Tác giả đề xuất một phương pháp phân cụm gia tăng trực tuyến để nhận ra các mẫu hành vi truy cập và sử dụng CCA để mô hình hóa mối tương quan giữa khối lượng công việc và các chỉ số về hiệu suất ứng dụng/ sử dụng tài nguyên trong một mẫu hành vi truy cập cụ thể. Phương pháp của tác giả phát hiện sự bất thường bằng cách khám phá sự thay đổi đột ngột của các hệ số tương quan với biểu đồ kiểm soát EWMA, sau đó xác định các chỉ số đáng ngờ bằng cách sử dụng phương

pháp lựa chọn tính năng kết hợp ReliefF và SVM-RFE. Tác giả xác nhận phương pháp của mình bằng cách đưa các lỗi điển hình vào TPC-W, một điểm chuẩn cho tiêu chuẩn công nghiệp và kết quả thử nghiệm chứng minh rằng nó có thể phát hiện hiệu quả các lỗi điển hình [15].

Chương 2 – PHƯƠNG PHÁP PHÂN LOẠI LỖI MẠNG

2.1. Mô hình Two-Phase Defect Detection

Quá trình phân loại lỗi mạng thông qua mô hình Two-Phase Defect Detection được thể hiện qua mô hình như hình bên dưới.



Hình 2.1: Mô hình Two-Phase Defect Detection

Trong mô hình trên, dữ liệu log sẽ được thu thập từ các thiết bị trên một hệ thống (ở đây là HDFS logs) [6], sau đó dữ liệu log sẽ được phân loại dựa vào phương pháp gom cụm để chia các dữ liệu log thành 3 loại chính:

- Bình thường
- Bất bình thường
- Không xác định

Lỗi được thu thập từ các BTS thông qua công cụ thu thập dữ liệu từ giao diện HTML. Các lỗi sau đó được lưu thành một Bug Database với một định dạng chuẩn hóa duy nhất. Tập dữ liệu lỗi sẽ được sử dụng để huấn luyện máy học thông qua thuật toán Rừng ngẫu nhiên và kỹ thuật tf x idf để phân loại thành các dạng lỗi nhỏ, lỗi thông thường và lỗi đặc biệt nghiêm trọng.

Các dữ liệu log file bất bình thường sau khi được gom cụm tại Phase 1 sẽ được đưa vào mô hình phân loại lỗi như một đầu vào, từ đó phân loại lỗi trên có ảnh hưởng đến hệ thống hay không dưới dạng 3 lớp như sau:

- Lỗi thông thường
- Lỗi nhỏ
- Lỗi nghiêm trọng

Mô hình phân loại lỗi trong luận văn chính là Phase 2 của mô hình **Two-Phase Defect Detection.**

2.2. Mô hình dữ liệu lỗi

Để tích hợp các dữ liệu lỗi từ nhiều nguồn BTS khác nhau ta phải đưa các dữ liệu lỗi trên về cùng một mô hình thống nhất cho các lỗi. Các lỗi trên các nền tảng BTS khác nhau cũng có nhiều trường thuộc tính dữ liệu tương tự và có thể chia làm hai nhóm chính sau đây:

Trường dữ liệu quản trị thường được biểu diễn dưới dạng ngữ nghĩa rất chính xác như: ID, mức độ nghiêm trọng, báo cáo, tóm tắt.

Các mô tả chi tiết về lỗi hoặc các thảo luận về lỗi thường được thể hiện dưới dạng tệp đính kèm hoặc văn bản tự do.

Một ví dụ về các thông tin một lỗi trích xuất từ Bugzilla:

- ID: 1687606

- BTS: bugzilla

- Người báo cáo: vickyrathod055

- Thời gian báo lỗi: 01/19/2021 21:09

- Thời gian cập nhật gần nhất: 07/24/2011 06:31

- Mức độ nghiêm trọng: S3

- Mức độ ưu tiên: P2

- Trạng thái: UNCONFIRMED

- Nội dung: Loading a site hangs the page while loading a resource that failed at the HTTP2 server push

- Software: Firefox

- Component: Netwworking

- Nền tảng: mac

- Mô tả lỗi như sau:

"*User Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36*

Steps to reproduce:

Open the developer tool and go to https://factory.anntaylor.com

Search for the failed asset at HTTP2 server push (most common is foundation.css) (you'll see a 200)

Do a refresh and do a search for the same asset (foundation.css), now you'll see that Firefox hangs and take a long time to load that asset.

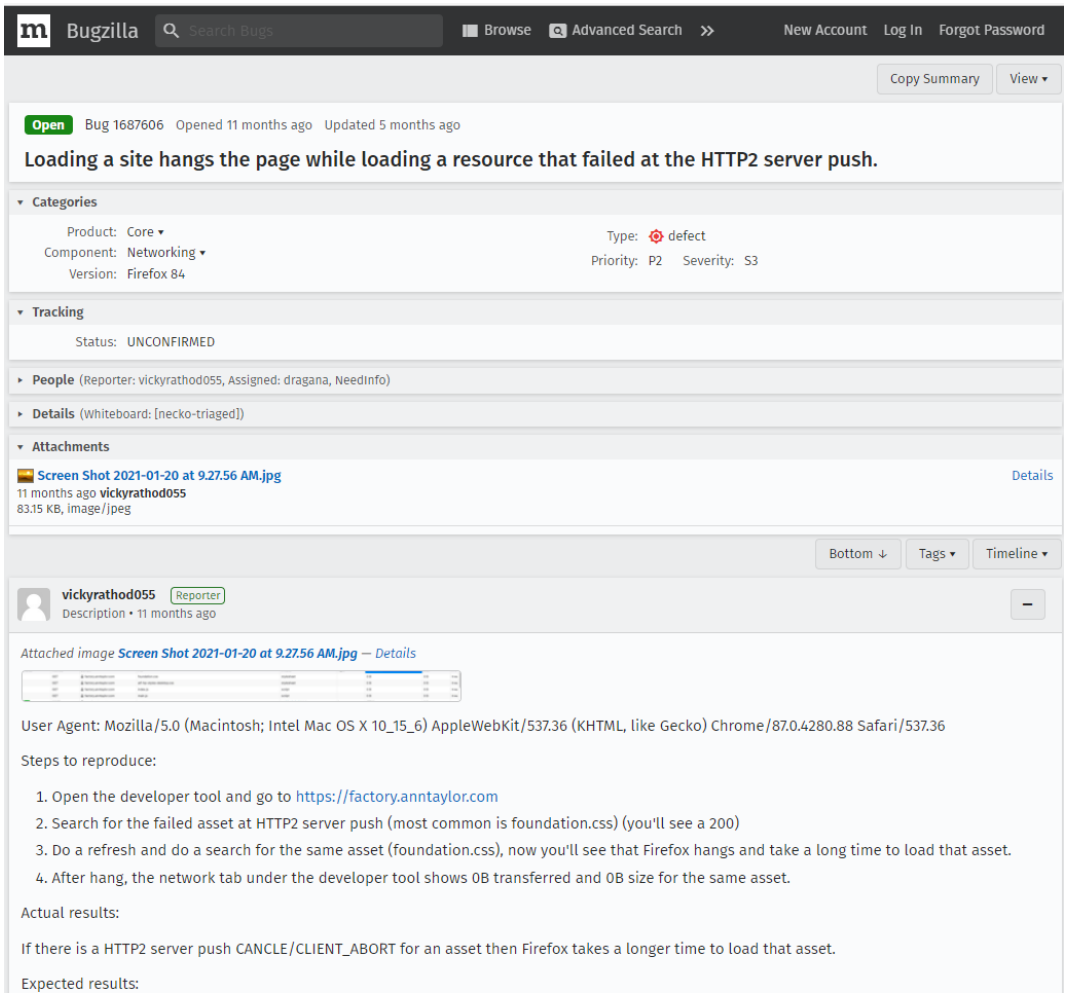
After hang, the network tab under the developer tool shows 0B transferred and 0B size for the same asset.

Actual results:

If there is a HTTP2 server push CANCLE/CLIENT_ABORT for an asset then Firefox takes a longer time to load that asset.

Expected results:

Loading same site on Crome works fine.”



Hình 2.2: Giao diện báo cáo lỗi trên Bugzilla

Một số thuộc tính đặc trưng của lỗi như:

+ ID: để xác định duy nhất một phiên bản lỗi.

+ Mức độ nghiêm trọng: hầu hết các BTS đều có phân loại mức độ nghiêm trọng của lỗi tuy nhiên nó là khác nhau giữa các BTS, việc cần làm là đồng bộ các giá trị từ các BTS khác nhau về cùng một loại cho thuộc tính trên.

+ Thuộc tính trạng thái của một lỗi: là lỗi mới hay lỗi đã được giải quyết, xác minh và đóng.

