ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN

ĐỒ ÁN TỐT NGHIỆP
CAPSTONE PROJECT

NGÀNH: KỸ THUẬT ĐIỀU KHIỂN VÀ TỰ ĐỘNG HOÁ

ĐỀ TÀI:

ỨNG DỤNG EDGE AI DỰ ĐOÁN BẢO TRÌ THIẾT BỊ IOT QUAN TRẮC MÔI TRƯỜNG

Người hướng dẫn: **TS. NGÔ ĐÌNH THANH**
ThS. NGUYỄN HUỲNH NHẬT THƯƠNG

Sinh viên thực hiện:

1. NGUYỄN TRƯỜNG BẢO NGÂN MSSV: 105200338

2. LÊ MINH SƠN MSSV: 105200343

LỚP: 20TDH2

Đà Nẵng, 6/2025

TÓM TẮT

Tên đề tài: Ứng dụng Edge AI dự đoán bảo trì thiết bị IoT quan trắc môi trường.

Sinh viên thực hiện:

Nguyễn Trường Bảo Ngân	Số thẻ SV: 105200338	Lớp: 20TDH2
Lê Minh Sơn	Số thẻ SV: 105200343	Lớp: 20TDH2

Trong bối cảnh biến đổi khí hậu và thiên tai ngày càng nghiêm trọng, việc đo mưa chính xác đóng vai trò quan trọng trong cảnh báo và quản lý tài nguyên nước. Tuy nhiên, các thiết bị đo mưa IoT hiện nay thường gặp lỗi tắc phễu, gây ảnh hưởng đến độ tin cậy của dữ liệu.

Đề tài này đề xuất giải pháp xây dựng hệ thống bảo trì dự đoán (Predictive Maintenance) dựa trên công nghệ Edge AI nhằm phát hiện lỗi tắc phễu cảm biến gầu lật trên thiết bị đo mưa. Hệ thống gồm hai tầng xử lý:

- Tầng Edge sử dụng các thuật toán học không giám sát (K-Means, Isolation Forest) để phát hiện bất thường tại thiết bị.
- Tầng Cloud ứng dụng mô hình học có giám sát (Random Forest, XGBoost, Decision Tree,….) để dự đoán chính xác lỗi dựa trên dữ liệu nhiều trạm đo.

Dữ liệu thực nghiệm thu thập từ hệ thống đo mưa Vrain được xử lý, trích xuất đặc trưng và áp dụng các kỹ thuật cân bằng dữ liệu nhằm cải thiện hiệu quả phân loại.

Phần mềm sử dụng: Toàn bộ quá trình xử lý, huấn luyện và đánh giá mô hình được thực hiện bằng Google Colab, với ngôn ngữ lập trình Python và thư viện như scikit-learn, pandas, matplotlib,...

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

TT	Họ tên sinh viên	Số thẻ SV	Lớp	Ngành
1	Nguyễn Trường Bảo Ngân	105200338	20TDH2	Kỹ thuật Điều khiển và Tự động hóa
2	Lê Minh Sơn	105200343	20TDH2	Kỹ thuật Điều khiển và Tự động hóa

1. *Tên đề tài đồ án:*

Ứng dụng Edge AI dự đoán bảo trì thiết bị IoT quan trắc môi trường.

2. *Đề tài thuộc diện: ☐Có ký kết thỏa thuận sở hữu trí tuệ đối với kết quả thực hiện*
3. *Các số liệu và dữ liệu ban đầu:*

……..…….....
……..
……..……..……

4. *Nội dung các phần thuyết minh và tính toán:*

a. Phần chung:

TT	Họ tên sinh viên	Nội dung
1	Nguyễn Trường Bảo Ngân	• Tìm hiểu tổng quan đề tài. • Nghiên cứu các lỗi thực tế của thiết bị IoT đo mưa. • Nghiên cứu các chiến lược bảo trì thiết bị IoT. • Xây dựng kiến trúc Hybrid Edge – Cloud AI.
2	Lê Minh Sơn	

b. Phần riêng:

TT	Họ tên sinh viên	Nội dung
1	Nguyễn Trường Bảo Ngân	• Nghiên cứu các ứng dụng của Edge AI trong hệ thống thiết bị IoT. • Tiền xử lý dữ liệu cho bài toán tại Edge.

		• Áp dụng các thuật toán học máy không giám sát cho dữ liệu thực tế. • Đánh giá và phân tích kết quả.
2	Lê Minh Sơn	• Nghiên cứu các ứng dụng của Cloud AI trong hệ thống thiết bị IoT. • Tiền xử lý dữ liệu cho bài toán tại Cloud. • Áp dụng các thuật toán học máy có giám sát cho dữ liệu thực tế. • Đánh giá và phân tích kết quả.

5. *Các bản vẽ, đồ thị (ghi rõ các loại và kích thước bản vẽ):*

a. Phần chung:

TT	Họ tên sinh viên	Nội dung

b. Phần riêng:

TT	Họ tên sinh viên	Nội dung

6. Họ tên người hướng dẫn:	*Phần/ Nội dung:*
TS. Ngô Đình Thanh	• Định hướng đề tài. • Đề xuất giải pháp. • Hướng dẫn thuyết minh báo cáo đề tài.
Ths. Nguyễn Huỳnh Nhật Thương	• Định hướng đề tài.

	• Đề xuất giải pháp.

7. *Ngày giao nhiệm vụ đồ án: 17/2/2025*
8. *Ngày hoàn thành đồ án: 17/6/2025*

Đà Nẵng, ngày tháng 6 năm 2025

Trưởng Bộ môn Tự động hóa	**Người hướng dẫn**
TS. Giáp Quang Huy	TS. Ngô Đình Thanh

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

PHIẾU KIỂM SOÁT TIẾN ĐỘ LÀM ĐỒ ÁN TỐT NGHIỆP

(Phiếu dành cho người hướng dẫn/sinh viên)

Họ tên sinh viên: Nguyễn Trường Bảo Ngân Số thẻ SV : 105200338

Lê Minh Sơn Số thẻ SV : 105200343

Tên đề tài ĐATN: Ứng dụng Edge AI dự đoán bảo trì thiết bị IoT quan trắc môi trường

Họ tên người HD: TS. Ngô Đình Thanh Đơn vị: Khoa Điện

Tuần	**Ngày**	**Khối lượng**		**GVHD ký tên**
		đã thực hiện (%)	**tiếp tục thực hiện (%)**	
1	17/02/2025	Nhận đề tài.	Tìm hiểu tổng quan đề tài.	
2	24/02/2025	Xác định tính cấp thiết, mục tiêu và phạm vi đối tượng, phương pháp nghiên cứu.	Tìm hiểu các yêu cầu của đề tài.	
3	03/03/2025	Xác định các lỗi xảy ra trong hệ thống IoT đo mưa thực tế.	Nghiên cứu lỗi tắc phễu hứng mưa.	
4	10/03/2025	Duyệt lần 1: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
5	17/03/2025	Nghiên cứu các chiến lược bảo trì trong hệ thống IoT.	Viết báo cáo thuyết minh Chương 1	
6	24/03/2025	Hoàn thành báo cáo thuyết minh Chương 1	Nghiên cứu các giải pháp phục vụ bảo trì dự đoán cho thiết bị IoT.	
7	31/03/2025	Xây dựng giải pháp với kiến trúc Hybrid Edge – Cloud AI.	Viết báo cáo thuyết minh Chương 2	
8	07/04/2025	Duyệt lần 2: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		

9	14/04/2025	Tiền xử lý dữ liệu thực tế từ công ty Watec.	Trích xuất, đánh giá, lựa chọn đặc trưng phù hợp cho dữ liệu tại Edge và Cloud.	
10	21/04/2025	Lựa chon và nghiên cứu các mô hình học máy dựa trên các công trình nghiên cứu đã có.	Áp dụng các mô hình học máy cho bài toán tại Edge và Cloud.	
11	28/4/2025	Kiểm nghiệm và phân tích đánh giá kết quả của các mô hình trên dữ liệu thực tế.	Cải tiến kết quả thông qua việc hiệu chỉnh thông số mô hình và xử lý lại dữ liệu thực tế.	
12	05/05/2025	Duyệt lần 3: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
13	12/05/2025	Đưa ra mô hình với thông số, đặc trưng, phương pháp xử lý phù hợp nhất với dữ liệu thực.	Kiểm nghiệm và so sánh hai mô hình tại Edge và Cloud trên cùng một tập dữ liệu thưc tế mới.	
14	19/05/2025	Hoàn thiện giải pháp	Viết báo cáo thuyết minh Chương 3	
15	26/05/2025	Hoàn thiện báo cáo Chương 3	Tổng hợp báo cáo và chỉnh sửa.	

LỜI NÓI ĐẦU VÀ LỜI CẢM ƠN

Trong suốt quá trình thực hiện đồ án tốt nghiệp, nhóm sinh viên chúng em đã nhận được rất nhiều sự giúp đỡ, đóng góp ý kiến và chỉ bảo tận tình từ quý thầy cô và bạn bè.

Chúng em xin bày tỏ lòng biết ơn sâu sắc tới người hướng dẫn trực tiếp của nhóm, Tiến sĩ Ngô Đình Thanh, vì sự chỉ dẫn tận tình, hỗ trợ liên tục và những lời động viên quý báu trong suốt quá trình thực hiện đề tài. Kiến thức chuyên môn cùng những góp ý sâu sắc của thầy đã đóng vai trò quan trọng trong việc hoàn thành đồ án này.

Nhóm cũng xin chân thành cảm ơn Thạc sĩ Nguyễn Huỳnh Nhật Thương đã đồng hành, đóng góp ý kiến và hỗ trợ nhiệt tình, góp phần giúp nhóm hoàn thiện tốt đề tài.

Bên cạnh đó, chúng em xin gửi lời cảm ơn đến quý thầy cô trường Đại học Bách khoa - Đại học Đà Nẵng nói chung, và đặc biệt là các thầy cô Bộ môn Tự động hóa nói riêng, những người đã cung cấp cho chúng em nền tảng kiến thức chuyên môn vững chắc, tạo điều kiện thuận lợi trong suốt quá trình học tập và nghiên cứu.

Cuối cùng, nhóm xin gửi lời tri ân sâu sắc tới gia đình và bạn bè, những người đã luôn đồng hành, động viên và tạo điều kiện tốt nhất để chúng em hoàn thành quá trình học tập cũng như đồ án tốt nghiệp này.

Do điều kiện thời gian và sự hiểu biết còn hạn chế dưới góc nhìn của sinh viên, trong quá trình thực hiện và báo cáo đề tài chắc chắn không tránh khỏi những thiếu sót. Chúng em rất mong nhận được những góp ý quý báu từ quý thầy cô để có thể bổ sung, hoàn thiện hơn kiến thức, phục vụ tốt hơn cho công việc thực tế sau này.

Nhóm xin chân thành cảm ơn!

LỜI CAM ĐOAN LIÊM CHÍNH HỌC THUẬT

Chúng em xin cam đoan rằng toàn bộ nội dung của đề tài: "Ứng dụng Edge AI dự đoán bảo trì thiết bị IoT quan trắc môi trường" là kết quả của quá trình nghiên cứu và thực hiện độc lập của nhóm sinh viên, dưới sự hướng dẫn của giảng viên hướng dẫn thầy TS. Ngô Đình Thanh và sự hỗ trợ từ các tài liệu tham khảo, bài báo, giáo trình có liên quan.

Trong quá trình thực hiện đề tài, nhóm có tham khảo các tài liệu, hình ảnh, dữ liệu từ nhiều nguồn khác nhau, và tất cả đều được trích dẫn rõ ràng tại phần tài liệu tham khảo. Nhóm không sao chép nội dung một cách nguyên văn hoặc vi phạm bản quyền từ bất kỳ nguồn nào.

Chúng em xin hoàn toàn chịu trách nhiệm về tính trung thực, bản quyền và tính chính xác của toàn bộ nội dung trong đề tài. Nếu phát hiện có bất kỳ sai phạm nào liên quan đến bản quyền, học thuật hoặc đạo đức nghiên cứu, chúng em xin hoàn toàn chịu trách nhiệm trước nhà trường và các quy định hiện hành.

Đà Nẵng, ngày 17 tháng 06 năm 2025

Nhóm sinh viên thực hiện

Nguyễn Trường Bảo Ngân – Lê Minh Sơn

MỤC LỤC
DANH SÁCH CÁC BẢNG, HÌNH VẼ

DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT

CHỮ VIẾT TẮT:

IoT	Internet of Things: Internet kết nối vạn vật
AI	Artificial Intelligence: Trí tuệ nhân tạo
Edge AI	Edge Artificial Intelligence: AI hoạt động tại thiết bị biên
Cloud AI	Cloud Artificial Intelligence: AI hoạt động trên nền tảng đám mây
CM	Corrective Maintenance: Bảo trì khắc phục
ICM	Immediate Corrective Maintenance: Bảo trì khắc phục tức thời
DCM	Deferred Corrective Maintenance: Bảo trì khắc phục trì hoãn
PM	Preventive Maintenance: Bảo trì phòng ngừa
CBM	Condition – based Maintenance: Bảo trì dựa trên tình trạng
PDM	Predictive Maintenance: Bảo trì dự đoán
RUL	Remaining Useful Life: Tuổi thọ còn lại của thiết bị
SMOTE	Synthetic Minority Over-sampling Technique: Kỹ thuật tăng cường mẫu thiểu số
IF	Isolation Forest: Thuật toán Rừng cô lập
LOF	Local Outlier Factor: Thuật toán Hệ số ngoại lai cục bộ

MỞ ĐẦU

1. Tính cấp thiết

Trong bối cảnh biến đổi khí hậu diễn biến ngày càng phức tạp, các hiện tượng thời tiết cực đoan như mưa lớn bất thường, lũ quét, ngập úng xảy ra với tần suất ngày càng cao, gây thiệt hại nghiêm trọng về người và tài sản. Vì vậy, việc quan trắc lượng mưa theo thời gian thực đóng vai trò then chốt trong công tác cảnh báo sớm thiên tai, quản lý tài nguyên nước và quy hoạch hạ tầng đô thị.

Để phục vụ mục tiêu này, hàng nghìn thiết bị IoT đo mưa đã được triển khai trên khắp lãnh thổ Việt Nam, đặc biệt tại các vùng có nguy cơ thiên tai cao. Tuy nhiên, khi quy mô triển khai ngày càng mở rộng, bài toán bảo trì và đảm bảo vận hành ổn định cho toàn bộ hệ thống trở thành một thách thức lớn. Với số lượng thiết bị lớn, phân bố rải rác ở nhiều địa hình khác nhau – từ vùng núi, nông thôn đến thành thị – thì việc bảo trì định kỳ hay phát hiện thủ công các hư hỏng sẽ rất tốn kém nhân lực, chi phí di chuyển, và thời gian xử lý. Việc cử kỹ thuật viên đi kiểm tra từng trạm đo theo lịch cố định không chỉ làm tăng gánh nặng vận hành, mà còn không đảm bảo được khả năng phát hiện và xử lý lỗi kịp thời.

Đứng trước những thách thức nêu trên, bảo trì dự đoán (Predictive Maintenance) trở thành một hướng tiếp cận thiết yếu, cho phép phát hiện sớm các dấu hiệu hư hỏng dựa trên phân tích dữ liệu thực tế. Trong các hệ thống IoT truyền thống, mô hình trí tuệ nhân tạo thường được triển khai trên nền tảng điện toán đám mây (Cloud AI), nơi toàn bộ dữ liệu được gửi về máy chủ để xử lý. Tuy nhiên, khi số lượng thiết bị lên đến hàng nghìn và phân bố rộng khắp, lượng dữ liệu truyền về tăng đột biến, dẫn đến nguy cơ quá tải xử lý ở phía server, làm giảm hiệu suất hệ thống và tăng độ trễ phản hồi.

Trước thực trạng đó, đề tài hướng đến giải pháp kết hợp giữa AI truyền thống và Edge AI, trong đó Edge AI – trí tuệ nhân tạo chạy trực tiếp trên thiết bị biên – sẽ đảm nhiệm một phần công việc dự đoán và phát hiện lỗi ngay tại nguồn dữ liệu. Cách tiếp cận này giúp hệ thống phân tán khối lượng xử lý, giảm phụ thuộc vào kết nối mạng, và đặc biệt là giảm tải cho máy chủ trung tâm. Các thiết bị Edge không chỉ phát hiện được các bất thường tại chỗ mà còn có thể đưa ra cảnh báo sớm mà không cần đợi phản hồi từ cloud.

Giải pháp này vừa nâng cao tính chủ động và linh hoạt trong giám sát hệ thống quan trắc, vừa đảm bảo khả năng mở rộng, tối ưu tài nguyên tính toán, và tăng độ tin cậy của dữ liệu đo mưa – đầu vào quan trọng cho các hệ thống cảnh báo và quản lý thiên tai.

Vì những lý do trên, nhóm sinh viên đã thực hiện đề tài: "Ứng dụng Edge AI để dự đoán bảo trì thiết bị IoT đo mưa" nhằm phát triển một giải pháp thông minh, bền vững và hiệu quả cho hệ thống quan trắc môi trường có quy mô lớn trong thực tế.

2. Mục tiêu nghiên cứu

Mục tiêu nghiên cứu của đồ án là ứng dụng được khả năng của AI đặc biệt là Edge AI để phát hiện sớm các lỗi thường xuyên xảy ra trên thiết bị quan trắc IoT.

Để đạt được mục tiêu này, nghiên cứu tập trung vào một số nội dung chính như sau:

1. Phân tích hoạt động của thiết bị IoT đo mưa và các lỗi kỹ thuật thường gặp trong quá trình vận hành thực tế.
2. Thu thập và xử lý dữ liệu lượng mưa từ các trạm quan trắc
3. Xây dựng và huấn luyện mô hình học máy để phát hiện sớm lỗi tắc phễu cảm biến.
4. Thử nghiệm mô hình trên tập dữ liệu thu thập từ các trạm đo thực tế và đánh giá độ chính xác, tốc độ phản hồi cũng như khả năng hoạt động ổn định của mô hình.

Thông qua việc thực hiện các mục tiêu trên, đề tài nhằm chứng minh tiềm năng của giải pháp Edge AI trong việc tăng cường hiệu quả giám sát, tối ưu chi phí vận hành và khả năng mở rộng hệ thống IoT đo mưa trong thực tế. Ngoài ra, đề tài cũng đề xuất hướng phát triển tiếp theo nhằm hoàn thiện và mở rộng khả năng ứng dụng công nghệ này.

3. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu trong đề tài là thiết bị IoT đo mưa với cảm biến dạng gầu lật (Tipping-Bucket Rain Gauge), cùng với bộ dữ liệu đo lường được truyền về từ các trạm quan trắc thực tế.

Phạm vi nghiên cứu: Nghiên cứu này tập trung vào hệ thống quan trắc lượng mưa với các thành phần chính sau:

- Dữ liệu đầu vào: thông tin lượng mưa theo thời gian.
- Thông tin đầu ra: thời điểm và xác suất xảy ra lỗi tắc phễu trên thiết bị.

4. Phương pháp nghiên cứu

Phương pháp luận: Dựa trên cơ sở các kết quả của các công trình nghiên cứu trước, đề xuất kiến trúc Hybrid dự đoán hiện tượng tắc phễu trong trên cảm biến IoT đo mưa, đưa ra những thuật toán phục vụ bảo trì dự đoán trong hệ thống cảm biến IoT đo mưa.

Phương pháp đánh giá bằng thực nghiệm: Thực nghiệm các thuật toán trên dữ liệu thật.

5. Cấu trúc đồ án

Mở đầu

Chương 1: Tổng quan vấn đề nghiên cứu

Chương 2: Xây dựng giải pháp bảo trì dự đoán

Chương 3: Thực nghiệm và đánh giá kết quả

Kết luận

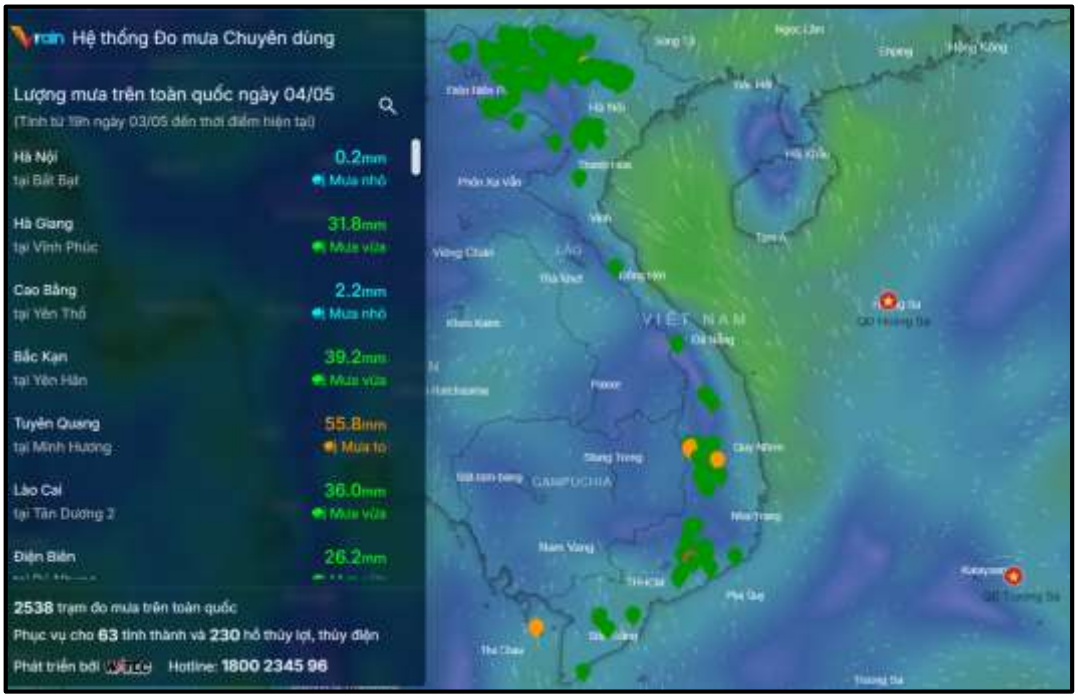
CHƯƠNG 1: TỔNG QUAN VẤN ĐỀ NGHIÊN CỨU

1.1. Hệ thống thiết bị IoT đo mưa tại Việt Nam

Hiện nay, hệ thống đo mưa tự động lớn nhất tại Việt Nam là Vrain, một nền tảng sử dụng thiết bị IoT đo mưa Vrain, do công ty Watec phát triển và ứng dụng rộng rãi trong các mạng lưới quan trắc môi trường [1]. Tính đến thời điểm hiện tại, Vrain đang có khoảng 2538 trạm đo phân bố khắp cả nước, tạo thành một mạng lưới giám sát thống nhất, đồng bộ, liên tục, hỗ trợ hiệu quả cho công tác dự báo, cảnh báo mưa lũ và điều hành hồ chứa.

Các trạm sử dụng thiết bị Datalogger được thiết kế chuyên biệt bởi đơn vị Việt Nam cho ứng dụng đo mưa mà không xét đến nhiều yếu tố khác (gió, độ ẩm…). Nhờ vậy, thiết bị trở nên gọn nhẹ, giá thành hợp lí và mức tiêu thụ năng lượng thấp [1]. Chỉ với 2 pin dự phòng là thiết bị có thể hoạt động ổn định từ 45-60 ngày trong điều kiện mất điện [2]. Ngoài ra, để nâng cao tính linh hoạt và khả năng thích nghi với nhiều điều kiện vận hành khác nhau, trạm đo mưa IoT được thiết kế sử dụng điện lưới ở những vị trí gần nguồn và điện mặt trời tại những điểm xa xôi, những khu vực có hạ tầng điện lưới còn hạn chế [3].

Thông tin và dữ liệu mưa từ các trạm Vrain được công khai sử dụng miễn phí qua nền tảng www.vrain,vn và ứng dụng "Vrain by Watec" trên điện thoại, giúp người dân và các nhà quản lý chỉ đạo phòng chống thiên tai truy cập và theo dõi dữ liệu mưa theo thời gian thực tại từng vị trí cụ thể, nhận cảnh báo khi mưa lớn vượt ngưỡng, tăng cường năng lực chủ động ứng phó thiên tai [1].



Hình 1.1 Giao diện trực quan hệ thống đo mưa tại Vrain trên nền tảng trực tuyến

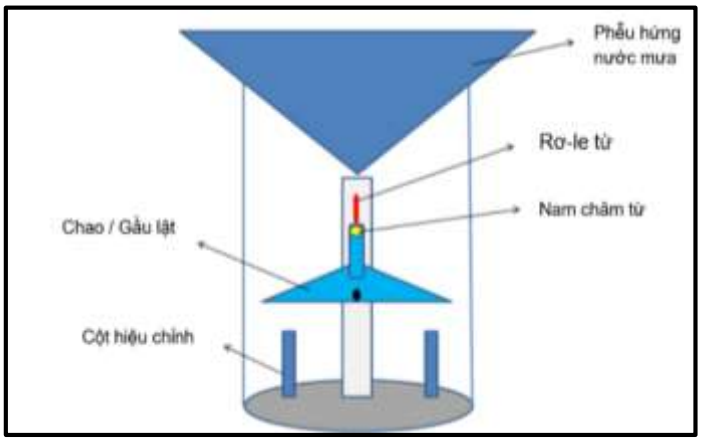
Thiết bị đo mưa IoT Vrain có dạng gầu lật (Tipping-Bucket Rain Gauge), được tích hợp công nghệ IoT và vi xử lý nhúng, cho phép thu thập và truyền dữ liệu lượng mưa theo thời gian thực về máy chủ và quản lý bằng phần mềm quản lý tập trung.

Hệ thống thiết bị bao gồm:

- Cảm biến đo mưa kiểu gầu lật: đo mưa và gửi về giá trị dòng (4-20 mA).
- Bộ thu thập, xử lý và truyền tín hiệu: nhận dữ liệu từ cảm biến, xử lí, đóng gói và gửi về máy chủ trung tâm thông qua mạng di động (4G/3G/2G/SMS).
- Nguồn cấp hỗn hợp: kết hợp điện lưới và pin năng lượng mặt trời, và pin dự phòng.

Hình 1.2 Thiết bị IoT đo mưa Vrain

Nguyên lý hoạt động: bên ngoài cảm biến là một thùng hứng mưa, nước mưa qua miệng thùng chảy vào phễu dẫn xuống gầu lật. Khi đủ lượng nước, gầu lật sẽ lật 1 lần. Trên gầu lật có gắn một thanh kim loại đính nam châm từ. Mỗi lần lật nam châm từ sẽ đi qua rơ-le từ một lần phát tín hiệu thông qua dây tín hiệu về tủ DataLogger, nơi đảm nhận việc thu thập, xử lí, đóng gói và truyền tín hiệu.

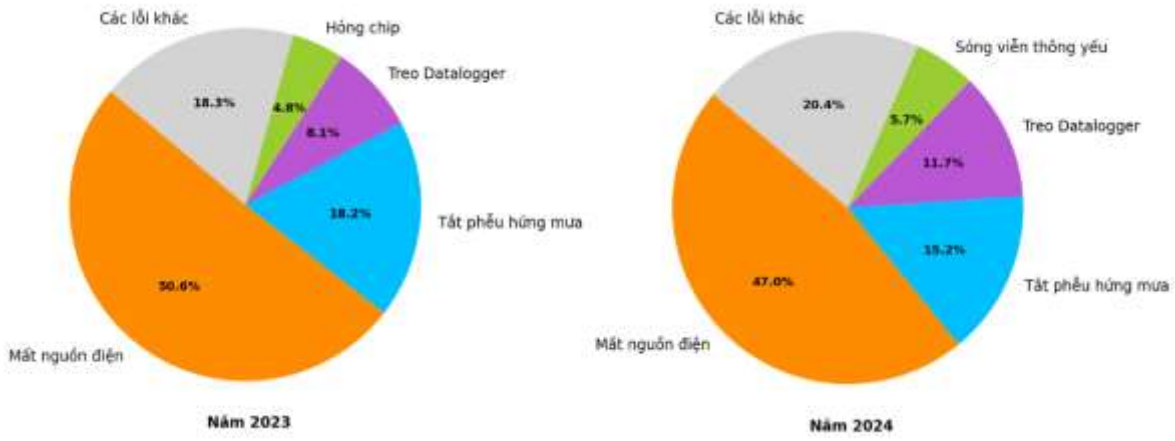


Hình 1.3 Cấu tạo cảm biến đo mưa dạng gầu lật

1.2. Các vấn đề thường gặp trong thiết bị IoT đo mưa

Thiết bị IoT đo mưa hoạt động ngoài trời thường gặp nhiều vấn đề kỹ thuật trong suốt quá trình vận hành do phải hoạt động liên tục trong các điều kiện thời tiết khắc nghiệt. Ông Văn Phú Chính, Chủ tịch Hội đồng Quản trị Công ty Cổ phần Tư vấn và Phát triển Kỹ thuật Tài nguyên Nước Watec, cho biết hiện nay hệ thống Vrain phải xử lý khoảng 10 – 15 sự cố trên toàn hệ thống [1]. Theo báo cáo số 42 và 47 của phòng giám sát vận hành (BC-GSVN), số lỗi xảy ra trên toàn bộ hệ thống thiết bị IoT đo mưa các năm 2023 – 2024 lần lượt là 2006 và 2341 sự cố. Trong đó, số lượng sự cố xảy ra ở năm 2024 đã tăng hơn 300 sự cố, khoảng 16.7% so với năm trước. Chính điều này đã gây ảnh hưởng nghiêm trọng đến quá trình thu thập dữ liệu liên tục phục vụ cho công tác dự báo và phòng chống thiên tai.

Cũng dựa vào hai báo cáo trên, ta có được các biểu đồ thống kê nguyên nhân lỗi trong hệ thống đo mưa trong hai năm 2023 – 2024. Trong đó, các biểu đồ tập trung thể hiện các lỗi thường xuyên xảy ra nhất tại các thiết bị đo mưa IoT. Đối với các lỗi ít xảy ra hơn hoặc xảy ra với số lần không đáng kể được liệt kê vào hạng mục "Các lỗi khác".



