Kỷ yếu Hội nghị Khoa học công nghệ Quốc gia lần thứ XVI về Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR); Đà Nẵng, ngày 28-29/9/2023
DOI: 10.15625/vap.2023.0034

DỰ ĐOÁN CODE-SMELL DỰA TRÊN PHÂN LOẠI ĐA NHÃN

Nguyễn Thanh Bình[1], Nguyễn Hữu Nhật Minh[2], Lê Thị Mỹ Hạnh[3], Nguyễn Thanh Bình[2]

[1]Trường Cao đẳng Cơ điện - Xây dựng và Nông Lâm Trung Bộ
[2]Trường Đại học Công nghệ Thông tin và Truyền thông Việt - Hàn, Đại học Đà Nẵng
[3]Trường Đại học Bách khoa, Đại học Đà Nẵng

thanhbinh@cdtb.edu.vn, nhnminh@vku.udn.vn, ltmhanh@dut.udn.vn, ntbinh@vku.udn.vn

TÓM TẮT: *Trong hai thập kỷ gần đây, dự đoán code-smell dựa trên học máy là một lĩnh vực đang được cộng đồng nghiên cứu quan tâm. Tuy nhiên, hầu hết các nghiên cứu đều sử dụng tập dữ liệu độ đo mã nguồn đơn nhãn trong các mô hình thử nghiệm, mặc dù trong thực tế một mẫu độ đo mã nguồn có thể tồn tại nhiều code-smell khác nhau. Bài báo này công bố một tập dữ liệu đa nhãn phù hợp cho các nghiên cứu liên quan đến phân loại đa nhãn và đồng thời đề xuất một mô hình học máy phát hiện code-smell dựa trên các phương pháp phân loại đa nhãn. Các mô hình học máy, dựa trên phương pháp phân loại đa nhãn, nhằm phát hiện code-smell trên tập dữ liệu được công bố đã cho thấy nhiều kết quả hứa hẹn của hướng nghiên cứu này.*

Từ khóa: *Phân loại đa nhãn, Dự đoán code-smell, Học máy.*

I. GIỚI THIỆU

Trong quá trình bảo trì và phát triển, các hệ thống phần mềm được các nhà phát triển cập nhật một cách liên tục nhằm các mục đích chính như: (i) thêm các cài đặt để đáp ứng các yêu cầu mới, (ii) cải thiện các chức năng hiện có, hoặc (iii) sửa các lỗi phần mềm nghiêm trọng [1]. Do các yếu tố liên quan như: áp lực về thời gian, thiếu kỹ năng hoặc kinh nghiệm, các nhà phát triển không phải lúc nào cũng có đủ thời gian và năng lực để sẵn sàng kiểm soát hoàn toàn độ phức tạp của hệ thống phần mềm nhằm tìm ra giải pháp tối ưu trước khi áp dụng các sửa đổi [2]. Kết quả là, các hoạt động bảo trì, phát triển thường được thực hiện bằng các giải pháp chưa tối ưu và có khả năng tác động làm sai lệch thiết kế ban đầu của hệ thống phần mềm; mặc dù các giải pháp này có thể được coi là phù hợp trong ngắn hạn, nhưng về lâu dài, chúng sẽ làm tăng độ phức tạp và chi phí bảo trì, phát triển đối với hệ thống; các vấn đề này được gọi là nợ kỹ thuật [3].

Code-smell là triệu chứng của việc thiết kế hoặc triển khai mã nguồn dưới mức tối ưu mà các nhà phát triển áp dụng cho hệ thống phần mềm, và đây cũng là một trong những dạng nợ kỹ thuật nghiêm trọng nhất. Code-smell không phải là một lỗi phần mềm, vì vậy sự hiện diện của code-smell trong mã nguồn không nhất thiết gây ra lỗi hoặc hành vi không mong muốn trong quá trình phần mềm được thực thi, nhưng chúng có thể khiến mã nguồn khó hiểu hoặc khó sửa đổi hơn hoặc tăng nguy cơ gây ra lỗi phần mềm hoặc các vấn đề khác trong tương lai. Thật vậy, các nghiên cứu trước đây cho thấy rằng code-smell không chỉ làm giảm mạnh khả năng hiểu mã nguồn của các nhà phát triển, mà còn tác động đáng kể đến chất lượng và khả năng bảo trì, mở rộng của mã nguồn [4-6]. Do đó, code-smell đang là một trong những thách thức quan trọng đối với các nhà phát triển về nỗ lực và chi phí bảo trì hệ thống [7, 8].

Bên cạnh các nghiên cứu về việc nhận diện và phân loại [9], cộng đồng nghiên cứu cũng rất tích cực trong việc đề xuất các kỹ thuật hỗ trợ phát hiện code-smell. Kết quả thống kê và phân loại các kỹ thuật phát hiện code-smell trong hai thập kỷ gần đây của Pereira và các cộng sự [10] đã cho thấy rằng nhiều kỹ thuật phát hiện code-smell đã được đề xuất, mỗi kỹ thuật đều dựa trên một hoặc nhiều phương pháp tiếp cận khác nhau và hầu hết đều đem lại các kết quả hứa hẹn. Xét về khía cạnh giải thuật được áp dụng cho phương pháp phát hiện code-smell, các nghiên cứu thử nghiệm nhằm phát hiện code-smell trong mã nguồn bằng cách sử dụng các kỹ thuật học máy (Machine learning - ML) dựa trên tập dữ liệu độ đo mã nguồn (Source code metric dataset - SMD) đã cho thấy nhiều kết quả tốt [11-19]; các thử nghiệm này đã cho thấy SMD là một yếu tố có vai trò quan trọng trong mô hình phát hiện code-smell dựa trên ML. Tuy nhiên, qua kết quả thống kê cho thấy [20], hầu hết các nghiên cứu đều xây dựng SMD đơn nhãn để huấn luyện cho các mô hình học máy phân loại đơn nhãn để dự đoán code-smell. Trong đó, chỉ có nghiên cứu [21] sử dụng mô hình học máy phân loại đa nhãn, nhưng SMD đa nhãn này chỉ được xây dựng bằng cách kết hợp dữ liệu từ nhiều SMD đơn nhãn từ một nghiên cứu trước đó. Cũng bởi vì lý do này, việc áp dụng các mô hình học máy phân loại đa nhãn để dự đoán code-smell chưa được nhiều nghiên cứu quan tâm, cho dù các nghiên cứu thuộc nhiều lĩnh vực khác đã công bố những kết quả tích cực từ những mô hình phân loại thuộc nhóm này [22-25].

