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 HÓA

ĐỀ TÀI:

NGHIÊN CỨU HỆ THỐNG THU THẬP DỮ LIỆU VÀ GIÁM SÁT TIÊU THỤ ĐIỆN BẰNG DEEP LEARNING

Người hướng dẫn: **TS. NGUYỄN THỊ KIM TRÚC**
Cán bộ hướng dẫn: **HUỲNH KIM SƠN**
HUỲNH THẢO NGUYÊN

Sinh viên thực hiện:
TRƯƠNG CÔNG CƯỜNG - MSSV: 105200294 – LỚP: 20TDH1
NHAN NGỌC SANG - MSSV: 105200425 – LỚP: 20TDHCLC2
PHAN ĐỨC NGỌC - MSSV: 105200418 – LỚP: 20TDHCLC2

Đà Nẵng, 6/2025

TÓM TẮT

Tên đề tài: ***Nghiên cứu hệ thống thu thập dữ liệu và giám sát tiêu thụ điện bằng Deep Learning.***

Sinh viên thực hiện:	Trương Công Cường	- MSSV: 105200294
	Nhan Ngọc Sang	- MSSV: 105200425
	Phan Đức Ngọc	- MSSV: 105200418

Tiêu thụ điện năng là một chỉ số quan trọng phản ánh hiệu quả vận hành và mức độ sử dụng năng lượng của các hộ gia đình, doanh nghiệp và toàn bộ hệ thống điện. Trong bối cảnh hiện nay, khi nhu cầu điện ngày càng gia tăng và áp lực tiết kiệm năng lượng, giảm phát thải ngày càng rõ rệt, thì việc giám sát và phân tích dữ liệu tiêu thụ điện một cách chính xác và liên tục trở nên vô cùng cần thiết. Tuy nhiên, hệ thống thu thập và xử lý dữ liệu truyền thống còn nhiều hạn chế như độ trễ cao, thiếu khả năng phát hiện bất thường theo thời gian thực, và khó mở rộng với quy mô lớn. Để khắc phục những thách thức đó, nhóm đã triển khai nghiên cứu đề tài: "Nghiên cứu hệ thống thu thập dữ liệu và giám sát tiêu thụ điện bằng Deep Learning". Đề tài hướng đến việc xây dựng một hệ thống thông minh có khả năng thu thập, xử lý và phân tích dữ liệu tiêu thụ điện một cách tự động và hiệu quả. Thông qua việc ứng dụng các mô hình học sâu (Deep Learning), hệ thống không chỉ hỗ trợ dự báo mức tiêu thụ điện mà còn phát hiện các bất thường trong hành vi sử dụng điện, từ đó góp phần nâng cao hiệu quả quản lý năng lượng và cảnh báo sớm các sự cố tiềm ẩn trong hệ thống.

Đề tài "Nghiên cứu hệ thống thu thập dữ liệu và giám sát tiêu thụ điện bằng Deep Learning" sẽ tập trung xây dựng ba mô hình dự báo của thuật toán học sâu để dự báo chỉ số tiêu thụ điện của khu vực dân cư ở Đà Nẵng. Ba mô hình đó là Neural Hierarchical Interpolation for Time Series (N-HiTs), Long Short Term Memory (LSTM), Bidirectional Long Short-Term Memory (BiLSTM). Kết quả mô hình được đánh giá thông qua hàm Căn bậc hai sai số bình phương trung bình (Root Mean Squared Error - RMSE) và hàm Sai số phần trăm tuyệt đối trung bình (Mean Absolute Percentage Error – MAPE).

Kết quả đề tài cho thấy rằng mô hình N-HiTs cho kết quả tốt nhất trong ba mô hình. Và việc ứng dụng trí tuệ nhân tạo để dự báo chỉ số điện và dự báo bất thường trong tiêu thụ điện năng là hoàn toàn khả thi.

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	Trương Công Cường	105200294	20TDH1	Kỹ thuật Điều khiển và Tự động hóa
2	Nhan Ngọc Sang	105200425	20TDHCLC2	Kỹ thuật Điều khiển và Tự động hóa
3	Phan Đức Ngọc	105200418	20TDHCLC2	Kỹ thuật Điều khiển và Tự động hóa

1. Tên đề tài đồ án:

Nghiên Cứu Hệ Thống Thu Thập Dữ Liệu Và Giám Sát Tiêu Thụ Điện Bằng Deep Learning

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	Trương Công Cường	- Tìm hiểu tổng quan tiêu thụ điện, tình hình phát triển hệ thống đo đếm từ xa, lập trình Python, thu thập dữ liệu từ xa - Nghiên cứu các mô hình học sâu và công nghệ AI trong giám sát tiêu thụ điện. - Tìm hiểu thư viện và công cụ triển khai mô hình học sâu trên nền tảng Python và web.
2	Nhan Ngọc Sang	
3	Phan Đức Ngọc	

b. Phần riêng:

TT	Họ tên sinh viên	Nội dung
1	Trương Công Cường	Xây dựng mô hình dự báo LSTM và Bi-LSTM, huấn luyện, đánh giá mô hình, phân tích kết quả
2	Nhan Ngọc Sang	Xử lý dữ liệu đầu vào, xây dựng mô hình dự báo N-HiTs, huấn luyện, đánh giá mô hình, phân tích kết quả

3	Phan Đức Ngọc	Xử lý dữ liệu đầu vào, Xây dựng và triển khai giao diện web tích hợp mô hình dự báo phát hiện bất thường.

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

TT	Họ tên sinh viên	Nội dung
1	Trương Công Cường	
2	Phan Đức Ngọc	
3	Nhan Ngọc Sang	

6. Họ tên người hướng dẫn:	*Phần/ Nội dung:*
TS. Nguyễn Thị Kim Trúc	
Huỳnh Thảo Nguyên	
Huỳnh Kim Sơn	

7. Ngày giao nhiệm vụ đồ án: 07/03/2025

8. Ngày hoàn thành đồ án: 14/06/2025

Đà Nẵng, ngày tháng năm 2025

Trưởng Bộ môn Tự động hóa

TS. Giáp Quang Huy

Người hướng dẫn 1

TS. Nguyễn Thị Kim Trúc

Người hướng dẫn 2

Huỳnh Kim Sơn

Người hướng dẫn 3

Huỳnh Thảo Nguyên

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: Trương Công Cường Số thẻ SV :105200294

Nhan Ngọc Sang 105200425

Phan Đức Ngọc 105200418

Tên đề tài ĐATN: Nghiên cứu hệ thống thu thập dữ liệu và giám sát tiêu thụ điện bằng deep learning

Họ tên người HD: TS. Nguyễn Thị Kim Trúc

Đơn vị: Trường Đại học Bách khoa – Đại học Đà Nẵng

Tuần	**Ngày**	**Khối lượng**		**GVHD ký tên**
		đã thực hiện (%)	**tiếp tục thực hiện (%)**	
1	7/03/2025	Tìm hiểu đề tài, tổng quan về hệ thống điện và vấn đề giám sát tiêu thụ điện		
2	14/03/2025	Lập trình Python cơ bản, làm quen với môi trường Colab, thư viện Pandas, Numpy		
3	21/03/2025	Thu thập và khảo sát dữ liệu tiêu thụ điện từ hệ thống đo xa tại Đà Nẵng		
4	28/03/2025	Duyệt lần 1: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
5	04/04/2025	Tiền xử lý dữ liệu: xử lý NaN, nội suy, chuẩn hóa, tạo đặc trưng thời gian		
6	11/04/2025	Phân chia dữ liệu thành tập Train, Validation, Test		
7	18/04/2025	Xây dựng mô hình N-HiTs và huấn luyện bước đầu		
8	25/04/2025	Duyệt lần 2: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
9	02/05/2025	Xây dựng mô hình LSTM, Bi-LSTM và so sánh kết quả		

10	09/05/2025	Đánh giá mô hình bằng MAE, RMSE, MAPE – chọn mô hình tốt nhất để triển khai		
11	23/05/2025	Viết giao diện web hiển thị dữ liệu, dự báo và phát hiện bất thường		
12	30/05/2025	Duyệt lần 3: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
13	06/06/2025	Triển khai mô hình lên web (Streamlit) – thử nghiệm với dữ liệu thực tế		
14	10/06/2025	Kiểm tra, đánh giá mô hình đã triển khai trên web		
15	14/06/2025	Tổng hợp báo cáo, hoàn chỉnh đồ án		

LỜI NÓI ĐẦU

Lời đầu tiên, nhóm chúng em xin gởi lời cảm ơn chân thành nhất đến TS. Nguyễn Thị Kim Trúc cùng các giảng viên bộ môn Tự động hóa, khoa Điện – Trường Đại học Bách khoa – Đại học Đà Nẵng. Trong quá trình học tập và thực hiện đồ án tốt nghiệp, chúng em đã nhận được sự quan tâm giúp đỡ, hướng dẫn rất tận hình, tâm huyết của Cô. Bên cạnh đó, chúng em cũng xin chân thành cảm ơn anh Huỳnh Kim Sơn - Phòng Điều Độ - Điện Lực Đà Nẵng và anh Huỳnh Thảo Nguyên - Phòng Kinh Doanh Điện Lực Đà Nẵng cùng tất cả các anh kỹ sư Điện tại phòng Điều Độ Điện Lực Đà Nẵng đã giúp chúng em rất tận tình trong lúc chúng em thực tập cũng như trong suốt quá trình làm đồ án. Chúng em xin chân thành cảm ơn các thầy cô trong trường Đại học Bách khoa – Đại học Đà Nẵng, cùng tất cả các Thầy cô trong khoa Điện đã truyền đạt kiến thức về các môn cơ sở và các môn chuyên ngành để chúng em có được cơ sở lý thuyết vững chắc để thực hiện đồ án tốt nghiệp này cũng như là những hành trang để thực hiện những nghiên cứu khác.

Chúng em xin chân thành cảm ơn!

Đà Nẵng, ngày 16 tháng 06 năm 2025

LỜI CAM ĐOAN LIÊM CHÍNH HỌC THUẬT

Em xin cam đoan đề tài: ***"Nghiên cứu hệ thống thu thập dữ liệu và giám sát tiêu thụ điện bằng Deep Learning"*** là sự nghiên cứu của nhóm em, dưới sự hướng dẫn của TS. Nguyễn Thị Kim Trúc (Khoa Điện – Trường Đại học Bách khoa – Đại học Đà Nẵng), anh Huỳnh Kim Sơn (Phó Phòng Điều Độ - Điện Lực Đà Nẵng) và anh Huỳnh Thảo Nguyên (Phòng Kinh Doanh- Điện Lực Đà Nẵng). Các số liệu kết quả chưa từng được công bố trong bất kì công trình khoa học nào khác.

Toàn bộ thông tin tham khảo phục vụ đồ án đều được nêu ở phần Tài liệu tham khảo.

Các số liệu, kết quả trong báo cáo này là hoàn toàn trung thực. Chúng em xin chịu hoàn toàn trách nhiệm, kỉ luật của khoa Điện và Nhà trường nếu như phát hiện có vấn đề sao chép không hợp lệ xảy ra.

Nhóm sinh viên thực hiện.

Trương Công Cường Nhan Ngọc Sang Phan Đức Ngọc

MỤC LỤC

DANH SÁCH CÁC BẢNG, HÌNH VẼ

MỞ ĐẦU

1. Lý do chọn đề tài.

Trong bối cảnh hiện đại, nhu cầu sử dụng điện ngày càng tăng cao, kéo theo những thách thức trong việc giám sát và kiểm soát sản lượng tiêu thụ. Việc phát hiện kịp thời các trường hợp tiêu thụ điện bất thường không chỉ giúp nâng cao hiệu quả vận hành của hệ thống điện mà còn góp phần giảm thất thoát điện năng, ngăn chặn gian lận và tối ưu hóa chi phí cho cả nhà cung cấp và khách hàng.

Hiện nay, nhiều hệ thống quản lý điện vẫn chủ yếu dựa vào phương pháp truyền thống, thiếu tính tự động và chưa tận dụng triệt để các công nghệ hiện đại như trí tuệ nhân tạo (AI), IoT (Internet of Things) và phân tích dữ liệu lớn (Big Data). Điều này dẫn đến việc phát hiện sai lệch sản lượng tiêu thụ còn chậm trễ và kém chính xác, gây tổn thất đáng kể cho ngành điện.

Do đó, đề tài ***"Nghiên cứu hệ thống thu thập dữ liệu và giám sát tiêu thụ điện bằng Deep Learning"*** được lựa chọn với mục tiêu xây dựng một hệ thống giám sát thông minh, có khả năng phân tích dữ liệu theo thời gian thực và cảnh báo khi phát hiện tiêu thụ điện bất thường. Hệ thống này không chỉ giúp nâng cao hiệu quả quản lý điện năng mà còn hỗ trợ người dùng tối ưu hóa việc sử dụng điện, hướng tới một hệ thống điện thông minh và bền vững hơn.

2. Mục tiêu nghiên cứu.

Mục đích của đồ án tốt nghiệp là ứng dụng các phương pháp Deep Learning hiện đại để giải quyết hai bài toán quan trọng trong quản lý tiêu thụ điện năng:

- Dự báo sản lượng điện năng tiêu thụ theo thời gian
- Phát hiện bất thường trong tiêu thụ điện năng

3. Đối tượng và phạm vi nghiên cứu.

3.1. Đối tượng nghiên cứu:

Đối tượng nghiên cứu của đồ án là các khách hàng sử dụng điện và dữ liệu tiêu thụ điện tại TP. Đà Nẵng, cùng với mạng nơ-ron nhân tạo và các thuật toán tối ưu hóa được ứng dụng để dự báo sản lượng điện tiêu thụ theo thời gian và phát hiện bất thường trong tiêu thụ.

3.2. Phạm vi nghiên cứu:

Phạm vi nghiên cứu trong đề tài bao gồm:

- Mạng nơron nhiều lớp truyền thẳng và các thuật toán tối ưu hóa dùng để xây dựng mô hình dự báo sản lượng điện năng tiêu thụ theo thời gian và thuật toán phát hiện bất thường trong tiêu thụ điện năng
- Nghiên cứu lập trình tính toán trên ngôn ngữ lập trình Python cho việc huấn luyện mô hình dự báo.
- Xây dựng giao diện Web để dự báo sản lượng điện năng tiêu thụ theo thời gian và phát hiện bất thường trong tiêu thụ điện năng.

4. Phương pháp nghiên cứu.

Tìm kiếm và nghiên cứu thông tin trong các sách, các bài báo trong và ngoài nước liên qua đến các kỹ thuật xử lý số liệu, mạng nơron, các thuật toán tối ưu, lập trình tính toán.

5. Kết cấu của đề tài.

Nội dung của đồ án gồm 3 chương chính sau:

- Chương 1: Tổng quan về năng lượng điện và Công Ty điện lực Đà Nẵng
- Chương 2: Hệ thống thu thập dữ liệu đo đếm từ xa
- Chương 3: Cơ sở lý thuyết và giải pháp dự báo sản lượng điện tiêu thụ và xác định bất thường
- Chương 4: Kết quả thực nghiệm và đánh giá mô hình
- Chương 5: Xây dựng giao diện web.
- Chương 6: Kết luận và hướng phát triển đề tài

CHƯƠNG 1: TỔNG QUAN VỀ NĂNG LƯỢNG ĐIỆN VÀ PHƯƠNG PHÁP GIÁM SÁT TIÊU THỤ ĐIỆN NĂNG

1.1. Giới thiệu về năng lượng điện.

Trong bối cảnh công nghiệp hóa và hiện đại hóa, nhu cầu tiêu thụ điện năng ngày càng gia tăng, kéo theo yêu cầu về quản lý và giám sát hiệu quả hệ thống điện. Việc tiêu thụ điện năng không chỉ ảnh hưởng đến hoạt động sản xuất, kinh doanh mà còn tác động trực tiếp đến ổn định kinh tế, an ninh năng lượng và bảo vệ môi trường. Một trong những thách thức lớn hiện nay là làm thế nào để giám sát, phát hiện sớm các dấu hiệu bất thường trong tiêu thụ điện năng, giúp giảm thiểu tổn thất và nâng cao hiệu quả vận hành hệ thống điện.

1.2. Tình hình phát triển hệ thống điện tại Việt Nam.

Việt Nam là một trong những quốc gia có tốc độ phát triển kinh tế nhanh trong khu vực Đông Nam Á, kéo theo nhu cầu tiêu thụ điện năng ngày càng gia tăng. Theo thống kê, tổng sản lượng điện tiêu thụ của Việt Nam tăng trung bình khoảng 8–10% mỗi năm, phản ánh sự mở rộng của các ngành công nghiệp, dịch vụ và đời sống dân cư.

Để đáp ứng nhu cầu này, hệ thống điện Việt Nam đã có những bước phát triển đáng kể với sự mở rộng của các nguồn phát điện, lưới truyền tải và hệ thống phân phối. Hiện nay, hệ thống điện quốc gia bao gồm các nguồn điện từ thủy điện, nhiệt điện than, nhiệt điện khí, điện mặt trời, điện gió và điện nhập khẩu. Chính phủ Việt Nam cũng đang đẩy mạnh chuyển đổi năng lượng xanh nhằm giảm sự phụ thuộc vào nguồn điện hóa thạch và hạn chế tác động tiêu cực đến môi trường.

Tuy nhiên, cùng với sự phát triển nhanh chóng, ngành điện Việt Nam cũng đối mặt với nhiều thách thức như quá tải lưới điện, tổn thất điện năng, thất thoát điện do gian lận, cũng như nhu cầu nâng cao tính tự động hóa và giám sát thông minh hệ thống điện. Trong bối cảnh đó, việc ứng dụng công nghệ hiện đại, đặc biệt là trí tuệ nhân tạo (AI) và hệ thống giám sát tự động, là một xu hướng tất yếu để nâng cao hiệu quả vận hành hệ thống điện.

1.3. Một số phương pháp dự báo

1.3.1. Một số phương pháp dự báo sản lượng điện năng tiêu thụ theo thời gian

a) **Phương pháp thống kê**

Các phương pháp này dựa trên mô hình toán học để phân tích xu hướng tiêu thụ điện trong quá khứ và dự báo tương lai.

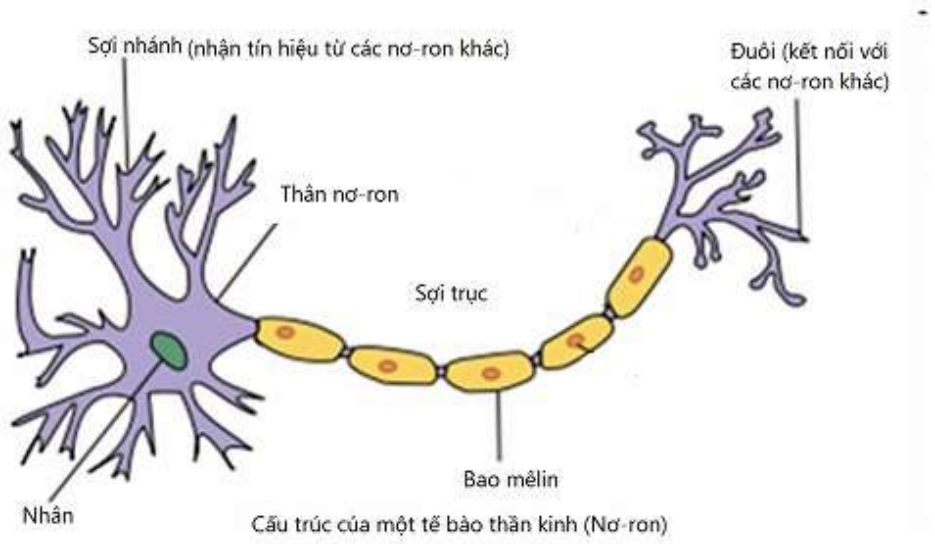
- Phân tích hồi quy tuyến tính: Dự báo sản lượng điện dựa trên các yếu tố đầu vào như nhiệt độ, thời gian trong ngày, ngày trong tuần, v.v.
- Phương pháp trung bình trượt (Moving Average): Dùng giá trị trung bình của một số chu kỳ trước đó để dự báo giá trị tiếp theo.
- Mô hình ARIMA (AutoRegressive Integrated Moving Average): Thích hợp cho dữ liệu có tính chu kỳ và có xu hướng.
- Mô hình SARIMA (Seasonal ARIMA): Là một biến thể của ARIMA, thích hợp cho dữ liệu có yếu tố mùa vụ (tháng, quý, năm).

b) **Phương pháp trí tuệ nhân tạo**

Phương pháp dự báo sản lượng điện năng tiêu thụ theo thời gian bằng trí tuệ nhân tạo thường bao gồm việc sử dụng các kỹ thuật như mạng Nơ-ron nhân tạo, máy học và học sâu để tìm ra mối quan hệ giữa các yếu tố sản lượng điện tiêu thụ, từ đó dự báo sản lượng điện trong tương lai.

❖ Mạng Nơ-ron nhân tạo

Mạng Nơ-ron nhân tạo là một thuật toán được lấy cảm hứng từ cách hoạt động của hệ thần kinh sinh học. Ở mạng Nơ-ron sinh học, mỗi Nơ-ron sẽ có các đầu vào, đầu ra, sợi liên kết vào bộ sử lý tín hiệu. Các đầu vào sẽ nhận được tín hiệu từ đầu ra của các Nơ-ron khác, sau đó truyền tín hiệu đến bộ xử lý nhờ sợi liên kết rồi đưa kết quả tới các đầu ra. Nhiều Nơ-ron liên kết với nhau tạo thành một mạng Nơ-ron.



Hình 1.1: Cấu trúc của một tế bào thần kinh

❖ Học máy

Học máy là một lĩnh vực trong khoa học máy tính và trí tuệ nhân tạo, nghiên cứu về các phương pháp để cho máy tính tự động học hỏi từ dữ liệu mà không cần được lập trình cụ thể cho từng tác vụ. Học máy là một công cụ quan trọng để giải quyết các bài toán phức tạp, đặc biệt là trong việc xử lý dữ liệu lớn.

Các mô hình học máy được chia thành ba loại chính:

- Học máy có giám sát

Học có giám sát được hiểu là sách sử dụng các tập dữ liệu được gắn nhãn để huấn luyện thuật toán phân loại hoặc dự đoán kết quả một cách chính xác.

Học tập có giám sát giúp các tổ chức giải quyết nhiều vấn đề trong thực tế trên quy mô lớn. Một số phương pháp được sử dụng trong học có giám sát bao gồm Mạng Nơ-ron, mô hình phân lớp, hồi quy tuyến tính, hồi quy logistic, Random forest và máy hỗ trợ Vector (SVM – support vector machine).

- Học máy không giám sát

Học không giám sát sử dụng các thuật toán học máy để phân tích và phân cụm các tập dữ liệu không được gắn nhãn. Các thuật toán này phát hiện ra các mẫu hoặc nhóm dữ liệu ẩn mà không cần sự can thiệp của con người.

Các thuật toán được sử dụng trong học tập không giám sát bao gồm Mang Nơ-ron, phân cụm K-means và các phương pháp phân cụm theo xác suất.

- Học máy bán giám sát

Học tập bán giám sát là sự kết hợp hài hòa giữ học tập có giám sát và không giám sát. Trong quá trình đào tạo, nó sử dụng một tập dữ liệu có nhãn nhỏ hơn học có giám sát để hướng dẫn phân loại và trích xuất tính năng từ một tập dữ liệu lớn hơn, không được gắn nhãn.

Học tập bán giám sát có thể giải quyết vấn đề trong trường hợp không có đủ dữ liệu được gắn nhãn cho thuật toán học có giám sát.

