BÁO CÁO CUỘC THI

AUTOSCALING ANALYSIS
DỰ ĐOÁN LƯU LƯỢNG HTTP VÀ TỰ ĐỘNG MỞ RỘNG MÁY CHỦ

CUỘC THI: CUỘC THI DATAFLOW 2026 - NĂM: 2026

[Tên thành viên 1] [MSSV 1] [Email 1]
[Tên thành viên 2] [MSSV 2] [Email 2]
[Tên thành viên 3] [MSSV 3] [Email 3]

BAN TỔ CHỨC: DATAFLOW 2026

Mục lục

Chương 1

GIỚI THIỆU BÀI TOÁN

1.1 GIỚI THIỆU

Trong bối cảnh điện toán đám mây phát triển mạnh mẽ, việc tối ưu hóa tài nguyên thông qua Autoscaling (tự động mở rộng quy mô) đã trở thành một yêu cầu cấp thiết. Do lưu lượng truy cập web thường xuyên biến động, các hệ thống luôn phải đối mặt với bài toán cân bằng giữa chi phí và hiệu năng: cấp phát thừa tài nguyên (over-provisioning) sẽ gây lãng phí ngân sách, trong khi cấp phát thiếu (under-provisioning) lại dẫn đến quá tải và gián đoạn dịch vụ. Để giải quyết vấn đề này, các chiến lược Autoscaling thường được chia thành hai nhóm chính là Reactive Scaling và Predictive Scaling. Trong khi Reactive Scaling chỉ phản ứng dựa trên các chỉ số hiện tại như CPU hay RAM, thì Predictive Scaling lại tận dụng kỹ thuật dự báo chuỗi thời gian (Time series forecasting) để phân tích dữ liệu lịch sử, nhận diện các chu kỳ (mùa vụ) và chủ động điều chỉnh tài nguyên trước khi lưu lượng thực tế tăng lên, giúp giảm thiểu độ trễ phản ứng của hệ thống.

1.2 BÀI TOÁN HỒI QUY: DỰ ĐOÁN LƯU LƯỢNG

Trọng tâm của bài toán hồi quy là xây dựng các mô hình máy học có khả năng dự báo chính xác hai chỉ số quan trọng: số lượng HTTP request và lượng dữ liệu (bytes) được chuyển tải trong tương lai. Để đảm bảo tính thực tiễn và linh hoạt, các mô hình này sẽ được huấn luyện và đánh giá trên ba khung thời gian khác nhau: cửa sổ 1 phút giúp hệ thống phản ứng nhanh với các biến động tức thời, cửa sổ 5 phút để cân bằng giữa độ nhạy và độ ổn định, và cửa sổ 15 phút phục vụ cho các chiến lược dài hạn. Dữ liệu huấn luyện được trích xuất từ logs của máy chủ NASA (giai đoạn 1/7 - 31/8/1995), bao gồm các thông tin chi tiết về host, thời gian, request và trạng thái phản hồi. Cần lưu ý rằng dữ liệu được phân chia theo trình tự thời gian nghiêm ngặt với tập Training từ 1/7 đến 22/8 và tập Test cho giai đoạn còn lại, đồng thời đã xử lý các khoảng downtime do sự cố thiên tai bằng dữ liệu giả lập.

Về phương pháp luận, dự án sẽ triển khai và so sánh hiệu quả giữa các nhóm kỹ thuật khác nhau, bao gồm các phương pháp thống kê truyền thống như ARIMA/SARIMA, các mô hình Deep Learning như LSTM/RNN và các thuật toán Machine Learning hiện đại như XGBoost hay LightGBM. Mô hình tối ưu nhất cho mỗi khung thời gian sẽ được lựa chọn dựa trên bộ chỉ số đánh giá toàn diện. Cụ thể, RMSE sẽ được dùng để phạt nặng các sai số lớn, trong khi MSE và MAE cung cấp cái nhìn tổng quát ít nhạy cảm với ngoại lai hơn, và MAPE sẽ giúp

đánh giá mức độ sai số dưới dạng phần trăm trực quan.

1.3 Bài toán tối ưu: Autoscaling

Bài toán tối ưu hóa đóng vai trò chuyển đổi các kết quả dự báo thành hành động cụ thể nhằm tìm ra chính sách autoscaling cân bằng nhất giữa việc giảm thiểu chi phí vận hành (tính theo server-hours) và đảm bảo cam kết chất lượng dịch vụ (SLA). Hệ thống sẽ triển khai ba chiến lược để so sánh: Scaling dựa trên CPU (điều chỉnh khi tải vượt ngưỡng 70% hoặc giảm dưới 30%), Scaling dựa trên số lượng request thực tế, và Predictive Scaling sử dụng kết quả từ mô hình học máy để đón đầu xu hướng. Để đảm bảo hệ thống hoạt động ổn định và tránh hiện tượng dao động liên tục (oscillation), các cơ chế như thời gian chờ (Cooldown Period) và ngưỡng trễ (Hysteresis) cũng được áp dụng chặt chẽ trong quá trình vận hành. Hiệu quả cuối cùng của các chính sách này sẽ được đo lường dựa trên tổng chi phí, tỷ lệ thời gian hệ thống không bị quá tải, tần suất thay đổi trạng thái máy chủ và độ trễ trong phản ứng.

1.4 Mục tiêu dự án

Dự án hướng tới việc xây dựng một quy trình khép kín từ phát triển mô hình đến triển khai hệ thống thực nghiệm.

Ở giai đoạn phát triển, mục tiêu là triển khai tối thiểu hai mô hình thuộc các nhóm thuật toán khác nhau cho

Chương 2

XỬ LÝ DỮ LIỆU

2.1 TỔNG QUAN BỘ DỮ LIỆU NASA HTTP LOGS

Bộ dữ liệu được sử dụng trong dự án là **NASA WWW Server Logs**, một tập dữ liệu kinh điển trong lĩnh vực phân tích lưu lượng mạng và chuỗi thời gian. Dữ liệu này ghi lại toàn bộ các request HTTP đến máy chủ WWW của NASA trong khoảng thời gian hai tháng, từ ngày 1/7/1995 đến ngày 31/8/1995, với độ phân giải thời gian là 1 giây. Tổng số lượng request trong bộ dữ liệu là khoảng 3.46 triệu dòng log, cung cấp một bức tranh toàn diện về mẫu hình truy cập web vào giữa thập niên 1990.