Bài báo này giới thiệu tập dữ liệu *Class-smell* đa nhãn, bao gồm hơn 90.000 mẫu độ đo mã nguồn (source code metric) ở cấp độ lớp (class-level), được trích xuất từ tập dữ liệu *ml-Codesmell* [26]; mỗi mẫu trong *Class-smell* bao gồm 41 đặc trưng và 1 tập hợp nhãn chứa một hoặc nhiều nhãn code-smell từ tập hợp 6 nhãn code-smell đã được xác định. Dựa trên *Class-smell*, các giải thuật phân loại đa nhãn được áp dụng cho các kỹ thuật ML để dự đoán code-smell; hiệu suất dự đoán của các kỹ thuật ML sẽ được cân nhắc để xác định giải thuật phù hợp trong việc xây dựng các mô hình học máy phân loại đa nhãn để dự đoán code-smell.

Phần tiếp theo của bài báo này là tổng quan về các nghiên cứu liên quan đến vấn đề phát hiện code-smell dựa trên các kỹ thuật ML. Mục III của bài báo sẽ đề cập đến các khái niệm liên quan đến phân loại trong học máy. Ý tưởng thiết kế và các cân nhắc để lựa chọn giải thuật phân loại đa nhãn, độ đo hiệu suất cho các mô hình thử nghiệm dự đoán

code-smell dựa trên học máy được trình bày trong Mục IV. Mục V của bài báo sẽ là các kết luận chung liên quan đến các kết quả thử nghiệm và hướng nghiên cứu cần phát triển dựa trên kết quả bài báo này.

II. CÁC NGHIÊN CỨU LIÊN QUAN

Trong hai thập kỷ gần đây, nhiều nghiên cứu đã áp dụng ML trong việc phát hiện code-smell và công bố kết quả thông qua các bài báo nghiên cứu, các bài thuyết trình trên các tạp chí và hội nghị học thuật. Tuy nhiên, theo kết quả tổng hợp của Zakeri và các cộng sự [20] cho thấy chỉ có 25 trong số 45 nghiên cứu (chiếm tỷ lệ khoảng 56%) cung cấp công khai tập dữ liệu để cộng đồng nghiên cứu sử dụng. Trong đó, có một số nghiên cứu tiêu biểu dưới đây.

Năm 2009, Khomh và cộng sự [27] đã đề xuất một phương pháp phát hiện code-smell *God-Class* dựa trên thuật toán Bayesian Belief Network. Tập dữ liệu của nghiên cứu này được xây dựng bằng hai dự án mã nguồn mở Java và có gần 800 mẫu dữ liệu, đồng thời các code-smell đã được xác thực bằng thủ công. Vào năm 2011, tập dữ liệu này đã được tái sử dụng, cải thiện và thêm các tính năng mới để phát hiện ba code-smell *God-Class, Functional-Decomposition* và *Spaghetti-Code* [28].

Năm 2016, Arcelli Fontana F. và cộng sự [12] đã tạo một mô hình ML để phát hiện code-smell. Tập dữ liệu của nghiên cứu này có hơn 1.600 mẫu dữ liệu được trích xuất từ 74 dự án mã nguồn mở Java. Ngoài ra, code-smell trong tập dữ liệu đã được xác thực thủ công. Tuy nhiên, tập dữ liệu này chỉ có 526 mẫu dương tính với các code-smell ở cấp độ lớp (Class-level) và cấp độ phương thức (Method-level), bao gồm *God-Class, Data-Class, Long-Method* và *Feature-Envy*.

Năm 2017, Ian Shoenberger và cộng sự [29] đã đề xuất tự động hóa các quy tắc phát sinh để phát hiện code-smell trong mã nguồn JavaScript. Tập dữ liệu của nghiên cứu này chứa các mẫu của 100 dự án JavaScript được trích xuất bằng cách sử dụng các công cụ InCode và PMD. Họ dựa vào tập dữ liệu này để huấn luyện các quy tắc phát hiện bằng cách sử dụng lập trình di truyền và tìm ra ngưỡng tốt nhất của các độ đo để tạo nên các quy tắc.

Năm 2019, Pecorelli và cộng sự [30] đã tạo tập dữ liệu bao gồm 8.534 mẫu được trích xuất từ 13 hệ thống mã nguồn mở. Dựa vào tập dữ liệu này, họ đã so sánh việc phát hiện bốn code-smell giữa hai phương pháp ML và Heuristic.

Năm 2020, nhóm nghiên cứu của L. Madeyski [31] đã xây dựng tập dữ liệu gồm 14.739 mẫu được kiểm tra và xác thực thủ công để xác định bốn loại code-smell *Blod, Data-Class, Long-Method* và *Feature-Envy* với nhiều mức độ nghiêm trọng khác nhau với kỳ vọng tập dữ liệu này sẽ hỗ trợ cộng đồng nghiên cứu trong việc phát hiện code-smell trong các dự án mã nguồn mở Java.

