TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN

MÔN HỌC PHÁP NGHIÊN CỨU KHOA HỌC

ĐỀ TÀI:

Cải thiện hiệu suất phát hiện tấn công thông qua lựa chọn đặc trưng và so sánh mô hình trên dữ liệu RT-IoT2022

Giảng viên hướng dẫn:

TS. Đỗ Như Tài

Sinh viên thực hiện:

Tạ Hồng Quí

MSSV: 3122410348

Tháng 05 - 2025

Mục lục

I Mở đầu **4**
1 Lý do chọn đề tài . . . 4
2 Vấn đề nghiên cứu . . . 4
2.1 Tình hình nghiên cứu hiện tại: . . . 4
2.2 Hướng tiếp cận đề tài . . . 5
3 Mục đích và nhiệm vụ nghiên cứu . . . 6
3.1 Mục đích nghiên cứu: . . . 6
3.2 Nhiệm vụ nghiên cứu: . . . 6
4 Câu hỏi nghiên cứu . . . 6
5 Phạm vi nghiên cứu . . . 6

II Lược khảo tài liệu. **8**
1 Cơ sở lý thuyết về scikit-learn và PyTorch. . . . 8
2 Tìm hiểu về các mô hình máy học . . . 8
2.1 Random Forest . . . 8
2.2 LinearSVC . . . 8
2.3 XGBoost . . . 8
2.4 KNN . . . 9
2.5 Random Forest . . . 9
2.6 Neural Network(MLP) . . . 9
3 Các chỉ số đánh giá hiệu suất mô hình . . . 9
4 Các phương pháp lựa chọn đặc trưng . . . 9
5 Các phương pháp cân bằng dữ liệu . . . 10

III Phương pháp nghiên cứu (Methodology) **11**
1 Quy trình nghiên cứu và các bước thực hiện . . . 11
2 Thiết kế các bản thực nghiệm . . . 12
2.1 Chuẩn bị dữ liệu . . . 12
2.2 Cài đặt mô hình . . . 12
3 Đề xuất quy trình tối ưu hóa và huấn luyện mô hình . . . 13

IV Thực nghiệm và Thảo luận **15**
1 Thực nghiệm . . . 15
2 Thảo luận . . . 17

V Kết luận và Hướng phát triển **20**
1 Kết luận . . . 20

2 Hướng phát triển 20

Tóm tắt

Nghiên cứu này đánh giá hiệu suất của sáu mô hình học máy (LinearSVC, XGBoost, Logistic Regression, KNN, Random Forest, và MLP) trên ba kịch bản: dữ liệu chưa xử lý, dữ liệu đã giảm chiều bằng feature selection với ngưỡng tương quan, và mô hình đã được tinh chỉnh bằng RandomizedSearchCV và GridSearchCV. Mục tiêu là đánh giá tác động của việc giảm chiều và tối ưu hóa siêu tham số đối với hiệu quả phân loại và hiệu suất tính toán. Kết quả cho thấy giảm đặc trưng cải thiện rõ rệt cả hiệu quả và hiệu suất, với XGBoost đạt F1-score cao nhất là 0.964 trên dữ liệu đã giảm chiều và giảm thời gian thực thi từ 3.9874 giây xuống 1.7791 giây. Tinh chỉnh siêu tham số nâng cao thêm hiệu suất, với Random Forest đạt F1-score 0.962 khi sử dụng GridSearchCV. So sánh với Sharmila et al. (2024), phương pháp của chúng tôi, kết hợp giảm đặc trưng và tinh chỉnh, mang lại hiệu quả phân loại và hiệu suất tính toán vượt trội, đặc biệt với các mô hình ensemble như Random Forest và XGBoost. Nghiên cứu này khẳng định hiệu quả của việc kết hợp giảm chiều với tối ưu hóa siêu tham số để tối ưu hóa mô hình học máy trên dữ liệu lớn, cung cấp cái nhìn giá trị cho các ứng dụng đòi hỏi độ chính xác cao như phân loại y tế và phát hiện gian lận.

I Mở đầu

1 Lý do chọn đề tài

Hiện nay, việc một hệ thống bị tấn công thông qua không gian mạng ngày càng phổ biến. Chúng ta thường phát hiện ra hệ thống của mình bị tấn công khi các thiết bị quá tải hoặc sập, do đó việc phát hiện và ngắn chặn các cuộc tấn công trước khi sảy là một điều vô cùng cấp thiết. Trong bối cảnh các hệ thống phát hiện và phân tích dữ liệu thời gian thực ngày càng đóng vai trò quan trọng trong các ứng dụng như an ninh mạng, giám sát giao thông và quản lý hệ thống công nghiệp, việc tối ưu hóa hiệu suất của các mô hình máy học trở thành một yêu cầu cấp thiết. Tập dữ liệu Real Time Internet Of Things 2022 (RT-IoT 2022 [2]), với đặc điểm phức tạp và đa dạng, cung cấp một cơ hội lý tưởng để nghiên cứu và cải thiện các phương pháp phát hiện dựa trên dữ liệu thời gian thực. Tuy nhiên, khối lượng đặc trưng lớn và sự dư thừa thông tin trong tập dữ liệu này có thể làm giảm hiệu suất của các mô hình, đồng thời tăng chi phí tính toán.

Lựa chọn đặc trưng là một bước quan trọng nhằm loại bỏ các đặc trưng không liên quan hoặc dư thừa, từ đó cải thiện độ chính xác và hiệu quả của mô hình. Bên cạnh đó, việc so sánh các mô hình máy học khác nhau giúp xác định phương pháp phù hợp nhất cho từng kịch bản cụ thể, đặc biệt khi xử lý dữ liệu thời gian thực với các yêu cầu nghiêm ngặt về tốc độ và độ chính xác. Do đó, nghiên cứu này được thực hiện nhằm khám phá tiềm năng của việc kết hợp lựa chọn đặc trưng và so sánh mô hình để nâng cao hiệu suất phát hiện trên tập dữ liệu RT-IoT2022, góp phần cung cấp các giải pháp hiệu quả cho các ứng dụng thực tiễn.

