

# DỰ ĐOÁN LỖI PHẦN MỀM DỰA TRÊN MÃ NGUỒN

TÓM TẮT LUẬN ÁN TIẾN SĨ KỸ THUẬT

Đà Nẵng, Năm 2026

# MỤC LỤC

## Mở đầu

### Chương 1: Tổng quan dự đoán lỗi phần mềm

- 1.1. Khái niệm và Quy trình dự đoán lỗi phần mềm
- 1.2. Đặc trưng mã nguồn và Lựa chọn đặc trưng
- 1.3. Trích xuất đặc trưng và Lấy mẫu dữ liệu
- 1.4. Dự đoán lỗi phần mềm dựa trên đặc trưng ngữ nghĩa mã nguồn

### Chương 2: Nâng cao khả năng dự đoán lỗi phần mềm bằng các phương pháp lựa chọn đặc trưng Wrapper

- 2.1. Giới thiệu
- 2.2. Phương pháp nghiên cứu
- 2.3. Kết quả thực nghiệm và phân tích
  - 2.3.1. Hiệu quả của các kỹ thuật lựa chọn đặc trưng (Câu hỏi nghiên cứu 1)
  - 2.3.2. Phương pháp wrapper hoạt động tốt nhất (Câu hỏi nghiên cứu 2)
  - 2.3.3. Kết quả kiểm định thống kê
- 2.4. Tổng kết chương

### Chương 3: Kết hợp các kỹ thuật lựa chọn đặc trưng và lấy mẫu dữ liệu trong dự đoán lỗi phần mềm

- 3.1. Giới thiệu
- 3.2. Phương pháp đề xuất
- 3.3. Kết quả thực nghiệm và phân tích
  - 3.3.1. Kết quả kết hợp RFE và các mô hình GAN (Câu hỏi nghiên cứu 1)
  - 3.3.2. Kết quả kết hợp Autoencoders và các mô hình GAN (Câu hỏi nghiên cứu 1 tiếp theo)
  - 3.3.3. Phương pháp GAN vượt trội nhất (Câu hỏi nghiên cứu 2)
  - 3.3.4. Kết quả kiểm định thống kê
- 3.4. Tổng kết chương

### Chương 4: Xây dựng mô hình dự đoán lỗi phần mềm dựa trên Transformer sử dụng Syntax Tree và Word Embedding

- 4.1. Giới thiệu
- 4.2. Các nghiên cứu liên quan và vấn đề nghiên cứu

#### 4.3. Đề xuất giải pháp

#### 4.4. Triển khai thực nghiệm, kết quả và phân tích

4.4.1. Dự đoán lỗi trong cùng một dự án (WPDP - Câu hỏi nghiên cứu 1)

4.4.2. Dự đoán lỗi giữa các dự án (CPDP - Câu hỏi nghiên cứu 2)

4.4.3. So sánh chi phí thời gian huấn luyện

4.4.4. Các mối đe dọa đến tính tin cậy và giá trị của nghiên cứu

#### 4.5. Tổng kết chương

#### Kết luận và hướng phát triển

Kết luận

Hướng nghiên cứu tương lai

## Mở đầu

Phần mềm đóng vai trò thiết yếu trong nhiều hệ thống và ứng dụng công nghệ hiện đại, từ quốc phòng, kiểm soát không lưu đến các thiết bị tiêu dùng. Với sự gia tăng về quy mô và độ phức tạp của phần mềm, việc đảm bảo chất lượng và độ tin cậy trở nên vô cùng quan trọng. Tuy nhiên, do những hạn chế về ngân sách và thời gian, việc tạo ra một phần mềm hoàn toàn không có lỗi là điều bất khả thi. Các lỗi phần mềm, dù nhỏ nhất, cũng có thể dẫn đến kết quả sai lệch và hành vi không mong muốn của chương trình, gây thất bại cho dự án.

Dự đoán lỗi phần mềm (Software Fault Prediction - SFP) là một lĩnh vực nghiên cứu quan trọng trong nhiều thập kỷ qua, nhằm mục đích xác định sớm các mô-đun phần mềm có khả năng chứa lỗi ngay từ giai đoạn đầu của quá trình phát triển. Bằng cách này, các nhà phát triển có thể tập trung nguồn lực và nỗ lực kiểm thử vào các khu vực tiềm ẩn rủi ro, từ đó tối ưu hóa quy trình phát triển, kiểm soát chi phí và nâng cao chất lượng phần mềm.

Dự đoán lỗi phần mềm tập trung vào việc thiết kế và phát triển các mô hình dự đoán dựa trên dữ liệu phần mềm (ví dụ: độ phức tạp của mã nguồn, lịch sử thay đổi, v.v.) để dự đoán lỗi trong các phần mã mới (tức là tệp, lớp, phương thức, v.v.). Các mô hình dự đoán này giúp các nhà phát triển phần mềm tập trung vào các mô-đun dễ xảy ra lỗi và tối ưu hóa nỗ lực cũng như tài nguyên của họ. Trong nhiều năm qua, các kỹ thuật học máy đã được áp dụng để phát triển các mô hình dự đoán lỗi phần mềm. Những kỹ thuật này sử dụng dữ liệu lỗi phần mềm lịch sử (thu thập từ các dự án phần mềm trước đó) để huấn luyện mô hình dự đoán và xác định các mô-đun có lỗi. Ban đầu, các mô hình dự đoán lỗi sử dụng các kỹ thuật học máy truyền thống như Naive Bayes, Support Vector Machine (SVM) và Decision Tree, được huấn luyện trên dữ liệu lỗi lịch sử và các độ đo mã nguồn. Tuy nhiên, các phương pháp này thường đối mặt với một số thách thức đáng kể:

1. Đặc trưng không liên quan hoặc dư thừa (Irrelevant or Redundant Features): Các tập dữ liệu lỗi thường có quá nhiều đặc trưng, trong đó không phải tất cả đều có giá trị phân loại. Sự dư thừa này làm tăng chi phí tính toán và có thể làm giảm hiệu suất của mô hình. Việc lựa chọn đặc trưng là cần thiết để loại bỏ thông tin không cần thiết, giúp mô hình học nhanh hơn và chính xác hơn.
2. Mất cân bằng lớp (Class Imbalance): Các tập dữ liệu lỗi phần mềm thường bị mất cân bằng nghiêm trọng, với số lượng mô-đun không lỗi (lớp đa số) lớn hơn rất nhiều so với mô-đun có lỗi (lớp thiểu số). Điều này khiến các mô hình học máy có xu hướng thiên vị lớp đa số, dẫn đến dự đoán không đáng tin cậy cho các mô-đun có lỗi.
3. Thiếu thông tin ngữ nghĩa và cú pháp (Lack of Semantic and Syntactic Information): Các mô hình dự đoán lỗi truyền thống thường dựa vào các độ đo thủ công, không thể nắm bắt đầy đủ cấu trúc cú pháp và ngữ nghĩa của mã nguồn. Hai tập

mã nguồn có cùng độ đo truyền thống nhưng có thể khác biệt về ngữ nghĩa, ảnh hưởng đến hiệu suất dự đoán. Do đó, các nghiên cứu gần đây đã chuyển sang học sâu để tự động trích xuất các đặc trưng này.

