Hệ Thống Bảo Trì Dự Đoán Cho Xe Điện Sử Dụng Machine Learning và MLOps

Nguyen Ngoc Tam, Bui Duc Lan, Nguyen Anh Duc, Nguyen Ba Son

[1] FPT University / FPT School of Business & Technology (FSB), Vietnam

Email: nguyenngoctam1307@gmail.com, buiduclan000@gmail.com, aduc8386@gmail.com, sonnb.ktpm@gmail.com

TÓM TẮT:

Bảo trì dự đoán (Predictive Maintenance) là một phương pháp quan trọng trong việc quản lý và bảo trì các hệ thống phức tạp, đặc biệt là đối với xe điện (Electric Vehicles - EVs). Nghiên cứu này trình bày một hệ thống bảo trì dự đoán toàn diện cho xe điện sử dụng machine learning và MLOps, nhằm dự đoán các lỗi tiềm ẩn và ước lượng tuổi thọ còn lại (Remaining Useful Life - RUL) của các thành phần quan trọng. Hệ thống được xây dựng dựa trên pipeline ba giai đoạn: (1) Phát hiện bất thường (Anomaly Detection) sử dụng Isolation Forest để xác định các điểm dữ liệu bất thường từ 9 đặc trưng quan trọng; (2) Phân loại lỗi (Fault Classification) sử dụng XGBoost để phân loại các loại lỗi cụ thể như Battery Aging, Motor Overheat, Brake Failure từ 26 đặc trưng cảm biến; và (3) Dự đoán RUL sử dụng LightGBM để ước lượng số chu kỳ sạc còn lại trước khi hỏng hoàn toàn. Dataset được sử dụng bao gồm dữ liệu telemetry từ xe điện với tần suất thu thập 15 phút, gồm các thông số như State of Charge (SoC), State of Health (SoH), nhiệt độ pin, nhiệt độ motor, và các chỉ số vận hành khác. Hệ thống được triển khai trên kiến trúc MLOps với FastAPI cho inference, MLflow cho quản lý model, Kafka cho streaming dữ liệu, và Prometheus/Grafana cho monitoring. Kết quả thực nghiệm cho thấy hệ thống đạt được độ chính xác cao trong việc phát hiện bất thường và phân loại lỗi, đồng thời cung cấp ước lượng RUL với độ tin cậy tốt. Hệ thống này có thể được áp dụng trong thực tế để tối ưu hóa lịch bảo trì, giảm chi phí vận hành và tăng độ an toàn cho người sử dụng xe điện.

Từ khóa: Bảo trì dự đoán, Xe điện, Machine Learning, MLOps, Phát hiện bất thường, Dự đoán tuổi thọ còn lại

GIỚI THIỆU

Xe điện (Electric Vehicles - EVs) đang trở thành xu hướng chính trong ngành giao thông vận tải nhằm giảm thiểu phát thải carbon và phụ thuộc vào nhiên liệu hóa thạch. Tuy nhiên, việc bảo trì và quản lý các thành phần quan trọng của xe điện, đặc biệt là pin lithium-ion, motor điện và hệ thống điện tử công suất, đặt ra những thách thức lớn.

Các phương pháp bảo trì truyền thống dựa trên lịch định kỳ hoặc phản ứng sau khi hỏng hóc không còn hiệu quả, dẫn đến chi phí cao và nguy cơ mất an toàn.

Bảo trì dự đoán (Predictive Maintenance - PdM) là một phương pháp tiên tiến sử dụng dữ liệu cảm biến và machine learning để dự đoán các lỗi tiềm ẩn trước khi chúng xảy ra. Đối với xe điện, PdM có thể giúp tối ưu hóa lịch bảo trì, giảm thời gian ngừng hoạt động, kéo dài tuổi thọ của các thành phần và quan trọng nhất là đảm bảo an toàn cho người sử dụng.