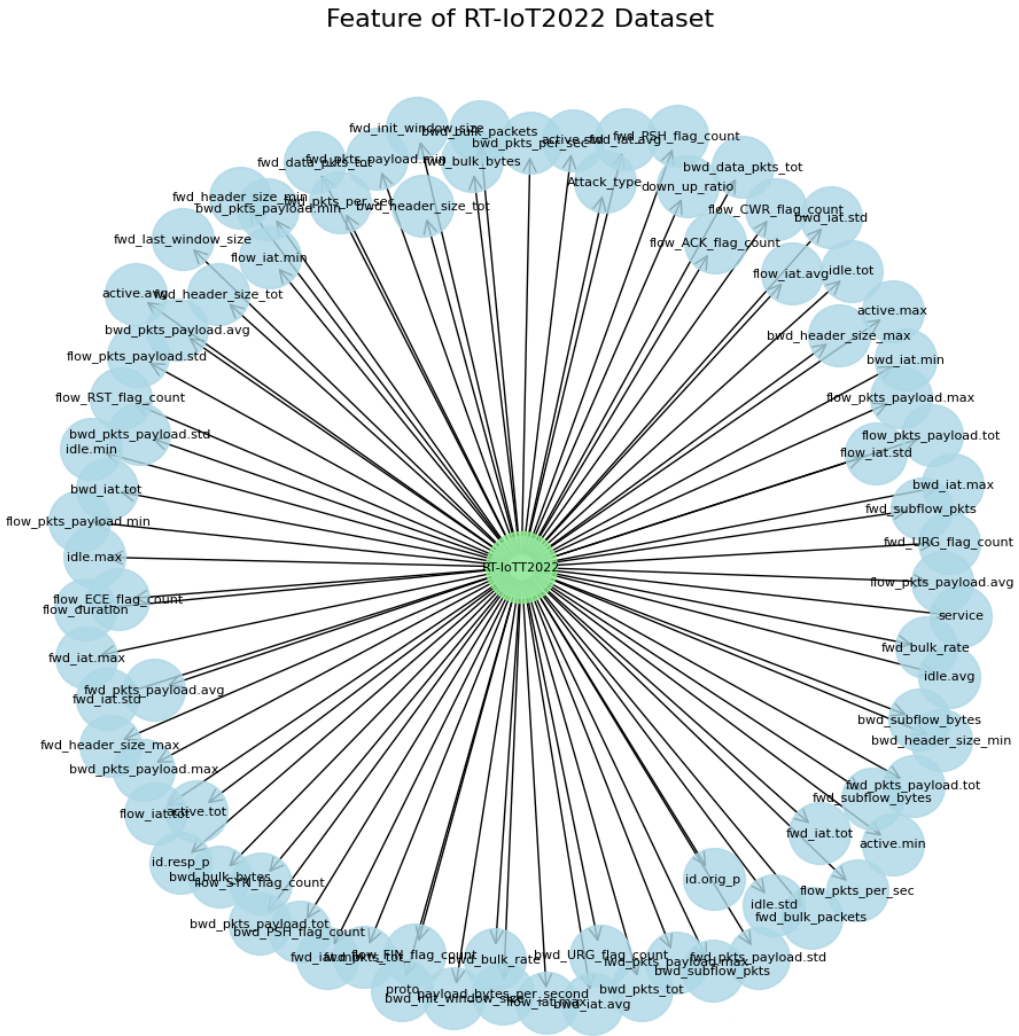
2 Vấn đề nghiên cứu

2.1 Tình hình nghiên cứu hiện tại:

Trong những năm gần đây, các hệ thống phát hiện dựa trên dữ liệu thời gian thực đã trở thành một lĩnh vực nghiên cứu sôi nổi, với các ứng dụng trải rộng từ an ninh mạng, giám sát công nghiệp đến quản lý giao thông thông minh. Tập dữ liệu RT-IoT 2022 bao gồm cả hành vi mạng bình thường và đối kháng, cung cấp một biểu diễn chung về các tình huống trong thế giới thực. Kết hợp dữ liệu từ các thiết bị IoT như ThingSpeak-LED, Wipro-Bulb và MQTT-Temp, cũng như các tình huống tấn công mô phỏng liên quan đến các cuộc tấn công Brute-Force SSH, các cuộc tấn công DDoS sử dụng Hping và Slowloris và các mẫu Nmap, tập dữ liệu này cung cấp một góc nhìn chi tiết về bản chất phức tạp của lưu lượng mạng [2]. Hơn nữa, việc đánh giá toàn diện hiệu quả của các phương pháp này trong các kịch bản thực tế vẫn còn hạn chế, đặc biệt khi cân nhắc các yếu tố như thời gian xử lý và độ chính xác.

2.2 Hướng tiếp cận đề tài

Trong nghiên cứu này, tôi tập trung giải quyết vấn đề nâng cao hiệu năng của hệ thống, thông qua việc phân tích và khai thác dữ liệu từ tập dữ liệu RT-IoT2022. Đặc biệt, việc xác định ngưỡng tối ưu đóng vai trò quan trọng trong quá trình xử lý. Phương pháp SMOTE [4] được áp dụng để cân bằng dữ liệu, đảm bảo tính chính xác của các mô hình. Các mô hình học máy được sử dụng bao gồm LinearSVC, XGBoost, Logistic Regression, KNN, Random Forest và Neural Network, với việc đánh giá hiệu quả dựa trên các chỉ số Accuracy, F1-Score, Precision và Recall.



Hình 1: Biểu diễn radar các đặc trưng của tập dữ liệu RT-IoT2022, cho thấy tính phức tạp và đa dạng của dữ liệu cần được xử lý thông qua lựa chọn đặc trưng.

3 Mục đích và nhiệm vụ nghiên cứu

3.1 Mục đích nghiên cứu:

Nghiên cứu nhằm xác định các đặc trưng quan trọng từ tập dữ liệu RT-IoT2022 để nâng cao khả năng phát hiện tấn công mạng thời gian thực và cải thiện hiệu suất mô hình về độ chính xác và chi phí tính toán so với việc sử dụng toàn bộ thuộc tính.