Nghiên cứu này tập trung giải quyết các thách thức trên thông qua ba đóng góp chính: (1) Trên cơ sở nâng cao hiệu quả dự đoán lỗi phần mềm thông qua việc đánh giá và so sánh có hệ thống các phương pháp lựa chọn đặc trưng filter và wrapper nhằm xác định tập đặc trưng tối ưu, góp phần giảm chiều dữ liệu, loại bỏ các đặc trưng dư thừa và không liên quan, từ đó cải thiện độ chính xác và cung cấp cơ sở thử nghiệm giúp lựa chọn phương pháp phù hợp cho bài toán dự đoán lỗi phần mềm; (2) Đề xuất và đánh giá thử nghiệm việc ứng dụng các mô hình sinh dữ liệu đối kháng GAN và các biến thể của chúng nhằm giải quyết vấn đề mất cân bằng dữ liệu trong các tập dữ liệu lỗi phần mềm dạng bảng. Áp dụng các mô hình GAN nhằm tạo dữ liệu tổng hợp chất lượng cao cho lớp thiểu số, từ đó cân bằng phân phối lớp, cải thiện độ chính xác và tăng khả năng khái quát hóa của các mô hình dự đoán lỗi phần mềm. Kết quả nghiên cứu cung cấp cơ sở thử nghiệm quan trọng giúp các nhà nghiên cứu lựa chọn phương pháp sinh dữ liệu phù hợp cho các bài toán học máy có dữ liệu mất cân bằng trong lĩnh vực công nghệ phần mềm; (3) Đề xuất mô hình học sâu dựa trên kiến trúc Transformer, kết hợp với biểu diễn ngữ nghĩa mã nguồn bằng kỹ thuật FastText embedding được trích xuất từ cây cú pháp trừu tượng. Mô hình được thiết kế nhằm khai thác và học tự động các đặc trưng cú pháp và ngữ nghĩa tiềm ẩn trong mã nguồn, từ đó nâng cao khả năng nhận diện và dự đoán các mô-đun dễ phát sinh lỗi. Khác với các phương pháp truyền thống dựa trên đặc trưng thủ công, mô hình được đề xuất có khả năng trích chọn biểu diễn véc-tơ ngữ nghĩa có ý nghĩa từ cấu trúc AST, đồng thời khắc phục hạn chế của các mạng tuần tự trong việc mô hình hóa các phụ thuộc phức tạp trong mã nguồn. Bằng cách sử dụng cơ chế tự chú ý (*self-attention*) của Transformer, mô hình có thể nắm bắt quan hệ ngữ cảnh toàn cục, góp phần cải thiện độ chính xác và tính khái quát hóa trong dự đoán lỗi phần mềm.

# Chương 1: Tổng quan dự đoán lỗi phần mềm

Chương này cung cấp cái nhìn tổng quan về dự đoán lỗi phần mềm, các kỹ thuật cải tiến hiệu suất của mô hình dự đoán lỗi dựa trên mã nguồn, và các thách thức liên quan.

## 1.1. Khái niệm và Quy trình dự đoán lỗi phần mềm

Dự đoán lỗi phần mềm tập trung vào việc xây dựng các mô hình dự đoán bằng cách sử dụng dữ liệu phần mềm lịch sử (như độ phức tạp mã nguồn, lịch sử thay đổi) để xác định lỗi trong các phần mã mới (ví dụ: tệp, lớp, phương thức). Mục tiêu là giúp các nhà phát triển tập trung vào các mô-đun dễ xảy ra lỗi và tối ưu hóa tài nguyên.

Quy trình dự đoán lỗi phần mềm bao gồm các bước chính sau:

1. Thu thập dữ liệu: Tập dữ liệu lỗi được lấy từ các kho dữ liệu công khai như PROMISE, NASA, Apache.
2. Tiền xử lý dữ liệu: Xử lý nhiễu, loại bỏ đặc trưng không liên quan/dư thừa, xử lý giá trị ngoại lai và cân bằng lớp dữ liệu để cải thiện hiệu suất. Các kỹ thuật như chuẩn hóa dữ liệu, lựa chọn đặc trưng, trích xuất đặc trưng và lấy mẫu dữ liệu được áp dụng.
3. Trích xuất đặc trưng và tạo tập dữ liệu huấn luyện: Các đặc trưng được trích xuất từ mã nguồn hoặc nhật ký lịch sử, sau đó kết hợp với thông tin lỗi để tạo tập dữ liệu huấn luyện.
4. Xây dựng mô hình dự đoán lỗi: Sử dụng các kỹ thuật thống kê và học máy để xây dựng mô hình.
5. Đánh giá hiệu suất: Đánh giá mô hình bằng tập dữ liệu kiểm thử, sử dụng các độ đo như accuracy, precision, recall, F1-score và AUC.
6. Điều chỉnh tham số: Tinh chỉnh tham số mô hình để đạt độ chính xác và hiệu suất cao nhất.
7. Dự đoán lỗi: Áp dụng mô hình đã huấn luyện để xác định mô-đun có lỗi trong các dự án thực tế.

## 1.2. Đặc trưng mã nguồn và Lựa chọn đặc trưng

Một yếu tố then chốt của bất kỳ quy trình kỹ thuật nào chính là đo lường. Các phép đo được sử dụng nhằm hiểu rõ hơn các thuộc tính của mô hình mà chúng ta xây dựng. Quan trọng hơn, chúng được dùng để đánh giá chất lượng của sản phẩm kỹ thuật hoặc quy trình tạo ra sản phẩm đó. Khác với các ngành kỹ thuật khác, công nghệ phần mềm không dựa trên các định luật định lượng cơ bản của vật lý. Những phép đo tuyệt đối như điện áp, khối lượng, vận tốc hay nhiệt độ hiếm khi xuất hiện trong lĩnh vực phần mềm. Thay vào đó, chúng ta tìm cách xây dựng một tập hợp các phép đo gián tiếp để hình thành nên các độ đo (metrics) nhằm phản ánh chất

lượng của một dạng biểu diễn phần mềm nào đó. Nhận thức được tầm quan trọng của các độ đo phần mềm, đã có nhiều độ đo được đề xuất để đo lường. Những độ đo này cố gắng nắm bắt các khía cạnh khác nhau của sản phẩm phần mềm cũng như quy trình phát triển phần mềm. Một số độ đo được sử dụng để đo lường cùng một khía cạnh của phần mềm; ví dụ, có nhiều độ đo được đề xuất để đo mức độ liên kết (coupling) giữa các lớp khác nhau. Do đó, các nhà phát triển phần mềm cần chỉ rõ mối quan hệ giữa các độ đo khác nhau cùng đo lường một khía cạnh phần mềm. Chẳng hạn, nếu chúng ta muốn biết kích thước của một chiếc bàn, có thể có nhiều độ đo liên quan như chiều dài, chiều rộng, diện tích, hay đường chéo của bàn. Tuy nhiên, các phép đo chiều dài và chiều rộng đã đủ, bởi các độ đo khác có thể được suy ra từ chúng khi cần thiết. Tương tự trong lĩnh vực phần mềm, cần xác định những độ đo thực sự cần thiết và cung cấp thông tin hữu ích; nếu không, nhà quản lý sẽ bị rối bởi quá nhiều con số và mục tiêu sử dụng độ đo sẽ bị sai lệch. Khi đối chiếu số lượng độ đo hiện có với số lượng đặc trưng cốt lõi mà chúng thực sự phản ánh, có thể nhận thấy một mức độ dư thừa đáng kể. Đặc trưng mã nguồn (Code Metrics) là các phép đo gián tiếp nhằm phản ánh chất lượng của phần mềm. Nhiều độ đo đã được đề xuất để đo lường các khía cạnh khác nhau như độ liên kết (coupling), độ phức tạp mã nguồn, số dòng mã (LOC). Tuy nhiên, việc có quá nhiều độ đo có thể dẫn đến sự dư thừa thông tin.

Lựa chọn đặc trưng (Feature Selection) là một kỹ thuật quan trọng để giảm chiều dữ liệu bằng cách loại bỏ các đặc trưng không liên quan hoặc dư thừa, giúp tăng tốc độ huấn luyện mô hình và cải thiện độ chính xác. Có bốn loại phương pháp lựa chọn đặc trưng chính:

- Filter: Đánh giá đặc trưng dựa trên thuộc tính nội tại (thông tin, phụ thuộc, tính nhất quán, khoảng cách), độc lập với thuật toán học máy.
- Wrapper: Chọn tập hợp đặc trưng tối ưu dựa trên hiệu suất của thuật toán học máy cụ thể, bằng cách giảm thiểu lỗi ước lượng của bộ phân loại. Phương pháp này thường tốn nhiều thời gian tính toán hơn so với filter.
- Embedded: Tích hợp quá trình lựa chọn đặc trưng vào thuật toán học, có hiệu suất tính toán cao hơn wrapper.
- Hybrid: Kết hợp hai hoặc nhiều phương pháp khác nhau để tận dụng ưu điểm bổ trợ.
- Ngoài ra, các thuật toán metaheuristic cũng được sử dụng rộng rãi trong lựa chọn đặc trưng, mặc dù không đảm bảo tìm được tổ hợp tối ưu nhất do tính chất ngẫu nhiên.

### 1.3. Trích xuất đặc trưng và Lấy mẫu dữ liệu

Trích xuất đặc trưng (Feature Extraction) là kỹ thuật biến đổi tập hợp đặc trưng đầu vào thành một tập hợp các đặc trưng mới, không dư thừa và có liên quan hơn, nhằm giảm độ phức tạp và cung cấp biểu diễn dễ hiểu hơn. Các phương pháp phổ biến bao gồm Phân tích thành phần chính (PCA) và Phân tích phân biệt tuyến tính (LDA).