+ Các mô tả dạng văn bản được mô hình hóa dưới dạng tệp đính kèm, mỗi tệp đính kèm thuộc về chính xác một lỗi nào đó. Phần này có thể có các thuộc tính như loại phức tạp, kiểu sự cố, phân vùng lỗi…

2.3. Sử dụng phương pháp tf x idf để lọc nội dung quan trọng từ nội dung mô tả lỗi

TF-IDF (Term Frequency – Inverse Document Frequency) [7] là 1 kĩ thuật sử dụng trong khai phá dữ liệu văn bản. Trọng số này được sử dụng để đánh giá tầm quan trọng của một từ trong một văn bản. Giá trị cao thể hiện độ quan trọng cao và nó phụ thuộc vào số lần từ xuất hiện trong văn bản nhưng bù lại bởi tần suất của từ đó trong tập dữ liệu.

TF: Term Frequency (Tần suất xuất hiện của từ) là số lần từ xuất hiện trong văn bản. Vì các văn bản có thể có độ dài ngắn khác nhau nên một số từ có thể xuất hiện nhiều lần trong một văn bản dài hơn là một văn bản ngắn. Như vậy, term frequency thường được chia cho độ dài văn bản (tổng số từ trong một văn bản).

$$tf(t,d) = \frac{f(t,d)}{\max\{f(w,d): w \in d\}}$$

Trong đó:

- tf(t, d): tần suất xuất hiện của từ t trong văn bản d

- f(t,d): số lần xuất hiện của từ t trong văn bản d

- $\max\{f(w,d): w \in d\}$: số lần xuất hiện của từ có số lần xuất hiện nhiều nhất trong văn bản d

IDF: Inverse Document Frequency (Nghịch đảo tần suất của văn bản) giúp đánh giá tầm quan trọng của một từ. Khi tính toán TF, tất cả các từ được coi như có độ quan trọng bằng nhau. Nhưng một số từ như “is”, “of” và “that” thường

xuất hiện rất nhiều lần nhưng độ quan trọng là không cao. Như thế chúng ta cần giảm độ quan trọng của những từ này xuống.

$$idf(t, D) = log \frac{|D|}{|\{d \in D: t \in d\}|}$$

Trong đó:

- idf (t, D): giá trị idf của từ t trong tập văn bản

- |D|: Tổng số văn bản trong tập D

- |{d ∈ D: t ∈ d}|: thể hiện số văn bản trong tập D có chứa từ t.

Cụ thể, chúng ta có công thức tính TF-IDF hoàn chỉnh như sau:

TF-IDF(t, d, D) = tf(t, d) x idf(t, D)

Khi đó những từ có giá trị TF-IDF cao là những từ xuất hiện nhiều trong văn bản này, và xuất hiện ít trong các văn bản khác. Việc này giúp lọc ra những từ phổ biến và giữ lại những từ có giá trị cao (từ khoá của văn bản đó).

2.4. Sử dụng thuật toán phân lớp Rừng ngẫu nhiên thông qua bộ thư viện Scikit-learn

Các tham số trong Rừng Ngẫu nhiên được sử dụng để tăng khả năng dự đoán của mô hình hoặc để làm cho mô hình nhanh hơn. Luận văn sẽ thảo luận về các siêu tham số của hàm *random forest* gói *sklearn* [30].

Các tham số cần quan tâm trong khi xây dựng Rừng ngẫu nhiên như sau:

n_estimators: int, mặc định=100.

Số lượng cây trong Rừng ngẫu nhiên. Nói chung, số lượng cây cao làm tăng hiệu suất và làm cho các dự đoán ổn định hơn, nhưng nó cũng làm chậm quá trình tính toán.

criterion {"gini", "entropy"}, mặc định="gini"

Chức năng đo lường chất lượng của một cây quyết định được xây dựng. Các tiêu chí được hỗ trợ là “gini” và “entropy”. Lưu ý: tham số này dành riêng cho cây quyết định.

max_depth: int, mặc định=None

Chiều sâu tối đa của cây. Nếu không có, thì các nút được mở rộng cho đến khi tất cả các lá đều thuần túy hoặc cho đến khi tất cả các lá chứa ít hơn mẫu *min_samples_split*.

min_samples_split, kiểu dữ liệu là int hoặc float, mặc định=2

Số lượng mẫu tối thiểu cần thiết để tách một nút nội bộ một cây quyết định:

- Nếu kiểu dữ liệu là int, thì hãy coi *min_samples_split* là số mẫu nhỏ nhất cho mỗi lẫn tách.

- Nếu kiểu dữ liệu là float, thì min_samples_split là một phân số và (*min_samples_split* * *n_samples*) là số lượng mẫu tối thiểu cho mỗi lần tách.

min_samples_leaf: kiểu dữ liệu là int hoặc float, mặc định=1

Số lượng mẫu tối thiểu cần thiết có ở một nút lá. Một điểm phân tách ở bất kỳ độ sâu nào sẽ chỉ được xem xét nếu nó để lại ít nhất các mẫu huấn luyện *min_samples_leaf* trong mỗi nhánh. Điều này có thể có tác dụng làm mịn mô hình.

Nếu kiểu dữ liệu là int, thì hãy coi *min_samples_leaf* là số mẫu nhỏ nhất tại một lá.

Nếu kiểu dữ liệu là float, thì min_samples_leaf là một phân số và (*min_samples_leaf* * *n_samples*) là số lượng mẫu tối thiểu cho mỗi lá.

max_features{“auto”, “sqrt”, “log2”}, kiểu dữ liệu là int hoặc float, mặc định=”auto”

Số lượng các thuộc tính cần xem xét khi tìm kiếm sự phân chia tốt nhất:

- Nếu kiểu dữ liệu là int, thì hãy xem xét max_features các thuộc tính tại mỗi lần phân chia.

- Nếu float, thì max_features là một phân số và (max_features * n_features) các thuộc tính được xem xét ở mỗi lần tách.

- Nếu "auto", thì max_features = sqrt(n_features).

- Nếu "sqrt" thì max_features = sqrt (n_features) (giống như "auto").

- Nếu "log2", thì max_features = log2 (n_features).

- Nếu không, thì max_features = n_features.

max_leaf_nodes, mặc định=None

Xây dựng cây quyết định với max_leaf_nodes nút lá tối đa. Nếu Không thì không giới hạn số nút lá.

bootstrap, mặc định=True

Các mẫu bootstrap có được sử dụng khi xây dựng cây hay không. Nếu là False, toàn bộ tập dữ liệu được sử dụng để xây dựng từng cây.

oob_score, mặc định=False

Có sử dụng các mẫu ngoài túi để ước tính điểm tổng quát hay không. Chỉ khả dụng nếu bootstrap = True.

max_samples: kiểu dữ liệu là int hoặc float, mặc định=None

Nếu bootstrap là True, thì số lượng mẫu sẽ lấy từ tập mẫu X để xây dựng cây quyết định, trong đó:

- Nếu None (default), thì lấy X.shape [0] mẫu.

- Nếu int, thì lấy max_samples mẫu.

- Nếu float, thì lấy max_samples * X.shape [0] mẫu. Do đó, max_samples phải nằm trong khoảng (0.0, 1.0].

2.5. Sử dụng tf x idf trong thư viện Scikit-learn

Bộ thư viện Scikit-learn hỗ trợ áp dụng phương pháp tf x idf thông qua hàm ***sklearn.feature_extraction.text.TfidfVectorizer***, các tham số dưới đây được áp dụng để sử dụng hiệu quả phương pháp trên trong luận văn.

input{'filename', 'file', 'content'}, default='content'

Nếu *'filename'*, chuỗi được truyền dưới dạng đối số để phù hợp với mong đợi là danh sách các tên tệp cần đọc để tìm nạp nội dung phân tích.

Nếu là *'file'*, các mục trình tự phải có phương thức *"read"* được gọi để tìm nạp các byte trong bộ nhớ.