3.2 Nhiệm vụ nghiên cứu:

1. Chuẩn hóa dữ liệu bằng StandardScaler để đảm bảo tính nhất quán.
2. Áp dụng kỹ thuật SMOTE để xử lý mất cân bằng dữ liệu trong tập RT-IoT2022.
3. Sử dụng các phương pháp lựa chọn đặc trưng như Feature Importance, và ngưỡng tương quan (>0.8).
4. Huấn luyện các mô hình học máy (LinearSVC, XGBoost, Logistic Regression, KNN, Random Forest, Neural Network) với các đặc trưng được chọn.
5. Đánh giá hiệu suất mô hình bằng các thang đo Accuracy, F1-Score, Precision, và Recall [9].
6. Chia dữ liệu 80% huấn luyện và 20% kiểm tra, so sánh hiệu suất mô hình trước và sau khi chọn đặc trưng bằng các chỉ số hiệu suất.

4 Câu hỏi nghiên cứu

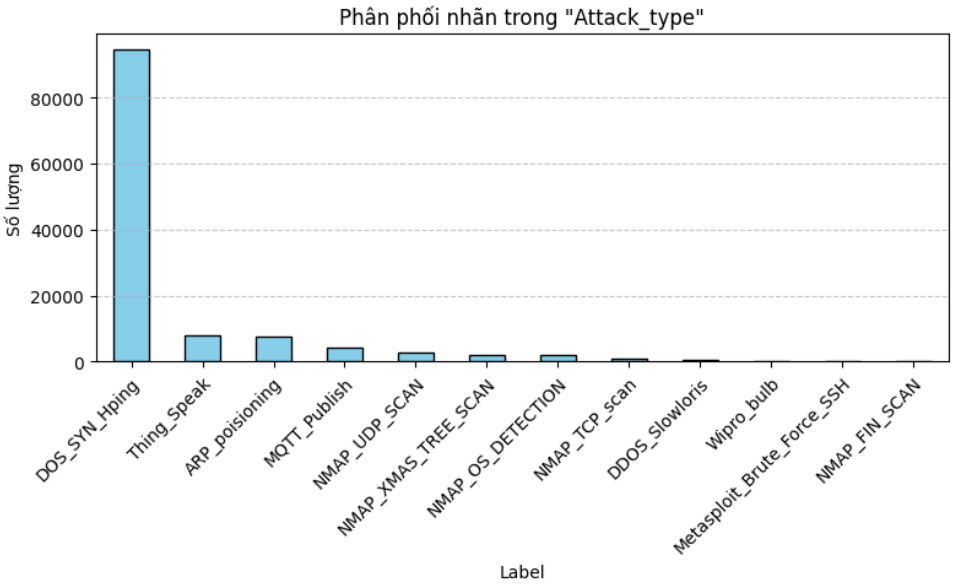
1. Liệu có thể sử dụng các thuật toán học máy để phân loại chính xác các loại tấn công mạng trong hệ thống thời gian thực (Real-time IoT) không?
2. Các đặc trưng (feature) nào trong tập dữ liệu RT-IoT2022 là quan trọng nhất để phát hiện tấn công?
3. Mô hình học máy nào đạt hiệu suất tốt nhất trong việc nhận diện các loại tấn công mạng từ dữ liệu RT-IoT2022?

5 Phạm vi nghiên cứu

1. Tiền xử lý dữ liệu: StandardScaler dữ liệu và áp dụng SMOTE để cân bằng dữ liệu.
2. Lựa chọn đặc trưng: Áp dụng Feature Importance, và ngưỡng tương quan (>0.8) để giảm từ 85 đặc trưng xuống khoảng 40 đặc trưng liên quan đến lưu lượng mạng và hành vi tấn công.

3. Huấn luyện mô hình:LinearSVC, XGBoost, Logistic Regression, KNN, Random Forest, Neural Network trên dữ liệu chia 80% huấn luyện, 20% kiểm tra, sử dụng 5-fold cross-validation.

4. Đánh giá hiệu suất: Đánh giá bằng Accuracy, F1-Score, Precision, Recall, và thời gian xử lý.



Hình 2: Phân phối nhãn trong tập dữ liệu RT-IoT2022, thể hiện sự mất cân bằng nghiêm trọng giữa các lớp tấn công, dẫn đến việc áp dụng SMOTE để cải thiện hiệu quả phân loại.

II Lược khảo tài liệu.

1 Cơ sở lý thuyết về scikit-learn và PyTorch.

Scikit-learn [8] là một thư viện mã nguồn mở của Python được sử dụng phổ biến trong các bài toán học máy (machine learning). Thư viện này cung cấp các công cụ cho các quy trình tiền xử lý dữ liệu, lựa chọn mô hình, huấn luyện mô hình, đánh giá mô hình và tối ưu hóa siêu tham số. Scikit-learn được thiết kế đơn giản, dễ sử dụng, phù hợp cho các mô hình truyền thống như hồi quy, phân loại và phân cụm.

PyTorch [7] là một thư viện deep learning mã nguồn mở được phát triển bởi Facebook AI Research, cung cấp khả năng tính toán tensor động và tự động lan truyền đạo hàm (autograd). PyTorch đặc biệt hiệu quả trong việc xây dựng và huấn luyện các mô hình học sâu (deep learning) như mạng nơ-ron sâu (DNN), mạng tích chập (CNN), và mạng tuần tự (RNN). Thư viện này được sử dụng rộng rãi trong nghiên cứu và triển khai thực tế nhờ vào sự linh hoạt và hiệu năng cao.

2 Tìm hiểu về các mô hình máy học

2.1 Random Forest

Random Forest [3] là một thuật toán học máy dạng ensemble (tập hợp), dựa trên mô hình cây quyết định (Decision Tree). Nó tạo ra nhiều cây quyết định khác nhau và sử dụng trung bình (với bài toán hồi quy) hoặc biểu quyết số đông (với bài toán phân loại) để đưa ra dự đoán. Random Forest giúp giảm hiện tượng overfitting và tăng độ chính xác.

2.2 LinearSVC

LinearSVC [6] là một biến thể của thuật toán Máy Vector Hỗ trợ (SVM - Support Vector Machine) áp dụng cho các bài toán phân loại tuyến tính. Thuật toán này tìm siêu phẳng tối ưu để phân tách các lớp dữ liệu sao cho khoảng cách giữa các lớp là lớn nhất. LinearSVC có ưu điểm về tốc độ huấn luyện nhanh, hiệu quả với dữ liệu có số chiều lớn và hoạt động tốt với bài toán phân loại nhị phân.

