TẠP CHÍ KHOA HỌC VÀ CÔNG NGHỆ ĐẠI HỌC DUY TÂN
DTU Journal of Science and Technology 01(74) (2026) 79-92
ĐẠI HỌC DUY TÂN
DUYTAN UNIVERSITY

Đánh giá tác động của việc điều chỉnh siêu tham số đối với hiệu năng của các mô hình học máy truyền thống

An Evaluation of the Impact of Hyperparameter Adjustment on the Performance of Traditional Machine Learning Models

Trịnh Quang Tin[a], Phan Long[b], Phạm Phú Khương[b], Ngô Văn Hiếu[b], Nguyễn Tấn Quốc[b], Hồ Lê Viết Nin[b*]
Trinh Quang Tin[a], Phan Long[b], Pham Phu Khuong[b], Ngo Van Hieu[b], Nguyen Tan Quoc[b], Ho Le Viet Nin[b*]

[a]Khoa Khoa học Máy tính, Trường Khoa học Máy tính và Trí tuệ Nhân tạo, Đại học Duy Tân, Đà Nẵng, Việt Nam
[a]Faculty of Computer Science, School of Computer Science and Artificial Intelligence, Duy Tan University, Da Nang, 550000, Viet Nam
[b]Khoa Công nghệ Thông tin, Trường Khoa học Máy tính và Trí tuệ Nhân tạo, Đại học Duy Tân, Đà Nẵng, Việt Nam
[b]Faculty of Information Technology, School of Computer Science and Artificial Intelligence, Duy Tan University, Da Nang, 550000, Viet Nam

(Ngày nhận bài: 16/06/2025, ngày phản biện xong: 19/11/2025, ngày chấp nhận đăng: 20/01/2026)

Tóm tắt

Trong học máy, việc lựa chọn mô hình phân loại cùng chiến lược điều chỉnh siêu tham số phù hợp đóng vai trò quan trọng trong việc nâng cao hiệu năng dự đoán. Bài báo này đề xuất một quy trình thực nghiệm nhằm đánh giá hiệu năng của ba mô hình học máy truyền thống gồm SVM, Random Forest và XGBoost, kết hợp với ba chiến lược điều chỉnh siêu tham số là Grid Search, Random Search và Bayesian optimization. Dữ liệu được biểu diễn thông qua ba loại đặc trưng: TF-IDF, AST và sự kết hợp của cả hai. Mỗi cấu hình mô hình được huấn luyện lặp lại 50 lần nhằm đảm bảo độ tin cậy thống kê. Hiệu năng được đánh giá dựa trên hai chỉ số chính là F1-score và ROC AUC. Kết quả thực nghiệm cho thấy mô hình XGBoost với đặc trưng kết hợp và điều chỉnh bằng Bayesian optimization đạt hiệu năng cao nhất, với F1-score đạt 92.0% và ROC AUC đạt 94.7%, tăng lần lượt 2.1% và 1.3% so với thiết lập mặc định. Phân tích chi tiết cho thấy mối liên hệ chặt chẽ giữa cách biểu diễn đặc trưng, thuật toán phân loại và chiến lược điều chỉnh, từ đó đưa ra các khuyến nghị thực tiễn cho việc lựa chọn mô hình trong các bài toán phân loại.

Từ khóa: AST, Bayesian optimization, điều chỉnh siêu tham số, học máy, XGBoost

Abstract

In machine learning, selecting an appropriate classification model along with a suitable hyperparameter adjustment strategy plays a crucial role in enhancing predictive performance. This paper presents an experimental framework to evaluate the performance of three traditional machine learning models: SVM, Random Forest and XGBoost, combined with three hyperparameter adjustment strategies: Grid Search, Random Search, and Bayesian optimization. The input data is represented using three types of features: TF-IDF, AST and their combination. Each model configuration is trained 50 times to ensure statistical reliability. Model performance is evaluated based on two key metrics: F1-score and ROC AUC. Experiental results show that the XGBoost model, when using combined features and optimized via Bayesian

[*]*Tác giả liên hệ:* Hồ Lê Viết Nin
Email: holvietnin@dtu.edu.vn

optimization, achieves the highest performance with an F1-score of 92.0% and a ROC AUC of 94.7%, representing respective improvements of 2.1% and 1.3% over the default settings. A detailed analysis reveals a strong relationship between feature representation, classification algorithm, and adjustment strategy, providing practical insights for selecting machine learning models in classification tasks in practical applications.

Keywords: AST, Bayesian optimization, hyperparameter adjustment, traditional machine learning, XGBoost

1. Giới thiệu

Trong học máy, phân loại là một trong những nhiệm vụ nền tảng và được áp dụng rộng rãi trong nhiều lĩnh vực, với ứng dụng trải rộng từ phân tích dữ liệu y tế, tài chính cho đến xử lý ngôn ngữ tự nhiên và hệ thống đề xuất. Trong nhiều bài toán thực tế, các mô hình học máy truyền thống như Support Vector Machine (SVM) [1], Random Forest (RF) [2] và Extreme Gradient Boosting (XGBoost) [3] vẫn được ưa chuộng nhờ hiệu quả, khả năng diễn giải cao và thời gian huấn luyện hợp lý.

Tuy nhiên, hiệu năng của một mô hình học máy không chỉ phụ thuộc vào bản thân thuật toán mà còn bị chi phối mạnh bởi hai yếu tố then chốt: cách biểu diễn dữ liệu đầu vào và phương pháp điều chỉnh siêu tham số. Một mô hình tốt nhưng sử dụng đặc trưng không phù hợp hoặc thiết lập siêu tham số chưa tối ưu có thể dẫn đến kết quả kém ổn định hoặc hiệu năng thấp. Do đó, việc lựa chọn hợp lý giữa đặc trưng đầu vào và chiến lược điều chỉnh siêu tham số là điều kiện tiên quyết để khai thác tối đa năng lực của mô hình.

Trong bối cảnh đó, các kỹ thuật điều chỉnh siêu tham số như Grid Search [4], Random Search [5], và đặc biệt là Bayesian optimization [6] đã và đang được nghiên cứu, ứng dụng rộng rãi. Mỗi phương pháp mang lại những ưu, nhược điểm riêng về độ chính xác, thời gian xử lý và khả năng khám phá không gian tham số. Tuy nhiên, hiện vẫn còn thiếu các nghiên cứu thực nghiệm đối sánh có hệ thống giữa các chiến lược này trên nhiều mô hình phân lớp, đặc biệt khi kết hợp với các loại đặc trưng đầu vào khác nhau.

Bài báo này trình bày một nghiên cứu thực nghiệm có cấu trúc chặt chẽ nhằm phân tích hiệu năng của ba mô hình học máy truyền thống (SVM, RF, XGBoost) khi được kết hợp với ba chiến lược điều chỉnh siêu tham số phổ biến (Grid Search, Random Search, Bayesian optimization). Đặc trưng đầu vào được xây dựng từ ba phương pháp: TF-IDF, AST và sự kết hợp giữa hai kỹ thuật này. Mỗi mô hình được huấn luyện nhiều lần để đảm bảo tính ổn định của kết quả, tập trung đánh giá qua hai chỉ số chính: F1-score và ROC AUC.

Nghiên cứu hướng tới mục tiêu cung cấp một cái nhìn khách quan và toàn diện về mối quan hệ giữa mô hình học máy, đặc trưng đầu vào và chiến lược điều chỉnh siêu tham số, từ đó hỗ trợ lựa chọn cấu hình phù hợp hơn trong các bài toán phân loại thực tế.