Lấy mẫu dữ liệu (Data Sampling) giải quyết vấn đề mất cân bằng lớp bằng cách điều chỉnh sự phân bố của dữ liệu.

- **Oversampling:** Tạo ra các mẫu tổng hợp mới cho lớp thiểu số để tăng số lượng mẫu. Các kỹ thuật phổ biến là SMOTE và các biến thể (Borderline-SMOTE, ADASYN). Tuy nhiên, chúng có thể dẫn đến overfitting do các mẫu tổng hợp quá giống với mẫu hiện có.
- **Undersampling:** Loại bỏ một số mẫu thuộc lớp đa số để cân bằng tỷ lệ. Kỹ thuật phổ biến nhất là Random Under-Sampling (RUS). Nhược điểm là có thể làm mất thông tin quan trọng. Trong dự đoán lỗi phần mềm, oversampling thường được ưu tiên hơn undersampling.

#### **1.4. Dự đoán lỗi phần mềm dựa trên đặc trưng ngữ nghĩa mã nguồn**

Các mô hình học sâu, xây dựng trên mạng nơ-ron nhân tạo, đã được ứng dụng rộng rãi trong dự đoán lỗi phần mềm để khai thác thông tin ngữ nghĩa và cú pháp của mã nguồn. Thay vì dựa vào các độ đo thủ công, các mô hình này trích xuất các nút cụ thể từ cây cú pháp trừu tượng (Abstract Syntax Tree - AST) của mã nguồn và chuyển chúng thành các chuỗi/véc-tơ để đưa vào mô hình học sâu. Các mô hình học sâu phổ biến bao gồm Deep Belief Network (DBN), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), và Gated Recurrent Unit (GRU).

Tuy nhiên, các mạng tuần tự như RNN và LSTM vẫn có hạn chế trong việc biểu diễn đầy đủ thông tin cú pháp và ngữ nghĩa từ AST, cũng như gặp khó khăn trong việc nắm bắt các phụ thuộc phức tạp trong mã nguồn. Đồng thời, các vấn đề như vanishing gradient và exploding gradient vẫn có thể xảy ra.

## Chương 2: Nâng cao khả năng dự đoán lỗi phần mềm bằng các phương pháp lựa chọn đặc trưng Wrapper

Chương này nghiên cứu việc áp dụng các phương pháp lựa chọn đặc trưng wrapper để giảm dư thừa dữ liệu và nâng cao khả năng dự đoán lỗi phần mềm.

### 2.1. Giới thiệu

Lựa chọn đặc trưng là một bước quan trọng trong quá trình phân tích dữ liệu và áp dụng mô hình học máy. Mục tiêu của các phương pháp này là xác định tập các đặc trưng hoặc biến có mức độ liên quan cao nhất đến biến mục tiêu. Các mô hình phân loại được huấn luyện trên không gian đặc trưng đã được rút gọn thường có tính ổn định và khả năng tái lập cao hơn so với các mô hình được xây dựng trên toàn bộ không gian đặc trưng ban đầu có kích thước lớn. Kỹ thuật lựa chọn đặc trưng giúp cải thiện hiệu suất của mô hình bằng cách giảm độ phức tạp tính toán và tăng khả năng diễn giải của kết quả mô hình. Các phương pháp lựa chọn đặc trưng đóng vai trò then chốt trong quá trình xây dựng mô hình học máy. Trong quá trình lựa chọn đặc trưng, mục tiêu chính là tìm kiếm các đặc trưng có ý nghĩa hoặc các đặc trưng có tương quan cao. Các đặc trưng không cung cấp thông tin hữu ích được gọi là đặc trưng không liên quan trong khi các đặc trưng không mang thêm thông tin mới so với các đặc trưng đã được chọn được gọi là đặc trưng dư thừa. Bên cạnh đó, những đặc trưng không có mối quan hệ hoặc tương quan với biến mục tiêu được xem như là nhiễu. Sự tồn tại của nhiễu sẽ làm sai lệch quá trình dự đoán, từ đó làm giảm hiệu suất phân loại. Do đó, việc xử lý nhiễu là cần thiết để cải thiện hiệu quả dự đoán, và điều này có thể đạt được thông qua giảm chiều dữ liệu [54]. Một thuật toán lựa chọn đặc trưng tốt hơn nên mang lại nhiều lợi ích như cung cấp cái nhìn sâu sắc về dữ liệu, xây dựng mô hình phân loại hiệu quả hơn, tăng cường khả năng khái quát hóa và nhận diện các đặc trưng không liên quan. Bên cạnh đó, nó cũng giúp hiểu rõ hơn mối quan hệ giữa các đặc trưng và biến mục tiêu, giảm yêu cầu tính toán trong quá trình giải quyết một bài toán cụ thể. Trong các tập dữ liệu có số chiều lớn, nơi số thuộc tính vượt trội so với số lượng mẫu, việc lựa chọn đặc trưng thích hợp giúp giảm chiều dữ liệu, nâng cao hiệu suất mô hình, và tiết kiệm chi phí cũng như thời gian tính toán. Mục tiêu chính của lựa chọn đặc trưng wrapper là trích xuất một tập con các đặc trưng liên quan nhất từ tập dữ liệu ban đầu để giảm số chiều mà vẫn duy trì hoặc cải thiện hiệu suất phân loại. Các phương pháp wrapper thường cho kết quả vượt trội so với các phương pháp filter. Nghiên cứu này đánh giá thực nghiệm mười thuật toán metaheuristic khác nhau trong các phương pháp wrapper để giảm dư thừa dữ liệu trong các bộ dữ liệu dự đoán lỗi.

## 2.2. Phương pháp nghiên cứu

Nghiên cứu sử dụng chín bộ dữ liệu lỗi phần mềm từ kho PROMISE và AEEEM, bao gồm các biến độc lập như Lines of Code (LOCcode), Lines of Comments (LOComment) và biến phụ thuộc (lỗi/không lỗi). Các bộ dữ liệu này có đặc điểm là tỷ lệ mẫu không lỗi chiếm ưu thế so với mẫu lỗi, gây ra vấn đề mất cân bằng dữ liệu.

Mười phương pháp lựa chọn đặc trưng wrapper dựa trên metaheuristic đã được khảo sát:

- Artificial Butterfly Optimization (ABO)
- Atom Search Optimization (ASO)
- Equilibrium Optimizer (EO)
- Henry Gas Solubility Optimization (HGSO)
- Poor and Rich Optimization (PRO)
- Generalized Normal Distribution Optimization (GNDO)
- Slime Mould Algorithm (SMA)
- Harris Hawks Optimization (HHO)
- Path Finder Algorithm (PFA)
- Manta Ray Foraging Optimization (MRFO)
- Recursive Feature Elimination (RFE)

Các phương pháp này được so sánh với mô hình cơ sở (baseline), trong đó các thuật toán học máy được áp dụng trực tiếp lên dữ liệu gốc mà không qua lựa chọn đặc trưng.

Ba mô hình học máy được sử dụng để đánh giá hiệu suất phân loại:

- Random Forest (RF): Kết hợp nhiều cây quyết định, nổi tiếng về độ chính xác cao và khả năng chống overfitting.
- AdaBoost: Cải thiện các bộ phân loại yếu bằng cách huấn luyện tuần tự nhiều bộ phân loại và điều chỉnh trọng số.
- Extra Trees (ET): Biến thể của RF, trong đó lựa chọn thuộc tính được thực hiện ngẫu nhiên, giúp tăng tốc độ huấn luyện và giảm overfitting.

Các độ đo đánh giá hiệu suất bao gồm Precision, Recall, F1-score và Area Under the Curve (AUC). Kiểm định Wilcoxon signed-rank được sử dụng để xác định ý nghĩa thống kê của sự khác biệt hiệu suất giữa các kỹ thuật ở mức ý nghĩa 0.05. Mỗi thực nghiệm được lặp lại mười lần để đảm bảo độ tin cậy.