Nếu là *'content'*, đầu vào được mong đợi là một chuỗi các mục có thể là kiểu chuỗi hoặc byte.

lowercasebool, default=True

Chuyển đổi tất cả các ký tự thành chữ thường trước khi mã hóa.

max_df: float or int, default=1.0

Khi xây dựng tập từ vựng, hãy bỏ qua các thuật ngữ có tần suất xuất hiện trong tài liệu cao hơn ngưỡng nhất định. Nếu float trong phạm vi [0.0, 1.0].

min_df float or int, default=1

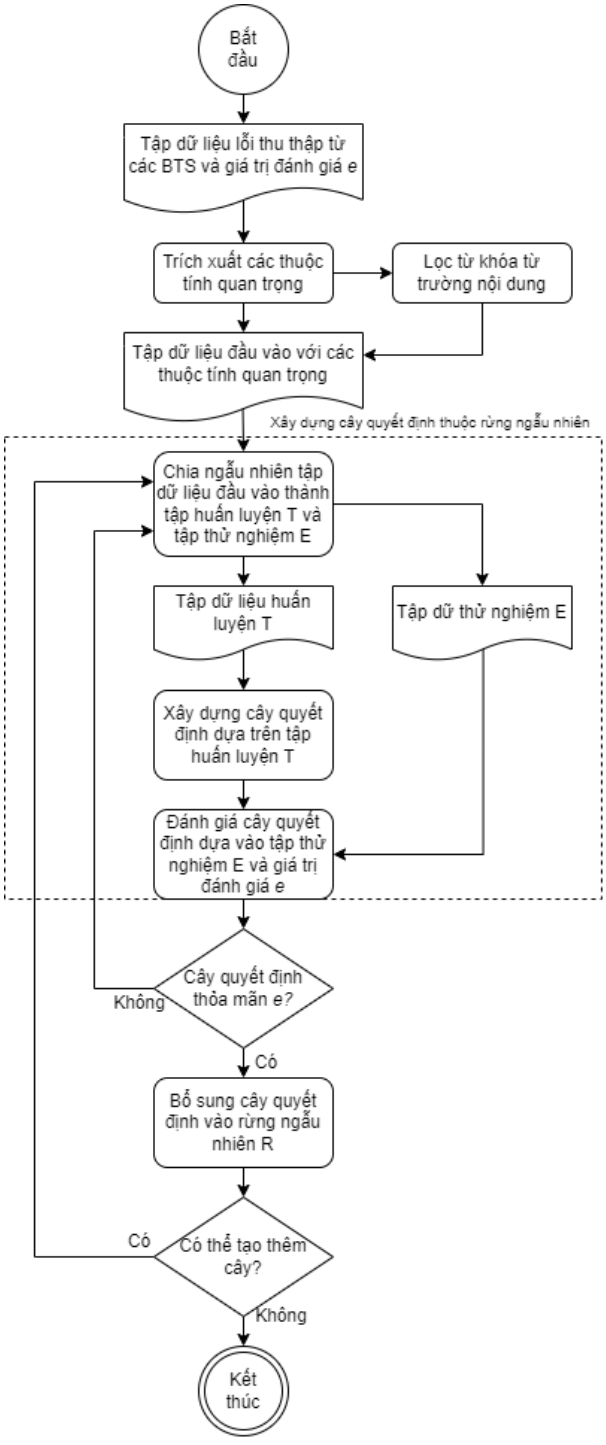
Khi xây dựng tập từ vựng, hãy bỏ qua các thuật ngữ có tần suất xuất hiện trong tài liệu thấp hơn ngưỡng nhất định. Nếu float trong phạm vi [0.0, 1.0].

use_idf: bool, default=True

Cho phép tính lại inverse-document-frequency. Nếu False, idf (t) = 1

Chương 3 - XÂY DỰNG MÔ HÌNH PHÂN LOẠI LỖI MẠNG

Quá trình để xây dựng rừng ngẫu nhiên cho tập dữ liệu lỗi mạng được biểu diễn qua lưu đồ giải thuật như sau.



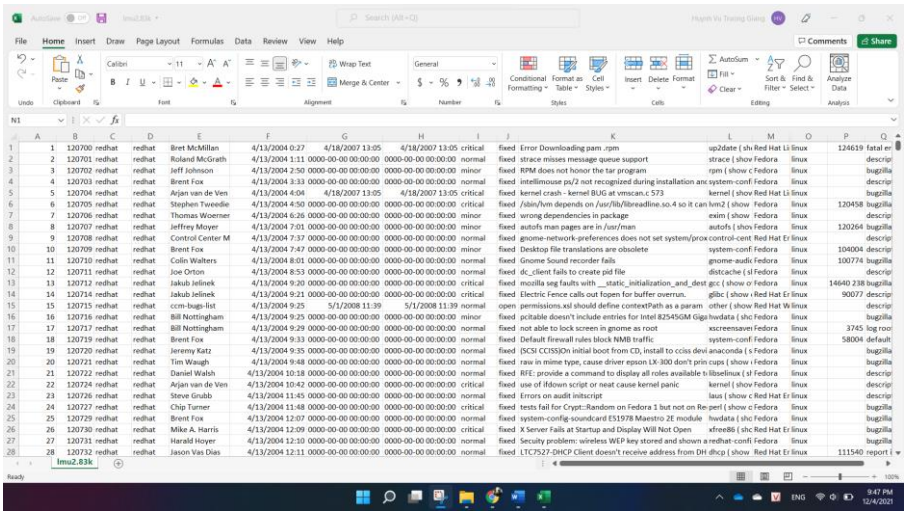
Hình 3.1: Lưu đồ giải thuật xây dựng rừng ngẫu nhiên

3.1. Tập dữ liệu lỗi thu thập từ các BTS

Đối với đề xuất hiện tại của luận văn, dữ liệu lỗi được lấy từ nguồn là các báo cáo lỗi trên các hệ thống Bugtracking System. Đầu vào ở dạng thô bao gồm các thông tin của một lỗi trên một hệ thống BTS như thời gian báo cáo, người báo cáo, nội dung lỗi và các trường thông tin khác. Từ dữ liệu Bugs trên, các file bugs nhận được là dữ liệu thô được trích xuất từ giao diện HTML, ta cần phân loại, định nghĩa lại và đồng bộ các trường, các trạng thái, nội dung đồng nhất để hệ thống xử lý. Một vài số liệu thống kê về thông tin lỗi như sau:

- Số lượng bug files: 5
- Tổng dung lượng: 0.77 GB
- Số lượng bug reports: 483000
- Số lượng gentoo bug report: 150000
- Số lượng redhat bug report: 83000
- Số lượng mozilla bug report: 250000

Dữ liệu lỗi sau khi thi thập từ các hệ thống được lưu dưới dạng .csv như ảnh minh họa.



Hình 3.2: Dữ liệu lỗi sau khi thu thập từ các hệ thống BTS

Dữ liệu lỗi sau khi thu thập từ các BTS cần thực hiện các bước tiền xử lý để loại bỏ các mẫu nhiễu trong tập dữ liệu như các dòng trống, các dòng không có giá trị.

3.2. Trích xuất thuộc tính quan trọng của lỗi

Các lỗi từ tập dữ liệu lỗi được trích xuất để lấy các thuộc tính quan trọng với quá trình phân loại lỗi, các lỗi được trích xuất cụ thể như sau:

Bảng 3.1: Các thuộc tính quan trọng của lỗi

Thuộc tính	Nội dung	Kiểu dữ liệu
Mức độ nghiêm trọng	Mức độ ảnh hưởng	Liệt kê
Trạng thái	Trạng thái của lỗi	Liệt kê
Thành phần	Thành phần xảy ra lỗi	Liệt kê
Phần mềm	Phần mềm xảy ra lỗi	Liệt kê
Nền tảng	Nền tảng xảy ra lỗi	Liệt kê
Từ khóa	Cụm từ riêng biệt	Chuỗi
Mối liên hệ	Mối liên hệ với lỗi khác	Định danh

	State	Content	Component	Software	Platform	Relation	Severity
0	fixed	event-dialog needs to work with the actual cal...	Internal Compone	Calendar	win	296893.0	feature
1	fixed	Wildcards in install locations list not recogn...	Add-ons Manager	Toolkit	win	259006.0	feature
2	fixed	Convert xmlextras tests to use the glue	DOM: Mozilla Ext	Core	all	305949.0	feature
3	fixed	Unable to view deleted comments [as developer]	Developer Pages	addons.mozilla.o	all	NaN	feature
4	fixed	Freeze upon connex to many URLs, & enterin...	General	Firefox	mac	319747.0	critical
...	...	...	...	...	...	...	...
95	fixed	E-Mail Synchronisation with Palm	Palm Sync	MailNews Core Gr	all	36836.0	feature
96	fixed	File - Save As - Template menu item does nothi...	Mail Window Fron	Thunderbird	all	331876.0	feature
97	fixed	In <prefs.js>, "Warning: reference ...	ChatZilla	Other Applicatio	all	NaN	feature
98	fixed	Message search dialog title is "Find in T...	Mail Window Fron	Thunderbird	all	328795.0	feature
99	open	Updates to the content does not update the scr...	Layout: View Ren	Core	win	NaN	normal

Hình 3.3: Dữ liệu lỗi sau khi Import

Trường trạng thái: chứa trạng thái của lỗi trên hệ thống BTS, có hai trạng thái là đã xử lý (fixed) và vẫn còn đang báo cáo (open). Dữ liệu dạng Category.

Trường thành phần: chứa thành phần nơi xảy ra lỗi được báo cáo, nơi lỗi trên xuất hiện trong một hệ thống. Dữ liệu dạng Category.

Trường phần mềm: chứa thông tin phần mềm xảy ra lỗi. Dữ liệu dạng Category.

Trường nền tảng: chứa thông tin nền tảng xảy ra lỗi. Dữ liệu dạng Category.

Trường mối liên hệ: Chứa ID của các lỗi liên quan đến lỗi đang báo cáo. Kiểu dữ liệu dạng định danh.