Các đóng góp chính của bài báo gồm: (i) xây dựng một khung thực nghiệm toàn diện kết hợp ba mô hình, ba dạng đặc trưng và ba chiến lược điều chỉnh siêu tham số trong cùng một ma trận đánh giá; (ii) thực hiện lặp lại toàn bộ thực nghiệm nhiều lần để đảm bảo độ tin cậy thống kê và phân tích được mức độ ổn định của từng cấu hình; (iii) chỉ ra mối quan hệ giữa đặc trưng, mô hình, chiến lược điều chỉnh, đồng thời giải thích nguyên nhân của một số kết hợp, đặc biệt là XGBoost với đặc trưng kết hợp và tối ưu Bayesian, đạt hiệu năng vượt trội so với các phương án còn lại, qua đó đưa ra các khuyến nghị thực tiễn cho các bài toán phân loại mã nguồn.

2. Cơ sở lý thuyết và thuật toán sử dụng

Trong khuôn khổ nghiên cứu này, việc lựa chọn mô hình phân lớp và phương pháp điều chỉnh siêu tham số được xem là hai yếu tố then chốt ảnh hưởng đến hiệu năng của hệ thống học máy. Ba thuật toán phân lớp tiêu biểu được lựa chọn gồm: SVM, RF và XGBoost. Mỗi thuật

toán đại diện cho một hướng tiếp cận khác nhau trong xây dựng mô hình: từ phân tách tuyến tính (SVM), kết hợp ngẫu nhiên nhiều cây quyết định (RF), đến tăng cường dần theo độ dốc (XGBoost).

Song song với đó, ba chiến lược điều chỉnh siêu tham số gồm Grid Search, Random Search và Bayesian optimization cũng được đưa vào phân tích. Việc hiểu rõ nguyên lý hoạt động, vai trò và đặc trưng của từng mô hình và phương pháp điều chỉnh sẽ là cơ sở để thiết kế và triển khai các thực nghiệm ở phần sau.

2.1. Support Vector Machine

SVM là một mô hình học có giám sát được thiết kế để giải quyết bài toán phân loại nhị phân bằng cách xác định một siêu phẳng tối ưu nhằm phân tách các điểm dữ liệu thuộc hai lớp khác nhau. Trong trường hợp dữ liệu tuyến tính, mục tiêu của SVM là tìm một siêu phẳng sao cho khoảng cách (margin) giữa hai lớp là lớn nhất. Bài toán tối ưu tương ứng có thể được phát biểu dưới dạng:

$$\min_{\mathbf{w},b} \frac{1}{2} \|\mathbf{w}\|^2 \text{ với điều kiện: } y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \forall i \quad (1)$$

trong đó: $\mathbf{w}$ là vector trọng số xác định hướng của siêu phẳng phân tách, b là hệ số dịch điều chỉnh vị trí của siêu phẳng so với gốc tọa độ, $\mathbf{x}_i$ là vector đặc trưng thứ i, y_i là nhãn của mẫu i thuộc tập $\{-1,+1\}$, $\mathbf{w}^\top \mathbf{x}_i + b$ là giá trị dự đoán tuyến tính trước dấu của mẫu i, $y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1$ là ràng buộc đảm bảo rằng điểm dữ liệu phân loại đúng và nằm ngoài hoặc trên biên phân tách, $\frac{1}{2}\|\mathbf{w}\|^2$ là hàm mục tiêu cần tối thiểu hóa bình phương chuẩn Euclid của vector trọng số $\mathbf{w}$ giúp tối đa hóa margin (khoảng cách giữa hai lớp).

Trong thực tế, dữ liệu huấn luyện thường không hoàn toàn phân tách tuyến tính do nhiễu hoặc phân bố chồng lấn giữa các lớp. Khi đó, SVM được mở rộng bằng kỹ thuật soft-margin, cho phép một số điểm dữ liệu vi phạm điều kiện phân tách. Bài toán tối ưu trở thành:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i \text{ với điều kiện: } y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i,\ \xi_i \geq 0 \quad (2)$$

trong đó: $\mathbf{w}, b$, $\mathbf{x}_i$, y_i giống công thức (1), ξ_i là biến trượt cho phép điểm $\mathbf{x}_i$ vi phạm nhẹ điều kiện phân tách, C là siêu tham số điều chỉnh mức phạt cho các điểm vi phạm, $C\sum_{i=1}^{n} \xi_i$ là thành phần phạt, càng lớn nếu có nhiều điểm bị phân loại sai.

Công thức (2) là tổng quát hóa của (1), giúp mô hình vẫn hoạt động tốt trong trường hợp dữ liệu nhiễu hoặc không tuyến tính hoàn toàn.

Để xử lý các bài toán phi tuyến, SVM sử dụng các hàm kernel nhằm ánh xạ dữ liệu từ không gian ban đầu sang không gian đặc trưng có số chiều cao hơn, nơi có thể tồn tại một siêu phẳng tuyến tính để phân tách dữ liệu. Các hàm kernel phổ biến bao gồm: kernel tuyến tính, hàm Gaussian (RBF), và đa thức (polynomial).

Trong quá trình huấn luyện, một số siêu tham số cần được thiết lập trước gồm: C - điều chỉnh mức độ cho phép vi phạm margin và ảnh hưởng đến mức độ phạt cho các điểm sai lệch; tham số kernel như không có (với kernel tuyến tính), γ trong RBF (điều khiển độ rộng ảnh hưởng của điểm dữ liệu) hoặc bậc đa thức trong kernel polynomial.

2.2. Random Forest

RF là một thuật toán học máy thuộc nhóm mô hình tổ hợp (ensemble learning), được phát triển dựa trên nguyên lý kết hợp nhiều cây quyết định

nhằm cải thiện độ chính xác và tính ổn định trong dự đoán. Phương pháp này sử dụng kỹ thuật bagging (bootstrap aggregating), trong đó nhiều cây được huấn luyện trên các tập con dữ liệu được lấy mẫu ngẫu nhiên có lặp lại từ tập huấn luyện gốc, sau đó kết quả dự đoán được tổng hợp từ các cây thành phần.

Khác với các mô hình cây quyết định truyền thống, RF tại mỗi nút phân chia không xét toàn bộ tập đặc trưng, mà chỉ chọn ngẫu nhiên một tập con nhỏ để quyết định điều kiện phân tách. Cách tiếp cận này giúp tăng cường tính ngẫu nhiên giữa các cây, giảm hiện tượng quá khớp (overfitting) và hạn chế việc mô hình bị chi phối bởi các đặc trưng mạnh.

Trong giai đoạn dự đoán, đầu ra của mô hình được xác định như sau. Với bài toán phân loại, RF thực hiện bỏ phiếu đa số (majority voting) để chọn ra nhãn dự đoán:

$$\hat{y} = \text{mode}\left\{h_t(x)\right\}_{t=1}^{T} \quad (3)$$

trong đó: $\hat{y}$ là nhãn phân loại đầu ra của mô hình, x là điểm dữ liệu cần phân loại, T là tổng số cây trong mô hình, $h_t(x)$ dự đoán của cây quyết định thứ t với đầu vào x, $\text{mode}\left\{h_t(x)\right\}$ là giá trị xuất hiện nhiều nhất trong các dự đoán của các cây.

Còn với bài toán hồi quy, mô hình trả về giá trị đầu ra bằng trung bình các dự đoán từ toàn bộ cây thành phần:

$$\hat{y} = \frac{1}{T}\int\sum_{t-1}^{T} h_i(x) \quad (4)$$

trong đó: $\hat{y}$ ở công thức (4) là giá trị dự đoán liên tục của mô hình, x, T, $h_i(x)$ giống như trong công thức (3), $\frac{1}{T}\int\sum_{t-1}^{T} h_i(x)$ là trung bình các giá trị dự đoán của tất cả cây trong rừng.

Nhờ khả năng kết hợp nhiều mô hình học yếu, RF có thể tổng quát hóa tốt và thường cho hiệu năng ổn định ngay cả khi dữ liệu huấn luyện có nhiễu hoặc không cân bằng. Một số siêu tham số quan trọng trong mô hình RF bao gồm:

- Số lượng cây (n_estimators): ảnh hưởng đến độ chính xác tổng thể và chi phí tính toán;
- Số đặc trưng được chọn ngẫu nhiên tại mỗi nút (max_features): điều chỉnh mức độ ngẫu nhiên và khả năng phân chia tại mỗi cây;
- Độ sâu tối đa của cây (max_depth): kiểm soát độ phức tạp của từng cây và giúp ngăn chặn hiện tượng quá khớp (overfitting).