## 2.3. Kết quả thực nghiệm và phân tích

Nghiên cứu giải quyết hai câu hỏi chính:

1. Các kỹ thuật lựa chọn đặc trưng được áp dụng hiệu quả như thế nào trong việc nâng cao khả năng dự đoán lỗi phần mềm bằng cách loại bỏ các đặc trưng không liên quan hoặc dư thừa?
2. Phương pháp lựa chọn đặc trưng wrapper nào hoạt động tốt nhất trong việc chọn ra các đặc trưng tối ưu cho dự đoán lỗi phần mềm?

### 2.3.1. Hiệu quả của các kỹ thuật lựa chọn đặc trưng (Câu hỏi nghiên cứu 1)

Kết quả thực nghiệm cho thấy tất cả các phương pháp lựa chọn đặc trưng wrapper nói chung đều vượt trội hơn mô hình cơ sở (không áp dụng lựa chọn đặc trưng). Điều này chứng tỏ việc loại bỏ các đặc trưng không liên quan và dư thừa là một bước quan trọng để nâng cao hiệu suất dự đoán lỗi phần mềm.

- Với Random Forest: Recursive Feature Elimination (RFE) kết hợp với Random Forest đạt hiệu suất trung bình cao nhất trên tất cả các độ đo (Precision, Recall, AUC, F1-score), tiếp theo là Equilibrium Optimizer (EO), Poor and Rich Optimization (PRO).
- Với AdaBoost: RFE liên tục đạt giá trị Precision và AUC cao nhất trên phần lớn các bộ dữ liệu. EO đạt giá trị Recall và F1-score trung bình cao nhất.
- Với Extra Trees: RFE vượt trội hơn các phương pháp khác về Recall, F1-score và AUC trung bình. ASO đạt giá trị Precision trung bình cao nhất.

### 2.3.2. Phương pháp wrapper hoạt động tốt nhất (Câu hỏi nghiên cứu 2)

Dựa trên giá trị AUC, RFE, EO và PRO liên tục mang lại kết quả vượt trội trên nhiều bộ dữ liệu. Cụ thể, EO đạt các giá trị AUC cao nhất trên KC1, EQ, Lucene, PDE. PRO cũng đạt AUC đáng chú ý trên PC1, EQ, JDT.

RFE kết hợp với Random Forest đạt điểm AUC trung bình cao nhất (0.87), EO và Random Forest, tiếp theo là PRO kết hợp với Random Forest (0.84). Điều này khẳng định RFE là phương pháp lựa chọn đặc trưng hiệu quả nhất cho dự đoán lỗi phần mềm, giúp cải thiện hiệu suất của các bộ phân loại. EO, PRO và HGSO cũng thể hiện hiệu suất mạnh mẽ và ổn định.

### 2.3.3. Kết quả kiểm định thống kê

Kiểm định Wilcoxon signed-rank ở mức ý nghĩa  $\alpha = 0.05$  đã được thực hiện để so sánh các phương pháp wrapper với mô hình cơ sở.

- RFE, ASO, HGSO, PRO, PFA và MRFO đạt cải thiện F1-score có ý nghĩa thống kê so với phương pháp cơ sở khi sử dụng Random Forest.
- Ví dụ, ASO có P-value là 0.026 và effect size  $r = 0.913$ , cùng chênh lệch trung bình 0.071.
- EO thể hiện effect size  $r$  trung bình nhưng P-value không có ý nghĩa thống kê (0.293).
- SMA và HHO không có sự khác biệt thống kê đáng kể.

Điều này nhấn mạnh rằng không phải mọi khác biệt về giá trị số đều có ý nghĩa thống kê. Tổng thể, RFE, ASO, HGSO, PRO, PFA, và MRFO được khuyến nghị triển khai trong các mô hình dự đoán lỗi thực tế do hiệu quả và ý nghĩa thống kê.

## 2.4. Tổng kết chương

Chương 2 đã chứng minh rằng việc loại bỏ các đặc trưng không liên quan và dư thừa bằng các phương pháp lựa chọn đặc trưng wrapper là thiết yếu để xây dựng mô hình dự đoán lỗi phần mềm vững chắc. Các phương pháp wrapper được đánh giá đều cho hiệu suất tốt hơn so với mô hình cơ sở. Trong đó, RFE là phương pháp hiệu quả nhất, tiếp theo là RFE, EO, PRO và HGSO. Tổng hợp các kết quả thực nghiệm cho thấy rằng việc kết hợp lựa chọn đặc trưng và kỹ thuật cân bằng dữ liệu thích hợp có thể cải thiện đáng kể hiệu năng của các mô hình dự đoán lỗi phần mềm, cả về độ chính xác lẫn tính ổn định.

Tuy nhiên, nghiên cứu này vẫn còn một số hạn chế: các bộ dữ liệu được sử dụng là tĩnh và tương đối cân bằng, có thể chưa phản ánh đầy đủ các tình huống thực tế. Việc đánh giá chỉ trên ba bộ phân loại cũng có thể hạn chế khả năng khái quát hóa. Trong tương lai, nghiên cứu sẽ tích hợp các phương pháp lựa chọn đặc trưng wrapper với các kỹ thuật lấy mẫu (như SMOTE, ADASYN) để giải quyết vấn đề dữ liệu mất cân bằng và có chiều dữ liệu cao.

## **Chương 3: Kết hợp các kỹ thuật lựa chọn đặc trưng và lấy mẫu dữ liệu trong dự đoán lỗi phần mềm**

Chương này tập trung nghiên cứu các mô hình lấy mẫu dữ liệu nhằm giải quyết vấn đề mất cân bằng dữ liệu, từ đó cải thiện độ chính xác và hiệu quả của mô hình dự đoán lỗi.

### **3.1. Giới thiệu**

Để phát triển các mô hình dự đoán lỗi phần mềm, nhiều kỹ thuật học máy đã được nghiên cứu và cho thấy kết quả triển vọng trên các bộ dữ liệu lỗi phần mềm. Trong quá trình huấn luyện, mô hình dự đoán sẽ học các đặc trưng của các dự án phần mềm trước đó và đưa ra dự đoán liệu một mô-đun mới có khả năng bị lỗi hay không. Tuy nhiên, trong nhiều trường hợp, số lượng mô-đun có lỗi trong tập dữ liệu huấn luyện rất nhỏ so với số lượng mô-đun không có lỗi, dẫn đến vấn đề mất cân bằng dữ liệu. Điều này dẫn đến các mô hình dự đoán được xây dựng trên các bộ dữ liệu phần mềm mất cân bằng thường không thể học được các mẫu đặc trưng của các mô-đule lỗi và do đó có xu hướng dự đoán sai nhiều mô-đun lỗi. Do đó, các mô hình dự đoán trở nên thiếu tin cậy và khó có thể được ứng dụng trong thực tế. Vì vậy, việc đề xuất các kỹ thuật hoặc phương pháp nhằm khắc phục vấn đề mất cân bằng lớp trong các bộ dữ liệu lỗi đã và đang được xem là một vấn đề quan trọng trong lĩnh vực dự đoán lỗi phần mềm. Nhiều nghiên cứu trước đây đã được công bố nhằm phát triển các phương pháp khác nhau để giải quyết vấn đề mất cân bằng lớp này. Các kỹ thuật lấy mẫu dữ liệu như oversampling (tạo mẫu mới cho lớp thiểu số) và undersampling (giảm mẫu lớp đa số) được sử dụng để cân bằng dữ liệu. Tuy nhiên, oversampling truyền thống (như SMOTE) có thể dẫn đến overfitting do các mẫu tổng hợp quá giống nhau, trong khi undersampling có thể làm mất thông tin quan trọng.

Generative Adversarial Networks (GAN) gần đây đã được áp dụng thành công để giải quyết vấn đề mất cân bằng dữ liệu trong nhiều lĩnh vực, bao gồm tạo ảnh và video, tăng cường dữ liệu và xử lý ngôn ngữ tự nhiên. Các kỹ thuật lấy mẫu lại hiện có như SMOTE chủ yếu dựa vào thông tin cục bộ (tức là các điểm dữ liệu lân cận) để tạo ra các mẫu dữ liệu tổng hợp. Tuy nhiên, phương pháp GAN có khả năng học được phân phối tổng thể của dữ liệu trên các đặc trưng khác nhau cũng như mối tương quan giữa các đặc trưng đó. Nhờ vậy, GAN có thể tạo ra các mẫu dữ liệu tổng hợp có tính chân thực cao hơn so với các kỹ thuật truyền thống. GAN ban đầu được phát triển để tạo ra hình ảnh, dữ liệu dạng bảng và các loại dữ liệu khác. Các phương pháp tiếp cận dựa trên GAN đã được chứng minh có khả năng học phân phối dữ liệu tốt hơn so với các kỹ thuật lấy mẫu dữ liệu truyền thống và giúp giải quyết hiệu quả vấn đề mất cân bằng dữ liệu. Mạng GAN gần đây đã được áp dụng thành công để giải quyết vấn đề mất cân bằng dữ liệu. Không giống SMOTE chỉ dựa vào thông tin cục bộ, GAN có khả năng

học được phân phối tổng thể của dữ liệu trên các đặc trưng khác nhau và mối tương quan giữa chúng, giúp tạo ra các mẫu tổng hợp chân thực và đa dạng hơn.