Trường mức độ nghiêm trọng: mức độ nghiêm trọng của hệ thống, đây là thuộc tính để quyết định việc phân lớp của dữ liệu. Các lỗi đặt biệt nghiêm trọng sẽ được phân loại là gây ảnh hưởng cho hệ thống, các lỗi còn lại phân loại là không ảnh hưởng cho hệ thống.

Trường nội dung: chứa nội dung báo cáo của một lỗi trên hệ thống BTS. Thuộc tính nội dung lỗi được sử dụng để làm đầu vào modul lọc từ khóa, từng từ trong mỗi câu sẽ được tính giá trị tf x idf và sau đó chọn ra từ có giá trị nhất. Trường nội dung lỗi được lưu với dạng văn bảng tiếng anh.

Trường từ khóa: được lọc ra nhờ phương pháp tf x idf để trích xuất ra từ quan trọng nhất trong phần mô tả của một lỗi. Từ khóa này là từ xuất hiện nhiều trong mô tả của lỗi này nhưng lại ít xuất hiện trong mô tả của các lỗi khác. Các từ khóa này là các từ có giá trị cao trong phần mô tả của một lỗi và đã được lọc bỏ các từ phổ biến trong các văn bản. Quá trình trích xuất từ khóa từ nội dung văn bản được thực hiện như các bước tiếp theo sau đây.

Xét tập nội dung lỗi có n phần tử, m là số từ tối đa có thể có của lỗi. Ta xây dựng mảng hai chiều arr[i][j] với nhãn cột là các từ riêng biệt xuất hiện trong toàn bộ tập nội dung lỗi, các hàng là thứ tự các lỗi trong tập dữ liệu. Giá trị arr[i][j] là giá trị tf x idf của hàng i cột j và được tính thông qua quá trình sau:

1. Tại cột thứ j, ứng với nhãn là một từ X, dùng nhãn này để tìm giá trị ở bước tiếp theo.

2. Tìm số lần xuất hiện của từ X trong hàng thứ i và số lượng từ của nội dung lỗi thứ i.

3. Tìm số nội dung lỗi có chứa từ X

4. Tính giá trị tf x idf của từ X

$$arr[i][j]=\frac{\text{số lần xuất hiện của từ X trong hàng thứ i}}{\text{số lượng từ của nội dung lỗi thứ i}}\ x\ log(\text{số nội dung lỗi có chứa từ X})$$

5. Theo trên, từ nào tại cột j không xuất hiện trong nội dung lỗi thứ i sẽ có giá trị arr[i][j] = 0.

Sau khi tính toàn bộ giá trị của các phần tử trong mảng, tiến hành sử dụng vòng lặp for để tìm ra từ khóa của nội dung lỗi như sau:

```
for i in range(số lượng lỗi):
        max = giá trị max của các phần tử trong hàng thứ i)
        for j in range(số lượng từ):
                if arr[i][j] = max:
                        gán từ khóa của hàng thứ i là nhãn của cột j
                        break
return(tập từ khóa của tất cả các nội dung lỗi)
```

0b1	100	2003	233	4096	64	85	absolute	access	accessible	account	accounts	actual	add	address	amp	apostrophe	application
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.356481	0.0	0.0	0.000000	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.297308	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0

Hình 3.4: Giá trị tf x idf sau khi tính toán

Các giá trị bằng 0 do tại hàng thứ i nội dung lỗi không chứa từ được gán nhãn tương ứng với cột đang xét. Ví dụ tại hàng đầu tiên từ "actual" có giá trị tf x idf = 0.356481, là giá trị lớn nhất trong hàng đầu tiên nên từ khóa của nội dung lỗi thứ nhất chính là "actual". Tương tự như trên ta có trường thuộc tính từ khóa được bổ sung vào tập dữ liệu huấn luyện như hình bên dưới.

State	Component	Software	Platform	Relation	Effect	Keyword
fixed	up2date	Red Hat Linux	linux	124619	yes	pam
fixed	strace	Fedora	linux	0	no	strace
fixed	rpm	Fedora	linux	0	no	honor
fixed	system-config-mo	Fedora	linux	0	no	installation
fixed	kernel	Red Hat Linux	linux	0	yes	573
...	...	...	...	...	...	...
fixed	kernel	Fedora	linux	0	yes	trace
fixed	kernel	Fedora	linux	122301	no	video
fixed	rpmdb-redhat	Red Hat Enterpri	linux	0	no	arpwatch
fixed	anaconda	Fedora	linux	120289	no	network
fixed	rpm	Fedora	linux	0	no	incompatibility

Hình 3.5: Dữ liệu lỗi sau khi bổ sung thuộc tính từ khóa

Thuộc tính từ khóa với giá trị là từ có giá trị cao nhất được bổ sung vào tập dữ liệu để làm đầu vào cho thuật toán Rừng ngẫu nhiên.

3.3. Xây dựng rừng ngẫu nhiên

3.3.1. Chuẩn hóa dữ liệu sang dạng số

Dữ liệu được chuẩn hóa sang kiểu dữ liệu dạng số để phù hợp với mô hình thuật toán Rừng ngẫu nhiên. Gọi n_c là số phần tử riêng biệt của thuộc tính c trong tập huấn luyện. Mỗi giá trị của thuộc tính c được gán nhãn từ 0 đến $n_c - 1$. Bảng giá trị của các thuộc tính thuộc tập huấn luyện được thể hiện qua hình sau.

State	Component	Software	Platform	Relation	Keyword	Severity
0	16	1	3	22	8	1
0	0	14	3	17	65	1
0	8	3	0	24	43	1
0	9	15	0	56	35	1
0	15	5	2	35	39	0
...	...	...	...	...	...	...
0	31	7	0	2	70	1
0	27	13	0	50	58	1
0	5	10	0	56	77	1
0	27	13	0	49	77	1
1	24	3	3	56	41	3

Hình 3.6: Các thuộc tính sau khi được chuyển về dạng số

Ví dụ như trường thuộc tính trạng thái với hai trạng thái là mở và đóng được chuyển về dạng số dưới dạng giá trị 0 (ứng với trạng thái đóng) và 1 (ứng với trạng thái mở).

Một lỗi với các thuộc tính quan trọng sau khi được trích xuất và chuẩn hóa được chuyển về dạng véc tơ để làm đầu vào xây dựng Rừng ngẫu nhiên với đầu vào và đầu ra như sau:

Đầu vào: tập dataset Y là tập dữ liệu lỗi với các thuộc tính trạng thái, thành phần xảy ra lỗi, phần mềm, nền tảng, mối liên hệ, từ khóa, mức độ ưu tiên và giá trị ngưỡng đánh giá σ (giá trị trong luận văn sử dụng là giá trị F1 score và điều kiện là $\sigma > 0.8$).

Đầu ra: Rừng ngẫu nhiên với tập hợp các cây quyết định tối ưu R.

Quá trình xây dựng một Rừng ngẫu nhiên được thực hiện qua các bước tiếp sau đây.

3.3.2. Lấy mẫu dữ liệu cho việc xây dựng cây quyết định

Chia tập dataset Y ngẫu nhiên thành 2 phần: tập dữ liệu để kiểm tra E (20%) và tập dữ liệu để huấn luyện T (80%). Sử dụng T để xây dựng Rừng ngẫu nhiên. Dữ liệu huấn luyện T với các thuộc tính trạng thái, thành phần xảy ra lỗi, phần mềm, nền tảng, mối liên hệ, từ khóa để xây dựng mô hình phân lớp. Tổng cộng có 6 thuộc tính dùng cho việc phân lớp dữ liệu. Giả sử có 1000 mẫu dữ liệu lỗi, 800 mẫu sẽ được dùng để xây dựng một cây quyết định, 200 mẫu lỗi còn lại dùng để đánh giá mô hình cây quyết định xây dựng được có thỏa mãn giá trị đánh giá σ hay không. Nếu thỏa mãn cây vừa xây dựng sẽ được thêm vào tập cây của rừng ngẫu nhiên.