Năm 2020, Guggulothu và cộng sự [21] đã có những bước đầu tiên trong việc tiến hành xây dựng tập dữ liệu đa nhãn và xây dựng các mô hình học máy áp dụng các kỹ thuật ML để dự đoán code-smell dựa trên tập dữ liệu này. Tập dữ liệu được xây dựng một cách thủ công dựa trên tập dữ liệu Fontana [12] và có số lượng mẫu hạn chế. Mặc dù vậy nghiên cứu cũng đã công bố một số kết quả đáng khích lệ.

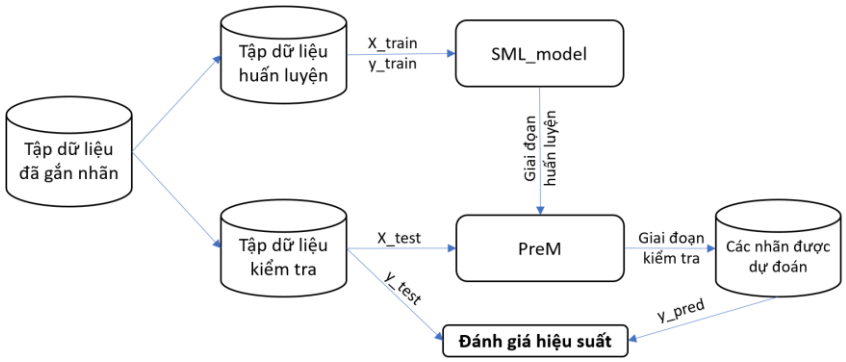
Đánh giá một cách tổng thể qua kết quả của các công trình nghiên cứu, các mô hình ML đã chứng tỏ chúng có thể được huấn luyện dựa trên các tập dữ liệu đơn nhãn để tự động xác định code-smell tiềm ẩn trong các hệ thống phần mềm. Tuy nhiên, đến thời điểm hiện tại, mỗi mẫu trong các tập dữ liệu của hầu hết các nghiên cứu chỉ được gắn một nhãn tương ứng với một code-smell duy nhất, vì vậy rõ ràng việc gắn nhãn cho các mẫu có nhiều khả năng sẽ bỏ sót các code-smell đang tồn tại trong các mẫu này. Đây chính là một nguyên nhân mà các code-smell sẽ bị bỏ qua trong quá trình phát hiện.

Trong tập dữ liệu *Class-smell* được giới thiệu trong bài báo này, mỗi mẫu có thể được gắn với nhiều nhãn code-smell khác nhau; với đặc điểm khác biệt này của *Class-smell*, nghiên cứu này đề xuất việc xem xét mối tương quan giữa các code-smell đối với việc xuất hiện của chúng bằng cách sử dụng các kỹ thuật phân loại đa nhãn trong các mô hình ML. Mô hình thử nghiệm sẽ được tiến hành với một số thuật toán thông dụng, kết quả của thử nghiệm được kỳ vọng sẽ hữu ích cho các nghiên cứu liên quan trong quá trình phát triển phần mềm, đặc biệt là các nghiên cứu về phát hiện code-smell dựa trên học máy.

III. MỘT SỐ KHÁI NIỆM VỀ PHÂN LOẠI TRONG HỌC MÁY CÓ GIÁM SÁT

A. Mô hình học máy có giám sát

Trong các mô hình học máy có giám sát (Supervised machine learning model - SML_model) được minh họa như Hình 1, có hai giai đoạn khác nhau: Giai đoạn huấn luyện và giai đoạn kiểm tra. Trong giai đoạn huấn luyện, SML_model được huấn luyện bằng tập dữ liệu huấn luyện để tạo ra một mô hình dự đoán (Prediction model - PreM) nhằm ánh xạ một tập các đặc trưng của tập dữ liệu huấn luyện (*X_train*) tới tập các nhãn mục tiêu (*y_train*) tương ứng. Trong giai đoạn kiểm tra, PreM được tạo ra từ giai đoạn huấn luyện sẽ được sử dụng để gắn nhãn (*y_pred*) cho tập các đặc trưng (*X_test*) trong tập dữ liệu kiểm tra. Hiệu suất dự đoán của PreM được đánh giá bằng độ chính xác khi so sánh giữa *y_pred* và các nhãn thực tế của tập dữ liệu kiểm tra (*y_test*).



Hình 1. Mô hình học máy có giám sát

B. Phân loại đơn nhãn

Với nhiệm vụ phân loại đơn nhãn (Single label classification - SLC), PreM sẽ thực hiện việc gán một *y_pred* duy nhất tương ứng với mỗi mẫu trong tập dữ liệu kiểm tra. Trong trường hợp giá trị của *y_pred* chỉ nhận 1 trong 2 giá trị khác nhau (Hình 2), nghĩa là PreM sẽ tách tập tập dữ liệu kiểm tra thành 2 nhóm phân biệt dựa trên *X_test* của chúng, thì nhiệm vụ phân loại được gọi là phân loại nhị phân (Binary classification - BC).

X_test				PreM	y_pred
x_1	x_2	...	x_m	→	0
x_1	x_2	...	x_m	→	1
x_1	x_2	...	x_m	→	1
x_1	x_2	...	x_m	→	0

Hình 2. Phân loại nhị phân

Ngoài ra, nếu *y_pred* có thể nhận được một trong nhiều giá trị khác nhau (Hình 3) thì được gọi là phân loại đa lớp (Multi-class classification - MCC).

X_test				PreM	y_pred
x_1	x_2	...	x_m	→	A
x_1	x_2	...	x_m	→	B
x_1	x_2	...	x_m	→	A
x_1	x_2	...	x_m	→	C

Hình 3. Phân loại đa lớp

C. Phân loại đa nhãn

Phân loại đa nhãn (Multi-label classification - MLC) là nhiệm vụ của PreM nhằm gán cho một mẫu của tập dữ liệu kiểm tra đồng thời nhiều *y_pred* dựa vào *X_test* của chúng (Hình 4).