Hình 1.4 Biểu đồ thống kê nguyên nhân lỗi trong hệ thống đo mưa trong hai năm 2023-2024

Dựa vào hình 1.4 ta thấy được lỗi thường xuyên xảy ra ở thiết bị là vấn đề mất nguồn. Vấn đề này được chia làm hai phần là sự cố mất nguồn từ pin mặt trời và sự cố mất nguồn từ điện lưới. Đối với trường hợp sự cố đến từ nguồn điện mặt trời, nguyên nhân chủ yếu đến từ các yếu tố thời tiết như mây mù hoặc những điều kiện tự nhiên bất lợi. Trong khi đó, sự cố từ nguồn điện lưới thường do các vấn đề kỹ thuật từ phía các đơn vị cung cấp điện, ngoài tầm kiểm soát và khả năng nhận biết của hệ thống. Do thiếu dữ liệu và thông tin cảnh báo, các sự cố này thường khó dự đoán và chỉ được phát hiện sau khi đã xảy ra.

Một trong những lỗi phổ biến đứng thứ hai trong danh sách là tình trạng tắc phễu hứng mưa tại các thiết bị. Các trạm đo mưa thường đặt ở ngoài trời thường xuyên chịu

ảnh hưởng bởi các yếu tố môi trường gió mạnh, mưa lớn mang theo lá cây, bụi bẩn hoặc côn trùng. Theo thời gian, những tác nhân này tích tụ dần gây ra hiện tượng tắc nghẽn tại miệng phễu, gây cản trở dòng chảy nước mưa vào cơ cấu gầu lật, từ đó dẫn đến dữ liệu ghi nhận bị sai lệch hoặc mất hoàn toàn.

1.3. Bảo trì và các chiến lược

Thuật ngữ "bảo trì" được định nghĩa:

"Tổng hợp tất cả các nhiệm vụ kỹ thuật và quản lý nhằm duy trì hoặc khôi phục một thiết bị về trạng thái mà nó có thể thực hiện được chức năng yêu cầu" [4].

Mục tiêu của bảo trì là nhằm:

- Ngăn ngừa sự cố/hỏng hóc,
- Giảm thời gian ngừng hoạt động,
- Kéo dài tuổi thọ thiết bị,
- Cải thiện độ an toàn,
- Giảm tiêu thụ năng lượng,
- Phòng ngừa hoặc giảm thiểu tác động gây ô nhiễm môi trường,

từ đó nâng cao hiệu suất hệ thống và giảm chi phí tổng thể của hệ thống [4].

Hiện nay, có nhiều cách phân loại chiến lược bảo trì. Theo chuẩn quốc tế ISO 14224:2016 [5], chuẩn châu Âu EN 13306:2017 [6], chuẩn quân đội Mỹ MIL-HDBK-338B [7] và chuẩn Pháp AFNOR-2001 [8], bảo trì đều được chia thành 2 chiến lược chính:

- Bảo trì khắc phục (Corrective Maintenance – CM).
- Bảo trì phòng ngừa (Preventive Maintenance – PM).

1.3.1. Bảo trì khắc phục

Bảo trì khắc phục (Corrective Maintenance – CM) còn được gọi bảo trì phản ứng hay bảo trì sửa chữa là chiến lược bảo trì được thực hiện sau khi sự cố hoặc hỏng hóc đã xảy ra nhằm khôi phục thiết bị về trạng thái hoạt động bình thường theo một điều kiện xác định. Bảo trì khắc phục có thể thực hiện ngay lập thức sau khi phát hiện lỗi hoặc hoãn lại cho đến khi có điều kiện thuận lợi. Vì vậy, chiến lược này phù hợp cao trong các trường hợp sự cố hiếm gặp hoặc thiết bị không yêu cầu độ sẵn sàng cao. Tuy nhiên, ở các trường hợp không cho phép hoặc thời gian dừng hạn chế, chiến lược này có thể làm tăng chi phí vận hành và sửa chữa.

Phân loại dựa theo các chuẩn [5,6], bảo trì khắc phục thường được chia thành hai dạng chính:

- ***Bảo trì khắc phục tức thời*** (Immediate Corrective Maintenance – ICM): Là tập hợp các hoạt động sửa chữa được tiến hành ngay lập tức sau khi phát

hiện lỗi. Chiến lược này thường áp dụng cho các hệ thống đòi hỏi tính sẵn sàng cao hoặc có ảnh hưởng lớn nếu ngừng hoạt động. ICM yêu cầu phản ứng nhanh và thường kéo theo chi phí cao do sử dụng nguồn lực khẩn cấp, chi phí bảo quản tồn kho và ảnh hưởng đến lịch trình vận hành.

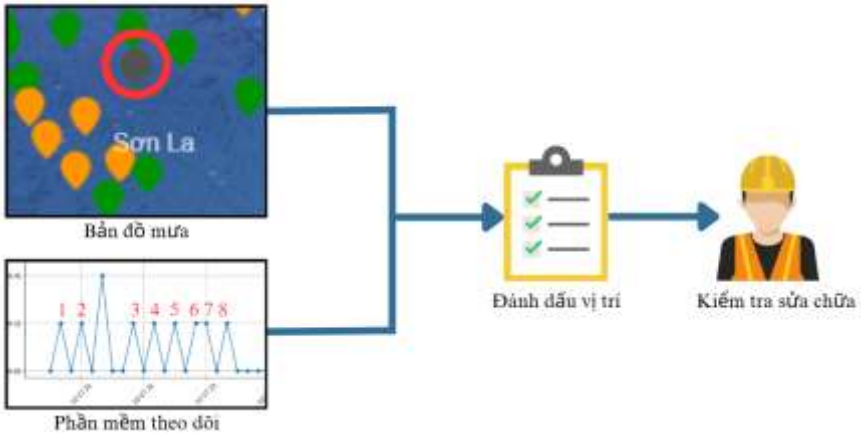
- ***Bảo trì khắc phục trì hoãn*** (Deferred Corrective Maintenance – DCM): Là tập hợp các hoạt động sửa chữa được hoãn lại đến thời điểm thuận lợi hơn, chẳng hạn khi có đủ nguồn lực, linh kiện hoặc vào đúng kỳ bảo trì định kỳ. Chiến lược này phù hợp với các phần tử không quan trọng hoặc khi lỗi không gây ảnh hưởng tức thì đến hiệu suất hệ thống. Tuy nhiên, trì hoãn quá lâu có thể làm gia tăng mức độ hư hại hoặc chi phí sửa chữa sau này.

Bảo trì khắc phục có thể bao gồm một hoặc nhiều bước theo quy trình sau [4]:

Hình 1.5 Quy trình thực hiện bảo trì khắc phục

Hiện nay, tại công ty Watec đã áp dụng quy trình này để phát hiện lỗi. Quy trình được thực hiện dựa vào các dữ liệu của thiết bị gửi về thiết bị. Theo kinh nghiệm vận hành của kỹ sư vận hành, một số lỗi đã được phát hiện như sau:

Lỗi tắc phễu hứng mưa:



Hình 1.6 Quy trình thực hiện bảo trì khắc phục tại công ty Watec

Lỗi tắc phễu hứng mưa thường được kỹ sư phát hiện thông qua hai công cụ chính: bản đồ mưa và phần mềm theo dõi dữ liệu trạm.

- Đối với bản đồ mưa, trạm có dấu hiệu lỗi sẽ được nhận biết khi liên tục gửi về tín hiệu không mưa, trong khi các trạm xung quanh vẫn ghi nhận có mưa trong cùng một khu vực. Hiện tượng này cho thấy khả năng cao phễu thu nước mưa đã bị tắc nghẽn.

- Đối với phần mềm theo dõi, kỹ sư thường đặt ngưỡng đếm lượng mưa bất thường (thường là từ 12 đến 13) nhằm phân biệt giữa những đợt mưa ngâu kéo dài. Nếu dữ liệu gửi về từ trạm có đặc điểm lặp lại hoặc không biến động trong một khoảng thời gian nhất định (thường trong ngày), thì trạm này sẽ bị đánh dấu là nghi ngờ có lỗi.

Sau khi đánh dấu vị trí, các kỹ sư sẽ thực hiện các bước kiểm tra, xử lý tiếp theo như mô tả trong quy trình Hình 1.6.

Việc áp dụng hai biện pháp song song này là hoàn toàn cần thiết bởi lỗi tắc phễu hứng mưa có biểu hiện không cố định. Dấu hiệu bị tắc có thể dài ngắn khác nhau trước khi xảy ra hiện tượng tắc phễu hoàn toàn. Lỗi phụ thuộc vào lượng mưa và lượng chất rắn mang theo trong nước. Trường hợp chỉ sử dụng lượng mưa ở một trạm rất có thể nhầm lẫn với hiện tượng mưa ngâu kéo dài.

Lưu ý rằng việc phát hiện lỗi tắc phễu trong trường hợp hiện tại thường diễn ra ở gần giữa hoặc cuối giai đoạn thu thập dữ liệu, tức là khi hiện tượng tắc nghẽn đã không còn ở giai đoạn bắt đầu biểu hiện mà đã gây ảnh hưởng rõ rệt đến kết quả đo mưa tại trạm.

Lỗi kẹt chao lật/ rơi nam châm từ/ hỏng rơ-le từ

Nhóm lỗi này được phát hiện khi dữ liệu lượng mưa ghi nhận tại trạm liên tục bằng 0 trong nhiều giờ, trong khi các trạm lân cận vẫn ghi nhận có mưa trong cùng thời gian và khu vực. Đây là dấu hiệu bất thường cho thấy khả năng cao phễu hứng mưa bị tắc nghẽn hoàn toàn ở giai đoạn cuối hoặc cảm biến gặp lỗi cơ học. Để xác định nguyên nhân cụ thể kỹ sư vận hành sẽ:

- Xác định trạm nghi ngờ lỗi bằng cách phân tích bản đồ mưa theo thời gian thực để nhận diện khu vực bất thường.
- Đối chiếu lịch sử lỗi của cảm biến tại vị trí được xác định để xem xét khả năng mắc lỗi nào cao nhất. Từng loại cảm biến khác nhau sẽ có tỷ lệ lỗi khác nhau.
  Ví dụ: đối với cảm biến Davis, tỷ lệ lỗi do hỏng rơ-le từ được ghi nhận lên đến 26% – cao hơn so các loại lỗi khác.
- Các bước tiếp theo thực hiện như quy trình Hình 1.6.

Lỗi sóng viễn thông yếu

Lỗi sóng viễn thông yếu được phát hiện khi hệ thống ghi nhận cường độ tín hiệu thấp hơn –108 dBm. Trong trường hợp này, thiết bị vẫn hoạt động bình thường nhưng gặp khó khăn trong việc gửi dữ liệu về máy chủ. Khi lỗi này xảy ra, thiết bị có thể tự động chuyển giảm dần chế độ kết nối, từ 4G → 3G → 2G → SMS để đáp ứng với mức tín hiệu của thực tế

Không giống như các lỗi cơ học, lỗi viễn thông không yêu cầu sửa chữa và xử lý theo quy trình vật lý ngay tại thiết bị. Thay vào đó, thiết bị sẽ tự động lưu trữ toàn bộ dữ liệu chưa gửi, và tự động gửi lại khi kết nối được khôi phục.

Lỗi số liệu bất thường

Lỗi số liệu bất thường thường được phát hiện khi dữ liệu lượng mưa ghi nhận tại trạm có những biểu hiện không hợp lý so với thực tế hoặc dữ liệu của các trạm xung quanh. Nhóm lỗi này không nhất thiết liên quan đến hỏng hóc phần cứng, nhưng ảnh hưởng đến độ tin cậy của số liệu đo. Một số biểu hiện thường gặp bao gồm:

- Trạm báo có mưa trong khi thực tế không có mưa xảy ra tại khu vực hoặc các trạm lân cận không ghi nhận mưa.
- Trạm báo sai số liệu đột biến, ví dụ: từ 10mm nhảy lên 100mm trong thời gian rất ngắn mà không có dấu hiệu mưa lớn thật sự.

Việc phát hiện nhóm lỗi này chủ yếu dựa vào phân tích dữ liệu từ phần mềm, kết hợp với kinh nghiệm thực tiễn của kỹ sư vận hành sau thời gian dài theo dõi hệ thống. Không có ngưỡng cố định cho từng trường hợp, mà đòi hỏi phán đoán linh hoạt dựa trên bối cảnh vận hành và lịch sử dữ liệu của từng trạm.

Lỗi hiển thị sai thời gian

Việc đồng bộ thời gian bị sai ở Datalogger gây ra mất dữ liệu tại thời điểm đó, lỗi này được xác định trên phần mềm và được sửa chữa khi thiết bị không thể đồng bộ lại thời gian.

1.3.2. Bảo trì phòng ngừa

Bảo trì phòng ngừa (Preventive Maintenance – PM) là tập hợp các hoạt động bảo trì định kỳ, theo kế hoạch trước nhằm giảm nguy cơ xảy ra sự cố hoặc suy giảm chức năng của thiết bị. Đối với các dạng bảo trì phòng ngừa truyền thống, bảo trì phòng ngừa thường dựa trên kinh nghiệm hoặc lịch sử hoạt động của thiết bị, thay vì theo dõi tình trạng thực tế, nên các công việc bảo trì phòng ngừa được thực hiện đều đặn theo lịch trình đã tính toán. Điều có thể dẫn đến hoạt động bảo trì rơi vào những điểm thời gian không cần thiết hoặc không tối ưu. Tuy nhiên, với sự phát triển của công nghệ thu thập dữ liệu, đã có sự xuất hiện của các hình thức bảo trì tiên tiến dựa trên tình trạng thiết bị.

Phân loại dựa theo các chuẩn [5,6], bảo trì phòng ngừa thường được chia thành hai dạng chính:

- ***Bảo trì dựa theo lịch trình*** (Predetermined/Systematic Maintenance).
  - ***Bảo trì dựa trên tuổi thọ*** (Age-based PM): Được thực hiện vào những thời điểm đã xác định dựa trên "tuổi thọ" của thiết bị. Ở đây, tuổi thọ có thể đo bằng thời gian hoạt động, số ki-lô-mét đã đi, số lần sử dụng, hoặc các khái niệm tương tự.
  - ***Bảo trì dựa trên thời gian*** (Clock-based PM): Được thực hiện tại những thời điểm định trước theo lịch.
- ***Bảo trì dựa trên tình trạng*** (Condition-based Maintenance – CBM): Dựa trên việc đo lường một hay nhiều biến trạng thái của thiết bị. Các hoạt động bảo trì sẽ

được thực hiện khi một biến điều kiện (hoặc hàm số của nhiều biến điều kiện) tiệm cận hoặc vượt qua một ngưỡng xác định (sử dụng mô hình toán).

Hiện nay tại Watec đã áp dụng chiến lược bảo trì dựa trên tuổi thọ (Age-based PM) cho các linh kiện như pin và ắc quy tại các điểm trạm đo mưa với tần suất 3 - 5 năm / lần.

1.3.3. Bảo trì dự đoán

Bảo trì dự đoán (Predictive Maintenance - PdM) là một phương pháp tiên tiến phát triển từ chiến lược bảo trì dựa trên tình trạng (CBM). Sử dụng các công cụ và kỹ thuật phân tích dữ liệu lặp lại hoặc các đặc tính đã biết, người quản lý có thể dự đoán, theo dõi và phát hiện những bất thường hoặc sai số [9]. Bảo trì dự đoán khác với các phương pháp bảo trì phòng ngừa truyền thống dựa trên thống kê trung bình hay mô hình toán. Thay vào đó, chiến lược này sử dụng dữ liệu thực để xác định thời điểm bảo trì, giảm rủi ro và tăng độ chính xác. Mục tiêu của việc bảo trì dự đoán là tối thiểu hóa sự cố trong khi vẫn đảm bảo về mặt kinh tế.

Hình 1.7 Hàm mục tiêu của chiến lược bảo trì

Những lợi ích chính của bảo trì dự đoán gồm [9][10]:

- Cung cấp dữ liệu thực về tình trạng vật lý của thiết bị: các dữ liệu thu thập từ cảm biến đưa về phản ánh tình trạng thiết bị, hỗ trợ quyết định cho các kỹ sư vận hành và bảo trì.
- Giảm thời gian chết (downtime) : Bằng cách phát hiện các vấn đề tiềm ẩn trước khi chúng dẫn đến hỏng hóc hoàn toàn, từ đó giảm thiểu thời gian chết ngoài kế hoạch.
- Tiết kiệm chi phí: Tối ưu hóa việc sử dụng tài nguyên bằng cách tập trung bảo trì chỉ khi cần thiết thay vì kiểm tra thường xuyên tất cả thiết bị, từ đó giảm chi phí nhân công, phụ tùng thay thế và mức tiêu thụ năng lượng.

- Tối ưu hóa tuổi thọ thiết bị: thường xuyên và đúng hạn thực hiện bảo trì dựa trên dự đoán có thể ngăn ngừa sự cố, kéo dài tuổi thọ thiết bị.

Theo bài nghiên cứu [13], bảo trì dự đoán được chia ra làm 3 phương pháp phân tích chính:

- Phân tích hồi quy: Dự đoán giá trị như RUL dựa vào nhiều yếu tố.
- Phân tích tình trạng: Phân loại tình trạng thiết bị (bình thường, cảnh báo, hỏng)
- Phân tích chuỗi thời gian: Phân tích dự đoán xu hướng của một dữ liệu theo thời gian.

Trong bối cảnh các thiết bị IoT đo mưa, bảo trì dự đoán đóng vai trò đặc biệt quan trọng bởi:

- Thiết bị thường đặt tại các vị trí xa hoặc khó tiếp cận và phân bố rải rác trên phạm vi lớn.
- Việc kiểm tra thủ công thường xuyên là tốn kém và không khả thi.
- Hầu hết việc phát hiện lỗi hiện tại chỉ diễn ra ở vào cuối giai đoạn, dấu hiệu đã rõ rệt và đã gây ra ảnh hưởng nghiêm trọng.

Hình 1.8 Các chiến lược bảo trì trên trục thời gian

Thực tế, thời điểm diễn ra bảo trì dự đoán không cố định mà phụ thuộc vào chất lượng dữ liệu (số lượng, tần suất, mức độ đa dạng và độ phức tạp) và năng lực suy luận (khả năng học và khái quát hóa) của các mô hình trí tuệ nhân tạo. Trong trường hợp dữ liệu đầy đủ và mô hình có khả năng học tốt, các bất thường có thể được phát hiện ngay từ giai đoạn đầu hoặc thậm chí trước khi nó xảy ra. Ngược lại, nếu dữ liệu hạn chế và kém chất lượng hay mô hình không đủ mạnh để nhận diện các tín hiệu cảnh báo, việc phát hiện sự cố chỉ có thể xảy ra khi những biểu hiện của nó đã rõ rệt.

1.4. Kết luận chương 1

Trong chương 1, đồ án đã trình bài tổng quan về thực trạng, các lỗi thường gặp và các chiến lược bảo trì. Trong số các lỗi thường gặp, tắc phễu cảm biến đo mưa là một lỗi phổ biến chiếm tỉ trọng cao ảnh hưởng trực tiếp đến độ chính xác và dữ liệu quan trắc. Đây là lỗi xảy ra dần theo thời gian, có thể được phát hiện thông qua các biểu hiện từ sớm thông qua dữ liệu mưa.

Đối với các phương pháp thực hiện bài toán dự đoán, nhóm chọn phương pháp phân tích xu hướng chuỗi thời gian vì dữ liệu vào đến từ một nguồn cảm biến từ đó đưa ra xác suất lỗi theo thời gian.

Ở chương tiếp theo, đồ án sẽ tập trung vào việc đề xuất các giải pháp xây dựng hệ thống phát hiện lỗi theo hướng bảo trì dự đoán.

CHƯƠNG 2: XÂY DỰNG GIẢI PHÁP BẢO TRÌ DỰ ĐOÁN

2.1 Tổng quan các giải pháp

Hệ thống thiết bị IoT đo mưa đóng vai trò quan trọng trong việc thu thập dữ liệu khí tượng thời gian thực, phục vụ các bài toán dự báo, cảnh báo và quản lý thiên tai. Với đặc thù phân bố rộng khắp và thường được triển khai ở những khu vực xa trung tâm (vùng núi, hải đảo, nông thôn), việc bảo trì thủ công các thiết bị này gặp nhiều khó khăn cả về chi phí lẫn thời gian. Do đó, yêu cầu về một hệ thống bảo trì dự đoán nhằm phát hiện sớm các dấu hiệu bất thường, cảnh báo về tiềm năng lỗi xảy ra sớm, từ đó tối ưu hóa lịch bảo trì trở nên cấp thiết.

Một hướng tiếp cận phổ biến hiện nay là tận dụng các mô hình trí tuệ nhân tạo (AI) triển khai tại máy chủ trung tâm (Cloud AI) để xử lý tập trung dữ liệu thu thập được từ các thiết bị. Cách tiếp cận này có thể đem lại độ chính xác cao trong phân tích và dự đoán lỗi nhờ sức mạnh xử lý và khả năng lưu trữ lớn. Tuy nhiên, trong bối cảnh thực tế của các hệ thống IoT đo mưa, việc phụ thuộc hoàn toàn vào hạ tầng Cloud gây ra nhiều bất cập như độ trễ khi phát hiện bất thường và là không thể phản ứng kịp thời trong điều kiện mạng không ổn định.

Từ thực tiễn đó, giải pháp ứng dụng trí tuệ nhân tạo tại biên (Edge AI) được xem là hướng đi triển vọng, cho phép thiết bị tự phát hiện bất thường ngay tại chỗ, phản ứng nhanh chóng mà không cần chờ xử lý từ trung tâm. Tuy nhiên, Edge AI cũng tồn tại những giới hạn nhất định về tài nguyên phần cứng và độ chính xác khi chỉ xử lý cục bộ. Do đó, một mô hình kết hợp giữa Edge AI và Cloud AI – trong đó Edge phụ trách phát hiện bất thường sơ bộ, còn Cloud đảm nhiệm dự đoán chính xác – sẽ là giải pháp cân bằng hiệu quả giữa tính linh hoạt và độ tin cậy.

Trong chương này sẽ lần lượt phân tích các giải pháp Cloud AI và Edge AI, đánh giá các ưu – nhược điểm khi áp dụng vào hệ thống thiết bị đo mưa, từ đó đề xuất mô hình hai tầng ứng dụng Edge AI cho bảo trì dự đoán. Giải pháp này hướng đến khả năng triển khai thực tế trong các hệ thống quan trắc khí tượng hiện nay.

2.2. Giải pháp Cloud AI trong bảo trì dự đoán tại thiết bị IoT

2.2.1. Tổng quan về Cloud AI.

Cloud AI (Artificial Intelligence on Cloud Platform) là mô hình điện toán kết hợp giữa nền tảng đám mây (Cloud Computing) và các thuật toán trí tuệ nhân tạo/học máy (AI/ML) nhằm xử lý khối lượng dữ liệu lớn với hiệu suất cao. Trong lĩnh vực bảo trì dự đoán (Predictive Maintenance - PdM), Cloud AI đóng vai trò quan trọng trong việc phân tích dữ liệu thu thập từ hệ thống thiết bị IoT, từ đó phát hiện sớm các dấu hiệu hư hỏng và đưa ra cảnh báo chủ động.

Hệ thống Cloud AI trong PdM hoạt động theo mô hình tập trung, trong đó dữ liệu từ các thiết bị IoT được truyền về máy chủ đám mây thông qua các phương thức kết nối mạng như Wifi, GSM, LTE, v.v. Tại đây, các mô hình học máy (Machine Learning) và học sâu (Deep Learning) được triển khai để thực hiện các nhiệm vụ như phân tích:

- Phát hiện bất thường (Anomaly Detection).
- Chẩn đoán lỗi (Fault Diagnosis).
- Dự báo tuổi thọ còn lại (Remaining Useful Life - RUL).
- Phân tích xu hướng (Trend Analysis).
- Phân tích tình trạng thiết bị (Equipment Condition Analysis).

Theo nghiên cứu [11], kiến trúc hệ thống Cloud AI bao gồm bốn thành phần chính: thu thập dữ liệu (Telemetry Collection), lưu trữ dữ liệu (Data Storage Layer), xử lý AI (AI Processing Engine), và hệ thống phản hồi tự động (Automated Response Systems). Việc triển khai AI trên nền tảng đám mây cho phép xử lý dữ liệu theo thời gian thực và đưa ra các hành động bảo trì kịp thời.

Hình 2.1 Quy trình thực hiện bài toán dự báo trên hệ thống đám mây

Quy trình vận hành của hệ thống Cloud AI trong PdM được thực hiện tuần tự qua các giai đoạn sau:

- **Bước 1:** Thu thập dữ liệu (Data Acquisition): Các cảm biến (sensor) gắn trên thiết bị thu thập thông số vận hành theo thời gian thực. Dữ liệu được truyền tải lên hệ thống đám mây thông qua các giao thức như MQTT (Message Queuing Telemetry Transport), HTTP/HTTPS, hoặc API Gateway.
- **Bước 2:** Lưu trữ và tiền xử lý (Data Storage & Preprocessing): Dữ liệu được lưu trữ trên các nền tảng như AWS S3, Google BigQuery, hoặc Azure Blob Storage. Quá trình tiền xử lý bao gồm làm sạch dữ liệu (Data Cleaning), chuẩn hóa (Normalization), và xử lý nhiễu (Noise Reduction).
- **Bước 3:** Xây dựng và huấn luyện mô hình (Model Training): Sử dụng các framework như TensorFlow, PyTorch, hoặc các dịch vụ AutoML để huấn luyện mô hình trên tập dữ liệu lịch sử nhằm nhận diện các mẫu hư hỏng.

- **Bước 4:** Phân tích và dự đoán (Analysis & Prediction): Dữ liệu mới được đưa vào mô hình để dự báo xác suất hư hỏng và thời điểm dự đoán. Kết quả được hiển thị qua dashboard giám sát hoặc hệ thống cảnh báo (email/SMS).
- **Bước 5:** Tích hợp hệ thống (System Integration): Kết nối với các nền tảng quản lý như ERP (Enterprise Resource Planning) hoặc CMMS (Computerized Maintenance Management System) để tự động hóa quy trình bảo trì.

2.2.2. Một số nghiên cứu liên quan đến Cloud AI trong bảo trì dự đoán thiết bị IoT.

Trong bối cảnh các hệ thống IoT ngày càng đóng vai trò trung tâm trong giám sát và vận hành thiết bị từ công nghiệp đến đô thị, việc chuyển từ phương pháp bảo trì định kỳ sang bảo trì dự đoán (Predictive Maintenance – PdM) dựa trên dữ liệu thời gian thực đang trở thành một xu thế tất yếu. Đặc biệt, sự kết hợp giữa IoT và trí tuệ nhân tạo (AI) cho phép phát hiện sớm các dấu hiệu hư hỏng, tối ưu hóa chi phí bảo trì và nâng cao độ tin cậy của toàn bộ hệ thống. Một số nghiên cứu gần đây đã tập trung vào việc khai thác sức mạnh của điện toán đám mây để triển khai các mô hình AI phục vụ bảo trì thông minh.

Nghiên cứu của Peruthambi và cộng sự (2025) [12] triển khai một hệ thống bảo trì dự đoán cho môi trường IIoT dựa trên dữ liệu lớn, kết hợp các mô hình học máy và công nghệ blockchain để tối ưu hoá hiệu quả và độ tin cậy. Dữ liệu được thu thập từ cảm biến mô phỏng với 50.000 bản ghi thời gian thực, bao gồm các đặc trưng như nhiệt độ, rung động, áp suất và năng lượng tiêu thụ. Quá trình xử lý và huấn luyện được thực hiện trong môi trường cục bộ cấu hình cao (Intel i7, 32GB RAM, GPU RTX 3060), với các thư viện Python như TensorFlow, Scikit-learn và XGBoost. Bốn mô hình được đánh giá gồm: Random Forest (RF), Support Vector Machine (SVM), Long Short-Term Memory (LSTM), và Extreme Gradient Boosting (XGBoost). Trong đó, XGBoost thể hiện hiệu năng cao nhất với độ chính xác 96.1%, precision 95.6% và F1-score 95.4% (Trang 27, Bảng 3.2). LSTM đạt 95.4% accuracy, thích hợp với dữ liệu chuỗi thời gian nhưng tiêu tốn thời gian huấn luyện gấp đôi RF. RF đạt 92.3%, vẫn giữ độ ổn định và khả năng chống nhiễu. SVM là mô hình kém hiệu quả nhất (89.7%), không phù hợp với dữ liệu có quan hệ phi tuyến phức tạp. Cách tiếp cận đa mô hình này cho phép phát hiện sớm các dấu hiệu bất thường nhỏ vốn có thể bị bỏ qua bởi các phương pháp truyền thống và giúp giảm tới 28% cảnh báo giả và tăng cường bảo mật. Tuy nhiên, nghiên cứu cũng chỉ ra một số hạn chế như khó mở rộng trong các hệ thống lớn, yêu cầu cao về tài nguyên phần cứng, và thách thức trong tích hợp vào hạ tầng công nghiệp hiện có.