	State	Component	Software	Platform	Relation	Effect	Keyword
0	fixed	up2date	Red Hat Linux	linux	124619	yes	pam
1	fixed	strace	Fedora	linux	0	no	strace
2	fixed	rpm	Fedora	linux	0	no	honor
3	fixed	system-config-mo	Fedora	linux	0	no	installation
4	fixed	kernel	Red Hat Linux	linux	0	yes	573
...	...	...	...	...	...	...	...
995	fixed	kernel	Fedora	linux	0	yes	trace
996	fixed	kernel	Fedora	linux	122301	no	video
997	fixed	rpmdb-redhat	Red Hat Enterpri	linux	0	no	arpwatch
998	fixed	anaconda	Fedora	linux	120289	no	network
999	fixed	rpm	Fedora	linux	0	no	incompatibility

1000 rows × 7 columns

Hình 3.7: Tập dữ liệu 1000 mẫu lỗi

	State	Component	Software	Platform	Relation	Effect	Keyword
743	fixed	rp-pppoe	Fedora	linux	0	no	pppoe
381	fixed	control-center	Fedora	linux	0	no	theme
914	fixed	gtkam	Fedora	linux	0	no	gtkam
137	fixed	kernel	Red Hat Enterpri	linux	130338	no	32768
396	fixed	openoffice.org	Fedora	linux	0	yes	errata
...	...	...	...	...	...	...	...
723	fixed	rsync	Fedora	linux	0	no	quot
341	fixed	policy	Fedora	linux	0	no	filesystems
622	fixed	system-config-pr	Fedora	linux	0	no	cups
887	fixed	kdebase	Fedora	linux	0	no	quot
268	fixed	qt	Fedora	linux	0	no	lib6464

800 rows × 7 columns

Hình 3.8: Tập huấn luyện cây quyết định với 800 mẫu được lấy ngẫu nhiên

	State	Component	Software	Platform	Relation	Effect	Keyword
518	fixed	kernel	Fedora	linux	0	no	cdc_acm
970	fixed	kernel	Fedora	linux	0	yes	350
171	fixed	policy	Fedora	linux	0	no	enforcing
40	fixed	gnome-media	Fedora	linux	100774	no	quot
772	fixed	xorg-x11	Fedora	linux	118734	no	xrender
...	...	...	...	...	...	...	...
765	fixed	xfree86	Fedora	linux	73733	no	board
16	fixed	xscreensaver	Fedora	linux	3745	no	lock
766	fixed	xscreensaver	Fedora	linux	71940	no	themed
320	fixed	gcc	Red Hat Linux	linux	0	yes	fcomi
549	fixed	a2ps	Fedora	linux	0	no	a2ps

200 rows × 7 columns

Hình 3.9: Tập thử nghiệm với 200 mẫu còn lại để đánh giá cây quyết định

3.3.3 Xây dựng cây quyết định

Quá trình tạo cây quyết định với từng tập dữ liệu sau quá trình lấy mẫu ngẫu nhiên ở bước trước được thực hiện như sau:

Quá trình xây dựng cây bắt đầu bằng việc khởi động nút gốc thành nút nhị phân. Ban đầu, tất cả các mẫu dữ liệu là nằm trong nút gốc. CART triển khai một tổ hợp các thuật toán chuyên sâu tìm kiếm cách phân chia tốt nhất ở tất cả các điểm phân chia có thể có cho mỗi biến. Các phương pháp mà CART sử dụng để xây dựng cây quyết định được gọi là phân vùng đệ quy nhị phân. Thông qua chỉ số Gini như một quy tắc tách.

Bước 1: CART tách biến đầu tiên trong tất cả các biến tại các điểm phân tách có thể xảy ra, tại tất cả các giá trị mà biến giả định có trong tập mẫu. Tại mỗi điểm phân chia có thể có của một biến, mẫu phân tách thành hai nút con. Các trường hợp có câu trả lời "có" cho câu hỏi được đặt ra được gửi đến nút bên trái và những trường hợp phản hồi "không" được gửi đến đúng nút bên phải.

Bước 2: CART sau đó áp dụng tiêu chí phân tách dựa trên công thức $GINI_{split}$ tại mỗi điểm phân tách và đánh giá mức giảm tạp chất đạt được bằng cách sử dụng công thức:

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

Bước 3: CART chọn cách tách tốt nhất cho các biến với sự phân chia mà sự giảm tạp chất là cao nhất. Ba bước trên được lặp lại cho mỗi biến còn lại ở nút gốc.

Bước 4: CART sau đó xếp hạng tất cả các cách chia tốt nhất trên mỗi biến theo sự giảm tạp chất đạt được bằng mỗi lần tách và chọn biến cùng điểm tách mà nó làm giảm tạm chất của nút gốc và nút trung gian nhiều nhất.

Bước 5: CART sau đó gán lớp cho các nút này theo quy tắc giảm thiểu phân loại. CART có một thuật toán tích hợp để cho phép người dùng xác định điểm gán lớp trong quá trình tách. Mặc định là 1 đơn vị hoặc bằng giá trị phân loại. Bởi vì thủ tục CART là đệ quy, các bước 1 - 5 được áp dụng nhiều lần cho mỗi phần tử không phải nút lá ở mỗi giai đoạn kế tiếp.

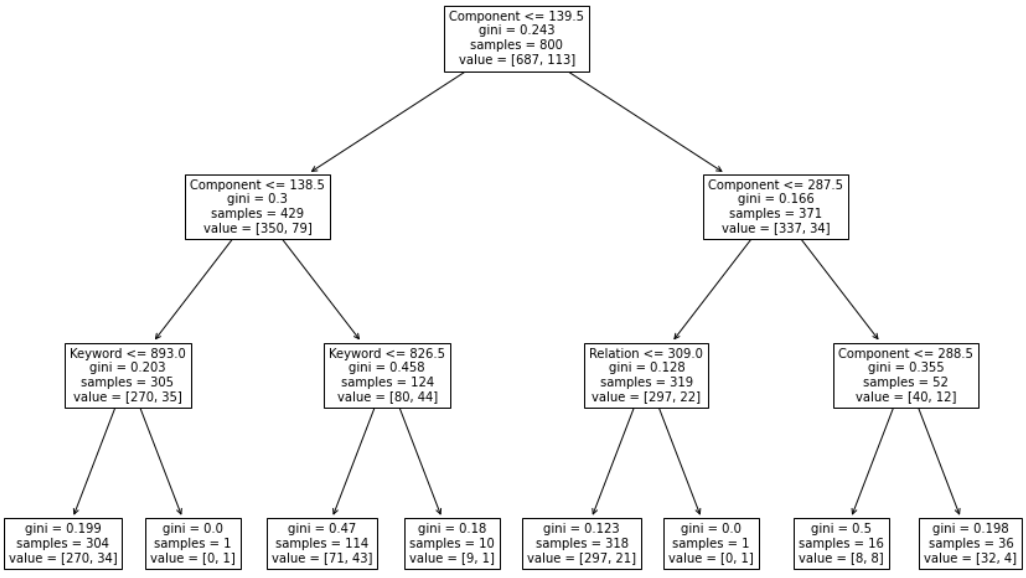
CART dừng quá trình phân tách khi:

- Chỉ có một mẫu trong mỗi nút.

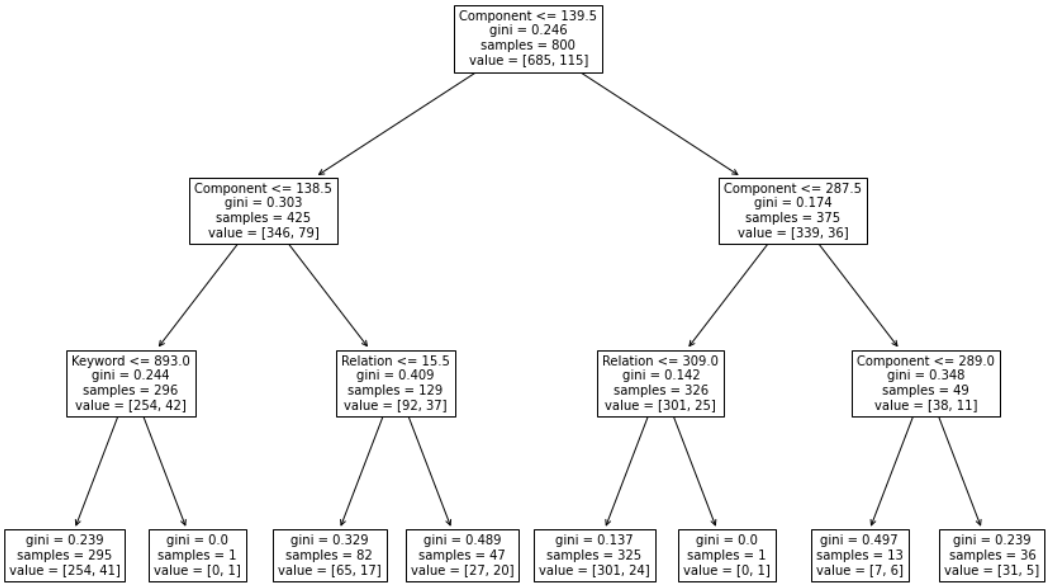
- Tất cả các mẫu trong mỗi nút con có phân loại giống hệt nhau của các biến phân lớp, tức là không thể tách.

- Cây đạt đủ độ sâu theo cài đặt (max_depth=10).