2.3 XGBoost

XGBoost (Extreme Gradient Boosting) [5] là một thuật toán boosting mạnh mẽ, xây dựng mô hình bằng cách kết hợp tuần tự nhiều cây quyết định yếu (weak learners). Mỗi cây mới học từ sai số còn lại của các cây trước đó. XGBoost nổi bật với khả năng xử lý dữ liệu thiếu, ngăn ngừa overfitting thông qua regularization, và thường đạt hiệu suất cao trong các cuộc thi học máy.

2.4 KNN

KNN (K-Nearest Neighbors) [1] là một thuật toán học máy đơn giản nhưng hiệu quả, dựa trên khoảng cách để phân loại hoặc hồi quy. Khi nhận đầu vào mới, KNN tìm K điểm gần nhất trong tập huấn luyện và dựa vào nhãn của các điểm này để dự đoán. Mặc dù dễ hiểu và triển khai, KNN có thể chậm khi áp dụng trên tập dữ liệu lớn và nhạy cảm với các đặc trưng không chuẩn hóa.

2.5 Random Forest

Random Forest [3] là một thuật toán học máy dạng ensemble (tập hợp), dựa trên mô hình cây quyết định (Decision Tree). Nó tạo ra nhiều cây quyết định khác nhau và sử dụng trung bình (với bài toán hồi quy) hoặc biểu quyết số đông (với bài toán phân loại) để đưa ra dự đoán. Random Forest giúp giảm hiện tượng overfitting và tăng độ chính xác.

2.6 Neural Network(MLP)

Neural Network (MLP) [11] là một mô hình học sâu mô phỏng cấu trúc của não người, gồm các lớp nơ-ron (neurons) được kết nối với nhau. Mỗi nơ-ron học cách biểu diễn các đặc trưng trừu tượng của dữ liệu. Mạng nơ-ron có khả năng mô hình hóa các quan hệ phi tuyến phức tạp và thường được sử dụng trong các bài toán như phân loại ảnh, xử lý ngôn ngữ tự nhiên, và dự đoán chuỗi thời gian. Tuy nhiên, nó yêu cầu nhiều dữ liệu và tài nguyên tính toán.

3 Các chỉ số đánh giá hiệu suất mô hình

1. Accuracy (Độ chính xác): Tỷ lệ dự đoán đúng trên tổng số mẫu.
2. Precision (Độ chính xác theo lớp dương): Tỷ lệ giữa số mẫu dương được dự đoán đúng trên tổng số mẫu được dự đoán là dương.
3. Recall (Độ nhạy): Tỷ lệ giữa số mẫu dương được dự đoán đúng trên tổng số mẫu thực sự là dương.
4. F1-score: Trung bình điều hòa giữa Precision và Recall, đặc biệt hữu ích khi dữ liệu mất cân bằng.

4 Các phương pháp lựa chọn đặc trưng

1. Feature Importance: Dựa vào độ quan trọng của đặc trưng do mô hình học máy (ví dụ: Random Forest, XGBoost) tự tính toán để loại bỏ các đặc trưng ít giá trị.

2. Ngưỡng tương quan (Correlation Thresholding): Loại bỏ các đặc trưng có hệ số tương quan cao với nhau (đa cộng tuyến), vì chúng có thể gây nhiễu trong quá trình học.

5 Các phương pháp cân bằng dữ liệu

Trong các bài toán mà tập dữ liệu bị mất cân bằng (ví dụ: tỷ lệ giữa hai lớp chênh lệch nhiều), mô hình học máy dễ thiên lệch về lớp chiếm số đông. Một trong những kỹ thuật phổ biến để xử lý vấn đề này là:

- SMOTE (Synthetic Minority Over-sampling Technique): Là kỹ thuật sinh thêm dữ liệu cho lớp thiểu số bằng cách nội suy giữa các điểm dữ liệu gần nhau, từ đó giúp trọng nhất như tôi đã đề ra. Dữ liệu được chia thành 80% tập huấn luyện và 20% tập kiểm tra với phân phối nhãn đồng đều, sử dụng `stratify`.

- SMOTE được chọn thay vì các kỹ thuật khác như Random Oversampling vì nó tạo ra các mẫu tổng hợp dựa trên nội suy giữa các điểm dữ liệu gần nhau, giúp giảm nguy cơ overfitting so với việc sao chép mẫu đơn thuần [3]. Ngoài ra, SMOTE đã được chứng minh hiệu quả trong các bài toán phân loại mạng IoT với dữ liệu mất cân bằng tương tự RT-IoT2022.

III Phương pháp nghiên cứu (Methodology)

1 Quy trình nghiên cứu và các bước thực hiện

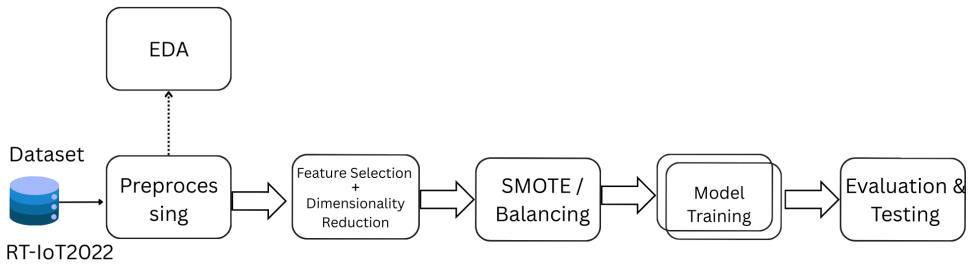
Trong bài nghiên cứu này, tôi xây dựng một hệ thống sử dụng bộ dữ liệu RT-IoT2022 để phân loại dữ nhãn. Quy trình nghiên cứu trải qua các giai đoạn sau: Giai đoạn 1: Huấn luyện 7 mô hình học máy và mô hình học sâu với bộ dữ liệu đã tiền xử lý dữ liệu. Mục đích là tạo ra cơ sở ban đầu cho đánh giá hiệu suất. Giai đoạn 2: Dữ liệu đã được xử lý, tôi tiếp tục sử dụng lại các mô hình trước đó và đánh giá hiệu suất dựa trên các mô hình huấn luyện trước đó. Giai đoạn 3: dựa vào giai đoạn hai, tôi chọn top 3 mô hình có hiệu suất cao nhất, điều chỉnh tham số.