Tiếp nối hướng tiếp cận dựa trên nền tảng đám mây, nhóm tác giả Shanbhag và các cộng sự [13] đã đề xuất một giải pháp toàn diện cho bảo trì dự đoán và phát hiện bất thường trong môi trường IIoT, tập trung vào tích hợp trí tuệ nhân tạo (AI) và học máy trên nền tảng đám mây Microsoft Azure. Các mô hình được sử dụng để phát hiện bất thường và đưa ra dự đoán trong nghiên cứu gồm: Support Vector Machine (SVM), Isolation Forest và Extreme Gradient Boosting (XGBoost). Trong đó, XGBoost được

đánh giá là mô hình hiệu quả nhất khi xử lý dữ liệu lớn, đa chiều và không đồng nhất – đặc biệt nhờ khả năng học các quan hệ phi tuyến và tương tác phức tạp giữa các đặc trưng. Kết quả triển khai thực tế trên Azure cho thấy hệ thống giúp giảm đến 30% thời gian dừng máy ngoài kế hoạch và tăng đáng kể độ chính xác trong phát hiện lỗi sớm. Hơn nữa, hệ thống còn giúp nâng cao hiệu suất tổng thể của nhà máy thông qua tích hợp Digital Twin và công cụ giám sát từ xa. Tuy nhiên, nhóm tác giả cũng thẳng thắn nêu ra các hạn chế chính của kiến trúc dựa trên đám mây: độ trễ truyền dữ liệu, chi phí lưu trữ và tính toán cao, và các yêu cầu nghiêm ngặt về an toàn thông tin và quyền riêng tư trong môi trường IIoT.

Tiếp nối các hướng tiếp cận tích hợp IIoT và học máy trong bảo trì dự đoán, nghiên cứu của Mohammed và cộng sự. (2023) [14] tập trung phát triển một hệ thống nhẹ, chi phí thấp, phù hợp với môi trường công nghiệp cỡ nhỏ và vừa, đặc biệt trong giám sát tình trạng động cơ điện. Hệ thống được xây dựng dựa trên nền tảng Raspberry Pi kết hợp với các cảm biến rung, dòng điện và nhiệt độ, cho phép thu thập dữ liệu theo thời gian thực. Dữ liệu từ cảm biến được truyền đến máy chủ đám mây thông qua giao thức MQTT, nổi bật với khả năng tiêu thụ băng thông thấp, độ trễ nhỏ phù hợp với IIoT. Tập dữ liệu thu thập được gồm hơn 35.000 mẫu từ 5 trạng thái lỗi khác nhau, bao gồm: không lỗi, lỗi rung, lỗi dòng điện cao, lỗi bạc đạn và lỗi dừng quay. Sau khi thu thập và xử lý, dữ liệu được chia theo tỷ lệ 80/10/10 cho huấn luyện, xác thực và kiểm thử. Năm mô hình học máy được so sánh gồm: Random Forest (RF), Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Naive Bayes (NB), và Linear Regression (LR). Kết quả thực nghiệm cho thấy Random Forest là mô hình có hiệu quả cao nhất, với độ chính xác vượt trội trong việc dự đoán lỗi và phản ứng nhanh với dữ liệu thực tế. Tuy nhiên, nghiên cứu vẫn tồn tại một số hạn chế: quy mô dữ liệu còn khiêm tốn, chỉ thử nghiệm trên một loại động cơ, và chưa tích hợp các kỹ thuật học sâu. Nhóm tác giả đề xuất mở rộng sang nhiều loại thiết bị công nghiệp khác nhau, bổ sung cảm biến, cũng như áp dụng mô hình học sâu như LSTM hoặc CNN để cải thiện khả năng nhận diện lỗi phức tạp trong môi trường thực tế.

Tổng hợp từ ba nghiên cứu đã phân tích, XGBoost, Random Forest được xác định là các mô hình học máy có hiệu năng cao nhất khi triển khai trên nền tảng điện toán đám mây, nhờ khả năng xử lý dữ liệu lớn, học quan hệ phi tuyến và cung cấp kết quả chính xác vượt trội trong bài toán bảo trì dự đoán. Về mặt nền tảng, các nghiên cứu đều cho thấy Microsoft Azure là môi trường triển khai phù hợp, cung cấp hệ sinh thái đầy đủ gồm Azure Machine Learning cho huấn luyện mô hình, Azure IoT Hub để thu thập dữ liệu thời gian thực từ thiết bị và Azure Stream Analytics để xử lý chuỗi dữ liệu.

2.2.3. Ưu điểm Cloud AI

- Khả năng xử lý dữ liệu quy mô lớn: Xử lý đồng thời hàng triệu điểm dữ liệu từ nhiều thiết bị IoT nhờ sức mạnh điện toán đám mây, hỗ trợ các thuật toán AI/ML

phức tạp như Deep Learning và Ensemble Learning, khả năng mở rộng tài nguyên linh hoạt (CPU/GPU/TPU) theo nhu cầu thực tế.

- Tính linh hoạt trong triển khai và tích hợp: Tích hợp sẵn với các nền tảng đám mây hàng đầu (AWS IoT Core, Azure IoT Hub, Google Cloud IoT), hỗ trợ đa dạng giao thức kết nối: MQTT, HTTP, WebSocket, cũng như dễ dàng kết nối với các hệ thống quản lý bảo trì hiện có.
- Hiệu quả kinh tế vượt trội: Giảm đáng kể chi phí đầu tư phần cứng do không cần xây dựng hạ tầng máy chủ cục bộ, tự động hóa quy trình bảo trì giúp tiết kiệm nhân lực vận hành, giảm thời gian ngừng hoạt động (downtime) nhờ khả năng dự báo sự cố chính xác.
- Nâng cao hiệu suất vận hành: Giảm thiểu gián đoạn hoạt động nhờ bảo trì chủ động, tối ưu hóa hiệu suất thiết bị, tiết kiệm năng lượng tiêu thụ, phòng ngừa sự cố bất ngờ, giảm chi phí sửa chữa khẩn cấp.
- Bảo mật và độ tin cậy cao: Áp dụng các chuẩn mã hóa và xác thực tiên tiến, hệ thống sao lưu tự động đảm bảo an toàn dữ liệu, tuân thủ các tiêu chuẩn bảo mật quốc tế.

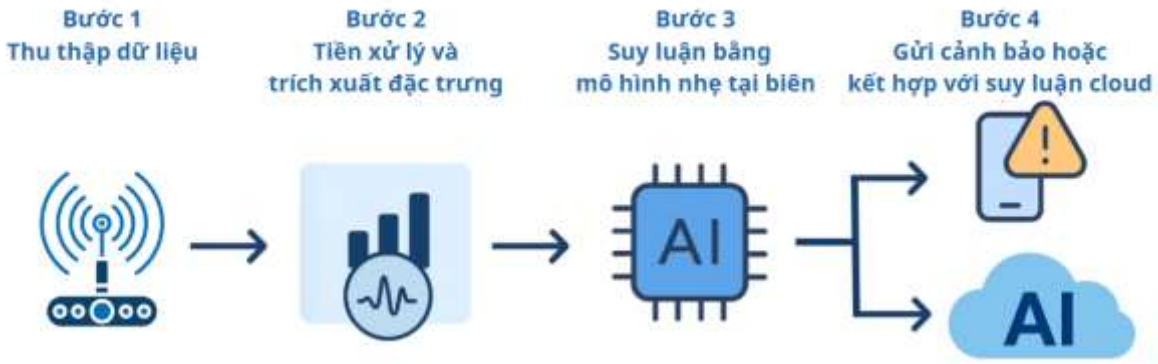
2.2.4 Nhược điểm Cloud AI

- Độ trễ cao: Việc truyền dữ liệu cảm biến liên tục từ thiết bị công nghiệp lên cloud và chờ phản hồi gây ra độ trễ cao, đặc biệt trong các ứng dụng yêu cầu phản ứng tức thì.
- Phụ thuộc vào kết nối mạng: Cloud AI yêu cầu kết nối Internet ổn định để truyền dữ liệu lên cloud và nhận kết quả phân tích. Điều này không phù hợp với các nhà máy ở vùng hẻo lánh hoặc môi trường nhiễu sóng.
- Bảo mật và quyền riêng tư dữ liệu: Dữ liệu công nghiệp có thể nhạy cảm hoặc mang tính độc quyền, nên việc truyền tải và lưu trữ trên cloud có nguy cơ bị rò rỉ hoặc tấn công mạng.
- Chi phí vận hành: Mặc dù cloud giúp tiết kiệm chi phí ban đầu về phần cứng, nhưng chi phí vận hành dài hạn, bao gồm băng thông, lưu trữ dữ liệu lớn, truy vấn và tính toán AI/ML có thể rất tốn kém.
- Quá tải dữ liệu (Bandwidth & Storage): IIoT tạo ra lượng dữ liệu rất lớn, nếu truyền hết lên cloud sẽ gây quá tải băng thông và làm chậm xử lý.

2.3. Giải pháp Edge AI

2.3.1. Tổng quan về Edge AI

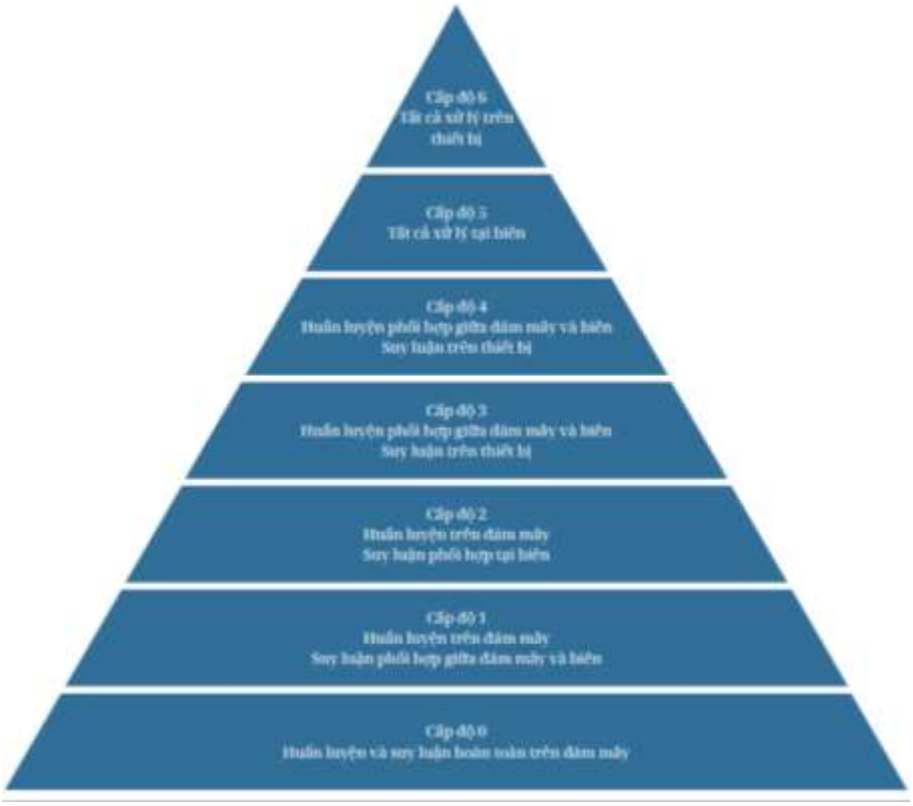
Edge AI (Trí tuệ nhân tạo tại thiết bị biên) là sự kết hợp giữa công nghệ Trí tuệ nhân tạo (AI) và Điện toán biên (Edge Computing) [15], trong đó các thuật toán AI được triển khai và thực thi trực tiếp trên các thiết bị ở gần nguồn dữ liệu như cảm biến, vi điều khiển, gateway hoặc thiết bị nhúng. Phương pháp này cho phép hệ thống xử lý, phân tích dữ liệu và đưa ra quyết định ngay tại hiện trường, thay vì phải truyền dữ liệu đến máy chủ trung tâm hoặc nền tảng đám mây. Để có thể thực hiện được thì các thiết bị đầu cuối Edge AI phải có đủ sức mạnh tính toán để chạy các thuật toán xử lý dữ liệu theo thời gian.



Hình 2.2 Quy trình thực hiện bài toán phát hiện bất thường tại biên

Quy trình vận hành của một hệ thống Edge AI trong các ứng dụng giám sát, phát hiện bất thường và bảo trì dự đoán bao gồm bốn bước chính: (1) thu thập dữ liệu, (2) xử lý tại thiết bị biên, (3) suy luận và phát hiện bất thường, (4) phản hồi hoặc phối hợp với cloud nếu cần thiết.

- **Bước 1:** Thu thập dữ liệu cảm biến: Dữ liệu được thu thập trực tiếp từ các cảm biến lắp đặt tại thiết bị hoặc môi trường cần giám sát. Loại dữ liệu thu được có thể là nhiệt độ, rung động, dòng điện, độ ẩm, áp suất, lưu lượng hoặc âm thanh, tùy theo ứng dụng cụ thể.
- **Bước 2:** Tiền xử lý, trích xuất đặc trưng tại thiết bị biên: Tại thiết bị edge, dữ liệu đầu vào được xử lý sơ bộ nhằm loại bỏ nhiễu, đồng bộ hóa thời gian và chuẩn hóa đầu vào. Sau đó, đặc trưng được trích xuất từ dữ liệu để phục vụ cho bước suy luận.
- **Bước 3:** Suy luận bằng mô hình AI tại thiết bị: Mô hình AI được triển khai trực tiếp trên thiết bị edge (vi điều khiển, gateway) và thực hiện nhiệm vụ phân tích dữ liệu đầu vào để phát hiện bất thường hoặc dự đoán hỏng hóc. Các thuật toán thường được sử dụng gồm: Isolation Forest, LSTM Autoencoder, CNN/RNN nhẹ,…
- **Bước 4:** Cảnh báo tại chỗ hoặc phối hợp với cloud: Khi phát hiện bất thường, hệ thống sẽ gửi cảnh báo tại chỗ (gửi tín hiệu đến hệ thống giám sát, đèn cảnh báo, hoặc app di động), gửi metadata lên cho cloud để xác minh lại độ tin cậy hoặc xử lý, lưu trữ.



Hình 2.3 Các cấp độ triển khai ứng dụng trí tuệ nhân tạo tại biên

Để xác định được mức độ phụ thuộc của thiết bị biên và các thiết bị mạng trung tâm, bài báo [15] đã giới thiệu các cấp độ khác nhau của việc triển khai trí tuệ biên, thể hiện tại Hình 2.3. Ở đây ta có thể hiểu các cấp độ này như là cấp độ di chuyển của các thuật toán ML từ đám mây mang biên. Các cấp độ này được trình bày tóm tắt như sau:

1) *Huấn luyện và suy luận trên đám mây:* tất cả các quy trình của học máy (ML) đều phụ thuộc vào điện toán đám mây.
2) *Suy luận phối hợp giữa đám mây và biên & Huấn luyện trên đám mây:* việc huấn luyện các thuật toán học máy dựa vào điện toán đám mây do khả năng tính toán mạnh mẽ, trong khi quá trình suy luận được thực hiện dưới dạng hợp tác giữa đám mây và thiết bị biên thông qua việc phân chia dữ liệu.
3) *Suy luận phối hợp trong biên & Huấn luyện trên đám mây:* giống như cấp trước, việc huấn luyện vẫn dựa vào đám mây, nhưng lần này quá trình suy luận được thực hiện trong mạng biên, thông qua việc phân chia dữ liệu hoàn toàn hoặc một phần.
4) *Suy luận trên thiết bị & Huấn luyện phối hợp giữa đám mây và biên:* quá trình huấn luyện diễn ra trên cả đám mây và biên, trong khi suy luận kết quả được thực hiện hoàn toàn cục bộ trên thiết bị.
5) *Huấn luyện phối hợp giữa đám mây và biên & suy luận trên thiết bị:* cả hai quá trình được thực hiện theo cách hợp tác giữa đám mây và thiết bị biên.

6) *Tất cả thực hiện tại biên:* quá trình huấn luyện và suy luận đều được thực hiện tại biên mạng, hiểu rằng biên mạng là một mạng lưới cộng tác của các thiết bị biên.
7) *Tất cả thực hiện trên thiết bị:* cả quá trình huấn luyện và suy luận đều được thực hiện hoàn toàn trong thiết bị biên.

2.3.2. Một số nghiên cứu liên quan đến Edge AI trong phát hiện bất thường tại thiết bị IoT.

Trong những năm gần đây, xu hướng đưa trí tuệ nhân tạo (AI) từ nền tảng đám mây xuống thiết bị biên (Edge AI) đang thu hút sự quan tâm mạnh mẽ từ cộng đồng nghiên cứu và công nghiệp, đặc biệt trong các ứng dụng yêu cầu phản ứng nhanh, tiết kiệm năng lượng và đảm bảo quyền riêng tư dữ liệu. Khác với các kiến trúc truyền thống phụ thuộc hoàn toàn vào cloud, mô hình Edge AI cho phép xử lý dữ liệu trực tiếp tại thiết bị đầu cuối (như vi điều khiển, gateway), từ đó mở ra hướng tiếp cận mới cho bảo trì dự đoán trong môi trường phân tán, có tài nguyên hạn chế. Một số các công trình gần đây đã làm rõ vai trò và tiềm năng của Edge AI trong phát hiện bất thường và bảo trì thiết bị IoT.

Nghiên cứu của Reis và Serôdio [15] giới thiệu một kiến trúc Edge AI đa tầng nhằm phát hiện bất thường theo thời gian thực trong các hệ thống nhà thông minh, nơi dữ liệu được thu thập từ các cảm biến IoT như nhiệt độ, chuyển động và tiêu thụ điện năng. Mô hình AI được triển khai trực tiếp trên thiết bị biên (Raspberry Pi), cho phép xử lý dữ liệu tại chỗ mà không cần gửi toàn bộ dữ liệu lên cloud. Trong hệ thống này, Isolation Forest (IF) đóng vai trò cốt lõi trong việc cách ly và phát hiện các điểm dữ liệu bất thường từ đầu vào cảm biến. IF được kết hợp với LSTM Autoencoder để tăng khả năng phát hiện các bất thường theo chuỗi thời gian. Nhờ cơ chế xử lý cục bộ và chiến lược chỉ gửi các sự kiện nghi vấn lên cloud, hệ thống không chỉ giảm độ trễ suy luận xuống dưới 50ms mà còn giảm đáng kể lưu lượng truyền tải dữ liệu và tiêu thụ năng lượng. Đặc biệt, việc áp dụng lượng tử hóa mô hình giúp tiết kiệm đến 35% năng lượng tiêu thụ. Kết quả thực nghiệm ghi nhận độ chính xác lên đến 93,6% trong việc phát hiện bất thường, cho thấy tính hiệu quả cao của giải pháp IF kết hợp Edge AI trong các hệ thống cảm biến IoT yêu cầu phản hồi nhanh, tiết kiệm năng lượng và duy trì kết nối ổn định.

Trong hai nghiên cứu tiếp theo của Szydlo (2022) [16] và nhóm tác giả của Budiarto (2019) [17] đều nhấn mạnh tính hiệu quả của thuật toán Local Outlier Factor (LOF) trong phát hiện bất thường từ dữ liệu chuỗi thời gian không nhãn trong hệ thống IoT. Nghiên cứu của Szydlo tập trung vào triển khai LOF trực tiếp trên vi điều khiển (MCU), nơi dữ liệu cảm biến được xử lý theo cửa sổ trượt và chọn mẫu bằng reservoir sampling để huấn luyện mô hình. Nhờ đó, hệ thống có thể phát hiện bất thường với thời gian suy luận chỉ ~20ms và không cần gửi dữ liệu lên cloud, rất phù hợp với thiết bị biên tài nguyên thấp. Trong khi đó, Budiarto et al. áp dụng LOF để phát hiện bất thường trong dữ liệu sử dụng thuốc tại bệnh viện, gồm hai chuỗi thời gian theo tuần. Mặc dù môi

trường nhẹ hơn, kết quả cho thấy LOF hoạt động ổn định, có khả năng phát hiện outlier hiệu quả và thời gian xử lý nhanh (~3ms), đáp ứng yêu cầu hệ thống IoT y tế. Sự kết hợp giữa hai nghiên cứu cho thấy LOF là thuật toán nhẹ, chính xác và phù hợp để triển khai trên thiết bị biên, đặc biệt trong môi trường không có nhãn, cần phản hồi nhanh và tiết kiệm năng lượng.

Bên cạnh đó, theo nghiên cứu của hai nhóm tác giả Kong [18] và Ruksana [19] đều nhấn mạnh tiềm năng của Edge AI và thuật toán K-means trong việc nâng cao hiệu quả phát hiện bất thường và bảo trì dự đoán tại các hệ thống công nghiệp IoT. Trong nghiên cứu của Kong và cộng sự, các tác giả đề xuất một kiến trúc IIoT với khả năng nén dữ liệu cảm biến tại thiết bị biên bằng một thuật toán dựa trên ngưỡng sai số (error threshold e) và kích thước nhóm dữ liệu k. Sau khi nén, dữ liệu được phân cụm bằng K-means clustering để phát hiện bất thường. Kết quả cho thấy khi chọn thông số tối ưu là k = 20 và e = 0.01, hệ thống đạt được tỷ lệ nén dữ liệu đến 66% (chỉ giữ lại 408 trên 1200 mẫu) trong khi vẫn phát hiện đúng 30/31 điểm bất thường, chỉ có 1 điểm bị nhầm lẫn – thể hiện độ chính xác rất cao. Ngoài ra, thời gian xử lý toàn bộ chuỗi dữ liệu chỉ mất 6.75 giây, phù hợp với yêu cầu thời gian thực trong công nghiệp. Trong khi đó, bài viết của Ruksana và cộng sự đã tổng quan hóa kiến trúc tích hợp Edge AI – IoT – Cloud theo hướng ba tầng: cảm biến – edge – cloud, trong đó lớp edge xử lý dữ liệu cục bộ bằng các mô hình học máy nhẹ như K-means, SVM hoặc Random Forest. Tác giả đặc biệt nhấn mạnh rằng Edge AI giúp giảm đáng kể độ trễ do xử lý dữ liệu ngay tại thiết bị biên, đồng thời giảm băng thông truyền dữ liệu lên cloud đến hơn 70%, nhờ chỉ gửi các chỉ số bất thường thay vì toàn bộ dữ liệu thô. Mô hình này không chỉ nâng cao tốc độ phản hồi mà còn đảm bảo độ tin cậy trong các môi trường kết nối không ổn định như vùng sâu vùng xa hoặc khu công nghiệp nguy hiểm. Sự kết hợp giữa hai nghiên cứu cho thấy K-means là một thuật toán phù hợp để triển khai trên nền tảng edge nhờ tính đơn giản, hiệu quả và khả năng hoạt động trong môi trường không có dữ liệu nhãn (unsupervised). Khi được áp dụng tại lớp biên, nó có thể phân cụm dữ liệu cảm biến theo hành vi và phát hiện các mẫu bất thường một cách nhanh chóng. Đồng thời, khả năng nén dữ liệu thông minh tại thiết bị biên giúp hệ thống trở nên nhẹ, phản ứng nhanh và dễ mở rộng, đặc biệt trong các hệ thống IIoT có hàng ngàn thiết bị hoạt động song song.

2.3.3. Ưu điểm

- Giảm độ trễ, phản ứng tức thì: Edge AI xử lý dữ liệu trực tiếp tại thiết bị biên, giúp giảm thời gian từ lúc phát hiện bất thường đến khi phản hồi. Điều này rất quan trọng trong các hệ thống yêu cầu phản ứng thời gian thực.
- Giảm băng thông, tiết kiệm chi phí truyền tải: Chỉ dữ liệu đã qua xử lý, hoặc các cảnh báo, mới được gửi lên cloud thay vì toàn bộ dữ liệu cảm biến. Điều này giúp giảm tới 80% lưu lượng mạng.
- Tăng bảo mật và quyền riêng tư: Dữ liệu nhạy cảm không cần rời khỏi thiết bị, hạn chế nguy cơ bị đánh cắp hoặc xâm nhập trong quá trình truyền tải. Điều này đặc biệt quan trọng trong môi trường công nghiệp.

- Phù hợp môi trường không ổn định mạng: Edge AI có thể hoạt động độc lập, ngay cả khi mất kết nối internet. Điều này giúp hệ thống vẫn duy trì khả năng giám sát tại các vị trí hẻo lánh hoặc nhiều nhiễu.
- Hiệu quả năng lượng và tốc độ nhờ tối ưu mô hình: Edge AI có thể hoạt động độc lập, ngay cả khi mất kết nối internet. Điều này giúp hệ thống vẫn duy trì khả năng giám sát tại các vị trí hẻo lánh hoặc nhiều nhiễu.
- Khả năng chạy trên thiết bị nhẹ: Các kỹ thuật như lượng tử hóa (quantization) giúp giảm kích thước mô hình và tiết kiệm năng lượng khi suy luận, thích hợp với thiết bị công suất thấp.

2.3.4. Nhược điểm

- Hạn chế tài nguyên phần cứng: Thiết bị edge thường có bộ nhớ, năng lượng và khả năng tính toán thấp, không đáp ứng tốt với các mô hình AI phức tạp hoặc dung lượng lớn.
- Khó triển khai mô hình AI sâu, phức tạp: Những mô hình học sâu như LSTM đa tầng hoặc Transformer yêu cầu tài nguyên vượt quá khả năng phần cứng edge hiện tại, làm giảm tính ứng dụng.
- Thiếu khả năng đồng bộ giữa các node: Các thiết bị edge hoạt động rời rạc, gây khó khăn trong việc đồng bộ mô hình hoặc cập nhật tập trung, đặc biệt khi số lượng node lớn.
- Khó bảo trì và cập nhật mô hình từ xa: Việc triển khai lại mô hình hoặc cập nhật firmware trên hàng trăm thiết bị edge là một thách thức vận hành và bảo trì đáng kể.
- Không lưu trữ được dữ liệu dài hạn: Vì giới hạn bộ nhớ, edge devices không thể lưu dữ liệu lịch sử, làm giảm khả năng phân tích xu hướng dài hạn hoặc huấn luyện lại mô hình.

2.4 Mô hình đề xuất: Kiến trúc hai tầng (Hybrid) Edge + Cloud AI

2.4.1. Cơ sở đề xuất mô hình hai tầng

Dựa vào những phân tích về ưu và nhược điểm của giải pháp Edge AI và Cloud AI từ các công trình nghiên cứu từ trước cũng như những phân tích về lỗi tắc phễu hứng mưa thực tế, ta có bảng sau:

Bảng 2.1 Ưu nhược điểm của các giải pháp đối với bài toán phát hiện lỗi tắc phễu hứng mưa

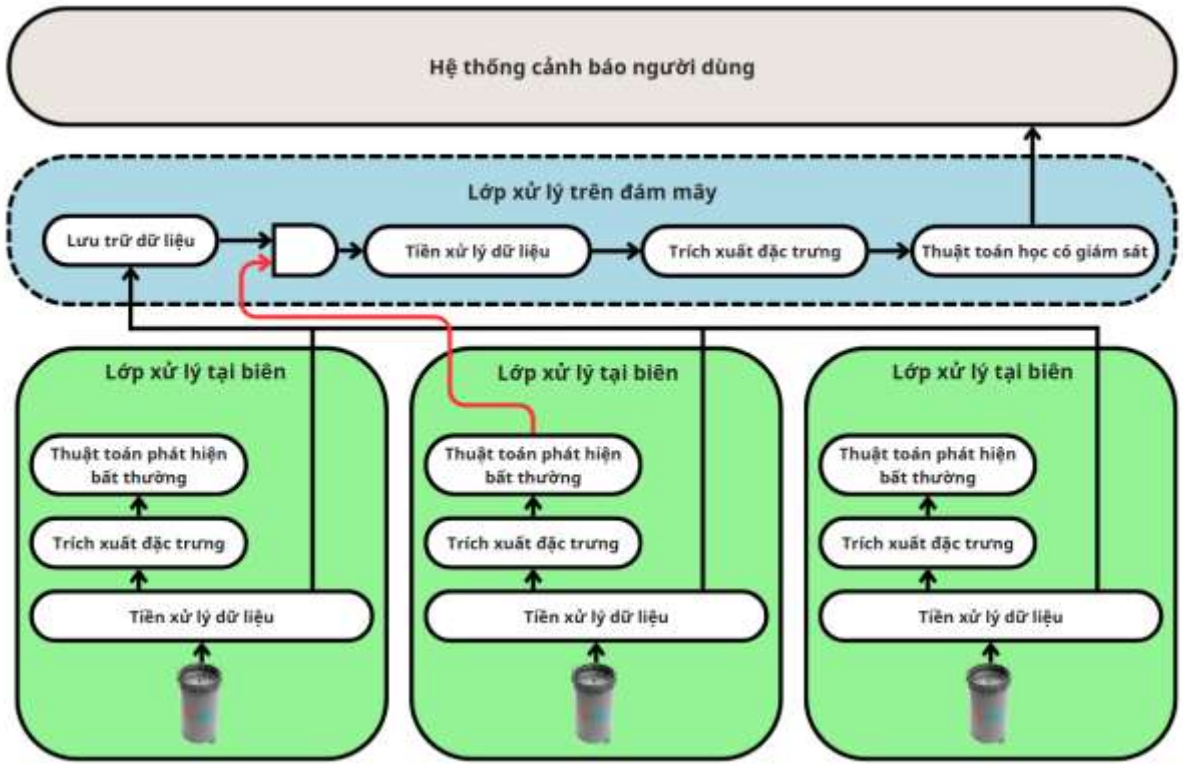
Tiêu chí	Edge AI	Cloud AI
Độ trễ	Cực thấp vì dữ liệu xử lý tại nguồn và không phụ thuộc vào kết nối mạng.	Phụ thuộc nhiều vào kết nối mạng và lượng dữ liệu truyền tải.
Độ tin cậy	Cao vì không phụ thuộc vào kết nối mạng và các thiết bị khác.	Phụ thuộc vào mạng và các thiết bị trong mạng

Tài nguyên phần cứng	Hạn chế, phù hợp với các mô hình đơn giản	Có thể triển khai mô hình phức tạp và chuyên sâu.
Độ chính xác	Dữ liệu cục bộ có thể gây ra hiện tượng bắt nhầm hoặc bắt thiếu.	Dữ liệu đến từ nhiều nguồn giúp phân biệt được mưa ngâu.