Mỗi dòng log trong bộ dữ liệu NASA được lưu trữ theo định dạng ASCII chuẩn, chứa các trường thông tin quan trọng cho việc phân tích chuỗi thời gian và autoscaling. Các trường này bao gồm: **Host** là địa chỉ IP hoặc tên miền của máy khách gửi request; **Timestamp** là thời điểm request được thực hiện với định dạng `[dd/Mon/YYYY:HH:MM:SS -0400]`; **Request** là chuỗi request HTTP bao gồm phương thức (GET, POST, v.v.), đường dẫn URL và phiên bản giao thức; **Status Code** là mã phản hồi HTTP (ví dụ: 200 cho thành công, 404 cho not found, 500 cho lỗi server); và **Bytes** là số byte dữ liệu được trả về cho request.

Ví dụ một dòng log điển hình có dạng sau:

```
199.72.81.55 - - [01/Jul/1995:00:00:01 -0400] "GET /history/apollo/ HTTP/1.0" 200 6245
```

Trong ví dụ này, `199.72.81.55` là địa chỉ IP của client, `[01/Jul/1995:00:00:01 -0400]` là thời gian request với timezone UTC-4 (Eastern Time), `"GET /history/apollo/ HTTP/1.0"` là request HTTP GET đến đường dẫn `/history/apollo/`, `200` là mã trạng thái HTTP 200 (Success), và `6245` là số byte dữ liệu được trả về.

Một điểm quan trọng cần lưu ý về bộ dữ liệu này là sự tồn tại của các khoảng thời gian *system downtime* do bão. Theo tài liệu chính thức của NASA, máy chủ đã gặp sự cố từ ngày 01/08/1995 14:52:01 đến ngày 03/08/1995 04:36:13 do ảnh hưởng của bão. Tuy nhiên, thay vì để lại các khoảng trống trong dữ liệu, các khoảng thời gian này đã được điền bằng dữ liệu giả lập (mock data) để duy trì tính liên tục của chuỗi thời gian. Ngoài ra, dự án cũng phát hiện thêm một khoảng thời gian có dấu hiệu bất thường từ ngày 28/07/1995 13:32:26 đến ngày 01/08/1995 00:00:00, trong đó số lượng request giảm đáng kể so với bình thường. Để đảm bảo chất lượng dữ liệu cho việc huấn luyện mô hình, các khoảng thời gian này được đánh dấu bằng cột `is_system_down` và các giá trị request được khôi phục bằng cách sử dụng dữ liệu từ tuần trước (shift 7 ngày).

2.2 Cấu trúc dữ liệu và Pipeline xử lý

2.2.1 Log Parser và Trích xuất trường

Pipeline xử lý dữ liệu bắt đầu bằng việc đọc và parse các file log thô từ thư mục `data/raw/`. Do dữ liệu được lưu trữ dưới dạng ASCII với cấu trúc cố định, chúng tôi sử dụng Regular Expression để trích xuất các trường thông tin một cách hiệu quả và chính xác. Biểu thức Regular Expression được sử dụng có dạng sau:

```
(?P<host>\S+) - - \[(?P<timestamp>.*?)\] "(?P<request>.*?)" (?P<status>\d{3}) (?P<bytes>
```

Hàm `parse_log_line` chịu trách nhiệm chuyển đổi mỗi dòng log thành một dictionary với các trường đã được trích xuất. Trong quá trình này, chúng tôi thực hiện các bước xử lý sau. Đầu tiên, trích xuất **Host** bằng cách lấy địa chỉ IP hoặc tên miền của client. Tiếp theo, parse **Timestamp** và chuyển đổi chuỗi timestamp sang định dạng datetime với timezone UTC-4 để đảm bảo tính chính xác về thời gian. Sau đó, trích xuất **Request** để lấy chuỗi request HTTP đầy đủ bao gồm phương thức, đường dẫn và phiên bản giao thức. Tiếp theo, parse **Status Code** và chuyển đổi mã trạng thái sang dạng integer để dễ dàng xử lý. Cuối cùng, parse **Bytes** và chuyển đổi số byte sang dạng integer, xử lý trường hợp giá trị là `'-'` (không có dữ liệu trả về) bằng cách gán giá trị 0.

Sau khi parse tất cả các dòng log, dữ liệu được chuyển đổi thành **pandas DataFrame** và sắp xếp theo thứ tự thời gian để đảm bảo tính liên tục của chuỗi thời gian. Điều này rất quan trọng vì các mô hình chuỗi thời gian yêu cầu dữ liệu được sắp xếp theo trình tự thời gian để học được các mẫu hình phụ thuộc vào quá khứ.

2.2.2 Aggregation theo thời gian

Do dữ liệu gốc có độ phân giải là 1 giây, chúng tôi thực hiện aggregation theo các cửa sổ thời gian khác nhau để phù hợp với các mục đích phân tích và mô hình hóa. Ba cửa sổ thời gian được sử dụng trong dự án là: cửa sổ **1-minute (1m)** với 89,280 điểm dữ liệu, phù hợp cho việc phản ứng nhanh với các biến động tức thời; cửa sổ **5-minute (5m)** với 17,856 điểm dữ liệu, cân bằng tốt giữa độ nhạy và độ ổn định; và cửa sổ **15-minute (15m)** với 5,952 điểm dữ liệu, phù hợp cho các chiến lược dài hạn và giảm nhiễu.

Hàm `resample_traffic` thực hiện aggregation với các bước sau. Đầu tiên, thiết lập cột `timestamp` làm index của DataFrame. Tiếp theo, sử dụng phương thức `resample` của pandas với các cửa sổ thời gian `'1min'`, `'5min'`, và `'15min'`. Sau đó, áp dụng các hàm aggregation cho từng trường: `requests` sử dụng `count` để đếm số lượng request trong mỗi cửa sổ; `bytes` sử dụng `sum` để tổng số byte được chuyển tải; `hosts` sử dụng `nunique` để đếm số lượng host duy nhất (đại diện cho số lượng người dùng); và `errors` sử dụng `lambda x: (x >= 400).sum()` để đếm số lượng lỗi (status code ≥ 400). Cuối cùng, **Fill missing values** bằng cách điền 0 cho các khoảng thời gian không có request để đảm bảo tính liên tục của chuỗi thời gian.