Với đặc tính ngẫu nhiên hóa và cơ chế tổng hợp mạnh mẽ, RF thường được sử dụng như một mô hình khởi đầu mặc định trong nhiều bài toán phân loại và hồi quy, đặc biệt khi chưa có giả định rõ ràng về phân bố dữ liệu.

2.3. Extreme Gradient Boosting

XGBoost là một thuật toán tăng cường (boosting) hiện đại, được thiết kế để cải thiện hiệu năng dự đoán thông qua việc kết hợp nhiều mô hình cây quyết định theo hướng tuần tự. Không giống như RF xây dựng các cây độc lập, XGBoost huấn luyện từng cây mới dựa trên sai số còn lại (residuals) của tổ hợp các cây trước đó, từ đó từng bước cải thiện kết quả.

Hàm mục tiêu trong XGBoost bao gồm hai thành phần: hàm mất mát đo sai số giữa dự đoán và nhãn thực tế, và một hàm điều chuẩn để kiểm soát độ phức tạp của mô hình:

$$\mathcal{L}(\phi) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \quad (5)$$

trong đó, $\mathcal{L}(\phi)$ là tổng mục tiêu cần tối ưu, $l(y_i, \hat{y}_i)$ là hàm mất mát giữa nhãn thực tế y_i và $\hat{y}_i$ (thường là log loss hoặc squared error), f_k là cây quyết định tại vòng lặp thứ k và $\Omega(f_k)$ là hàm điều chuẩn nhằm phạt các mô hình quá phức tạp, n là số lượng mẫu trong tập huấn luyện và K là tổng số cây trong mô hình.

Quá trình huấn luyện trong XGBoost sử dụng thuật toán greedy để tìm điểm chia (split) tối ưu tại mỗi node, dựa trên tiêu chí tối đa hóa mức giảm của hàm mục tiêu tại bước đó. Ngoài ra, XGBoost còn tích hợp nhiều kỹ thuật nâng cao như:

- Shrinkage (learning rate): giúp kiểm soát tốc độ cập nhật;
- Subsampling: lấy ngẫu nhiên một phần dữ liệu để giảm overfitting;
- Regularization (L1, L2): giúp ổn định mô hình và tăng khả năng tổng quát.
- Các siêu tham số chính trong XGBoost bao gồm:
- Số lượng cây (n_estimators);
- Tốc độ học (learning_rate);
- Độ sâu tối đa (max_depth);
- Tỷ lệ mẫu (subsample, colsample_bytree);
- Hệ số điều chuẩn (lambda, alpha).

Với cơ chế tăng cường dần dần, khả năng kiểm soát overfitting qua regularization và tốc độ hội tụ nhanh, XGBoost được sử dụng rộng rãi trong các bài toán học máy trên dữ liệu dạng bảng (tabular data), đặc biệt trong các cuộc thi lớn và ứng dụng thực tế.

2.4. Chiến lược điều chỉnh siêu tham số

Hiệu quả của các mô hình học máy không chỉ phụ thuộc vào cấu trúc thuật toán mà còn chịu ảnh hưởng lớn từ cách thiết lập các siêu tham số (hyperparameters). Các siêu tham số như độ sâu cây, tốc độ học, hoặc hệ số phạt không thể học trực tiếp từ dữ liệu, do đó cần được điều chỉnh thông qua quá trình tìm kiếm có kiểm soát. Ba chiến lược điều chỉnh phổ biến hiện nay gồm: Grid Search, Random Search và Bayesian optimization.

Grid Search: Đây là phương pháp đơn giản nhất, trong đó thuật toán duyệt qua tất cả các tổ hợp tham số trong một lưới được xác định trước. Mặc dù đảm bảo bao phủ toàn bộ không gian tìm kiếm, Grid Search tiêu tốn nhiều thời gian khi số lượng siêu tham số lớn hoặc mỗi tham số có nhiều giá trị. Độ phức tạp tăng theo cấp số nhân với số chiều của không gian tìm kiếm.

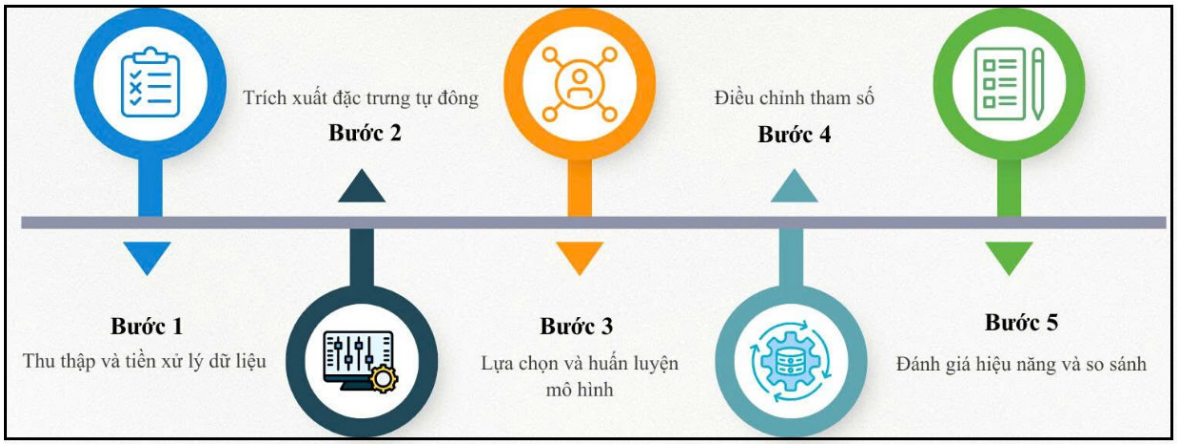
Random Search: Thay vì duyệt toàn bộ tổ hợp, Random Search chọn ngẫu nhiên các điểm trong không gian siêu tham số. Phương pháp này thường đạt kết quả tương đương (hoặc tốt hơn) so với Grid Search trong cùng ngân sách tính toán, đặc biệt khi chỉ một vài siêu tham số thực sự ảnh hưởng mạnh đến hiệu năng mô hình.

Bayesian optimization: Đây là phương pháp tiên tiến hơn, xây dựng một mô hình xác suất (gọi là surrogate model) để dự đoán hiệu năng của các cấu hình siêu tham số. Dựa trên mô hình này, một hàm lựa chọn (acquisition function) sẽ xác định điểm tiếp theo cần đánh giá sao cho cân bằng giữa khai phá (exploration) và khai thác (exploitation). Nhờ đó, Bayesian optimization thường hội tụ nhanh hơn và tiêu tốn ít lần đánh giá hơn so với hai phương pháp còn lại.

Mỗi chiến lược có ưu và nhược điểm riêng, và lựa chọn phù hợp thường phụ thuộc vào độ phức tạp của mô hình, kích thước không gian tham số và giới hạn tài nguyên tính toán.

3. Phương pháp đề xuất

Nghiên cứu này đề xuất một quy trình thực nghiệm gồm năm bước chính: (1) Thu thập và tiền xử lý dữ liệu, (2) Trích xuất đặc trưng tự động, (3) Lựa chọn và huấn luyện mô hình, (4) Điều chỉnh siêu tham số, và (5) Đánh giá hiệu năng và so sánh. Sơ đồ tổng quan của toàn bộ quy trình năm bước được trình bày trong Hình 1 để giúp người đọc dễ hình dung hơn.