X_test				PreM	y_preds			
x_1	x_2	...	x_m	→	0	1	1	0
x_1	x_2	...	x_m	→	1	1	0	0
x_1	x_2	...	x_m	→	1	0	0	0
x_1	x_2	...	x_m	→	0	0	1	1

Hình 4. Phân loại đa nhãn

Khác với trường hợp MCC, các giá trị của *y_pred* trong MLC không còn loại trừ lẫn nhau mà mỗi mẫu dữ liệu có thể được gắn với một tập hợp giá trị có liên quan (*y_preds*); bên cạnh đó, cũng khác với SLC, MLC bị ảnh hưởng bởi các mối tương quan tiềm ẩn nội tại giữa các *y_pred*, có nghĩa là một hoặc nhiều giá trị của *y_preds* có khả năng đóng vai trò như là các đặc trưng hữu ích để cùng với *X_test* nhằm dự đoán các giá trị *y_pred* còn lại. Ví dụ, một mẫu

dữ liệu có code-smell *Long-Method* và *Long-Parameter-List* thì có nhiều khả năng là đang ẩn chứa code-smell *Large-Class*.

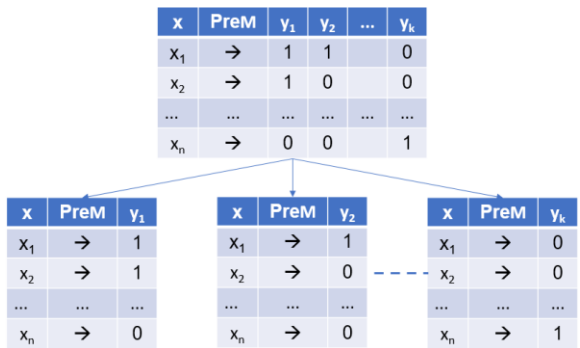
D. Các phương pháp giải quyết vấn đề phân loại đa nhãn

Nhiều phương pháp đã được đề xuất trong các nghiên cứu để giải quyết các vấn đề trong MLC. Dựa vào kết quả của các nghiên cứu [22, 23], các thuật toán phân loại đa nhãn hiện tại có thể được phân thành ba nhóm chính như sau: Chuyển đổi vấn đề (*Problem-transformation*), Thích ứng thuật toán (*Algorithm-adaption*) và Phương pháp tập hợp (*Ensemble-methods*); mỗi nhóm phương pháp đều có những mặt tích cực và hạn chế tùy thuộc vào đặc tính của từng bài toán cần giải quyết. Trong bài báo này, với mục tiêu ứng dụng các kỹ thuật MLC cơ bản trong bài toán dự đoán code-smell dựa trên ML, các phương pháp thuộc nhóm Chuyển đổi vấn đề được cân nhắc để áp dụng trong các mô hình thử nghiệm.

Các phương pháp thuộc nhóm Chuyển đổi vấn đề có thể được sử dụng với bất kỳ mô hình MLC nào. Trong cách tiếp cận này, một bài toán MLC được chuyển đổi thành một hoặc nhiều bài toán SLC; sau đó, các giải pháp của những bài toán SLC này được kết hợp để giải quyết mục tiêu của bài toán MLC. Nhóm Chuyển đổi vấn đề bao gồm ba phương pháp cơ bản là *Binary-relevance*, *Classifier-chain* và *Label-powerset* được mô tả ngắn gọn dưới đây.

1. Binary-relevance

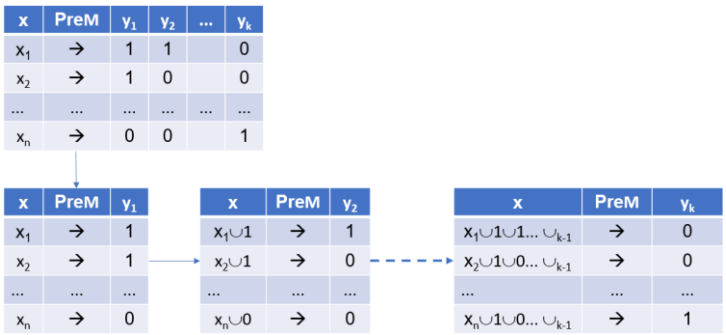
Binary-relevance (BR) [22] chia bài toán MLC (giả sử với *k* nhãn) thành *k* bài toán SLC (Hình 5). Mỗi bài toán SLC này được huấn luyện trên *X_train* và nhằm mục đích xác định mức độ liên quan của nhãn cụ thể của nó đối với một mẫu nhất định. Kết quả MLC sau đó được xác định bởi sự kết hợp từ các kết quả của các bài toán SLC đã được giải quyết. Cách tiếp cận BR rất đơn giản để thực hiện và độ phức tạp của nó là tuyến tính với số lượng nhãn có thể. Tuy nhiên, BR bỏ qua mối tương quan giữa các nhãn bằng cách xử lý từng nhãn một cách độc lập, đây là hạn chế lớn nhất của hướng tiếp cận này.



Hình 5. Phương pháp Binary-relevance

2. Classifier-chain

Để khắc phục hạn chế của BR về tính độc lập của các nhãn, phương pháp Classifier-chain (CC) [32] tạo *k* bộ SLC nhưng được liên kết theo cách sao cho mỗi bộ SLC cũng bao gồm các kết quả dự đoán của các bộ phân loại trước đó dưới dạng các tính năng đầu vào bổ sung (Hình 6). Theo cách này và không giống như BR, CC có thể mô hình hóa các mối quan hệ giữa các nhãn mà không gây ra nhiều phức tạp hơn. Tuy nhiên, mặc dù CC xử lý mối quan hệ giữa các nhãn, nhưng thứ tự của các nhãn chỉ được chọn ngẫu nhiên mà không được xem xét liệu thứ tự của việc lựa chọn có ảnh hưởng quyết định đến hiệu suất chung của mô hình.