2.2.3 Xử lý missing values và anomalies

Sau khi aggregation, chúng tôi thực hiện các bước xử lý để đảm bảo chất lượng dữ liệu. Đầu tiên, xử lý **System Downtime** bằng cách tạo cột `is_system_down` để đánh dấu các khoảng

thời gian có dữ liệu giả lập. Các giá trị request trong khoảng này được khôi phục bằng cách sử dụng dữ liệu từ tuần trước (shift 2016 bước cho 5-minute window, tương ứng với 7 ngày). Sau đó, thực hiện **Forward-fill và Backward-fill** bằng cách sử dụng phương thức forward-fill (sử dụng giá trị trước đó) và backward-fill (sử dụng giá trị sau đó) để đảm bảo tính liên tục của chuỗi thời gian. Tiếp theo, loại bỏ các giá trị infinity có thể xuất hiện sau khi tính toán các đặc trưng thống kê. Cuối cùng, thực hiện **Type casting** bằng cách chuyển đổi các cột sang kiểu dữ liệu phù hợp (int cho requests, bytes, hosts; float cho các đặc trưng thống kê) để tối ưu hóa bộ nhớ và tốc độ xử lý.

2.3 PHÂN TÍCH KHÁM PHÁ DỮ LIỆU (EDA)

2.3.1 Thống kê tổng quan

Sau khi xử lý và làm sạch dữ liệu, chúng tôi thực hiện phân tích khám phá dữ liệu (Exploratory Data Analysis - EDA) để hiểu rõ các đặc điểm và mẫu hình của lưu lượng truy cập NASA. Thống kê cơ bản cho khung 5-minute cho thấy tổng số điểm dữ liệu là 17,856, khoảng thời gian từ 01/07/1995 00:00:00 đến 31/08/1995 23:55:00 (UTC-4). Trung bình số request mỗi 5 phút là khoảng 200-300 request. Peak traffic đạt mức cao nhất vào các giờ làm việc trong ngày (9h-17h) phản ánh hành vi người dùng điển hình vào giờ làm việc. Low traffic giảm đáng kể vào các giờ đêm (0h-6h) và cuối tuần, phản ánh sự khác biệt trong hành vi truy cập giữa ngày làm việc và thời gian nghỉ ngơi.

2.3.2 Phân phối Status Codes

Phân tích mã trạng thái HTTP cho thấy các đặc điểm sau. Mã **200 (Success)** chiếm phần lớn các request, cho thấy hệ thống hoạt động ổn định và trả về dữ liệu thành công cho hầu hết các request. Mã **304 (Not Modified)** xuất hiện thường xuyên do client sử dụng cache để giảm tải cho server. Mã **404 (Not Found)** có tỷ lệ thấp, cho thấy cấu trúc website được duy trì tốt và các đường dẫn không bị hỏng nhiều. Mã **500 (Server Error)** rất hiếm, cho thấy hệ thống có độ tin cậy cao và ít gặp sự cố nghiêm trọng.

Tỷ lệ lỗi được tính toán bằng công thức sau để đánh giá chất lượng hệ thống:

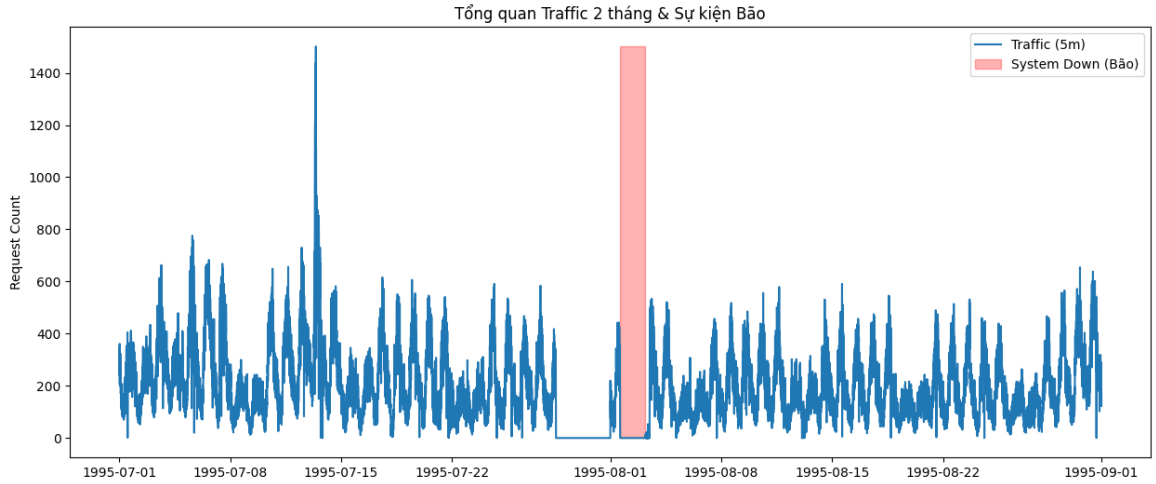
$$\text{error\_rate} = \frac{\text{errors}}{\text{requests}} \tag{2.1}$$

Trong đó `errors` là số lượng request có status code ≥ 400. Tỷ lệ lỗi trung bình trong toàn bộ bộ dữ liệu rất thấp, thường dưới 1%, cho thấy hệ thống hoạt động ổn định và đáng tin cậy.

2.3.3 Mẫu hình Traffic theo thời gian

Biểu đồ traffic theo thời gian (Hình 2.1) cho thấy các mẫu hình rõ ràng về chu kỳ hàng ngày và hàng tuần. Chu kỳ **hàng ngày** thể hiện rõ nét với traffic tăng mạnh vào các giờ làm việc (9h-17h) và giảm vào ban đêm (0h-6h), phản ánh hành vi người dùng điển hình trong ngày làm việc. Chu kỳ **hàng tuần** cho thấy traffic cao hơn vào các ngày trong tuần (Thứ 2 - Thứ 6) và thấp hơn vào cuối tuần (Thứ 7 - Chủ Nhật), phản ánh sự khác biệt trong hành vi truy cập giữa ngày làm việc và cuối tuần. Các **sự kiện bất thường** được đánh dấu bằng vùng màu

đỏ thể hiện các khoảng thời gian system downtime do bão, trong đó traffic giảm đáng kể do dữ liệu giả lập.

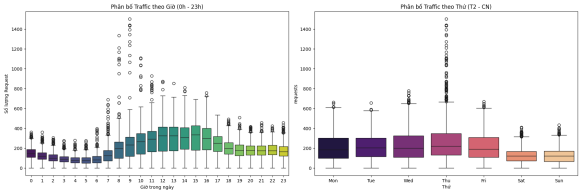


Hình 2.1: Tổng quan Traffic 2 tháng và Sự kiện Bão

2.3.4 Phân tích theo Giờ và Thứ

Biểu đồ phân bố traffic theo giờ trong ngày (Hình 2.2) cho thấy các đặc điểm sau. **Giờ cao điểm** từ 10h-16h có lượng request cao nhất, phản ánh thời gian làm việc chính thức. **Giờ thấp điểm** từ 0h-6h có lượng request thấp nhất, phản ánh thời gian nghỉ ngơi của người dùng. Độ biến động của traffic ở các giờ cao điểm lớn hơn, phản ánh sự không ổn định của traffic trong giờ làm việc.