Hình 1. Quy trình thực nghiệm đánh giá tác động của việc điều chỉnh siêu tham số

Toàn bộ thực nghiệm được áp dụng cho bài toán phân loại mã nguồn có và không có lỗ hổng bảo mật, sử dụng hai đặc trưng chính là Term Frequency-Inverse Document Frequency (TF-IDF) [7] và Abstract Syntax Tree (AST) [8], ba mô hình học máy là SVM, RF, XGBoost, cùng ba chiến lược điều chỉnh là Grid Search, Random Search và Bayesian optimization. Các kết quả thu được sẽ giúp phân tích mối liên hệ giữa loại đặc trưng, thuật toán phân lớp và chiến lược điều chỉnh siêu tham số trong bối cảnh thực tiễn.

3.1. Thu thập và tiền xử lý dữ liệu

Dữ liệu đầu vào được lấy từ hai bộ dữ liệu mã nguồn có gán nhãn phổ biến là Juliet Test Suite v1.3 [9] và CodeXGLUE Defect Detection [10]. Đây là hai nguồn dữ liệu thường được sử dụng trong các nghiên cứu về phát hiện lỗi hoặc lỗ hổng phần mềm. Mỗi đoạn mã trong bộ dữ liệu đều đi kèm nhãn xác định xem đó là đoạn mã an toàn hay dễ gây ra lỗi, từ đó cho phép huấn luyện mô hình phân loại nhị phân.

Trước khi đưa vào mô hình, dữ liệu trải qua các bước xử lý bao gồm chuẩn hóa định dạng, loại bỏ dòng trống, ký tự đặc biệt và mã không hợp lệ. Các đoạn mã được kiểm tra tính đầy đủ của nhãn và loại bỏ trường hợp trùng lặp. Tập dữ liệu sau xử lý được chia theo tỷ lệ 80% để huấn luyện và 20% để kiểm tra, đảm bảo không rò rỉ dữ liệu và phản ánh chính xác khả năng tổng quát hóa của mô hình.

3.2. Trích xuất đặc trưng tự động

Đặc trưng đầu vào đóng vai trò quyết định đến hiệu năng của các mô hình học máy. Trong nghiên cứu này, hai phương pháp trích xuất đặc trưng phổ biến được sử dụng là TF-IDF và AST. Mỗi phương pháp cung cấp một góc nhìn khác nhau về thông tin bên trong mã nguồn: TF-IDF phản ánh tầm quan trọng của từ khóa, trong khi AST nắm bắt cấu trúc cú pháp của chương trình.

Với TF-IDF, các đoạn mã được xử lý như văn bản tự nhiên, trải qua bước tách token, chuyển về chữ thường, loại bỏ ký tự không cần thiết và biểu diễn thành vector dựa trên tần suất và mức độ đặc trưng của từng từ. Phương pháp này giúp mô hình học được các mẫu biểu hiện nguy cơ từ tần suất xuất hiện của từ khóa trong mã.

Trong khi đó, AST được xây dựng bằng cách phân tích cú pháp của đoạn mã để tạo thành cây biểu diễn cấu trúc chương trình. Từ cây này, các đặc trưng có thể được trích xuất thông qua duyệt cây, đếm node hoặc ánh xạ thành biểu diễn vector hóa. Việc khai thác AST giúp mô hình tiếp cận các thông tin liên quan đến ngữ cảnh và logic lập trình, vượt ra khỏi giới hạn của phân tích văn bản thuần túy.

Cuối cùng, nghiên cứu cũng kết hợp cả hai phương pháp thông qua việc nối các vector TF-IDF và AST thành một đặc trưng thống nhất, nhằm tận dụng cả thông tin ngôn ngữ và cấu trúc để nâng cao khả năng phân loại.

3.3. Lựa chọn và huấn luyện mô hình

Ba mô hình học máy truyền thống được lựa chọn để thực hiện nhiệm vụ phân loại là SVM, RF và XGBoost. Việc lựa chọn này nhằm đảm bảo sự đa dạng trong nguyên lý hoạt động: SVM dựa trên tìm siêu mặt phẳng tối ưu, RF khai thác tổ hợp cây ngẫu nhiên, còn XGBoost là mô hình tăng cường dựa trên gradient có hiệu quả huấn luyện cao

Mỗi mô hình được huấn luyện độc lập trên từng loại đặc trưng (TF-IDF, AST, kết hợp cả hai), giúp đánh giá ảnh hưởng của cách biểu diễn dữ liệu đến hiệu năng phân loại. Trong giai đoạn đầu, các siêu tham số của mô hình được thiết lập ở mức mặc định để làm cơ sở so sánh về sau.

Để đảm bảo độ tin cậy thống kê và giảm thiểu ảnh hưởng của yếu tố ngẫu nhiên trong quá trình chia dữ liệu và huấn luyện, tất cả các thực nghiệm trong bài đều được lặp lại 50 lần. Số lần lặp được lựa chọn là 50 vì đây là mức đủ lớn để làm mượt các dao động ngẫu nhiên sinh ra từ ba mô hình học máy SVM, RF, XGBoost và ba chiến lược điều chỉnh siêu tham số Grid Search, Random Search, Bayesian optimization, vốn đều có các bước khởi tạo hoặc lấy mẫu ngẫu nhiên. Thử nghiệm ban đầu cho thấy từ mức 50 trở lên, giá trị trung bình và độ lệch chuẩn gần như ổn định, trong khi tăng lên 60 hoặc 70 lần không mang lại sự khác biệt đáng kể nhưng làm kéo dài thời gian chạy không cần thiết. Do chi phí huấn luyện của các mô hình học máy truyền thống là thấp, mức 50 lần lặp được xem là cân bằng hợp lý giữa độ tin cậy thống kê và hiệu quả thực nghiệm.

Sau mỗi lần huấn luyện, mô hình được đánh giá trên tập kiểm thử độc lập, với các chỉ số F1-score và ROC AUC được ghi nhận để phục vụ phân tích thống kê. Kết quả được trình bày dưới dạng giá trị trung bình và độ lệch chuẩn (mean ± std) nhằm phản ánh hiệu năng và mức độ ổn định của mô hình.

3.4. Điều chỉnh siêu tham số

Sau khi huấn luyện mô hình với các thiết lập mặc định, bước tiếp theo là điều chỉnh siêu tham số nhằm nâng cao hiệu năng phân loại. Mỗi mô hình học máy đều có một tập các siêu tham số điều chỉnh khả năng học, độ phức tạp và tốc độ hội tụ. Chẳng hạn, với SVM các siêu tham số như hệ số phạt C và tham số kernel và các tham số liên quan như γ; với RF là số cây, số đặc trưng được chọn tại mỗi nút và độ sâu cây; còn XGBoost bao gồm tốc độ học, độ sâu cây và hệ số điều chuẩn.

Ba chiến lược được sử dụng để điều chỉnh siêu tham số là Grid Search, Random Search và Bayesian optimization. Grid Search thực hiện tìm kiếm toàn bộ các tổ hợp trong một lưới định nghĩa trước, đảm bảo bao phủ nhưng tốn nhiều tài nguyên. Random Search chọn ngẫu nhiên các cấu hình, giúp tiết kiệm thời gian trong không gian lớn. Trong khi đó, Bayesian optimization sử dụng mô hình xác suất để dự đoán hiệu năng và lựa chọn các điểm điều chỉnh tiếp theo một cách hiệu quả hơn.

Mỗi mô hình đều được điều chỉnh riêng biệt trên ba loại đặc trưng và dưới cả ba chiến lược, với cùng một số lần đánh giá để đảm bảo so sánh công bằng. Kết quả điều chỉnh được lưu lại và sử dụng trong bước đánh giá cuối cùng để phân tích hiệu quả từng cách kết hợp.

3.5. Đánh giá hiệu năng và so sánh

Sau khi hoàn tất huấn luyện và điều chỉnh, các mô hình được đánh giá trên tập kiểm thử thông qua hai chỉ số chính: F1-score và ROC AUC. F1-score giúp đo lường sự cân bằng giữa Precision và Recall, phù hợp với bài toán phân loại mất cân bằng. ROC AUC phản ánh khả năng phân biệt giữa hai lớp trên toàn bộ ngưỡng, cho thấy hiệu quả tổng thể của mô hình.