❖ Học sâu

Học sâu là một nhánh của học máy, tập trung vào việc xây dựng các mô hình máy học có khả năng học tập và dự đoán dữ liệu. Các mô hình này được xây dựng dựa trên một kiến trúc mạng lưới Nơ-ron nhân tạo, được cấu thành từ nhiều lớp Nơ-ron

Các mô hình học sâu có thể được sử dụng để giải quyết nhiều vấn đề khác nhau trong các lĩnh vực như thị giác máy tính, nhận dạng giọng nói, xử lý ngôn ngữ tự nhiên, Robotics và nhiều ứng dụng khác. Học sâu đã tạo ra những tiến bộ đáng kể trong nhiều lĩnh vực ứng dụng và đang được sử dụng rộng rãi trong các sản phẩm và dịch vụ.

Các kiến trúc mạng Nơ-ron trong học sâu được xây dựng dựa trên các thuật toán lan truyền ngược và phân loại, bao gồm:

- Mạng Nơ-ron tích chập (Convolutional Neural Network – CNN)
- Mạng Nơ-ron hồi quy (Recurrent Neural Network - RNN) & LSTM (Long Short-Term Memory)
- Mạng Nơ-ron tự động mã hóa (Autoencoder Neural Network – AE)
- Mạng Nơ-ron học tăng cường (Reinforcement Learning Neural Network)
- Mạng Nơ-ron Feedforward (Neural Hierarchical Interpolation – N-HiTs)

1.4. Phân loại và mục đích của giám sát tiêu thụ điện

1.4.1 Phân loại

Hệ thống giám sát tiêu thụ điện có thể được phân loại theo nhiều tiêu chí khác nhau:

Theo phạm vi giám sát:

- Giám sát hộ gia đình: Theo dõi mức tiêu thụ điện của từng hộ dân, giúp người dùng quản lý chi phí điện hiệu quả.
- Giám sát khu vực: Đo lường và phân tích mức tiêu thụ điện tại các khu vực nhằm tối ưu hóa vận hành.
- Giám sát lưới điện: Theo dõi hệ thống truyền tải và phân phối điện nhằm phát hiện bất thường, tránh sự cố mất điện trên diện rộng.

Theo phương pháp giám sát:

- Giám sát theo thời gian thực: Sử dụng cảm biến và hệ thống đo xa để thu thập dữ liệu liên tục, cho phép phản ứng nhanh với các sự cố.
- Giám sát theo chu kỳ: Phân tích dữ liệu theo từng khoảng thời gian nhất định (giờ, ngày, tháng) để phát hiện xu hướng và dự báo.
- Giám sát thông minh bằng AI: Ứng dụng trí tuệ nhân tạo để phát hiện bất thường và đưa ra cảnh báo tự động.

1.4.2. Mục đích

Việc giám sát tiêu thụ điện có vai trò quan trọng trong việc đảm bảo hoạt động ổn định và tối ưu hóa hiệu suất của hệ thống điện. Một số mục tiêu chính của giám sát tiêu thụ điện bao gồm:

- Phát hiện và cảnh báo sớm các bất thường trong tiêu thụ điện: Giúp nhận diện nhanh chóng các hành vi bất thường như rò rỉ điện, gian lận điện, hoặc các sự cố kỹ thuật.

- Tối ưu hóa vận hành và giảm tổn thất điện năng: Cho phép ngành điện có các biện pháp điều chỉnh hợp lý để nâng cao hiệu suất sử dụng điện.

- Dự báo nhu cầu tiêu thụ điện: Giúp lập kế hoạch cung cấp điện hợp lý, tránh tình trạng quá tải hệ thống.

- Tăng cường quản lý và tiết kiệm điện: Hỗ trợ doanh nghiệp và hộ gia đình theo dõi, kiểm soát lượng điện tiêu thụ để giảm chi phí và tiết kiệm năng lượng.

- Cải thiện dịch vụ khách hàng: Cung cấp các báo cáo tiêu thụ điện chi tiết, giúp khách hàng hiểu rõ và điều chỉnh hành vi sử dụng điện hợp lý hơn.

1.5. Kết luận chương 1

Chương này đã trình bày tổng quan về tình hình phát triển của hệ thống điện tại Việt Nam, các phương pháp dự báo tiêu thụ điện và hệ thống giám sát hiện tại. Qua đó cho thấy rằng, các phương pháp truyền thống vẫn còn tồn tại nhiều hạn chế trong việc phát hiện bất thường và tối ưu hóa quản lý điện năng. Việc ứng dụng công nghệ Deep Learning là cần thiết nhằm cải thiện độ chính xác và hiệu quả giám sát, tạo nền tảng cho các giải pháp sẽ được trình bày trong các chương tiếp theo.

CHƯƠNG 2: HỆ THỐNG THU THẬP DỮ LIỆU ĐO ĐẾM TỪ XA

2.1. Hệ thống thu thập số liệu công tơ đầu nguồn từ xa DSPM

Nguyên lý hoạt động:

- Mỗi trạm biến áp 110 kV lắp đặt máy tính công nghiệp (RTU) hoạt động 24/24 có kết nối với cổng giao tiếp RS485, cổng giao tiếp này kết nối với các RS232 hoặc RS485 của công tơ để thu thập số liệu. Số liệu file thu thập được từ công tơ (file thô) được lưu trữ vào RTU, đồng thời thông qua mạng Internet để truyền về Server đặt tại Công ty.

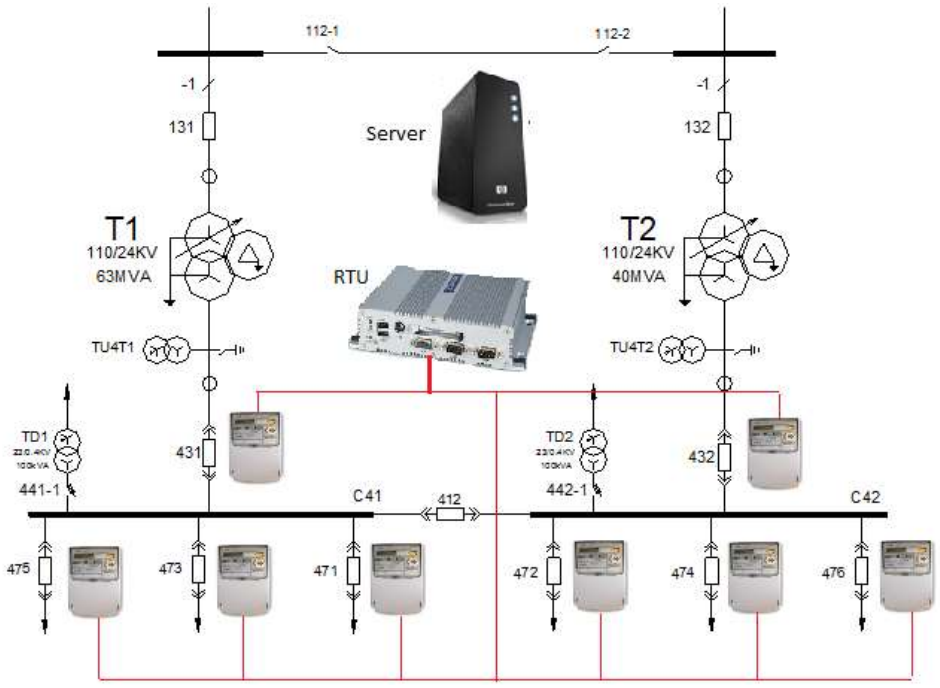
- File thu thập được từ công tơ (file thô) được giải mã, biên dịch để đẩy vào cơ sở dữ liệu thông qua phần mềm HES DSPM.

- Thông qua phần mềm DSPM do CPCiT viết được cài đặt trên các máy tính sẽ cho phép hiển thị các thông số của công tơ, hiển thị biểu đồ phụ tải, xem danh sách các trạm mất tín hiệu, tổng hợp phụ tải các Điện lực, phân quyền truy cập.

- Cài đặt phần mềm DSPM: http://113.160.224.170:8000/mdms/publish.htm

- Địa chỉ máy chủ hiện nay : 10.123.96.3; data : MDMS.

- Tần suất thu thập hằng ngày : hàng giờ, 24 lần/ ngày.



Hình 2.1: Nguyên lý hoạt động hệ thống thu thập đầu nguồn từ xa DSPM

- Đối với các TBA 220kV/ 500 kV thuộc truyền tải điện quản lý, dữ liệu đo xa được đồng bộ về Tổng Công ty Điện lực Miền Trung, sau đó đồng bộ về PC Đà Nẵng.

2.2. Hệ thống đo xa GPRS/3G

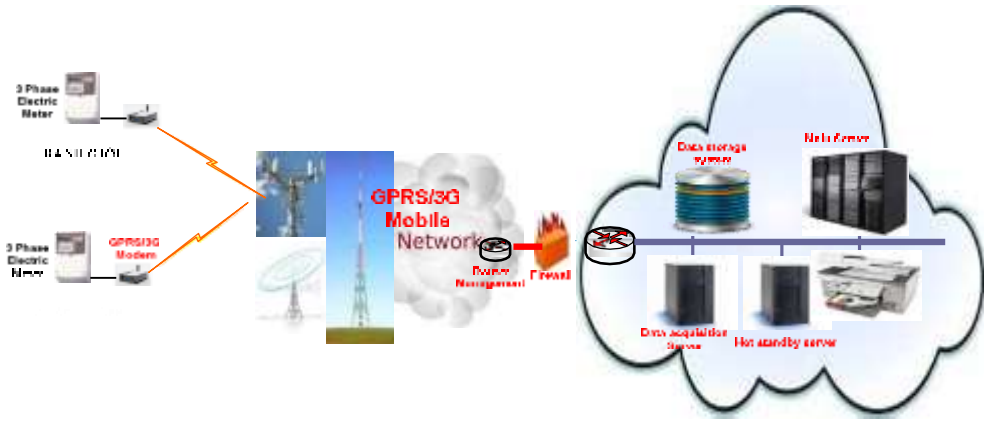
- Thời gian triển khai: từ năm 2013, PC Đà Nẵng hợp đồng với Công ty Cổ phần Tư vấn Đầu tư Phát triển Hạ tầng Viễn thông (IFC) cung cấp dịch vụ đo xa.

- Đối tượng : các TBA công cộng, chuyên dùng, các khách hàng có sản lượng từ 5.000 kWh/tháng trở lên. Đây là đối tượng được lắp công tơ điện tử 3 pha nhiều biểu giá đa chức năng.

- Nguyên lý : tại mỗi điểm đo, công tơ điện tử được gắn modem lắp sim di động, số liệu công tơ được thu thập qua đường truyền GPRS/3G, tần suất thu thập 30 phút/ lần (48 lần/ ngày).

- Số lượng hiện nay : 5.114 điểm đo. Tỷ lệ online trên 99%.

- Các thông số được thu thập : chỉ số (từng thời điểm, chốt tháng); thông số vận hành 3 pha (công suất, dòng điện, điện áp, tần số, cosj); biểu đồ phụ tải 30 phút; sự kiện (mất điện, mất pha, …).



Hình 2.2: Mô hình hệ thống đo xa GPRS/3G

Hệ thống đo xa GPRS/3G gồm có 02 thành phần cơ bản sau:

Tại các điểm ranh giới hoặc khách hàng:

- Công tơ điện tử đã được tích hợp sẵn cổng giao tiếp RS232/485

- Modem GPRS: để truyền dữ liệu về Trung tâm OMC.

Trung tâm Thu thập và Xử lý Số liệu (OMC):

- Servers + Hệ điều hành OS.

- Phần mềm quản lý kết nối và đọc dữ liệu công tơ.

- Phần mềm xử lý dữ liệu: chuyển dữ liệu dạng HEXA sang dạng ASCII.

- Phần mềm báo cáo số liệu.

2.3. Hệ thống đo xa công nghệ RF-SPIDER

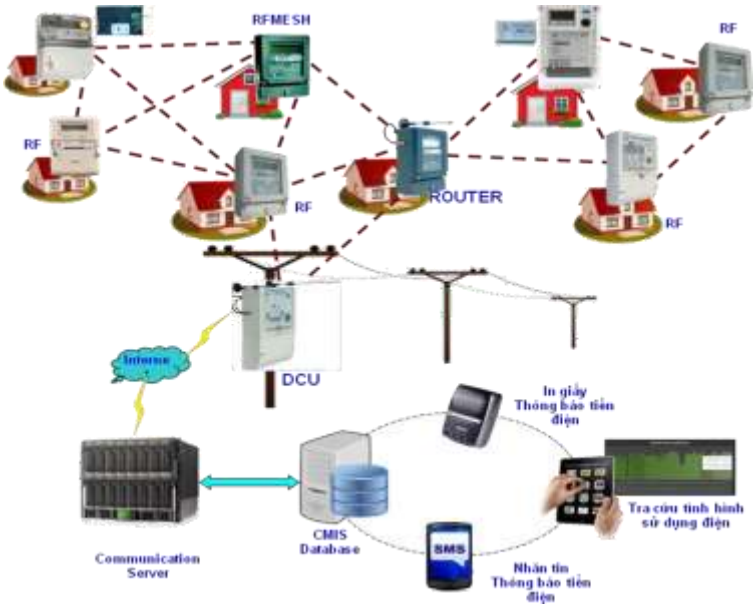
- Thời gian triển khai: Từ tháng 6/2013, PC Đà Nẵng đã lựa chọn và đưa vào sử dụng công nghệ đọc chỉ số công tơ điện tử từ xa bằng sóng RF qua thiết bị máy tính cầm tay Handheld Unit (HHU); đến 2016, chuyển sang công nghệ tự động RF-Spider do Trung tâm sản xuất thiết bị đo điện tử Điện lực miền Trung tư vấn cung cấp thiết bị (công tơ RF, công tơ Mesh, DCU, Router).

- Đối tượng: được lắp đặt tại các công tơ điện tử RF của khách hàng sau trạm biến áp công cộng. Công nghệ này được áp dụng đối với khu vực dân cư, nơi mà tập trung phần lớn khách hàng sử dụng điện mục đích sinh hoạt.

- Nguyên lý: Tại mỗi trạm biến áp công cộng, sẽ có 1 bộ tập trung DCU, có các bộ khuếch đại sóng RF Router, hoặc công tơ có chức năng Mesh làm nhiệm vụ thu thập các công tơ RF xung quanh theo mô hình mắt lưới (mạng nhện) một cách thông minh, tự động theo thời gian được lập trình sẵn. Ưu điểm của RF-Mesh là sử dụng sóng RF miễn phí, nhu cầu chỉ cần thu thập một vài thông số cơ bản như chỉ số, điện áp, dòng điện để tính hóa đơn cho khách hàng sinh hoạt, kiểm tra phát hiện trộm cắp điện. Toàn bộ dữ liệu thu thập được tại lưu trữ bộ tập trung DCU, sẽ được gửi về server qua sóng di động nhờ có gắn sim card. Lịch đọc chỉ số được thực hiện tự động hằng ngày trên các phần mềm trên máy tính. Dữ liệu chỉ số thu thập được xuất ra file xml, sau đó file xml sẽ được cập nhật vào chương trình CMIS để tính hóa đơn cho khách hàng. Tần suất thu thập hiện nay là 6 giờ/ lần (4 lần/ ngày).

- Số lượng hiện nay: đo xa Spider là 355.091 điểm đo với 2106 DCU, 13.614 Router, 62.800 công tơ Mesh. Tỷ lệ online hiện nay trên 99%.

- Các thông số được thu thập: chỉ số; điện áp, dòng điện (chỉ thu thập được đối với chủng loại DT01P80-RF.



Hình 2.3: Mô hình hệ thống đo xa RF-Spider

2.3.1. DCU (Data Collection Unit)

DCU là thiết bị chính của hệ thống, được tích hợp sẵn modem GPRS/3G. DCU thu thập chỉ số từ xa cho tất cả các công tơ có trạng bị RF trong phạm vi khai báo, gửi dữ liệu thu thập về Server trung tâm thông qua đường truyền dữ liệu GPRS/3G

Hình 2.4: DCU

2.3.2. Router

Là thiết bị định tuyến nhằm quản lý, thu thập dữ liệu của một số lượng công tơ nhất định, dữ liệu được gửi về DCU khi có yêu cầu. Router còn có chức năng chuyển tiếp dữ liệu từ gói tin từ DCU đến công tơ tích hợp RF-Mesh. Nhờ đó mà DCU chỉ lấy số liệu từ Router mà không cần liên lạc trực tiếp đến các công tơ trong vùng phủ sóng

của Router. Router liên lạc với các RF-Mesh thông qua sóng vô tuyến, liên lạc với các công tơ qua sóng vô tuyến. Chuyển dữ liệu đến DCU qua mạng di động GPRS/3G.

Hình 2.5: Router

2.3.3. Công tơ

Hình 2.6: Công tơ

a) **Công tơ RF-Mesh**

Là sản phẩm đo đếm điện năng có tích hợp khối thu phát sóng vô tuyến (RF), bên cạnh những chức năng đo đếm của Công tơ điện tử thông thường, công tơ RF-Mesh có khả năng tự tìm đường nhờ giải thuật định tuyến thông minh, tự thiết lập nên mạng lwosi RF-Mesh, chuyển thông tin của tất cả các công tơ trong phạm vi phủ sóng của nó về đến vộ thu thập tập trung DCU.

b) **Công tơ RF**

Các công tơ RF khác do CPC EMEC sản xuất đều có khả năng hoạt động tương thích với hệ thống RF-Mesh triển khai cơ CPC

Các loại công tơ của các nhà sản xuất khác như Elster, Landis+Gyr... có thể kết nối với hệ thống RF-Spider nếu được gắn thêm module RF-Elster, RF-Multimeter của CPC EMEC.

c) Server

Mỗi công ty điện lực sẽ được cung cấp 1 Server để vận hành chương trình thu thập chỉ số theo nguyên tắc:

- DCU kết nối với Server qua giao thức TCP/IP

- Server lấy thông tin của khác hàng từ Application Server để thực hiện khám phá, thu thập các chỉ số và thông số vận hành của công tơ khách hàng.

- Server thực hiện đọc chỉ số vận hành chuyển về Application Server để phân tích dữ liệu và lưu trữ vào Database Server.

Các ứng dụng triển khai trên Application Server bao gồm: Web RF-Spider, cung cấp webserver cho hệ thống khai thác số liệu.

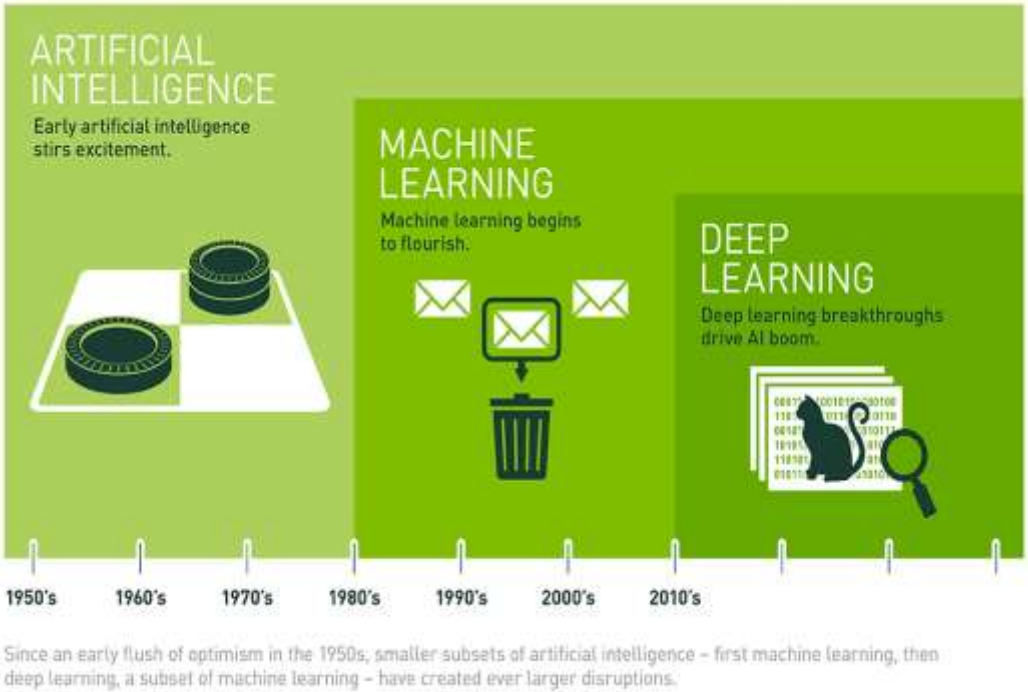
2.4. Kết luận chương 2

Trong chương này, nhóm đã trình bày tổng quan chi tiết về ba hệ thống thu thập dữ liệu đo đếm từ xa đang được triển khai tại PC Đà Nẵng, bao gồm: hệ thống DSPM, hệ thống đo xa GPRS/3G và công nghệ RF-Spider. Mỗi hệ thống có đặc điểm kỹ thuật và đối tượng áp dụng riêng, góp phần nâng cao tính tự động hóa và độ chính xác trong việc thu thập dữ liệu công tơ điện. Tuy nhiên, điểm chung của các hệ thống hiện tại là chỉ tập trung vào việc thu thập và lưu trữ dữ liệu, chưa tích hợp các chức năng giám sát thông minh hoặc phát hiện bất thường tự động. Điều này gây hạn chế trong việc phát hiện sớm các vấn đề như thất thoát điện, trộm cắp điện hay tiêu thụ điện không bình thường của khách hàng. Chính vì vậy, trong chương tiếp theo, nhóm sẽ đề xuất giải pháp xây dựng hệ thống giám sát tiêu thụ điện thông minh dựa trên công nghệ học sâu (Deep Learning). Các mô hình sẽ được xây dựng để dự báo sản lượng tiêu thụ điện và phát hiện bất thường trong dữ liệu, góp phần nâng cao hiệu quả giám sát và vận hành hệ thống điện.

CHƯƠNG 3: CƠ SỞ LÝ THUYẾT VÀ GIẢI PHÁP DỰ BÁO SẢN LƯỢNG ĐIỆN NĂNG TIÊU THỤ VÀ XÁC ĐỊNH BẤT THƯỜNG

3.1. Giới thiệu phương pháp

Hiện nay có rất nhiều phương pháp để thực hiện dự báo sản lượng điện tiêu thụ như đã trình bày ở chương 1. Mỗi phương pháp có đặc điểm và ứng dụng khác nhau, tùy thuộc vào mục đích sử dụng và tính chất của dữ liệu. Trong đồ án tốt nghiệp này, chúng em sẽ sử dụng phương pháp trí tuệ nhân tạo để thực hiện dự báo sản lượng điện năng tiêu thụ và xác định bất thường.