Nghiên cứu này đánh giá thực nghiệm năm biến thể của GAN để tạo dữ liệu lỗi phần mềm dạng bảng: VanillaGAN, CTGAN, WGAN-GP, CopulaGAN và TGAN. Các mẫu tổng hợp do GAN tạo ra có sự tương đồng cao với phân phối dữ liệu gốc của lớp thiểu số, giúp giải quyết hiệu quả vấn đề mất cân bằng.

Nghiên cứu sử dụng năm tập dữ liệu lỗi từ kho PROMISE, áp dụng sáu thuật toán học máy (Random Forest, XGBoost, HGB, MLP, Extra Trees, AdaBoost) và đánh giá bằng Precision, Recall, F1-score, AUC, và MCC. Kiểm định Wilcoxon cũng được sử dụng để đánh giá sự khác biệt ý nghĩa thống kê. Ngoài ra, hiệu suất giữa các mô hình GAN và các mô hình lấy mẫu cơ sở, bao gồm SMOTE, ADASYN, Borderline-SMOTE và RUS đã được thực hiện so sánh nhằm xác thực hiệu quả của việc tạo dữ liệu tổng hợp trong dự đoán lỗi phần mềm.

### 3.2. Phương pháp đề xuất

Quy trình nghiên cứu bao gồm các bước chính:

1. Thu thập dữ liệu: Năm tập dữ liệu lỗi từ PROMISE (CM1, KC1, KC2, PC1, JM1) được sử dụng, có đặc điểm không cân bằng về số mẫu lỗi và không lỗi.
2. Tiền xử lý dữ liệu: Điền giá trị thiếu, chuẩn hóa dữ liệu bằng z-normalization, và chia dữ liệu thành tập huấn luyện (70%) và kiểm thử (30%) bằng StratifiedKFold.
3. Lựa chọn đặc trưng và trích xuất đặc trưng:
  - Recursive Feature Elimination (RFE) được chọn làm phương pháp lựa chọn đặc trưng hiệu quả nhất. RFE loại bỏ đệ quy các đặc trưng dư thừa và không liên quan, giúp giảm chiều dữ liệu và tăng độ chính xác. Kết quả so sánh với ABO, ASO, HGSO, PRO cho thấy RFE đạt hiệu suất cao nhất trên tất cả các tập dữ liệu.
  - Autoencoders được sử dụng để trích xuất đặc trưng, dựa trên nghiên cứu của Malhotra và cộng sự.
4. Lấy mẫu dữ liệu bằng GAN: Năm biến thể GAN (VanillaGAN, CTGAN, WGAN-GP, CopulaGAN, TGAN) được áp dụng lên các tập con huấn luyện đã được xử lý đặc trưng (bằng RFE hoặc Autoencoders) để tạo ra các tập dữ liệu cân bằng.
5. Huấn luyện và đánh giá mô hình: Các tập dữ liệu huấn luyện đã cân bằng được đưa vào các thuật toán học máy để xây dựng mô hình dự đoán. Các mô hình này được đánh giá bằng Precision, Recall, F1-score, AUC và MCC trên tập kiểm thử.

### 3.3. Kết quả thực nghiệm và phân tích

Nghiên cứu trả lời hai câu hỏi chính:

1. Các biến thể của GAN có hiệu quả như thế nào trong việc lấy mẫu dữ liệu để xử lý mất cân bằng lớp trong dự đoán lỗi phần mềm?
2. Trong số các kỹ thuật tiên tiến nhất hiện nay, biến thể nào của GAN vượt trội hơn trong việc xử lý mất cân bằng dữ liệu trong dự đoán lỗi phần mềm?

#### 3.3.1. Kết quả kết hợp RFE và các mô hình GAN (Câu hỏi nghiên cứu 1)

- Với Random Forest: WGANGP đạt hiệu suất cao nhất trên tất cả các độ đo, tiếp theo là VanillaGAN và CTGAN.
- Với XGBoost: CTGAN thể hiện hiệu suất tốt nhất, tiếp theo là WGANGP và VanillaGAN. WGANGP nổi bật trên PC1, còn CTGAN trên CM1, KC1, KC2.
- Với AdaBoost: CTGAN vượt trội về recall và F1-score trên hầu hết các tập dữ liệu (CM1, KC1, JM1). WGANGP tốt nhất trên PC1. CopulaGAN đạt precision cao nhất trên CM1, KC1, KC2.
- Với HGBost: Tương tự XGBoost, CTGAN đạt kết quả tốt nhất, tiếp theo là WGANGP và VanillaGAN.
- Với Multilayer Perceptron (MLP): Các mô hình GAN đều cho hiệu suất tốt hơn. CopulaGAN đạt precision cao nhất trên CM1, PC1, KC1, JM1. WGANGP đạt AUC cao nhất trên PC1, KC2, JM1.
- Với Extra Trees: CTGAN vượt trội trên CM1, KC1, KC2, JM1 về recall, F1-score, AUC, MCC. CopulaGAN đạt precision tốt nhất trên CM1 và KC1. WGANGP tốt nhất trên PC1.

Tổng quan, các phương pháp GAN (đặc biệt là CTGAN, WGANGP và VanillaGAN) mang lại hiệu suất tốt hơn so với các phương pháp lấy mẫu cơ sở truyền thống (SMOTE, ADASYN, Borderline-SMOTE, RUS). Hiệu suất dự đoán lỗi được cải thiện từ 1-4% so với các mô hình cơ sở.

#### 3.3.2. Kết quả kết hợp Autoencoders và các mô hình GAN (Câu hỏi nghiên cứu 1 tiếp theo)

- Với Random Forest: CopulaGAN đạt giá trị trung bình cao nhất về recall và F1-score, và tốt nhất về precision và MCC.
- Với XGBoost: CopulaGAN đạt hiệu suất tốt hơn trên CM1, KC2, JM1.
- Với AdaBoost: CopulaGAN đạt precision cao nhất, còn CTGAN cho kết quả tốt hơn về recall và F1-score.

- Với HGBost: CTGAN cho thấy hiệu suất tốt hơn trên CM1, JM1, đạt các giá trị cao nhất về recall, F1-score, AUC trung bình.
- Với Multilayer Perceptron (MLP): CTGAN cũng đạt các giá trị cao nhất về recall, F1-score, AUC trung bình.
- Với Extra Trees: WGANGP đạt giá trị cao nhất về precision, recall và F1-score trung bình.

Kết quả cho thấy sự kết hợp giữa Autoencoders và CTGAN mang lại hiệu suất tốt nhất, tiếp theo là cặp Autoencoders và CopulaGAN. Trích xuất đặc trưng bằng Autoencoders là hiệu quả.

### 3.3.3. Phương pháp GAN vượt trội nhất (Câu hỏi nghiên cứu 2)

Hình 3.2 và các phân tích cho thấy CTGAN kết hợp với Extra Trees vượt trội hơn tất cả các phương pháp lấy mẫu khác về giá trị AUC trung bình (0.806). Các mô hình GAN được đề xuất đều có giá trị AUC trung bình lớn hơn 0.704. TGAN và CopulaGAN cho kết quả tương tự như SMOTE, ADASYN, RUS và BorderMOTE.

Do đó, CTGAN là phương pháp hiệu quả nhất để xử lý vấn đề mất cân bằng dữ liệu trong dự đoán lỗi phần mềm, tiếp theo là WGANGP và VanillaGAN. Điều này là do kiến trúc GAN có khả năng học được phân phối của các lớp lỗi, tạo ra các mẫu tổng hợp đa dạng và chất lượng cao hơn.