Để đảm bảo kết quả ổn định và đáng tin cậy, mỗi cấu hình mô hình bao gồm lựa chọn đặc trưng, thuật toán phân lớp và chiến lược điều

chỉnh được thực nghiệm 50 lần. Kết quả sau mỗi lần chạy được ghi nhận và tổng hợp thành giá trị trung bình cùng độ lệch chuẩn, từ đó đánh giá mức độ biến thiên và tính nhất quán của mô hình.

Các kết quả thu được được trình bày dưới dạng bảng số liệu và biểu đồ so sánh, cho phép quan sát trực quan ảnh hưởng của từng yếu tố: loại đặc trưng (TF-IDF, AST, kết hợp cả hai), mô hình học máy (SVM, RF, XGBoost), và chiến lược điều chỉnh (Grid, Random, Bayesian). Phân tích chi tiết giúp làm rõ mối quan hệ giữa cấu hình mô hình và hiệu năng đạt được, từ đó rút ra các khuyến nghị phù hợp cho các ứng dụng thực tế.

4. Thực nghiệm và kết quả

Sau khi hoàn thiện quy trình huấn luyện và điều chỉnh siêu tham số, các thực nghiệm được tiến hành trên hai bộ dữ liệu Juliet Test Suite v1.3 và CodeXGLUE Defect Detection để kiểm tra hiệu quả phân loại giữa mã nguồn an toàn và mã có lỗ hổng. Quá trình thực nghiệm được thiết kế nhằm phân tích ảnh hưởng của từng yếu tố: đặc trưng đầu vào (TF-IDF, AST, kết hợp cả hai) mô hình phân lớp (SVM, RF, XGBoost) và kỹ thuật điều chỉnh siêu tham số (Grid Search, Random Search, Bayesian optimization).

Mỗi cấu hình mô hình được chạy lặp lại nhiều lần để thu thập giá trị trung bình và độ lệch chuẩn của các chỉ số đánh giá, từ đó đảm bảo tính ổn định và độ tin cậy của kết quả. Trong bối cảnh bài toán phân loại nhị phân có thể mất cân bằng dữ liệu, hai chỉ số F1-score và ROC AUC được lựa chọn để so sánh hiệu năng, vì chúng phản ánh hiệu quả tổng thể và khả năng phân biệt giữa hai lớp tốt hơn so với Accuracy. Các kết quả thực nghiệm sẽ được trình bày và phân tích chi tiết trong các mục tiếp theo, thông qua bảng số liệu và biểu đồ minh họa.

4.1. Thiết lập thực nghiệm

Các thực nghiệm được triển khai bằng ngôn ngữ Python 3.10 trong môi trường lập trình Jupyter Notebook, kết hợp với các thư viện chuyên dụng như Scikit-learn, XGBoost và Optuna. Toàn bộ quá trình xử lý dữ liệu, huấn luyện mô hình và điều chỉnh siêu tham số được thực hiện trên một máy tính cá nhân với cấu hình cụ thể như sau: CPU Intel Core i7-11800H, RAM 32GB, ổ cứng SSD 1TB, hệ điều hành Windows 11, không sử dụng GPU.

Tập dữ liệu sử dụng gồm hai nguồn chuẩn là Juliet Test Suite v1.3 và CodeXGLUE Defect Detection, với nhãn nhị phân thể hiện trạng thái an toàn hoặc có lỗ hổng của đoạn mã. Dữ liệu được chia thành hai phần: 80% phục vụ huấn luyện và điều chỉnh, 20% dùng để kiểm thử mô hình, bảo đảm không trùng lặp và giữ nguyên tỷ lệ phân bố lớp.

Mỗi mô hình được đánh giá trên ba dạng đặc trưng: TF-IDF, AST và kết hợp cả hai. Quá trình huấn luyện và đánh giá được lặp lại nhiều lần với các cấu hình tham số khác nhau để giảm thiểu ảnh hưởng của ngẫu nhiên và thu thập kết quả có độ tin cậy cao. Các chiến lược điều chỉnh siêu tham số gồm Grid Search, Random Search và Bayesian optimization đều được áp dụng riêng cho từng mô hình, đặc trưng, nhằm đảm bảo tính công bằng và khả năng so sánh toàn diện giữa các phương án.

4.2. Kết quả nền với thiết lập siêu tham số mặc định

Trong giai đoạn đầu, các mô hình học máy được huấn luyện với các siêu tham số mặc định nhằm thiết lập một mốc hiệu năng nền (baseline). Ba thuật toán phân lớp truyền thống được sử dụng là SVM, RF và XGBoost, lần lượt được áp dụng trên ba loại đặc trưng đầu vào: TF-IDF, AST và kết hợp của cả hai.

Các chỉ số F1-score và ROC AUC được tính theo trung bình và độ lệch chuẩn giúp phản ánh cả hiệu năng và độ ổn định trong phân loại. Kết quả được trình bày trong Bảng 1 và Bảng 2 dưới đây.

Bảng 1. Kết quả đánh giá F1-score (%) các mô hình với từng loại đặc trưng (thiết lập mặc định)

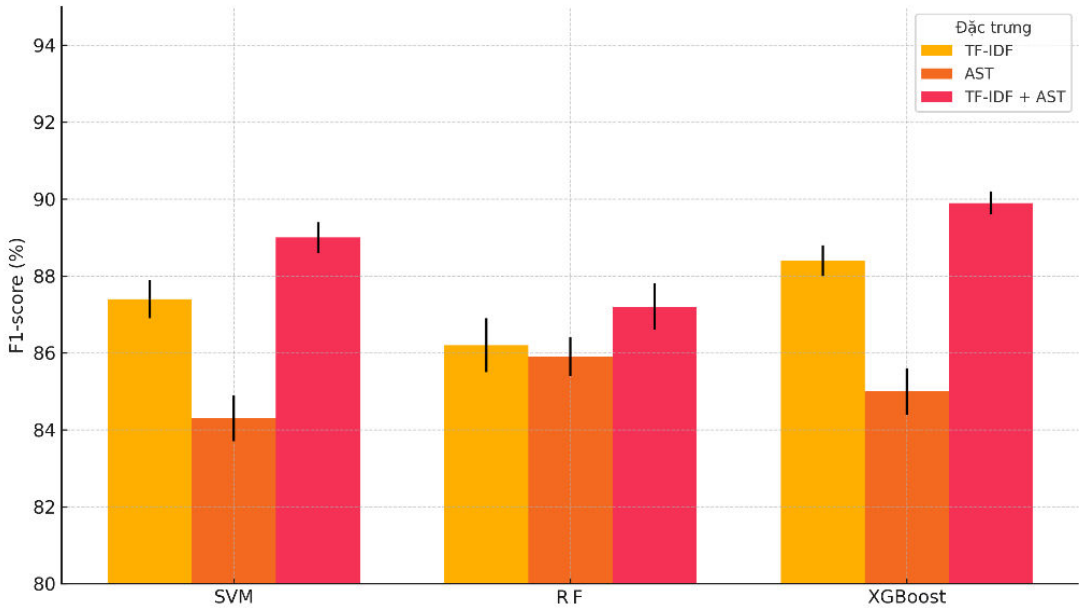
Mô hình	TF-IDF	AST	TF-IDF + AST
SVM	87.4 ± 0.5	84.3 ± 0.6	89.0 ± 0.4
RF	86.2 ± 0.7	85.9 ± 0.5	87.2 ± 0.6
XGBoost	88.4 ± 0.4	85.0 ± 0.6	89.9 ± 0.3

Bảng 2. Kết quả đánh giá ROC AUC (%) các mô hình với từng loại đặc trưng (thiết lập mặc định)