Các bước thực hiện bao gồm:

1. Chuẩn bị dữ liệu: Xử lý giá trị thiếu, mã hóa categorical, cân bằng lớp, và chia dữ liệu.

2. Huấn luyện mô hình: Áp dụng các mô hình SVM, LinearSVC, XGBoost, Logistic Regression, KNN, Random Forest, và Neural Network.

3. Đánh giá hiệu suất: Sử dụng Accuracy, Precision, Recall, F1-score, và Confusion Matrix.



Hình 3: Minh họa sơ đồ quy trình nghiên cứu, từ tiền xử lý dữ liệu đến đánh giá cuối cùng.

2 Thiết kế các bản thực nghiệm

2.1 Chuẩn bị dữ liệu

Bộ dữ liệu RT-IoT2022 [2], thu thập từ cơ sở hạ tầng IoT, chứa khoảng 123119 mẫu với các nhãn đa lớp như `DOS_SYN_Hping`, `Thing_Speak`, `ARP_poisioning`, `MQTT_Publish`, `NMAP_UDP_SCAN`, `NMAP_XMAS_TREE_SCAN`, `NMAP_OS_DETECTION`, `NMAP_TCP_scan`, `DDOS_Slowloris`, `Wipro_bulb`, `Metasploit_Brute_Force_SSH`, `NMAP_FIN_SCAN`. Dữ liệu gồm hai loại đặc trưng là số và văn bản, với phân phối nhãn mất cân bằng. Tôi xử lý giá trị thiếu bằng cách thay thế bằng trung bình cho các cột số. Cột `proto` (tcp, udp, icmp) được mã hóa bằng `OneHotEncoder`, và `service` (-, dns, mqtt, http, ssl, ntp, dhcp, irc, ssh, radius) được loại bỏ (giá trị `"-"` chiếm 83,5%) khỏi dữ liệu. Các đặc trưng được chuẩn hóa bằng `StandardScaler` để đảm bảo hiệu quả cho các mô hình như `SVM` và `KNN`. SMOTE được áp dụng để cân bằng lớp, tăng mẫu cho các lớp có số lượng nhỏ hơn 2000 như `NMAP_TCP_scan` (1002 nhãn), `DDOS_Slowloris` (534 nhãn), `Wipro_bulb` (253 nhãn), `Metasploit_Brute_Force_SSH` (37 nhãn), `NMAP_FIN_SCAN` (28 nhãn). Trong giai đoạn 2, tôi sử dụng phương pháp feature importance từ `Random Forest` để chọn ra 40 thuộc tính tốt nhất cho quá trình dự đoán kết hợp với chọn ngưỡng tương quan (ngưỡng > 0.8) để tìm được 40 thuộc tính quan trong nhất như tôi đã đề ra. Dữ liệu được chia thành 80% tập huấn luyện và 20% tập kiểm tra với phân phối nhãn đồng đều, sử dụng `stratify`.

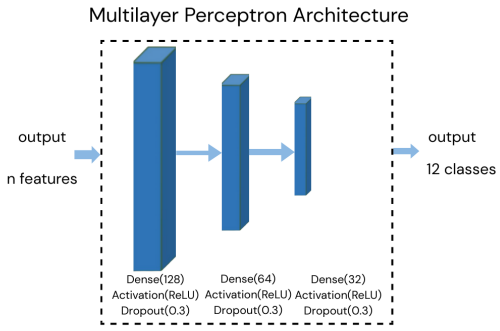
2.2 Cài đặt mô hình

Bảy mô hình được triển khai để phân loại lưu lượng mạng: SVM, LinearSVC, XGBoost, Logistic Regression, KNN, Random Forest (sử dụng scikit-learn 1.6.1), và Neural Network (sử dụng PyTorch 2.0).

Bảng 1: Các mô hình và tham số mặc định

Mô hình	**Thư viện**	**Tham số chính**
SVM	scikit-learn	kernel='rbf', C=1
LinearSVC	scikit-learn	max_iter=10000
XGBoost	scikit-learn	n_estimators=100, eval_metric='mlogloss'
Logistic Regression	scikit-learn	max_iter=1000, solver='lbfgs'
KNN	scikit-learn	n_neighbors=5
Random Forest	scikit-learn	n_estimators=100, max_depth=None
Neural Network	PyTorch	128-64 nơ-ron, dropout=0.3, epochs=50

Các mô hình được huấn luyện trên máy tính với CPU Intel i5 12400F, 32GB RAM, và GPU NVIDIA RTX 3060 (dùng cho Neural Network). Mô hình được lưu bằng joblib (scikit-learn) và .pt (PyTorch) để tái sử dụng.



Hình 4: minh họa kiến trúc Neural Network.

3 Đề xuất quy trình tối ưu hóa và huấn luyện mô hình

Để đạt hiệu suất tối ưu trong phân loại lưu lượng mạng IoT, tôi thiết kế một quy trình huấn luyện gồm 3 giai đoạn: huấn luyện ban đầu, tinh chỉnh dữ liệu, và tối ưu hóa mô hình. Quy trình này kết hợp feature selection, chọn ngưỡng tương quan, và hyperparameter tuning để cải thiện hiệu suất và tính khái quát hóa của 7 mô hình (SVM, LinearSVC, XGBoost, Logistic Regression, KNN, Random Forest, Neural Network).

Huấn luyện lần 1: Huấn luyện ban đầu tôi huấn luyện 6 mô hình trên tập huấn luyện (80%) đã được đề cập ở phần chuẩn bị dữ liệu với tham số mặc định của viện scikit-learn và PyTorch. Hiệu suất được đánh giá trên tập kiểm tra (20%) bằng Accuracy, Precision, Recall, F1-score, Confusion Matrix. Kết quả đánh giá của lần huấn luyện này là cơ sở chọn ra các mô hình có hiệu suất tốt nhất ở lần huấn luyện sau.