Hình 3.1: Các giai đoạn phát triển của trí tuệ nhân tạo

3.2. Mạng Nơ-ron phi hồi quy và hồi quy.

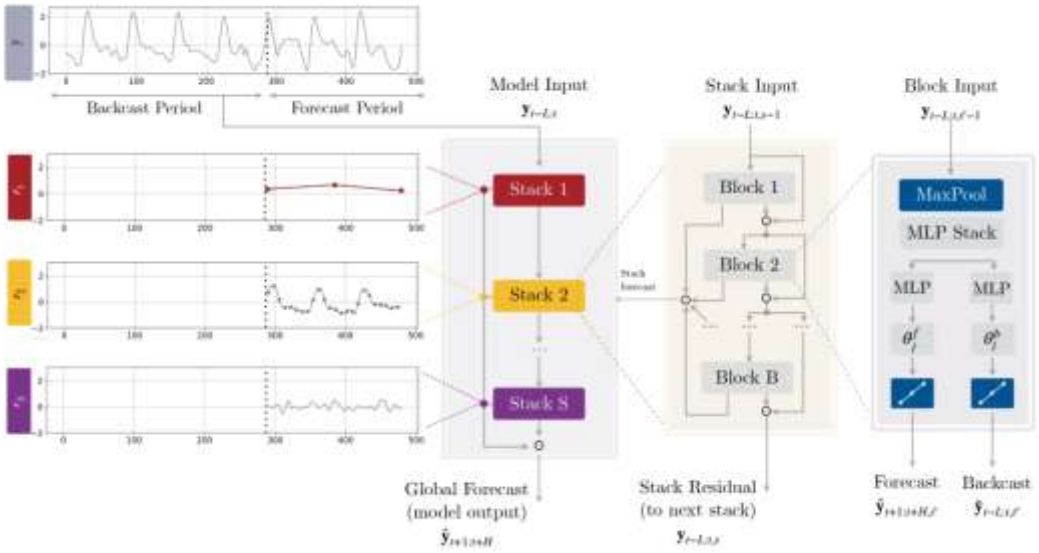
3.2.1. Mạng Nơ-ron phi hồi quy Neural Hierarchical Interpolation – N-HiTs

Mô hình N-HiTs (Neural Hierarchical Interpolation for Time Series Forecasting) được giới thiệu lần đầu vào năm 2023. Ban đầu, N-HiTs được phát triển nhằm cải thiện hiệu quả dự báo chuỗi thời gian với kiến trúc phân cấp sử dụng cơ chế nội suy phi tuyến.

Trong ngành kỹ thuật điện (KTD), N-HiTs ngày càng được ứng dụng phổ biến cho các bài toán như dự báo tải điện, phát hiện bất thường trong tiêu thụ năng lượng, và đặc

biệt là dự báo chỉ số công tơ điện thông minh. Nhờ khả năng xử lý tốt dữ liệu chuỗi đa biến với cấu trúc phức tạp và tính phi tuyến, N-HiTs giúp nâng cao độ chính xác trong dự báo cũng như khả năng nhận diện sớm các hành vi tiêu thụ bất thường.

- Kiến trúc mô hình:



Hình 3.2: Mô hình N-HiTs

Kiến trúc của mô hình N-HiTs bao gồm nhiều khối Stack khác nhau, trong mỗi khối Stack bao gồm nhiều khối nội suy phân cấp (hierarchical interpolation blocks), mỗi khối tập trung vào quy mô thời gian khác nhau. Mô hình N-HiTs sử dụng một cấu trúc phân cấp, trong đó mỗi khối thập trung vào một quy mô thời gian khác nhau. Điều này cho phép mô hình nắm bắt các mẫu thời gian ở nhiều quy mô, từ các biến động ngắn hạn đến các xu hướng dài hạn.

- Nguyên lý hoạt động

N-HiTs thực hiện các phép chiếu phi tuyến cục bộ lên hàm cơ sở thông qua nhiều khối, mỗi khối bao gồm một multilayer perceptron (MLP) học cách tạo ra hệ số cho đầu ra backcast và forecast. Các khối được nhóm thành các Stack, mỗi Stack học một đặc điểm khác nhau của dữ liệu. Đầu vào của toàn bộ mạng là chuỗi tín hiệu $y_{t-L:t}$ từ L độ trễ.

Lấy mẫu tín hiệu đa tỷ lệ: Tại đầu vào của mỗi khối ℓ, một lớp MaxPool với kích thước kernel k_ℓ được sử dụng để phân tích các thành phần tín hiệu ở một tỷ lệ cụ thể. Kích thước k_ℓ lớn hơn cắt bớt các thành phần tần số cao, giúp khối tập trung vào phân tích các nội dụng quy mô lớn hơn. Điều này giúp giảm số lượng tham số cần học, giảm thiểu overfitting và duy trì trường tiếp cận ban đầu.

$$y^{(p)}_{t-L:t,\ell} = Maxpool(y_{t-L:t,\ell}, k_\ell)$$

Hồi quy phi tuyến: Sau khi lấy mẫu, mỗi khối thực hiện hồi quy phi tuyến để tạo ra các hệ số MLP tiến và lùi. Các hệ số này được sử dụng để tổng hợp đầu ra backcast và forecast của khối. Quá trình này giúp tạo ra dự báo chính xác hơn bằng cách sử dụng các phép tính phi tuyến

$$h_\ell = MLP_\ell(y^{(p)}_{t-L:t,\ell})$$
$$\theta^f_\ell = \text{LINEAR}^f(h_\ell)$$
$$\theta^b_\ell = \text{LINEAR}^b(h_\ell)$$

Nội suy phân cấp (Hierarchical Interpolation): N-HiTs sử dụng nội suy thời gian. Các hệ số nội suy được xác định theo tỷ lệ biểu đạt (expressiveness ratio) r_ℓ điều chỉnh số lượng tham số trên mỗi đơn vị thời gian đầu ra, $\theta^f_\ell = [r_\ell H]$. Để khôi phục lại tốc độ lấy mẫu ban đầu và dự báo tất cả H điểm trong tầm nhìn, chúng ta sử dụng nội suy thời gian thông qua hàm hội suy g :

$$y_{\tau,\ell} = g(\tau, \theta^f_\ell), \forall \tau \in \{t+1, ..., t+H\},$$
$$y_{\tau,\ell} = g(\tau, \theta^b_\ell), \forall \tau \in \{t-L, ..., t\}.$$

3.2.2. Mạng Nơ-ron hồi quy.

Mạng Nơ-ron hồi quy (Recurrent Neural Network – RNN) là một loại mạng Nơ-ron nhân tạo được thiết kế để xử lý và phân tích các chuỗi dữ liệu.

Dựa trên số lượng xử lý của chuỗi đầu vào và chuỗi đầu ra, người ta chia mạng RNN thành 4 loại chính:

Trong mạng RNN, trạng thái ẩn tại mỗi bước thời gian sẽ được tính toán dựa vào dữ liệu đầu vào tại bước thời gian tương ứng và các thông tin có được từ bước thời gian trước đó, tạo khả năng ghi nhớ các thông tin đã được tính toán ở những bước thời gian trước cho mạng.

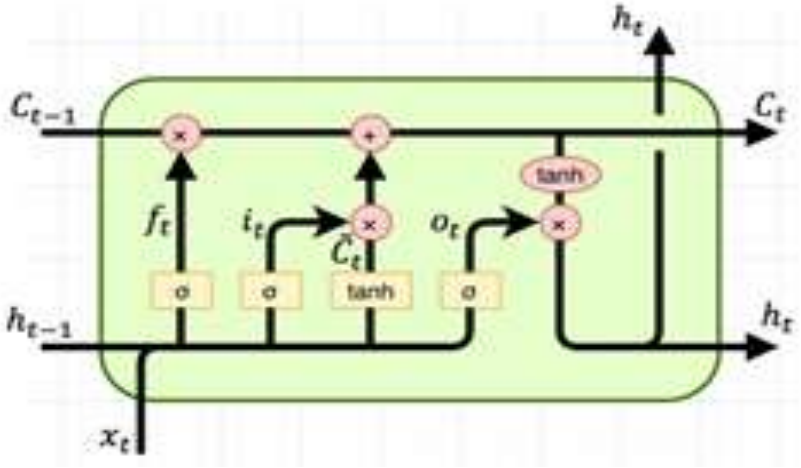
a) **Mô hình LSTM (Long Short-Term Memory)**

Cơ chế hoạt động của mạng LSTM:

LSTM là một phiên bản mở rộng của mạng RNN, được đề xuất vào năm 1997 bởi Sepp Hochreiter và Jurgen Schmidhuber. LSTM được thiết kế để giải quyết các bài toán về phụ thuộc xa (Long-term dependencies) trong mạng RNN do bị ảnh hưởng bởi vấn đề Gradient biến mất.

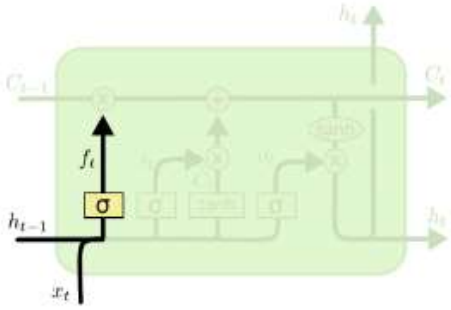
Khi các mạng RNN hoạt động, thông tin trước đó được ghi nhớ và sử dụng lại để xử lý cho đầu vào hiện tại. Tuy nhiên thì mạng RNN không thể ghi nhớ thông tin ở các bước có khoảng cách khá xa trước đó do vấn đề về Gradient biến mất. Do đó những

phần tử đầu tiên trong chuỗi đầu vào không có nhiều ảnh hưởng đến các kết quả tính toán dự đoán phần tử cho chuỗi đầu ra trong các bước sau. Mạng LSTM với các kết nối phản hồi (feedback connection) giúp khắc phục nhược điểm này.



Hình 3.3: Một cell của mô hình LSTM

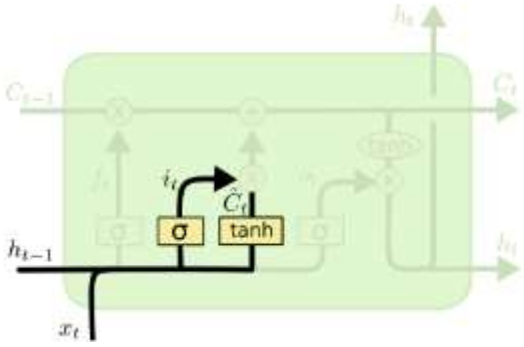
Bước đầu tiên trong LSTM sẽ quyết định xem thông tin nào chúng ta sẽ cho phép đi qua ô trạng thái (cell state). Nó được kiểm soát bởi hàm sigmoid trong một tầng gọi là tầng quên (forget gate layer). Đầu tiên nó nhận đầu vào là 2 giá trị h_{t-1} và x_t và trả về một giá trị nằm trong khoảng 0 và 1 cho mỗi giá trị của ô trạng thái c_{t-1}. Nếu giá trị bằng 1 thể hiện "giữ toàn bộ thông tin" và bằng 0 thể hiện "bỏ qua toàn bộ chúng".



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Hình 3.4: Tầng cổng quên

Bước tiếp theo chúng ta sẽ quyết định loại thông tin nào sẽ được lưu trữ trong ô trạng thái. Bước này bao gồm 2 phần. Phần đầu tiên là một tầng ẩn của hàm sigmoid được gọi là tầng cổng vào (input gate layer) quyết định giá trị bao nhiêu sẽ được cập nhật. Tiếp theo, tầng ẩn hàm tanh sẽ tạo ra một véc tơ của một giá trị trạng thái mới τ_{t-1} mà có thể được thêm vào trạng thái. Tiếp theo kết hợp kết quả của hai tầng này để tạo thành một cập nhật cho trạng thái.

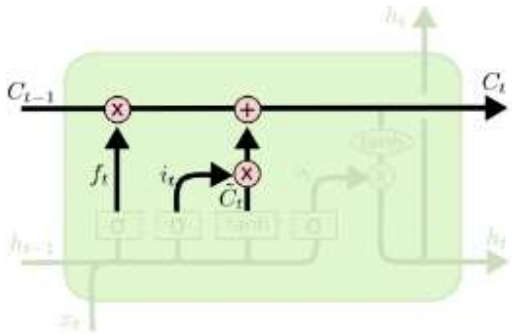


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Hình 3.5: Cập nhật giá trị cho ô trạng thái

Đây là thời điểm để cập nhật một ô trạng thái cũ, c_{t-1} sang một trạng thái mới c_t. Những bước trước đó đã quyết định làm cái gì, và tại đây bước này chỉ cần thực hiện nó.

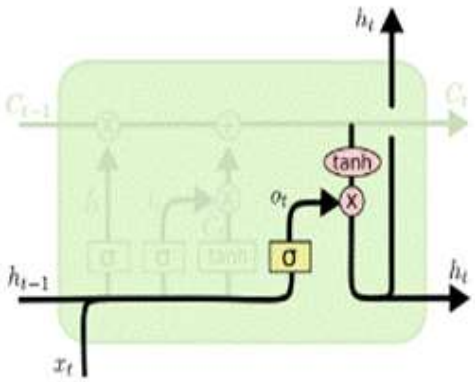
Chúng ta nhân trạng thái cũ với f_t tương ứng với việc quên những thứ quyết định được phép quên sớm. Phần tử đề cử $i_t * \tau_t$ là một giá trị mới được tính toán tương ứng với bao nhiêu được cập nhật vào mỗi giá trị trạng thái.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Hình 3.6: Ô trạng thái mới

Cuối cùng cần quyết định xem đầu ra sẽ trả về bao nhiêu. Kết quả ở đầu ra sẽ dựa trên ô trạng thái, những sẽ là một phiên bản được lọc. Đầu tiên, chúng ta chạy qua một tầngS sigmoid nơi quyết định phần nào của ô trạng thái sẽ ở đầu ra. Sau đó, ô trạng thái được đưa qua hàm tanh (để chuyển giá trị về khoảng -1 và 1) và nhân nó với đầu ra của một cổng sigmoid, do đó chỉ trả ra phần mà chúng ta quyết định.



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Hình 3.7: Điều chỉnh thông tin ở đầu ra thông qua hàm Tanh

Nhờ các cơ chế các cổng, LSTM có khả năng xử lý các chuỗi dữ liệu dài và phức tạp, giúp cải thiện hiệu suất của mạng Nơ-ron hồi quy trong nhiều ứng dụng.

b) **Mô hình Bi-LSTM (Bidirectional Long Short-Term Memory**

Cơ chế hoạt động và cấu trúc mô hình Bi-LSTM:

Bi-LSTM (Bidirectional LSTM) là phần mở rộng của mạng LSTM truyền thống. Thay vì chỉ xử lý chuỗi theo một chiều thời gian (từ quá khứ đến hiện tại), Bi-LSTM sử dụng hai LSTM song song:

- Một LSTM theo chiều thuận (forward),
- Một LSTM theo chiều ngược (backward).

Kết quả đầu ra tại mỗi bước thời gian là sự kết hợp thông tin từ cả hai hướng, nhờ đó mạng có thể hiểu rõ hơn ngữ cảnh đầy đủ – không chỉ từ quá khứ mà cả từ tương lai (trong phạm vi chuỗi đầu vào). Bi-LSTM đặc biệt hiệu quả trong bài toán mà ngữ cảnh ở cả trước và sau đều quan trọng, như: dự báo chuỗi có tính chu kỳ như dữ liệu điện năng tiêu thụ.

3.3. Giải pháp đề xuất và thực thi bài toán dự báo sản lượng điện năng tiêu thụ

3.3.1. Giới thiệu công cụ thực hiện

Trong quá trình thực hiện đề tài, nhóm sử dụng ngôn ngữ lập trình Python và môi trường Google Colaboratory để xây dựng, huấn luyện và triển khai các mô hình học sâu. Một số thư viện và công cụ chính được sử dụng bao gồm:

Python: Ngôn ngữ chính để phát triển toàn bộ pipeline xử lý dữ liệu, xây dựng mô hình, đánh giá và triển khai kết quả.

Google Colab: Môi trường Jupyter Notebook trực tuyến hỗ trợ GPU miễn phí, giúp thuận tiện trong việc huấn luyện mô hình.

TensorFlow & Keras: Dùng để xây dựng và huấn luyện các mô hình mạng nơ-ron sâu như LSTM, Bi-LSTM.

NumPy & Pandas: Hỗ trợ thao tác với dữ liệu, xử lý chuỗi thời gian và thống kê cơ bản.

NeuralForecast: Thư viện chuyên dụng cho dự báo chuỗi thời gian bằng các mô hình học sâu như N-HiTs, N-BEATS, DeepAR, v.v.

3.3.2. Qui trình thực hiện dự báo sản lượng điện năng tiêu thụ

Hình 3.8: Dữ liệu thu thập

Quy trình thực hiện trong Hình 3.8 mô tả các bước chính trong hệ thống dự báo chỉ số công tơ điện và phát hiện bất thường. Đầu tiên, dữ liệu được thu thập, tiền xử lý và chia thành các tập dữ liệu phù hợp, sau đó được chuẩn hóa để đảm bảo tính đồng nhất. Tiếp theo, ba mô hình dự báo khác nhau gồm N-HiTs, LSTM và Bi-LSTM được áp dụng song song. Sau khi lựa chọn mô hình phù hợp, quá trình huấn luyện được thực hiện và mô hình được đánh giá bằng tập dữ liệu kiểm tra. Kết quả dự báo được sử dụng để xác định các điểm bất thường, từ đó tính toán sai số và đánh giá hiệu suất của mô

hình. Cuối cùng, các kết quả được so sánh và tổng hợp để đưa ra kết luận về chất lượng dự báo và khả năng phát hiện bất thường

3.3.3. Thu thập dữ liệu.

Dữ liệu được thu thập từ các công tơ điện tử thông qua các giao thức truyền thông được gửi về và lưu trữ trong cơ sở dữ liệu của trung tâm điều khiển công ty Điện lực Đà Nẵng. Dữ liệu thu thập bao gồm 271121 mẫu 7 tính năng từ 26/12/2024 đến 22/4/2025 của 559 đơn vị khách hàng trong khu vực với tần suất lấy mẫu 4 lần 1 ngày.

	MA_DVIQLY	MA_TRAM	NGAYGIO	MA_DIEMDO	SO_CTO	IMPORT_KWH	EXPORT_KWH
0	PP0900	FC53FFRU	26/12/2024 00:16	PP09000904667001	2002350862	99964.2920	0.0000
1	PP0900	FC53FFRU	26/12/2024 00:24	PP09000819619001	17333956	21626.2200	0.0000
2	PP0900	FC53FFRU	26/12/2024 00:25	PP09000832892001	2200281931	7441.7000	0.0000
3	PP0900	FC53FFRU	26/12/2024 00:25	PP09000832824001	2200160241	20711.5400	0.0000
4	PP0900	FC53FFRU	26/12/2024 00:25	PP09000833025001	2200161889	6640.3000	0.0000
...	...	...	...	...	...	...	...
271117	PP0900	FC53FFRU	22/04/2025 12:00	PP09000927611001	16041420	114573.3132	6.8836
271118	PP0900	FC53FFRU	22/04/2025 12:00	PP09000927611001	16041420	114573.3132	6.8836
271119	PP0900	FC53FFRU	22/04/2025 23:55	PP09000927611001	16041420	114580.3476	6.8836
271120	PP0900	FC53FFRU	22/04/2025 23:55	PP09000927611001	16041420	114580.3476	6.8836
271121	PP0900	FC53FFRU	22/04/2025 23:55	PP09000927611001	16041420	114580.3476	6.8836

```
Số lượng mã điểm đo khác nhau: 559
```

Hình 3.9: Dữ liệu thu thập

Trong đó:

MA_DVIQLY: Mã đơn vị quản lí công tơ

MA_TRAM: Mã trạm

NGAYGIO: Thời gian ghi dữ liệu từ công tơ

MA_DIEMDO: Mã điểm đo

SO_CTO: Số công tơ đại diện chỗ mỗi khách hàng

IMPORT_KWH: Sản lượng điện tiêu thụ

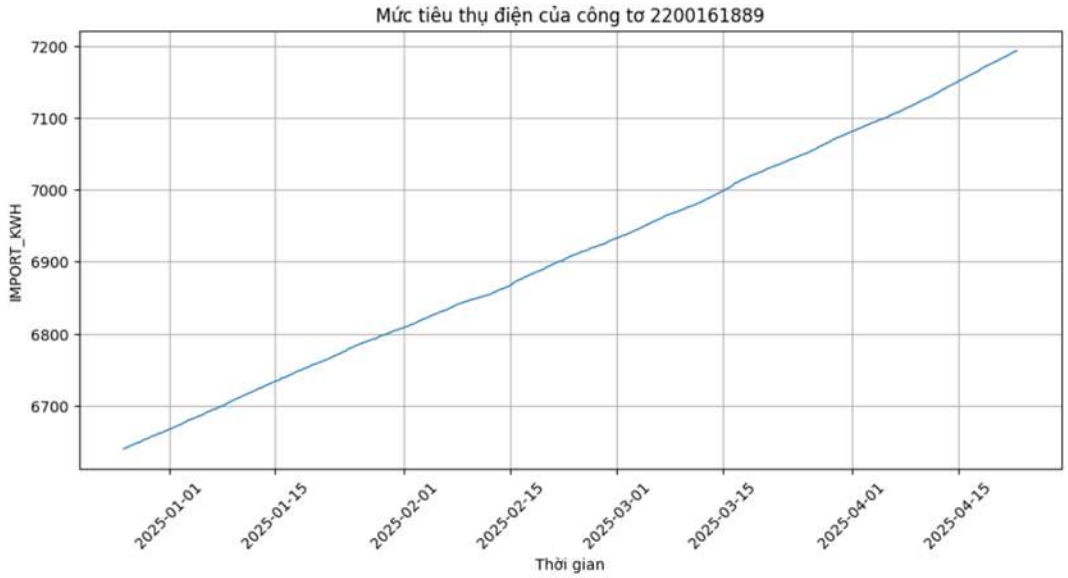
EXPORT_KWH: Sản lượng điện phát ra

3.3.4. Tiền xử lý dữ liệu.

Tiền xử lí dữ liệu (preprocessing) là quá trình xử lý và chuẩn bị dữ liệu trước khi áp dụng các mô hình phân tích. Tiền xử lí dữ liệu đóng vai trò quan trọng trong việc

đảm bảo chính xác và hiệu quả của các mô hình, đồng thời cũng giúp cho các nhà phân tích dữ liệu có thể hiểu và phân tích dữ liệu một cách dễ dàng hơn.

Từ bảng dữ liệu thu thập được, tính năng mà chúng ta quan tâm nhất là IMPORT_KWH (sản lượng điện tiêu thụ được đồng hồ điện tử ghi lại). Sau đây là 1 số đồ thị về mức tiêu thụ điện của 1 số điểm đo:



Hình 3.10: Mức tiêu thụ điện

Vì dữ liệu được thu thập từ đồng hồ điện tử vậy nên dữ liệu mang tính tích lũy, vậy nên cần tạo ra 1 tính năng mới 'value delta' về lượng tiêu thụ điện qua khoảng thời gian lấy mẫu.

Các bước tiền xử lí được áp dụng cho bài toán này như sau:

a) Lọc ra các dữ liệu có cùng MA_DIEMDO, SO_CTO, MA_TRAM, MA_DVIQLY

b) Xử lí các giá trị bị thiếu

c) Sử dụng các kĩ thuật như Differencing, Lagged value để tạo các tính năng mới phù hợp với mục đích của bài toán

d) Loại bỏ các giá trị outlier

Như ta thấy từ dữ liệu thu thập, mặc dù tần suất ghi dữ liệu trung bình là 4 lần/ngày, nhưng các mốc thời gian lấy mẫu không đều và thậm chí bị lặp lại trong cùng một khoảng thời gian. Điều này gây khó khăn cho các mô hình học sâu (Deep Learning), vốn học hiệu quả hơn trên dữ liệu có tính chu kỳ và phân bố đều theo thời gian.

Ví dụ, có những công tơ ghi dữ liệu tại 00h10, nhưng lại tiếp tục ghi thêm tại 00h20 trong cùng một giờ, dẫn đến các giá trị bị chồng chéo.