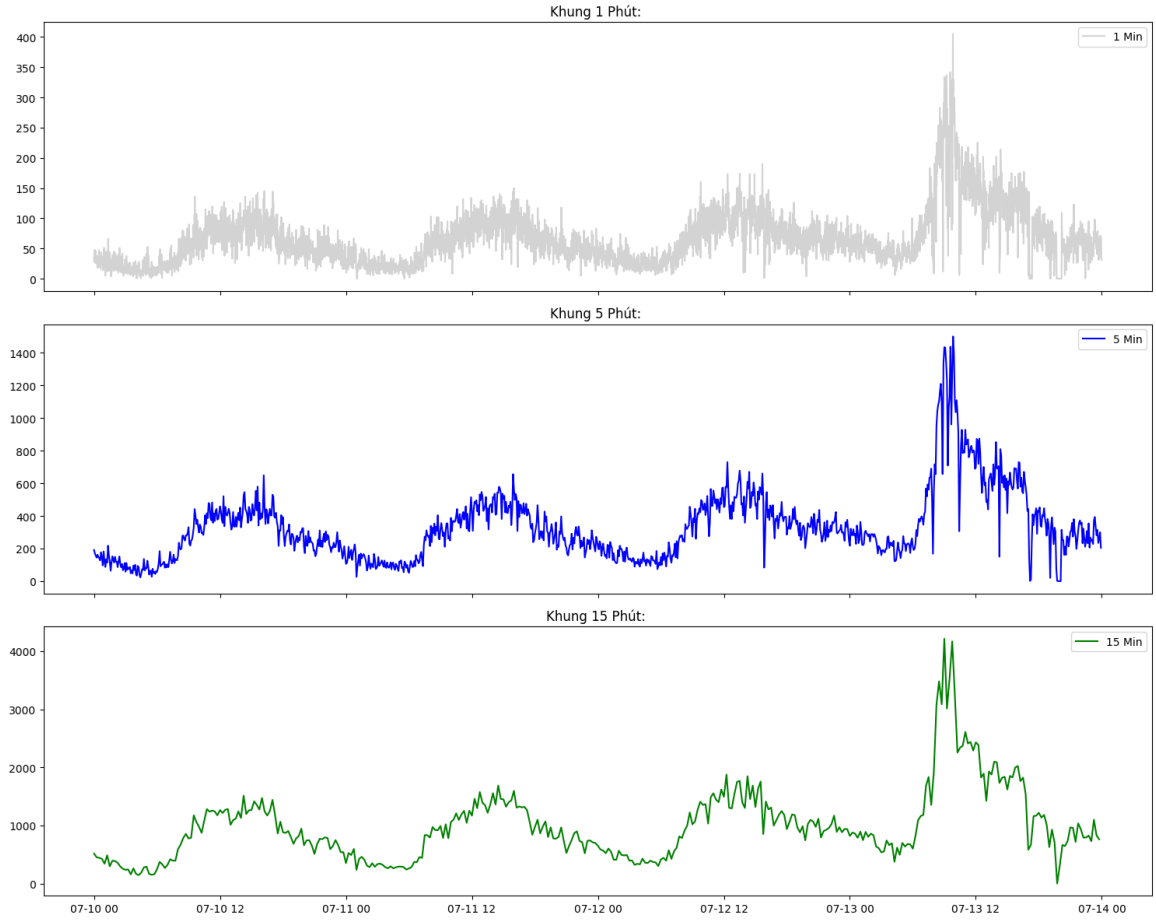
Biểu đồ phân bố traffic theo thứ trong tuần cho thấy các đặc điểm sau. **Ngày trong tuần** như Thứ 3, Thứ 4, Thứ 5 có traffic cao nhất, có thể do đây là những ngày làm việc bận rộn nhất trong tuần. **Cuối tuần** như Thứ 7 và Chủ Nhật có traffic thấp hơn đáng kể, phản ánh hành vi người dùng ít truy cập vào ngày nghỉ ngơi.



Hình 2.2: Phân bố Traffic theo Giờ và Thứ trong tuần

2.3.5 So sánh 3 khung thời gian

Biểu đồ so sánh traffic giữa 3 khung thời gian (Hình 2.3) cho thấy các đặc điểm khác nhau của mỗi cửa sổ. **Khung 1m** cho thấy chi tiết cao nhất nhưng có nhiều nhiễu, phù hợp cho việc phản ứng nhanh với các biến động tức thời nhưng khó dự báo do nhiễu lớn. **Khung 5m** cân bằng tốt giữa chi tiết và độ ổn định, giảm nhiễu đáng kể so với khung 1m nhưng vẫn giữ được các mẫu hình quan trọng. **Khung 15m** mượt mà hơn nhưng mất đi một số chi tiết quan trọng, phù hợp cho các chiến lược dài hạn và giảm nhiễu tối đa.



Hình 2.3: So sánh Traffic giữa 3 khung thời gian

2.4 FEATURE ENGINEERING

2.4.1 Time-based Features

Để nắm bắt các mẫu hình theo thời gian, chúng tôi tạo ra các đặc trưng dựa trên thời gian. **hour_of_day** là giờ trong ngày (0-23) phản ánh chu kỳ hàng ngày của traffic. **day_of_week** là thứ trong tuần (0-6, với 0 là Thứ 2) phản ánh chu kỳ hàng tuần của traffic. **is_weekend** là biến nhị phân (0/1) chỉ ra ngày cuối tuần, giúp phân biệt hành vi người dùng giữa ngày làm việc và cuối tuần.