Từ bảng phân tích trên cho thấy, cả hai mô hình Edge AI và Cloud AI đều có những hạn chế nhất định khi áp dụng riêng lẻ trong bài toán bảo trì dự đoán cho thiết bị IoT – cụ thể là trong việc phát hiện lỗi tắc phễu hứng mưa, ưu điểm của giải pháp này chính là nhược điểm của giải pháp kia và ngược lại. Các yêu cầu về độ chính xác, tốc độ xử lý và khả năng ứng dụng thực tế đều chưa được đáp ứng toàn diện.

Do đó, đề tài này đề xuất một kiến trúc Hybrid hai tầng kết hợp giữa Edge AI và Cloud AI, trong đó mỗi tầng đảm nhận vai trò phù hợp với thế mạnh riêng giúp cân bằng giữa lợi ích xử lý tại chỗ và năng lực phân tích nâng cao.

2.4.2. Mô tả kiến trúc



Hình 2.4 Kiến trúc hệ thống Hybrid đề xuất

Tầng 1: Lớp xử lý tại biên (Edge Layer)

Chức năng chính:

- Thu thập dữ liệu lượng mưa thời gian thực từ cảm biến tại trạm đo.

- Tiền xử lý dữ liệu tại chỗ: lọc nhiễu, loại bỏ dữ liệu bất thường thô, chuẩn hóa và trích xuất các đặc trưng đơn giản như trung bình trượt, phương sai theo thời gian.
- Phát hiện bất thường tức thời sử dụng các thuật toán học máy không giám sát (Unsupervised ML) nhẹ như: K-Means Clustering, Isolation Forest, Local Outlier Factor.
    - Dữ liệu đầu vào: chuỗi thời gian lượng mưa từ một trạm đo.
    - Dữ liệu đầu ra: các mốc thời gian nghi ngờ có dấu hiệu bất thường.
- Khi phát hiện bất thường, chỉ thông tin rút gọn (metadata) và đoạn dữ liệu liên quan được gửi lên tầng Cloud, giúp giảm lưu lượng truyền tải và bảo vệ quyền riêng tư dữ liệu.

Phần cứng đề xuất để triển khai: Raspberry Pi 4 (RAM 4GB) hoặc thiết bị nhúng có khả năng tính toán tương đương.

Tầng 2: Xử lý tại đám mây (Cloud Layer)

Chức năng chính:

- Nhận và lưu trữ dữ liệu mưa từ các trạm đo IoT.
- Phân tích tương quan giữa các trạm lân cận để nhận diện sự khác biệt lượng mưa (dấu hiệu tắc phễu).
- Tiền xử lý dữ liệu trên đám mây (loại bỏ giá trị nhiễu, giá trị trống,..), trích xuất các đặc trưng. Sử dụng các kỹ thuật cân bằng dữ liệu để xử lý mất cân bằng lớp trong huấn luyện mô hình.
- Ứng dụng các mô hình học máy nâng cao (Decision Tree, Random Forest, XGBoost, Logistic Regression, AdaBoost,...) để phân loại và dự đoán xác suất lỗi theo thời gian.
    - Dữ liệu đầu vào: giá trị lượng mưa của các trạm gần nhau.
    - Dữ liệu đầu ra: xác xuất lỗi theo thời gian của trạm gửi tín hiệu lỗi.
- Khi phát hiện xác suất lỗi vượt ngưỡng, hệ thống gửi cảnh báo đến giao diện quản lý người dùng.

Nền tảng đề xuất triển khai: Microsoft Azure

- Azure IoT Hub: kết nối và nhận dữ liệu từ các thiết bị.
- Azure Stream Analytics: xử lý dữ liệu chuỗi theo thời gian thực.
- Azure Machine Learning: huấn luyện và triển khai mô hình học máy.

Hệ thống cảnh báo và giao diện quản lý

Chức năng chính:

- Nhận tín hiệu cảnh báo từ tầng Cloud khi phát hiện lỗi hoặc nghi ngờ lỗi xảy ra.
- Hiển thị cảnh báo lỗi (thời gian, vị trí trạm, xác suất lỗi) trên giao diện người dùng.
- Cho phép truy xuất lịch sử lỗi theo thời gian và xuất báo cáo tổng hợp phục vụ phân tích và bảo trì.

2.5. Kết luận chương 2

Chương này đã trình bày về Edge AI và Cloud AI các nghiên cứu liên quan đến việc ứng dụng Cloud AI và Edge AI trong bảo trì dự đoán và phát hiện bất thường tại thiết bị IoT. Thông qua đánh giá chi tiết ưu và nhược điểm của từng giải pháp, nhóm nghiên cứu nhận thấy rằng:

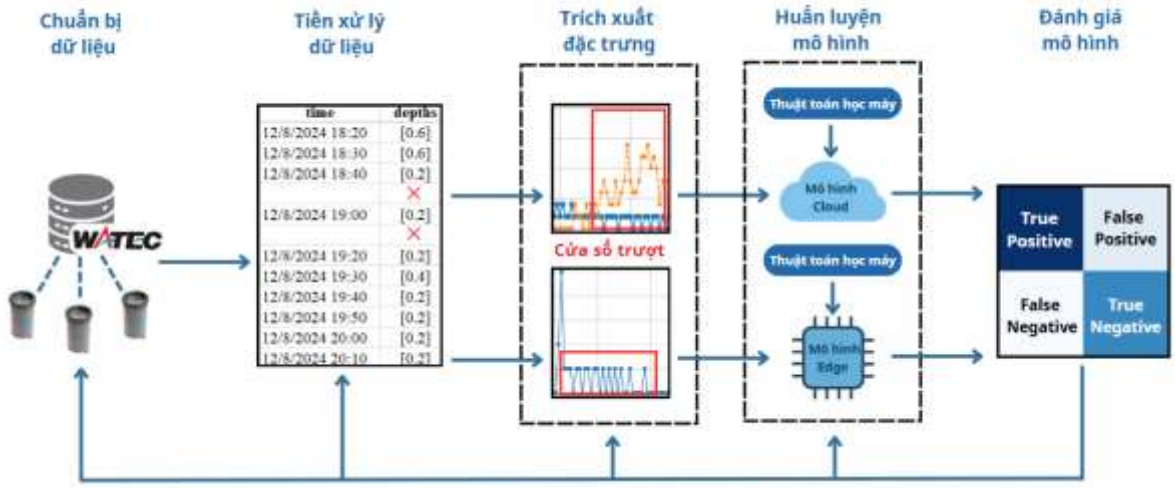
- Cloud AI có thế mạnh về khả năng xử lý dữ liệu quy mô lớn, hỗ trợ các mô hình học sâu và phân tích chuyên sâu, nhưng gặp hạn chế trong việc phản ứng thời gian thực và phụ thuộc lớn vào hạ tầng mạng.
- Trong khi đó, Edge AI cho phép xử lý dữ liệu trực tiếp tại thiết bị, mang lại độ trễ thấp, phản ứng nhanh. Tuy nhiên, khả năng học và phân tích cục bộ còn giới hạn do tài nguyên phần cứng hạn chế.

Từ thực tiễn đó, nhóm đã đề xuất một kiến trúc hai tầng (Hybrid), kết hợp Edge AI và Cloud AI một cách hợp lý. Trong kiến trúc này, Edge AI đóng vai trò tuyến đầu trong phát hiện bất thường tại chỗ, còn Cloud AI đảm nhiệm phân tích liên trạm, chẩn đoán chuyên sâu và tổng hợp dữ liệu phục vụ bảo trì chính xác. Cách tiếp cận này không chỉ giải quyết bài toán phát hiện lỗi nhanh chóng, mà còn đảm bảo tính mở rộng, độ tin cậy và hiệu quả chi phí cho toàn hệ thống IoT đo mưa.

Trong chương tiếp theo, nhóm sẽ trình bày quá trình thực nghiệm, đánh giá kết quả các mô hình đề xuất trên tập dữ liệu thực tế từ các trạm đo, nhằm đánh giá hiệu năng, độ chính xác và khả năng triển khai của hệ thống trong điều kiện thực tế.

CHƯƠNG 3: THỰC NGHIỆM VÀ ĐÁNH GIÁ KẾT QUẢ

3.1. Phương pháp thực hiện



Hình 3.1 Quy trình thực hiện các bài toán

Cả 2 bài toán phát hiện bất thường và phân tích tình trạng lỗi đều thực hiện theo quy trình ở Hình 3.1:

Bước 1: Chuẩn bị dữ liệu: dữ liệu nhóm thực hiện là dữ liệu lượng mưa theo giờ thực tế lấy từ hệ thống đo mưa quốc gia Vrain do công ty Watec cung cấp.

Bước 2: Tiền xử lý dữ liệu: các tập dữ liệu sẽ được chuẩn hóa về định dạng thời gian chuẩn và kiểu dữ liệu thống nhất. Đồng thời các giá trị trống sẽ được loại bỏ để tránh gây nhiễu.

Bước 3: Trích xuất đặc trưng: dữ liệu sau khi được xử lý sẽ được chuyển đổi thành tập đại diện thông tin có tính phân biệt mô tả rõ ràng đặc điểm nhận dạng dữ liệu bình thường và bất thường. Tùy vào từng bài toán cụ thể các đặc trưng được lựa chọn phù hợp với mục tiêu và loại dữ liệu đang xử lý. Các đặc trưng ở đây được trích xuất bằng cửa sổ trượt (sliding window). N giá trị đặc trưng được lấy ra từ một cửa sổ sẽ gộp lại thành một véc-tơ đặc trưng có N chiều để đưa vào các mô hình học máy.

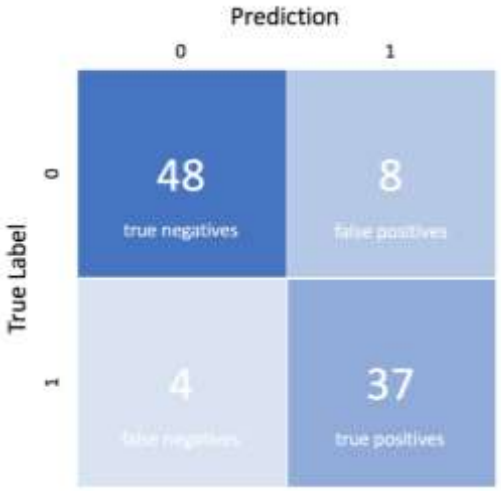
Bước 4: Huấn luyện mô hình: các thuật toán học máy được áp dụng cùng một tập dữ liệu với các thông số tốt nhất để tìm ra mô hình có đáp ứng tốt với các tiêu chí đề ra ở hai bài toán tại biên và trên đám mây.

Bước 5: Đánh giá mô hình: sau huấn luyện, các mô hình được kiểm nghiệm với các chỉ số đánh giá đã được xác định trước. Nếu chưa đạt yêu cầu, quy trình lặp lại từ các bước trước đó cho đến khi đạt mục tiêu.

Nhìn chung, cả hai bài toán tại biên và đám mây có bản chất của bài toán phân loại, có bốn loại kết quả có thể xảy ra:

- **True Positive (TP):** Số lượng mẫu mà mô hình dự đoán là Positive (1) và thực tế cũng là Positive (1).
- **False Positive (FP):** Số lượng mẫu mà mô hình dự đoán là Positive (1) nhưng thực tế là Negative (0).
- **False Negative (FN):** Số lượng mẫu mà mô hình dự đoán là Negative (0) nhưng thực tế là Positive (1).
- **True Negative (TN):** Số lượng mẫu mà mô hình dự đoán là Negative (0) và thực tế cũng là Negative (0).

Bốn kết quả này thường được vẽ trên một ma trận nhầm lẫn (Confusion Matrix). Ma trận nhầm lẫn sau đây là một ví dụ cho trường hợp phân loại nhị phân. Bạn sẽ tạo ma trận này sau khi đưa ra dự đoán trên dữ liệu thử nghiệm của mình và sau đó xác định từng dự đoán là một trong bốn kết quả có thể được mô tả ở trên.



Hình 3.2 Ma trận nhầm lẫn của mô hình phân loại nhị phân

1. Accuracy: Tỷ lệ dự đoán đúng (TP + TN) trên tổng số mẫu

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

2. Precision (Độ chính xác): Tỉ lệ giữa số dự đoán đúng Positive (TP) so với tổng số dự đoán là Positive.

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

3. Recal (Độ nhạy): Tỉ lệ giữa số dự đoán đúng Positive (TP) so với tổng số thực tế là Positive.

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

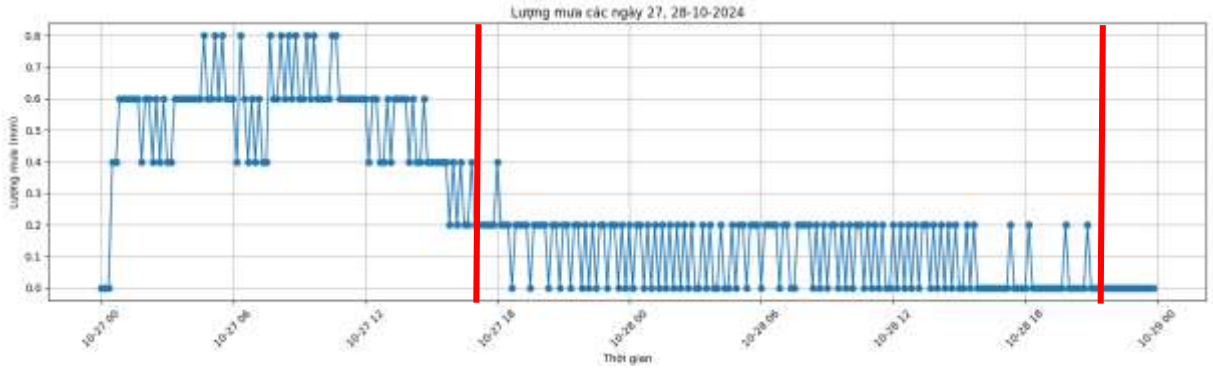
4. F1 – Score: Trung bình điều hòa giữa Precision và Recall.

$$F1 - \ Score = 2 * \frac{Precision * \ Recall}{Precision + \ Recall} \tag{3.4}$$

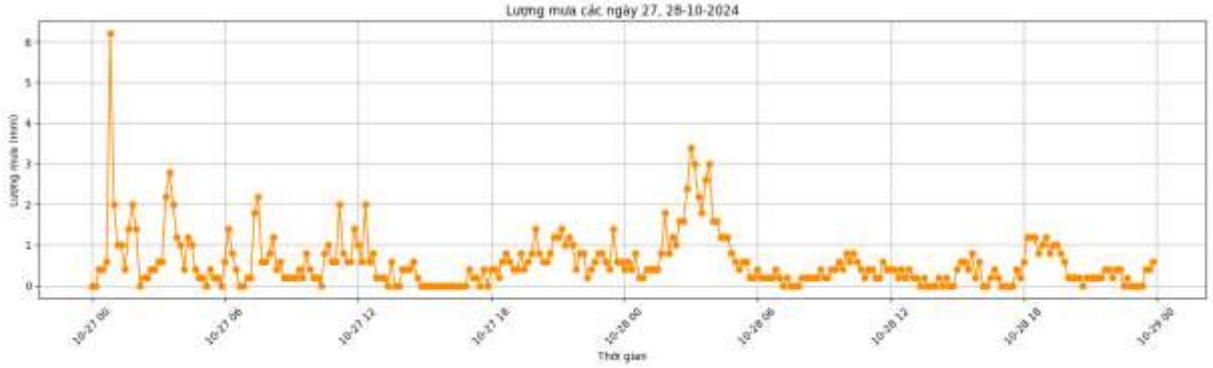
3.2. Bài toán phát hiện bất thường tại Edge

3.2.1. Dữ liệu

Dữ liệu liệu sử dụng cho bài toán phát hiện bất thường tại biên là tập dữ liệu lượng mưa thực tế theo thời gian từ các trạm IoT đo mưa do Watec cung cấp với tần suất lấy mẫu 10 phút/lần. Ở bài toán này nhóm sử dụng các tập dữ liệu lỗi để đưa vào dự đoán và chọn các thông số tối ưu cho mô hình.



Hình 3.3 Biểu đồ lượng mưa bất thường được ghi nhận tại trạm đo Kỳ Phong



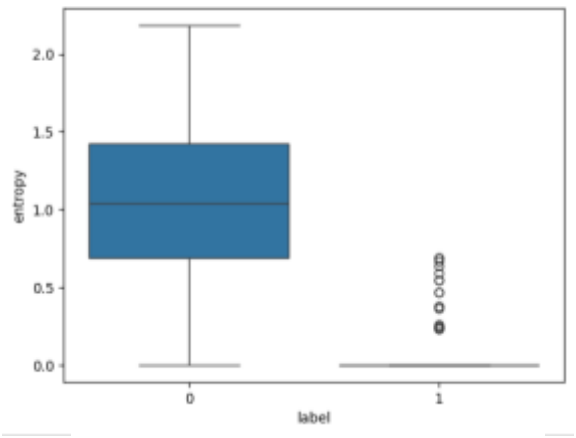
Hình 3.4 Biểu đồ lượng mưa bình thường được ghi nhận trại trạm đo Kỳ Bắc

Dựa vào Hình 3.3 và Hình 3.4 ta thấy lượng mưa bất thường thường dao động với biên độ rất nhẹ và thường trùng lặp nhau trong môt khoảng thời gian nhất định. Ngược lại, ở mẫu đo bình thường có giá trị biên độ lớn và dao động mang tính ngẫu nhiên. Dựa vào sự khác biệt này, nhóm đã đưa ra những đặc trưng làm nổi bật những điểm dữ liệu bất thường trong bộ dữ liệu.

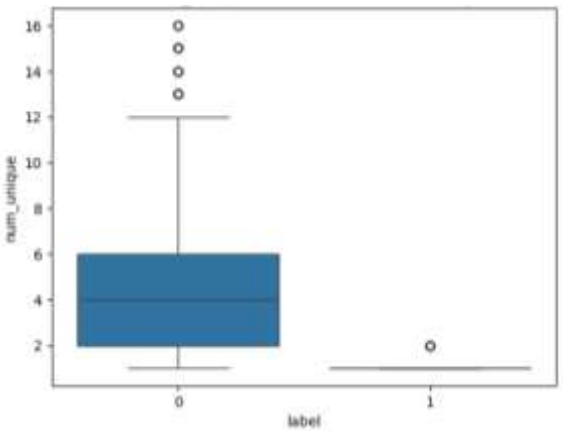
3.2.2. Tiền xử lý và trích xuất đặc trưng

Sau khi dữ liệu đã được chuẩn hóa về thời gian chuẩn datetime64 và kiểu dữ liệu số thực chuẩn float64, các tập dữ liệu được áp dụng cửa sổ trượt để trích xuất đặc trưng. Ở bài toán phát hiện bất thường, sau nhiều quá trình thử và quan sát, nhóm nhận thấy các đặc trưng thường dùng ở các bài nghiên cứu chịu ảnh hưởng nhiều bởi các giá trị ngoại lai (trường hợp ở đây là giá trị cao bất thường), điều này khiến các mô hình học không giám sát sẽ tập trung nhiều các giá trị này. Vì vậy, sau nhiều lần thử và đánh giá,

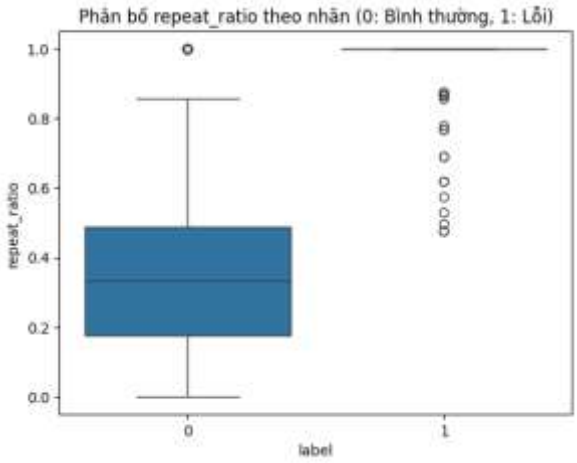
nhóm đã đưa ra được các đặc trưng ít bị ảnh hưởng, phân bố các đặc trưng được quan sát thông qua biểu đồ hộp (Box plot / Box and Whisker plot).



(a) Phân bố đặc trưng entrophy



(b) Phân bố đặc trưng num_unique



(c) Phân bố đặc trưng repeat_raito

Hình 3.5 Biểu đồ phân bố các giá trị đặc trưng của hai loại dữ liệu bình thường (0) – lỗi (1)

Đặc trưng về mức độ hỗn loạn – entrophy

Entrophy là đại lượng đo mức độ hỗn loạn hoặc không chắc chắn trong phân phối tín hiệu. Các giá càng ít thay đổi thì giá trị entrophy càng thấp. Dựa vào Hình 3.5 (a), ta thấy:

Đối với dữ liệu bình thường (label = 0), các giá trị entropy có phân bố rộng (0.0:2.0) với trung vị nằm ở khoảng (1.1:1.2). Điều này cho thấy ở những đoạn mưa bình thường, các giá trị lượng mưa có xu hướng biến động nhiều và không đều dẫn đến mức độ hỗn loạn (entrophy) cao.

Đối với dữ liệu bất thường (label = 1), các giá trị entrophy có phân bố tập trung rất thấp (hầu hết 0.0), với giá trị trung vị thấp hơn đang kể (0.0:0.1). Điều này cho thấy ở những đoạn bất thường, các giá trị mưa rất ít khi thay đổi. Ngoài ra, còn có một số giá

trị ngoại lai xuất hiện (0.3:0.7) nhưng vẫn thấp hơn so với dữ liệu bình thường. Những ngoại lệ này do một vài đoạn dữ liệu lỗi có biến động nhẹ trong tín hiệu.

Đặc trưng về số lượng giá trị khác nhau trong chuỗi – num_unique

Num_unique là đặc trưng dùng để kiểm tra xem trong một đoạn tí hiệu có bao nhiêu giá trị thực sự khác nhau. Càng nhiều giá trị giống nhau trong một đoạn dữ liệu thì giá trị đặc trưng num_unique càng thấp. Dựa vào Hình 3.5 (b), ta thấy:

Đối với dữ liệu bình thường (label = 0), các giá trị num_unique có phân bố rộng (1:12), với trung vị nằm gần giá trị 4. Ngoài ra, có nhiều giá trị ngoại lai (outlier) trong khoảng giá tị (13:16), cho thấy một số đoạn tín hiệu có số lượng giá trị khác nhau vượt trội, phản ánh tính đa dạng trong giá trị bình thường.

Đối với dữ liệu bất thường (label = 1), các giá trị num_unique tập trung ở mức thấp (hầu hết là 1), gần như không có sự phân tán. Đây là đặc trưng điển hình của lỗi tắc phễu, cảm biến chỉ ghi nhận một giá trị lặp lại trong thời gian dài. Ngoài ra, ngoại lệ duy nhất là một giá trị ngoại lai bằng 2, cho thấy có đoạn tín hiệu lỗi nhưng chỉ ghi nhận hai mức giá trị khác nhau.

Đặc trưng về tỷ lệ cặp giá trị giá trị liên tiếp lặp lại trong chuỗi – repeat_ratio

Reapeat_ratio là đặc trưng đo tỷ lệ số cặp giá trị liên tiếp trong chuỗi có sự giống nhau tuyệt đối, các giá trị liên tiếp càng giống nhau thì giá trị repeat_ratia có giá trị càng cao. Dựa vào hình 3.5 (c), ta thấy:

Đối với dữ liệu bình thường (label = 0), các giá trị repeat_ratio có phân bố trong khoảng (0.0:0.9), với trung vị nằm khoảng (0.25:0.30). Phân bố rộng và nghiêng về phía thấp cho thấy tín hiệu bình thường có xu hướng thay đổi nhiều, ít giá trị liên tiếp lặp lại.

Đối với dữ liệu bất thường (label = 1), các giá trị repeat_ratio có phân bố cao hơn đáng kể, dao động trong khoảng (0.9:1.0), với trung vị gần như bằng 1. Điều này cho thấy ở các đoạn tín hiệu lỗi, các giá trị lặp lại nhiều, tín hiệu đều đặn hơn, phù hợp với đặc trưng của tắc phễu. Ngoài ra, một số giá trị ngoại lai thấp hơn (0.3:0.8), cho thấy có một vài đoạn lỗi có sự thay đổi nhẹ.

3.2.3. Huấn luyện mô hình

Các đặc trưng sau khi được trích xuất được chuẩn hóa bằng phương pháp RobustScaler để loại bỏ giá trị ngoại lai, tránh trường hợp mô hình đánh giá quá cao hoặc quá thấp tầm quan trọng của một số đặc trưng do tồn tại giá trị ngoại lệ. Sau khi chuẩn hóa, các đặc trưng sẽ gộp lại thành các véc-tơ đặc trưng, trong đó mỗi véc-tơ sẽ chứa ba giá trị đặc trưng được trích xuất từ một cửa sổ trượt.

Sau đó, nhóm tiến hành huấn luyện và so sánh nhiều mô hình học máy không giám sát được để xuất từ các bài báo ở Chương 2 như K – means Clustering, Isolation Forest, Local Outlier Factor. Để hiểu rõ cơ chế hoạt động cũng như lý do vì sao từng mô hình

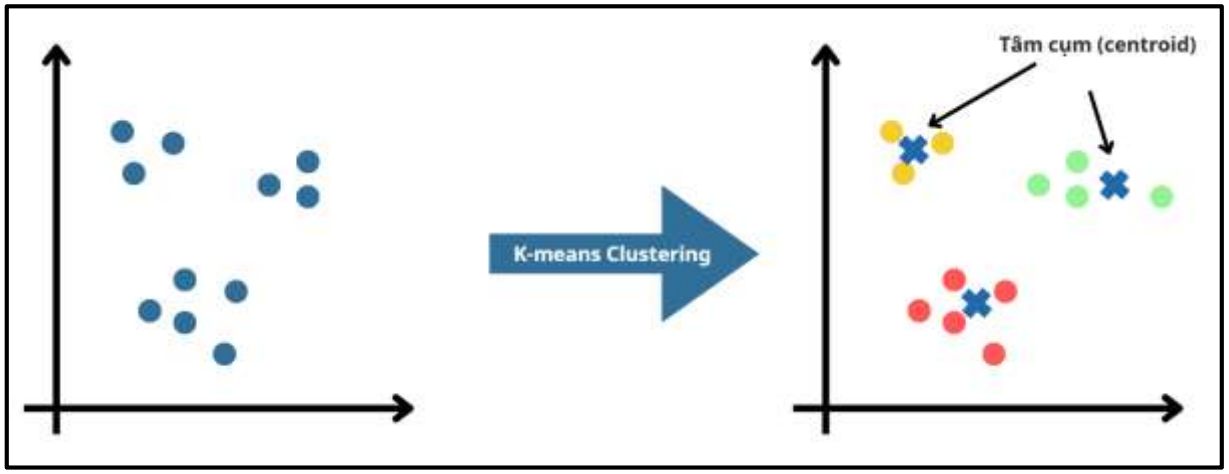
có thể phù hợp hoặc không phù hợp với bài toán này, nhóm sẽ trình bày nguyên lý hoạt động chính của các mô hình được sử dụng.

Cuối cùng, để đơn giản trong việc giải thích, các véc-tơ đặc trưng sẽ được hiểu là những điểm dữ liệu đưa vào mô hình.

a) Thuật toán K – means

K-Means Clustering là một thuật toán hoạt động dựa trên nguyên lý phân cụm. Mục tiêu chính của K-Means là chia một tập hợp các điểm dữ liệu thành k cụm (clusters) sao cho các điểm trong cùng một cụm có sự tương đồng cao nhất với nhau và khác biệt tối đa với các cụm khác.

Trong thuật toán K-Means, mỗi cụm được đại diện bởi một centroid (trung tâm cụm), là giá trị trung bình của các điểm dữ liệu trong cụm đó. Thuật toán sẽ liên tục điều chỉnh các centroid và tái phân cụm các điểm dữ liệu cho đến khi đạt được sự hội tụ – các điểm dữ liệu không còn thay đổi cụm hay các centroid không còn thay đổi giá trị đáng kể sau mỗi vòng lặp.



Hình 3.6 Minh họa thuật toán K-Means với nguyên lý phân cụm

Nguyên lý hoạt động của thuật toán qua các bước sau:

Bước 1: Khởi tạo các trung tâm cụm (Centroids)

Xác định số lượng cụm k. Sau đó, chọn ngẫu nhiên k điểm từ tập dữ liệu làm các trung tâm cụm ban đầu, được gọi là các centroid.

Bước 2: Gán mỗi điểm dữ liệu vào cụm gần nhất

Với mỗi điểm dữ liệu, ta tính khoảng cách của nó tới các trung tâm cụm bằng khoảng cách Ơ-clit (Euclidean). Khoảng cách giữa một điểm dữ liệu x_i và một centroid c_j được tính bằng công thức:

$$d(x_i, c_j) = \sqrt{\sum_{k=1}^{n} (x_{ik} - c_{jk})^2} \quad (3.5)$$

Trong đó:

- x_i: là điểm dữ liệu thứ i.
- c_j: là centroid của cụm thứ j.
- n: là số chiều của dữ liệu.