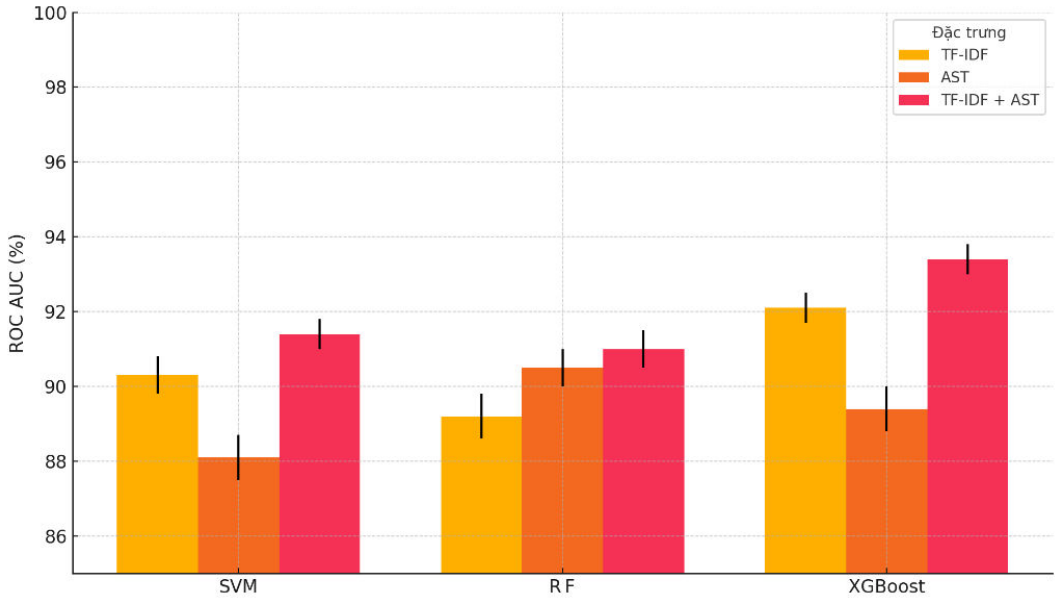
Mô hình	TF-IDF	AST	TF-IDF + AST)
SVM	90.3 ± 0.5	88.1 ± 0.6	91.4 ± 0.4
RF	89.2 ± 0.6	90.5 ± 0.5	91.0 ± 0.5
XGBoost	92.1 ± 0.4	89.4 ± 0.6	93.4 ± 0.4

Các kết quả cho thấy rằng việc kết hợp đặc trưng TF-IDF và AST giúp cải thiện hiệu năng rõ rệt ở cả ba mô hình. Trong đó, mô hình XGBoost luôn cho kết quả vượt trội nhất, với F1-score đạt 89.9 ± 0.3% và ROC AUC đạt 93.4 ± 0.4% khi sử dụng đặc trưng kết hợp. Mô hình SVM cũng cho thấy sự cải thiện ổn định khi chuyển từ AST đơn lẻ sang TF-IDF và đặc biệt khi kết hợp cả hai đặc trưng. Mô hình RF có độ lệch chuẩn thấp và hiệu năng trung bình, phù hợp với các bài toán yêu cầu ổn định nhưng không quá phức tạp.

Để hỗ trợ trực quan, Hình 2 và Hình 3 bên dưới trình bày biểu đồ cột tích hợp thanh sai số (error bar) biểu diễn độ lệch chuẩn (±) của các chỉ số đánh giá. Các biểu đồ này giúp làm rõ sự khác biệt về hiệu năng trung bình cũng như mức độ ổn định giữa các mô hình học máy khi áp dụng trên từng loại đặc trưng đầu vào. Các đoạn thẳng đứng mảnh phía trên mỗi cột chính là các error bar, thể hiện mức dao động kết quả qua các lần chạy lặp lại, từ đó phản ánh trực quan về tính nhất quán và độ tin cậy của từng mô hình.



Hình 2. F1-score trung bình ± độ lệch chuẩn theo từng loại đặc trưng



Hình 3. ROC AUC trung bình ± độ lệch chuẩn theo từng loại đặc trưng

Kết quả ở bảng trên là cơ sở để đánh giá mức độ cải thiện khi tiến hành điều chỉnh siêu tham số ở bước tiếp theo. Việc lựa chọn chiến lược điều chỉnh phù hợp sẽ giúp tăng hiệu năng phân loại và làm rõ ưu, nhược điểm của từng mô hình trên các loại đặc trưng khác nhau.

4.3. Kết quả sau khi điều chỉnh siêu tham số

Sau khi áp dụng ba chiến lược điều chỉnh siêu tham số gồm Grid Search, Random Search và Bayesian optimization, hiệu năng của các mô hình cải thiện rõ rệt trên cả ba loại đặc trưng. Việc tinh chỉnh tham số đã giúp các thuật toán học tốt hơn cấu trúc và ngữ nghĩa trong dữ liệu, từ đó nâng cao độ chính xác và khả năng phân biệt.

- Với F1-Score

Kết quả thực nghiệm chi tiết được tổng hợp trong Bảng 3 cho thấy XGBoost tiếp tục khẳng định ưu thế vượt trội so với hai mô hình còn lại, đặc biệt khi sử dụng đặc trưng kết hợp TF-IDF + AST. Trong cấu hình tối ưu, XGBoost đạt F1-score trung bình 92.0%, cải thiện tăng khoảng 2.1% so với trước điều chỉnh.

Sở dĩ XGBoost đạt hiệu quả cao nhất là vì mô hình này có khả năng học tốt các quan hệ phi tuyến phức tạp trong dữ liệu và tận dụng hiệu quả cả hai loại đặc trưng TF-IDF và AST. Trong cơ chế tăng cường, mỗi cây mới được xây dựng để sửa lỗi cho cây trước đó nên mô hình được cải thiện dần theo từng vòng học. Bên cạnh đó, XGBoost có các thành phần điều chuẩn giúp giảm quá khớp và hoạt động ổn định trong không gian đặc trưng lớn. Khi kết hợp với các chiến lược tối ưu siêu tham số, đặc biệt là Bayesian optimization, mô hình dễ dàng tìm được cấu hình phù hợp hơn so với SVM và RF.

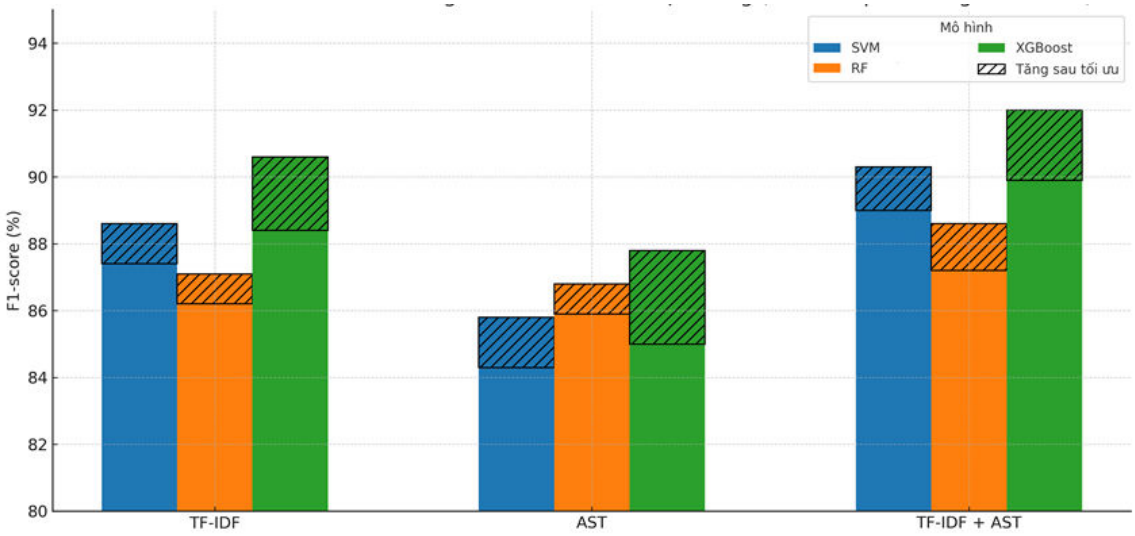
Sự gia tăng này phản ánh khả năng tận dụng hiệu quả cả thông tin ngữ nghĩa (TF-IDF) lẫn cấu trúc cú pháp (AST) khi được cung cấp đồng thời. Đáng chú ý, với đặc trưng AST, mô hình này tăng mạnh nhất với mức cải thiện lên đến 2.8%, từ 85.0% lên 87.8%.