Hình 6. Phương pháp Classifier-chain

3. Label-powerset

Label-powerset (LP) [22, 32] chuyển đổi bài toán MLC thành bài toán MCC, tạo ra một tập dữ liệu đơn nhãn trong đó mỗi tập nhãn riêng biệt được coi là một lớp khác nhau, như trong Hình 7. Sau đó, LP sử dụng bất kỳ phương pháp MCC nào để huấn luyện một mô hình với dữ liệu mới và dự đoán cuối cùng thu được bằng cách chuyển đổi lớp

dự đoán thành tập nhãn tương ứng của nó. LP xem xét tất cả các mối tương quan nhãn nhưng độ phức tạp của nó là cấp số nhân với số lượng nhãn. Bên cạnh đó, nó không thể dự đoán một tập nhãn không xuất hiện trong tập dữ liệu huấn luyện và vì nhiều lớp thường chỉ được liên kết với một số mẫu, điều này có thể dẫn đến tập dữ liệu mất cân bằng cao, khiến quá trình huấn luyện trở nên khó khăn và kém chính xác hơn.

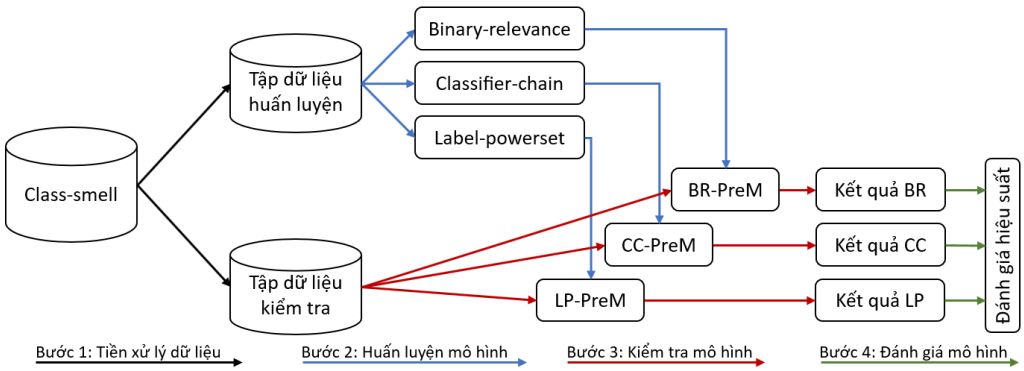
x	PreM	y_1	y_2	...	y_k
x_1	→	1	1		0
x_2	→	1	0		0
...	...	...	...	...	...
x_n	→	0	0		1

x	PreM	y
x_1	→	11...0
x_2	→	10...0
...	...	...
x_n	→	00...1

Hình 7. Phương pháp Label-powerset

IV. XÂY DỰNG MÔ HÌNH HỌC MÁY DỰ ĐOÁN CODE SMELL DỰA TRÊN PHÂN LOẠI ĐA NHÃN

Mô hình học máy dự đoán code-smell áp dụng kỹ thuật phân loại đa nhãn được xây dựng dựa trên tập dữ liệu *Class-smell* và các phương pháp MLC thuộc nhóm *Problem-transformation*. Mô hình được minh họa như Hình 8.



Hình 8. Mô hình học máy dự đoán code-smell dựa trên phân loại đa nhãn

A. *Tập dữ liệu Class-smell*

Tập dữ liệu *Class-smell* là tập hợp các mẫu độ đo mã nguồn ở cấp độ lớp được trích xuất từ tập dữ liệu *ml-Codesmell* [26], được sử dụng cho các thử nghiệm MLC trong nghiên cứu này.

Trong *Class-smell*, một mẫu được được gọi là *Dương tính* nếu được gắn với tối thiểu một nhãn, ngược lại nó được gọi là *Âm tính* nếu không được gắn với bất kỳ một nhãn nào. Với khái niệm trên, sau khi thực hiện các tác vụ tiền xử lý, *Class-smell* sẵn sàng phục vụ cho việc thử nghiệm với các thông số như sau:

- Tổng số đặc trưng: 41.
- Tổng số nhãn code-smell: 6.
- Số mẫu *Âm tính*: 45.895.
- Số mẫu *Dương tính*: 45.895, trong đó số lượng nhãn được gắn đồng thời cho các mẫu như sau:
  - 1 nhãn: 43.606.
  - 2 nhãn: 2.263.
  - 3 nhãn: 26.

Bảng 1. Số lượng mẫu được gắn nhãn trong Class-smell

Số lượng mẫu / Code-smell	Âm tính	Dương tính
Brain Class	90.947	843
Data Class	69.620	22.170
Futile Abstract Pipeline	91.452	338
Futile Hierarchy	91.426	364
God Class	88.238	3.552
Schizofrenic Class	70.847	20.943

Đánh giá chung về *Class-smell*: Số lượng mẫu đáp ứng đủ cho các mô hình ML; Các mẫu có thể được gắn đồng thời với nhiều nhãn, phù hợp với yêu cầu của các phương pháp phân loại đa nhãn; *Class-smell* được cân bằng về số lượng mẫu *Dương tính* và *Âm tính*.

Tập dữ liệu *Class-smell* được công bố trên figshare[1].