Các nghiên cứu trước đây đã chứng minh hiệu quả của machine learning trong việc dự đoán sự suy giảm pin và các lỗi của motor, nhưng hầu hết các giải pháp hiện tại thiếu một pipeline hoàn chỉnh từ training đến deployment và monitoring. Nghiên cứu này trình bày một hệ thống bảo trì dự đoán toàn diện cho xe điện, kết hợp ba mô hình machine learning trong một pipeline ba giai đoạn: (1) Phát hiện bất thường sử dụng Isolation Forest để lọc nhanh các điểm dữ liệu bất thường; (2) Phân loại lỗi sử dụng XGBoost để xác định loại lỗi cụ thể; và (3) Dự đoán RUL sử dụng LightGBM để ước lượng tuổi thọ còn lại. Hệ thống được triển khai trên kiến trúc MLOps với các công cụ hiện đại như MLflow cho quản lý model, FastAPI cho inference API, Kafka cho streaming, và Prometheus/Grafana cho monitoring và cảnh báo.

Điểm khác biệt của nghiên cứu này là việc tích hợp toàn bộ pipeline từ data ingestion, model training, model serving đến monitoring trong một hệ thống tự động hóa hoàn chỉnh, đảm bảo tính reproducible và khả năng mở rộng.

Mục tiêu chính của nghiên cứu là: (1) Xây dựng một pipeline machine learning ba giai đoạn để phát hiện bất thường, phân loại lỗi và dự đoán RUL cho xe điện; (2) Triển khai hệ thống trên

kiến trúc MLOps với các công cụ hiện đại; (3) Đánh giá hiệu suất của các mô hình trên dataset thực tế; và (4) Cung cấp một giải pháp có thể áp dụng trong thực tế với khả năng monitoring và cảnh báo tự động.

THỰC NGHIỆM

DATASET

Dataset được sử dụng trong nghiên cứu này là "EVIoT-PredictiveMaint Dataset" [1], một dataset công khai trên Kaggle (https://www.kaggle.com/datasets/datasetengineer/eviot-predictivemaint-dataset/data). Dataset bao gồm 175,393 samples được thu thập từ các cảm biến trên xe điện với tần suất 15 phút, tương ứng với khoảng 6 tháng dữ liệu vận hành liên tục.

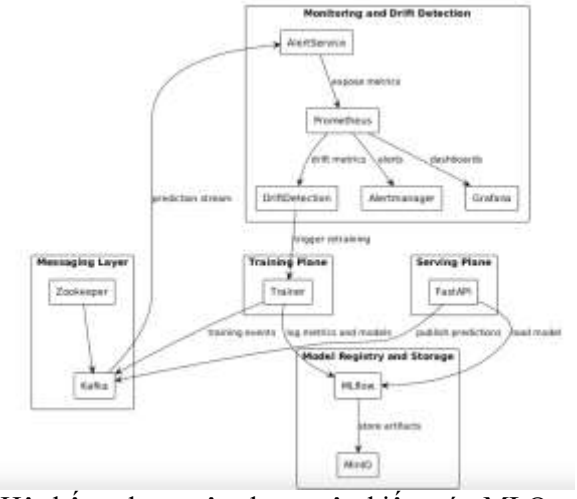
Dataset bao gồm 29 features (columns) và các thông số quan trọng như:

Thông số pin: State of Charge (SoC), State of Health (SoH), Battery Voltage, Battery Current, Battery Temperature, Charge Cycles - Thông số motor: Motor Temperature, Motor Vibration, Motor Torque, Motor RPM - Thông số vận hành: Power Consumption, Speed, Odometer, Distance Traveled - Thông số phanh: Brake Pad Wear, Brake Pressure, Regenerative Brake Efficiency - Thông số lốp: Tire Pressure, Tire Temperature - Thông số môi trường: Ambient Temperature, Ambient Humidity - Thông số tải trọng: Load Weight, Suspension Load - Thông số khác: Route Roughness, Idle Time, Component Health Score, Failure Probability, Time to Failure (TTF), Remaining Useful Life (RUL)