Hai mô hình còn lại là SVM và RF cũng ghi nhận mức tăng hiệu năng rõ rệt. Với SVM, cải thiện đến từ việc điều chỉnh hệ số C và lựa chọn hàm kernel phù hợp. Với RF, hiệu quả tăng khi số lượng cây và độ sâu được điều chỉnh.

Bảng 3. Kết quả đánh giá F1 - Score (sau điều chỉnh tham số)

Mô hình	Đặc trưng	Trước điều chỉnh	Sau điều chỉnh	Tăng
SVM	TF-IDF	87.4 ± 0.5	88.6 ± 0.4	1.2
	AST	84.3 ± 0.6	85.8 ± 0.6	1.5
	TF-IDF + AST	89.0 ± 0.4	90.3 ± 0.3	1.3
RF	TF-IDF	86.2 ± 0.7	87.1 ± 0.5	0.9
	AST	85.9 ± 0.5	86.8 ± 0.4	0.9
	TF-IDF + AST	87.2 ± 0.6	88.6 ± 0.5	1.4
XGBoost	TF-IDF	88.4 ± 0.4	90.6 ± 0.3	2.2
	AST	85.0 ± 0.6	87.8 ± 0.5	2.8
	TF-IDF + AST	89.9 ± 0.3	92.0 ± 0.3	2.1

Để hỗ trợ phân tích kết quả, Hình 4 dưới đây trình bày biểu đồ cột minh họa hiệu năng trung bình theo chỉ số F1-score của từng mô hình trên ba loại đặc trưng đầu vào: TF-IDF, AST và kết hợp cả hai. Trong biểu đồ, các cột màu đầy biểu thị kết quả đạt được trước khi tiến hành điều chỉnh siêu tham số, trong khi các phần sọc chéo được chồng lên thể hiện mức cải thiện về hiệu năng sau khi áp dụng các chiến lược tối ưu hóa, bao gồm Grid Search, Random Search và Bayesian optimization.



Hình 4. So sánh F1-Score trước và sau điều chỉnh siêu tham số

• Với ROC AUC

Tương tự như F1-score, về chỉ số ROC AUC trong Bảng 4 cho thấy hiệu năng các mô hình đều được cải thiện rõ rệt sau khi áp dụng các chiến lược điều chỉnh siêu tham số. Trong đó, XGBoost tiếp tục đạt kết quả cao nhất, đặc biệt khi sử dụng đặc trưng kết hợp (TF-IDF + AST), với ROC AUC trung bình đạt 94.7%, tăng 1.3% so với trước điều chỉnh (93.4%). Với đặc trưng AST, mô hình này cũng cải thiện từ 89.4% lên 90.6% (+1.2%), và với TF-IDF là từ 92.1% lên 93.2% (+1.1%). Hai mô hình còn lại, SVM và RF, cũng cho thấy hiệu quả học tốt hơn khi được điều chỉnh siêu tham số, với mức tăng dao động từ 0.9% đến 1.3% tùy đặc trưng đầu vào.

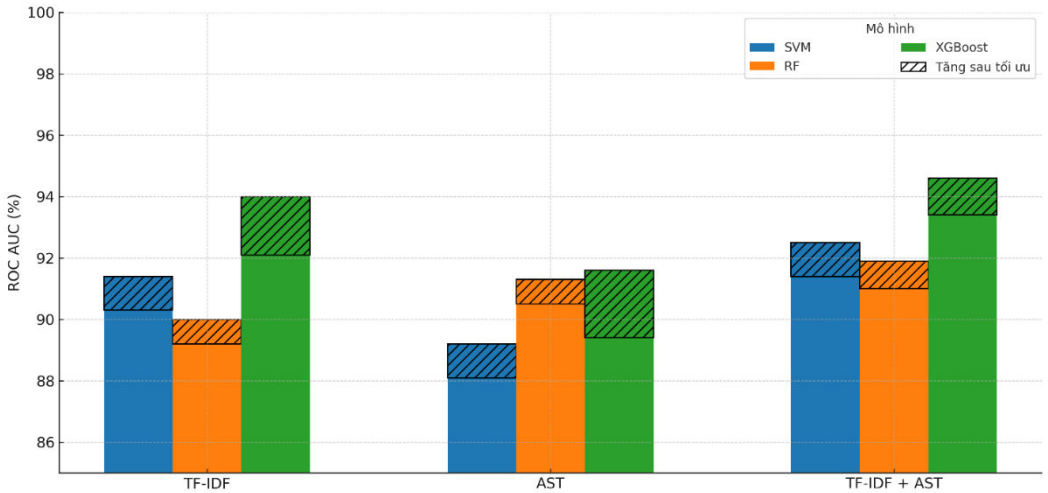
Đáng chú ý, Bayesian optimization tiếp tục chứng tỏ hiệu quả vượt trội, không chỉ giúp tìm được tổ hợp tham số tốt hơn trong không gian lớn mà còn rút ngắn thời gian tìm kiếm đáng kể so với các phương pháp truyền thống như Grid Search hay Random Search.

Những kết quả này chứng minh rằng tối ưu siêu tham số là yếu tố then chốt để khai thác tối đa tiềm năng của mô hình học máy, đặc biệt trong các bài toán phân loại phức tạp như phát hiện lỗ hổng bảo mật trong mã nguồn, nơi yêu cầu cả độ chính xác lẫn khả năng phân biệt cao giữa các trường hợp nhạy cảm.

Bảng 4. Kết quả đánh giá ROC AUC (sau điều chỉnh tham số)

Mô hình	Đặc trưng	Trước điều chỉnh	Sau điều chỉnh	Tăng
SVM	TF-IDF	90.3 ± 0.5	91.4 ± 0.4	1.2
	AST	88.1 ± 0.6	89.4 ± 0.6	1.3
	TF-IDF + AST	91.4 ± 0.4	92.6 ± 0.3	1.2
RF	TF-IDF	89.2 ± 0.6	90.1 ± 0.5	0.9
	AST	90.5 ± 0.5	91.4 ± 0.4	0.9
	TF-IDF + AST	91.0 ± 0.5	92.0 ± 0.5	1.0
XGBoost	TF-IDF	92.1 ± 0.4	93.2 ± 0.3	1.1
	AST	89.4 ± 0.6	90.6 ± 0.4	1.2
	TF-IDF + AST	93.4 ± 0.4	94.7 ± 0.4	1.3

Hình 5 trình bày sự cải thiện về chỉ số ROC AUC trung bình của các mô hình học máy khi áp dụng ba loại đặc trưng đầu vào, với cách thể hiện trực quan tương tự Hình 4. Mỗi cột trong biểu đồ được cấu trúc nhằm phân biệt rõ giữa hiệu năng ban đầu và phần cải thiện sau điều chỉnh siêu tham số, qua đó làm nổi bật mức độ tiến bộ của từng mô hình một cách trực quan và dễ so sánh.