B. Các độ đo hiệu suất mô hình

Trong các mô hình SLC, hiệu suất tổng thể của mô hình ML được đánh giá bằng các độ đo thông thường như *precision*, *recall*, *F1-score* hoặc *AUC,...* Tuy nhiên, việc đánh giá hiệu suất trong MLC phức tạp hơn nhiều so với SLC, vì mỗi mẫu dữ liệu có thể được liên kết với nhiều nhãn cùng một lúc. Trong bài báo này, hai độ đo hiệu suất được sử dụng để so sánh hiệu suất giữa các mô hình MLC được đề xuất, đó là Hamming-Loss [33] và F1-score [32].

- Hamming-Loss: Hàm này trả về giá trị là phần nhãn bị dự đoán sai. Khi đánh giá hiệu suất trong phân loại nhiều lớp, Hamming-Loss không phạt các mẫu dự đoán có bộ nhãn không khớp hoàn toàn mà phạt từng nhãn riêng lẻ của chúng; do đó, giá trị của Hamming-Loss rơi vào trong khoảng từ 0 đến 1. Hamming-Loss càng bé thì hiệu suất mô hình càng tốt. Ví dụ:

  y_pred = [1, 2, 3, 4],
  y_test = [2, 2, 3, 4],
  hamming_loss(*y_test*, *y_pred*) $\rightarrow$ 0,25.
- F1-score: Đây là một độ đo hiệu suất phổ biến và được xem là hài hòa giữa *precision* và *recall*. Trong mô hình MLC, trước tiên, giá trị của F1-score sẽ được xác định dựa trên từng nhãn độc lập; F1-score tổng thể của mô hình được kết hợp từ kết quả F1-score của từng nhãn độc lập. Tham số *average* được sử dụng cho độ đo này để xác định cách kết hợp các kết quả từ các nhãn độc lập, *average* có thể nhận một trong các giá trị sau:
  - *micro*: Hiệu suất tổng thể được xác định bằng cách đếm tổng số kết quả dương tính thật, âm tính giả và dương tính giả.
  - *macro*: Tính toán hiệu suất cho từng nhãn, hiệu suất tổng thể là giá trị trung bình không tính trọng số của chúng. Điều này không tính đến sự mất cân bằng nhãn.
  - *weighted*: Tính toán hiệu suất cho từng nhãn, hiệu suất tổng thể là trọng số trung bình của chúng dựa trên số lượng mẫu dữ liệu của mỗi nhãn. Điều này giải thích cho sự mất cân bằng nhãn.
  - *samples*: Tính toán hiệu suất cho từng mẫu, hiệu suất tổng thể là giá trị trung bình của chúng.

C. Kết quả thử nghiệm

Các mô hình học máy sử dụng ba phương pháp BR, CC và LP để chuyển đổi tập dữ liệu huấn luyện nhiều nhãn thành tập hợp các tập dữ liệu nhị phân hoặc đa lớp; sau đó, dựa vào mức độ phổ biến, hiệu quả và đơn giản, bốn giải thuật phân loại là Gaussian Naïve Bayes, Logistic Regression, Decision Tree và K-Nearest Neighbors được sử dụng để dự đoán các code-smell trên các tập dữ liệu đã chuyển đổi. Hiệu suất tổng thể của mô hình được thể hiện trong Bảng 2, với kết quả dự đoán tốt nhất giữa các phương pháp chuyển đổi trên cùng một giải thuật phân loại được in đậm.

Bảng 2. Kết quả dự đoán code-smell dựa trên mô hình học máy áp dụng phương pháp MLC

Kết quả / **Phân loại & chuyển đổi**	**F1_score**				**Hamming loss**
	micro	macro	weighted	samples	
Gaussian Naïve Bayes					
BR	0,83	0,50	0,90	0,45	0,03
CC	0,84	0,45	0,87	0,45	0,03
LP	**0,87**	**0,57**	**0,90**	**0,47**	**0,03**
Logistic Regression					
BR	0,97	**0,59**	0,97	0,48	0,01
CC	0,98	0,59	0,97	0,49	**0,00**
LP	**0,97**	0,58	**0,97**	**0,49**	0,01
Decision Tree					
BR	**1,00**	**0,89**	**0,99**	**0,50**	**0,00**
CC	0,99	0,89	0,99	0,50	0,00
LP	0,99	0,85	0,99	0,50	0,00
K-Nearest Neighbors					
BR	**0,85**	**0,52**	**0,85**	0,42	**0,03**
CC	0,85	0,52	0,85	0,42	0,03
LP	0,85	0,51	0,84	**0,43**	0,03

D. Thảo luận kết quả

Kết quả thử nghiệm cho thấy hiệu suất dự đoán thấp của các giải thuật Gaussian Naïve Bayes, K-Nearest Neighbors; hiệu suất dự đoán của các mô hình áp dụng hai thuật toán này phụ thuộc nhiều vào sự tương quan giữa các đặc trưng với các nhãn (Gaussian Naïve Bayes) hoặc sự tương quan giữa các đặc trưng của mẫu kề nhau (K-Nearest

[1] https://figshare.com/s/61c1c31121f4d2a02546

Neighbors). Bên cạnh đó, các giải thuật phân loại Logistic Regression và Decision Tree hoạt động tốt với hầu hết các phương pháp MLC được xem xét; các độ đo Hamming-Loss và F1-score$_{micro}$ của các mô hình khi áp dụng hai giải thuật này đều cho kết quả khả quan. Mặt khác, hiệu suất dự đoán code-smell giữa ba phương pháp BR, CC và LP không chênh lệch nhiều, thậm chí BR còn có thể đạt hiệu suất cao nhất trong một vài độ đo.

Kết quả thử nghiệm đã cho thấy mức độ tương quan rất thấp giữa các code-smell và các đặc trưng trong tập dữ liệu Class-smell. Đây là các yếu tố quan trọng giúp hiệu suất của của hai thuật toán Logistic Regression và Decision Tree vượt trội hơn phần còn lại.