### 3.3.4. Kết quả kiểm định thống kê

Kiểm định Wilcoxon (mức ý nghĩa  $\alpha = 0.05$ ) đã được thực hiện để đánh giá các biến thể GAN so với các phương pháp lấy mẫu truyền thống.

- VanillaGAN có sự khác biệt hiệu suất có ý nghĩa thống kê so với BorderSMOTE và RUS về Recall, F1-score, và MCC.
- CTGAN có sự khác biệt ý nghĩa thống kê so với hầu hết các phương pháp (trừ SMOTE) trên nhiều độ đo.
- WGANGP cũng có sự khác biệt ý nghĩa thống kê so với các phương pháp khác.
- CopulaGAN và TGAN có sự khác biệt ý nghĩa thống kê về Precision và MCC, nhưng giá trị trung bình có thể thấp hơn ở một số độ đo khác.

Các kiểm định thống kê này xác nhận rằng CTGAN, WGANGP và VanillaGAN mang lại hiệu suất dự đoán tốt hơn và vượt trội hơn so với các phương pháp lấy mẫu dữ liệu truyền thống.

### 3.4. Tổng kết chương

Chương 3 đã chứng minh rằng các đặc trưng dư thừa và mất cân bằng lớp là những yếu tố chính ảnh hưởng đến hiệu suất mô hình dự đoán lỗi. Luận án đề xuất hướng tiếp cận sử dụng các biến thể của mạng sinh đối kháng GAN bao gồm VanillaGAN, CTGAN, WGANGP, CopulaGAN và TGAN cho tổng hợp dữ liệu lỗi phần mềm dạng bảng. Ngoài ra, để chọn một tập đặc trưng tối ưu, chúng tôi sử dụng kỹ thuật lựa chọn đặc trưng RFE và trích xuất đặc trưng Autoencoders trên tập dữ liệu mất cân bằng, sau đó áp dụng các biến thể GAN để tạo ra các mẫu mới cho lớp thiểu số nhằm cân bằng tập dữ liệu. Tiếp theo, chúng tôi kiểm tra hiệu suất của các thuật toán học máy trên tập dữ liệu đã được cân bằng (Random Forest, XGBoost, MLP, AdaBoost, Extra Trees và HistGradientBoosting), sử dụng tập đặc trưng tối ưu được trích xuất từ các bước trước đó. Chúng tôi đã tiến hành các thử nghiệm trên năm tập dữ liệu lỗi phần mềm và sử dụng các chỉ số đánh giá như precision, recall, F1-score, AUC và MCC để đánh giá hiệu quả của các mô hình GAN trong việc lấy mẫu dữ liệu bổ sung. Ngoài ra, nghiên cứu này đã thực hiện phân tích so sánh giữa hiệu suất của các biến thể GAN và các phương pháp lấy mẫu dữ liệu truyền thống. Bằng cách kết hợp kỹ thuật lựa chọn đặc trưng RFE/trích xuất đặc trưng Autoencoders với các biến thể GAN, nghiên cứu đã cải thiện đáng kể hiệu suất của các mô hình học máy trong môi trường dữ liệu mất cân bằng. CTGAN, WGANGP và VanillaGAN là các phương pháp lấy mẫu hiệu quả nhất. Trong tương lai, các phương pháp này có thể được mở rộng cho bài toán dự đoán lỗi giữa các dự án và các ngôn ngữ lập trình khác.

## Chương 4: Xây dựng mô hình dự đoán lỗi phần mềm dựa trên Transformer sử dụng Syntax Tree và Word Embedding

Chương này nghiên cứu việc áp dụng kỹ thuật học sâu để xây dựng các mô hình dự đoán lỗi dựa trên việc học các đặc trưng ngữ nghĩa và cú pháp của mã nguồn, nhằm nâng cao hiệu quả dự đoán so với các mô hình truyền thống.

### 4.1. Giới thiệu

Các mô hình học sâu là một hướng tiếp cận đầy hứa hẹn để học các đặc trưng ngữ nghĩa ẩn, nâng cao hiệu suất dự đoán lỗi. Tuy nhiên, các mạng tuần tự (RNN, LSTM) gặp khó khăn trong việc biểu diễn đầy đủ thông tin cú pháp và ngữ nghĩa từ cây cú pháp trừu tượng (AST) và nắm bắt các phụ thuộc phức tạp trong mã nguồn.

Kiến trúc Transformer, dựa trên cơ chế self-attention và khả năng tính toán song song, đã chứng minh hiệu quả trong nhiều lĩnh vực và có thể khắc phục vấn đề vanishing/exploding gradients.

Chương này đề xuất mô hình dự đoán lỗi dựa trên Transformer, sử dụng Gensim FastText với nhúng (embedding) các từ tố (token) từ cây cú pháp trừu tượng (AST). AST là một cây biểu diễn cấu trúc của mã nguồn, được sử dụng để mô tả cấu trúc cú pháp và thông tin ngữ nghĩa của mã nguồn, chẳng hạn như cấu trúc luồng điều khiển với các câu lệnh “while, do, for” cũng như tên của phương thức.

Cụ thể, đối với mỗi tệp dự án, trước tiên chúng tôi xây dựng AST từ mã nguồn, sau đó trích xuất các nút AST để tạo các vec-tơ token. Do đó, mỗi tệp mã nguồn được biểu diễn bằng các vec-tơ token. Tiếp theo, chúng tôi sử dụng một mô hình biểu diễn từ dưới dạng vec-tơ đã được huấn luyện trước, có tên Gensim FastText để chuyển đổi các vec-tơ token thành các vec-tơ số có độ dài cố định. Các vec-tơ thu thập được này sau đó được đưa vào mô hình Transformer để thực hiện dự đoán lỗi. Đóng góp chính của chương này là đề xuất mô hình dự đoán lỗi dựa trên FastText-Transformer, có khả năng học các đặc trưng ngữ nghĩa quan trọng của chương trình phần mềm giúp cải thiện đáng kể hiệu quả dự đoán lỗi. Mô hình có thể nắm bắt thông tin ngữ nghĩa trong các biểu diễn vec-tơ của các token mã nguồn, áp dụng cho cả trong cùng dự án và giữa các dự án.

Ngoài ra, chúng tôi thực hiện kiểm định Wilcoxon để đánh giá mức độ khác biệt có ý nghĩa thống kê giữa mô hình FastText-Transformer và các mô hình đối sánh. Kết quả đánh giá trên 10 dự án Java cho thấy mô hình được đề xuất của chúng tôi vượt trội hơn so với các mô hình dự đoán lỗi mới hiện nay. Đóng góp chính là đề xuất một mô hình dự đoán lỗi dựa trên

Transformer, có khả năng học các đặc trưng ngữ nghĩa quan trọng của chương trình phần mềm, cải thiện đáng kể hiệu quả dự đoán lỗi cho cả dự án nội bộ (Within-Project Defect Prediction - WPDP) và dự án chéo (Cross-Project Defect Prediction - CPDP). Mô hình cũng có thời gian huấn luyện hiệu quả hơn.

#### **4.2. Các nghiên cứu liên quan và vấn đề nghiên cứu**

Các nghiên cứu gần đây đã áp dụng các mô hình học sâu để trích xuất các thông tin ngữ nghĩa và cú pháp của mã nguồn. Các chương trình có cú pháp được xác định rõ ràng và có thể được biểu diễn bằng cây cú pháp trừu tượng [173] và đã được sử dụng thành công để nắm bắt các thông tin mã nguồn. Nhiều nghiên cứu đã chỉ ra rằng thông tin ngữ nghĩa của chương trình có ích cho các tác vụ như tự động hoàn thành mã và phát hiện lỗi [174]. Thông tin ngữ nghĩa này cũng có thể hữu ích trong việc mô tả lỗi nhằm cải thiện dự đoán lỗi. Cụ thể, các thông tin ngữ nghĩa này có thể đưa ra các dự đoán chính xác, các đặc trưng cần có tính phân biệt, tức là có khả năng phân biệt các vùng mã khác nhau. Các mô hình dự đoán lỗi truyền thống thường bỏ qua cấu trúc cú pháp và ngữ nghĩa của mã nguồn. Các nghiên cứu gần đây đã áp dụng học sâu để trích xuất các thông tin này từ AST.