2.4.2 Cyclical Encoding

Để mô hình hiểu được tính chu kỳ của thời gian (ví dụ: 23h và 0h gần nhau hơn 23h và 12h), chúng tôi sử dụng *Cyclical Encoding* với các hàm sin và cos. Cách tiếp cận này giúp mô hình học được các mẫu hình chu kỳ mà không bị nhầm lẫn bởi sự gián đoạn tại các điểm biên. Các công thức được sử dụng như sau:

$$\text{hour\_sin} = \sin\left(\frac{2\pi \times \text{hour\_of\_day}}{24}\right) \tag{2.2}$$

$$\text{hour\_cos} = \cos\left(\frac{2\pi \times \text{hour\_of\_day}}{24}\right) \tag{2.3}$$

2.4.3 Lag Features

Để nắm bắt tính autoregressive của chuỗi thời gian, chúng tôi tạo ra các đặc trưng lag. **req_lag_1** là giá trị request tại thời điểm $t - 1$, giúp capture các mẫu hình ngắn hạn và phản ứng nhanh với các biến động tức thời. **req_lag_12** là giá trị request tại thời điểm $t - 12$ (1 giờ trước cho khung 5m), giúp capture chu kỳ hàng ngày của traffic. **req_lag_288** là giá trị request tại thời điểm $t - 288$ (24 giờ trước cho khung 5m), giúp capture chu kỳ hàng tuần của traffic. Các đặc trưng lag này đặc biệt quan trọng cho các mô hình như ARIMA và LSTM, giúp mô hình học được các mẫu hình phụ thuộc vào quá khứ.

2.4.4 Rolling Statistics

Để nắm bắt xu hướng và độ biến động của traffic, chúng tôi tính toán các thống kê rolling. **rolling_mean_1h** là trung bình động trong 1 giờ (12 bước cho khung 5m), phản ánh xu hướng ngắn hạn của traffic. **rolling_std_1h** là độ lệch chuẩn động trong 1 giờ, phản ánh độ biến động ngắn hạn của traffic. **rolling_mean_24h** là trung bình động trong 24 giờ (288 bước cho khung 5m), phản ánh xu hướng dài hạn của traffic. Các đặc trưng này giúp mô hình nhận diện các giai đoạn tăng trưởng, giảm sút và ổn định của traffic.

2.5 AGGREGATION VÀ TRAIN/TEST SPLIT

2.5.1 Train/Test Split theo trình tự thời gian

Đối với bài toán chuỗi thời gian, việc split dữ liệu phải tuân thủ nguyên tắc *chronological split* - không được random shuffle vì sẽ làm mất tính liên tục của thời gian. Chúng tôi thực hiện split như sau: **Training set** từ 01/07/1995 đến 22/08/1995 (khoảng 52 ngày), chiếm khoảng 85% dữ liệu. **Test set** từ 23/08/1995 đến 31/08/1995 (khoảng 9 ngày), chiếm khoảng 15% dữ liệu. Tỷ lệ train/test là khoảng 85%/15%, phù hợp cho việc đánh giá khả năng dự báo của mô hình.

Việc split theo trình tự thời gian có những ưu điểm quan trọng. Đầu tiên, **Giả lập thực tế**: Mô hình được huấn luyện trên quá khứ và dự báo cho tương lai, giả lập đúng cách mà các hệ thống dự báo hoạt động trong thực tế. Thứ hai, **Tránh data leakage**: Không có thông tin từ tương lai "rò rỉ"vào tập training, đảm bảo tính công bằng của đánh giá mô hình. Tuy nhiên, việc split theo trình tự thời gian cũng có nhược điểm. Nếu mẫu hình traffic thay đổi đáng kể giữa train và test (concept drift), mô hình có thể hoạt động kém. Ngoài ra, nếu test set chứa các sự kiện đặc biệt không có trong train set, dự báo sẽ kém chính xác.

2.5.2 Lưu trữ dữ liệu

Sau khi xử lý hoàn tất, 3 bộ dữ liệu được lưu trữ trong thư mục `data/cleaned/` để sử dụng cho việc huấn luyện và đánh giá các mô hình. `data_1m.csv` là dữ liệu aggregation theo 1 phút với 89,280 điểm dữ liệu. `data_5m.csv` là dữ liệu aggregation theo 5 phút với 17,856 điểm dữ liệu. `data_15m.csv` là dữ liệu aggregation theo 15 phút với 5,952 điểm dữ liệu. Các file này được sử dụng trực tiếp cho việc huấn luyện và đánh giá các mô hình trong các chương tiếp theo.

Chương 3

MÔ HÌNH ARIMA

3.1 TỔNG QUAN MÔ HÌNH ARIMA

Mô hình ARIMA (AutoRegressive Integrated Moving Average) là một phương pháp thống kê kinh điển trong dự báo chuỗi thời gian, kết hợp ba thành phần chính để nắm bắt và dự báo các mẫu hình trong dữ liệu. Ba thành phần này bao gồm: **AR (AutoRegressive)** sử dụng các giá trị quá khứ để dự báo giá trị tương lai; **I (Integrated)** áp dụng phép sai phân (differencing) để biến đổi chuỗi thời gian thành chuỗi dừng (stationary); và **MA (Moving Average)** sử dụng các sai số dự báo quá khứ để cải thiện độ chính xác của dự báo.

Mô hình ARIMA được ký hiệu là `ARIMA(p,d,q)`, trong đó: `p` là bậc của thành phần tự hồi quy (số lượng giá trị quá khứ được sử dụng), `d` là số lần áp dụng sai phân để đạt được tính dừng, và `q` là bậc của thành phần trung bình động (số lượng sai số dự báo quá khứ được sử dụng). Công thức toán học của mô hình ARIMA có thể được biểu diễn như sau:

$$y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} \tag{3.1}$$

Trong đó: `y_t` là giá trị tại thời điểm `t`, `c` là hằng số, ϕ_i là các hệ số tự hồi quy, θ_j là các hệ số trung bình động, và ε_t là sai số trắng (white noise error).

Mô hình ARIMA đặc biệt phù hợp cho bài toán dự báo lưu lượng truy cập web vì: (1) dữ liệu chuỗi thời gian đơn biến (số lượng request) có tính dừng hoặc có thể biến đổi thành dừng thông qua sai phân; (2) có cấu trúc tự tương quan (autocorrelation) rõ rệt trong dữ liệu traffic; (3) các tham số có thể giải thích được, giúp hiểu rõ các yếu tố ảnh hưởng đến lưu lượng; và (4) phù hợp làm baseline trước khi thử nghiệm các mô hình phức tạp hơn như Prophet, LSTM, hay XGBoost.