Bước 3: Cập nhật các trung tâm cụm

Sau khi tất cả các điểm dữ liệu đã được gán vào một cụm, các centroid được tính lại bằng cách lấy trung bình cộng các điểm dữ liệu trong mỗi cụm:

$$c_j = \frac{1}{|C_j|} \sum_{x \in C_j} x_i \quad (3.6)$$

Trong đó:

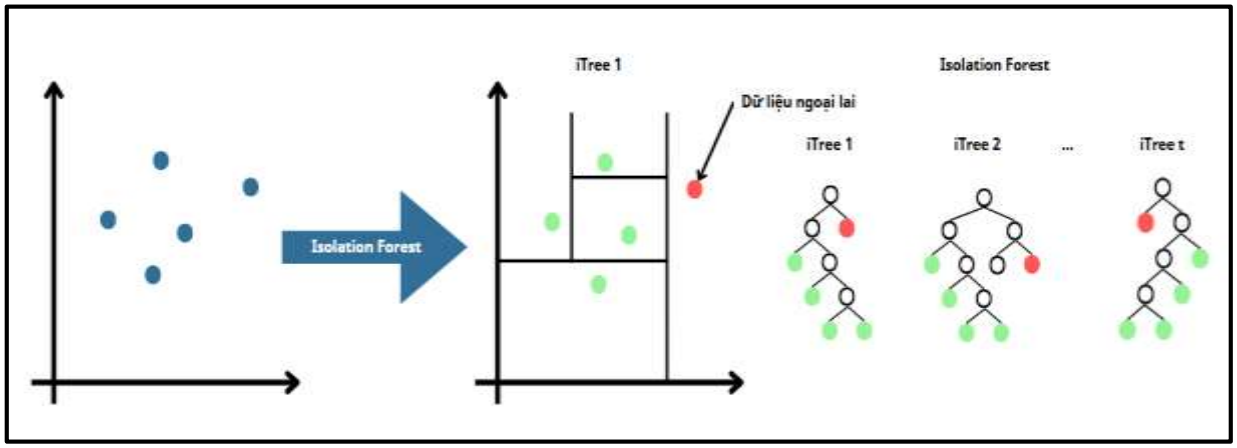
- C_j: là tập hợp các điểm dữ liệu trong cụm j.
- $|C_j|$: là số lượng điểm dữ liệu trong cụm j.

Bước 4: Lặp lại quá trình

Các bước gán cụm và cập nhật centroid được lặp lại cho đến khi các centroid không còn thay đổi đáng kể hoặc đạt đến số lần lặp tối đa. Thuật toán hội tụ khi không có sự thay đổi trong gán cụm của các điểm dữ liệu hoặc sự thay đổi rất nhỏ.

b) Thuật toán Isolation Forest

Isolation Forest (iForest) là một thuật toán học máy không giám sát (unsupervised learning), chuyên dụng cho bài toán phát hiện bất thường (anomaly detection). Khác với các phương pháp dựa trên mật độ (density-based) hoặc khoảng cách (distance-based), iForest dựa trên nguyên lý cách ly (isolation) điểm dữ liệu trong không gian đặc trưng.



Hình 3.7 Minh họa thuật toán Isolation Forest với nguyên lý cách ly điểm dữ liệu

Thuật toán hoạt động dựa trên các bước chính:

- Input: Tập dữ liệu $X = \{x_1, x_2, \dots, x_n\}$, số cây iTree là t.
- Output: Điểm bất thường $s(x, n)$ cho mỗi điểm x.

Bước 1: Xây dựng các cây Isolation (iTree)

Mỗi iTree được xây dựng bằng cách:

- Chọn ngẫu nhiên một tập con dữ liệu, rồi tiếp tục chọn ngẫu nhiên một đặc trưng, một giá trị phân tách (split value) trong khoảng (min,max) của thuộc tính đó.
- Tách dữ liệu thành hai nhánh:
  - Nhánh trái: Các điểm có giá trị ≤ split value.
  - Nhánh phải: Các điểm có giá trị > split value.
- Lặp lại cho đến khi: Mọi điểm bị cô lập, mỗi điểm nằm ở một lá.

Bước 2: Tính độ sâu trung bình $E(h(x))$

$h(x)$: Số lần phân chia để cô lập x trong một cây iTree (độ sâu).

$E\big(h(x)\big)$: Độ sâu trung bình của x qua tất cả các cây iTree.

Bước 3: Tính hằng số chuẩn hóa $c(n)$ để chuẩn hóa độ sâu khi so sánh giữa các tập dữ liệu có kích thước khác nhau

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n}, H(k) \approx \ln(k) + 0.5772 \qquad (3.7)$$

Trong đó:

- $H(k)$: Số điều hòa (Harmonic number)
- n: Số lượng điểm trong tập dữ liệu

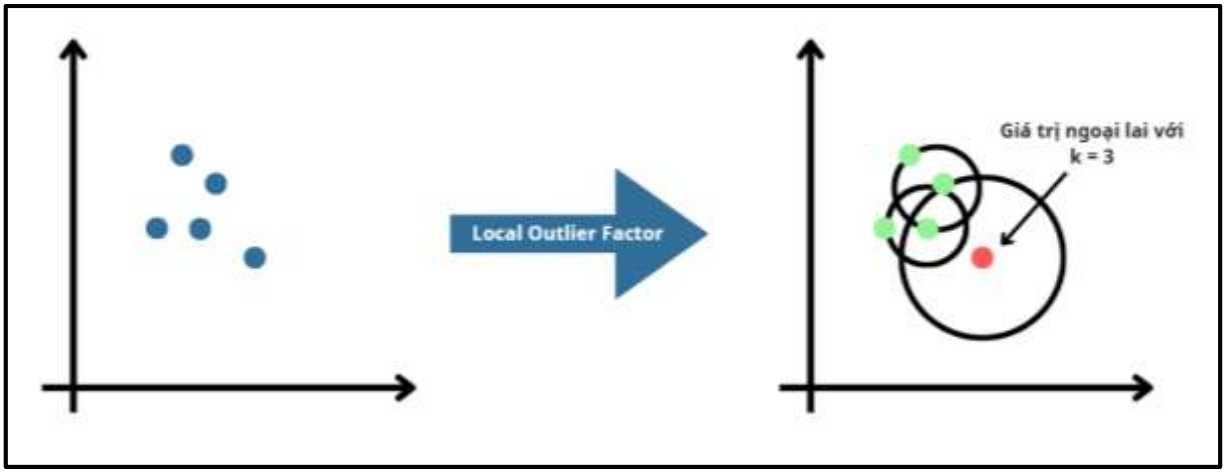
Bước 4: Tính điểm bất thường $s(x, n)$

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}} \qquad (3.8)$$

Nếu $E\big(h(x)\big) \ll c(n) \rightarrow\ s(x, n) \approx 1$ → Điểm bất thường (cách ly nhanh, độ sâu thấp).

Nếu $E\big(h(x)\big) \gg c(n) \rightarrow\ s(x, n) \approx 0$ → Điểm bình thường (cách ly chậm, độ sâu cao).

c) Thuật toán Local Outlier Factor



Hình 3.8 Minh họa thuật toán Local Outlier Factor với nguyên lý so sánh mật độ lân cận

Thuật toán Local Outlier Factor (LOF) sử dụng các điểm dữ liệu lân cận để phát hiện ngoại lệ. LOF hoạt động dựa trên nguyên tắc các điểm dữ liệu bình thường có mật độ phân bố dày đặc, các điểm dữ liệu ngoại lai có mật độ thấp hơn và cách xa các điểm khác. Thuật toán LOF gồm các bước:

Bước 1: Xác định tập lân cận gần nhất $N_k(A)$

Với mỗi điểm dữ liệu A, thuật toán xác định tập k điểm dữ liệu lân cận gần nhất, ký hiệu là $N_k(A)$ dựa vào khoảng cách Ơ-clit (Eclidean). Trong bài toán này, dữ liệu đưa vào mô hình là véc-tơ đặc trưng ba chiều nên công thức sẽ là:

$$d(A,B) = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2 + (z_A - z_B)^2} \quad (3.9)$$

Trong đó:

- $d(A,B)$: khoảng cách Ơ-clit từ véc-tơ A đến véc-tơ B
- x_A, y_A, z_A: ba giá trị đặc trưng được trích xuất từ cửa sổ A.
- x_B, y_B, z_B: ba giá trị đặc trưng được trích xuất từ cửa sổ B.

Bước 2: Tính khoảng cách khả đạt (Reachability Distance)

Khoảng cách khả đạt dùng để thay thế cho khoảng cách Ơ-clit trong trường hợp các điểm dữ liệu trong vùng thưa quá gần nhau khiến cho giá trị mật độ địa phương khả đạt tính ở Bước 3 lớn, thuật toán sẽ hiểu đây mật độ cục bộ tại điểm dữ liệu đó lớn.

Với hai điểm dữ liệu A và B, khoảng cách khả đạt từ điểm A đến điểm B được xác định như sau:

$$RD_k(A,B) = \max\big(d_k(B), d(A,B)\big) \quad (3.10)$$

Trong đó:

- $RD_k(A,B)$: khoảng cách khả đạt từ điểm A đến điểm B.

- $d(A,B)$: khoảng cách Ơ-clit giữa điểm A và điểm B
- $d_k(B)$: khoảng cách từ điểm B đến điểm lân cận xa nhất trong tập $N_k(B)$.

Bước 3: Tính mật độ cục bộ khả đạt (Local Reachability Density – LRD)

Mật độ cục bộ khả đạt của một điểm được định nghĩa là nghịch đảo trung bình của các khoảng cách khả đạt tới các điểm lân cận gần nhất, giá trị này càng lớn thì điểm A càng gần tập $N_k(A)$. Mật độ cục bộ khả đạt của điểm A được xác định:

$$LRD_k(A) = \frac{|N_k(A)|}{\sum_{B \in N_k(A)} RD_k(A,B)} \tag{3.11}$$

Trong đó:

- $LRD_k(A,B)$: mật độ khả đạt của điểm A trong tập $N_k(A)$.
- $|N_k(A)|$: số điểm dữ liệu trong tập $N_k(A)$.

Bước 4: Tính hệ số bất thường cục bộ (Local Outlier Factor – LOF)

Cuối cùng, điểm A có phải ngoại lệ hay không được xác định bằng hệ số bất thường cục bộ. Hệ số này được tính bằng cách so sánh LDR của A với LDR của các điểm lân cận của nó:

$$LOF_k(A) = \frac{1}{|N_k(A)|} * \sum_{B \in N_k(A)} \frac{LRD_k(B)}{LRD_k(A)} \tag{3.12}$$

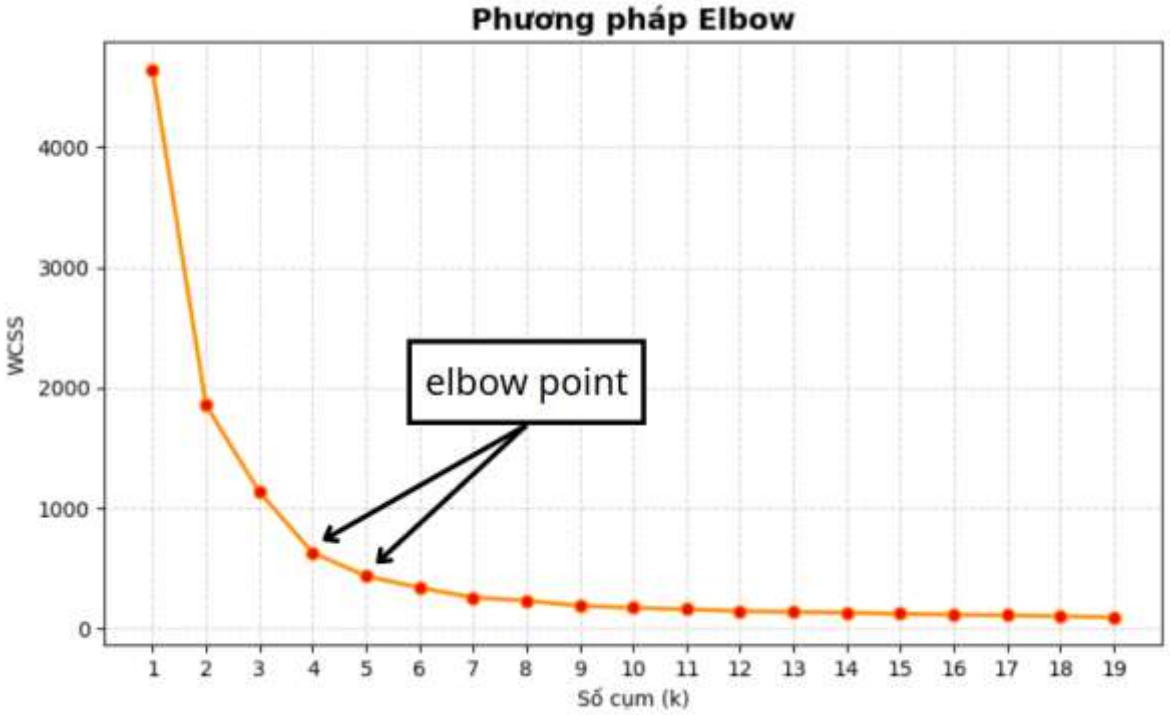
Trong đó:

- $LOF_k(A,B)$: hệ số bất thường cục bộ của điểm A trong tập $N_k(A)$.

Hệ số này được đánh giá như sau:

- $LOF_k(A) \leq 1$: điểm A có mật độ cao hơn hoặc bằng các điểm lân cận → bình thường.
- $LOF_k(A) > 1$: điểm A có mật độ thấp hơn các điển lân cận → bất thường.

d) Chọn thông số tối ưu cho mô hình



Hình 3.9 Biểu đồ phương pháp Elbow xác định số cụm tối ưu cho mô hình K-means

Đối với mô hình sử dụng thuật toán K-means Clustering, ta sẽ quan tâm đến thông số k để chia cụm hiệu quả. Nhóm sử dụng phương pháp Elbow để lựa chọn số cụm k tối ưu. Phương pháp này dựa trên việc quan sát độ biến thiên của tổng bình phương bình phương khoảng giữa mỗi điểm và tâm điểm cụm của nó (WCSS) để xác định điểm khủy tay (elbow point) – nơi mà việc tăng số cụm không giúp giảm độ phân tán trong cụm. Giá trị WCSS được tính theo công thức:

$$WCSS = \sum_{i=1}^{k} \sum_{j=1}^{n_i} d\left(x_i^{(j)}, c_j\right)^2 \tag{3.13}$$

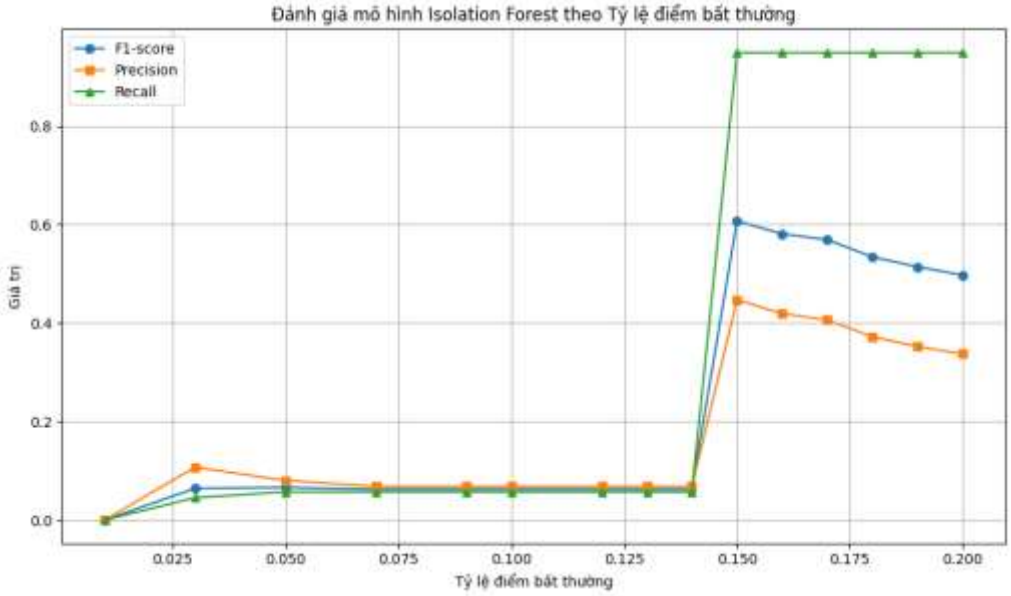
Trong đó:

- WCSS : Tổng bình phương khoảng cách giữa mỗi điểm và tâm điểm cụm của nó.
- Khoảng cách $d\left(x_i^{(j)}, c_j\right)^2$ biểu thị khoảng cách từ điểm dữ liệu thứ $x_i^{(j)}$ trong cụm j tới tâm cụm c_j.
- k: Số cụm.
- n_i: Số điểm trong cụm thứ i.

Dựa vào Hình 3.9 ta có kết quả:

- Khi số cụm tăng từ $k = 1$ đến $k = 3$, giá trị WCSS giảm mạnh cho thấy khi cụm tăng việc phân tách dữ liệu hiệu quả hơn.

- Điểm khủy tay xuất hiện khi $k = 4$ và $k = 5$, lúc này giá trị WCSS không còn giảm đáng kể. Ở bài toán này, số cụm được chọn là 5.



Hình 3.10 Ảnh hưởng của tham số tỷ lệ điểm bất thường đến các chỉ số đánh giá trong mô hình Isolation Forest

Đối với mô hình sử dụng thuật toán Isolation Forest, nhóm sử dụng biểu đồ Hình 3.10 để đánh giá các chỉ số (Precison, Recall, F1-score) theo từng giá trị của thông số về tỷ lệ điểm bất thường (contamination) để chọn ra thông số tối ưu cho mô hình. Dựa vào Hình 3.10 ta thấy khi giá trị tỷ lệ điểm bất thường đạt 0.15, các chỉ số đánh giá Precision, Recall và F1-score của mô hình Isolation Forest đạt đỉnh. Lúc này việc tăng tỷ lệ điểm bất thường cho mô hình sẽ không hiệu quả, thậm chí khiến Precision và F1-score suy giảm. Thông số về số cây iTree sẽ cố định t = 100 để cân bằng giữa độ chính xác và tốc độ.

Tương tự cho trường hợp mô hình sử dụng thuật toán Local Outlier Factor, nhóm cũng sử dụng biểu đồ cho hai thông số là số điểm dữ liệu lân cận k và giá trị tỷ lệ điểm bất thường, kết quả đạt được lần lượt là 45 và 0.03.

3.2.4. Kết quả đánh giá và nguyên nhân

a) Kết quả đánh giá với các chỉ số Precison – Recall – F1-score

Các mô hình học không giám sát được huấn luyện với thông số tốt nhất để chọn ra mô hình có chỉ số tốt nhất. Với bài toán phát hiện bất thường tại biên, chỉ số Recall được ưu tiên cao, với mục tiêu hạn chế bỏ sót các điểm bất thường với độ chính sát chấp nhận được (> 70%).

Bảng 3.1 Kết quả đánh giá ba mô hình trên các chỉ số đối với lớp bất thường

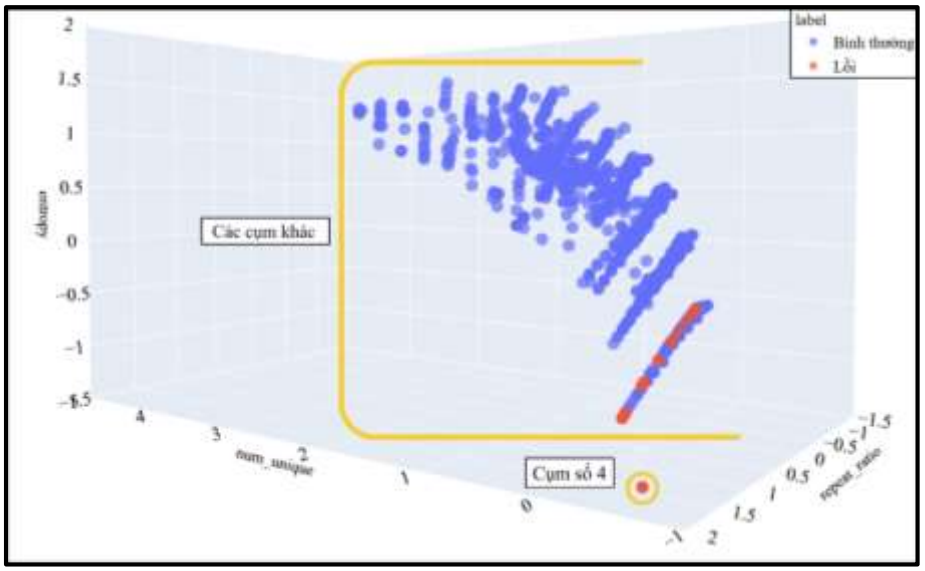
Mô hình	**Precision (1)**	**Recall (1)**	**F1-score (1)**
K-means Clustering	74%	93%	82%
Isolation Forest	45%	95%	61%
Local Outlier Factor	14%	5.7%	8%

Kết quả cho thấy mô hình K-means Clustering đạt hiệu suất cao nhất trong ba mô hình được so sánh, với Precision 74%, Recall 93%, và F1-score 82%. Điều này cho thấy mô hình không chỉ phát hiện được hầu hết các điểm bất thường (chỉ số Recall cao), mà còn giữ được độ chính xác tương đối tốt khi dự đoán điểm bất thường (chỉ số Precision cao hơn so với các mô hình còn lại).

Trong khi đó, Isolation Forest cũng đạt chỉ số Recall khá cao (95%) nhưng chỉ số Precision chỉ đạt 45%, cho thấy mô hình phát hiện gần như tất cả điểm bất thường, nhưng cũng tạo ra nhiều cảnh báo sai (false positives).

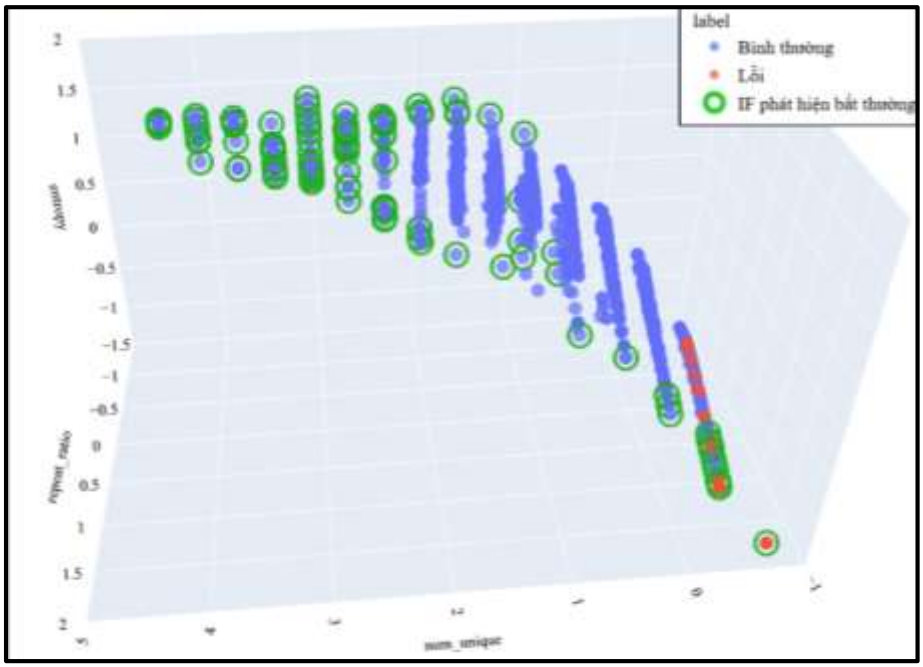
Ngược lại, Local Outlier Factor (LOF) có kết quả khá kém với các chỉ số Precision 14%, Recall 5.7%, và F1-score 8%, cho thấy mô hình này không phù hợp với đặc trưng của dữ liệu.

Để giải thích cho các kết quả trên, nhóm sẽ dùng đồ thị phân bố véc-tơ đặc trưng trong không gian 3 chiều để quan sát sự phân bố của các véc-tơ và các kết quả của các mô hình sau khi được áp dụng. Tuy nhiên, để đơn giản trong việc giải thích, ta sẽ gọi một véc-tơ đặc trưng là một điểm dữ liệu.



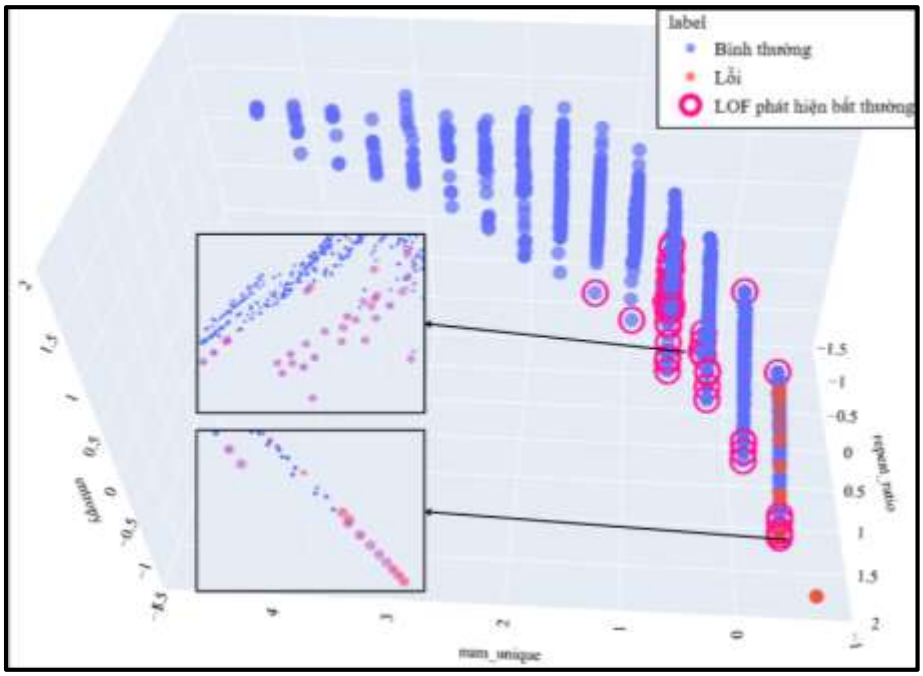
Hình 3.11 Đồ thị phân bố véc-tơ đặc trưng và kết quả phân cụm bằng mô hình K-means Clustering

Dựa vào Hình 3.11, ta thấy các điểm dữ liệu bị lỗi có các giá trị đặc trưng tách biệt so với các điểm dữ liệu bình thường và hầu như giống nhau. Vì vậy các dữ liệu này tập trung thành một cụm tại Cụm số 4 khá tách biệt so với những dữ liệu còn lại. Khi K-means thực hiện phân cụm, thuật toán sẽ tách những điểm giá trị này thành một cụm riêng. Điều này giúp cho mô hình đạt Recall cao mà vẫn giữ Precision tốt. Ngoài ra, ta từ Hình 3.10 ta thấy có vài điểm dữ liệu lỗi mang giá trị ngoại lai nằm ngoài cụm số 4, nguyên nhân là các điểm này mặc dù được trích từ các đoạn dữ liệu có dấu hiệu tắc nhưng có chứa dao động nhẹ. Tuy nhiên, các điểm này chiếm tỉ lệ rất ít (7%) trong tổng số lỗi nên không ảnh hưởng nhiều đến kết quả của mô hình.



Hình 3.12 Đồ thị phân bố véc-tơ đặc trưng và kết quả nhận diện điểm bất thường bằng mô hình Isolation Forest

Quan sát từ Hình 3.12 ta thấy mô hình Isolation Forest có khả năng phát hiện tốt các điểm nằm tách biệt khỏi phân bố chung bằng cách xây dựng nhiều cây nhị phân để cô lập từng điểm. Nhờ đó, các điểm lỗi có xu hướng được đánh dấu chính xác là bất thường. Tuy nhiên, do dữ liệu bình thường có phân bố rộng và có nhiều điểm thưa tại vùng biên rất dễ phân tách nên mô hình đánh dấu nhiều điểm biên của cụm bình thường là bất thường, dẫn đến số lượng báo động giả cao. Vì vậy, mặc dù Isolation Forest có khả năng bao phủ gần như toàn bộ lỗi, nhưng chỉ số Precison lại rất thấp.



Hình 3.13 Đồ thị phân bố véc-tơ đặc trưng và kết quả nhận diện điểm bất thường bằng mô hình Local Outlier Factor

Từ Hình 3.13 có thể thấy mô hình sử dụng thuật toán Local Outlier Factor chỉ bắt được các điểm dữ liệu lỗi có giá trị ngoại lai nằm xa cụm lỗi chính và bắt nhầm các điểm ở những vùng có mật độ điểm dữ liệu thấp. Nguyên nhân chính là LOF hoạt động dựa trên nguyên tắc so sánh mật độ cục bộ, các điểm dữ liệu có mật độ thấp hay càng xa một cụm dữ liệu lân cận sẽ được coi là điểm dữ liệu bất thường. Tuy nhiên, trong bài toán này, các điểm dữ liệu lỗi hầu hết nằm trong vùng khá đồng nhất và gần nhau về khoảng cách, dẫn đến LOF không xem chúng là bất thường. Thay vào đó, những điểm ở các vùng mật độ thấp hoặc lệch khỏi các cụm dữ liệu sẽ bị đánh dấu sai là bất thường.

b) Kết quả về thời gian dự đoán

Bảng 3.2 So sánh thời gian dự đoán của các mô hình học không giám sát.

Mô hình	**Thời gian dự đoán (s)**
K-means Clustering	0.039
Isolation Forest	0.438
Local Outlier Factor	0.098

Từ Bảng 3.2 cho thấy mô hình K-means Clustering có thời gian dự đoán nhanh nhất, chỉ mất 0.039 giây, nhờ vào cơ chế gán nhãn cụm đơn giản bằng khoảng cách đến tâm cụm. Local Outlier Factor (LOF) có thời gian dự đoán trung bình (0.098 giây) chậm hơn K-means do phải tính toán mật độ cục bộ và khoảng cách giữa các điểm lân cận. Trong khi đó, Isolation Forest có thời gian dự đoán lâu nhất (0.438 giây), do phải thực hiện quá trình đánh giá nhiều cây phân tách trong rừng ngẫu nhiên.