Một số ví dụ về cây quyết định được xây dựng với giá trị max_depth=3 được thể hiện qua các hình minh họa như sau:



Hình 3.10: Cây quyết định xây dựng trên mẫu huyến luyện ngẫu nhiên thứ nhất

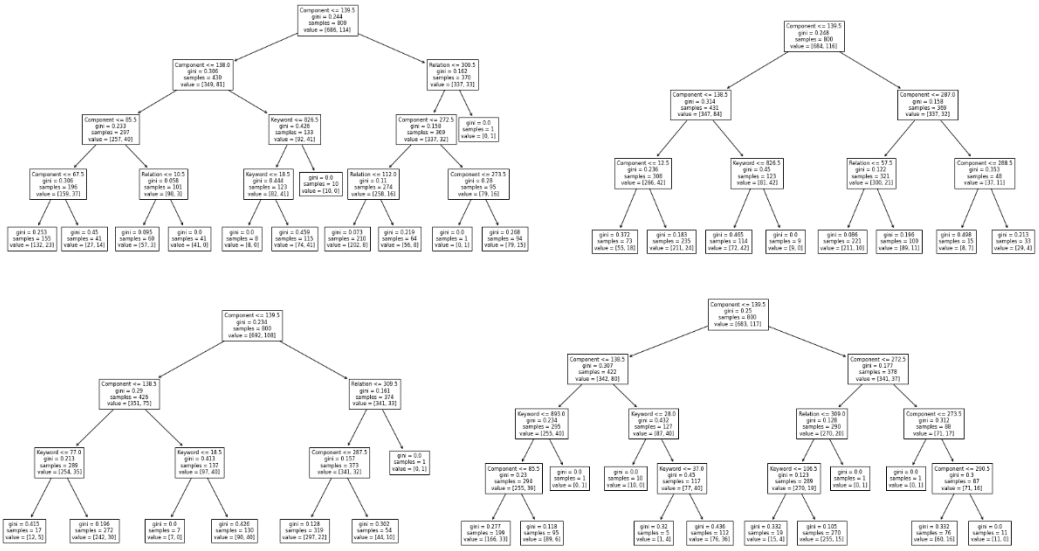


Hình 3.11: Cây quyết định xây dựng trên mẫu huyến luyện ngẫu nhiên thứ hai

1.3.4. Xây dựng rừng ngẫu nhiên

Cây quyết định sau khi xây dựng xong nếu thỏa giá trị đánh giá ban đầu sẽ được cập nhật vào rừng ngẫu nhiên R. Việc đánh giá dựa trên tập thử nghiệm E và giá trị đánh giá σ. Độ chính xác của cây quyết định, cụ thể là giá trị F1 Score của cây sẽ được tính dựa vào kết quả phân loại tập thử nghiệm E, nếu F1 Score > 0.8 thì cây sẽ được đưa vào rừng ngẫu nhiên và ngược lại.

Toàn bộ quá trình lấy mẫu ngẫu nhiên dữ liệu và xây dựng cây quyết định được thực hiện lặp lại cho đến khi không thể tạo thêm cây mới hoặc rừng ngẫu nhiên đạt đủ số lượng cây theo cài đặt ban đầu. Sau khi kết thúc ra sẽ thu được một mô hình rừng ngẫu nhiên gồm nhiều cây quyết định tối ưu được xây dựng trên tập mẫu được lấy ngẫu nhiên từ tập dataset ban đầu.



Hình 3.12: Một ví dụ rừng ngẫu nhiên với 4 cây quyết định

Chương 4 – PHÂN TÍCH VÀ ĐÁNH GIÁ

4.1. Phân tích độ chính xác của mô hình

Để đánh giá độ chính xác của mô hình đã xây dựng ta dựa vào kết quả phân lớp tập thử nghiệm là tập dữ liệu được trích xuất từ tập dataset ban đầu với 1000 mẫu lỗi được xây dựng thủ công, trường thuộc tích từ khóa được chọn lọc từ nội dung lỗi để lấy từ liên quan đến lỗi đó nhất. Ta có ma trận hỗn loạn sau:

Bảng 4.1: Ma trận hỗn loạn cho kết quả phân loại lỗi

		Mức độ ảnh hưởng của lỗi thông qua mô hình phân lớp	
		Ảnh hưởng đến hệ thống	**Không ảnh hưởng đến hệ thống**
Mức độ ảnh hưởng thực tế	**Ảnh hưởng đến hệ thống**	TP	FN
	Không ảnh hưởng đến hệ thống	FP	TN

Trong đó:

TP: số lượng lỗi thực tế ảnh hưởng đến hệ thống được phân loại đúng là có ảnh hưởng đến hệ thống.

FN: số lượng lỗi thực tế ảnh hưởng đến hệ thống bị phân loại sai là không ảnh hưởng đến hệ thống.

FP: số lượng lỗi thực tế không ảnh hưởng đến hệ thống bị phân loại sai là có ảnh hưởng đến hệ thống.

TN: số lượng lỗi thực tế không ảnh hưởng đến hệ thống được phân loại đúng là không ảnh hưởng đến hệ thống.

Các giá trị Recall Score, Precision Score, F1 Score dùng để đánh giá mô hình được tính như sau:

Recall Score = $\frac{TP}{TP+FN}$

Precision Score = $\frac{TP}{TP+FP}$

F1 Score = $\frac{2*(Recall\ Score)*(Precision\ Score)}{Recall\ Score+Precision\ Score}$

Với tập thử nghiệm 1000 mẫu lỗi được lấy ngẫu nhiên từ tập dataset sau đó xây dựng các thuộc tính thủ công ta có các thông tin sau:

- Tổng số lỗi: 1000
- Tổng số lỗi ảnh hưởng đến hệ thống: 200
- Tổng số lỗi không ảnh hưởng đến hệ thống: 800

Thực hiện dự đoán mức độ ảnh hưởng của 1000 mẫu lỗi trên thông qua mô hình rừng ngẫu nhiên đã xây dựng, kết quả dự đoán mức độ ảnh hưởng của lỗi như sau:

- Tổng số lỗi được dự báo có ảnh hưởng đến hệ thống: 209
- Tổng số lỗi được dự báo không ảnh hưởng đến hệ thống: 791
- Tổng số lỗi thực tế ảnh hưởng đến hệ thống được mô hình rừng ngẫu nhiên dự báo chính xác: 179
- Tổng số lỗi thực tế ảnh hưởng đến hệ thống bị mô hình rừng ngẫu nhiên dự báo sai thành không ảnh hưởng: 21
- Tổng số lỗi thực tế không ảnh hưởng đến hệ thống bị mô hình rừng ngẫu nhiên dự báo sai thành có ảnh hưởng: 30
- Tổng số lỗi thực tế không ảnh hưởng đến hệ thống được mô hình rừng ngẫu nhiên dự báo đúng là không ảnh hưởng: 770

Từ các kết quả trên ta tính được các chỉ số sau:

Recall Score = $\frac{TP}{TP+FN} = \frac{179}{179+21} = 89.50\%$

Precision Score = $\frac{TP}{TP+FP} = \frac{179}{179+30} = 85.65\%$

F1 Score = $\frac{2*(Recall\ Score)*(Precision\ Score)}{Recall\ Score+Precision\ Score} = 87.53\%$

Giá trị Recall Score đạt 89.50% chứng tỏ tỉ lệ lỗi ảnh hưởng đến hệ thống được phân loại thành không ảnh hưởng đến hệ thống vẫn còn cao, điều này có thể gây ra việc bỏ qua các lỗi đặc biệt nghiêm trọng của hệ thống.

Giá trị Precision Score đạt 85.65%, chứng tỏ tỉ lệ lỗi không ảnh hưởng đến hệ thống bị phân loại thành có ảnh hưởng đến hệ thống cao hơn trường hợp ngược lại, điều này có nghĩa cảnh báo giả cũng sẽ xuất hiện nhiều hơn với 30 cảnh báo giả được dự báo trên 1000 mẫu lỗi được xử lý chiến tỉ lệ 3%.

Giá trị F1 Score đạt 87.53%, giá trị trên với mô hình phát hiện lỗi sử dụng rừng ngẫu nhiên cho kết quả ở mức chấp nhận được.

Để đánh giá mức độ ảnh hưởng của các tham số quan trọng trong quá trình xây dựng rừng ngẫu nhiên là max_depth (độ sâu của cây quyết định) và n_ estimators (số lượng cây quyết định trong rừng ngẫu nhiên), các thử nghiệm được thực hiện 10 lần và các kết quả thu được dựa trên giá trị trung bình của các lần chạy như bảng kết quả đánh giá sau.