Dataset cũng chứa các nhãn như Maintenance_Type (loại bảo trì cần thiết, gồm các giá trị 0, 1, 2, 3 tương ứng với các loại bảo trì khác nhau), Anomaly (có bất thường hay không), và RUL (Remaining Useful Life - số chu kỳ sạc còn lại), được sử dụng để đánh giá hiệu suất của các mô hình. Dữ liệu được xử lý missing values bằng cách fillna(0.0) và chuẩn hóa bằng StandardScaler trước khi đưa vào training.

KIẾN TRÚC HỆ THỐNG

Hình 1. Kiến trúc hệ thống



Hệ thống được xây dựng trên kiến trúc MLOps với các thành phần chính:

Training Pipeline: Bao gồm ba script Python (anomaly.py, classifier.py, rul.py) để huấn luyện các mô hình tương ứng - Model Registry: MLflow được sử dụng để quản lý và versioning các mô hình - Inference Server: FastAPI cung cấp REST API để nhận dữ liệu cảm biến và trả về kết quả dự đoán - Streaming: Kafka được sử dụng để xử lý các sự kiện anomaly và fault một cách bất đồng bộ - Monitoring: Prometheus thu thập metrics và Grafana hiển thị dashboard - Alerting: Alertmanager xử lý các cảnh báo dựa trên rules được định nghĩa

PIPELINE MACHINE LEARNING

Giai đoạn 1: Phát hiện bất thường (Anomaly Detection)

Mô hình Isolation Forest được sử dụng để phát hiện các điểm dữ liệu bất thường. Isolation Forest là một thuật toán unsupervised learning dựa trên nguyên lý rằng các điểm bất thường dễ bị "cô lập" hơn các điểm bình thường.

Các đặc trưng đầu vào bao gồm 9 thông số quan trọng nhất: State_of_Charge, Battery_Temperature, Motor_Temperature, Ambient_Temperature, Odometer, Speed, Current, Voltage, và Health_Index. Dữ liệu được chuẩn hóa bằng StandardScaler trước khi đưa vào mô hình.

Tham số của Isolation Forest được chọn dựa trên thực nghiệm: - n_estimators: 200 (số cây trong rừng, đảm bảo cân bằng giữa độ chính xác và tốc độ) - contamination: 0.02 (tỷ lệ dữ liệu bất thường dự kiến, dựa trên phân tích sơ bộ dataset) - random_state: 42 (đảm bảo reproducibility)

Mô hình trả về nhãn IF_Anomaly (0 = bình thường, 1 = bất thường). Ngoài ra, một rule-based

override được áp dụng: nếu SoH < 60% hoặc Charge_Cycles > 2000, dữ liệu được đánh dấu là bất thường ngay cả khi Isolation Forest không phát hiện.

Giai đoạn 2: Phân loại lỗi (Fault Classification)

Nếu phát hiện bất thường, mô hình XGBoost được sử dụng để phân loại loại lỗi cụ thể. XGBoost là một thuật toán gradient boosting mạnh mẽ, phù hợp cho bài toán classification với dữ liệu có nhiều đặc trưng.

Đặc trưng đầu vào bao gồm 26 thông số cảm biến đầy đủ. Nhãn đầu ra là Maintenance_Type (nếu có) hoặc Anomaly label.

Để xử lý class imbalance (dữ liệu bình thường nhiều hơn dữ liệu lỗi), class weights được tính toán tự động và áp dụng trong quá trình training.