Vì vậy, để chuẩn hóa dữ liệu thời gian, ta tiến hành làm tròn các mốc thời gian theo quy tắc: nếu số phút nhỏ hơn 30 thì làm tròn xuống đầu giờ, nếu lớn hơn hoặc bằng 30 thì làm tròn lên đầu giờ tiếp theo và chỉ lưu giá trị dữ liệu đo ở thời gian đầu tiên trong cùng một giờ.Nhờ đó, dữ liệu sẽ có cấu trúc ổn định và đều đặn hơn về mặt thời gian, giúp mô hình học tốt hơn.

Vì IMPORT_KWH chỉ là giá trị tích lũy, nó không phản ánh mức độ tiêu thụ tại từng thời điểm. Việc tính tv delta giúp ta chuẩn hóa dữ liệu theo thời gian, tạo đầu vào phù hợp hơn cho các mô hình phân tích, dự báo hoặc phát hiện bất thường.

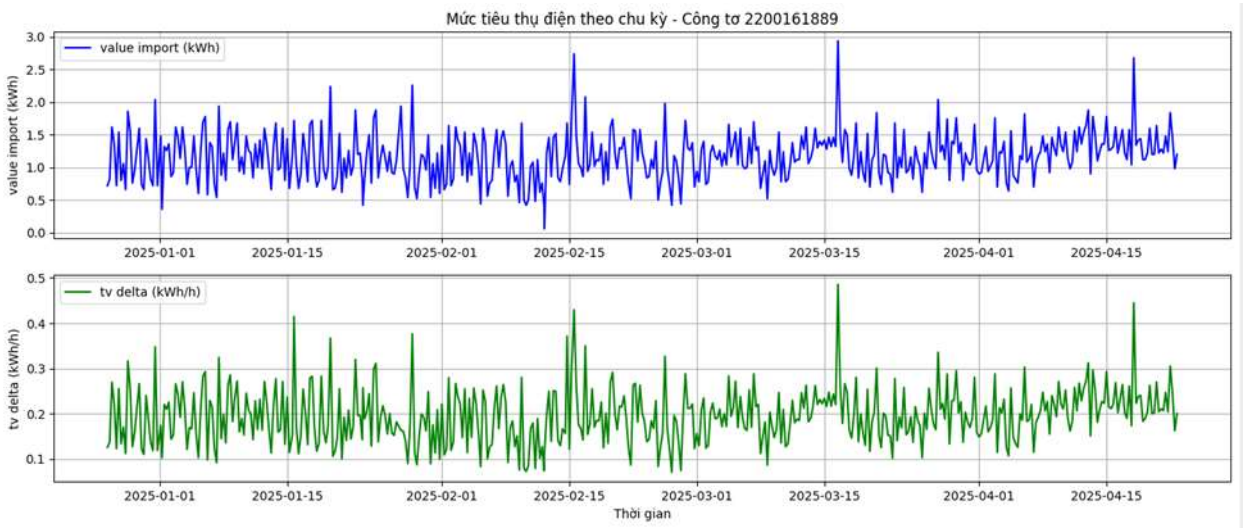
Đặc biệt hữu ích khi khoảng cách giữa các lần ghi dữ liệu không cố định (không đều đặn).

Nên cần tạo ra 1 tính năng mới 'tv delta' bằng cách thực hiện kĩ thuật differencing, lấy giá trị 'value delta' giữa 2 giá trị đo liên tiếp chia cho chênh lệch thời gian lấy mẫu. Ở đây 'tv delta' có nghĩa là tốc độ tiêu thụ điện năng theo thời gian của khách hàng sử dụng điện.

	MA_DVIQLY	MA_TRAM	NGAYGIO	MA_DIEMDO	SO_CTO	IMPORT_KWH	EXPORT_KWH	Time Delta	value import	tv delta
0	PP0900	FC53FFRU	2024-12-26 02:00:00	PP09000912959001	13102373	28627.81	0.0	NaN	NaN	NaN
1	PP0900	FC53FFRU	2024-12-26 08:00:00	PP09000912959001	13102373	28629.01	0.0	6.0	1.20	0.200000
2	PP0900	FC53FFRU	2024-12-26 13:00:00	PP09000912959001	13102373	28629.86	0.0	5.0	0.85	0.170000
3	PP0900	FC53FFRU	2024-12-26 20:00:00	PP09000912959001	13102373	28631.61	0.0	7.0	1.75	0.250000
4	PP0900	FC53FFRU	2024-12-27 02:00:00	PP09000912959001	13102373	28634.13	0.0	6.0	2.52	0.420000
...	...	...	...	...	...	...	...	...	...	...
264101	PP0900	FC53FFRU	2025-04-21 18:00:00	PP09000836981001	2500017751	74.04	0.0	6.0	1.70	0.283333
264102	PP0900	FC53FFRU	2025-04-22 00:00:00	PP09000836981001	2500017751	77.24	0.0	6.0	3.20	0.533333
264103	PP0900	FC53FFRU	2025-04-22 06:00:00	PP09000836981001	2500017751	79.06	0.0	6.0	1.82	0.303333
264104	PP0900	FC53FFRU	2025-04-22 12:00:00	PP09000836981001	2500017751	80.30	0.0	6.0	1.24	0.206667
264105	PP0900	FC53FFRU	2025-04-22 18:00:00	PP09000836981001	2500017751	82.74	0.0	6.0	2.44	0.406667

264106 rows × 10 columns

Hình 3.11: Dữ liệu sau khi xử lý thời gian và tính tv delta



Hình 3.12: Mức tiêu thụ điện của công tơ 2200161889

3.3.5. Nội suy dữ liệu

Nội suy tuyến tính theo thời gian là phương pháp ước lượng giá trị bị thiếu dựa trên thời điểm xảy ra của các giá trị đã biết.

Công thức tính nội suy như sau:

Giả sử có 2 điểm đã biết:

- t_1, y_1 là thời gian và giá trị trước giá trị cần dự báo

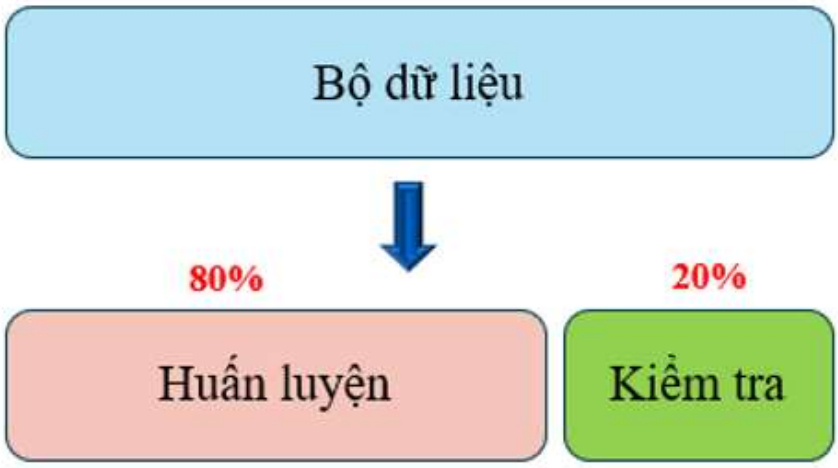
- t_2, y_2 là thời gian và giá trị sau giá trị cần dự báo

- t, y là thời gian và giá trị cần dự báo

$$y = y_1 + (y_2 - y_1).\frac{t - t_1}{t_2 - t_1}$$

3.3.6. Phân chia dữ liệu

Trước khi xây dựng mô hình dự báo, chúng ta cần phải phân chia tập dữ liệu thành tập huấn luyện và kiểm tra. Phân chia tập dữ liệu giúp chúng ta đánh giá hiệu quả của mô hình trên các tập dữ liệu khác nhau và tránh tình trạng Overfitting. Overfitting là hiện tượng mô hình huấn luyện quá tốt với những dữ liệu huấn luyện nhưng không tốt với dữ liệu mới. Khi một mô hình quá tập trung vào dữ liệu huấn luyện, nó có thể bị mất khả năng tổng quát hóa, không thể áp dụng vào các dữ liệu mới và không nhận diện được các mẫu mới mà nó chưa từng thấy trước đó.



Hình 3.13: Chia bộ dữ liệu

Chúng ta sẽ chia bộ dữ liệu theo tỉ lệ 8:2, 80% dữ liệu dùng để huấn luyện mô hình, 20% dữ liệu còn lại để kiểm tra mô hình.

3.3.7. Loại bỏ outlier và nội suy lại dữ liệu trên tập Train

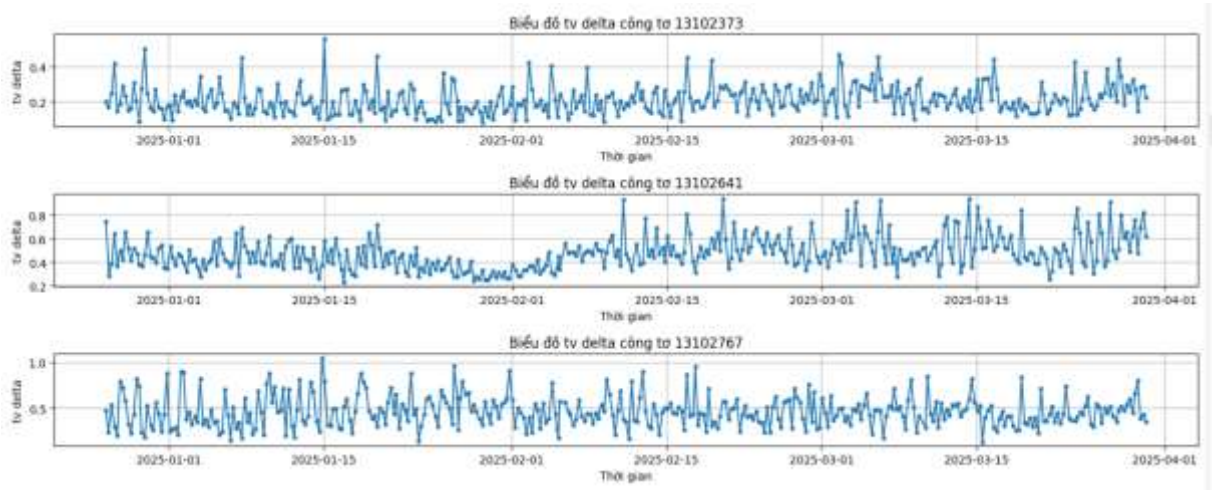
Việc loại bỏ outlier là rất cần thiết trong quá trị xử lý để đưa dữ liệu vào dự báo. Tiến hành loại bỏ outlier để dữ liệu trở nên mềm hơn và có thể dễ học hơn. Z-score là độ lệch giữa giá trị xem xét đến đối với giá trị trung bình theo đơn vị độ lệch chuẩn (standard deviation).

$$z = \frac{x - \mu}{\sigma}$$

- x: là giá trị hiện tại
- μ: trung bình
- σ: độ lệch chuẩn (standard deviation)

Nếu độ lớn của z lớn hơn threshold thì giá trị đó được xem là oulier

Mục đích của việc nội suy lại các giá trị oulier bị loại bỏ trong tập train để dữ liệu có thể phụ hợp với chuẩn của mô hình N-HiTs.



Hình 3.14: Biểu đồ tv delta của một số công tơ

3.3.8. Chuẩn hóa dữ liệu

Trong quá trình xử lý và phân tích dữ liệu, việc chuẩn hóa dữ liệu là một bước quan trọng nhằm đảm bảo rằng các đặc trưng và biến số trong tập dữ liệu đều nằm trong cùng một phạm vi hoặc phân phối. Điều này giúp tạo điều kiện thuận lợi cho các phân tích và mô hình dự báo hiệu quả. Trong báo cáo này, ta sẽ sử dụng phương pháp Min-Max Scaling.

Phương pháp Min-Max Scaling (còn được gọi là Normalization) là một phương pháp chuẩn hóa dữ liệu phổ biến và đơn giản trong xử lý dữ liệu. Phương pháp này chuyển đổi các giá trị của biến số thành khoảng giá trị mới, thường là khoảng [0;1]. Quá trình chuẩn hóa dữ liệu theo phương pháp này được thực hiện theo công thức sau:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

Trong đó:

- x là giá trị gốc của đặc trưng cần chuẩn hóa.
- $x_{\min}$ là giá trị nhỏ nhất của đặc trưng đó trong tập dữ liệu.
- $x_{\max}$ là giá trị lớn nhất của đặc trưng đó trong tập dữ liệu.
- x' là giá trị sau khi đã được chuẩn hóa về khoảng 0, 1.

3.3.9. Xây dựng mô hình

a) Yêu cầu bài toán

Bài toán đặt ra: Dùng một mô hình tổng quát để dự báo mức tiêu thụ điện và phát hiện điểm bất thường trong tiêu thụ của 559 hộ gia đình trên địa bàn thành phố Đà Nẵng. Để thực hiện điều đó, nhóm đã quyết định dùng 3 mô hình là N-HiTs, LSTM, Bi-LSTM để xây dựng và dự đoán, từ đó chọn ra mô hình tốt nhất để thực hiện xác định bất thường.

b) Mô hình N-HiTs

- **Xác định độ dài chuỗi đầu vào (input size):** Mô hình sử dụng input_size = 56, tức lấy 56 bước thời gian trước (tương đương 14 ngày nếu lấy mẫu mỗi giờ) để dự báo chuỗi tiếp theo
- **Xác định độ dài chuỗi dự báo đầu ra (forecast horizon):** h = 28, tương ứng với việc dự báo sản lượng tiêu thụ điện trong 7 ngày kế tiếp.
- **Cấu trúc các tầng downsampling:** Mô hình sử dụng 3 tầng với các hệ số kernel [4, 2, 1] tương ứng với mức giảm tần số dần qua từng tầng, giúp trích xuất đặc trưng ở nhiều thang thời gian khác nhau
- **Xác định hàm kích hoạt:** Mô hình sử dụng hàm kích hoạt ReLU để tăng khả năng học phi tuyến trong quá trình huấn luyện.
- **Sử dụng dropout:** Áp dụng dropout tại tầng dự đoán với xác suất dropout_prob_theta = 0.5 nhằm tránh overfitting trong quá trình huấn luyện.
- **Tham số huấn luyện:** Mô hình được huấn luyện với learning_rate = 0.001, batch_size = 64 và max_steps = 1500. Sử dụng hàm tối ưu Adam, tích hợp sẵn trong thư viện NeuralForecast
- **Tần suất dữ liệu:** Dữ liệu đầu vào được xử lý theo chu kỳ 6 giờ/lần (freq = '6H'), phù hợp với tần suất lấy mẫu dữ liệu thực tế.

c) Mô hình LSTM

Trong kiến trúc LSTM cổ điển, mô hình chưa thể khai thác đầy đủ đặc trưng phân biệt giữa các hộ gia đình, do dữ liệu đầu vào là chuỗi thời gian dài gồm nhiều hộ khác nhau, được phân biệt bằng mã số công tơ (SO_CTO). Điều này khiến mô hình khó học được hành vi tiêu thụ điện đặc thù của từng hộ.

Để khắc phục hạn chế này, chúng em đề xuất một kiến trúc mô hình mới bằng cách bổ sung thêm một lớp Embedding cho SO_CTO – ID đại diện cho từng hộ gia đình. Lớp embedding này sẽ học biểu diễn đặc trưng riêng biệt cho mỗi hộ dưới dạng vector không gian liên tục. Sau đó, embedding vector sẽ được ghép nối với chuỗi dữ liệu đầu vào tại mỗi bước thời gian, trước khi đưa vào mạng LSTM.

Cách tiếp cận này giúp mô hình không chỉ học được mối quan hệ theo thời gian trong dữ liệu tiêu thụ điện, mà còn đồng thời học được đặc trưng riêng biệt của từng hộ gia đình, từ đó cải thiện hiệu quả dự báo trong bài toán chuỗi thời gian đa đối tượng.

Mô hình LSTM được xây dựng với kiến trúc như sau:

- Xác định số chiều của lớp Embedding.
- Xác định số lượng lớp LSTM trong mô hình.
- Xác định số lượng đơn vị LSTM trong mỗi lớp LSTM.
- Xác định hàm kích hoạt cho mỗi lớp LSTM.
- Xác định các lớp đầy đủ kết nối (fully connected layer) để thực hiện dự báo.

Cấu trúc mô hình được xây dựng cụ thể như sau:

Layer (type)	Output Shape	Param #	Connected to
household_id (InputLayer)	(None, 1)	0	-
embedding (Embedding)	(None, 1, 16)	8,192	household_id[0][…
flatten (Flatten)	(None, 16)	0	embedding[0][0]
input_seq (InputLayer)	(None, 56, 11)	0	-
repeat_vector (RepeatVector)	(None, 56, 16)	0	flatten[0][0]
concatenate (Concatenate)	(None, 56, 27)	0	input_seq[0][0], repeat_vector[0]…
lstm (LSTM)	(None, 128)	79,872	concatenate[0][0]
dense (Dense)	(None, 28)	3,612	lstm[0][0]

Hình 3.15: Kiến trúc mô hình LSTM đã xây dựng

Cấu trúc mô hình áp dụng cho bài toán này sẽ được ghép thông tin hộ tiêu thụ (household_id) được mã hóa bằng lớp embedding, sau đó mới đưa vào mô hình Bi-LSTM. Điều này cho phép mô hình học được các đặc trưng ẩn riêng biệt của từng hộ. Mô hình LSTM không chỉ khai thác được thông tin phụ thuộc dài hạn trong chuỗi thời gian, mà còn tận dụng hiệu quả đặc trưng phân loại (household_id) để cá nhân hóa dự báo cho từng hộ tiêu thụ điện, từ đó nâng cao độ chính xác và khả năng tổng quát hóa của mô hình.

d) Mô hình Bi-LSTM

Tương tự như cấu trúc mô hình LSTM, kiến trúc của Bi-LSTM được xây dựng như sau:

Layer (type)	Output Shape	Param #	Connected to
household_id (InputLayer)	(None, 1)	0	-
embedding (Embedding)	(None, 1, 16)	8,912	household_id[0][…
flatten (Flatten)	(None, 16)	0	embedding[0][0]
input_seq (InputLayer)	(None, 28, 10)	0	-
repeat_vector (RepeatVector)	(None, 28, 16)	0	flatten[0][0]
concatenate (Concatenate)	(None, 28, 26)	0	input_seq[0][0], repeat_vector[0]…
bidirectional (Bidirectional)	(None, 128)	46,592	concatenate[0][0]
dense (Dense)	(None, 28)	3,612	bidirectional[0]…

Hình 3.16: Kiến trúc mô hình Bi-LSTM đã xây dựng

3.3.10. Đánh giá mô hình

Sau khi mô hình được huấn luyện xong, chúng ta sẽ tiến hành đánh giá mô hình dựa trên tập dữ liệu kiểm tra và tập dữ liệu đã dự báo được. Kết quả các sai số giữa công suất thực tế và công suất dự báo sẽ phản ánh mô hình đã xây dựng có tốt hay không.

Để đánh giá mô hình, ta sử dụng độ đo sai số RMSE (Root Mean Squared Error) MAPE (Mean Absolute Percentage Error) và MAE (Mean Absolute Error). Trong đó, RMSE là phương phát đánh giá mô hình Deep Learning phổ biến nhất, MAPE là độ đo sai số đặc biệt được sử dụng nhiều trong bài toán dự báo còn MAE đặc biệt phù hợp với các mô hình Deep Learning do tính đơn giản và khả năng phản ánh sai số trung bình tuyệt đối giữa giá trị dự đoán và thực tế.. RMSE , MAPE và MAE có công thức tính như sau:

$$RMSE = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(y_j - y_j)^2}$$

$$MAPE = \frac{100\%}{N}\sum_{j=1}^{N}\left|\frac{y_j - y_j}{y_j}\right|$$

$$MAE = \frac{1}{N}\sum_{j=1}^{N}\left|y_j - y_j\right|$$

Trong đó:

- y_j : là giá trị công suất thực tế
- y_j : là giá trị công suất dự đoán
- N : là số lượng mẫu kiểm tra

3.3.11. Xác định điểm bất thường

Phát hiện điểm bất thường (anomaly detection) bằng cách dùng ngưỡng (threshold) là một phương pháp thống kê phổ biến, dựa trên giả định rằng hầu hết các điểm dữ liệu đều là "bình thường" và các điểm "bất thường" sẽ khác biệt đáng kể so với phần còn lại.

Ý tưởng: Giả sử bạn có một tập dữ liệu các giá trị đo lường, hoặc sai số giữa giá trị thực tế và giá trị dự báo $error = |y_{true} - y_{pred}|$. Nếu một giá trị sai số vượt quá một mức ngưỡng nhất định (threshold), ta coi đó là bất thường.

Cách xác định ngưỡng dựa trên thống kê chuẩn (z-score): Với phân phối Gaussian, có thể tính threshold bằng:

$$threshold = \mu + k \cdot \sigma$$

Trong đó

- μ :là giá trị trung bình của sai số.
- σ :là độ lệch chuẩn của sai số.
- k :là một hằng số ($k = 2$)

Trường hợp phát hiện bất thường: Điểm xác định là bất thường nếu $|y_{true} - y_{pred}| > threshold$

3.4. Kết luận chương 3

Trong chương này đã giới thiệu công cụ dùng để lập trình chương trình dự báo mức tiêu thụ điện và xác định bất thường trong tiêu thụ. Sau khi xây dựng lưu đồ thuật

toán, ta sẽ lần lượt thực hiến xác bước tuân theo lưu đồ. Có 2 bước quan trọng để có kết quả chính xác đó là "Tiền xử lý dữ liệu" và "Xây dựng mô hình", ta phải xử lý dữ liệu chính xác để đưa vào mô hình, đồng thời thay đổi thông số trong mô hình để tìm ra kết quả tối ưu nhất của mô hình đã xây dựng. Để đánh giá hiệu suất của mô hình, ta sử dụng các hàm sai số RMSE (Root Mean Squared Error) và MAPE (Mean Absolute Percentage Error). Kết quả sau khi huấn luyện sẽ được trình bày ở Chương 4.