Bảng 4.2: Giá trị F1 Score với hai tham số quan trọng của rừng ngẫu nhiên

max_depth\n_estimators	**20**	**30**	**50**	**100**	**200**
5	87.38	87.38	87.39	87.34	87.39
10	87.4	87.41	87.39	87.39	87.39
15	87.18	87.37	87.37	87.39	87.38
20	86.78	86.65	86.77	86.81	86.87
25	85.81	85.75	85.9	85.97	86.03

Dựa vào các chỉ số đánh giá độ chính xác của mô hình trên có thể đưa ra các nhận định sau:

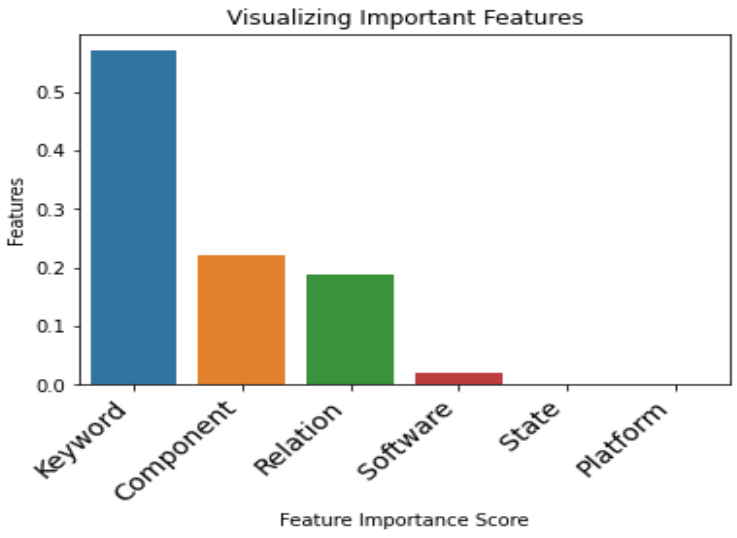
- Mô hình cho độ chính xác cao nhất với giá trị max_depth = 10, giá trị n_estimators = 30 với các giá trị F1 Score = 87.41%.

- Số lượng mẫu lỗi ảnh hưởng đến hệ thống còn ít nên chưa tạo được sự bao quát cho mô hình phân lớp nên độ chính xác của mô hình còn chưa thực sự cao. Nguyên nhân có thể do giá trị của các thuộc tính lỗi trong tập dataset còn rời rạc với nhau dẫn đến mô hình phân lớp phù hợp với lỗi này nhưng lại không phù hợp để đánh giá lỗi khác với các giá trị thuộc tính khác biệt. Phần tiếp theo của luận văn sẽ đánh giá mức độ quan trọng của các thuộc tính để làm rõ vấn đề trên.

4.2. Xác định mức độ quan trọng của các thuộc tính

Mức độ quan trọng của các thuộc tính được xác định bằng độ giảm của chỉ số gini tại mỗi nút trong quá trình xây dựng cây quyết định. Độ giảm chỉ số gini càng nhiều ứng với mức độ quan trọng của thuộc tính càng cao.

Việc đánh giá mức độ quan trọng của các thuộc tính cho phép chúng ta phân tích được vai trò của mỗi thuộc tính trong việc xây dựng mô hình phân lớp. Trong luận văn mức độ quan trọng của các thuộc tính được thể hiện qua kết quả sau đây.



Hình 4.1: Biểu đồ mức độ quan trọng của các thuộc tính

feature	importance
Keyword	5.708934e-01
Component	2.195647e-01
Relation	1.880771e-01
Software	2.021638e-02
State	1.247604e-03
Platform	9.017770e-07

Hình 4.2: Kết quả mức độ quan trọng của các thuộc tính

Như kết quả trong hình, thuộc tính từ khóa ảnh hưởng đến hơn 57% kết quả phân lớp của mô hình, đây là thuộc tính quan trọng nhất trong việc xác định một lỗi có ảnh hưởng hay là không đến một hệ thống. Trái ngược với nó là các thuộc tính như nên tảng và trạng thái hầu như không làm ảnh hưởng đến kết quả phân loại lỗi. Từ kết quả trên, hướng phát triển của luận văn là tập trung nghiên cứu cách trích xuất từ khóa một cách tối ưu nhất để thuộc tính này đặc trưng cho một lỗi nghiêm trọng.

Tuy độ quan trọng của thuộc tính từ khóa chiếm tỉ lệ rất cao nhưng dựa vào kết quả phân lớp các lỗi ảnh hưởng đến hệ thống chưa cao chứng tỏ giá trị của thuộc tính này khi trích xuất từ nội dung lỗi qua thuộc tính tf x idf còn chưa hoàn thiện. Giá trị từ khóa này chưa thực sự là từ có giá trị nhất trong nội dung lỗi vì kỹ thuật tf x idf dùng để đánh giá tầm quan trọng của các từ khóa trong phần mô tả. Đặc điểm của mô tả lỗi thường dài và dùng ngôn ngữ tự nhiên nên có thể cần đến phương phân tích ngữ nghĩa đối với nội dung lỗi trên để loại bỏ các từ không có giá trị như địa chỉ bộ nhớ, thông tin gỡ lỗi, đường dẫn hệ thống… Phương pháp hiện tại chỉ so sánh từ khóa hoặc đánh giá từ quan trọng trong một nội dung lỗi. Đây chính là nhược điểm của quá trình khai thác thuộc tính mô tả lỗi.

Thuộc tính có độ quan trọng tiếp theo là thành phần xảy ra lỗi, việc thuộc tính này đóng vai trò quan trọng trong việc phân loại các lỗi đặc biệt nghiêm trọng

gây ảnh hưởng đến hệ thống chứng tỏ các lỗi trên thường xuất hiện trong một thành phần nào đó của một phần mềm, một nền tảng ứng dụng chứ thường không xuất hiện đồng loạt trên cả một phần mềm hay nền tảng nào đó.

Đối với các thuộc tính có mức độ quan trọng cao nhưng kết quả phân lớp cho thấy các lỗi ảnh hưởng là chưa cao, cụ thể là thuộc tính từ khóa ta cần phải xử lý thêm đối với nội dung lỗi để trích xuất ra thuộc tính từ khóa có giá trị. Vì nội dung lỗi thường được thể hiện với dạng ngôn ngữ tự nhiên nên việc áp dụng phương pháp LSI/LSA là cần thiết để phân tích ngữ nghĩa văn bản và kết hợp với phương pháp tf x idf để cho ra thuộc tính từ khóa có giá trị phân lớp cao hơn. Ngoài ra việc xử lý nội dung lỗi để loại bỏ các thông tin rác như địa chỉ ô nhớ, link đính kèm là cần thiết vì các nội dung trên có giá trị tf x idf rất cao và nhiều khả năng sẽ bị hiểu lầm là từ khóa của một lỗi.

Chương 5 - KẾT LUẬN

5.1. Kết quả đạt được

5.1.1. Về mặt lý thuyết

Khai thác được mô hình dữ liệu lỗi để xây dựng mô hình phát hiện lỗi mạng.

Ứng dụng Trí tuệ nhân tạo (AI), Machine Learning, các thuật toán học máy và phương pháp khai phá dữ liệu văn bản vào việc phát hiện lỗi mạng.

Khai thác được các thuật toán phân lớp dữ liệu, cụ thể là mô hình cây quyết định và rừng ngẫu nhiên. Nắm bắt được quá trình xây dựng một cây quyết định dựa trên giá trị gini index hay entropy và quá trình xây dựng một rừng ngẫu nhiên dựa trên các cây quyết định.

Ứng dụng thư viện scikit-learn trên nền tảng python vào việc nghiên cứu các vấn đề học máy, sử dụng được các tham số để tối ưu mô hình rừng ngẫu nhiên xây dựng được.

5.1.2. Về mặt thực tiễn

Luận văn đã đưa ra được giải pháp phân loại lỗi và cảnh báo sớm sự cố có thể ảnh hưởng đến một hệ thống mạng dựa vào việc phân tích lỗi từ các hệ thống báo cáo lỗi hiện có. Việc này sẽ là tiền đề để xây dựng một công cụ phát hiện lỗi mạng ảnh hưởng đến một hệ thống mạng và truyền thông trong tương lai, thay thế người vận hành với vai trò là một chuyên gia nhận định lỗi và đưa ra dự báo về mức độ nghiêm trọng của một lỗi.

Mô hình trên có thể hỗ trợ người dùng trên các hệ thống BTS, khi người dùng báo cáo một lỗi mới lên hệ thống sẽ có thể nhận định sớm từ ban đầu mức độ ảnh hưởng của lỗi trên đến hệ thống của họ.

Đưa ra giải pháp phân tích lỗi dựa vào phương pháp khai phá văn bản tf x idf, phương pháp trên giúp luận văn trích xuất được các thông tin quan trọng của một

lỗi trong một nội rộng lớn. Phương pháp này thực sự có ý nghĩa đối với một báo cáo lỗi vì người dùng thường sẽ báo cáo một lỗi với những thông tin rất dễ trùng lặp với các lỗi khác như không thể truy cập, không thể mở, hệ thống bị treo… trong khi vấn đề thực sự của hệ thống lại bị bỏ qua.

Xây dựng thành công mô hình phát hiện lỗi mạng và truyền thông, phân tích và đánh giá mô hình xây dựng được để hiểu rõ hơn về cách thức hoạt động của thuật toán rừng ngẫu nhiên và việc phân tích các báo cáo lỗi.