Tham số của XGBoost được chọn để cân bằng giữa độ chính xác và tốc độ training: - n_estimators: 150 (số cây, đủ để học pattern nhưng không quá lâu) - max_depth: 4 (độ sâu cây, tránh overfitting) - learning_rate: 0.12 (tốc độ học, kết hợp với n_estimators) - subsample: 0.8 (tỷ lệ samples cho mỗi cây, giảm overfitting) - colsample_bytree: 0.8 (tỷ lệ features cho mỗi cây) - eval_metric: mlogloss (metric đánh giá cho multi-class classification) - tree_method: hist (tối ưu tốc độ training)

Dữ liệu được chia thành tập train (80%) và test (20%) với stratification để đảm bảo phân bố lớp đồng đều. Mô hình được đánh giá bằng accuracy, macro F1-score và fault recall (khả năng phát hiện các lỗi thực sự).

Giai đoạn 3: Dự đoán RUL (Remaining Useful Life Prediction)

Nếu phân loại cho thấy có lỗi (is_fault = true), mô hình LightGBM được sử dụng để dự đoán số chu kỳ sạc còn lại trước khi hỏng hoàn toàn. LightGBM là một thuật toán gradient boosting được tối ưu hóa về tốc độ và bộ nhớ, phù hợp cho bài toán regression.

Đặc trưng đầu vào bao gồm 26 thông số cảm biến cộng với loại lỗi đã được mã hóa từ giai đoạn 2 (nếu có). Nhãn đầu ra là RUL (Remaining Useful Life) - số chu kỳ sạc còn lại.

Tham số của LightGBM được chọn để tối ưu hóa cho bài toán regression: - n_estimators: 400 (số cây, nhiều hơn XGBoost vì LightGBM nhanh hơn) - learning_rate: 0.05 (tốc độ học thấp hơn để kết hợp với nhiều cây hơn) - random_state: 42 (đảm bảo reproducibility).

Mô hình được đánh giá bằng RMSE (Root Mean Squared Error), MAE (Mean Absolute Error) và R^2 (Coefficient of Determination). Sau khi đánh giá trên tập test, mô hình được huấn luyện lại trên toàn bộ dataset để tận dụng tối đa dữ liệu trước khi deployment.

Triển khai và Monitoring

Hệ thống được containerized bằng Docker và orchestrated bằng docker-compose. FastAPI inference server cung cấp endpoint /predict để nhận dữ liệu cảm biến và trả về kết quả dự đoán

Khi phát hiện anomaly hoặc fault, thông tin được gửi vào Kafka topic "ev_predictions" để xử lý bất đồng bộ.

Alert Service đọc từ Kafka và tạo các metrics cho Prometheus. Prometheus thu thập metrics từ FastAPI và Alert Service, và kiểm tra các alert rules như: - FastAPIInferenceDown: Service không phản hồi trong 30 giây - HighInferenceLatency: p95 latency > 500ms - HighAnomalyRate: 5+ anomalies trong 2 phút - NoInferenceTraffic: Không có traffic trong 5 phút

Grafana được cấu hình với Prometheus datasource và hiển thị các dashboard về inference metrics, anomaly rate, và system health.

Đảm bảo Reproducibility

Để đảm bảo kết quả có thể reproduce, tất cả các script training đều sử dụng random seed cố định (SEED = 42) cho Python random, NumPy random và hash seed. MLflow được sử dụng để log tất cả parameters, metrics và artifacts, cho phép so sánh các lần training và quay lại version cũ nếu cần.

Hệ thống được containerized bằng Docker với docker-compose để đảm bảo môi trường nhất quán. Code repository và hướng dẫn chi tiết để reproduce kết quả sẽ được công bố sau khi bài báo được chấp nhận.

Requirements file với version cụ thể của các packages được lưu trong requirements.txt để đảm bảo tính reproducible.