CHƯƠNG 4: KẾT QUẢ THỰC NGHIỆM VÀ ĐÁNH GIÁ MÔ HÌNH

4.1. Kết quả dự báo.

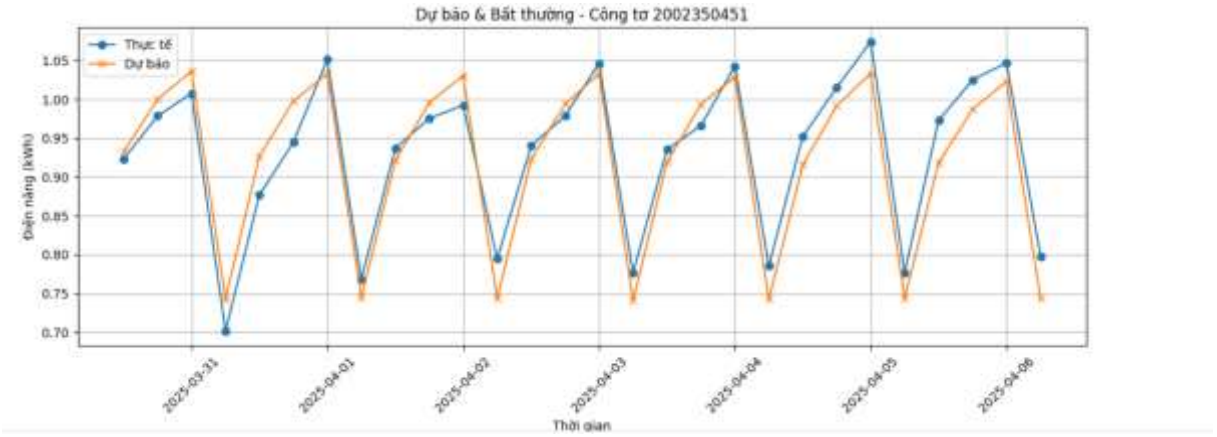
Với mục đích dự báo chỉ số điện và phát hiện bất thường trong tiêu thụ điện, việc sử dụng các mô hình khác nhau để dự báo và sử dụng giá trị MAPE và RMSE để đánh giá hiệu quả của các mô hình. Trong quá trình huấn luyện, mỗi loại mô hình mạng Nơ-ron sẽ được huấn luyện nhiều lần với cấu trúc mạng khác nhau để so sánh và tìm ra cấu trúc có kết quả tốt nhất

4.1.1. Mạng Nơ-ron phi hồi quy Neural Hierarchical Interpolation N-HiTs

Cấu trúc mô hình dự báo	Batch size	RMSE (kWh)	MAE (kWh)	MAPE (%)
N-HiTs	32	0.152	0.094	25.43
N-HiTs	64	0.154	0.095	25.6

Bảng 4.1: Kết quả sai số RMSE, MAE và MAPE của các cấu trúc mạng N-HiTs

Nhận xét: Mặc dù mô hình đạt được kết quả dự báo tương đối tốt đối với phần lớn các hộ gia đình như Hình 4.1 là kết quả dự báo Công tơ 2002350451 nhưng sai số tổng thể vẫn còn khá cao. Nguyên nhân chính là do mô hình được huấn luyện chung cho toàn bộ tập dữ liệu, bao gồm nhiều hộ gia đình với đặc điểm tiêu thụ điện khác nhau. Trong khi phần lớn các hộ có mức tiêu thụ ổn định và theo quy luật rõ ràng, một số hộ lại có hành vi tiêu thụ bất thường, biến động mạnh hoặc không theo mẫu hình cụ thể. Điều này khiến mô hình khó học được đặc trưng phù hợp cho tất cả các trường hợp, từ đó dẫn đến sai số tăng cao đối với các trường hợp ngoại lệ.



Hình 4.1: Kết quả dự báo Công tơ 2002350451 của model N-HiTs

4.1.2. Mạng Nơ-ron hồi quy bộ nhớ dài-ngắn Long Short-Term Memory (LSTM)

Kết quả dự báo của mạng LSTM, có cấu trúc mạng bao gồm 1 lớp LSTM, với số lượng neuron trong các lớp được chọn là 64,128. Dựa vào Bảng 4.2, chúng ta thấy rằng mạng LSTM với 1 lớp LSTM, trong đó mỗi lớp có 128 nơ-ron cho kết quả tốt nhất, với RMSE,MAE và MAPE trung bình tương ứng là 0.1802, 0.1154 và 40.5% trên tập dữ liệu đánh giá.

Cấu trúc mô hình dự báo	**Batch size**	**RMSE (kWh)**	**MAE (kWh)**	**MAPE (%)**
(Embedding,Input) - LSTM64 – Dense28	64	0.1770	0.1148	41.72
(Embedding,Input) - LSTM128 – Dense28	64	0.1802	0.1154	40.50
(Embedding,Input) - LSTM64 – Dense28	128	0.1820	0.1143	41.41

Bảng 4.2: Kết quả sai số RMSE, MAE và MAPE của các cấu trúc mạng LSTM

Nhận xét:

Cấu trúc LSTM64 với batch size 64 cho kết quả RMSE thấp nhất (0.1770 kWh), cho thấy mô hình này có khả năng ước lượng gần đúng hơn so với thực tế, tức ít sai lệch hơn.

Tuy nhiên, mô hình LSTM128 với batch size 64 cho kết quả MAPE thấp nhất (40.50%), thể hiện mô hình này có khả năng tổng quát hóa tốt hơn theo tỉ lệ phần trăm sai lệch.

Batch size 128 không cho thấy cải thiện rõ rệt, thậm chí có xu hướng giảm nhẹ độ chính xác về RMSE (cao hơn 0.005–0.006) → điều này có thể do batch size lớn dẫn đến mô hình hội tụ nhanh nhưng thiếu ổn định cục bộ trong dữ liệu chuỗi thời gian.

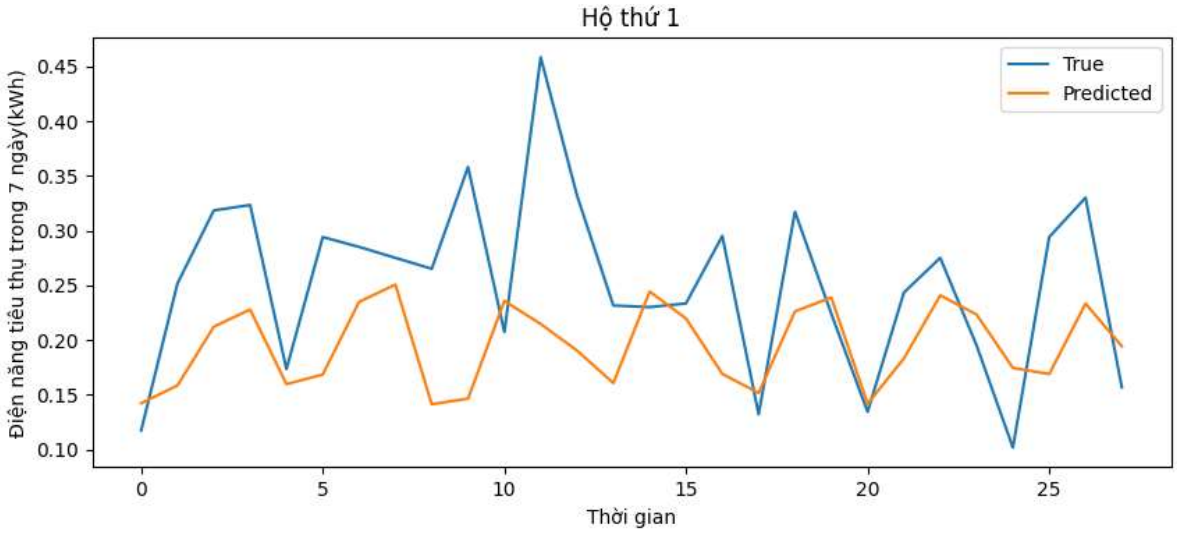
Nguyên nhân khiến kết quả chưa tốt:

Dữ liệu đầu vào biến động mạnh theo thời gian, đặc biệt với dữ liệu điện năng sinh hoạt thường bị ảnh hưởng bởi yếu tố ngoại sinh (thời tiết, hành vi người dùng, ngày lễ...).

Số lượng dữ liệu của từng hộ khi đưa vào mô hình còn ít và số bước dự báo 28 bước đầu ra khá nhiều.

Số lượng dữ liệu lịch sử sử dụng hoặc số giờ dự báo có thể chưa được tối ưu.

Ngoài ra, chỉ sử dụng đơn biến cũng là một yếu tố khiến mô hình khó học được bối cảnh đầy đủ.



Hình 4.2: Kết quả dự báo của mô hình LSTM

4.1.3. Mạng Nơ-ron hồi quy hai chiều bộ nhớ dài-ngắn (Bi-LSTM)

Kết quả dự báo của mạng LSTM, có cấu trúc mạng bao gồm 1 lớp Bi-LSTM, với số lượng neuron trong các lớp được chọn là 64,128. Dựa vào Bảng 4.3, chúng ta thấy rằng mạng Bi-LSTM với 1 lớp Bi-LSTM, trong đó mỗi lớp có 128 nơ-ron cho kết quả tốt nhất, với RMSE, MAE và MAPE trung bình tương ứng là 0.177, 0.1125 và 38.45% trên tập dữ liệu đánh giá.

Cấu trúc mô hình dự báo	**Batch size**	**RMSE (kWh)**	**MAE (kWh)**	**MAPE (%)**
(Embedding,Input)-BiLSTM64– Dense28	64	0.175	0.1091	39.02
(Embedding,Input)-BiLSTM128– Dense28	64	0.177	0.1125	38.45
(Embedding,Input)-BiLSTM64– Dense28	128	0.177	0.1144	41.63
(Embedding,Input)-BiLSTM128– Dense28	128	0.174	0.1097	38.47

Bảng 4.3: Kết quả sai số RMSE, MAE và MAPE của các cấu trúc mạng Bi-LSTM

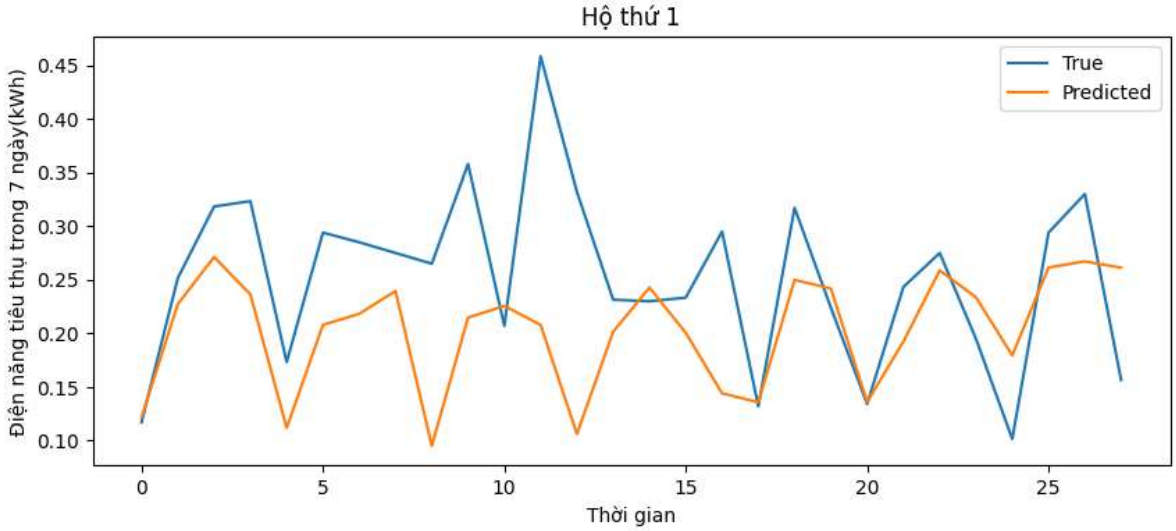
Nhận xét:

Cấu trúc Bi-LSTM128 với batch size 64 đạt kết quả MAPE thấp nhất (38.45%), tức là mô hình này có khả năng ước lượng tỷ lệ sai lệch so với thực tế tốt nhất trong số các cấu hình.

Cấu trúc Bi-LSTM64 với batch size 64 cho RMSE và MAE thấp nhất (0.175 và 0.1091 kWh), thể hiện độ sai lệch tuyệt đối nhỏ hơn → dự đoán gần đúng hơn về giá trị thực tế từng điểm.

Tuy nhiên, Bi-LSTM128 với batch size 128 vẫn đạt RMSE thấp nhất tuyệt đối (0.174) và MAPE chỉ cao hơn rất nhỏ (38.47%), chứng tỏ cấu hình này cũng rất tiềm năng.

Sai số vẫn còn cao do nhiều nguyên nhân tượng tự như mô hình LSTM được giải thích phía trên.



Hình 4.3: Kết quả dự báo của mô hình Bi-LSTM

4.1.4. Nhận xét, đánh giá mô hình dự báo.

Từ kết quả đồ thị của 3 mô hình ở trên, ta thấy mô hình N-HiTs cho ra đồ thị dự báo khá bám so với giá trị thực, tuy nhiên, đồ thị ở 2 mô hình LSTM và Bi-LSTM lại thể hiện sai lệch rất lớn. Dưới đây là bảng so sánh các giá trị sai số của cả 3 mô hình:

Cấu trúc mô hình dự báo	**RMSE (kWh)**	**MAE (kWh)**	**MAPE (%)**
N-HiTs	0.152	0.094	25.43

(Embedding,Input)-BiLSTM128– Dense28	0.177	0.1125	38.45
(Embedding,Input) - LSTM128 – Dense28	0.1802	0.1154	40.50

Bảng 4.4: So sánh RMSE, MAE và MAPE của các cấu trúc mạng

4.1.5. Nhận xét, đánh giá bất thường

Từ Bảng 4.4 ta chọn mô hình N-HiTs để xác định điểm bất thường trong tiêu thụ điện. Bằng phương pháp xác đinh bất thường ở trên ta thu được một số kết quả dự báo bất thường của một công tơ như sau:

Hình 4.4: Dự báo và phát hiện bất thường

Nhận xét: bằng phương pháp xác định điểm bất thường, cho ra đồ thị ở trên ta nhận thấy rằng phương pháp xác định điểm bất thường cho ra kết quả đúng. Số điểm bất thường dựa vào đường ngưỡng mà ta quy định.

4.2. Kết luận chương 4

Trong chương này, nhóm đã trình bày kết quả dự báo của các mô hình học sâu như N-HiTs, LSTM và Bi-LSTM. Dựa vào các chỉ số đánh giá MAE, RMSE và MAPE, mô hình N-HiTs cho kết quả chính xác nhất trong ba mô hình để triển khai thực tế. Hệ thống đã được tích hợp lên giao diện web, giúp hỗ trợ giám sát dữ liệu tiêu thụ điện và phát hiện bất thường hiệu quả. Những nội dung triển khai giao diện được trình bày ở chương tiếp theo.

CHƯƠNG 5: XÂY DỰNG GIAO DIỆN WEB

Để xây dựng giao diện Web trực quan và thân thiện với người dùng, nhóm đã lựa chọn sử dụng Visual Studio Code làm môi trường phát triển chính, kết hợp với ngôn ngữ lập trình Python cùng thư viện Streamlit – một công cụ mạnh mẽ và đơn giản giúp tạo ứng dụng web tương tác. Giao diện Web được thiết kế nhằm hỗ trợ trực quan hóa dữ liệu, cho phép người dùng dễ dàng lựa chọn công tơ điện, thời gian dự báo, điều chỉnh tham số phát hiện bất thường và theo dõi kết quả dự báo thông qua các biểu đồ động. Nhờ vào sự linh hoạt của Streamlit và khả năng mở rộng cao của Python, hệ thống không chỉ thân thiện với người dùng mà còn dễ dàng tích hợp các mô hình học sâu, phục vụ cho bài toán dự báo và giám sát sản lượng điện năng một cách hiệu quả.

5.1. Mục tiêu của giao diện Web.

- Trực quan hóa dữ liệu tiêu thụ điện từ các hộ gia đình.
- Hiển thị kết quả dự báo tiêu thụ điện từ mô hình Deep Learning.
- Giám sát, phát hiện bất thường tiêu thụ điện và cảnh báo người dùng

5.2. Yêu cầu chức năng chính.

Yêu cầu chức năng:

- Tải và hiển thị dữ liệu tiêu thụ điện từ tệp/lưu trữ.
- Chọn meter (ID khách hàng) để xem chi tiết.
- Hiển thị biểu đồ: tiêu thụ thực tế, dự báo, sai lệch.
- Phát hiện bất thường và cảnh báo bằng màu sắc.

Yêu cầu phi chức năng:

- Tính bảo mật
- Giao diện dễ dùng, đơn giản, chạy trên trình duyệt.
- Thời gian phản hồi nhanh, hỗ trợ hiển thị biểu đồ thời gian thực.

5.3. Công nghệ sử dụng.

- Ngôn ngữ lập trình: Python
- Thư viện giao diện: Streamlit
- Xử lý dữ liệu: Pandas, Numpy

- Biểu đồ: Matplotlib, Plotly, Streamlit Chart
- Mô hình học sâu: sử dụng mô hình N-HiTs

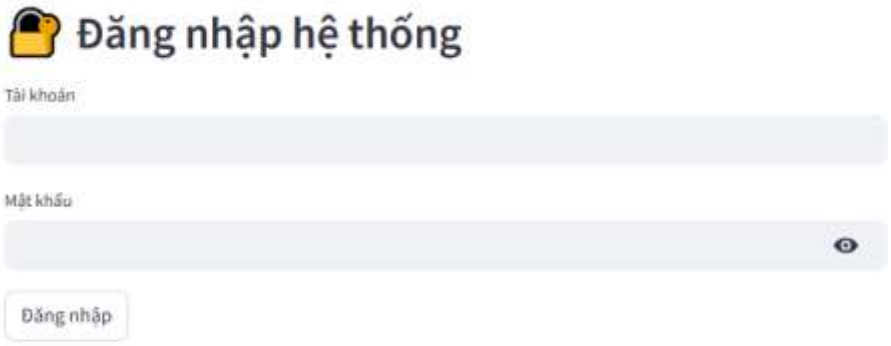
5.4. Thiết kế giao diện tổng thể.

- Sơ đồ bố cục:

- Sidebar: chọn meter, khoảng thời gian, tùy chọn hiển thị.
- Main view: biểu đồ, bảng dữ liệu, kết quả dự báo.
- Alert zone: thông báo bất thường.
- Giao diện chọn meter
- Giao diện hiển thị biểu đồ dự báo
- Giao diện hiển thị cảnh báo bất thường

5.5. Các chức năng chính.

Trang đăng nhập: Trang đăng nhập cho phép người dùng đã đăng ký tài khoản truy cập vào hệ thống bằng cách nhập thông tin xác thực như tên đăng nhập và mật khẩu.



Hình 5.1: Trang đăng nhập

Trang Dashboard tổng quan: Hiển thị tổng mức tiêu thụ của các meter.Biểu đồ đường (line chart) theo thời gian.

Hình 5.2: Trang Dashboard tổng quan

Trang Dự báo: Hiển thị kết quả từ mô hình Deep Learning và so sánh đường dự báo và đường thực tế.

Hình 5.3: Trang dự báo

Trang Phát hiện bất thường: Hiển thị các thời điểm có tiêu thụ vượt ngưỡngvà tô màu đỏ/cảnh báo

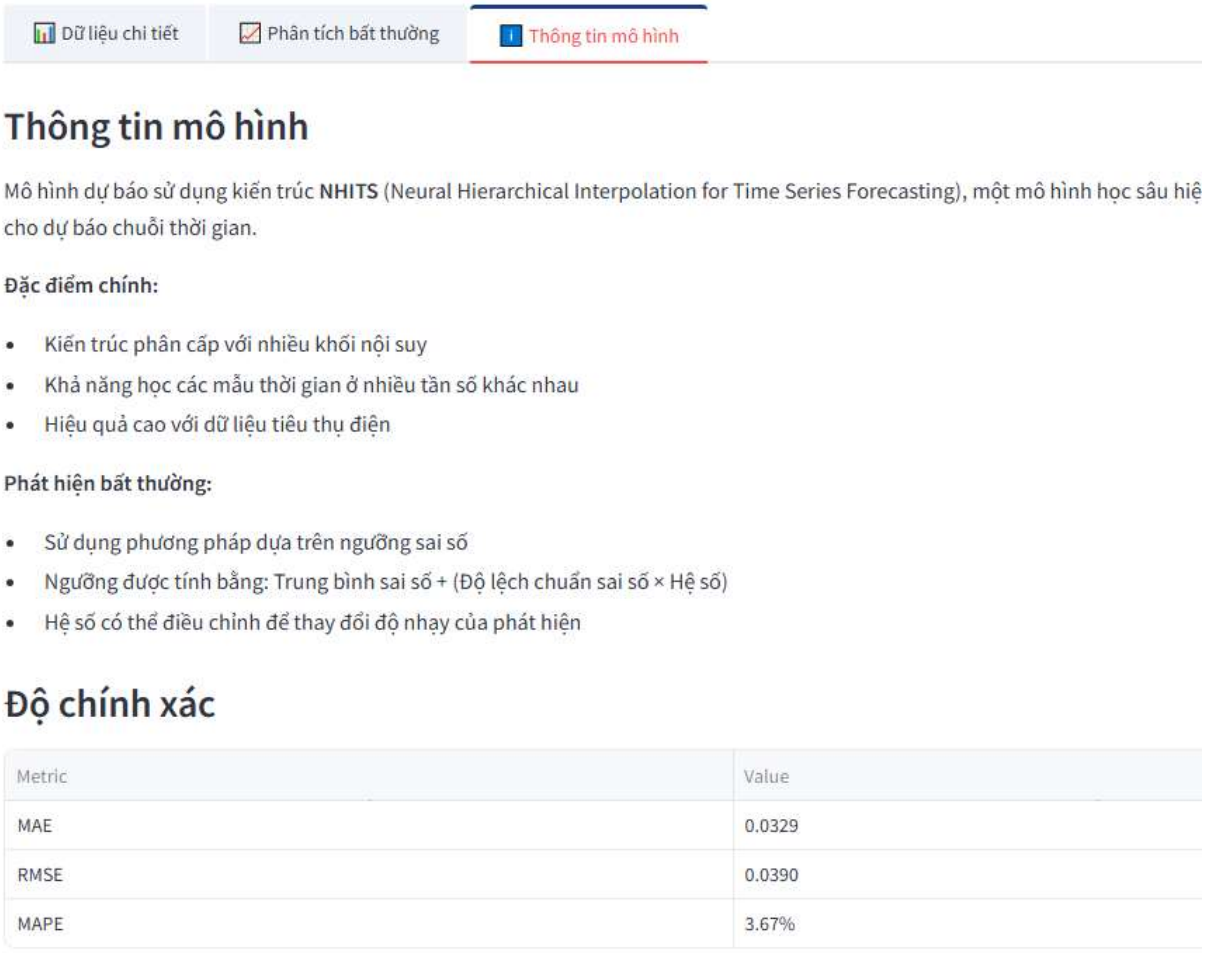
Hình 5.4: Trang phát hiện bất thường

Trang dữ liệu chi tiết: Có thể xem được thời cụ thể chính xác dữ liệu của công tơ muốn xem, đồng thời chỉ ra được điểm bất thường tại vị trí cụ thể nào

Thời gian	Thực tế (kWh)	Dự báo (kWh)	Sai số (kWh)	Trạng thái
2025-04-03 12:00:00	0.04	0.1036	0.0636	Bình thường
2025-04-03 18:00:00	0.5467	0.4446	0.1021	Bình thường
2025-04-04 00:00:00	0.146	0.113	0.033	Bình thường
2025-04-04 06:00:00	0.12	0.1588	0.0388	Bình thường
2025-04-04 12:00:00	0.055	0.1004	0.0454	Bình thường
2025-04-04 18:00:00	0.76	0.4308	0.3292	Bất thường
2025-04-05 00:00:00	0.0943	0.1257	0.0314	Bình thường
2025-04-05 06:00:00	0.145	0.1511	0.0061	Bình thường
2025-04-05 12:00:00	0.0667	0.0927	0.026	Bình thường
2025-04-05 18:00:00	0.5217	0.4125	0.1092	Bình thường

Hình 5.5: Trang dữ liệu chi tiết

Trang thông tin mô hình: Cho biết kết quả đánh giá model



Hình 5.6: Trang thông tin mô hình

5.6. Triển khai (deployment) ứng dụng lên GitHub và chia sẻ công khai

BƯỚC 1: Chuẩn bị ứng dụng Streamlit

Tạo một thư mục mới cho ứng dụng của bạn.

Tạo file Python chính của ứng dụng

Tạo file requirements.txt để khai báo các thư viện cần cài khi deploy.