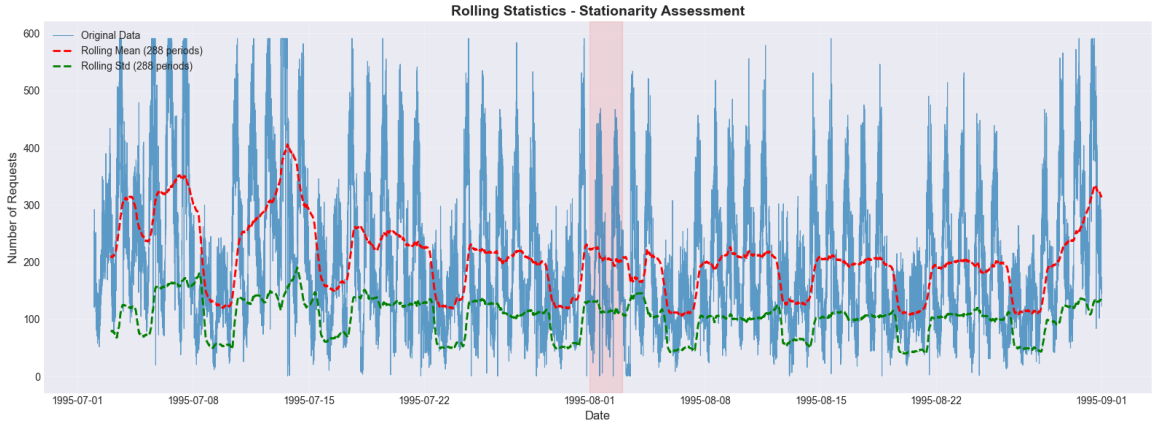
3.2 PHƯƠNG PHÁP HUẤN LUYỆN

Trước khi huấn luyện mô hình ARIMA, chúng tôi thực hiện các bước chuẩn bị dữ liệu và phân tích đặc điểm để đảm bảo dữ liệu đáp ứng các giả định của mô hình.

3.2.1 Đánh giá tính dừng (Stationarity Assessment)

Mô hình ARIMA yêu cầu chuỗi thời gian phải có tính dừng, nghĩa là: giá trị trung bình không thay đổi theo thời gian; phương sai không thay đổi theo thời gian; và tự tương quan giữa các giá trị tại các khoảng thời gian khác không thay đổi theo thời gian. Để đánh giá tính dừng, chúng tôi sử dụng hai phương pháp: kiểm tra trực quan bằng cách vẽ các thống kê rolling (rolling mean và rolling standard deviation) và kiểm tra thống kê Augmented Dickey-Fuller (ADF test).

Hình 3.1 cho thấy thống kê rolling của chuỗi thời gian số lượng request theo cửa sổ 5 phút. Rolling mean (đường nét đứt màu đỏ) và rolling standard deviation (đường nét đứt màu xanh) tương đối ổn định trong suốt giai đoạn huấn luyện, cho thấy chuỗi có tính dừng. Kết quả kiểm tra ADF cho thấy `p-value = 0.000000`, nhỏ hơn nhiều so với mức ý nghĩa 0.05, cho phép chúng tôi bác bỏ giả thuyết về unit root và kết luận chuỗi là dừng. Do đó, tham số `d = 0` (không cần áp dụng sai phân).

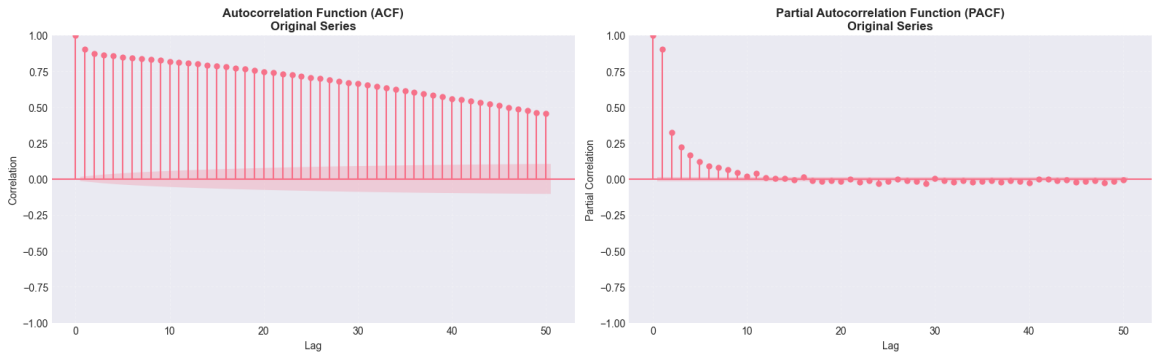


Hình 3.1: Rolling Statistics - Đánh giá tính dừng của chuỗi thời gian

3.2.2 Phân tích tự tương quan (ACF và PACF)

Sau khi xác định chuỗi là dừng, chúng tôi phân tích cấu trúc tự tương quan để xác định các tham số p và q cho mô hình ARIMA. **ACF (Autocorrelation Function)** đo lường tương quan giữa chuỗi thời gian và các giá trị lag của nó, trong khi **PACF (Partial Autocorrelation Function)** đo lường tương quan loại bỏ ảnh hưởng của các lag trung gian.

Hình 3.2 cho thấy đồ thị ACF và PACF của chuỗi thời gian. Đồ thị ACF (bên trái) cho thấy tương quan giảm rất chậm theo thời gian, biểu hiện tính persistence cao của chuỗi. Đồ thị PACF (bên phải) cho thấy có đỉnh rõ rệt tại lag 1 (tương quan rất mạnh) và đỉnh nhỏ hơn tại lag 2, sau đó các giá trị giảm vào vùng tin cậy (confidence interval). Dựa trên các mẫu hình này, chúng tôi đề xuất mô hình ban đầu là `ARIMA(1,0,0)` hoặc `ARIMA(2,0,0)`.



Hình 3.2: Đồ thị ACF và PACF cho phân tích tham số mô hình

3.2.3 Lựa chọn tham số tự động (Auto-ARIMA)

Thay vì chọn tham số `(p,d,q)` thủ công dựa trên phân tích ACF/PACF, chúng tôi sử dụng thuật toán `auto_arima` từ thư viện `pmdarima` để tìm kiếm tự động các tham số tối ưu. Thuật toán này tìm kiếm qua lưới các tổ hợp tham số và chọn mô hình có tiêu chuẩn thông tin Akaike (AIC) thấp nhất, cân bằng giữa độ phù hợp và độ phức tạp của mô hình.

Chúng tôi huấn luyện mô hình ARIMA trên cả ba cửa sổ thời gian khác nhau (1m, 5m, 15m) để so sánh hiệu quả dự báo theo các mức độ phân giải dữ liệu khác nhau. Kết quả lựa chọn tham số cho từng cửa sổ được tóm tắt trong Bảng 3.1: `ARIMA(5,0,1)` cho cửa sổ 1m với `AIC = 605677.82`; `ARIMA(2,0,1)` cho cửa sổ 5m với `AIC = 159089.09`; và `ARIMA(3,0,2)` cho cửa sổ 15m với `AIC = 62299.72`.