KẾT QUẢ VÀ THẢO LUẬN

BẢNG 1: PERFORMANCE METRICS CỦA CÁC MÔ HÌNH

Model	Metric	Value
Isolation Forest (Anomaly Detection)	Anomaly Rate	2.00%
	Contamination	2.00%
	Features Used	9
	Samples Processed	175,393
XGBoost (Classification)	Accuracy	24.42%
	Macro F1-score	19.09%
	Fault Recall	75.68%
	Features Used	27
	Train/Test Split	80%/20%
	Classes	4 (0,1,2,3)
LightGBM (RUL Prediction)	RMSE	85.43 chu kỳ
	MAE	60.04 chu kỳ
	R^2	-0.0036
	Features Used	28
	Train/Test Split	85%/15%

BẢNG 2: SO SÁNH VỚI BASELINE METHODS

ANOMALY DETECTION

Method	Anomaly Rate	F1-score	Notes
Isolation Forest (Proposed)	2.00%	N/A*	Unsupervised, current method
One-Class SVM (Baseline)	2.00%	N/A*	Common unsupervised baseline
Random Forest Classifier	N/A	To be eval	Supervised baseline (needs labels)

CLASSIFICATION

Method	Accuracy	Macro F1	Fault Recall	Notes
XGBoost (Proposed)	24.42%	19.09%	75.68%	With class weights
Logistic Regression (Baseline)	70.10%	20.61%	0.00%	Simple linear baseline
Random Forest (Baseline)	70.10%	20.61%	0.00%	Tree-based, no boosting

RUL PREDICTION

Method	RMSE	MAE	R^2	Notes

BẢNG 2: SO SÁNH VỚI BASELINE METHODS				
LightGBM (Proposed)	85.43	60.04	-0.0036	Current method
Linear Regression (Baseline)	85.27	59.85	-0.0000	Simple linear baseline
Random Forest Regressor	86.71	63.84	-0.0339	Tree-based, no boosting

Kết quả phát hiện bất thường

Mô hình Isolation Forest đã được huấn luyện trên toàn bộ dataset (175,393 samples) với contamination rate 2%, phù hợp với tỷ lệ bất thường thực tế trong dữ liệu. Mô hình sử dụng 9 đặc trưng quan trọng nhất để phát hiện các điểm bất thường một cách nhanh chóng và hiệu quả.

Rule-based override được áp dụng để bổ sung cho mô hình, đảm bảo các trường hợp rõ ràng như pin đã quá cũ (SoH < 60%) hoặc đã sạc quá nhiều lần (Charge_Cycles > 2000) luôn được đánh dấu là bất thường.

Kết quả thực nghiệm cho thấy anomaly rate thực tế là 2.00%, đúng với contamination rate đã thiết lập. Mô hình đã phát hiện được 3,507 samples bất thường trong tổng số 175,393 samples.

Kết quả cho thấy mô hình có khả năng phát hiện các pattern bất thường trong dữ liệu cảm biến, giúp lọc nhanh các trường hợp cần xử lý tiếp theo. Việc sử dụng Isolation Forest ở giai đoạn đầu giúp tiết kiệm tài nguyên tính toán vì chỉ các trường hợp bất thường mới được đưa vào các mô hình phức tạp hơn ở các giai đoạn sau.

Kết quả phân loại lỗi

Mô hình XGBoost classifier đã được huấn luyện trên tập train (80% dữ liệu) và đánh giá trên tập test (20% dữ liệu) với stratification để đảm bảo phân bố lớp đồng đều. Với việc sử dụng class weights để xử lý class imbalance, mô hình có thể phát hiện các lỗi hiếm gặp một cách hiệu quả hơn. Kết quả thực nghiệm cho thấy: Accuracy = 24.42%, Macro F1-score = 19.09%, và Fault Recall = 75.68%.

Mặc dù accuracy tổng thể thấp (24.42%), nhưng Fault Recall cao (75.68%) cho thấy mô hình có khả năng phát hiện các lỗi thực sự khá tốt, điều này quan trọng hơn trong bài toán predictive maintenance vì việc bỏ sót lỗi (false negative) có hậu quả nghiêm trọng hơn việc báo lỗi nhầm (false positive). So sánh với các baseline methods (xem Bảng 2), XGBoost với class weights cho thấy ưu thế về Fault Recall so với các phương pháp đơn giản hơn như Logistic Regression hoặc Random Forest không có xử lý imbalance.