Hình 5. So sánh ROC AUC trước và sau điều chỉnh siêu tham số

Tổng hợp lại, các kết quả ở mục 4 cho thấy hiệu năng của các mô hình học máy đều được cải thiện đáng kể sau khi áp dụng các chiến lược điều chỉnh siêu tham số. Mức độ cải thiện khác nhau tùy theo loại mô hình và đặc trưng đầu vào, trong đó XGBoost đạt hiệu quả cao nhất trên cả hai chỉ số F1-score và ROC AUC. Những kết quả này là cơ sở để đưa ra nhận định tổng quát về hiệu quả mô hình trong phần kết luận sau đây.

5. Kết luận

Từ những phân tích và so sánh trong mục 4, có thể thấy rằng mỗi mô hình học sâu đều có những ưu điểm và hạn chế riêng, phụ thuộc vào ngữ cảnh ứng dụng, dữ liệu đầu vào và mục tiêu triển khai. Nhằm hướng đến việc xây dựng các hệ thống phát hiện lỗ hổng hiệu quả hơn trong thực tế, chúng tôi đề xuất một số hướng cải tiến và phát triển mô hình trong tương lai như sau.

5.1. Đánh giá mô hình đề xuất

Nghiên cứu này đã xây dựng và triển khai một hệ thống đánh giá hiệu năng của các thuật toán phân lớp học máy truyền thống, bao gồm SVM, RF và XGBoost, trong bối cảnh phân loại mã nguồn theo hai lớp: an toàn và có lỗ hổng. Quy trình đề xuất được thiết kế theo hướng thực nghiệm toàn diện, với ba dạng đặc trưng đầu vào (TF-IDF, AST, kết hợp cả 2) và ba chiến lược điều chỉnh siêu tham số (Grid Search, Random Search, Bayesian optimization). Các thực nghiệm được thực hiện độc lập trên hai bộ dữ liệu tiêu chuẩn là Juliet Test Suite v1.3 và CodeXGLUE Defect Detection, giúp đảm bảo tính khách quan và khả năng tổng quát hóa của mô hình.

Kết quả cho thấy XGBoost là mô hình hoạt động hiệu quả nhất trong cả ba phương án đặc trưng, đặc biệt nổi bật khi kết hợp TF-IDF + AST. Sau khi điều chỉnh, mô hình này đạt F1-score lên đến 93.2% và ROC AUC lên tới 94.7%, vượt trội so với các mô hình còn lại. Điều này khẳng định rằng việc chọn đúng mô hình, kết hợp đặc trưng hợp lý và áp dụng kỹ thuật điều chỉnh phù hợp có thể tạo nên sự cải thiện đáng kể trong các hệ thống học máy truyền thống, đặc biệt với dữ liệu dạng mã nguồn có cấu trúc và tính ngữ nghĩa đan xen.

5.2. Lý do chọn 2 đặc trưng TF-IDF và AST

TF-IDF và AST được chọn vì mỗi phương pháp phản ánh một khía cạnh khác nhau của đoạn mã. TF-IDF là kỹ thuật truyền thống thường dùng trong xử lý ngôn ngữ tự nhiên, biểu diễn văn bản dưới dạng vector tần suất, đặc trưng, giúp mô hình học được thông tin ngữ liệu như tên biến, hàm, từ khóa, vốn rất hữu ích trong phát hiện mẫu mã nguy hiểm. Trong khi đó, AST là biểu diễn cây cú pháp, cho phép mô hình khai thác thông tin về cấu trúc chương trình, trình tự lệnh, và các khối điều khiển, điều mà TF-IDF không thể hiện được.

Việc kết hợp hai loại đặc trưng này mang lại hiệu quả đáng kể vì nó giúp mô hình học được cả bề rộng (từ khóa, ngữ cảnh) lẫn bề sâu (cấu trúc logic, quan hệ giữa các node mã). Các kết quả thực nghiệm cho thấy tổ hợp TF-IDF + AST luôn đạt hiệu năng cao hơn so với khi sử dụng một trong hai đặc trưng đơn lẻ, cả về F1-score và ROC AUC, trên mọi mô hình và bộ dữ liệu được kiểm thử.

5.3. Chiến lược điều chỉnh tham số: Grid Search, Random Search và Bayesian optimization

Điều chỉnh siêu tham số là bước quan trọng trong quy trình huấn luyện mô hình học máy, đặc biệt đối với các thuật toán có nhiều tham số điều chỉnh như XGBoost và RF. Trong nghiên cứu này, ba chiến lược phổ biến được áp dụng để lựa chọn cấu hình tham số phù hợp: Grid Search, Random Search và Bayesian optimization.

Grid Search là phương pháp đơn giản nhưng tiêu tốn nhiều thời gian, vì nó duyệt toàn bộ không gian tham số theo lưới. Random Search khắc phục được phần nào nhược điểm này bằng cách chọn ngẫu nhiên, tuy ít tốn thời gian hơn nhưng kết quả không ổn định. Trong khi đó, Bayesian optimization tỏ ra vượt trội khi sử dụng mô hình surrogate để dự đoán hiệu năng và chọn điểm tiếp theo cần đánh giá. Chiến lược này không những đạt kết quả cao hơn mà còn giảm đáng kể số lần huấn luyện cần thiết, nhất là với các mô hình phức tạp.

Kết quả thực nghiệm xác nhận rằng Bayesian optimization thường dẫn đến mô hình có hiệu năng tốt nhất trong thời gian ngắn hơn, là lựa chọn phù hợp cho các bài toán yêu cầu độ chính xác cao và tài nguyên huấn luyện có giới hạn.

Tài liệu tham khảo

[1] Benítez-Peña, S., Blanquero, R., Carrizosa, E. & Ramírez-Cobo, P. (2023). "*On support vector machines under a multiple-cost scenario*". *Truy cập ngày 10/06/2025, từ* https://arxiv.org/pdf/2312.14795

[2] Toth, M., & McConville, W. (2024). "*Design consistent random forest models for data collected from a complex sample*". *Survey Methodology*, 50(1), 187–205.

[3] Zheng, R., Liu, Y. & Tan, M. (2025). "Bayesian Optimization of Lasso and XGBoost Models for Comparative Analysis in Housing Price Prediction". *ITM Web of Conferences*, 120, 02016.

[4] Subaşı, N. (2024). "*Comprehensive Analysis of Grid and Randomized Search on Dataset Performance". European Journal of Engineering and Applied Sciences, 7(2), 77–83. doi:10.55581/ejeas.1581494.*

[5] Franceschi, L., Franceschi, L., Donini, M., Perrone, V., Klein, A., Archambeau, C., Seeger, M., Pontil, M., & Frasconi, P. (2024). *Hyperparameter Optimization in Machine Learning*: A Comprehensive Review. Truy cập ngày 15/01/2025, từ https://arxiv.org/abs/2410.22854

[6] González Duque, M., Bartels, S., Zainchkovskyy, Y., Hauberg, S., & Boomsma, W. (2024). *A Survey and Benchmark of High-Dimensional Bayesian Optimization of Discrete Sequences*. Truy cập ngày 15/01/2025, từ https://arxiv.org/abs/2406.04739

[7] *He, Y., & Li, X. (2024). "An extended Term Frequency–Inverse Document Frequency approach for improved keyword extraction". Information Processing & Management, 61(3), 103123.*

[8] [8] Sun, Sun, W., Fang, C., Miao, Y., Yuan, M., Chen, Y., Zhang, Q., & Chen, Z. (2023). *Abstract Syntax Tree for Programming Language Understanding and Representation:How Far Are We?*. Truy cập ngày 15/01/2025, từ https://arxiv.org/abs/2312.00413

[9] Grahn, D., & Zhang, J. (2021). An analysis of C/C++ datasets for machine learning assisted software vulnerability detection. *Proceedings of the Conference on Applied Machine Learning for Information Security* (tr. 1–13).

[10] Liu, S., Ma, W., Wang, J., Xie, X., Feng, R., & Liu, Y. (2024). Enhancing code vulnerability detection via vulnerability preserving data augmentation. *Proceedings of the ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, Tools and Theory for Embedded Systems (LCTES)* (tr. 1–12).