V. KẾT LUẬN

Bài báo giới thiệu tập dữ liệu *Class-smell* đa nhãn để sử dụng cho các nghiên cứu về lĩnh vực dự đoán code-smell dựa trên học máy áp dụng các phương pháp phân loại đa nhãn. Đến hiện nay, *Class-smell* có thể được xem là một trong những tập dữ liệu đa nhãn về code-smell được công bố có dung lượng đáng kể, với hơn 90.000 mẫu độ đo mã nguồn ở cấp độ lớp, 41 đặc trưng và 6 nhãn code-smell. Xét về mặt tổng thể, tập dữ liệu được cân bằng giữa số lượng mẫu *Dương tính* và *Âm tính*. Tuy nhiên, Class-smell vẫn còn tồn tại sự mất cân bằng giữa số lượng mẫu *Dương tính* và *Âm tính* trong mỗi code-smell.

Các phương pháp phân loại đa nhãn đơn giản, như Binary-relevance, Classifier-chain và Label-powerset cùng với bốn giải thuật phân loại phổ biến hiện nay, được áp dụng để xây dựng các mô hình học máy nhằm phát hiện các code-smell có thể đồng thời tồn tại trong một mẫu độ đo mã nguồn; hiệu suất dự đoán tổng thể của các mô hình thử nghiệm cho thấy tiềm năng hứa hẹn của học máy đối với vấn đề này.

Các vấn đề về mất cân bằng dữ liệu trên mỗi code-smell và sự khác biệt về kết quả dự đoán code-smell giữa các giải thuật phân loại trong bài báo này sẽ được giải quyết bằng các phương pháp phân loại đa nhãn nâng cao thuộc các nhóm khác, như Thích ứng thuật toán và Phương pháp tập hợp trong các nghiên cứu tương lai.

TÀI LIỆU THAM KHẢO

[1] M. M. Lehman, "Programs, life cycles, and laws of software evolution," *Proceedings of the IEEE*, vol. 68, no. 9, pp. 1060-1076, 1980, doi: 10.1109/PROC.1980.11805.

[2] D. L. Parnas, "Software aging," in *Proceedings of 16th International Conference on Software Engineering*, IEEE Comput. Soc. Press, 1994, pp. 279-287. doi: 10.1109/ICSE.1994.296790.

[3] P. Avgeriou and P. Kruchten, "Managing Technical Debt in Software Engineering," *Dagstuhl Seminar 16162*, 2016, doi: 10.4230/DagRep.6.4.110.

[4] M. Abbes, F. Khomh, Y.-G. Guéhéneuc, and G. Antoniol, "An Empirical Study of the Impact of Two Antipatterns, Blob and Spaghetti Code, On Program Comprehension," *Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR*, pp. 181-190, Aug. 2011, doi: 10.1109/CSMR.2011.24.

[5] F. Khomh, M. Di Penta, Y.-G. Guéhéneuc, and G. Antoniol, "An exploratory study of the impact of antipatterns on class change- and fault-proneness," *Empir Softw Eng*, vol. 17, pp. 243-275, Aug. 2012, doi: 10.1007/s10664-011-9171-y.

[6] D. Spadini, F. Palomba, A. Zaidman, M. Bruntink, and A. Bacchelli, "On the Relation of Test Smells to Software Code Quality," *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 1-12, Aug. 2018, doi: 10.1109/ICSME.2018.00010.

[7] R. D. Banker, S. M. Datar, C. F. Kemerer, and D. Zweig, "Software complexity and maintenance costs," *Commun ACM*, vol. 36, no. 11, pp. 81-94, Nov. 1993, doi: 10.1145/163359.163375.

[8] A. Yamashita and L. Moonen, "Do code smells reflect important maintainability aspects?," *IEEE International Conference on Software Maintenance, ICSM*, pp. 306-315, Aug. 2012, doi: 10.1109/ICSM.2012.6405287.

[9] W. Kessentini, M. Kessentini, H. Sahraoui, S. Bechikh, and A. Ouni, "A Cooperative Parallel Search-Based Software Engineering Approach for Code-Smells Detection," *IEEE Transactions on Software Engineering*, vol. 40, no. 9, pp. 841-861, Sep. 2014, doi: 10.1109/TSE.2014.2331057.

[10] J. Pereira dos Reis, F. Brito e Abreu, G. de Figueiredo Carneiro, and C. Anslow, "Code Smells Detection and Visualization: A Systematic Literature Review," *Archives of Computational Methods in Engineering*, vol. 29, no. 1, pp. 47-94, Jan. 2022, doi: 10.1007/s11831-021-09566-x.

[11] F. Arcelli Fontana and M. Zanoni, "Code smell severity classification using machine learning techniques," *Knowl Based Syst*, vol. 128, pp. 43-58, Jul. 2017, doi: 10.1016/j.knosys.2017.04.014.

[12] F. Arcelli Fontana, M. V. Mäntylä, M. Zanoni, and A. Marino, "Comparing and experimenting machine learning techniques for code smell detection," *Empir Softw Eng*, vol. 21, no. 3, pp. 1143-1191, Jun. 2016, doi: 10.1007/s10664-015-9378-4.

[13] F. A. Fontana, M. Zanoni, A. Marino, and M. V. Mantyla, "Code Smell Detection: Towards a Machine Learning-Based Approach," in *2013 IEEE International Conference on Software Maintenance*, IEEE, Sep. 2013, pp. 396-399. doi: 10.1109/ICSM.2013.56.

[14] U. Mansoor, M. Kessentini, B. R. Maxim, and K. Deb, "Multi-objective code-smells detection using good and bad design examples," *Software Quality Journal*, vol. 25, no. 2, pp. 529-552, Jun. 2017, doi: 10.1007/s11219-016-9309-7.