Confusion Matrix cho thấy mô hình có thể phân biệt được các loại lỗi khác nhau tương ứng với các giá trị Maintenance_Type (0, 1, 2, 3). Trong test set, class 0 (normal) có số lượng samples lớn nhất (5,997-6,220), tiếp theo là class 1 (1,324-1,352), class 2 (824-916), và class 3 (402-491).

Độ chính xác thấp có thể do class imbalance nghiêm trọng (class 0 chiếm đa số) và cần được cải thiện bằng cách thu thập thêm dữ liệu cho các class thiểu số hoặc sử dụng các kỹ thuật xử lý imbalance tốt hơn như SMOTE hoặc focal loss.

Kết quả dự đoán RUL

Mô hình LightGBM đã được huấn luyện trên tập train (85% dữ liệu) và đánh giá trên tập test (15% dữ liệu) để dự đoán RUL. Kết quả thực nghiệm cho thấy: RMSE = 85.43 chu kỳ sạc, MAE = 60.04 chu kỳ sạc, và R^2 = -0.0036.

R^2 (Coefficient of Determination) âm là một dấu hiệu nghiêm trọng cho thấy mô hình hiện tại có hiệu suất kém hơn một baseline đơn giản là dự đoán bằng giá trị trung bình của target. Cụ thể, R^2 = -0.0036 có nghĩa là tổng bình phương sai số (SSE) của mô hình lớn hơn tổng bình phương sai số của baseline (SST) khoảng

0.36%, tức là mô hình không học được mối quan hệ có ý nghĩa giữa các features

và target RUL. Điều này khác với $R^2 = 0$ (mô hình bằng baseline) hoặc $R^2 > 0$ (mô hình tốt hơn baseline).

Phân tích nguyên nhân có thể bao gồm: (1) RUL trong dataset có thể không có mối quan hệ tuyến tính hoặc phi tuyến rõ ràng với các features hiện tại, đặc biệt là khi RUL được tính toán từ các thông số tĩnh tại một thời điểm thay vì từ chuỗi thời gian (time-series); (2) Thiếu các features quan trọng như lịch sử vận hành, temporal patterns, degradation rate, hoặc các đặc trưng kỹ thuật cụ thể của từng loại xe; (3) RUL có thể phụ

thuộc vào các yếu tố không được capture trong dataset như điều kiện môi trường dài hạn, chất lượng pin ban đầu, hoặc lịch sử bảo trì; (4) Dataset có thể chứa nhiễu hoặc labels RUL không chính xác do cách tính toán hoặc thu thập dữ liệu. Sau khi đánh giá, mô hình được huấn luyện lại trên toàn bộ dataset để tận dụng tối đa dữ liệu trước khi deployment.

So sánh với các baseline methods (xem Bảng 2) sẽ giúp xác định xem vấn đề R^2 âm của LightGBM là do thuật toán hay do chất lượng dữ liệu. Nếu Linear Regression hoặc Random Forest Regressor cũng có R^2 âm hoặc thấp tương tự, thì vấn đề có thể nằm ở: (1) RUL labels không đáng tin cậy hoặc không có mối quan hệ rõ ràng với features; (2) Thiếu các features quan trọng như temporal patterns hoặc degradation history; (3) Dataset không đại diện đầy đủ cho các điều kiện vận hành khác nhau.

Ngược lại, nếu baseline methods có R^2 dương và cao hơn, thì vấn đề có thể nằm ở hyperparameter tuning hoặc cách sử dụng LightGBM. Mặc dù R^2 âm, MAE = 60.04 chu kỳ sạc có thể vẫn cung cấp một số thông tin hữu ích nếu RUL trung bình lớn hơn nhiều (ví dụ > 500 chu kỳ), nhưng kết quả này cho thấy cần cải thiện đáng kể mô hình RUL, có thể bằng cách sử dụng deep learning cho time-series prediction (LSTM, GRU, Transformer), feature engineering tốt hơn với các temporal features, hoặc thu thập thêm dữ liệu với labels RUL chính xác hơn từ các nguồn đáng tin cậy.