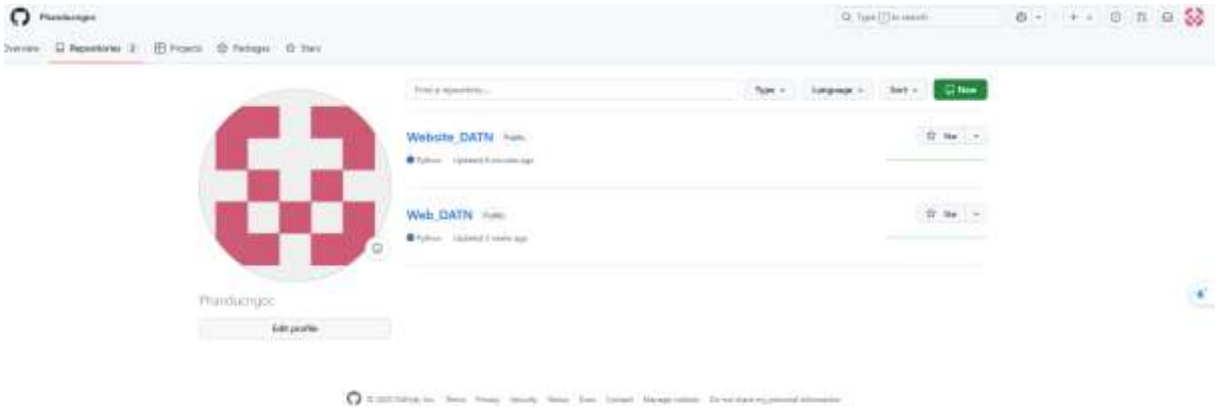
Hình 5.7: Tạo file requirements.txt

BƯỚC 2: Tạo và đẩy dự án lên GitHub

Mở Git Bash hoặc Terminal trong thư mục chứa ứng dụng.

Khởi tạo Git repository trong thư mục ứng dụng.

Tạo một repository mới trên GitHub (public).



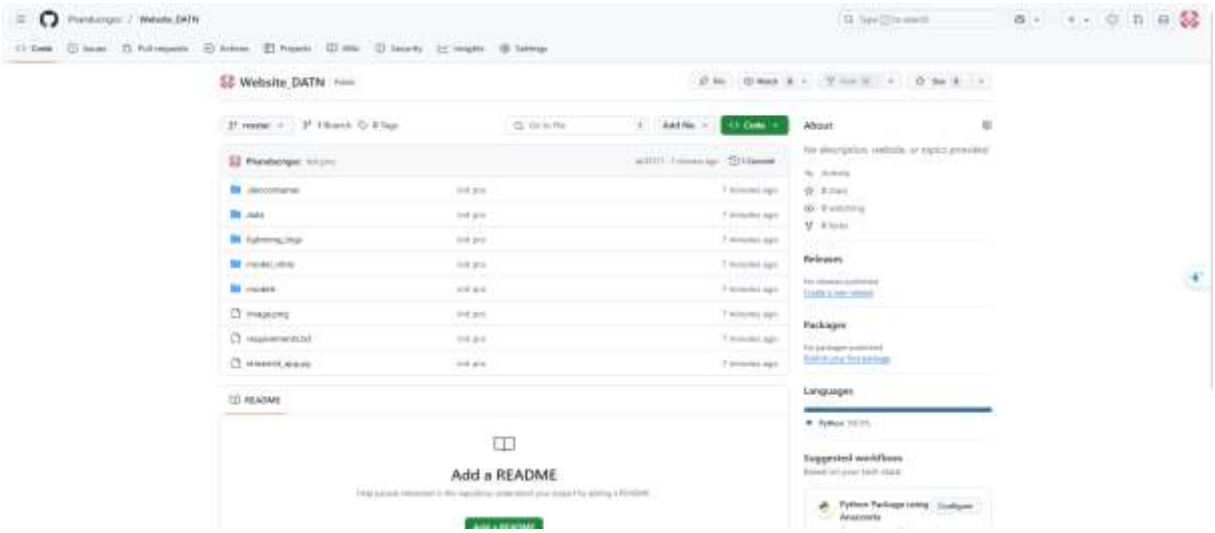
Hình 5.8: Tạo một repository mới trên GitHub

Liên kết repository trên máy tính với repository trên GitHub bằng địa chỉ (URL) từ GitHub.

Thêm tất cả các file và tạo commit đầu tiên.

Đặt tên nhánh chính là main (hoặc master nếu dùng mặc định cũ).

Đẩy toàn bộ mã nguồn lên GitHub.



Hình 5.9: Đẩy toàn bộ mã nguồn lên GitHub

BƯỚC 3: Deploy ứng dụng lên Streamlit Community Cloud

Truy cập trang https://share.streamlit.io.

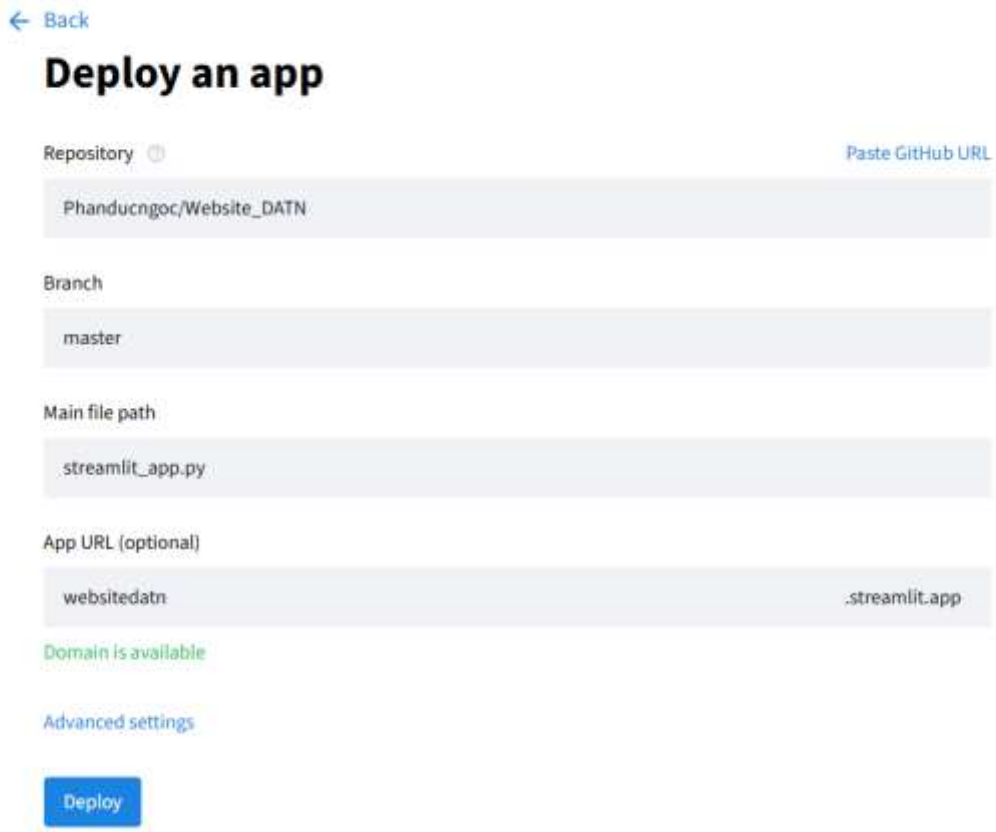
Đăng nhập bằng tài khoản GitHub.

Nhấn nút "Create app".

Chọn repository chứa ứng dụng của bạn từ GitHub.

Chọn nhánh và file chạy chính.

Nhấn nút "Deploy".



Hình 5.10: Deploy ứng dụng lên Streamlit Community Cloud

5.7. Đánh giá mô hình.

- ***Ưu điểm***:
  - Dễ triển khai, chạy trực tiếp bằng Streamlit.
  - Hiển thị trực quan, dễ sử dụng.
- ***Hạn chế:***

- Tối ưu cho nhiều người truy cập đồng thời.
- Cần nâng cấp nếu triển khai thực tế trên server lớn.

5.8. Kết luận chương 5

Trong chương này, nhóm đã trình bày quá trình xây dựng và triển khai hệ thống giám sát điện năng tiêu thụ thông qua giao diện web sử dụng nền tảng Streamlit. Giao diện được thiết kế trực quan, cho phép hiển thị dữ liệu, dự báo sản lượng tiêu thụ điện và cảnh báo bất thường theo thời gian thực. Việc triển khai hệ thống đã chứng minh tính khả thi và hiệu quả của mô hình trong ứng dụng thực tế, đáp ứng mục tiêu đề ra của đề tài.

5.9. Đánh giá tổng thể hệ thống

Hệ thống được xây dựng trong đề tài bao gồm ba thành phần chính: thu thập dữ liệu từ xa, dự báo sản lượng điện tiêu thụ bằng các mô hình học sâu, và giao diện web hiển thị kết quả cùng chức năng cảnh báo bất thường. Việc tích hợp toàn bộ quy trình này cho phép giám sát tự động sản lượng điện của hàng trăm khách hàng theo thời gian thực.

Dựa trên tập dữ liệu hơn 270.000 mẫu của 559 hộ tiêu thụ thu thập từ hệ thống công tơ tại TP. Đà Nẵng, các mô hình học sâu như LSTM, Bi-LSTM và N-HiTS đã được huấn luyện và so sánh. Kết quả cho thấy mô hình N-HiTS có độ chính xác cao nhất. Điều này chứng minh rằng mô hình N-HiTS phù hợp hơn với bài toán chuỗi thời gian có dao động mạnh như dữ liệu tiêu thụ điện.

Ngoài ra, việc triển khai hệ thống dự báo trên giao diện web giúp hiển thị sản lượng điện từng khách hàng theo thời gian, đồng thời xác định bất thường nếu sai số giữa dự báo và thực tế vượt quá ngưỡng cho phép. Đây là bước tiến quan trọng so với hệ thống thu thập dữ liệu truyền thống vốn chỉ dừng lại ở việc ghi nhận chỉ số công tơ.

Hệ thống có thể mở rộng và áp dụng trong các trạm biến áp với hàng trăm hộ tiêu thụ. Nhờ vào tính tự động, tính toán nhanh và có thể phát hiện bất thường theo thời gian thực, hệ thống này hoàn toàn có tiềm năng triển khai trong thực tế để hỗ trợ quản lý năng lượng, cảnh báo sớm thất thoát, gian lận điện hoặc sử dụng điện không hiệu quả.

CHƯƠNG 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN ĐỀ TÀI

6.1. Kết luận

Trong quá trình thực hiện đồ án "Nghiên cứu hệ thống thu thập dữ liệu và giám sát tiêu thụ điện bằng Deep Learning", nhóm đã nghiên cứu và đề xuất một giải pháp ứng dụng công nghệ học sâu để dự báo sản lượng điện năng tiêu thụ và phát hiện bất thường trong dữ liệu điện theo thời gian thực. Dựa trên dữ liệu thu thập từ hệ thống đo đếm từ xa tại TP. Đà Nẵng, nhóm đã tiến hành xử lý dữ liệu, xây dựng các mô hình học sâu như N-HiTs, LSTM và Bi-LSTM, so sánh kết quả dự báo để lựa chọn mô hình tối ưu, sau đó triển khai mô hình vào một hệ thống giao diện web có khả năng hiển thị và cảnh báo trực quan. Kết quả thực nghiệm cho thấy mô hình N-HiTs có độ chính xác cao nhất trong các mô hình về dự báo sản lượng điện tiêu thụ, với sai số thấp hơn so với các mô hình còn lại theo các chỉ số MAE, RMSE và MAPE. Việc tích hợp mô hình vào hệ thống giao diện web giúp người dùng dễ dàng giám sát dữ liệu, kiểm tra xu hướng tiêu thụ và phát hiện các trường hợp tiêu thụ bất thường vượt quá ngưỡng cho phép. Đây là một bước tiến đáng kể trong việc tự động hóa và nâng cao hiệu quả công tác vận hành hệ thống điện tại địa phương. Đồ án đã thể hiện được một hướng tiếp cận mới trong việc sử dụng trí tuệ nhân tạo để giám sát dữ liệu đo đếm điện, từ đó mở ra nhiều khả năng ứng dụng trong các lĩnh vực khác liên quan đến năng lượng và giám sát hệ thống.

6.2. Hướng phát triển đề tài

Tuy đã đạt được những kết quả bước đầu khả quan, đề tài vẫn còn một số hạn chế và có thể phát triển thêm theo các hướng sau:

Mở rộng quy mô dữ liệu: Dữ liệu hiện tại chỉ giới hạn trong một khu vực, có thể mở rộng ra các khu vực khác và xử lý đồng thời nhiều loại dữ liệu hơn như dữ liệu thời tiết, dữ liệu phụ tải đặc biệt, để cải thiện chất lượng dự báo.

Tự động cập nhật mô hình: Hiện tại mô hình chưa tự động học từ dữ liệu mới. Trong tương lai, có thể phát triển thêm khả năng huấn luyện lại định kỳ theo thời gian để mô hình thích nghi tốt hơn với sự thay đổi của hệ thống.

Tối ưu thuật toán phát hiện bất thường: Hiện tại việc phát hiện bất thường chủ yếu dựa vào ngưỡng sai lệch so với dự báo. Có thể nghiên cứu thêm các thuật toán AI khác như Isolation Forest, AutoEncoder hoặc thống kê thống kê Bayes để tăng độ tin cậy của cảnh báo.

Triển khai thực tế quy mô lớn: Xây dựng hệ thống dashboard giám sát theo cấp độ ngành điện, áp dụng vào giám sát nhiều trạm biến áp, nhiều nhóm khách hàng và cung cấp API tích hợp vào các hệ thống SCADA, MDMS hiện hành.

Xây dựng hệ thống cảnh báo thông minh đa kênh: Ngoài giao diện web, có thể tích hợp cảnh báo qua email, SMS,… để kịp thời thông báo bất thường cho cán bộ vận hành.

TÀI LIỆU THAM KHẢO

[1] L. H. C. V. H. Huỳnh Thảo Nguyên, "Ứng dụng lý thuyết thống kê trong phân tích sản lượng điện bất thường thời gian thực từ đo xa," 2021.

[2] J. Brownlee, "How to Develop LSTM Models for Time Series Forecasting," Machine Learning Mastery, 2020.

[3] guslovesmath, "Amazing NVDA Forecasting – neuralForecast," Kaggle.

[4] B. Ibrahim, L. Rabelo, E. Gutierrez-Franco and N. Clavijo-Buritica, "Machine Learning for Short-Term Load Forecasting in Smart Grids," *Energies,* vol. 15, no. 21, p. 8079, 2022.

[5] S. Xu, H. Sun, B. Li, D. Zhao, H. Li, K. Wang, J. Liu and X. Wang, "-LSTM-based load forecasting for power grid," *Conference Series,* vol. 2781, no. 1, p. 012002, 2024.

[6] Y. Yu, X. Si, C. Hu and J. Zhang, "LSTM Cells and Network Architectures," *Neural Computation ,* vol. 31, no. 7, pp. 1235-1270, 2019.

[7] S. Siami-Namini, N. Tavakoli and A. S. Namin, "The Performance of LSTM and BiLSTM in Forecasting Time Series," in *IEEE International Conference on Big Data*, 2019.

[8] T. H. Phúc, N. V. Hoàng, L. B. N. Long, N. H. Tuấn and L. A. Duy, "Dự đoán giá cổ phiếu ba ngân hàng: BIDV, Vietcombank, Eximbank," Trường ĐH Công nghệ Thông tin, TP. Hồ Chí Minh, Hồ Chí Minh, Việt Nam.

[9] C. Challu, K. G. Olivares, B. N. Oreshkin, F. G. Ramírez, M. Mergenthaler-Canseco and A. Dubrawski, "N-HiTs: Neural Hierarchical Interpolation for Time Series Forecasting," *AAAI Conference on Artificial Intelligence,* vol. 37, no. 6, pp. 6989-6997, 2023.

[10] N. (. documentation), "NHITS – NeuralForecast Models Usage Example," Nixtlaverse – Nixtla, 2025.

[11] H. Alizadegan, B. R. Malki, A. Radmehr, H. Karimi and M. A. Ilani, "Comparative study of long short-term memory (LSTM), bidirectional LSTM, and traditional machine learning approaches for energy consumption prediction," *Energy Exploration & Exploitation,* vol. 43, no. 1, pp. 281-301, 2024.

[12] B. N. Oreshkin, D. Carpov, N. Chapados and Y. Bengio, "N-BEATS: Neural Basis Expansion Analysis for Interpretable Time Series Forecasting," in *International Conference on Learning Representations (ICLR)*, 2020.

PHỤ LỤC 1
Chương trình tiền xử lí dữ liệu

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import numpy as np
df=pd.read_csv('FC53FFRU.csv')
df.head(-1)
# Giả sử df là DataFrame của bạn và cột NGAYGIO đã là datetime
df['NGAYGIO'] = pd.to_datetime(df['NGAYGIO'], dayfirst=True,
errors='coerce')
# Hàm làm tròn theo quy tắc 30 phút
def round_nearest_hour(ts):
    minute = ts.minute
    if minute < 30:
        return ts.replace(minute=0, second=0, microsecond=0)
    else:
        return (ts + pd.Timedelta(hours=1)).replace(minute=0, second=0,
microsecond=0)

# Áp dụng
df['NGAYGIO'] = df['NGAYGIO'].apply(round_nearest_hour)

# Loại bỏ các timestamp bị lỗi nếu có
df = df.dropna(subset=['NGAYGIO'])

# Nhóm theo từng số công tơ ()
def remove_duplicate_timestamps(group):
# Xoá các bản ghi có NGAYGIO trùng nhau trong cùng điểm đo → giữ dòng
đầu tiên
    return group.drop_duplicates(subset='NGAYGIO', keep='first')

# Áp dụng cho từng nhóm SO_CTO
df_cleaned = df.groupby('SO_CTO').apply(remove_duplicate_timestamps)

# Bỏ multi-index sau groupby
df_cleaned = df_cleaned.reset_index(drop=True)

# Đảm bảo NGAYGIO là kiểu datetime
df_cleaned['NGAYGIO'] = pd.to_datetime(df_cleaned['NGAYGIO'])
```

```
# Sắp xếp theo SO_CTO và thời gian
df_cleaned.sort_values(by=['SO_CTO', 'NGAYGIO'], inplace=True)

# Tính toán cho từng SO_CTO
def compute_tv_delta(group):
    group['Time Delta'] = group['NGAYGIO'].diff() / np.timedelta64(1,
'h')  # đổi ra giờ
    group['value import'] = group['IMPORT_KWH'].diff()
    group['tv delta'] = group['value import'] / group['Time Delta']
    return group

# Áp dụng theo nhóm
df_cleaned = df_cleaned.groupby('SO_CTO').apply(compute_tv_delta)

# Xoá multi-index nếu có
df_cleaned = df_cleaned.reset_index(drop=True)
#Xoá NAn
df_cleaned = df_cleaned.dropna()
df_cleaned.reset_index(drop=True, inplace=True)
df_cleaned = df_cleaned.drop(['MA_DIEMDO','MA_TRAM',
'MA_DVIQLY','EXPORT_KWH'], axis=1)

def train_test_split_by_group(df, test_ratio=0.2):
    train_val_parts = []
    test_parts = []

    for id_, group in df.groupby("SO_CTO"):
        group = group.sort_values("NGAYGIO").reset_index(drop=True)
        n_total = len(group)
        n_test = int(n_total * test_ratio)

        test_part = group.iloc[-n_test:]
        train_val_part = group.iloc[:-n_test]

        train_val_parts.append(train_val_part)
        test_parts.append(test_part)

    df_train_val = pd.concat(train_val_parts).reset_index(drop=True)
    df_test = pd.concat(test_parts).reset_index(drop=True)

    return df_train_val, df_test

df_train_val, df_test = train_test_split_by_group(df_cleaned,
test_ratio=0.2)

from scipy.stats import zscore
```

```
import numpy as np
import pandas as pd

# Giả sử bạn đã có df_cleaned với cột 'tv delta'

# Tính Z-score cho từng công tơ riêng biệt để tránh sai lệch
def remove_outliers_and_interpolate(group, threshold=3):
    # Tính z-score
    z_scores = zscore(group['tv delta'], nan_policy='omit')

    # Đánh dấu các giá trị là outlier nếu |z| > threshold
    mask_outliers = np.abs(z_scores) > threshold

    # Gán NaN cho các outlier
    group.loc[mask_outliers, 'tv delta'] = np.nan

    # Nội suy (interpolate) lại giá trị thiếu theo thời gian
    group['tv delta'] = group['tv delta'].interpolate(method='linear',
limit_direction='both')
    return group

# Áp dụng theo từng SO_CTO
df_train_val =
df_train_val.groupby('SO_CTO').apply(remove_outliers_and_interpolate)
# Reset lại index nếu cần
df_train_val = df_train_val.reset_index(drop=True)
df=df_train_val
# Đảm bảo kiểu đúng
df['SO_CTO'] = df['SO_CTO'].astype(str)
df['NGAYGIO'] = pd.to_datetime(df['NGAYGIO'], utc=True)
# 2) Hàm resample + nội suy + fill cho mỗi SO_CTO
def resample_and_fill(group, freq='6H'):
    # Đặt index để resample
    g = group.set_index('NGAYGIO').sort_index()

    # Resample giá trị gốc về 6H, dùng hàm first() để giữ tv delta ở
mốc thật
    y = g['tv delta'].resample(freq).first()

    # 2.2 Interpolate theo time, cả đầu/cuối
    y = y.interpolate(method='time', limit_direction='both')

    # 2.3 Forward-fill và backward-fill để lấp nếu vẫn còn NaN
    y = y.ffill().bfill()

    # 2.4 Trả về DataFrame với 2 cột: ds và tv delta
```

```
    df_out = y.reset_index().rename(columns={'NGAYGIO':'NGAYGIO', 'tv
delta':'tv delta'})
    df_out['SO_CTO'] = group['SO_CTO'].iloc[0]
    return df_out

# 3) Áp dụng cho toàn bộ các SO_CTO
df_interp = (
    df
    .groupby('SO_CTO', group_keys=False)
    .apply(lambda g: resample_and_fill(g, freq='6H'))
    .reset_index(drop=True)
)
df_list = []

for so_cto, group in df.groupby('SO_CTO'):
    group = group.sort_values('NGAYGIO')
    split_idx = int(len(group) * 0.8)

    train_part = group.iloc[:split_idx].copy()
    test_part = group.iloc[split_idx:].copy()

    df_list.append((train_part, test_part))

# Ghép lại toàn bộ dữ liệu
df_train = pd.concat([pair[0] for pair in df_list])
df_test = pd.concat([pair[1] for pair in df_list])
pd.DataFrame(df_train, columns=df.columns).to_csv('train.csv',
index=False)
pd.DataFrame(df_test, columns=df.columns).to_csv('test.csv',
index=False)
from google.colab import files
files.download('train.csv')
files.download('test.csv')
```

PHỤ LỤC 2
Chương trình xây dựng và huấn luyện mô hình N-HiTs

```
#import các thư viện cần thiết
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error,
mean_absolute_percentage_error, mean_squared_error
from neuralforecast import NeuralForecast
from neuralforecast.models import NHITS
import logging
import warnings
# --------- STEP 1: Load và chuẩn hóa dữ liệu từ 2 file riêng ---------
-
def load_and_preprocess(train_path, test_path):
    train_df = pd.read_csv(train_path, parse_dates=['NGAYGIO'])
    test_df  = pd.read_csv(test_path,  parse_dates=['NGAYGIO'])

# Đổi tên cột và chuẩn hóa ds
    for df in (train_df, test_df):
        df.rename(columns={
            'SO_CTO': 'unique_id',
            'NGAYGIO': 'ds',
            'tv delta': 'y'
        }, inplace=True)
        if df['ds'].dt.tz is not None:
            df['ds'] = df['ds'].dt.tz_convert(None)
        df.sort_values(['unique_id', 'ds'], inplace=True)
        df.reset_index(drop=True, inplace=True)