Word Embedding là kỹ thuật biểu diễn từ/cụm từ dưới dạng véc-tơ trong không gian liên tục, giúp các thuật toán học máy hiểu ngôn ngữ tự nhiên tốt hơn. FastText là một thuật toán hiệu quả, xem một từ là tập hợp các ký tự n-grams, cho phép tạo biểu diễn véc-tơ cho các từ chưa từng xuất hiện. Nghiên cứu này sử dụng Gensim FastText để học biểu diễn véc-tơ cho các token từ AST của các dự án Java.

Các mô hình học sâu khác như CNN, DBN, LSTM, GRU cũng đã được sử dụng trong dự đoán lỗi. Tuy nhiên, các mô hình tuần tự này không đủ mạnh để học các mối quan hệ phức tạp và ngữ cảnh ngắn trong mã nguồn, dễ gặp vấn đề vanishing/exploding gradients.

#### **Vấn đề nghiên cứu:**

Sự khác biệt giữa giải pháp được đề xuất trong luận án này và các nghiên cứu trên được trình bày như sau. Đầu tiên, các phương pháp truyền thống sử dụng các độ đo phần mềm để xây dựng các mô hình dự đoán. Tuy nhiên, những độ đo này không thể nắm bắt được các đặc trưng cú pháp và ngữ nghĩa của mã nguồn. Giải pháp đề xuất có thể nắm bắt thông tin ngữ nghĩa bằng cách sử dụng FastText, một mô hình học sâu để học các biểu diễn vector từ AST của chương trình. Thứ hai, các nghiên cứu trước đây sử dụng các mô hình học sâu tuần tự như DBN, RNN và LSTM, những mô hình này gặp phải vấn đề về gradient biến mất và gradient bùng nổ. Bằng cách sử dụng kiến trúc Transformer song song với cơ chế self-attention và có khả năng song song hóa, phương pháp đề xuất của chúng tôi khắc phục được các vấn đề như hiện tượng mất mát gradient và rút ngắn đáng kể thời gian huấn luyện so với các mô hình tuần

tự như RNN và LSTM. Tổng thể, phương pháp này tích hợp ba thành phần chính trong một quy trình thống nhất nhằm nâng cao độ chính xác trong dự đoán lỗi. Trước tiên, mã nguồn được phân tích cú pháp thành cây AST để nắm bắt thông tin cấu trúc và cú pháp. Tiếp theo, các token trích xuất từ AST được nhúng bằng FastText để duy trì sự tương đồng ngữ nghĩa, bao gồm cả các mẫu ở cấp độ từ con. Cuối cùng, mô hình Transformer Encoder với cơ chế tự chú ý có khả năng nắm bắt các phụ thuộc dài hạn và mối quan hệ phân cấp giữa các thành phần mã nguồn một cách hiệu quả. Sự kết hợp này cho phép trích xuất hiệu quả các đặc trưng ngữ nghĩa và cấu trúc, vốn đóng vai trò then chốt trong việc xác định các đoạn mã dễ phát sinh lỗi.

### 4.3. Đề xuất giải pháp

Giải pháp đề xuất, Gensim FASTText-Transformer, gồm ba phần chính:

1. Biểu diễn mã nguồn và trích xuất đặc trưng: Công cụ javalang được sử dụng để chuyển đổi tệp Java thành AST. Từ AST, các nút liên quan đến lớp (Class Node), khai báo (Declaration Node), và điều khiển (Control Node) được trích xuất làm token. Thuật toán Breadth-First Search (BFS) được dùng để duyệt AST và thu thập các token, tạo ra các "context windows".
2. Áp dụng FastText cho Token Embedding: Mô hình Gensim FastText được sử dụng để chuyển đổi các token thành các véc-tơ số có độ dài cố định (100). FastText có khả năng xử lý các từ con (sub-word information) và từ ngoài từ vựng, giúp tạo ra các biểu diễn ý nghĩa hơn cho các token chưa thấy.
3. Bộ mã hóa Transformer (Transformer Encoder): Các véc-tơ embedding từ FastText được đưa vào kiến trúc Transformer. Transformer sử dụng cơ chế self-attention và feed-forward, cùng các lớp normalization và dropout, để xử lý các véc-tơ từ. Kiến trúc này cho phép xử lý song song, nắm bắt các phụ thuộc phức tạp và giảm thiểu overfitting. Mô hình được huấn luyện với hàm mất mát "sparse categorical cross-entropy" và bộ tối ưu hóa "Adam".

### Câu hỏi nghiên cứu:

1. Mô hình FastText-Transformer được đề xuất có vượt trội hơn so với các mô hình dự đoán lỗi tiên tiến hiện nay đối với dự đoán lỗi trong cùng một dự án không?
2. Mô hình FastText-Transformer được đề xuất có vượt trội hơn so với các mô hình dự đoán lỗi tiên tiến hiện nay đối với dự đoán lỗi giữa các dự án không?

#### 4.4. Triển khai thực nghiệm, kết quả và phân tích

Nghiên cứu đánh giá mô hình đề xuất trên mười tập dữ liệu Apache (Ant, Camel, Jedit, Log4j, Lucene, Poi, Synapse, Ivy, Xalan, Xerces), được sử dụng rộng rãi trong nghiên cứu SDP.

Các mô hình so sánh bao gồm:

- Seml
- DP-Transformer
- DBN-CP
- DTL-DP
- TCA+
- Và FastText-LSTM (một mô hình đối chứng được xây dựng để so sánh trực tiếp với FastText-Transformer).

Các độ đo đánh giá hiệu suất là Precision, Recall và F1-score. Kiểm định Wilcoxon signed-rank được sử dụng để đánh giá ý nghĩa thống kê.

Tham số huấn luyện: Các tham số như Epochs (200), Batch-size (16), Dropout (0.2-0.25 cho Transformer, 0.5 cho LSTM), Learning Rate ( $1e-4$  cho Transformer, 0.001 cho LSTM), Activation Function (ReLU, Softmax), và Optimizer (Adam) được điều chỉnh để tối ưu hóa hiệu suất.

##### 4.4.1. Dự đoán lỗi trong cùng một dự án (WPDP - Câu hỏi nghiên cứu 1)

Kết quả so sánh hiệu suất WPDP như sau:

- Mô hình FastText-Transformer đạt hiệu suất tốt hơn so với các phương pháp khác trên 11 trong số 19 tập dữ liệu.
- FastText-Transformer đạt giá trị trung bình F1-score là 0.768, Precision là 0.762, và Recall là 0.807.
- So với DTL-DP, Seml, DBN-CP, và FastText-LSTM, mô hình đề xuất có F1-score trung bình cao hơn lần lượt 19.6%, 25.4%, 19.8% và 7%.
- Hình 4.5 cho thấy FastText-Transformer vượt trội hơn DP-Transformer trên Lucene và Synapse, và tương đương trên Poi.
- Kết quả thống kê (Bảng 4.10) cho thấy sự khác biệt hiệu suất có ý nghĩa thống kê giữa FastText-Transformer và hầu hết các mô hình so sánh ( $P\text{-value} < 0.05$ ). Giá trị effect size  $r$  cao ( $>0.2$ ) cũng chỉ ra sự khác biệt đáng kể.

Điều này chứng tỏ FastText-Transformer hiệu quả trong việc nắm bắt các đặc trưng ngữ nghĩa của chương trình, cải thiện hiệu suất dự đoán lỗi trong cùng dự án.

#### 4.4.2. Dự đoán lỗi giữa các dự án (CPDP - Câu hỏi nghiên cứu 2)

Giá trị F1-score cho CPDP như sau:

- Mô hình Gensim FastText-Transformer đạt hiệu suất tốt hơn so với các phương pháp còn lại về giá trị F1-score trên 12 trong số 22 tập thực nghiệm.
- Giá trị trung bình F1-score của Gensim FastText-Transformer trong CPDP là 0.667, vượt trội hơn DTL-DP (0.618), TCA+ (0.479), DBN-CP (0.568) và FastText-LSTM (0.589) lần lượt là 7.9%, 39.2%, 17.4% và 13.2%.
- Sự vượt trội này cũng rõ rệt so với FastText-LSTM trên hầu hết các tập dữ liệu thực nghiệm (ví dụ: trên Poi 2.5 -> Synapse 1.2, FastText-Transformer đạt 0.774, cao hơn 0.274 so với 0.50 của FastText-LSTM).
- Kết quả thống kê cũng xác nhận sự khác biệt hiệu suất có ý nghĩa thống kê giữa FastText-Transformer và các mô hình khác trong tất cả các trường hợp CPDP (P-value < 0.05,  $r > 0.6$ ).