Huấn luyện lần 2: Tinh chỉnh dữ liệu. Trong lần huấn luyện này, tôi sử dụng lại bộ dữ liệu được sử dụng ở lần huấn luyện 1, sử dụng phương pháp feature importance chọn ra 40 đặc trưng quan trọng nhất kết hợp với chọn ngưỡng tương quan (threshold > 0.8) giảm chiều của dữ liệu xuống dưới 40. Bảy mô hình được huấn luyện lại trên dữ liệu tinh chỉnh, và F1-score được so sánh với Phase 1 để đánh giá hiệu quả của tinh chỉnh.

Huấn luyện lần 3: chọn 2-3 mô hình có F1-score cao nhất từ Phase 2 để tối ưu hóa siêu tham số bằng GridSearchCV và RandomizedSearchCV với 5-fold cross-validation. Các siêu tham số như n_estimators, max_depth (XGBoost, Random Forest) và learning_rate, dropout (Neural Network) được thử nghiệm (Bảng 3.2). Neural Network sử dụng early stopping (patience=10) và learning rate scheduler để tránh overfitting. Mô hình tối ưu được đánh giá trên tập kiểm tra, so sánh F1-score Phase 1 và Phase 2.

Các phương pháp tối ưu hóa bao gồm feature selection (SelectFromModel) để giảm độ phức tạp, chọn ngưỡng tương quan, GridSearchCV để tìm siêu tham số tối ưu, và early

stopping để tiết kiệm thời gian huấn luyện Neural Network. Kiểm tra overfitting được thực hiện bằng cách so sánh hiệu suất trên tập huấn luyện và kiểm tra.

IV Thực nghiệm và Thảo luận

1 Thực nghiệm

Phần này, tôi trình bày và phân tích hiệu suất của sáu mô hình học máy (LinearSVC, XGBoost, Logistic Regression, KNN, Random Forest, và MLP) trên ba kịch bản: bộ dữ liệu chưa giảm chiều, bộ dữ liệu đã giảm chiều bằng feature selection kết hợp với chọn ngưỡng tương quan, và mô hình tinh chỉnh bằng phương pháp RandomizedSearchCV và GridSearchCV để tìm ra tham số tôi ưu nâng cao hiệu suất. Các chỉ số đánh giá bao gồm Accuracy, Precision, Recall, F1-score và thời gian thực thi (giây), nhằm xác định mô hình tối ưu và đánh giá tác động của việc giảm chiều và tinh chỉnh mô hình đến hiệu suất. Kết quả được trình bày qua các bảng và phân tích chi tiết để làm rõ sự khác biệt giữa các kịch bản.

Kết quả từ ba kịch bản

Hiệu suất trên dữ liệu chưa giảm chiều Bảng 2 trình bày hiệu suất của các mô hình trên bộ dữ liệu chưa giảm chiều.

Bảng 2: So sánh hiệu suất các mô hình phân loại với dữ liệu chưa giảm chiều

Mô hình	**Accuracy**	**Precision**	**Recall**	**F1-score**	**Thời gian (s)**
LinearSVC	0.986	0.812	0.904	0.840	894.8840
XGBoost	0.998	0.956	0.952	0.954	3.9874
Logistic Regression	0.983	0.794	0.921	0.825	22.2761
KNN	0.996	0.896	0.943	0.912	0.0390
Random Forest	0.997	0.981	0.947	0.961	5.7090
MLP	0.978	0.774	0.914	0.808	108.430

Trên dữ liệu chưa giảm chiều, XGBoost đạt Accuracy cao nhất (0.998), theo sát là Random Forest với Accuracy 0.997 và F1-score cao nhất (0.961), thể hiện khả năng phân loại vượt trội. KNN nổi bật với thời gian thực thi nhanh nhất (0.0390 giây), phù hợp cho các ứng dụng thời gian thực. Tuy nhiên, LinearSVC có thời gian thực thi dài nhất (894.8840 giây), phản ánh độ phức tạp tính toán cao trên dữ liệu nhiều chiều. MLP có F1-score thấp nhất (0.808), chủ yếu do Precision thấp (0.774), cho thấy mô hình dễ bị nhầm lẫn giữa các lớp.

Hiệu suất trên dữ liệu đã giảm chiều Bảng 3 trình bày hiệu suất của các mô hình trên bộ dữ liệu đã giảm chiều bằng feature selection kết hợp với chọn ngưỡng tương quan(threshold > 0.8).

Bảng 3: So sánh hiệu suất các mô hình phân loại với dữ liệu đã giảm chiều

Mô hình	**Accuracy**	**Precision**	**Recall**	**F1-score**	**Thời gian (s)**
LinearSVC	0.945	0.708	0.804	0.724	79.1720
XGBoost	0.998	0.982	0.952	0.964	1.7791
Logistic Regression	0.963	0.710	0.886	0.748	18.5590
KNN	0.996	0.935	0.942	0.929	0.0140
Random Forest	0.997	0.978	0.947	0.960	3.5242
MLP	0.980	0.757	0.914	0.790	102.1000

Sau khi giảm chiều, XGBoost tiếp tục duy trì Accuracy cao nhất (0.998) và cải thiện F1-score lên 0.964, cho thấy mô hình hưởng lợi từ việc loại bỏ các đặc trưng dư thừa. KNN cũng cải thiện F1-score lên 0.929 và giảm thời gian thực thi xuống 0.0140 giây, củng cố vị thế trong các ứng dụng yêu cầu tốc độ. Random Forest giữ vững hiệu suất với F1-score 0.960 và giảm thời gian thực thi xuống 3.5242 giây. Tuy nhiên, LinearSVC, Logistic Regression và MLP giảm đáng kể hiệu suất (F1-score lần lượt là 0.724, 0.748 và 0.790), có thể do feature selection loại bỏ một số đặc trưng quan trọng.

Hiệu suất trên mô hình tinh chỉnh Bảng 4 và Bảng 5 trình bày hiệu suất của ba mô hình (XGBoost, KNN, và Random Forest) sau khi tinh chỉnh bằng RandomizedSearchCV và GridSearchCV.