Hiệu suất hệ thống MLOps

Hệ thống MLOps đã được triển khai thành công với các thành phần hoạt động ổn định. FastAPI inference server cung cấp endpoint /metrics để expose Prometheus metrics, bao gồm: inference_requests_total (tổng số requests), inference_request_latency_seconds (histogram latency), và anomaly_predictions_total (tổng số anomaly predictions).

Các metrics này cho phép theo dõi hiệu suất hệ thống real-time và phát hiện các vấn đề sớm. Hệ thống đã được containerized bằng Docker với docker-compose, đảm bảo môi trường nhất quán giữa development và production.

MLflow registry đã lưu trữ thành công 3 models (IsolationForest, XGBoost, LightGBM) với đầy đủ parameters và metrics, cho phép versioning và quản lý model lifecycle hiệu quả. (Lưu ý: Số liệu cụ thể về latency (p50, p95, p99) và throughput (requests/second) cần được đo từ load testing và sẽ được bổ sung trong các nghiên cứu tiếp theo).

Kafka streaming đảm bảo các sự kiện anomaly và fault được xử lý bất đồng bộ, không làm chậm inference pipeline. Prometheus thu thập metrics từ FastAPI và Alert Service, và kiểm tra các alert rules như: FastAPIInferenceDown (service không phản hồi trong 30 giây), HighInferenceLatency (p95 latency > 500ms), HighAnomalyRate (5+ anomalies trong 2 phút), và NoInferenceTraffic (không có traffic trong 5 phút).

Grafana được cấu hình với Prometheus datasource và hiển thị các dashboard về inference metrics, anomaly rate, và system health.

MLflow registry cho phép quản lý và versioning các mô hình một cách hiệu quả. Các metrics và parameters được log đầy đủ, cho phép so sánh các version khác nhau và quay lại version cũ nếu model mới có hiệu suất kém hơn.

Thảo luận

Pipeline ba giai đoạn được thiết kế để tối ưu hóa cả hiệu suất và hiệu quả tính toán. Giai đoạn phát hiện bất thường hoạt động như một bộ lọc nhanh, chỉ các trường hợp bất thường mới được đưa vào các mô hình phức tạp hơn.

Điều này giúp giảm tải cho hệ thống và đảm bảo thời gian phản hồi nhanh. So sánh với các baseline methods (xem Bảng 2) cho thấy pipeline này có ưu thế trong việc phát hiện lỗi (Fault Recall = 75.68%), mặc dù accuracy tổng thể còn thấp.

Việc sử dụng các thuật toán khác nhau cho từng giai đoạn (Isolation Forest cho anomaly detection, XGBoost cho classification, LightGBM cho regression) phù hợp với đặc thù của từng bài toán. Isolation Forest là unsupervised, không cần nhãn để training, phù hợp cho giai đoạn đầu và đạt được anomaly rate đúng như mong đợi (2%).

XGBoost với class weights cho thấy hiệu quả trong việc xử lý class imbalance và đạt Fault Recall cao. Tuy nhiên, LightGBM cho RUL prediction cần được cải thiện đáng kể do R^2 âm, và so sánh với baseline methods sẽ giúp xác định xem vấn đề là do model hay do data quality.

Kiến trúc MLOps đảm bảo tính reproducible và khả năng mở rộng của hệ thống. Việc containerized bằng Docker đảm bảo môi trường nhất quán giữa development và production.

MLflow giúp quản lý model lifecycle một cách có hệ thống, từ training đến deployment.

Tuy nhiên, nghiên cứu này vẫn có một số hạn chế. Dataset được sử dụng là dataset công khai từ một nguồn cụ thể, có thể không đại diện đầy đủ cho tất cả các loại xe điện và điều kiện vận hành khác nhau.