5.2. Hạn chế

Kết quả phân loại lỗi đạt được chỉ ở mức tốt chứ chưa thật sự cao. Kết quả đạt được chưa bao quát được hết các trường hợp. Dữ liệu mẫu cần được training và mở rộng môi trường áp dụng.

Các trường hợp lỗi nghiêm trọng bị phân loại sai thành không nghiêm trọng còn nhiều, gây nhầm lẫn cho người sử dụng nếu áp dụng với thực tế.

Mô hình rừng ngẫu nhiên trong luận văn còn ở mức cơ bản, chưa phân tích sâu vào các tham số để phù hợp với mô hình dữ liệu lỗi.

Việc phân tích nội dung lỗi bằng phương pháp tf x idf còn nhiều thiếu sót, chưa đủ để trích xuất được nội dung quan trọng với việc phân loại lỗi từ nội dung mô tả.

5.3. Hướng phát triển

Tập trung nghiên cứu rút trích các đặc trưng thuộc tính lỗi phù hợp hơn cho quá trình phân tích, tăng độ chính xác trong việc phân loại lỗi. Nghiên cứu các mô hình lỗi mạng để cải thiện mô hình phân loại lỗi được tốt hơn.

Áp dụng kết hợp phương pháp phân tích ngữ nghĩa LSI/LSA kết hợp với phương pháp tf x idf trong việc trích xuất từ khóa từ nội dung lỗi để mang lại kết quả có giá trị hơn đối với việc phân loại lỗi.

Tiến hành áp dụng cho cho hệ thống mạng lưới mạng di động của Viễn thông Tây Ninh. Cảnh báo sớm các lỗi nghiêm trọng có thể xảy ra dựa trên cơ sở dữ liệu là các lỗi và mức độ ảnh hưởng của chúng trong quá khứ. Trích xuất các nội dung ghi chú của các lỗi để tìm ra đặc trưng của một lỗi nghiêm trọng trên hệ thống mạng di động thuộc Viễn thông Tây Ninh góp phần cảnh báo sớm các sự cố ảnh hưởng nghiêm trọng đến hệ thống.

DANH MỤC TÀI LIỆU THAM KHẢO

Tiếng Việt

[1] Hoàng Ngọc Thanh, Trần Văn Lăng, Hoàng Tùng (2016), "*Một tiếp cận máy học để phân lớp các kiểu tấn công trong hệ thống phát hiện xâm nhập mạng*", Kỷ yếu Hội nghị khoa học Quốc gia FAIR'9, T. 502-507

[2] Nguyễn Thị Thanh Hương, Đoàn Minh Trung (2018), "*Áp dụng thuật toán phân loại Random Forest để xây dựng bản đồ sử dụng đất/thảm phủ tỉnh Đắk Lắk dựa vào ảnh vệ tinh Landsat 8 OLI*", Tạp chí Nông nghiệp & Phát triển nông thôn, T. 122-129

[3] Dang N. H. Thanh, Nguyen Quoc Hung, Tran Le Phuc Thinh (2020), "*Một góc nhìn từ bài toán phân lớp dữ liệu: Thang điểm đánh giá nào là quan trọng?*", Kỷ yếu hội thảo khoa học quốc gia Hệ thống thông tin trong kinh doanh và quản lý ISBM20, T. 276-279

[4] Đỗ Thị Lương (2019), "*Nghiên cứu một số thuật toán học máy để phân lớp dữ liệu và thử nghiệm*", Hà Nội, 62 trang

[5] Nguyễn Thị Thùy Linh (2005), "*Nghiên cứu các thuật toán phân lớp dữ liệu dựa trên cây quyết định*", Hà Nội, T. 21-27

Tiếng Anh

[6] Ha Manh Tran, Tuan Anh Nguyen, Son Thanh Le, Giang Vu Truong Huynh, Tuan Bao Lam (2021), "*Two-Phase Defect Detection Using Clustering and Classification Methods*", REV Journal on Electronics and Communications: Article scheduled for publication in Vol. 11, No. 3–4

[7] Sinh Van Nguyen, Ha Manh Tran (2020), "*An automated fault detection system for communication networks and distributed systems"*, Springer Science+Business Media, LLC, part of Springer Nature, Available: https://doi.org/10.1007/s10489-020-02026-2

[8] Buckley MF, Siewiorek DP (1995), "*VAX/VMS Event monitoring and analysis. In: Proceedings 25th international symposium on fault-tolerant computing (FTCS'95)*", IEEE computer society, pp 414–423

[9] O'Sullivan, Dympna, et al. (2008), "*Using Secondary Knowledge to Support Decision Tree Classification of Retrospective Clinical Data*", Mining Complex Data (2008), pp. 238-251

[10] Christopher J.C. Burges (2000), "*A Tutorial on Support Vector Machines for Pattern Recognition*", Kluwer Academic Publishers, Boston

[11] Eslamloueyan, R. (2011), "*Designing a hierarchical neural network based on fuzzy clustering for fault diagnosis of the Tennessee - Eastman process*", Applied Soft Computing, Volume 11, Issue 1, pp. 1407-1415

[12] Sunil Kumar, Saroj Ratnoo, Renu Bala (2020), "*Enhanced Decision Tree Algorithm for Discovery of Exceptions*", Department of Computer Science & Engineering, Guru Jambheshwar University of Science & Technology, Hisar, Haryana, India, pp. 3-7.

[13] M. Uddin, R. Stadler, and A. Clemm (2013), "*A Query Language for Network Search*", Proceedings of the 13th IFIP/IEEE International Symposium on Integrated Network Management (IM'13). IEEE, pp. 109–117

[14] H. M. Tran and S. T. Le (2014), "*Software Bug Ontology Supporting Semantic Bug Search on Peer-to-Peer Networks*" New Generation Computing, vol. 32, no. 2, pp. 145–162, Available: https://doi.org/10.1007/s00354-014-0203-1

[15] T. Wang, W. Zhang, J. Wei, and H. Zhong (2015), "*Fault Detection for Cloud Computing Systems with Correlation Analysis*" in Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM'15). IEEE, pp. 652–658, Available: https://doi.org/10.1109/INM.2015.7140351

[16] W. Zhou, L. Tang, C. Zeng, T. Li, L. Shwartz, and G. Y. Grabarnik (2016), "*Resolution recommendation for event tickets in service management*", IEEE Transactions on Network and Service Management, vol. 13, no. 4, pp. 954–967, Available: https://doi.org/10.1109/TNSM.2016.2587807

[17] V. C. Ferreira, R. C. Carrano, J. O. Silva, C. V. N. Albuquerque, D. C. Muchaluat-Saade, and D. G. Passos (2017), "*Fault Detection and Diagnosis for Solar-Powered Wireless Mesh Networks Using Machine Learning*", Proceedings of the IFIP/IEEE Symposium on Integrated Network and Service Management (IM'17), IEEE, pp. 456–462, Available: https://doi.org/10.23919/INM.2017.7987312

[18] D. Hausheer and C. Morariu (2008), "*Distributed Test-Lab: EMANICSLab*", University of Zurich, Switzerland, The 2nd International Summer School on Network and Service Management (ISSNSM'08)

[19] M. Foundation (1998), "*Bugzilla bug tracker*", Available: https://www.bugzilla.org/

[20] E. Software (2004), "*Trac bug tracker*", Available: https://trac.edgewall.org/

[21] M. Team (2000), "*Mantis bug tracker*", Available: https://www.mantisbt.org

[22] C. Company (2004), "*Launchpad bug tracker*", Available: https://bugs.launchpad.net/

[23] L. Breiman (2001), "*Random Forests*", Machine Learning, vol. 45, no. 1, pp. 5–32

[24] Gilles Louppe, "*Understanding Random Forest from theory to pratic*", University of Liège, Faculty of Applied Sciences, Department of Electrical Engineering & Computer Science, pp. 55-115

[25] S. Chatrchyan, V. Khachatryan, A. Sirunyan, A. Tumasyan, W. Adam,

E. Aguilo, T. Bergauer, M. Dragicevic, J. Erö, C. Fabjan (2012) "*Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC*", Physics Letters B, pp. 2

[26] A. Criminisi and J. Shotton (2013), "*Decision Forests for Computer Vision and Medical Image Analysis*". Springer, pp. 2, 39, 106 and 107

[27] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager (2010), "*Building Watson: An overview of the Deep QA project*", AI magazine, 31(3):59–79

[28] https://cdspninhthuan.edu.vn/, truy cập ngày 02/08/2021

[29] https://viblo.asia/, truy cập ngày 10/08/2021

[30] https://machinelearningcoban.com/, truy cập ngày 15/08/2021

[31] https://vi.wikipedia.org/, truy cập ngày 20/08/2021

[32] https://www.coursera.org/, truy cập ngày 20/08/2021

[33] https://scikit-learn.org/, truy cập ngày 27/11/202