Mặc dù sự khác biệt về thời gian tính toán chưa quá lớn, kết quả này cho thấy K-means Clustering không chỉ đạt hiệu suất dự đoán cao nhất về độ chính xác, mà còn có ưu thế rõ rệt về tốc độ, đặc biệt phù hợp với các ứng dụng tại thiết bị biên (Edge) vốn bị giới hạn tài nguyên xử lý và có yêu cầu cao về tốc độ.

c) Kết quả dự đoán của mô hình K-means Clustering

Hình 3.14 Kết quả dự đoán của mô hình K-means Clustering tại một trạm lỗi

Dựa hình hình 3.14 ta thấy mặc dù mô hình đã dự đoán đúng vị trí lỗi (được đánh dấu bằng khung màu xanh lá) nhưng vẫn tồn tại nhiều đoạn bị đánh nhầm là bất thường. Nguyên nhân là mô hình không thể phát hiện được dữ liệu lỗi do tắc phễu và dữ liệu tạo bởi mưa ngâu (mưa nhỏ kéo dài), vốn có đặc trưng giống nhau. Điều này khiến cho chỉ số Recall cao nhưng chỉ số Precision rất thấp vì tạo nhiều cảnh báo sai.

3.3. Bài toán phân tích trình trạng lỗi trên Cloud

3.3.1. Dữ liệu

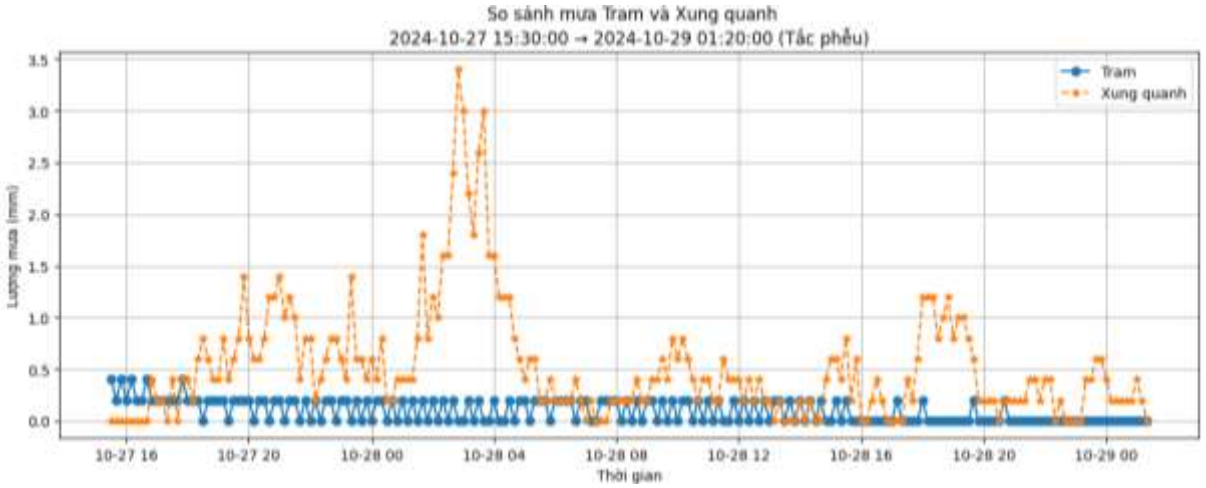
Dữ liệu được thu thập từ các trạm đo mưa của Vrain cung cấp, bao gồm hai trường thông tin chính: lượng mưa đo được tại trạm chính (Tram) và tại các khu vực xung quanh (Xung quanh), việc phải lấy dữ liệu của các trạm xung quanh để giúp việc đánh giá chính xác lỗi tắc phễu. Ngoài ra, tập dữ liệu được gắn nhãn thủ công với trường fault, trong đó:

- fault = 1 đại diện cho các đoạn dữ liệu nghi ngờ có hiện tượng tắc phễu.
- fault = 0 thể hiện trạng thái hoạt động bình thường.

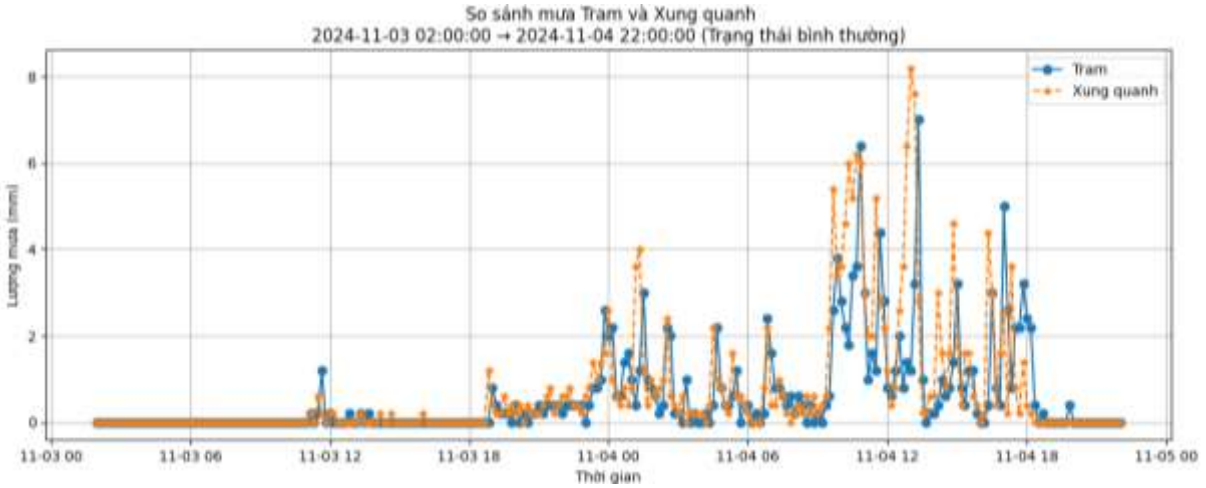
datetime	Tram	Xung quanh	fault
10/26/2024 22:40	0	0	0
10/26/2024 22:50	0	0	0
10/26/2024 23:00	0	0	0
10/26/2024 23:10	0	0	0
10/26/2024 23:20	0	0.4	0
10/26/2024 23:30	0.4	0.2	0
10/26/2024 23:40	0	0	0
10/26/2024 23:50	0	0	0
10/27/2024 0:00	0	0	0
10/27/2024 0:10	0	0.4	1
10/27/2024 0:20	0.4	0.4	1
10/27/2024 0:30	0.4	0.6	1
10/27/2024 0:40	0.6	6.2	1
10/27/2024 0:50	0.6	2	1

Hình 3.15 Gắn nhãn dữ liệu thực tế

Để trực quan hóa sự khác biệt giữa hai trạng thái, hai hình bên dưới minh họa một đoạn dữ liệu bình thường và một đoạn có dấu hiệu tắc phễu:

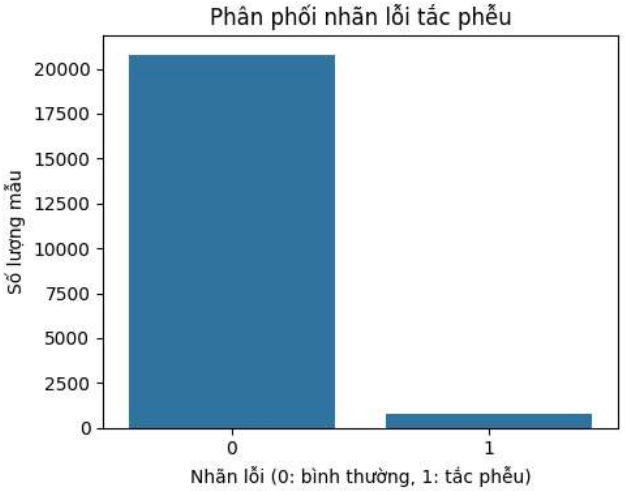


Hình 3.16 Biểu đồ so sánh lượng mưa hai trạm gần nhau trường hợp bình thường



Hình 3.17 Biểu đồ so sánh lượng mưa hai trạm gần nhau trong trường hợp tắc phễu

Dữ liệu đầu vào có định dạng CSV, trong đó trường datetime được chuẩn hóa về kiểu thời gian (datetime64) để phục vụ xử lý chuỗi thời gian. Các thao tác khám phá dữ liệu ban đầu cho thấy sự mất cân bằng đáng kể giữa hai nhãn, với tỷ lệ mẫu có lỗi thường thấp hơn 10%. Việc này đặt ra yêu cầu cần áp dụng các kỹ thuật cân bằng dữ liệu trong giai đoạn huấn luyện.



Hình 3.18 Biểu đồ mô tả sự mất cân bằng của hai nhãn

3.3.2. Tiền xử lý và trích xuất đặc trưng

Trong các bài toán học máy nói chung và phân tích chuỗi thời gian nói riêng, chất lượng dữ liệu đầu vào đóng vai trò then chốt trong việc đảm bảo hiệu năng và độ tin cậy của mô hình dự đoán. Dữ liệu thô thu thập từ thực địa thường không đồng nhất, có thể chứa nhiễu, giá trị thiếu hoặc sai lệch về định dạng thời gian. Đặc biệt, dữ liệu chuỗi thời gian yêu cầu các bước xử lý đặc thù như chuẩn hóa mốc thời gian, chia cửa sổ quan sát hợp lý và thống nhất đơn vị đo lường. Với bài toán này, quy trình tiền xử lý không chỉ nhằm làm sạch dữ liệu mà còn hướng tới việc trích xuất các đặc trưng quan trọng dưới dạng chuỗi thời gian. Bởi hiện tượng tắc phễu không nhất thiết xảy ra tại một thời điểm cụ thể mà còn kéo dài, kèm theo các biểu hiện tắc nhẹ trước khi tắc phễu hoàn toàn. Do đó, việc chuyển đổi dữ liệu thô thành dạng phù hợp, có thể khai thác hiệu quả cho mô hình học máy là vô cùng cần thiết.

Quá trình tiền xử lý và trích xuất các đặc trưng bao gồm các bước chính sau:

Bước 1: Chuẩn hóa dữ liệu thời gian: Trường datetime được chuyển về định dạng thời gian chuẩn, sau đó được đặt làm chỉ số (index) nhằm hỗ trợ xử lý theo chuỗi thời gian.

Bước 2: Kiểm tra và xử lý dữ liệu lỗi: Các dòng dữ liệu thiếu (NaN) ở các trường quan trọng (Tram, Xung quanh) bị loại bỏ để đảm bảo độ chính xác khi tính toán đặc trưng.

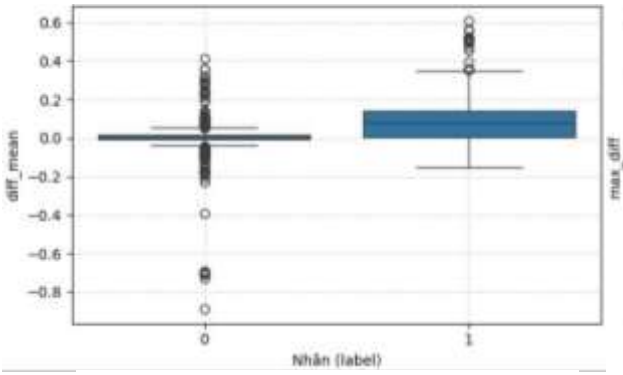
Bước 3: Trích xuất đặc trưng theo cửa sổ thời gian: Để chuyển dữ liệu thô thành dạng có thể khai thác hiệu quả cho mô hình học máy, dữ liệu chuỗi thời gian được chia thành các cửa sổ cố định. Mỗi cửa sổ chứa nhiều mẫu dữ liệu liên tiếp, từ đó trích xuất các đặc trưng thống kê đại diện cho hành vi mưa trong khoảng thời gian đó. Đồng thời,

nhãn được gán cho mỗi cửa sổ theo nguyên tắc đa số: nếu ≥ 50% số mẫu trong cửa sổ có nhãn fault = 1, thì cửa sổ đó được gán nhãn lỗi.

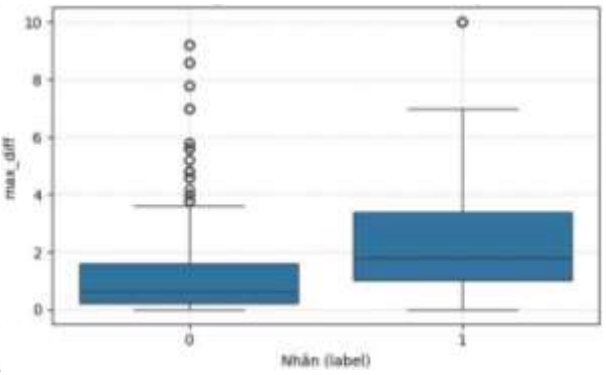
Sau khi chia cửa sổ, các đặc trưng thống kê sau được tính toán cho mỗi cửa sổ:

- **mean_A**: Trung bình lượng mưa tại trạm chính A. Phản ánh mức độ mưa tổng thể tại trạm, có thể phân biệt thời kỳ mưa rõ rệt với các giai đoạn khô hạn.
- **std_A**: Độ lệch chuẩn tại trạm A. Thể hiện mức độ dao động trong cửa sổ; giá trị thấp có thể là dấu hiệu trạm không ghi nhận đúng do tắc phễu.
- **mean_B**: Trung bình lượng mưa tại các trạm xung quanh. Dùng để so sánh với mean_A, giúp phát hiện sự không đồng bộ trong khu vực.
- **std_B**: Độ lệch chuẩn tại các trạm B. Cho biết mức biến thiên của mưa trong vùng lân cận, hỗ trợ đánh giá tính ổn định của hiện tượng mưa.
- **max_A, min_A**: Giá trị cực trị tại trạm A. Giúp xác định cường độ mưa bất thường hoặc giai đoạn hoàn toàn không mưa.
- **max_B, min_B**: Cực trị tại các trạm xung quanh, làm cơ sở so sánh với A để phát hiện bất thường cục bộ.
- **diff_mean**: Trung bình hiệu giữa lượng mưa tại B và A. Đặc trưng này giúp nhận biết nếu trạm A ghi nhận thấp hơn rõ rệt so với khu vực – dấu hiệu quan trọng của tắc phễu.
- **max_diff**: Hiệu lượng mưa lớn nhất giữa B và A. Nhấn mạnh sự chênh lệch cực trị, hữu ích trong phát hiện sai lệch rõ rệt giữa các trạm.
- **slope_A**: Độ dốc xu hướng lượng mưa tại trạm A trong 600 phút. Cho thấy xu hướng tăng/giảm lượng mưa tại trạm chính.
- **slope_B**: Độ dốc tại các trạm xung quanh. So sánh với slope_A giúp xác định sự không đồng bộ về xu hướng – dấu hiệu tiềm năng của lỗi.
- **ratio_A_to_B**: Tỷ lệ tổng lượng mưa giữa A và B. Nếu thấp đáng kể, có thể cho thấy trạm A ghi nhận thiếu – biểu hiện điển hình của tắc phễu.

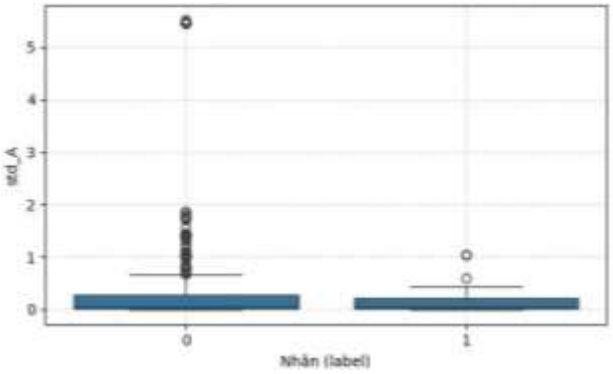
Để đánh giá trực quan khả năng phân tách của các đặc trưng đã lựa chọn, biểu đồ hộp (boxplot) được sử dụng nhằm mô tả sự phân bố của từng đặc trưng theo hai nhóm nhãn: 0 (bình thường) và 1 (lỗi tắc phễu). Biểu đồ cho phép quan sát sự khác biệt về giá trị trung vị, mức độ phân tán và khoảng giá trị giữa hai nhóm. Các đặc trưng có sự chênh lệch rõ ràng về phân bố giữa hai nhãn có tiềm năng cao trong việc hỗ trợ mô hình phát hiện lỗi. Một số boxplot tiêu biểu của các đặc trưng quan trọng được trình bày dưới đây.



(a) Phân bố đặc trưng diff_mean



(b) Phân bố đặc trưng max_diff



(c) Phân bố đặc trưng std_A

Hình 3.20 Biểu đồ phân bố các giá trị đặc trưng của hai nhãn dữ liệu

Đặc trưng diff_mean

Dựa vào hình 3.19 (a) cho thấy sự khác biệt rõ rệt giữa hai nhóm. Trong đó nhóm 0 (bình thường), các giá trị diff_mean phân bố tập trung quanh 0, trung vị gần như bằng 0, IQR rất hẹ, cho thấy sự đồng bộ giữa trạm A và các trạm lân cận. Một vài điểm ngoại biên (outlier) coi như chấp nhận được bởi vì có những lúc trạm này mưa nhưng trạm bên kia không mưa do bán kính của cơn mưa. Ngược lại, nhóm lỗi có phân bố diff_mean lệch hẳn về phía dương, trung vị tăng lên đáng kể (~0.1–0.15), và độ rộng IQR lớn hơn nhiều so với nhóm bình thường, phản ánh sự phân tán cao hơn trong dữ liệu lỗi. Ngoài ra, có nhiều điểm ngoại biên dương với giá trị lên đến 0.6, cho thấy trong một số tình huống lỗi, dữ liệu trạm tắc thấp hơn nhiều so với trạm bình thường. Từ biểu đồ này, thấy rằng đặc trưng diff_mean thể hiện khả năng phân tách giữa dữ liệu bình thường và lỗi, phù hợp để sử dụng làm đầu vào mô hình học máy.

Đặc trưng max_diff

Dựa biểu đồ boxplot ở Hình 3.19 (b) của max_diff, với nhóm bình thường, giá trị có xu hướng thấp, trung vị dưới 1, phần lớn dữ liệu nằm trong khoảng 0 đến 2. Điều này cho thấy trong điều kiện bình thường, mức chênh lệch tối đa giữa trạm A và các trạm lân cận thường nhỏ, phản ánh độ nhất quán cao trong dữ liệu đo mưa. Ngược lại, nhóm

lỗi thể hiện sự khác biệt rõ rệt: phân phối max_diff dịch chuyển mạnh lên phía trên với trung vị lớn hơn (~1.5–2), hộp IQR rộng hơn, và một số điểm ngoại biên lên đến 10. Điều này phản ánh hiện tượng khi có lỗi xảy ra thì sự chênh lệch lớn nhất giữa trạm A và các trạm lân cận tăng đột biến, thậm chí gấp nhiều lần so với điều kiện bình thường.

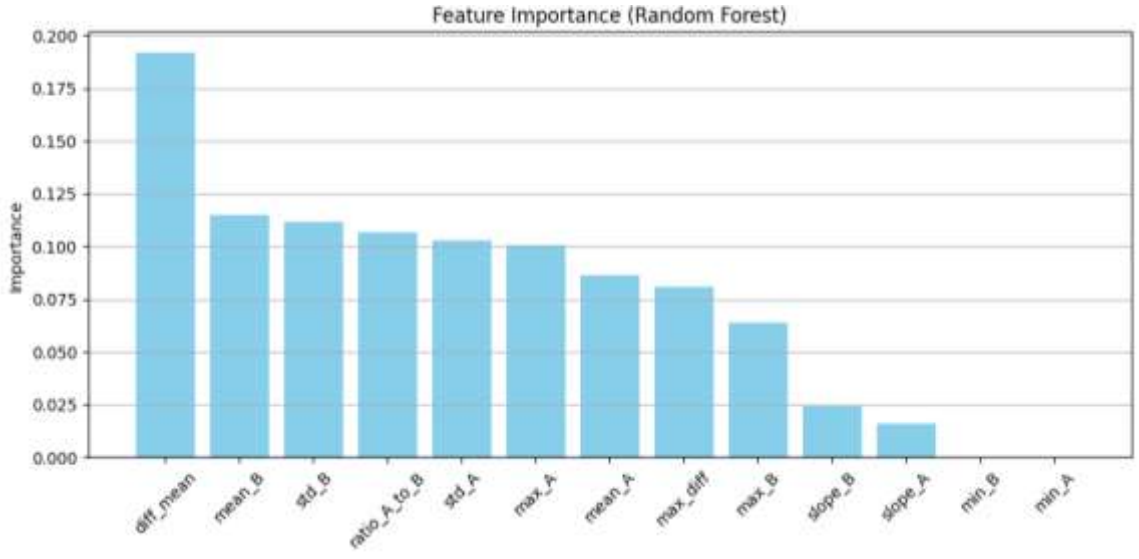
Đặc trưng std_A

Còn với biểu đồ tại Hình 3.19 (c) của std_A, thấy rằng sự khác biệt ngược lại so với 2 biểu đồ trên. Đối với nhóm bình thường, điểm trung vị gần bằng 0, phân bố khá rộng, xuất hiện nhiều điểm ngoại lai, có trường hợp độ lệch chuẩn cao bất thường (lớn hơn 5). Điều này cho thấy trong điều kiện bình thường, phần lớn thời gian lượng mưa tại trạm A ổn định, nhưng cũng có lúc xuất hiện biến động mạnh (mưa rào, mưa dồn dập ngắn hạn). Ngược lại, đối với nhóm lỗi, phân bố hẹp hơn, trung vị gần bằng 0, độ lệch chuẩn thấp hơn nhóm bình thường, không có nhiều điểm ngoại lai. Điều này phản ánh rằng khi trạm bị lỗi tắc phễu thì lượng mưa đo được thường ổn định, ít dao động, dẫn đến std_A nhỏ.

3.3.3. Lựa chọn đặc trưng và cân bằng dữ liệu

a. Lựa chọn đặc trưng

Sau khi trích xuất các đặc trưng từ từng cửa sổ thời gian, tổng cộng 13 đặc trưng thống kê được hình thành. Tuy nhiên, không phải tất cả đặc trưng đều có đóng góp đáng kể cho bài toán phân loại. Do đó, thuật toán Random Forest được sử dụng để ước lượng mức độ quan trọng (feature importance) của từng đặc trưng (kỹ thuật này sẽ được giải thích ở phần nguyên lý hoạt động của thuật toán ở mục 3.3.4). Các đặc trưng có chỉ số quan trọng dưới ngưỡng 0.05 được loại bỏ nhằm giảm nhiễu và cải thiện hiệu quả huấn luyện. Kết quả cho thấy một số đặc trưng như diff_mean, ratio_A_to_B và slope_A có độ quan trọng vượt trội, phản ánh rõ sự chênh lệch và xu hướng lượng mưa giữa các khu vực – đây là dấu hiệu trực tiếp của hiện tượng tắc phễu.



Hình 3.21 Đồ thị mô tả mức độ quan trọng của các đặc trưng

b. Cân bằng dữ liệu

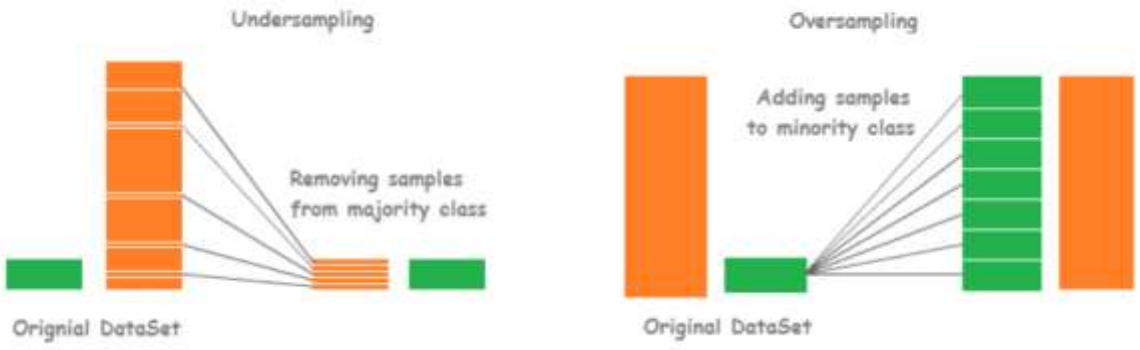
Tập dữ liệu đặc trưng sau khi chọn lọc được chuẩn hóa để đưa về cùng thang đo, giúp cải thiện hiệu năng của các mô hình học máy. Tuy nhiên, một vấn đề được đặt ra ở đây đó là tập dữ liệu sau khi trích xuất đặc trưng bị mất cân bằng: các trường hợp tắc chỉ chiếm tỷ lệ rất nhỏ so với dữ liệu bình thường. Nếu không xử lý phù hợp, mô hình dễ bị thiên lệch, chỉ học được đặc trưng của lớp phổ biến và bỏ qua các sự kiện hiếm nhưng quan trọng. Do đó, việc xử lý mất cân bằng dữ liệu là bước bắt buộc để mô hình có thể phát hiện chính xác các lỗi tắc phễu ngay cả khi chúng xảy ra không thường xuyên.

Dựa trên bài báo tổng quan [20] các nhóm giải pháp xử lý mất cân bằng dữ liệu có thể được chia làm ba cấp độ chính: cấp dữ liệu, cấp thuật toán và học tăng cường. Dưới đây là một số giải pháp phổ biến.

(1) Cấp dữ liệu (Data-level)

Các phương pháp ở cấp này tác động trực tiếp đến dữ liệu huấn luyện, nhằm điều chỉnh lại tỷ lệ giữa các lớp thông qua kỹ thuật resampling. Hai dạng chính gồm:

- Over-sampling: Tăng số lượng mẫu của lớp thiểu số bằng cách sao chép các mẫu hiện có hoặc tạo thêm dữ liệu nhân tạo, nhằm cân bằng tỷ lệ giữa các lớp.
- Under-sampling: Giảm số lượng mẫu của lớp đa số, thường bằng cách loại bỏ ngẫu nhiên một phần dữ liệu, giúp tỷ lệ phân bố lớp trở nên cân bằng hơn.



Hình 3.22 Minh họa hai kỹ thuật xử lý mất cân bằng dữ liệu: Undersampling – Oversampling

(2) Cấp thuật toán (Algorithm-level)

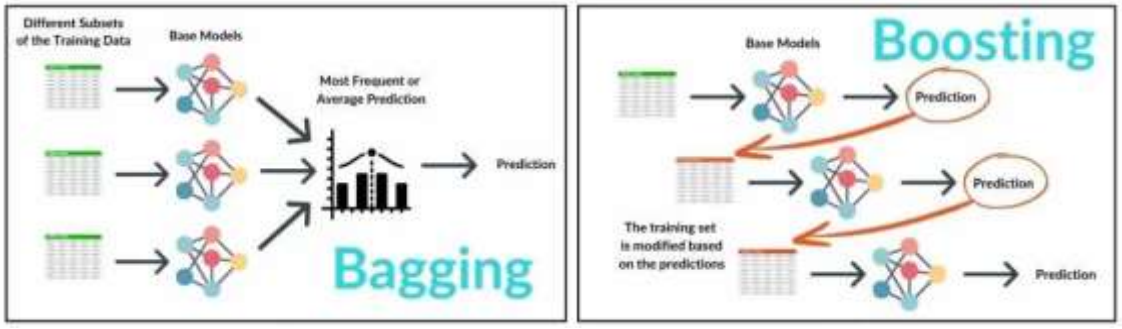
Thay vì thay đổi dữ liệu, nhóm giải pháp này điều chỉnh thuật toán học để làm cho mô hình trở nên nhạy hơn với dữ liệu mất cân bằng. Một số phương pháp bao gồm:

- Sử dụng các thuật toán phân loại được thiết kế hoặc điều chỉnh để xử lý dữ liệu lệch lớp.

- Cost-sensitive learning: Áp dụng các trọng số (weight) khác nhau cho các lớp trong hàm mất mát, nhằm tăng mức độ phạt đối với các lỗi xảy ra ở lớp thiểu số.

(3) Học tăng cường (Ensemble-level)

Các phương pháp tổ hợp nhiều mô hình học yếu, chẳng hạn như Bagging (Random Forest) và Boosting (như AdaBoost, XGBoost), đã chứng minh được hiệu quả cao trong việc giảm phương sai và tăng khả năng khái quát hóa của mô hình. Đặc biệt trong bài toán mất cân bằng dữ liệu, Boosting tỏ ra hữu ích nhờ cơ chế gán trọng số cao hơn cho các mẫu bị phân loại sai. Do các mẫu thuộc lớp thiểu số thường khó học và dễ bị phân loại nhầm, nên chúng sẽ được chú trọng nhiều hơn trong các vòng huấn luyện tiếp theo, giúp cải thiện độ chính xác cho nhóm dữ liệu này.



Hình 3.23 Hai phương pháp tổ hợp mô hình chủ yếu trong học máy

Trong khuôn khổ của đề tài này, nhóm nghiên cứu sử dụng kỹ thuật **SMOTE** (Synthetic Minority Over-sampling Technique) – một biến thể của over-sampling. SMOTE hoạt động bằng cách tạo ra các mẫu mới cho lớp thiểu số thông qua nội suy giữa các điểm gần nhau trong không gian đặc trưng. Điều này không chỉ giúp cân bằng dữ liệu mà còn hạn chế hiện tượng overfitting so với over-sampling ngẫu nhiên. Nguyên lý hoạt động của SMOTE như sau:

Bước 1: Chọn một mẫu lớp thiểu số từ tập dữ liệu gốc.

Bước 2: Tìm k láng giềng lớp thiểu số gần nhất của nó trong không gian.

Thuật toán tìm k điểm gần nhất với điểm mẫu trong cùng một lớp thiểu số dựa trên khoảng cách (thường dùng Euclidean distance)

$$d(A,B) = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2} \tag{3.14}$$

Bước 3: Chọn ngẫu nhiên một trong k hàng xóm gần nhất.