Hiệu suất vượt trội này là nhờ cơ chế self-attention của Transformer, giúp cải thiện khả năng dự đoán lỗi phần mềm.

#### 4.4.3. So sánh chi phí thời gian huấn luyện

So sánh chi phí thời gian huấn luyện giữa FastText-LSTM và FastText-Transformer (tính bằng giây) cho kết quả như sau:

- FastText-Transformer thể hiện lợi thế rõ rệt về hiệu suất thời gian, với chi phí thấp hơn đáng kể cho tất cả các dự án.
- Thời gian huấn luyện trung bình của FastText-Transformer là 2973.26 giây, thấp hơn đáng kể so với 4937.55 giây của FastText-LSTM.
- Ví dụ, trên dự án Camel, FastText-Transformer chỉ mất 160.67 giây so với 765.81 giây của FastText-LSTM.

Ưu điểm này là do khả năng xử lý song song của kiến trúc Transformer, không phụ thuộc vào tính toán tuần tự như LSTM hay RNN.

#### 4.4.4. Các mối đe dọa đến tính tin cậy và giá trị của nghiên cứu

Nghiên cứu cũng đã phân tích các mối đe dọa tiềm tàng đến độ tin cậy và giá trị.

1. Độ giá trị ngoại tại (External Validity): Kết quả có thể không tổng quát hóa hoàn toàn do tập dữ liệu hạn chế (chỉ các dự án Java từ PROMISE/Apache). Tương lai sẽ mở rộng sang các loại ứng dụng và ngôn ngữ lập trình khác (Python, C++, JavaScript).
2. Độ giá trị nội tại (Internal Validity): Mã nguồn của các nghiên cứu đối sánh không có sẵn, nên việc tái tạo thực nghiệm có thể không hoàn toàn nhất quán. Việc sử dụng các giá trị mặc định cho siêu tham số cũng là một mối đe dọa, sẽ được cải thiện bằng cách điều chỉnh tham số kỹ lưỡng hơn trong tương lai.
3. Độ giá trị khái niệm (Construct Validity): Các độ đo đánh giá (Precision, Recall, F1-score) là phổ biến, nhưng có thể xem xét thêm các độ đo khác để đánh giá toàn diện hơn trong tương lai.

#### 4.5. Tổng kết chương

Chương 4 đã đề xuất mô hình Gensim FastText-Transformer để cải thiện khả năng học đặc trưng và nâng cao hiệu suất dự đoán lỗi phần mềm. Mô hình này tận dụng Gensim FastText để chuyển đổi token thành véc-tơ và đưa vào Transformer để dự đoán lỗi. Kết quả thực nghiệm trên 10 dự án Apache cho thấy FastText-Transformer đạt kết quả vượt trội về precision, recall và F1-score trên cả dự đoán lỗi phần mềm trong cùng dự án và giữa các dự án. Mô hình cũng vượt trội hơn nhiều mô hình so sánh trong hầu hết các dự án.

Trong tương lai, để tăng tính tổng quát, mô hình có thể được thử nghiệm trên các tập dữ liệu từ nhiều lĩnh vực và ngôn ngữ lập trình khác nhau, và kết hợp với các kỹ thuật lấy mẫu (oversampling, undersampling) để xử lý dữ liệu mất cân bằng. Ngoài ra, có thể mở rộng để dự đoán lỗi ở các cấp độ chi tiết hơn như lớp, phương thức hoặc câu lệnh.

# Kết luận và hướng phát triển

## Kết luận

Đề tài đã thành công trong việc nghiên cứu và đề xuất các phương pháp nhằm cải thiện hiệu suất của mô hình dự đoán lỗi phần mềm, đặc biệt trong bối cảnh dữ liệu mất cân bằng và cần khai thác các đặc trưng ngữ nghĩa phong phú. Các đóng góp chính bao gồm:

- Đánh giá hiệu quả các phương pháp lựa chọn đặc trưng wrapper: Các phương pháp này, đặc biệt là Recursive Feature Elimination và Equilibrium Optimizer đã chứng minh khả năng giảm dư thừa dữ liệu và nâng cao hiệu suất dự đoán lỗi so với mô hình cơ sở.
- Kết hợp kỹ thuật xử lý đặc trưng và lấy mẫu dữ liệu tiên tiến: Nghiên cứu đã chứng minh rằng việc phối hợp giữa kỹ thuật lựa chọn đặc trưng RFE/ trích xuất đặc trưng Autoencoders và các biến thể của GAN (như CTGAN, WGANGP, VanillaGAN) giúp cải thiện đáng kể hiệu suất của các mô hình học máy truyền thống khi dữ liệu bị mất cân bằng. CTGAN được xác định là biến thể hiệu quả nhất trong việc tạo mẫu mới để cân bằng dữ liệu.
- Đề xuất mô hình FastText-Transformer mới: Mô hình này, dựa trên kiến trúc Transformer và biểu diễn từ FastText, đã thể hiện khả năng học đặc trưng ngữ nghĩa sâu sắc từ cây cú pháp trừu tượng của mã nguồn, từ đó cải thiện đáng kể các độ đo đánh giá các mô hình dự đoán lỗi (Precision, Recall, F1-score) cho cả WPDP và CPDP. Mô hình này cũng cho thấy hiệu quả vượt trội về thời gian huấn luyện so với FastText-LSTM.
- Đánh giá thực nghiệm và thống kê mạnh mẽ: Các thử nghiệm được thực hiện trên các bộ dữ liệu mã nguồn mở phổ biến và đánh giá bằng các độ đo định lượng cùng kiểm định thống kê Wilcoxon, cho thấy các kết quả thu được đều mang tính khả thi cao và có ý nghĩa thống kê rõ rệt.

Tóm lại, đề tài đã đóng góp các cách tiếp cận mới và hiệu quả trong lĩnh vực dự đoán lỗi phần mềm, góp phần vào mục tiêu nâng cao độ tin cậy và chất lượng của các hệ thống phần mềm hiện đại.

## Hướng nghiên cứu tương lai

- Mở rộng ứng dụng sang nhiều lĩnh vực và ngôn ngữ lập trình khác nhau (Python, JavaScript, C++) để tăng tính tổng quát của mô hình.
- Cải tiến các chiến lược lấy mẫu (oversampling, undersampling) và điều chỉnh tham số để mô hình hoạt động hiệu quả hơn trong các môi trường dữ liệu phức tạp.

- Mở rộng phương pháp để dự đoán lỗi ở các cấp độ chi tiết hơn như lớp, phương thức hoặc câu lệnh.

## BẢN KÊ KHAI CÁC CÔNG TRÌNH KHOA HỌC ĐÃ CÔNG BỐ

| TT | Thể loại                                               | Năm công bố | Nơi công bố                                                                                                                                     |
|----|--------------------------------------------------------|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| 1  | Tạp chí quốc tế trong danh mục Scopus                  | 2026        | Neural Computing and Applications, (Scopus, Q1).                                                                                                |
| 2  | Tạp chí quốc tế trong danh mục SCIE                    | 2025        | Applied Intelligence. (SCIE, Q2, IF 3.5)                                                                                                        |
| 3  | Tạp chí quốc tế                                        | 2025        | International Journal of Electrical and Computer Engineering.                                                                                   |
| 4  | Tạp chí trong nước                                     | 2025        | Tạp chí Khoa học và Công nghệ Đại học Đà Nẵng.                                                                                                  |
| 5  | Tạp chí trong nước                                     | 2024        | Tạp chí Khoa học và Công nghệ Đại học Đà Nẵng.                                                                                                  |
| 6  | Kỷ yếu Hội thảo gia                                    | 2023        | Hội nghị Khoa học công nghệ Quốc gia về Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR).                                               |
| 7  | Kỷ yếu Hội thảo quốc tế có trong danh mục Scopus, Sách | 2023        | CITA 2023: Conference on Information Technology and Its Applications. Số: <i>Part of the Lecture Notes in Networks and Systems book series.</i> |
| 8  | Kỷ yếu Hội thảo quốc tế                                | 2022        | ICIT 2022: Intelligence of Things: Technologies and Applications.                                                                               |
| 9  | Tạp chí trong nước                                     | 2021        | Chuyên san Các công trình nghiên cứu, phát triển và ứng dụng Công nghệ Thông tin và Truyền thông.                                               |