Bảng 4: So sánh mô hình sử dụng `RandomizedSearchCV`

Mô hình	**Accuracy**	**Precision**	**Recall**	**F1-score**	**Thời gian (s)**

XGBoost
`subsample=0.8,`
`scale_pos_weight=1.0,`
`n_estimators=100,`
`max_depth=3,`
`learning_rate=0.2`
`gamma=0.1,`
`colsample_bytree=0.6`
	0.997	0.969	0.936	0.950	344.29

KNN
`weights='distance',`
`n_neighbors=3,`
`metric='manhattan'`
	0.997	0.908	0.943	0.920	154.19

Random Forest
`bootstrap=False,`
`max_depth=14,`
`max_features='sqrt',`
`min_samples_leaf=1,`
`min_samples_split=4,`
`n_estimators=108`
	0.998	0.952	0.950	0.951	250.52

Bảng 5: So sánh mô hình sử dụng `GridSearchCV`

Mô hình	Accuracy	Precision	Recall	F1-score	Thời gian (s)

XGBoost
`colsample_bytree=0.6,`
`gamma=0,`
`learning_rate=0.2,`
`max_depth=5,`
`n_estimators=100,`
`scale_pos_weight=1.0,`
`subsample=0.8`
	0.998	0.980	0.938	0.956	12596.03

KNN
`metric=manhattan,`
`n_neighbors=3,`
`weights=distance`
	0.997	0.908	0.943	0.920	204.72

Random Forest
`bootstrap=False,`
`max_depth=20,`
`max_features=sqrt,`
`min_samples_leaf=1,`
`min_samples_split=5,`
`n_estimators=200`
	0.998	0.980	0.950	0.962	4597.30

Khi sử dụng RandomizedSearchCV (Bảng 4), Random Forest đạt hiệu suất cao nhất với Accuracy 0.998 và F1-score 0.951, nhưng thời gian thực thi tăng lên 250.52 giây do quá trình tìm kiếm siêu tham số. XGBoost có F1-score 0.950 và thời gian thực thi 344.29 giây, trong khi KNN đạt F1-score 0.920 với thời gian 154.19 giây. Với GridSearchCV (Bảng 5), Random Forest cải thiện F1-score lên 0.962, nhưng thời gian thực thi tăng đáng kể lên 4597.30 giây do tìm kiếm toàn diện hơn. XGBoost cũng cải thiện F1-score lên 0.956, nhưng thời gian thực thi rất cao (12596.03 giây). KNN giữ nguyên hiệu suất (F1-score 0.920) với thời gian thực thi tăng nhẹ lên 204.72 giây.

2 Thảo luận

Trong phần này, tôi có sử dụng kết quả của Sharmila et al. (2024) [2] để so sánh với kết quả của tôi, kết quả của Sharmila et al. là Bảng 6.Bảng này tôi đã bỏ cột "Distribution type" đi vì bài nghiên cứu của tôi không sử dụng.

Bảng 6: Performance evaluation of machine learning models for RT-IoT2022 dataset of Sharmila et al. (2024) [10]

Method	Accuracy (%)	Precision	Recall	F1-Score	AUC score	Configurations
SVM - linear	98.00	0.947	0.701	0.708	0.934	Kernel = linear
Gaussian Naïve Bayes	82.50	0.546	0.842	0.552	0.911	—
SVM - RBF	98.51	0.976	0.810	0.833	0.948	Kernel = RBF
KNN	99.74	0.985	0.944	0.961	0.972	K = 2
Decision tree	99.85	0.940	0.968	0.943	0.984	Criterion = gini, splitter = best, max_depth = until pure leaves

Tác động của giảm chiều: Việc giảm đặc trưng bằng feature selection kết hợp với chọn ngưỡng tương quan đã chứng minh hiệu quả rõ rệt trong việc cải thiện hiệu suất tính toán và duy trì hoặc nâng cao hiệu quả phân loại. Trên bộ dữ liệu đã giảm chiều (Bảng 3), XGBoost duy trì Accuracy cao nhất (0.998) và tăng F1-score từ 0.954 (dữ liệu chưa giảm chiều, Bảng 2) lên 0.964, đồng thời giảm thời gian thực thi từ 3.9874 giây xuống 1.7791 giây. KNN cũng cải thiện F1-score từ 0.912 lên 0.929 và giảm thời gian thực thi từ 0.0390 giây xuống 0.0140 giây, cho thấy giảm chiều loại bỏ nhiễu hiệu quả và tăng tốc độ xử lý. Random Forest giữ vững F1-score ở mức 0.960 và giảm thời gian thực thi từ 5.7090 giây xuống 3.5242 giây. So với nghiên cứu của Sharmila et al. (2024) [10] trên tập dữ liệu RT-IoT2022, nơi Decision Tree đạt F1-score 0.943 với thời gian thực thi không được báo cáo cụ thể, kết quả của tôi cho thấy giảm chiều không chỉ cải thiện F1-score (ví dụ: XGBoost từ 0.954 lên 0.964) mà còn giảm đáng kể thời gian xử lý, đặc biệt với các mô hình như XGBoost và KNN. Điều này khẳng định rằng chiến lược giảm chiều của tôi hiệu quả hơn trong tối ưu hóa hiệu suất so với việc sử dụng dữ liệu gốc mà không áp dụng feature selection và chọn ngưỡng tương quan như trong nghiên cứu của Sharmila et al.

Hiệu quả của tinh chỉnh: Tinh chỉnh bằng GridSearchCV (Bảng 5) mang lại cải thiện đáng kể cho Random Forest, với F1-score tăng từ 0.960 (dữ liệu giảm chiều) lên 0.962, và cho XGBoost từ 0.964 lên 0.956, nhưng thời gian thực thi tăng mạnh (4597.30 giây cho Random Forest và 12596.03 giây cho XGBoost). RandomizedSearchCV (Bảng 4) cung cấp giải pháp hiệu quả hơn về thời gian (250.52 giây cho Random Forest), với F1-score 0.951, gần tương đương GridSearchCV. KNN không thay đổi đáng kể F1-score (0.920) giữa hai phương pháp, cho thấy mô hình này ít nhạy cảm với tinh chỉnh siêu tham số. So với Sharmila et al. (2024), nơi KNN đạt F1-score 0.961 với K=2 nhưng không báo cáo thời gian thực thi, kết quả của tôi (F1-score 0.920, thời gian 154.19 giây với RandomizedSearchCV) cho thấy sự cân bằng tốt giữa hiệu quả và chi phí tính toán. Điều này chứng minh rằng việc kết hợp giảm chiều với tinh chỉnh siêu tham số giúp tối ưu hóa hiệu suất mà không làm tăng quá nhiều chi phí tính toán, vượt trội hơn so với cách tiếp cận chỉ tập trung vào tối ưu mô hình mà không giảm đặc trưng.