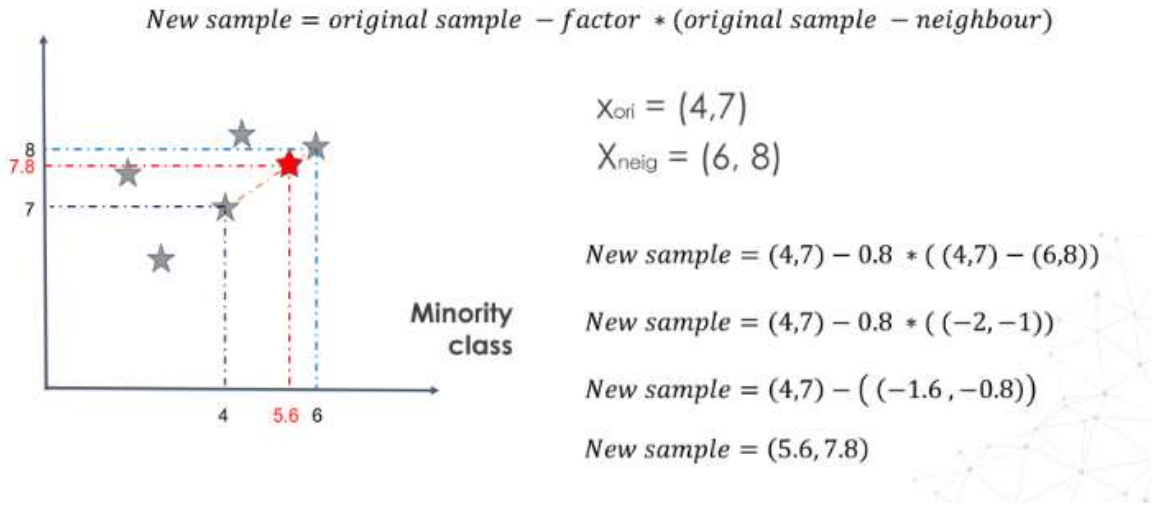
Bước 4: Tạo mẫu bằng cách nội suy giữa mẫu lớp thiểu số dã chọn và hàng xóm được chọn ngẫu nhiên.

$$p_i = X + rand(0,1) * (y_i - X) \tag{3.15}$$

Trong đó:

- X là điểm dữ liệu lớp thiểu số gốc

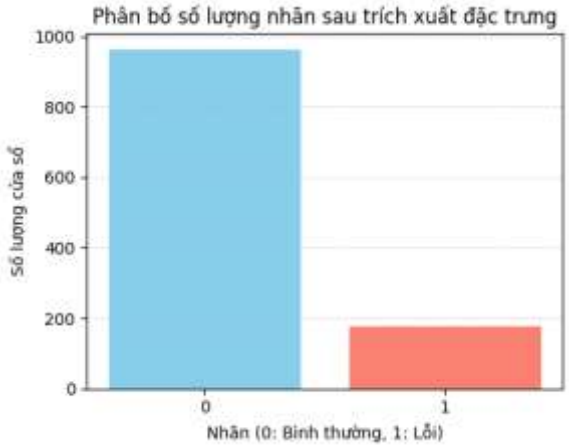
- y_i là một trong các làng giếng gần nhất
- $rand(0,1)$ là một số ngẫu nhiên trong khoảng (0,1)
- p_i là điểm dữ liệu nhân tạo mới.



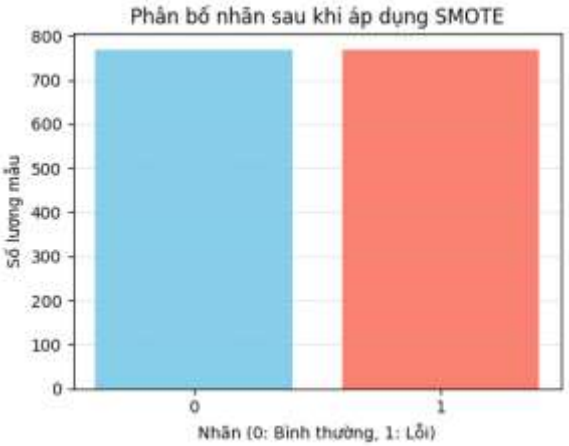
Hình 3.24 Ví dụ mô tả cách tạo ra một điểm mới bằng kỹ thuật SMOTE

Bước 5: Lặp lại các bước từ 1 – 4 cho đến khi tạo ra được số lượng mẫu tổng hợp mong muốn.

Sau khi áp dụng kỹ thuật SMOTE cho tập dữ liệu đặc trưng để tạo thêm nhiều mẫu nhân tạo cho lớp thiểu số (fault = 1), kết quả đạt được là một tập dữ liệu cân bằng hơn, giúp mô hình học được đặc điểm tốt hơn của cả hai lớp.



Hình 3.28 Biểu đồ số lượng nhãn của hai lớp trước khi dùng SMOTE



Hình 3.25 Biểu đồ số lượng nhãn của hai lớp sau khi dùng SMOTE

3.3.4. Huấn luyện mô hình

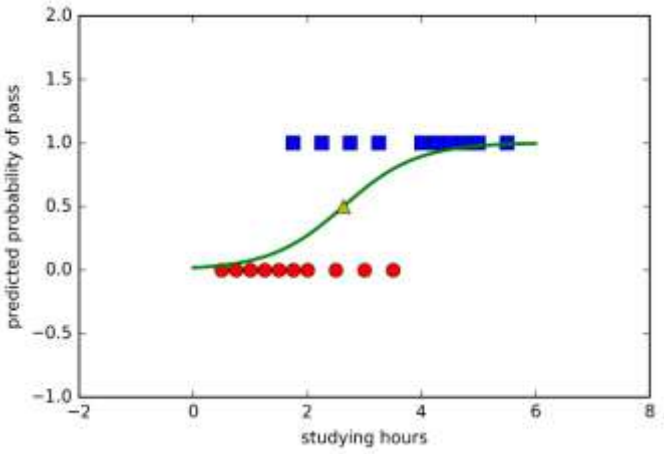
Sau bước tiền xử lý và cân bằng dữ liệu bằng kỹ thuật SMOTE, tập dữ liệu được chia thành hai phần: 80% dùng để huấn luyện, 20% dùng để đánh giá. Nhóm tiến hành huấn luyện và so sánh nhiều mô hình học máy phổ biến, bao gồm Logistic Regression,

Decision Tree, Random Forest, AdaBoost, Gradient Boosting và XGBoost. Việc lựa chọn các mô hình này không chỉ dựa trên tính đa dạng thuật toán (linear, tree-based, ensemble), mà còn dựa trên khảo sát ở các bài báo đã phân tích ở chương 2: theo đó, Random Forest và XGBoost được đánh giá là hai mô hình có hiệu suất vượt trội trong các nghiên cứu.

Để hiểu rõ cơ chế hoạt động cũng như lý do vì sao từng mô hình có thể phù hợp hoặc không phù hợp với bài toán này, nhóm trình bày nguyên lý hoạt động chính của các mô hình được sử dụng.

a. Logistic Regression

Trong lĩnh vực học máy (machine learning), việc phân loại nhị phân là một trong những bài toán phổ biến và thiết yếu. Thuật toán Logistic Regression, mặc dù có tên là "hồi quy", lại được sử dụng rộng rãi trong các bài toán phân loại nhị phân. Với cơ sở toán học vững chắc và khả năng giải thích trực quan, Logistic Regression là một trong những thuật toán nền tảng, được áp dụng từ khoa học dữ liệu đến y học, tài chính và phân tích hành vi người dùng.



Hình 3.30 Đồ thị minh họa mô hình hồi quy logistic

Nguyên lý hoạt động của Logistic Regression như sau:

Bước 1: Tính tổ hợp tuyến tính đầu vào

Mỗi mẫu dữ liệu đầu vào $x \in R^n$ được biến đổi thành một giá trị thực z bằng công thức tuyến tính:

$$z = w^T x + b \tag{3.16}$$

Trong đó:

- w: là vector trọng số (weight)
- b: là hệ số chệch (bias)
- x: là vector đặc trưng của mẫu dữ liệu

Bước 2: Áp dụng hàm kích hoạt sigmoid để đưa kết quả về xác suất:

Để ánh xạ giá trị z về khoảng xác suất [0,1], ta dùng hàm sigmoid:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.17}$$

Đầu ra $\hat{y} = \sigma(z)$ là xác suất dự đoán mẫu dữ liệu thuộc về lớp nào.

Bước 3: Tính hàm mất mát (Loss Function) – Cross Entropy

Để đánh giá độ chính xác của mô hình, ta sử dụng hàm mất mát Cross-Entropy:

$$L(w) = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \tag{3.18}$$

- y_i: Giá trị thực tế của đầu ra thứ i.
- $\hat{y}_i$: Xác suất dự đoán của mô hình cho đầu vào thứ i.
- n: Số lượng mẫu dữ liệu trong tập huấn luyện.

Bước 4: Tối ưu hóa trọng số bằng Gradient Descent

Cập nhật các trọng số w_j để tối thiểu hóa hàm mất mát bằng cách tính gradient:

$$\frac{\partial L(w)}{\partial w_j} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)x_{ij} \tag{3.19}$$

Và cập nhật:

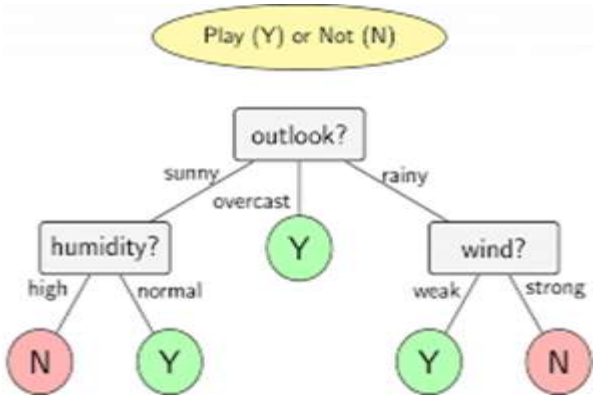
$$w_j = w_j - \alpha.\frac{\partial L(w)}{\partial w_j} \tag{3.20}$$

Trong đó:

- w_j: Trọng số hiện tại của đặc trưng thứ j.
- α: Tốc độ học (learning rate).
- x_{ij}: Giá trị đặc trưng j tại mẫu i.

b. Decision Tree

Decision Tree (Cây quyết định) là một thuật toán học có giám sát (supervised learning) được sử dụng cho cả bài toán phân loại (classification) và hồi quy (regression). Thuật toán xây dựng một mô hình hình cây bao gồm một nút gốc, các nút bên trong được gọi là nút quyết định kiểm tra đầu vào của nó dựa trên một biểu thức đã học và các nút lá tương ứng với một lớp hoặc quyết định cuối cùng.



Hình 3.31 Đồ thị minh họa một cây quyết định (decision tree)

– Nút gốc (Root Node): Nút đầu tiên, hứa toàn bộ dữ liệu.
– Nút con (Internal Node): Nút phân chia dựa trên thuộc tính.
– Nút lá (Leaf Node): Nút cuối cùng, chứa kết quả dự đoán.

Sức mạnh của thuật toán Decision Tree nằm ở khả năng phân chia dữ liệu một cách có hệ thống thông qua việc đặt ra các câu hỏi liên tiếp. Tuy nhiên, để xác định câu hỏi nào (thuộc tính nào) cần ưu tiên đặt trước, chúng ta cần những độ đo toán học khách quan. Các độ đo này giúp định lượng "chất lượng" của mỗi cách phân chia nhằm đảm bảo dữ liệu ở các nút con thuần nhất hơn nút gốc và ưu tiên thuộc tính mang lại nhiều thông tin nhất. Một số độ đo được ra đời để định lượng hóa các tiêu chí trên như Entropy, Information Gain (IG).

- Entropy: thước đo độ hỗn loạn của dữ liệu, giúp đánh giá mức độ ưu tiên phân chia tại mỗi nút.

$$I(D) = \sum_{i=1}^{c} p_i log_2(p_i) \tag{3.21}$$

Trong đó:

- p_i: Tỷ lệ lớp i trong tập dữ liệu D
- c: Số lớp
- Entropy càng cao (≈ 1): Dữ liệu hỗn loạn
- Entropy càng thấp (≈ 0): Dữ liệu thuần nhất

- Information Gain (IG) – Độ lợi thông tin, giúp đánh giá mức độ cải thiện nếu phân chia theo một thuộc tính để biết thuộc tính nào giảm entropy tốt nhất.

$$IG(D, A) = Entropy\ ban\ đầu - \sum(\frac{kích\ thước\ nhánh}{Kích\ thước\ tập\ gốc} * Entropy\ nhánh)$$

$$= Entropy(D) - \sum_{v \in Values(A)} \frac{D_v}{D} * Entropy(D_v) \tag{3.22}$$

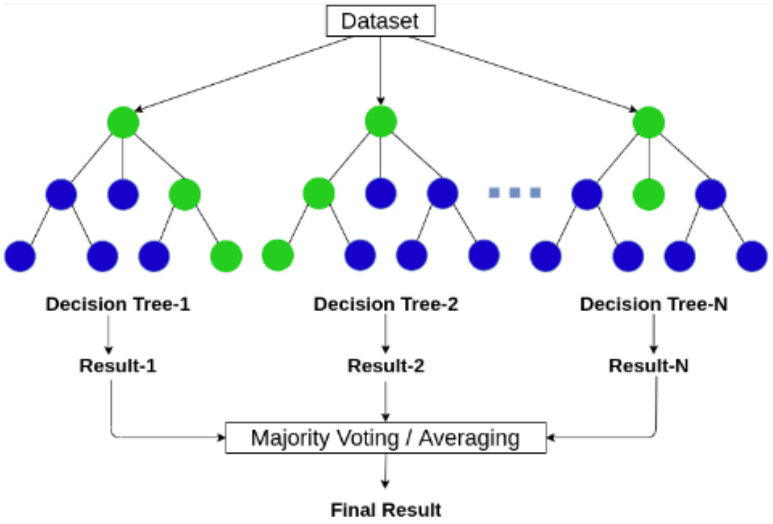
Trong đó:

- A: Thuộc tính đang xem xét
- D_v: Tập con của D khi thuộc tính A có giá trị v
- IG càng cao: thuộc tính A càng hữu ích cho việc phân chia

- IG càng thấp: thuộc tính A không giúp cải thiện phân loại

c. Random Forest

Random Forest là một thuật toán học máy thuộc nhóm học có giám sát (supervised learning) và được sử dụng phổ biến trong các bài toán phân loại (classification) và hồi quy (regression). Thuật toán này là một dạng của tập hợp học (ensemble learning), nơi mà nhiều mô hình yếu (weak learners), cụ thể là các cây quyết định (decision trees), được kết hợp lại để tạo thành một mô hình mạnh mẽ hơn, xử lý hiệu quả cả dữ liệu nhiễu, mất cân bằng và dữ liệu lớn, đồng thời đạt được độ chính xác cao mà vẫn giảm thiểu hiện tượng quá khớp (overfitting).



Hình 3.32 Đồ thị minh họa mô hình Random Forest

Thuật ngữ "Random" trong mô hình xuất phát từ hai yếu tố chính:

Ngẫu nhiên trong chọn mẫu: thay vì sử dụng toàn bộ dữ liệu huấn luyện để xây dựng từng cây quyết định, thuật toán Random Forest chọn một mẫu ngẫu nhiên từ tập dữ liệu (với hoàn lại) để xây dựng mỗi cây. Kỹ thuật này được gọi là Bagging (Bootstrap Aggregating). Bagging giúp giảm thiểu phương sai của mô hình, cải thiện độ chính xác tổng thể.

- Từ tập dữ liệu gốc D có N mẫu, tạo m tập con $D_1, D_2, \dots D_m$.
- Mỗi tập con được tạo bằng cách lấy ngẫu nhiên N mẫu có hoàn lại.

Ngẫu nhiên trong chọn đặc trưng: khi tạo các nút trong mỗi cây, chỉ một tập con ngẫu nhiên của tất cả các đặc trưng được xem xét để chọn đặc trưng tốt nhất tại mỗi bước. Điều này giúp các cây quyết định đa dạng hơn, giảm thiểu hiện tượng overfitting và đảm bảo rằng các cây không bị phụ thuộc quá mức vào một đặc trưng cụ thể nào đó.

- Với bài toán phân loại: thường chọn $\sqrt{p}$ đặc trưng (p là tổng số đặc trưng).
- Với bài toán hồi quy: thường chọn p/3 đặc trưng.

Do quá trình xây dựng mỗi cây quyết định đều có yếu tố ngẫu nhiên (random) nên kết quả là các cây quyết định trong thuật toán Random Forest có thể khác nhau. Thuật toán Random Forest sẽ bao gồm nhiều cây quyết định, mỗi cây được xây dựng dùng thuật toán Decision Tree trên tập dữ liệu khác nhau và dùng tập thuộc tính khác nhau. Sau đó kết quả dự đoán của thuật toán Random Forest sẽ được tổng hợp từ các cây quyết định.

- Với bài toán phân loại: $\hat{y} = majotiry\_vote(\hat{y}^{(1)}, \hat{y}^{(2)}, \dots, \hat{y}^{(B)})$
- Với bài toán hồi quy: $\hat{y} = \frac{1}{B}.\sum_{B=1}^{B} \hat{y}^{(b)}$

Trong đó: B là tổng số cây và $\hat{y}^{(b)}$ là dự đoán của cây quyết định thứ b cho mẫu x.

Một đặc điểm nổi bật của Random Forest là khả năng ước tính lỗi dự đoán mà không cần tách riêng một tập dữ liệu kiểm tra, thông qua khái niệm lỗi *Out-of-Bag (OOB)*. Trong quá trình huấn luyện, khoảng một phần ba mẫu trong mỗi bootstrap không được sử dụng để huấn luyện cây và do đó, có thể được sử dụng như một tập kiểm tra tự nhiên. OOB Error chính là sai số được ước lượng trên chính tập dữ liệu này mà không cần tập kiểm tra riêng. Sai số OOB càng thấp chứng tỏ mô hình học tốt.

$$OOB\ Error = \frac{1}{N}.\sum_{i=1}^{N} I(\hat{y_i}^{(OOB)} \neq y_i) \tag{3.23}$$

Trong đó:

- N: số mẫu trong tập huấn luyện.
- $\hat{y_i}^{(OOB)}$: nhãn được dự đoán bằng cách lấy đa số phiếu từ các cây mà không dùng mẫu x_i trong huấn luyện.
- y_i: nhãn thật của mẫu i.
- I(.) là hàm chỉ thị: bằng 1 nếu điều kiện đúng, 0 nếu sai.

Đặc điểm nổi bật thứ hai của Random Forest là khả năng đánh giá tầm quan trọng của từng đặc trưng trong quá trình xây cây. Ý tưởng cơ bản là: đặc trưng nào giúp giảm độ bất thuần (impurity) nhiều hơn trong quá trình phân tách thì quan trọng hơn. Tầm quan trọng này được tính trung bình từ mỗi lần chia nút trên tất cả các cây trong rừng.

$$\text{Feauture Importance}(X_j) = \frac{1}{B}\sum_{b=1}^{B}\sum_{T\in nodes}\Delta I(t)$$

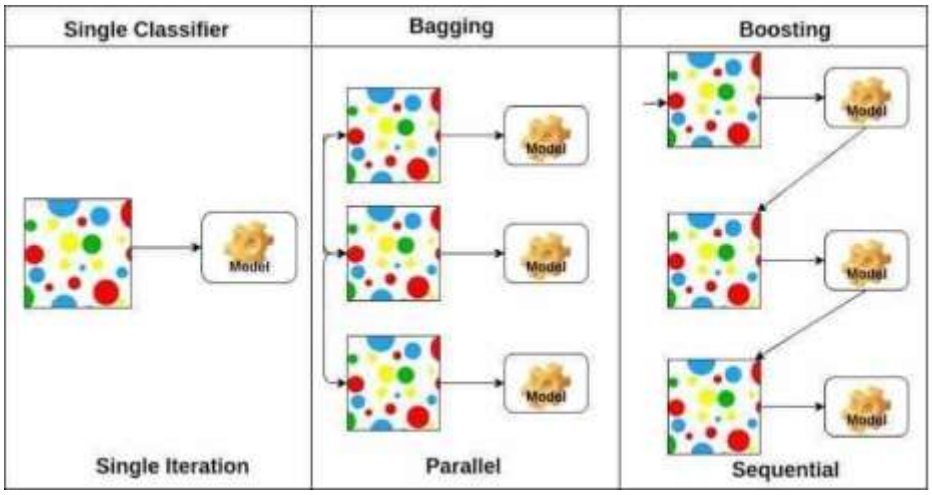
- $\Delta I(t)$: là mức giảm impurity tại nút t khi sử dụng đặc trưng X_j.
- B là tổng số cây trong rừng.
- Nodes là tập hợp các nút trong cây b nơi X_j được sử dụng.

d. AdaBoost – Gradient Boosting

Qua model Random Forest, chúng ta biết được Bagging được kết hợp từ các model được fit trên các tập dữ liệu con (được lấy theo bootstrap sample để các tập con được kỳ

vọng là độc lập), từ đó kết hợp các kết quả từ các model này để đưa ra kết quả cuối cùng. Tuy nhiên, có 1 số điều chúng ta có thể để ý ở đây:

- Các model trong Bagging đều là học một cách riêng rẽ, không liên quan hay ảnh hưởng gì đến nhau, điều này trong một số trường hợp có thể dẫn đến kết quả tệ khi các model có thể học cùng ra 1 kết quả. Chúng ta không thể kiểm soát được hướng phát triển của các model con thêm vào bagging
- Chúng ta mong đợi các model yếu của thể hỗ trợ lẫn nhau, học được từ nhau để tránh đi vào các sai lầm của model trước đó. Đây là điều Bagging không làm được.



Hình 3.33 Hình ảnh minh họa sự khác nhau giữa mô hình đơn, bagging, boosting

Boosting ra đời dựa trên việc mong muốn cải thiện những hạn chế trên. Ý tưởng cơ bản là Boosting sẽ tạo ra một loạt các model yếu, học bổ sung lẫn nhau. Nói cách khác, trong Boosting, các model sau sẽ cố gắng học để hạn chế lỗi lầm của các model trước.

Vậy làm thế nào để hạn chế được sai lầm từ các model trước. Boosting tiến hành đánh trọng số cho các mô hình mới được thêm vào dựa trên các cách tối ưu khác nhau. Tùy theo cách đánh trọng số (cách để các model được fit một cách tuần tự) và cách tổng hợp lại các model, từ đó hình thành nên 2 loại Boosting :

- Adaptive Boosting (AdaBoost)
- Gradient Boosting

Nguyên lý hoạt động thuật toán AdaBoost.

Giả sử tập dữ liệu huấn luyện $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$, với $y_i \in \{-1, +1\}$.

Bước 1: Khởi tạo trọng số: Gán trọng số ban đầu cho từng mẫu:

$$w_i^{(1)} = \frac{1}{N}, \forall i = 1,2, \dots, N \tag{3.24}$$

Bước 2: Lặp qua T weak learners:

- Huấn luyện weak learner $h_t(x)$ trên tập dữ liệu với trọng số $w_i^{(t)}$
- Tính tỷ lệ lỗi của mô hình h_t:

$$\epsilon_t = \sum_{i=1}^{N} w_i^t . I(h_t(x_i) \neq y_i) \tag{3.25}$$

- Tính trọng số cho weak learner h_t (alpha), biểu thị mức độ "tin cậy" của mô hình

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) \tag{3.26}$$

  Mô hình càng chính xác (lỗi nhỏ) thì α_t càng lớn.

- Cập nhật trọng số mẫu:

$$w_i^{t+1} = w_i^t . e^{-\alpha_t y_i h_t(x_i)} \tag{3.27}$$

- Chuẩn hóa trọng số:

$$w_i^{t+1} = \frac{w_i^{t+1}}{\sum_{j=1}^{N} w_j^{t+1}} \tag{3.28}$$

Bước 3: Kết hợp các weak learners:

Mô hình cuối cùng có tổng trọng số:

$$H(x) = sign\,(\sum_{t=1}^{T} \alpha_t h_t(x)) \tag{3.29}$$

Nguyên lý hoạt động thuật toán Gradient Boosting.

Gradient Boosting là một phương pháp tăng cường (boosting) được để xuất ra nhằm khắc phục hạn chế của AdaBoost trong việc tối ưu hóa hàm mất mát. Thay vì chỉ dựa vào hàm mũ như AdaBoost, Gradient Boosting tổng quát hóa quá trình bằng cách áp dụng Gradient Descent để tối ưu hóa trực tiếp bất kỳ hàm mất mát nào trong không gian hàm. Cơ chế này giúp mô hình học từ sai lệch (residual) của mô hình trước đó một cách linh hoạt và hiệu quả hơn. Nhờ khả năng tùy biến cao, Gradient Boosting là nền tảng cho các mô hình hiện đại như XGBoost, LightGBM và CatBoost.

Bài toán tổng quát: cho tập dữ liệu $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$, Gradient Boosting xây dựng mô hình F(x) dưới dạng tổng có trọng số của các weak learners $h_t(x)$:

$$F(x) = \sum_{t=0}^{T} \gamma_t h_t(x) \tag{3.30}$$

Trong đó:

- $F_0(x)$ là mô hình khởi tạo, thường là hằng số
- γ_t là hệ số học
- $h_t(x)$ là weak learner tại bước t

Bước 1: Khởi tạo mô hình ban đầu:

$$F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i, \gamma) \tag{3.31}$$

Thường lấy $F_0(x) = \bar{y}$

Bước 2: Lặp qua t =1 đến T:

- Tính phần dư giả gradient (pseudo – residuals):

$$r_{i,t} = -\left[\frac{\partial L\big(y_i, F_{t-1}(x_i)\big)}{\partial F_{t-1}(x_i)}\right], \forall i = 1, \dots N \tag{3.32}$$

- Huấn luyện weak learner $h_t(x)$ dể khớp $r_{i,t}$.
- Tìm hệ số γ_t tối ưu:

$$\gamma_t = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{N} L\big(y_i, F_{t-1}(x_i) + \gamma h_t(x_i)\big) \tag{3.33}$$

- Cập nhật mô hình:

$$F_t(x) = F_{t-1}(x) + v.\gamma_t h_t(x) \tag{3.34}$$

(trong đó v là learning rate, thường là 0.1).

e. XGBoost

XGBoost (Extreme Gradient Boosting) là một thuật toán học máy mạnh mẽ, kế thừa và phát triển từ Gradient Boosting. Nguyên lý hoạt động của XGBoost dựa trên tư tưởng tối ưu hàm mất mát bằng gradient descent, nhưng bổ sung thêm hàng loạt cải tiến để tăng độ chính xác và khả năng tổng quát hóa mô hình.

Cốt lõi của thuật toán là hàm tối ưu hàm mất mát kết hợp thêm thành phần regularization, giúp kiểm soát độ phức tạp của mô hình. Để tối ưu mô hình, XGBoost sử dụng khai triển chuỗi Taylor bậc hai của hàm mất mát quanh giá trị dự đoán hiện tại.

$$L^{(t)} = \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \Omega(f_t) \tag{3.35}$$

Trong đó:

- g_i: gradient (đạo hàm bậc nhất)
- h_i: hessian (đạo hàm bậc hai)
- $\Omega(f_t)$: thành phần regularization cho cây f_t

Quy trình huấn luyện như sau:

Bước 1: Khởi tạo mô hình ban đầu với dự đoán hằng số.

Bước 2: Lặp qua T vòng boosting:

- Tính gradient và hessian cho mỗi mẫu.
- Xây cây quyết định $f_t(x)$, chọn chia nhánh dựa trên chỉ số cải thiện (Gain)

$$Gain = \frac{1}{2}\left[\frac{(\sum_{i\in I_L} g_i)\text{^}2}{\sum_{i\in I_L} h_i + \lambda} + \frac{(\sum_{i\in I_R} g_i)\text{^}2}{\sum_{i\in I_R} h_i + \lambda} - \frac{(\sum_{i\in I} g_i)\text{^}2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma \quad (3.36)$$

- Gán giá trị cho mỗi lá theo công thức.

$$w_j = -\frac{\sum g_i}{\sum h_i + \lambda} \quad (3.37)$$

- Cập nhật mô hình tổng:

$$F_t(x) = F_{t-1}(x) + \eta f_t(x) \quad (3.38)$$

3.3.5. Đánh giá kết quả

Để đánh giá chất lượng mô hình, các chỉ số sau được sử dụng:

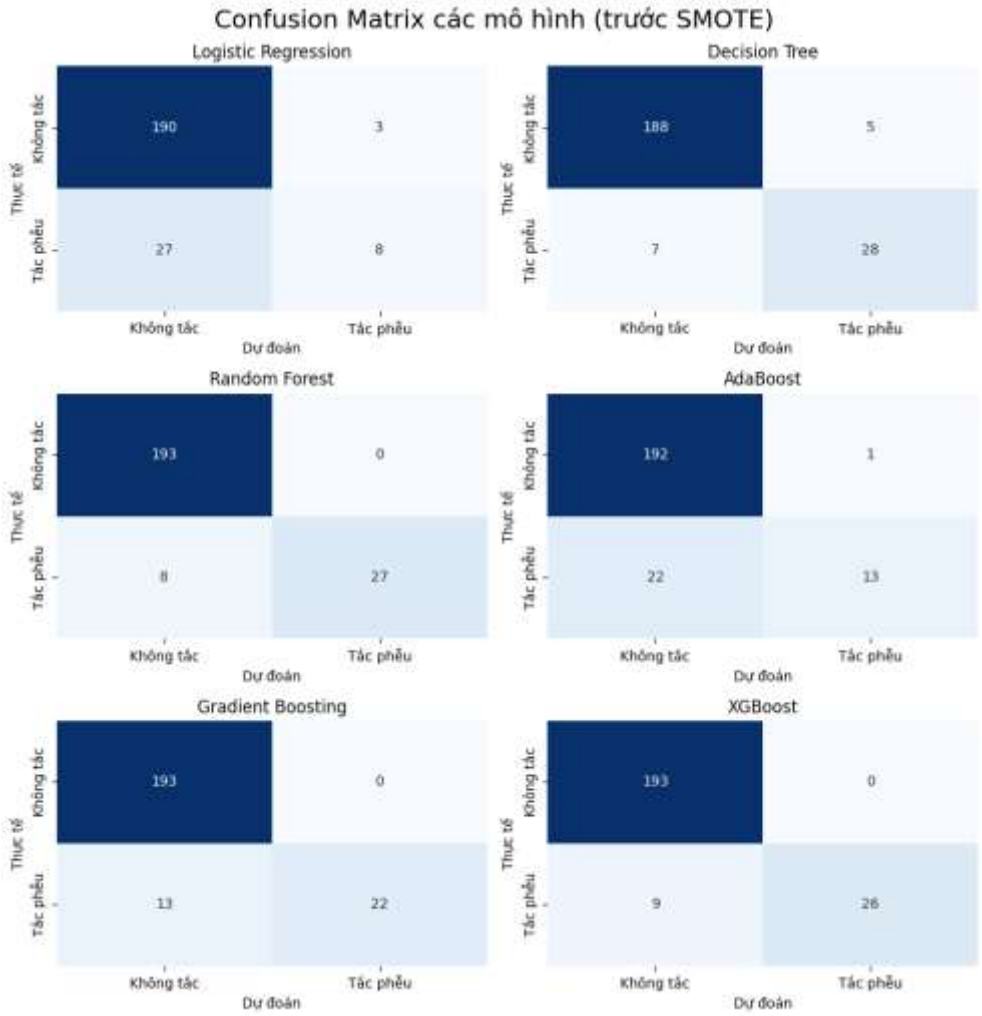
- Accuracy: tỷ lệ dự đoán đúng tổng thể;
- Precision, Recall, F1-score cho từng lớp (0: bình thường, 1: lỗi);
- Confusion matrix: ma trận sai số giúp phân tích nhầm lẫn.