# Fit scaler trên train và transform cả train + test
    scalers = {}
    train_scaled, test_scaled = [], []
    for uid, grp in train_df.groupby('unique_id'):
        g_train = grp.copy()
        scaler = MinMaxScaler()
        g_train['y'] = scaler.fit_transform(g_train[['y']])
        scalers[uid] = scaler
        train_scaled.append(g_train)

        g_test = test_df[test_df['unique_id'] == uid].copy()
        if not g_test.empty:
            g_test['y'] = scaler.transform(g_test[['y']])
            test_scaled.append(g_test)
```

```
    train_scaled_df = pd.concat(train_scaled).reset_index(drop=True)
    test_scaled_df  = pd.concat(test_scaled).reset_index(drop=True)
    return train_scaled_df, test_scaled_df, scalers


# --------- STEP 2: Xây dựng mô hình-----------
def train_nhits_model(train_df, input_len, forecast_len):
    nhits_params = {
        'h': forecast_len,
        'input_size': input_len,
        'n_pool_kernel_size': [4, 2, 1],
        'n_freq_downsample': [4, 2, 1],
        'dropout_prob_theta': 0.5,
        'activation': 'ReLU',
        'learning_rate': 0.001,
        'batch_size': 64,
        'max_steps': 1500,
        'random_seed': 42
    }

    model = NHITS(**nhits_params)

    nf = NeuralForecast(
        models=[model],
        freq='6H'
    )

    nf.fit(df=train_df)
    return nf

# --------- Helper: Lấy lịch sử input_len từ train + test ----------
def get_history_from_train_test(train_df, test_df, unique_id,
start_date, input_len):
    train_uid = train_df[train_df['unique_id'] ==
unique_id].sort_values('ds')
    test_uid  = test_df[test_df['unique_id'] ==
unique_id].sort_values('ds')
    full = pd.concat([train_uid,
test_uid]).sort_values('ds').reset_index(drop=True)
    if full['ds'].dt.tz is not None:
        full['ds'] = full['ds'].dt.tz_convert(None)
    start_date = pd.to_datetime(start_date)

    # Tìm index của ngày bắt đầu
    idxs = full.index[full['ds'] == start_date]
    if len(idxs) == 0 or idxs[0] < input_len:
        return None
```

```
    idx = idxs[0]
    return full.iloc[idx-input_len:idx].copy()

# --------- STEP 3: Rolling forecast sử dụng global model ----------
def forecast_for_single_uid(model, train_df, test_df, uid, start_date,
input_len):
    history = get_history_from_train_test(train_df, test_df, uid,
start_date, input_len)
    if history is None:
        print(f" Không đủ lịch sử {input_len} cho uid {uid} tại
{start_date}")
        return None
    df_input = history.copy()
    df_input['unique_id'] = uid
    try:
        pred = model.predict(df=df_input)
        return pred
    except Exception as e:
        print(f" Lỗi dự báo cho uid {uid}: {e}")
        return None
# --------- STEP 4: Đánh giá ----------
def evaluate_forecast_per_id(test_df, forecast_df, scalers):
    merged = pd.merge(test_df, forecast_df, on=['unique_id', 'ds'],
how='inner')
    if merged.empty:
        print("⚠ Không có điểm chung giữa test và forecast.")
        return None, None, None, None, None

    results = []
    all_y_true = []
    all_y_pred = []

    for uid, sub in merged.groupby('unique_id'):
        scaler = scalers.get(uid)
        y_true = scaler.inverse_transform(sub[['y']])
        y_pred = scaler.inverse_transform(sub[['NHITS']])

        all_y_true.append(y_true)
        all_y_pred.append(y_pred)

        mae = mean_absolute_error(y_true, y_pred)
        mape = mean_absolute_percentage_error(y_true, y_pred) * 100
        rmse = np.sqrt(mean_squared_error(y_true, y_pred))


        results.append({
            'unique_id': uid,
```

```
            'MAE': mae,
            'MAPE (%)': mape,
            'RMSE': rmse
        })

    res_df = pd.DataFrame(results)

    y_true_all = np.vstack(all_y_true)
    y_pred_all = np.vstack(all_y_pred)

    overall_mae = mean_absolute_error(y_true_all, y_pred_all)
    overall_mape = mean_absolute_percentage_error(y_true_all,
y_pred_all) * 100
    overall_rmse = np.sqrt(mean_squared_error(y_true_all, y_pred_all))

    return merged, res_df, overall_mae, overall_mape, overall_rmse


# --------- STEP 5: Phát hiện bất thường ----------
def detect_anomalies(merged_df, scalers, threshold_std=3):
    anomalies = []
    for uid, sub in merged_df.groupby('unique_id'):
        scaler = scalers[uid]
        y_true = scaler.inverse_transform(sub[['y']])
        y_pred = scaler.inverse_transform(sub[['NHITS']])
        error = np.abs(y_true - y_pred)
        th = error.mean() + threshold_std * error.std()
        sub['anomaly'] = (error > th).astype(int)
        anomalies.append(sub[sub['anomaly'] == 1])
    if anomalies:
        df_an = pd.concat(anomalies).reset_index(drop=True)
        df_an.to_csv('anomalies.csv', index=False)
        print(f" Đã lưu danh sách {len(df_an)} điểm bất thường vào
anomalies.csv")
        return df_an
    else:
        print(" Không phát hiện bất thường.")
        return pd.DataFrame()

# --------- STEP 6: Vẽ biểu đồ ----------
def plot_with_anomalies(merged_df, scalers, unique_id):
    sub = merged_df[merged_df['unique_id'] == unique_id].copy()
    scaler = scalers[unique_id]
    sub['y_true'] = scaler.inverse_transform(sub[['y']])
    sub['y_pred'] = scaler.inverse_transform(sub[['NHITS']])
    sub['error'] = np.abs(sub['y_true'] - sub['y_pred'])
    th = sub['error'].mean() + 3 * sub['error'].std()
```

```
    sub['anomaly'] = sub['error'] > th
    plt.figure(figsize=(12,5))
    plt.plot(sub['ds'], sub['y_true'], marker='o', label='Thực tế')
    plt.plot(sub['ds'], sub['y_pred'], marker='x', label='Dự báo')
    an = sub[sub['anomaly']]
    if not an.empty:
        plt.scatter(an['ds'], an['y_true'], color='red', label='Bất
thường', zorder=5)
    plt.title(f'Dự báo & Bất thường - Công tơ {unique_id}')
    plt.xlabel('Thời gian'); plt.ylabel('Điện năng (kWh)');
plt.legend(); plt.grid(True); plt.xticks(rotation=45);
plt.tight_layout(); plt.show()


# ==================== MAIN RUN ====================
if __name__ == '__main__':
    train_path = '/content/train.csv'
    test_path  = '/content/test.csv'
    input_len    = 56
    forecast_len = 28

    # Load & preprocess
    train_df, test_df, scalers = load_and_preprocess(train_path,
test_path)

    # Train global model trên toàn bộ train
    print(" Huấn luyện NHITS trên toàn bộ tập train...")
    model = train_nhits_model(train_df, input_len, forecast_len)

    # Dự báo cho toàn bộ công tơ và in Top 10 công tơ có MAPE thấp nhất
    all_preds = []
    for uid in test_df['unique_id'].unique():
        start_date = test_df[test_df['unique_id'] == uid]['ds'].min()
        pred = forecast_for_single_uid(model, train_df, test_df, uid,
start_date, input_len)
        if pred is not None:
            all_preds.append(pred)

    if all_preds:
                all_pred_df = pd.concat(all_preds)
                merged_all, res_df, overall_mae, overall_mape,
overall_rmse = evaluate_forecast_per_id(
                    test_df, all_pred_df, scalers
                )

    if res_df is not None and not res_df.empty:
        # Làm tròn kết quả
```

```
        res_df_rounded = res_df.copy()
        res_df_rounded['MAE'] = res_df_rounded['MAE'].round(6)
        res_df_rounded['MAPE (%)'] = res_df_rounded['MAPE
(%)'].round(6)
        res_df_rounded['RMSE'] = res_df_rounded['RMSE'].round(6)

        # In bảng đầy đủ (tùy bạn muốn giữ hay không)
        pd.set_option('display.max_rows', 10)  # Giới hạn số dòng in ra
(hoặc bỏ nếu bạn muốn xem hết)
        print(res_df_rounded)

        # Top công tơ MAPE thấp nhất
        top_mape = res_df_rounded.sort_values('MAPE (%)').head(10)

        print("\n Top 10 công tơ có MAPE nhỏ nhất:")
        print(top_mape.to_string(index=False))

        # Tổng quan mô hình toàn bộ
        print("\n Tổng quan mô hình (toàn bộ tập test):")
        print("=======================================")
        print(f" MAE  toàn mô hình : {overall_mae:.6f}")
        print(f" MAPE toàn mô hình : {overall_mape:.6f} %")
        print(f" RMSE toàn mô hình : {overall_rmse:.6f}")
        print("=======================================")

        # Phát hiện bất thường
        df_anomaly = detect_anomalies(merged_all, scalers)


    # Nhập tương tác từ user để dự báo cho từng công tơ cụ thể
    while True:
        user_input = input("Nhập unique_id và ngày bắt đầu dự báo
(YYYY-MM-DD HH:MM:SS), cách nhau bởi dấu phẩy, hoặc 'exit': ")
        if user_input.strip().lower() == 'exit':
            print("Kết thúc chương trình.")
            break
        try:
            uid_str, date_str = [s.strip() for s in
user_input.split(',')]
            uid = int(uid_str)
            start_date = pd.to_datetime(date_str)
        except Exception:
            print(" Định dạng sai. Vui lòng nhập: unique_id, YYYY-MM-DD
HH:MM:SS")
            continue

        if uid not in test_df['unique_id'].unique():
```

```
            print(f" unique_id {uid} không có trong test set.")
            continue

        pred = forecast_for_single_uid(model, train_df, test_df, uid,
start_date, input_len)
        if pred is None:
            continue
        print(pred)

        merged, res_df, overall_mae, overall_mape, overall_rmse =
evaluate_forecast_per_id(test_df, pred, scalers)

        if merged is not None:
            plot_with_anomalies(merged, scalers, uid)
```

PHỤ LỤC 3
Chương trình xây dựng và huấn luyện mô hình LSTM và Bi-LSTM

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import numpy as np

df=pd.read_csv('/content/FC53FFRU.csv')

# ---------- 1. Chia dữ liệu đào tạo mô hình và phát hiện bất thường --
--------
df['NGAYGIO'] = pd.to_datetime(df['NGAYGIO'], dayfirst=True,
errors='coerce')
def round_nearest_hour(ts):
    minute = ts.minute
    if minute < 30:
        return ts.replace(minute=0, second=0, microsecond=0)
    else:
        return (ts + pd.Timedelta(hours=1)).replace(minute=0, second=0,
microsecond=0)

df['NGAYGIO'] = df['NGAYGIO'].apply(round_nearest_hour)

df = df.dropna(subset=['NGAYGIO'])
def remove_duplicate_timestamps(group):
    # Xoá các bản ghi có NGAYGIO trùng nhau trong cùng điểm đo → giữ
dòng đầu tiên
    return group.drop_duplicates(subset='NGAYGIO', keep='first')
df_cleaned = df.groupby('SO_CTO').apply(remove_duplicate_timestamps)
df_cleaned = df_cleaned.reset_index(drop=True)

df_cleaned['NGAYGIO'] = pd.to_datetime(df_cleaned['NGAYGIO'])
df_cleaned.sort_values(by=['SO_CTO', 'NGAYGIO'], inplace=True)

def compute_tv_delta(group):
    group['Time Delta'] = group['NGAYGIO'].diff() / np.timedelta64(1,
'h')
    group['value import'] = group['IMPORT_KWH'].diff()
    group['tv delta'] = group['value import'] / group['Time Delta']
    return group
df_cleaned = df_cleaned.groupby('SO_CTO').apply(compute_tv_delta)
df_cleaned = df_cleaned.reset_index(drop=True)
df_cleaned = df_cleaned.dropna()
df_cleaned.reset_index(drop=True, inplace=True)
```

```
df_cleaned = df_cleaned.drop(['MA_DIEMDO','MA_TRAM',
'MA_DVIQLY','EXPORT_KWH'], axis=1)

def train_test_split_by_group(df, test_ratio=0.2):
    train_val_parts = []
    test_parts = []

    for id_, group in df.groupby("SO_CTO"):
        group = group.sort_values("NGAYGIO").reset_index(drop=True)
        n_total = len(group)
        n_test = int(n_total * test_ratio)
        test_part = group.iloc[-n_test:]
        train_val_part = group.iloc[:-n_test]

        train_val_parts.append(train_val_part)
        test_parts.append(test_part)

    df_train_val = pd.concat(train_val_parts).reset_index(drop=True)
    df_test = pd.concat(test_parts).reset_index(drop=True)

    return df_train_val, df_test

df_train_val, df_test = train_test_split_by_group(df_cleaned,
test_ratio=0.2)
train=df_train_val

# ---------- 2. Xử lí dữ liệu train mô hình ----------
def create_lag_features(df, target_col='tv delta', group_col='SO_CTO',
lags=[1, 2, 3, 4, 8, 12, 16, 20, 24, 28,56], fill_method='bfill'):
    df_sorted = df.sort_values(by=[group_col, 'NGAYGIO']).copy()
    for lag in lags:
        df_sorted[f'{target_col}_lag{lag}'] =
df_sorted.groupby(group_col)[target_col].shift(lag)

    # Xử lý NaN
    lag_cols = [f'{target_col}_lag{lag}' for lag in lags]
    if fill_method == 'bfill':
        df_sorted[lag_cols] =
df_sorted[lag_cols].fillna(method='bfill')
    elif fill_method == 'ffill':
        df_sorted[lag_cols] =
df_sorted[lag_cols].fillna(method='ffill')

    return df_sorted

df_lagged = create_lag_features(train, target_col='tv delta',
group_col='SO_CTO')
```

```
# Chia train/test theo thời gian của mỗi hộ
def split_by_time(df, test_ratio=0.3):
    train_parts, test_parts = [], []
    for id_, group in df.groupby('SO_CTO'):
        n_total = len(group)
        n_test = int(n_total * test_ratio)
        train_part = group.iloc[:-n_test]
        test_part = group.iloc[-n_test:]
        train_parts.append(train_part)
        test_parts.append(test_part)
    return pd.concat(train_parts), pd.concat(test_parts)

df_train, df_test = split_by_time(df_lagged, test_ratio=0.3)

# ---------- 3. Chuẩn hóa theo từng hộ ----------
from sklearn.preprocessing import StandardScaler
def scale_groupwise(df, features, group_col='SO_CTO'):
    df_scaled = pd.DataFrame(index=df.index)
    scalers = {}

    for so_cto in df[group_col].unique():
        group_idx = df[group_col] == so_cto
        scaler = StandardScaler()
        scaled_data = scaler.fit_transform(df.loc[group_idx, features])
        df_scaled.loc[group_idx, features] = scaled_data
        scalers[so_cto] = scaler

    return df_scaled.astype(float), scalers

feature_cols = ['tv delta_lag1', 'tv delta_lag2', 'tv delta_lag3', 'tv
delta_lag4',
                'tv delta_lag8', 'tv delta_lag12', 'tv delta_lag16',
'tv delta_lag20',
                'tv delta_lag24', 'tv delta_lag28','tv delta_lag56']

X_train_scaled, scalers_train = scale_groupwise(df_train, feature_cols,
group_col='SO_CTO')
X_test_scaled, _ = scale_groupwise(df_test, feature_cols,
group_col='SO_CTO')

# ---------- 4. Tạo LabelEncoder cho SO_CTO ----------
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
le.fit(df_lagged['SO_CTO'])

df_train['SO_CTO_enc'] = le.transform(df_train['SO_CTO'])
```

```
df_test['SO_CTO_enc'] = le.transform(df_test['SO_CTO'])

# ---------- 5. Tạo input output cho train/test----------
def create_dataset_grouped_from_scaled(X_scaled, y, so_cto_enc,
input_len=56, output_len=28):
    X_seq, y_seq, house_ids = [], [], []
    total_len = input_len + output_len

    df_tmp = pd.DataFrame(X_scaled.values, columns=[f'feature_{i}' for
i in range(X_scaled.shape[1])])
    df_tmp['tv delta'] = y.values
    df_tmp['SO_CTO_enc'] = so_cto_enc.values
    df_tmp = df_tmp.reset_index(drop=True)

    for house_id in df_tmp['SO_CTO_enc'].unique():
        df_house = df_tmp[df_tmp['SO_CTO_enc'] ==
house_id].reset_index(drop=True)

        features = df_house[[f'feature_{i}' for i in
range(X_scaled.shape[1])]].values
        labels = df_house['tv delta'].values

        for i in range(len(df_house) - total_len + 1):
            x = features[i:i + input_len]
            y_out = labels[i + input_len:i + input_len + output_len]
            if x.shape[0] == input_len and y_out.shape[0] ==
output_len:
                X_seq.append(x)
                y_seq.append(y_out)
                house_ids.append(house_id)

    return np.array(X_seq), np.array(y_seq), np.array(house_ids)

X_train_seq, y_train_seq, house_ids_train =
create_dataset_grouped_from_scaled(X_train_scaled, df_train['tv
delta'], df_train['SO_CTO_enc'])

X_test_seq, y_test_seq, house_ids_test =
create_dataset_grouped_from_scaled(X_test_scaled, df_test['tv delta'],
df_test['SO_CTO_enc'])

print("X_train_seq shape:", X_train_seq.shape)
print("y_train_seq shape:", y_train_seq.shape)
print("X_test_seq shape:", X_test_seq.shape)
print("y_test_seq shape:", y_test_seq.shape)

# ---------- 6. Xây dựng mô hình----------
```

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, LSTM, Bidirectional, Dense,
Embedding, Concatenate, Reshape, RepeatVector, Flatten

def build_lstm_model(input_seq_len, feature_dim, output_seq_len,
n_households, use_bilstm=False):
    # Inputs
    input_seq = Input(shape=(input_seq_len, feature_dim),
name='input_seq')
    household_id = Input(shape=(1,), name='household_id')

    # Embedding ID
    # Embedding output shape: (batch_size, 1, embed_dim)
    embed = Embedding(input_dim=n_households,
output_dim=16)(household_id)

    # Flatten the embedding to get shape (batch_size, embed_dim)
    embed_flat = Flatten()(embed)

    # Repeat the flattened embedding across the time steps
    # RepeatVector output shape: (batch_size, input_seq_len, embed_dim)
    embed_repeat = RepeatVector(input_seq_len)(embed_flat)

    # Concatenate embedding + input_seq
    # Concatenate output shape: (batch_size, input_seq_len, feature_dim
+ embed_dim)
    x = Concatenate(axis=-1)([input_seq, embed_repeat])

    # LSTM/Bi-LSTM
    if use_bilstm:
        # Bi-LSTM output shape (return_sequences=False): (batch_size,
units*2)
        x = Bidirectional(LSTM(128, return_sequences=False))(x)
    else:
        # LSTM output shape (return_sequences=False): (batch_size,
units)
        x = LSTM(128, return_sequences=False)(x)

    # Output 28 bước
    # Dense layer output shape: (batch_size, output_seq_len)
    output = Dense(output_seq_len)(x)

    model = Model(inputs=[input_seq, household_id], outputs=output)
    model.compile(optimizer='adam', loss='mse', metrics=['mae'])

    return model
```

```
# Dữ liệu đầu vào
input_seq_len = X_train_seq.shape[1]
feature_dim = X_train_seq.shape[2]
output_seq_len = y_train_seq.shape[1]
n_households = len(le.classes_)

model = build_lstm_model(input_seq_len, feature_dim, output_seq_len,
n_households, use_bilstm=False)
model.summary()

from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
from tensorflow.keras.models import load_model

# Callbacks
checkpoint_cb = ModelCheckpoint(
    filepath='best_model.keras',
    monitor='val_loss',
    save_best_only=True,
    verbose=1
)

early_stopping_cb = EarlyStopping(
    monitor='val_loss',
    patience=5,
    restore_best_weights=True,
    verbose=1
)

# Fit model
history = model.fit(
    x=[X_train_seq, house_ids_train],
    y=y_train_seq,
    validation_data=([X_test_seq, house_ids_test], y_test_seq),
    epochs=20,
    batch_size=64,
    callbacks=[checkpoint_cb, early_stopping_cb]
)

# Load best model
best_model = load_model('best_model.keras')

from sklearn.metrics import mean_absolute_error, mean_squared_error

# Dự báo
y_pred = best_model.predict([X_test_seq, house_ids_test])

# Đánh giá
```

```
y_true_flat = y_test_seq.flatten()
y_pred_flat = y_pred.flatten()

mae = mean_absolute_error(y_true_flat, y_pred_flat)
mse = mean_squared_error(y_true_flat, y_pred_flat)
rmse = np.sqrt(mse)

def mean_absolute_percentage_error(y_true, y_pred):
    mask = y_true != 0
    return np.mean(np.abs((y_true[mask] - y_pred[mask]) /
y_true[mask])) * 100

mape = mean_absolute_percentage_error(y_true_flat, y_pred_flat)

print(f"MAE: {mae:.4f}")
print(f"RMSE: {rmse:.4f}")
print(f"MAPE: {mape:.2f}%")

import matplotlib.pyplot as plt

for i in range(3):
    plt.figure(figsize=(10, 4))
    plt.plot(y_test_seq[i], label='True')
    plt.plot(y_pred[i], label='Predicted')
    plt.title(f"Hộ thứ {i}")
    plt.xlabel("Thời gian")
    plt.ylabel("Điện năng tiêu thụ trong 7 ngày(kWh)")
    plt.legend()
    plt.show()
```

PHỤ LỤC 4
Xây Dựng chương trình web

```
import streamlit as st
import pandas as pd
import numpy as np
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import plotly.express as px
from sklearn.preprocessing import MinMaxScaler
from neuralforecast import NeuralForecast
from neuralforecast.models import NHITS
import os
import datetime

# ============ BẢO MẬT ĐĂNG NHẬP ============ #
import hashlib

# Danh sách tài khoản và mật khẩu
USERS = {
    "admin": "admin123",
    "user1": "pass1"
}

def hash_password(password):
    return hashlib.sha256(password.encode()).hexdigest()

def login():
    st.markdown("## 🔐 Đăng nhập hệ thống")
    username = st.text_input("Tài khoản")
    password = st.text_input("Mật khẩu", type="password")
    if st.button("Đăng nhập"):
        if username in USERS and hash_password(password) ==
hash_password(USERS[username]):
            st.success("✅ Đăng nhập thành công!")
            st.session_state["logged_in"] = True
            st.session_state["username"] = username
            st.rerun()
        else:
            st.error("❌ Tài khoản hoặc mật khẩu không đúng.")

if "logged_in" not in st.session_state:
    st.session_state["logged_in"] = False

if not st.session_state["logged_in"]:
    login()
    st.stop()  # Dừng ứng dụng nếu chưa đăng nhập
```

```
# ========================================== #


# Cấu hình trang
st.set_page_config(
    page_title="Electric Power Forecasting & Anomaly Detection",
    page_icon="⚡",
    layout="wide",
    initial_sidebar_state="expanded",
)