[15] Pushpalatha M N and Mrunalini M, "Predicting the Severity of Open Source Bug Reports Using Unsupervised and Supervised Techniques," *International Journal of Open Source Software and Processes*, vol. 10, no. 1, pp. 1-15, Jan. 2019, doi: 10.4018/IJOSSP.2019010101.

[16] M. Y. Mhawish and M. Gupta, "Generating Code-Smell Prediction Rules Using Decision Tree Algorithm and Software Metrics," *International Journal of Computer Sciences and Engineering*, vol. 7, no. 5, pp. 41-48, May 2019, doi: 10.26438/ijcse/v7i5.4148.

[17] M. Y. Mhawish and M. Gupta, "Predicting Code Smells and Analysis of Predictions: Using Machine Learning Techniques and Software Metrics," *J Comput Sci Technol*, vol. 35, no. 6, pp. 1428-1445, Nov. 2020, doi: 10.1007/s11390-020-0323-7.

[18] S. Dewangan and R. S. Rao, "Code Smell Detection Using Classification Approaches," 2022, pp. 257-266. doi: 10.1007/978-981-19-0901-6_25.

[19] S. Dewangan, R. S. Rao, A. Mishra, and M. Gupta, "Code Smell Detection Using Ensemble Machine Learning Algorithms," *Applied Sciences (Switzerland)*, vol. 12, no. 20, Oct. 2022, doi: 10.3390/app122010321.

[20] M. Zakeri-Nasrabadi, S. Parsa, E. Esmaili, and F. Palomba, "A Systematic Literature Review on the Code Smells Datasets and Validation Mechanisms," *ACM Comput Surv*, May 2023, doi: 10.1145/3596908.

[21] T. Guggulothu and S. A. Moiz, "Code smell detection using multi-label classification approach," *Software Quality Journal*, vol. 28, no. 3, pp. 1063-1086, Sep. 2020, doi: 10.1007/s11219-020-09498-y.

[22] G. Tsoumakas and I. Katakis, "Multi-Label Classification," *International Journal of Data Warehousing and Mining*, vol. 3, no. 3, pp. 1-13, Jul. 2007, doi: 10.4018/jdwm.2007070101.

[23] G. Madjarov, D. Kocev, D. Gjorgjevikj, and S. Džeroski, "An extensive experimental comparison of methods for multi-label learning," *Pattern Recognit*, vol. 45, no. 9, pp. 3084-3104, Sep. 2012, doi: 10.1016/j.patcog.2012.03.004.

[24] M. Pushpa and S. Karpagavalli, "Multi-label Classification: Problem Transformation methods in Tamil Phoneme classification," *Procedia Comput Sci*, vol. 115, pp. 572-579, 2017, doi: 10.1016/j.procs.2017.09.116.

[25] J. Bogatinovski, L. Todorovski, S. Džeroski, and D. Kocev, "Comprehensive Comparative Study of Multi-Label Classification Methods," Feb. 2021, [Online]. Available: http://arxiv.org/abs/2102.07113

[26] B. Nguyen Thanh, M. Nguyen N. H., H. Le Thi My, and B. Nguyen Thanh, "ml-Codesmell: A code smell prediction dataset for machine learning approaches," in *The 11th International Symposium on Information and Communication Technology*, New York, NY, USA: ACM, Dec. 2022, pp. 368-374. doi: 10.1145/3568562.3568643.

[27] F. Khomh, S. Vaucher, Y.-G. Guéhéneuc, and H. Sahraoui, "A Bayesian Approach for the Detection of Code and Design Smells," *Proc Int Conf Qual Softw*, pp. 305-314, Aug. 2009, doi: 10.1109/QSIC.2009.47.

[28] F. Khomh, S. Vaucher, Y.-G. Guéhéneuc, and H. Sahraoui, "BDTEX: A GQM-based Bayesian approach for the detection of antipatterns," *Journal of Systems and Software*, vol. 84, pp. 559-572, Aug. 2011, doi: 10.1016/j.jss.2010.11.921.

[29] I. Shoenberger, M. W. Mkaouer, and M. Kessentini, "On the Use of Smelly Examples to Detect Code Smells in JavaScript," 2017, pp. 20-34. doi: 10.1007/978-3-319-55792-2_2.

[30] F. Pecorelli, F. Palomba, D. Di Nucci, and A. De Lucia, "Comparing Heuristic and Machine Learning Approaches for Metric-Based Code Smell Detection," in *2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC)*, IEEE, May 2019, pp. 93-104. doi: 10.1109/ICPC.2019.00023.

[31] L. Madeyski and T. Lewowski, "MLCQ," in *Proceedings of the Evaluation and Assessment in Software Engineering*, New York, NY, USA: ACM, Apr. 2020, pp. 342-347. doi: 10.1145/3383219.3383264.

[32] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Mach Learn*, vol. 85, no. 3, pp. 333-359, Dec. 2011, doi: 10.1007/s10994-011-5256-5.

[33] C. P. Prathibhamol, K. V Jyothy, and B. Noora, "Multi label classification based on logistic regression (MLC-LR)," in *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, IEEE, Sep. 2016, pp. 2708-2712. doi: 10.1109/ICACCI.2016.7732470.

CODE-SMELL PREDICTION BASED ON MULTI-LABEL CLASSIFICATION

Nguyen Thanh Binh, Nguyen Huu Nhat Minh, Le Thi My Hanh, Nguyen Thanh Binh

ABSTRACT: *In the last two decades, code-smell prediction based on machine learning has been a field of interest to the research community. However, most studies use single-label datasets in experimental models, even though a source code sample can hide many code smells in practice. This paper provides a multi-label dataset for studies related to multi-label classification, and the paper also introduces a method to build machine learning models to detect code-smell based on the simplest multi-label classification methods. Experimental models have shown promising results in this field of research.*

Keywords: *Multi-label classification, code-smell prediction, machine learning.*