Bảng 3.1: Tham số ARIMA tối ưu cho ba cửa sổ thời gian

Cửa sổ	ARIMA(p,d,q)	AIC	BIC	Số lượng huấn luyện
1m	(5,0,1)	605677.82	605751.48	74,880
5m	(2,0,1)	159089.09	159127.16	14,976
15m	(3,0,2)	62299.72	62341.01	4,992

3.2.4 So sánh thủ công và tự động

Để đánh giá hiệu quả của việc lựa chọn tham số thủ công so với tự động, chúng tôi so sánh ba mô hình trên dữ liệu cửa sổ 5m: `ARIMA(1,0,0)` (thủ công dựa trên PACF), `ARIMA(2,0,0)` (thủ công dựa trên PACF), và `ARIMA(2,0,1)` (tự động). Kết quả cho thấy mô hình tự động có `AIC` thấp nhất (159089.09), thấp hơn đáng kể so với hai mô hình thủ công (162404.80 và 160806.62). Điều này cho thấy thuật toán tự động tìm kiếm được các tổ hợp tham số tốt hơn so với phân tích trực quan, đặc biệt là khả năng kết hợp cả thành phần AR và MA.

3.3 KẾT QUẢ CHO CỬA SỔ 5 PHÚT

Sau khi huấn luyện xong mô hình ARIMA cho cửa sổ 5 phút, chúng tôi đánh giá chi tiết hiệu quả dự báo trên tập kiểm tra (test set) từ 23/8/1995 đến 31/8/1995. Cửa sổ 5 phút được chọn để phân tích sâu vì nó đạt được sự cân bằng tốt nhất giữa độ chi tiết và độ ổn định trong ba cửa sổ thời gian được đánh giá.

3.3.1 Cấu trúc mô hình

Mô hình `ARIMA(2,0,1)` được huấn luyện trên 14,976 quan sát dữ liệu 5 phút. Các hệ số tự hồi quy là $\phi_1 = 1.228, \phi_2 = -0.234$ và hệ số trung bình động là $\theta_1 = -0.771$. Cấu trúc mô hình đơn giản hơn so với cửa sổ 1 phút, cho thấy dữ liệu 5 phút đã được làm mượt mà hơn nhờ quá trình aggregation.

Tất cả các tham số đều có ý nghĩa thống kê (p-value < 0.001), cho thấy mô hình phù hợp tốt với dữ liệu huấn luyện. Tham số $\phi_1 = 1.228$ cho thấy giá trị hiện tại phụ thuộc mạnh vào giá trị ngay trước đó, trong khi tham số $\phi_2 = -0.234$ cho thấy sự điều chỉnh từ giá trị hai bước trước đó. Tham số $\theta_1 = -0.771$ cho thấy mô hình sử dụng sai số dự báo quá khứ để điều chỉnh dự báo hiện tại.

3.3.2 Hiệu quả dự báo

Khi dự báo trên tập kiểm tra (2,592 quan sát), mô hình đạt được các chỉ số hiệu quả sau: `RMSE = 121.86`, `MAE = 99.20`, và `MAPE = 86.26%`. Cửa sổ 5 phút đạt được sự cân bằng tốt giữa độ chi tiết và độ ổn định: `RMSE` và `MAE` thấp hơn so với baseline (mean baseline), nhưng `MAPE` vẫn cao do ảnh hưởng của các giá trị nhỏ vào ban đêm.

Bảng 3.2 tóm tắt hiệu quả dự báo của mô hình ARIMA cho cửa sổ 5 phút.

Bảng 3.2: Hiệu quả dự báo ARIMA cho cửa sổ 5 phút

Cửa sổ	MSE	RMSE	MAE	MAPE (%)
5m	14849.92	121.86	99.20	86.26

3.4 MODEL DIAGNOSTICS

Sau khi huấn luyện xong mô hình, chúng tôi thực hiện các kiểm tra chẩn đoán để xác nhận mô hình có đáp ứng các giả định của ARIMA.

3.4.1 Kiểm tra Ljung-Box

Kiểm tra Ljung-Box được sử dụng để đánh giá xem các phần dư (residuals) của mô hình có tự tương quan hay không. Giả thuyết null là các phần dư không có tự tương quan (white noise). Nếu `p-value > 0.05`, chúng tôi không thể bác bỏ giả thuyết null, cho thấy các phần dư là white noise (tốt). Nếu `p-value` $\leq$ `0.05`, các phần dư có tự tương quan, cho thấy mô hình chưa nắm bắt được hết các mẫu hình trong dữ liệu.

Kết quả kiểm tra Ljung-Box cho cửa sổ 5 phút có `p-value = 0.082677` (> 0.05), cho thấy các phần dư là white noise (tốt). Điều này cho thấy mô hình đã nắm bắt được hầu hết các mẫu hình trong dữ liệu.

3.4.2 Kiểm tra Jarque-Bera

Kiểm tra Jarque-Bera được sử dụng để đánh giá xem các phần dư có phân phối chuẩn hay không. Giả thuyết null là các phần dư có phân phối chuẩn. Nếu `p-value > 0.05`, chúng tôi không thể bác bỏ giả thuyết null, cho thấy các phần dư có phân phối chuẩn (tốt). Nếu `p-value` $\leq$ `0.05`, các phần dư không có phân phối chuẩn.