So sánh giữa các mô hình: Random Forest và XGBoost liên tục vượt trội trong cả ba kịch bản, với Random Forest tinh chỉnh bằng GridSearchCV đạt F1-score cao nhất (0.962). KNN là lựa chọn tốt cho tốc độ (thời gian thực thi 0.0140 giây sau giảm chiều), nhưng F1-score (0.920) thấp hơn Random Forest và XGBoost. LinearSVC, Logistic Regression, và MLP không phù hợp cho dữ liệu đã giảm chiều, tương tự như bài báo cáo trước khi MLP có F1-score thấp nhất (0.793). So với Sharmila et al. (2024), Decision Tree đạt F1-score 0.943, cao hơn SVM tuyến tính (0.708) và Naive Bayes (0.552), nhưng thấp hơn Random Forest (0.962) của tôi. Điều này nhấn mạnh rằng giảm đặc trưng kết hợp với tinh chỉnh giúp mô hình của tôi đạt hiệu quả cao hơn, đặc biệt với các mô hình ensemble như Random Forest và XGBoost, so với cách tiếp cận chỉ sử dụng dữ liệu gốc và tối ưu đơn giản như trong nghiên cứu của Sharmila et al.

Ứng dụng thực tế: Random Forest tinh chỉnh bằng GridSearchCV là lựa chọn tối ưu cho các bài toán yêu cầu độ chính xác cao, như phân loại y tế hoặc phát hiện gian lận. KNN phù hợp cho các ứng dụng thời gian thực, như hệ thống gợi ý trực tuyến. XGBoost cân bằng giữa độ chính xác và tốc độ, đặc biệt khi sử dụng RandomizedSearchCV. So với Sharmila et al. (2024), nơi Decision Tree và KNN được đề xuất cho phát hiện tấn công mạng IoT, kết quả của tôi cho thấy Random Forest và XGBoost, sau khi giảm chiều và tinh chỉnh, cung cấp hiệu quả cao hơn trong các kịch bản tương tự. Việc giảm chiều nên được áp dụng cẩn thận, với sự lựa chọn ngưỡng tương quan phù hợp, để tránh mất thông tin quan trọng như trong trường hợp LinearSVC, và kết hợp với tinh chỉnh để tối ưu hóa hiệu suất trên dữ liệu lớn.

V Kết luận và Hướng phát triển

1 Kết luận

Nghiên cứu này đã đánh giá hiệu suất của sáu mô hình học máy (LinearSVC, XGBoost, Logistic Regression, KNN, Random Forest, và MLP) trên ba kịch bản: dữ liệu chưa giảm chiều, dữ liệu đã giảm chiều bằng feature selection kết hợp với chọn ngưỡng tương quan, và mô hình tinh chỉnh bằng RandomizedSearchCV và GridSearchCV. Kết quả cho thấy việc giảm đặc trưng không chỉ duy trì mà còn cải thiện hiệu quả phân loại, với XGBoost đạt F1-score cao nhất (0.964) trên dữ liệu đã giảm chiều, đồng thời giảm đáng kể thời gian thực thi (từ 3.9874 giây xuống 1.7791 giây). Tinh chỉnh siêu tham số mang lại hiệu quả bổ sung, đặc biệt với Random Forest (F1-score 0.962) và XGBoost (0.956) khi sử dụng GridSearchCV, mặc dù chi phí tính toán tăng cao. KNN nổi bật về tốc độ (0.0140 giây sau giảm chiều), nhưng F1-score (0.920) thấp hơn các mô hình ensemble. So sánh với Sharmila et al. (2024), phương pháp của tôi vượt trội nhờ giảm đặc trưng, giúp cải thiện hiệu quả phân loại và hiệu suất tính toán, đặc biệt với các mô hình phức tạp như Random Forest và XGBoost. Nghiên cứu khẳng định rằng giảm chiều kết hợp với tinh chỉnh là chiến lược hiệu quả để tối ưu hóa hiệu suất trên dữ liệu lớn, với Random Forest và XGBoost là các lựa chọn tối ưu cho các bài toán phân loại đòi hỏi độ chính xác cao.

2 Hướng phát triển

Trong tương lai, tôi sẽ tập trung vào việc phát triển các phương pháp giảm chiều thích nghi, nhằm giảm thiểu mất mát thông tin quan trọng đối với các mô hình như LinearSVC và Logistic Regression. Ngoài ra, việc áp dụng các kỹ thuật học sâu (deep learning) có thể được nghiên cứu để khai thác các đặc trưng phức tạp hơn từ dữ liệu đã giảm chiều, nhằm nâng cao hiệu quả phân loại trên các tập dữ liệu không cân bằng. Tôi cũng dự định mở rộng nghiên cứu bằng cách tích hợp các phương pháp chọn siêu tham số tự động (ví dụ: Bayesian Optimization) để giảm chi phí tính toán của GridSearchCV, đồng thời thử nghiệm trên các tập dữ liệu lớn hơn và đa dạng hơn trong các ứng dụng thực tế như phát hiện gian lận hoặc phân loại y tế.

Tài liệu

[1] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[2] S. Bhunia and R. Nagapadma. RT-IoT2022. UCI Machine Learning Repository, 2023. `https://doi.org/10.24432/C5P338`.

[3] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16: 321–357, 2002.

[5] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

[6] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[7] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[9] D. M. Powers. Evaluation: From precision, recall and f-measure to roc. *Journal of Machine Learning Technologies*, 2(1):37–50, 2011.

[10] Sharmila B. S., Jayashree H. R., Pradeep R., and Vijayalakshmi R. Performance evaluation of parametric and non-parametric machine learning models using statistical analysis for rt-iot2022 dataset. *Journal of Scientific & Industrial Research*, 83 (8):864–872, 2024. doi: 10.56042/jsir.v83i8.7437.

[11] J. Smith et al. Deep learning for iot security. *IEEE Transactions on Network Security*, 2023.