Đặc biệt, do mục tiêu của hệ thống là phát hiện lỗi tắc phễu (lớp 1), nên F1-score của lớp này là tiêu chí đánh giá chính.

- Kết quả của các mô hình trước khi áp dụng SMOTE như sau:

Bảng 3.3 Bảng so sánh kết quả của các mô hình khi chưa xử lý mất cân bằng dữ liệu

Mô hình	**Accuracy**	**Precision (1)**	**Recall (1)**	**F1-score (1)**
Logistic Regression	0.8694	0.7272	0.2285	0.3478
Decision Tree	0.9473	0.8484	0.8000	0.8235
Random Forest	0.9649	1.0000	0.7714	0.8709
AdaBoost	0.8991	0.9285	0.3715	0.5306
Gradient Boosting	0.9429	1.0000	0.6285	0.7719
XGBoost	0.9605	1.0000	0.7428	0.8524



Hình 3.34 Các ma trận nhầm lẫn của các mô hình khi chưa xử lý mất cân bằng dữ liệu

Kết quả cho thấy hầu hết mô hình đều đạt độ chính xác tổng thể (accuracy) cao, dao động từ 86% đến 96%. Tuy nhiên, do mất cân bằng dữ liệu, độ chính xác này không phản ánh đúng hiệu quả nhận diện lỗi. Thay vào đó, các chỉ số Precision, Recall và F1-score đối với lớp lỗi được sử dụng làm tiêu chí chính để so sánh.

- Logistic Regression: Mặc dù đạt độ chính xác tổng thể 86.84%, mô hình chỉ phát hiện đúng 8/35 mẫu lỗi (Recall = 22.86%) và bỏ sót đến 77.14% lỗi. F1-score của lớp lỗi chỉ đạt 0.35. Đây là hạn chế điển hình của các mô hình tuyến tính khi không được điều chỉnh cho dữ liệu mất cân bằng. Logistic Regression do đó không phù hợp để phát hiện lỗi tắc phễu.
- Decision Tree: Mô hình đạt Accuracy 94.74% và thể hiện khả năng phân loại lỗi tốt hơn với Recall = 80.00%, Precision = 84.85%, và F1-score = 0.82 cho lớp lỗi. Với chỉ 7 lỗi bị bỏ sót và độ tin cậy cao, Decision Tree là một lựa chọn đơn giản nhưng hiệu quả trong bối cảnh hiện tại.
- Random Forest: Là mô hình có hiệu suất cao nhất trước SMOTE với Accuracy = 96.49%, Precision = 100%, Recall = 77.14%, và F1-score = 0.87. Mô hình không cảnh báo sai lỗi nào, đồng thời phát hiện được 27/35 lỗi. Đây là lựa chọn rất đáng tin cậy với độ chính xác tuyệt đối trong cảnh báo lỗi.

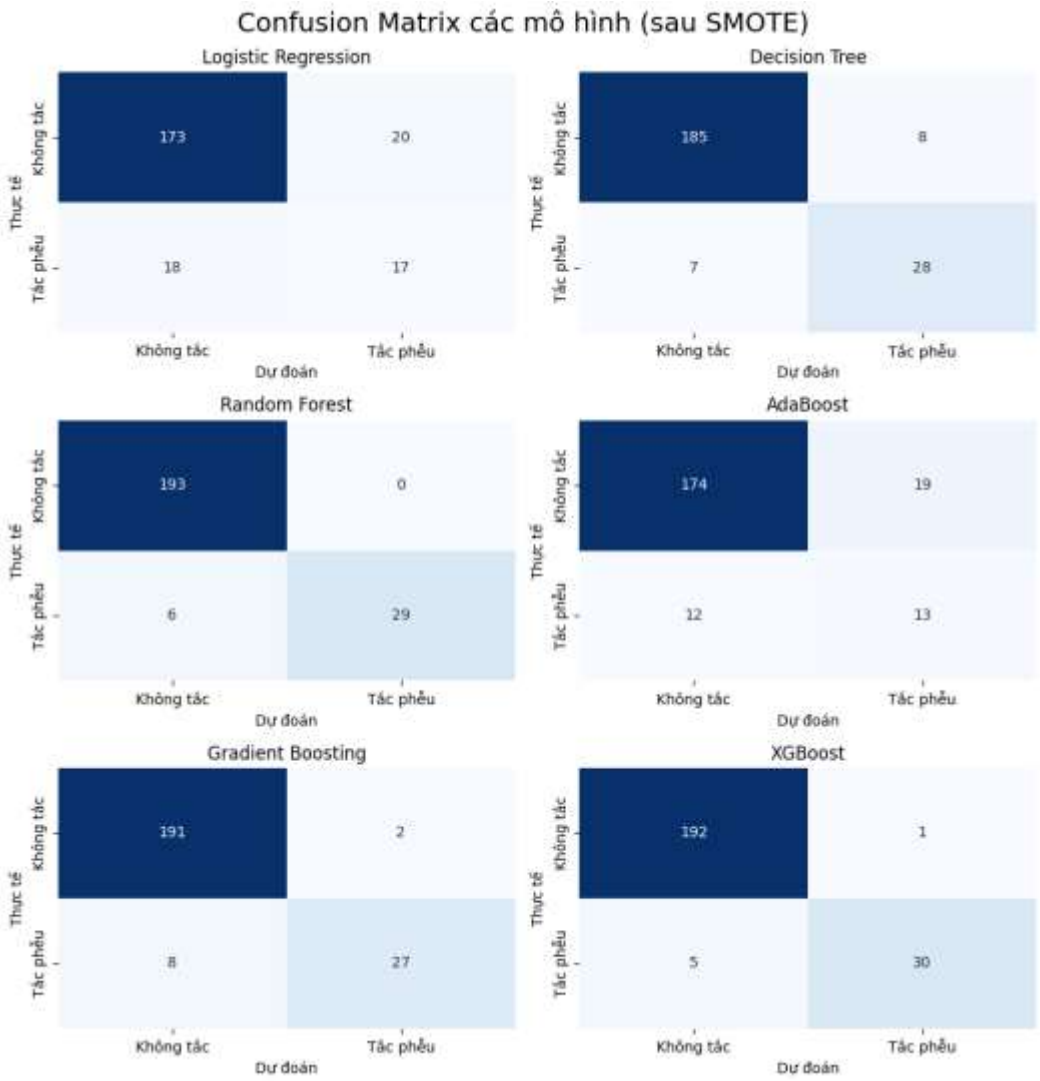
- AdaBoost: Tuy đạt Precision cao (92.86%) và Accuracy 89.91%, nhưng mô hình chỉ phát hiện được 13/35 lỗi (Recall = 37.14%), F1-score chỉ đạt 0.53. Với 22 lỗi bị bỏ sót, AdaBoost tỏ ra quá bảo thủ và không phù hợp cho bài toán yêu cầu phát hiện lỗi đầy đủ.
- Gradient Boosting: Mô hình đạt Accuracy 94.30%, Precision 100%, Recall 62.86%, và F1-score 0.77. Mặc dù không có cảnh báo sai, nhưng việc bỏ sót 13 lỗi khiến khả năng bao phủ lỗi chưa đủ mạnh. Gradient Boosting là lựa chọn an toàn, nhưng vẫn cần cải thiện độ nhạy.
- XGBoost: Đạt Accuracy 96.05%, Precision 100%, Recall 74.29% và F1-score 0.85. Với 9 lỗi bị bỏ sót và không có cảnh báo sai, XGBoost thể hiện hiệu quả ổn định và khái quát tốt. Đây là mô hình có tiềm năng ứng dụng thực tế cao trong phát hiện lỗi tắc phễu.

Trước khi áp dụng SMOTE, Random Forest và XGBoost là hai mô hình cho hiệu quả tốt nhất trong nhận diện lỗi tắc phễu. Cả hai đều đạt Precision tuyệt đối và Recall ở mức cao (>74%), phù hợp với yêu cầu phát hiện lỗi trong bối cảnh thực tế. Tuy nhiên, do đặc thù mất cân bằng dữ liệu, các mô hình vẫn còn bỏ sót một tỷ lệ lỗi đáng kể. Điều này cho thấy nhu cầu cần thiết áp dụng kỹ thuật tăng cường mẫu như SMOTE nhằm mở rộng không gian biểu diễn của lớp lỗi, từ đó cải thiện khả năng tổng quát của mô hình trên dữ liệu thực tế.

- Kết quả của các mô hình sau khi áp dụng SMOTE như sau:

Bảng 3.4 Bảng so sánh kết quả của các mô hình sau khi xử lý mất cân bằng dữ liệu

Mô hình	**Accuracy**	**Precision (1)**	**Recall (1)**	**F1-score (1)**
Logistic Regression	0.8333	0.4595	0.4857	0.4722
Decision Tree	0.9342	0.7778	0.8000	0.7887
Random Forest	0.9736	1.0000	0.8285	0.9062
AdaBoost	0.8640	0.5476	0.6571	0.5974
Gradient Boosting	0.9561	0.9310	0.7714	0.8437
XGBoost	0.9736	0.9677	0.8571	0.9091



Hình 3.35 Các ma trận nhầm lẫn của các mô hình sau khi xử lý mất cân bằng dữ liệu

Sau khi áp dụng kỹ thuật tổng hợp mẫu thiểu số SMOTE, hiệu năng của các mô hình học máy đã được cải thiện đáng kể, đặc biệt trong việc nhận diện chính xác lớp thiểu số (tắc phễu).

- Logistic Regression: Sau SMOTE, Recall cải thiện đáng kể từ 22.86% lên 48.57%, F1-score tăng từ 0.35 lên 0.47. Tuy nhiên, Precision giảm từ 0.73 xuống còn 0.46 do mô hình đưa ra nhiều cảnh báo sai hơn (20 false positives). Mặc dù độ nhạy tăng, tổng thể hiệu suất vẫn còn thấp và chưa đáng tin cậy để triển khai cảnh báo lỗi.
- Decision Tree: Hiệu suất ổn định sau SMOTE với Recall giữ nguyên ở mức 80% và F1-score duy trì quanh mức 0.79–0.82. Precision giảm nhẹ từ 0.85 xuống 0.78 do tăng số cảnh báo sai. Tuy nhiên, mô hình vẫn duy trì được độ bao phủ lỗi tốt, phù hợp với ứng dụng đơn giản yêu cầu khả năng phát hiện lỗi ổn định.
- Random Forest: Sau SMOTE, Recall tăng từ 77.14% lên 82.86%, F1-score cải thiện từ 0.87 lên 0.91 trong khi Precision vẫn duy trì ở mức tuyệt đối (1.00). Mô hình không cảnh báo sai và phát hiện được gần như toàn bộ lỗi. Đây là mô hình mạnh nhất về cả độ chính xác và độ nhạy, rất phù hợp để triển khai thực tế.

- AdaBoost: SMOTE giúp tăng Recall từ 37.14% lên 65.71% và F1-score từ 0.53 lên 0.60, nhưng Precision giảm mạnh từ 0.93 xuống 0.55. Mô hình trở nên nhạy hơn nhưng lại mất tính chính xác trong cảnh báo. Hiệu quả chưa ổn định, khó ứng dụng trong các hệ thống yêu cầu độ tin cậy cao
- Gradient Boosting: Sau SMOTE, mô hình đạt cân bằng tốt hơn: Recall tăng từ 62.86% lên 77.14%, F1-score từ 0.77 lên 0.84, Precision vẫn ở mức cao (93.1%). Số lỗi phát hiện tăng, trong khi số cảnh báo sai rất ít. Đây là mô hình mạnh và an toàn, phù hợp cho phát hiện lỗi dài hạn và đáng tin cậy.
- XGBoost: XGBoost tiếp tục giữ vững hiệu suất cao sau SMOTE: Recall tăng từ 74.29% lên 85.71%, F1-score từ 0.85 lên 0.91, Precision chỉ giảm nhẹ (từ 1.00 xuống 0.97). Mô hình phát hiện chính xác 30/35 lỗi với chỉ 1 cảnh báo sai. Đây là mô hình toàn diện nhất, lý tưởng cho triển khai thực tế trong môi trường yêu cầu độ chính xác và bao phủ cao.

Sau khi sử dụng SMOTE để cân bằng dữ liệu, các mô hình đều có sự cải thiện rõ rệt về độ nhạy (recall) và F1-score, đặc biệt với các lớp thiểu số. Mô hình Random Forest và XGBoost cho kết quả tốt nhất về độ chính xác tổng thể và khả năng phân biệt giữa các lớp, kết quả này khá tương quan như các bài báo đã nêu ở chương 2. Hai mô hình này nên được ưu tiên lựa chọn nhờ hiệu quả cao và độ ổn định trong việc dự đoán các lớp ít dữ liệu sau SMOTE.

3.4 Đánh giá mô hình trên tập dữ liệu mới.

Để kiểm tra hiệu quả của hệ thống phát hiện lỗi tắc phễu được đề xuất, nhóm nghiên cứu đã tiến hành thử nghiệm trên dữ liệu mới thu thập từ trong giai đoạn từ đầu tháng 2 đến cuối tháng 3 năm 2025. Theo thông tin thực tế từ đơn vị vận hành, sự cố tắc phễu xảy ra vào lúc 22h ngày 22/02/2025 và kéo dài đến cuối ngày 26/02/2025, thời điểm sự cố được xử lý.

Hai mô hình được kiểm tra song song: một mô hình nhẹ dùng để chạy tại thiết bị Edge (sử dụng tín hiệu mưa) và một mô hình phân loại phức hợp tại Cloud (sử dụng tập dữ liệu đa trạm kết hợp với kỹ thuật SMOTE). Hình 3.32 và Hình 3.33 dưới đây thể hiện kết quả dự đoán từ hai mô hình tương ứng.

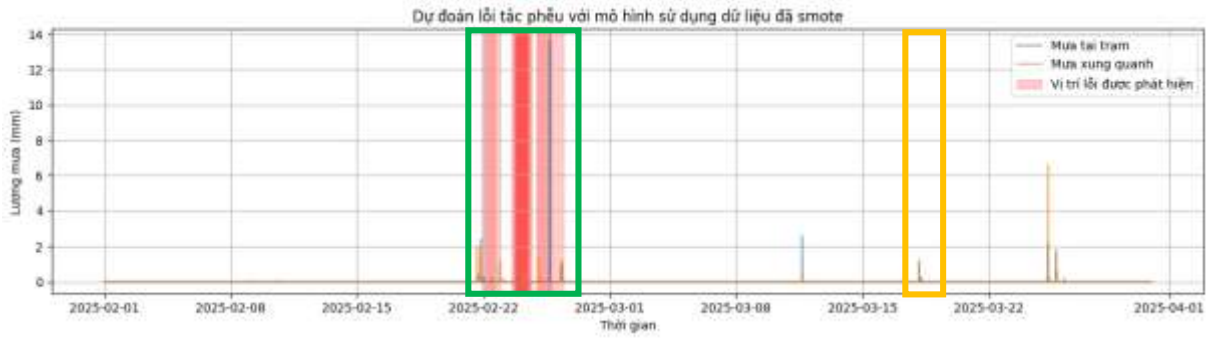
3.4.1 Kết quả mô hình tại Edge.

Hình 3.36 Lượng mưa tại trạm và đoạn mô hình dự đoán bất thường tại edge

Mô hình Edge sử dụng thuật toán phát hiện bất thường K-means Clustering trên tín hiệu lượng mưa tại trạm. Kết quả mô phỏng tại Hình 3.32 cho thấy một đoạn bất thường được phát hiện bắt đầu từ 20h50 đến 23h40 ngày 21/02 (được đánh dấu bằng khung màu xanh lá). Khoảng thời gian này diễn ra ngay trước thời điểm được xác nhận là khởi phát của sự cố tắc phễu (22h ngày 22/02), cho thấy mô hình đã có khả năng phát hiện sớm dấu hiệu bất thường liên quan đến tắc nghẽn trong hệ thống thoát nước. Tuy nhiên, từ sau 11h10 ngày 22/02 đến các ngày tiếp theo (dù lỗi vẫn còn), mô hình không phát hiện thêm bất thường. Điều này phản ánh điểm yếu điển hình của các mô hình sử dụng dữ liệu cục bộ (dữ liệu của một trạm duy nhất) tại Edge: chỉ có thể ghi nhận sự bất thường ở những thời điểm có nhiều thông tin như mưa kéo dài liên tục, không đủ dữ liệu để theo dõi quá trình phát triển kéo dài của lỗi. Ngoài ra, Hình 3.32 còn cho thấy mô hình tại Edge đã bắt nhầm một đoạn mưa ngâu vào ngày 18/3 (được đánh dấu bằng khung màu cam) vì hiện tượng mưa ngâu cho ra những đặc trưng rất giống với hiện tượng tắc phễu.

3.4.2 Kết quả mô hình tại Cloud.

Khác với mô hình tại Edge vốn chỉ đưa ra thời điểm bất thường, mô hình tại tầng Cloud được xây dựng để không chỉ đưa ra thời điểm tắc phễu mà còn dự đoán được xác suất lỗi tắc phễu tại thời điểm đó. Mô hình Cloud sử dụng thông tin lượng mưa tại trạm kết hợp với mưa từ các trạm xung quanh, cho phép ghi nhận sự khác biệt về mưa – phản ánh khả năng nước không thoát được do tắc phễu. Ngoài ra, dữ liệu được xử lý bằng kỹ thuật SMOTE giúp cải thiện hiệu suất phân loại trong điều kiện mất cân bằng nhãn.



Hình 3.37 Lượng mưa tại trạm và đoạn mô hình dự đoán lỗi tắc phễu tại cloud

Kết quả Hình 3.33 cho thấy mô hình đã phát hiện liên tục các đoạn lỗi từ ngày 22/02 đến 25/02 (được đánh dấu bằng khung màu xanh lá), bao phủ toàn bộ giai đoạn được xác nhận là có sự cố. Không chỉ phát hiện đúng khởi điểm, mô hình còn duy trì đánh dấu lỗi trong các ngày tiếp theo, theo sát thực tế vận hành. Điều này cho thấy khả năng nhận diện lỗi ổn định và chính xác nhờ sự hỗ trợ của thông tin dữ liệu đa trạm. Ngoài ra Hình 3.33 còn cho thấy mô hình đã không bắt nhầm tại ngày 18/3 (được đánh dấu bằng khung màu cam) trong trường hợp có mưa ngâu nhờ kết hợp thông tin từ các trạm khác.

Bên cạnh việc đánh dấu chính xác các đoạn có lỗi, mô hình Cloud còn đưa ra xác suất lỗi tại mỗi thời điểm – phản ánh mức độ tin cậy mà mô hình đánh giá về khả năng xảy ra lỗi tắc phễu. Việc áp dụng kỹ thuật SMOTE để xử lý mất cân bằng dữ liệu không chỉ cải thiện độ chính xác tổng thể, mà còn giúp mô hình nâng cao các giá trị xác suất lỗi tại những đoạn thực sự có lỗi. Cụ thể, một số đoạn thời gian trong giai đoạn từ 22/02 đến 25/02 sau khi áp dụng SMOTE đã cho xác suất dự đoán cao hơn rõ rệt so với trước đó.

Bảng 3.5 So sánh xác suất lỗi tại một số thời điểm trước và sau khi áp dụng SMOTE

Thời điểm	Xác suất lỗi (Trước SMOTE)	Xác suất lỗi (Sau SMOTE)
22/02/2025 – 16:00	0.54	0.92
22/02/2025 – 22:00	0.99	0.99
24/02/2025 – 08:00	0.63	0.94
24/02/2025 – 16:00	0.62	0.80
25/02/2025 – 12:00	0.62	0.99

Kết quả trên bảng 3.5 cho thấy mô hình sau SMOTE không chỉ phát hiện được lỗi, mà còn thể hiện mức độ tự tin cao hơn trong dự đoán – giúp hệ thống dễ dàng xác định các đoạn cần cảnh báo, đồng thời giảm tình trạng phân vân ở ngưỡng (borderline) và tránh bỏ sót lỗi. Khả năng này rất quan trọng trong triển khai thực tế, nơi các quyết định bảo trì cần dựa trên mức độ rủi ro cụ thể tại từng thời điểm.

Kết quả thực nghiệm cho thấy kiến trúc Hybrid kết hợp giữa Edge AI và Cloud AI (Hybrid Edge–Cloud AI) là một giải pháp hợp lý và hiệu quả. Mô hình tại Edge có khả

năng phát hiện sớm các tín hiệu bất thường một cách nhanh chóng và cục bộ, phù hợp với các yêu cầu về thời gian thực. Trong khi đó, mô hình trên Cloud, nhờ khai thác dữ liệu từ nhiều trạm lân cận và áp dụng các kỹ thuật học máy tiên tiến như SMOTE, thể hiện rõ khả năng phân tích sâu, phát hiện chính xác và theo dõi liên tục các giai đoạn phát triển của lỗi. Việc kết hợp hai cấp độ xử lý này giúp tận dụng được thế mạnh của từng mô hình: Edge xử lý nhanh tại chỗ, còn Cloud đảm bảo tính toàn diện và độ tin cậy cao. Như vậy kiến trúc Hybrid không chỉ phát huy thế mạnh riêng của từng tầng xử lý mà còn tạo ra một hệ thống giám sát thông minh, thích ứng với điều kiện hạ tầng phân tán. Đây là hướng tiếp cận phù hợp trong bối cảnh phát triển các hệ thống IIoT trong giám sát thời tiết, nơi yêu cầu đồng thời cả tính thời gian thực, khả năng mở rộng và độ tin cậy cao.

3.5. Kết luận chương 3

Chương này đã trình bày quá trình thực nghiệm và đánh giá hệ thống phát hiện tắc phễu theo kiến trúc kết hợp Edge–Cloud. Mô hình tại Edge cho thấy khả năng phát hiện sớm các bất thường, trong khi mô hình tại Cloud cung cấp phân tích chính xác hơn nhờ khai thác dữ liệu liên trạm và kỹ thuật tăng cường.

Kết quả kiểm thử trên tập dữ liệu mới cho thấy hệ thống có khả năng khái quát tốt và phù hợp với thực tế vận hành. Qua đó, kiến trúc Hybrid được chứng minh là hiệu quả, cân bằng giữa phản ứng nhanh và độ chính xác trong giám sát tắc phễu.

KẾT LUẬN

Kết quả nghiên cứu của đề tài

Đề tài đã xây dựng và đánh giá một hệ thống phát hiện sự cố tắc phễu trong các trạm đo mưa, dựa trên kiến trúc kết hợp giữa Edge AI và Cloud AI. Thông qua việc thu thập dữ liệu từ các trạm Vrain, gắn nhãn thủ công và áp dụng các mô hình học máy, hệ thống có khả năng phát hiện bất thường tại Edge một cách sớm và cảnh báo chính xác hơn tại Cloud nhờ khai thác thông tin liên trạm. Kết quả kiểm thử trên dữ liệu thực tế mới cho thấy hệ thống đáp ứng tốt cả về tốc độ phản ứng lẫn độ tin cậy, khẳng định tính khả thi và hiệu quả của kiến trúc hybrid trong ứng dụng giám sát môi trường.

Hạn chế của đề tài

Tuy nhiên, đề tài vẫn còn một số hạn chế. Quá trình gán nhãn dữ liệu được thực hiện thủ công nên cần có chuyên gia về dữ liệu đo mưa đánh giá dữ liệu. Mô hình tại Edge dù đảm bảo tính nhẹ và phản hồi nhanh, nhưng chưa đủ khả năng phân biệt các tình huống phức tạp.

Hướng phát triển trong tương lai

Trong thời gian tới, hướng phát triển của đề tài tập trung vào việc triển khai thử nghiệm hệ thống trên nhiều trạm đo thực tế để đánh giá độ ổn định và khả năng mở rộng; cải tiến mô hình tại Edge để nâng cao độ chính xác mà vẫn giữ được hiệu năng xử lý cục bộ; tự động hóa quy trình gán nhãn dữ liệu và mở rộng tích hợp thêm các nguồn dữ liệu khác khác như dòng chảy, vệ tinh thời tiết. Đây là những bước đi quan trọng nhằm đưa kết quả nghiên cứu vào thực tiễn, phục vụ công tác giám sát và ứng phó kịp thời với các sự cố môi trường.

TÀI LIỆU THAM KHẢO

[1] "Hệ thống đo mưa tự động Vrain cung cấp dữ liệu đo mưa để cảnh báo mưa lũ và vận hành hồ chứa," *Nông nghiệp & Môi trường*, 22/5/2024. [Online]. Xem tại: https://nongnghiepmoitruong.vn . [Ngày truy cập 24/4/2025].

[2] "Trạm đo mưa tự động Vrain," *Watec.vn*, [Online]. Xem tại: https://watec.vn. [Ngày truy cập 24/4/2025].

[3] "Hệ thống đo mưa tự động giúp giảm nhẹ thiên tai," *Nhân Dân*, 21/6/2021. [Online]. Xem tại: https://nhandan.vn. [Ngày truy cập 24/4/2025].

[4] M. Rausand, A. Barros, and A. Høyland, *System Reliability Theory: Models, Statistical Methods, and Applications*, 3rd ed. Hoboken: Wiley, 2020.

[5] International Organization for Standardization, *"Petroleum, petrochemical and natural gas industries – Collection and exchange of reliability and maintenance data for equipment,"* ISO 14224, 2016.

[6] European Committee for Standardization, *"Maintenance – Maintenance terminology,"* EN 13306, November 2017.

[7] Department of Defense, *"Electronic Reliability Design Handbook,"* MIL-HDBK-338B, October 1998.

[8] Association Française de Normalisation, *"Maintenance – Terminologie,"* NF X 60-010, 2001.

[9] Mobley, R. K. *An Introduction to Predictive Maintenance*. 2nd ed. Oxford: Butterworth-Heinemann, 2002.

[10] M. Ade and K. Sheriffdeen, "Leveraging Cloud-Based AI and ML for Predictive Maintenance: Enhancing Industrial IoT Systems," *ResearchGate*, Dec. 2024.

[11] C. Wilson-Clark, "Computers ranked as key literacy," *The West Australian*, para. 3, March 29, 2004. [Online]. Available: http://www.thewest.com.au. [Accessed May. 25, 2025].

[12] Peruthambi, V., Pandiri, L., Kaulwar, P. K., Koppolu, H. K. R., Adusupalli, B., & Pamisetty, A. (2025). *Big Data-Driven Predictive Maintenance for Industrial IoT (IIoT) Systems*. Metallurgical and Materials Engineering, 31(3), 21–30.

[13] Shanbhag, R. R., Dasi, U., Singla, N., Balasubramanian, R., & Benadikar, S. (2023). *Exploring the Use of Cloud-Based AI and ML for Real-Time Anomaly Detection and Predictive Maintenance in Industrial IoT Systems*. International Journal of Intelligent Systems and Applications in Engineering, 11(4), 925–937.

[14] Mohammed, N. A., Abdulateef, O. F., & Hamad, A. H. (2023). *An IoT and Machine Learning-Based Predictive Maintenance System for Electrical Motors*. Journal Européen des Systèmes Automatisés, 56(4), 651–656.

[15] M. J. C. S. Reis and C. Serôdio, "Edge AI for real-time anomaly detection in smart homes," *Future Internet*, vol. 17, no. 4, pp. 1–26, Apr. 2025.

[16] T. Szydlo, "Online anomaly detection based on reservoir sampling and LOF for IoT devices," *arXiv Preprint*, vol. arXiv:2206.14265, pp. 1–7, Jun. 2022.

[17] E. H. Budiarto, A. E. Permanasari, and S. Fauziati, "Unsupervised anomaly detection using K-means, local outlier factor and one class SVM," *Proc. Int. Conf. Sci. Technol. (ICST)*, pp. 1–6, Oct. 2019.

[18] D. Kong, D. Liu, L. Zhang, L. He, Q. Shi, and X. Ma, "Sensor anomaly detection in the industrial internet of things based on edge computing," *Turk. J. Electr. Eng. Comput. Sci.*, vol. 28, no. 1, pp. 331–346, Jan. 2020.

[19] Ruksana and J. Mathew, "Advancing predictive maintenance with Edge AI and IoT integration in industrial systems," *Int. J. Sci. Res. Eng. Trends*, vol. 11, no. 2, pp. 2155–2160, Mar.–Apr. 2025

[20] Y. Sun, A. K. C. Wong, and M. S. Kamel, "Classification of imbalanced data: A review," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 23, no. 4, pp. 687–719, Nov. 2009.