# Thiết lập theme và CSS nâng cao
st.markdown("""
    <style>
    /* Màu sắc chính và phụ */
    :root {
        --primary-color: #1E3A8A;
        --primary-light: #3B82F6;
        --primary-dark: #1E40AF;
        --secondary-color: #10B981;
        --accent-color: #F59E0B;
        --text-color: #1F2937;
        --text-light: #6B7280;
        --bg-light: #F9FAFB;
        --bg-card: #FFFFFF;
        --anomaly-high: #EF4444;
        --anomaly-high-bg: #FEE2E2;
        --anomaly-normal: #10B981;
        --anomaly-normal-bg: #ECFDF5;
    }

    /* Thiết lập chung */
    .block-container {
        padding-top: 1rem;
        padding-bottom: 1rem;
        max-width: 1200px;
        margin: 0 auto;
    }

    /* Header và tiêu đề */
    .main-header {
        background: linear-gradient(90deg, var(--primary-color), var(--
primary-dark));
        color: white;
        padding: 1.5rem;
        border-radius: 10px;
        margin-bottom: 2rem;
```

```
        box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    }

    .main-header h1 {
        margin: 0;
        font-weight: 800;
        font-size: 2.2rem;
    }

    .main-header p {
        margin-top: 0.5rem;
        opacity: 0.9;
        font-size: 1.1rem;
    }

    /* Sidebar */
    .css-1d391kg, .css-1v3fvcr {
        background-color: var(--bg-light);
    }

    /* Nút */
    .stButton>button {
        background: linear-gradient(90deg, var(--primary-color), var(--
primary-dark));
        color: white;
        border-radius: 6px;
        padding: 0.6rem 1.2rem;
        font-weight: 600;
        border: none;
        transition: all 0.3s ease;
        box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    }

    .stButton>button:hover {
        background: linear-gradient(90deg, var(--primary-dark), var(--
primary-color));
        transform: translateY(-2px);
        box-shadow: 0 4px 8px rgba(0, 0, 0, 0.15);
    }

    /* Card metrics */
    .metric-container {
        display: flex;
        flex-wrap: wrap;
        gap: 1rem;
        margin-bottom: 1.5rem;
    }
```

```
    .metric-card {
        background-color: var(--bg-card);
        border-radius: 8px;
        box-shadow: 0 4px 6px rgba(0, 0, 0, 0.05);
        padding: 1.2rem;
        text-align: center;
        flex: 1;
        min-width: 200px;
        transition: transform 0.3s ease, box-shadow 0.3s ease;
        border-top: 4px solid var(--primary-color);
    }

    .metric-card:hover {
        transform: translateY(-5px);
        box-shadow: 0 8px 15px rgba(0, 0, 0, 0.1);
    }

    .metric-value {
        font-size: 2.2rem;
        font-weight: 700;
        color: var(--primary-color);
        margin-bottom: 0.3rem;
        line-height: 1.2;
    }

    .metric-label {
        font-size: 1rem;
        color: var(--text-light);
        margin-top: 0;
    }

    /* Badges */
    .badge {
        padding: 0.25rem 0.75rem;
        border-radius: 9999px;
        font-weight: 600;
        font-size: 0.85rem;
        display: inline-block;
    }

    .anomaly-high {
        background-color: var(--anomaly-high-bg);
        color: var(--anomaly-high);
    }

    .anomaly-normal {
```

```
        background-color: var(--anomaly-normal-bg);
        color: var(--anomaly-normal);
    }

    /* Slider */
    .stSlider {
        padding-top: 1rem;
        padding-bottom: 1.5rem;
    }

    .stSlider > div > div {
        background-color: var(--primary-light) !important;
    }

    /* Tabs */
    .stTabs [data-baseweb="tab-list"] {
        gap: 2px;
    }

    .stTabs [data-baseweb="tab"] {
        background-color: #f1f5f9;
        border-radius: 4px 4px 0 0;
        padding: 10px 20px;
        border: none;
    }

    .stTabs [aria-selected="true"] {
        background-color: white !important;
        border-top: 3px solid var(--primary-color) !important;
    }

    /* Expander */
    .streamlit-expanderHeader {
        font-weight: 600;
        color: var(--primary-color);
    }

    /* Tooltip */
    .tooltip {
        position: relative;
        display: inline-block;
        cursor: help;
    }

    .tooltip .tooltiptext {
        visibility: hidden;
        width: 200px;
```

```
        background-color: #333;
        color: #fff;
        text-align: center;
        border-radius: 6px;
        padding: 5px;
        position: absolute;
        z-index: 1;
        bottom: 125%;
        left: 50%;
        margin-left: -100px;
        opacity: 0;
        transition: opacity 0.3s;
    }

    .tooltip:hover .tooltiptext {
        visibility: visible;
        opacity: 1;
    }

    /* Responsive adjustments */
    @media (max-width: 768px) {
        .metric-card {
            min-width: 100%;
        }
    }
    </style>
    """, unsafe_allow_html=True)


# Hàm cache cho việc tải mô hình
@st.cache_resource
def load_model():
    nf = NeuralForecast.load(path='model_nhits')
    return nf


# Hàm cache cho việc tải và tiền xử lý dữ liệu
# @st.cache_data
# def load_and_preprocess(train_path, test_path):
#     train_df = pd.read_csv(train_path,
parse_dates=['NGAYGIO_rounded'])
#     test_df = pd.read_csv(test_path, parse_dates=['NGAYGIO_rounded'])

#     for df in (train_df, test_df):
#         df.rename(columns={
#             'SO_CTO': 'unique_id',
#             'NGAYGIO_rounded': 'ds',
```

```
#             'tv delta': 'y'
#         }, inplace=True)
#         if df['ds'].dt.tz is not None:
#             df['ds'] = df['ds'].dt.tz_convert(None)
#        # df.sort_values(['unique_id', 'ds'], inplace=True)
#         # df.reset_index(drop=True, inplace=True)

#     scalers = {}
#     train_scaled, test_scaled = [], []
#     for uid, grp in train_df.groupby('unique_id'):
#         g_train = grp.copy()
#         scaler = MinMaxScaler()
#         g_train['y'] = scaler.fit_transform(g_train[['y']])
#         scalers[uid] = scaler
#         train_scaled.append(g_train)

#         g_test = test_df[test_df['unique_id'] == uid].copy()
#         if not g_test.empty:
#             g_test['y'] = scaler.transform(g_test[['y']])
#             test_scaled.append(g_test)

#     train_scaled_df = pd.concat(train_scaled).reset_index(drop=True)
#     test_scaled_df = pd.concat(test_scaled).reset_index(drop=True)
#     return train_scaled_df, test_scaled_df, scalers


@st.cache_data
def load_and_preprocess(train_path, test_path):
    train_df = pd.read_csv(train_path, parse_dates=['NGAYGIO_rounded'])
    test_df = pd.read_csv(test_path, parse_dates=['NGAYGIO_rounded'])

    for df in (train_df, test_df):
        df.rename(columns={
            'SO_CTO': 'unique_id',
            'NGAYGIO_rounded': 'ds',
            'tv delta': 'y'
        }, inplace=True)
        if df['ds'].dt.tz is not None:
            df['ds'] = df['ds'].dt.tz_convert(None)

        # Thêm cột đánh dấu thứ tự ban đầu
        df['order'] = range(len(df))

    scalers = {}
    train_scaled, test_scaled = [], []

    # Scale theo từng unique_id
```

```
    for uid, grp in train_df.groupby('unique_id'):
        g_train = grp.copy()
        scaler = MinMaxScaler()
        g_train['y'] = scaler.fit_transform(g_train[['y']])
        scalers[uid] = scaler
        train_scaled.append(g_train)

        g_test = test_df[test_df['unique_id'] == uid].copy()
        if not g_test.empty:
            g_test['y'] = scaler.transform(g_test[['y']])
            test_scaled.append(g_test)

    # Nối lại và sắp xếp theo thứ tự ban đầu
    train_scaled_df =
pd.concat(train_scaled).sort_values('order').reset_index(drop=True)
    test_scaled_df =
pd.concat(test_scaled).sort_values('order').reset_index(drop=True)

    # Xóa cột order nếu không cần dùng nữa
    train_scaled_df.drop(columns=['order'], inplace=True)
    test_scaled_df.drop(columns=['order'], inplace=True)

    return train_scaled_df, test_scaled_df, scalers



# Hàm lấy lịch sử dữ liệu
def get_history(train_df, test_df, unique_id, start_date, input_len):
    train_uid = train_df[train_df['unique_id'] ==
unique_id].sort_values('ds')
    test_uid = test_df[test_df['unique_id'] ==
unique_id].sort_values('ds')
    full = pd.concat([train_uid,
test_uid]).sort_values('ds').reset_index(drop=True)
    if full['ds'].dt.tz is not None:
        full['ds'] = full['ds'].dt.tz_convert(None)
    idxs = full.index[full['ds'] == pd.to_datetime(start_date)]
    if len(idxs) == 0 or idxs[0] < input_len:
        return None
    idx = idxs[0]
    return full.iloc[idx - input_len:idx].copy()


# Hàm dự đoán cho một ID duy nhất
def predict_single_uid(model, history, uid):
    df_input = history.copy()
    df_input['unique_id'] = uid
```

```
    pred = model.predict(df=df_input)
    return pred


# Hàm vẽ biểu đồ dự báo với phát hiện bất thường
def plot_forecast(pred_df, scaler, test_df, start_date,
anomaly_threshold=3.0):
    # Kết hợp dữ liệu dự báo và thực tế
    merged = pd.merge(test_df, pred_df, on=['unique_id', 'ds'],
how='inner')
    if merged.empty:
        st.warning("Không có dữ liệu khả dụng để hiển thị.")
        return None, None, None

    # Chuyển đổi giá trị về thang đo ban đầu
    merged['y_true'] = scaler.inverse_transform(merged[['y']])
    merged['y_pred'] = scaler.inverse_transform(merged[['NHITS']])

    # Tính toán sai số và xác định bất thường
    merged['error'] = np.abs(merged['y_true'] - merged['y_pred'])
    threshold = merged['error'].mean() + anomaly_threshold *
merged['error'].std()
    merged['is_anomaly'] = merged['error'] > threshold

    # Tạo biểu đồ với subplot
    fig = make_subplots(
        rows=2,
        cols=1,
        shared_xaxes=True,
        vertical_spacing=0.1,
        row_heights=[0.7, 0.3],
        subplot_titles=("Dự báo & Phát hiện bất thường", "Sai số dự
báo")
    )

    # Thêm dữ liệu thực tế
    fig.add_trace(
        go.Scatter(
            x=merged['ds'],
            y=merged['y_true'].values,
            mode='lines+markers',
            name='Thực tế',
            line=dict(color='#3B82F6', width=2),
            marker=dict(size=6)
        ),
        row=1, col=1
    )
```

```
    # Thêm dữ liệu dự báo
    fig.add_trace(
        go.Scatter(
            x=merged['ds'],
            y=merged['y_pred'].values,
            mode='lines',
            name='Dự báo',
            line=dict(color='#F59E0B', width=2)
        ),
        row=1, col=1
    )

    # Thêm điểm bất thường
    if any(merged['is_anomaly']):
        fig.add_trace(
            go.Scatter(
                x=merged[merged['is_anomaly']]['ds'],
                y=merged[merged['is_anomaly']]['y_true'].values,
                mode='markers',
                name='Bất thường',
                marker=dict(color='#EF4444', size=10, symbol='circle-
open', line=dict(width=2))
            ),
            row=1, col=1
        )

    # Thêm đường phân cách tại thời điểm bắt đầu dự báo
    start_date_pd = pd.to_datetime(start_date)
    fig.add_vline(
        x=start_date_pd,
        line_width=2,
        line_dash="dash",
        line_color="#6B7280",
        row=1, col=1
    )

    # Thêm chú thích cho đường phân cách
    max_y_value = merged['y_true'].max()
    fig.add_annotation(
        x=start_date_pd,
        y=max_y_value * 1.05,
        text="Bắt đầu dự báo",
        showarrow=True,
        arrowhead=1,
        ax=-40,
        ay=-30,
```

```
        row=1, col=1
    )

    # Thêm biểu đồ sai số
    fig.add_trace(
        go.Bar(
            x=merged['ds'],
            y=merged['error'].values,
            name='Sai số',
            marker=dict(
                color=merged['is_anomaly'].map({True: '#EF4444', False:
'#94A3B8'}),
                line=dict(width=0)
            )
        ),
        row=2, col=1
    )

    # Thêm đường ngưỡng bất thường
    fig.add_trace(
        go.Scatter(
            x=[merged['ds'].min(), merged['ds'].max()],
            y=[threshold, threshold],
            mode='lines',
            name=f'Ngưỡng bất thường ({anomaly_threshold}σ)',
            line=dict(color='#EF4444', width=2, dash='dash')
        ),
        row=2, col=1
    )

    # Cập nhật layout
    fig.update_layout(
        title=dict(
            text="Dự báo tiêu thụ điện & Phát hiện bất thường",
            font=dict(size=20, color='#1E3A8A')
        ),
        legend=dict(
            orientation="h",
            yanchor="bottom",
            y=1.02,
            xanchor="center",
            x=0.5,
            bgcolor='rgba(255,255,255,0.8)',
            bordercolor='rgba(0,0,0,0.1)',
            borderwidth=1
        ),
        hovermode="x unified",
```

```
        height=700,
        template="plotly_white",
        margin=dict(l=10, r=10, t=80, b=10),
    )

    # Cập nhật trục x và y
    fig.update_xaxes(
        title_text="Thời gian",
        showgrid=True,
        gridwidth=1,
        gridcolor='rgba(0,0,0,0.1)',
        row=1, col=1
    )

    fig.update_yaxes(
        title_text="Tiêu thụ điện (kWh)",
        showgrid=True,
        gridwidth=1,
        gridcolor='rgba(0,0,0,0.1)',
        row=1, col=1
    )

    fig.update_xaxes(
        title_text="Thời gian",
        showgrid=True,
        gridwidth=1,
        gridcolor='rgba(0,0,0,0.1)',
        row=2, col=1
    )

    fig.update_yaxes(
        title_text="Sai số (kWh)",
        showgrid=True,
        gridwidth=1,
        gridcolor='rgba(0,0,0,0.1)',
        row=2, col=1
    )

    # Tính toán các chỉ số thống kê
    stats = {
        'total_points': len(merged),
        'anomaly_count': merged['is_anomaly'].sum(),
        'anomaly_percent': (merged['is_anomaly'].sum() / len(merged)) *
100,
        'mean_error': merged['error'].mean(),
        'max_error': merged['error'].max(),
        'threshold': threshold
```

```
    }

    return fig, merged, stats


# Đường dẫn dữ liệu
train_path = 'data/train.csv'
test_path = 'data/test.csv'
input_len = 56
forecast_len = 28

# Header chính
st.markdown("""
<div class="main-header">
    <h1>⚡ Dự báo tiêu thụ điện & Phát hiện bất thường</h1>
    <p>Hệ thống dự báo và phát hiện bất thường trong tiêu thụ điện năng
dựa trên mô hình học sâu</p>
</div>
""", unsafe_allow_html=True)

# Tải dữ liệu và mô hình
train_df, test_df, scalers = load_and_preprocess(train_path, test_path)
model = load_model()

# Sidebar
with st.sidebar:
    st.image("image.png", width=300)

    st.markdown("### ⚙️ Tham số đầu vào")

    # Chọn công tơ
    unique_ids =
test_df['unique_id'].drop_duplicates(keep='first').tolist()
    uid = st.selectbox("Chọn công tơ (unique_id):", unique_ids)

    # unique_ids = test_df['unique_id'].unique()
    # uid = st.selectbox("Chọn công tơ (unique_id):", unique_ids)

    # Chọn thời điểm bắt đầu dự báo
    start_dates = test_df[test_df['unique_id'] ==
uid]['ds'].dt.strftime('%Y-%m-%d %H:%M:%S').tolist()
    start_date_str = st.selectbox("Chọn thời điểm bắt đầu dự báo:",
start_dates)
    start_date = pd.to_datetime(start_date_str)

    # Thêm slider điều chỉnh ngưỡng phát hiện bất thường
    st.markdown("### 🔍 Phát hiện bất thường")
```

```
    anomaly_threshold = st.sidebar.slider(
        "Ngưỡng phát hiện bất thường (số lần độ lệch chuẩn)",
        1.0, 5.0, 3.0, 0.1
    )

    st.markdown("""
    <div style="margin-top: 10px; font-size: 0.85rem; color: #6B7280;">
        <span style="font-weight: 600;">Lưu ý:</span> Giá trị thấp hơn
sẽ phát hiện nhiều bất thường hơn nhưng có thể tăng cảnh báo sai.
    </div>
    """, unsafe_allow_html=True)

    # Nút dự báo
    forecast_button = st.button("🔍 Dự báo & Phát hiện bất thường",
use_container_width=True)

    st.divider()
    st.markdown("### ℹ️ Thông tin")
    st.info(f"Mô hình sử dụng {input_len} điểm dữ liệu lịch sử")
    st.info(f"Độ dài dự báo: {forecast_len} bước")
    st.info("Dự báo 7 ngày tiếp theo từ thời điểm bắt đầu")


# Khu vực chính
if forecast_button:
    with st.spinner("Đang chạy dự báo..."):
        # Lấy dữ liệu lịch sử
        history = get_history(train_df, test_df, uid, start_date,
input_len)

        if history is None:
            st.error("Không đủ dữ liệu lịch sử để dự báo.")
        else:
            try:
                # Dự báo
                pred = predict_single_uid(model, history, uid)

                # Vẽ biểu đồ và tính toán thống kê
                fig, result_df, stats = plot_forecast(
                    pred,
                    scalers[uid],
                    test_df[test_df['unique_id'] == uid],
                    start_date,
                    anomaly_threshold
                )

                if fig is not None:
```

```
                    # Hiển thị các chỉ số thống kê
                    col1, col2, col3, col4 = st.columns(4)

                    with col1:
                        st.markdown(f"""
                        <div class="metric-card">
                            <div class="metric-
value">{stats['total_points']}</div>
                            <div class="metric-label">Tổng số điểm dữ
liệu</div>
                        </div>
                        """, unsafe_allow_html=True)

                    with col2:
                        st.markdown(f"""
                        <div class="metric-card">
                            <div class="metric-
value">{stats['anomaly_count']}</div>
                            <div class="metric-label">Số điểm bất
thường</div>
                        </div>
                        """, unsafe_allow_html=True)

                    with col3:
                        st.markdown(f"""
                        <div class="metric-card">
                            <div class="metric-
value">{stats['anomaly_percent']:.1f}%</div>
                            <div class="metric-label">Tỷ lệ bất
thường</div>
                        </div>
                        """, unsafe_allow_html=True)

                    with col4:
                        st.markdown(f"""
                        <div class="metric-card">
                            <div class="metric-
value">{stats['mean_error']:.2f}</div>
                            <div class="metric-label">Sai số trung bình
(kWh)</div>
                        </div>
                        """, unsafe_allow_html=True)

                    # Hiển thị biểu đồ
                    st.plotly_chart(fig, use_container_width=True)

                    # Tabs cho thông tin chi tiết
```

```
                    tab1, tab2, tab3 = st.tabs(["📊 Dữ liệu chi tiết",
"📈 Phân tích bất thường", "ℹ️ Thông tin mô hình"])

                    with tab1:
                        # Hiển thị dữ liệu chi tiết
                        if result_df is not None:
                            # Định dạng dữ liệu
                            display_df = result_df.copy()
                            display_df['ds'] =
display_df['ds'].dt.strftime('%Y-%m-%d %H:%M:%S')
                            display_df['Trạng thái'] = np.where(
                                display_df['is_anomaly'],
                                '⚠️ Bất thường',
                                '✅ Bình thường'
                            )

                            # Hiển thị bảng dữ liệu
                            st.dataframe(
                                display_df[['ds', 'y_true', 'y_pred',
'error', 'Trạng thái']].rename(
                                    columns={
                                        'ds': 'Thời gian',
                                        'y_true': 'Thực tế (kWh)',
                                        'y_pred': 'Dự báo (kWh)',
                                        'error': 'Sai số (kWh)'
                                    }
                                ),
                                use_container_width=True,
                                hide_index=True
                            )

                    with tab2:
                        if result_df is not None and
any(result_df['is_anomaly']):
                            # Phân tích bất thường
                            st.markdown("### Phân tích điểm bất
thường")

                            # Lọc các điểm bất thường
                            anomalies =
result_df[result_df['is_anomaly']].copy()
                            anomalies['ds'] =
anomalies['ds'].dt.strftime('%Y-%m-%d %H:%M:%S')
                            anomalies['error_percent'] =
(anomalies['error'] / anomalies['y_true']) * 100

                            # Hiển thị bảng bất thường
```

```
                            st.dataframe(
                                anomalies[['ds', 'y_true', 'y_pred',
'error', 'error_percent']].rename(
                                    columns={
                                        'ds': 'Thời gian',
                                        'y_true': 'Thực tế (kWh)',
                                        'y_pred': 'Dự báo (kWh)',
                                        'error': 'Sai số tuyệt đối
(kWh)',
                                        'error_percent': 'Sai số phần
trăm (%)'
                                    }
                                ),
                                use_container_width=True,
                                hide_index=True
                            )

                            # Biểu đồ phân bố sai số
                            st.markdown("### Phân bố sai số")
                            error_fig = px.histogram(
                                result_df,
                                x='error',
                                nbins=20,
                                color='is_anomaly',
                                color_discrete_map={True: '#EF4444',
False: '#94A3B8'},
                                labels={'error': 'Sai số (kWh)',
'count': 'Số lượng'},
                                title='Phân bố sai số dự báo'
                            )

                            error_fig.add_vline(
                                x=stats['threshold'],
                                line_width=2,
                                line_dash="dash",
                                line_color="#EF4444",
                                annotation_text=f"Ngưỡng:
{stats['threshold']:.2f}"
                            )

                            error_fig.update_layout(
                                showlegend=True,
                                legend_title_text='Bất thường',
                                legend=dict(
                                    orientation="h",
                                    yanchor="bottom",
                                    y=1.02,
```

```
                                    xanchor="right",
                                    x=1
                                )
                            )

                            st.plotly_chart(error_fig,
use_container_width=True)
                        else:
                            st.info("Không phát hiện bất thường nào
trong dữ liệu.")

                    with tab3:
                        # Thông tin mô hình
                        st.markdown("### Thông tin mô hình")
                        st.markdown("""
                        Mô hình dự báo sử dụng kiến trúc **NHITS**
(Neural Hierarchical Interpolation for Time Series Forecasting), một mô
hình học sâu hiện đại được thiết kế đặc biệt cho dự báo chuỗi thời
gian.

                        **Đặc điểm chính:**
                        - Kiến trúc phân cấp với nhiều khối nội suy
                        - Khả năng học các mẫu thời gian ở nhiều tần số
khác nhau
                        - Hiệu quả cao với dữ liệu tiêu thụ điện

                        **Phát hiện bất thường:**
                        - Sử dụng phương pháp dựa trên ngưỡng sai số
                        - Ngưỡng được tính bằng: Trung bình sai số +
(Độ lệch chuẩn sai số × Hệ số)
                        - Hệ số có thể điều chỉnh để thay đổi độ nhạy
của phát hiện
                        """)

                        # Hiển thị thông tin về độ chính xác
                        st.markdown("### Độ chính xác")
                        accuracy_data = {
                            'Metric': ['MAE', 'RMSE', 'MAPE'],
                            'Value': [f"{stats['mean_error']:.4f}",
f"{np.sqrt((result_df['error']**2).mean()):.4f}",
f"{(result_df['error'] / result_df['y_true']).mean() * 100:.2f}%"]
                        }

                        st.dataframe(pd.DataFrame(accuracy_data),
use_container_width=True, hide_index=True)
            except Exception as e:
                st.error(f"Lỗi khi chạy dự báo: {str(e)}")
```