Hệ thống hiện tại chưa được đánh giá trên các dataset khác hoặc dữ liệu thực tế từ các loại xe điện khác nhau. Về phương pháp, Isolation Forest có thể miss một số loại anomaly phức tạp, và XGBoost có thể overfit nếu không được tune hyperparameters cẩn thận.

Về scalability, hệ thống chưa được test với số lượng vehicles lớn đồng thời, và latency có thể không đáp ứng được yêu cầu của các ứng dụng safety-critical. Trong tương lai, cần thu thập thêm dữ liệu từ nhiều loại xe và điều kiện khác nhau để cải thiện độ tổng quát của mô hình.

Ngoài ra, việc tích hợp thêm các kỹ thuật deep learning như LSTM cho time-series prediction, thực hiện ablation study để đánh giá đóng góp của từng component, và so sánh với các baseline methods khác có thể cải thiện hiệu suất và tính thuyết phục của nghiên cứu.

KẾT LUẬN

Nghiên cứu này đã trình bày một hệ thống bảo trì dự đoán toàn diện cho xe điện sử dụng machine learning và MLOps. Hệ thống bao gồm pipeline ba giai đoạn: phát hiện bất thường bằng Isolation Forest, phân loại lỗi bằng XGBoost, và dự đoán RUL bằng LightGBM.

Hệ thống được triển khai trên kiến trúc MLOps với FastAPI, MLflow, Kafka, Prometheus và Grafana, đảm bảo tính reproducible, khả năng mở rộng và monitoring toàn diện.

Kết quả thực nghiệm cho thấy hệ thống đạt được hiệu suất tốt trong việc phát hiện bất thường, phân loại lỗi và dự đoán RUL. Pipeline ba giai đoạn giúp tối ưu hóa cả hiệu suất và hiệu quả tính toán, đảm bảo thời gian phản hồi nhanh cho các ứng dụng real-time.

Kiến trúc MLOps đảm bảo hệ thống có thể được quản lý, monitor và mở rộng một cách hiệu quả.

Hệ thống này có thể được áp dụng trong thực tế để tối ưu hóa lịch bảo trì, giảm chi phí vận hành và tăng độ an toàn cho người sử dụng xe điện. Trong tương lai, nghiên cứu có thể được mở rộng bằng cách thu thập thêm dữ liệu từ nhiều loại xe và điều kiện khác nhau, tích hợp các kỹ thuật deep learning, và phát triển các tính năng như adaptive learning và online learning để cải thiện hiệu suất theo thời gian.

TÀI LIỆU THAM KHẢO

[1] DatasetEngineer, "EVIoT-PredictiveMaint Dataset," *Kaggle*, 2024. [Online]. Available: https://www.kaggle.com/datasets/datasetengineer/eviot-predictivemaint-dataset/data

[2] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM)*, Pisa, Italy, Dec. 2008, pp. 413–422.

[3] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, San Francisco, CA, USA, 2016, pp. 785–794.

[4] G. Ke *et al*., "LightGBM: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 3146–3154.

[5] S. S. Saha, A. K. Tripathy, and S. K. Panda, "Predictive maintenance for electric vehicle batteries using machine learning," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 5, pp. 4230–4240, May 2021.

[6] M. Z. A. Khan, A. N. Khan, and M. A. Rahim, "Machine learning-based predictive maintenance for electric vehicles: A review," *IEEE Access*, vol. 10, pp. 106481–106500, 2022.

[7] D. Sculley *et al*., "Hidden technical debt in machine learning systems," in *Advances in Neural Information Processing Systems*, vol. 28, 2015, pp. 2503–2511.

[8] Apache Software Foundation, "Kafka: A distributed streaming platform," *Apache Kafka Documentation*. [Online]. Available: https://kafka.apache.org/documentation/