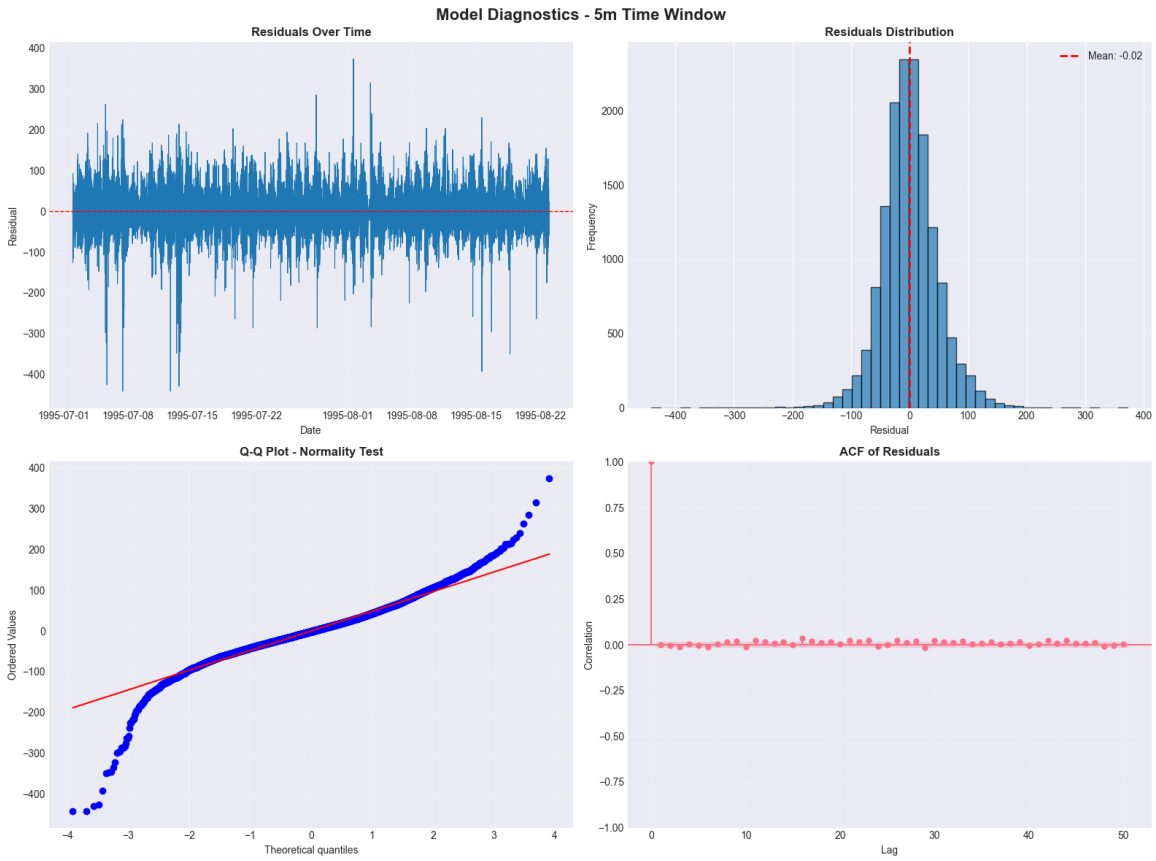
Kết quả kiểm tra Jarque-Bera cho cửa sổ 5 phút có `p-value = 0.000000`, nhỏ hơn nhiều so với mức ý nghĩa 0.05, cho thấy các phần dư không có phân phối chuẩn. Điều này cho thấy dữ liệu traffic có phân phối lệch (skewed) với nhiều giá trị outlier, đặc biệt vào giờ cao điểm.

3.4.3 Kiểm tra Durbin-Watson

Kiểm tra Durbin-Watson được sử dụng để đánh giá xem có tự tương quan bậc nhất trong các phần dư hay không. Thống kê Durbin-Watson có giá trị từ 0 đến 4, trong đó giá trị 2 cho thấy không có tự tương quan bậc nhất. Kết quả kiểm tra cho cửa sổ 5 phút có `DW statistic = 1.9967`, gần 2.0, cho thấy không có tự tương quan bậc nhất đáng kể trong các phần dư.

3.4.4 Đồ thị chẩn đoán mô hình

Hình 3.3 cho thấy đồ thị chẩn đoán mô hình cho cửa sổ 5 phút. Đồ thị bao gồm: (1) Residuals over time (phần trên trái) cho thấy các phần dư dao động quanh giá trị 0; (2) Residuals distribution (phần trên phải) cho thấy phân phối gần chuẩn với mean gần 0; (3) Q-Q plot (phần dưới trái) cho thấy các điểm gần như nằm trên đường chéo, cho thấy phân phối gần chuẩn; và (4) ACF of residuals (phần dưới phải) cho thấy không có cột nào vượt quá vùng tin cậy, cho thấy không có tự tương quan đáng kể trong các phần dư.



Hình 3.3: Chẩn đoán mô hình ARIMA cho cửa sổ 5 phút

3.5 PHÂN TÍCH VÀ ĐÁNH GIÁ

3.5.1 So sánh với baseline

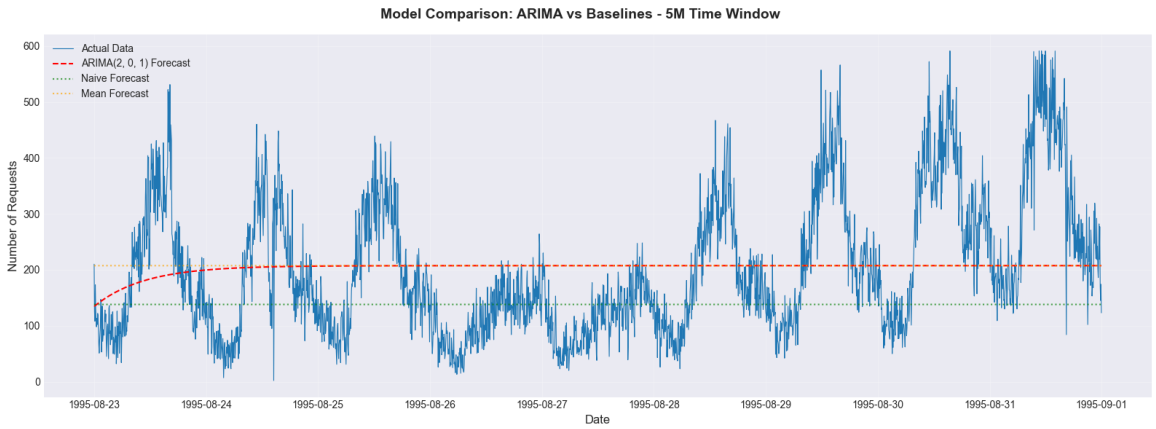
Để đánh giá giá trị thực tế của mô hình ARIMA, chúng tôi so sánh với hai mô hình baseline đơn giản: **Naive baseline** (dự báo bằng giá trị cuối cùng của tập huấn luyện) và **Mean baseline** (dự báo bằng giá trị trung bình của tập huấn luyện). Kết quả so sánh cho cửa sổ 5 phút được tóm tắt trong Bảng 3.3.

Bảng 3.3: So sánh mô hình ARIMA với các baseline cho cửa sổ 5 phút

Cửa sổ	Naive RMSE	Mean RMSE	ARIMA RMSE	Cải thiện (%)
5m	138.74	122.50	121.86	+0.5

Kết quả cho thấy mô hình ARIMA chỉ cải thiện nhẹ 0.5% so với baseline. Điều này cho thấy mô hình ARIMA gần như chỉ dự báo bằng giá trị trung bình, cho thấy giá trị gia tăng của mô hình là rất hạn chế.

Hình 3.4 cho thấy so sánh trực quan giữa dự báo của mô hình ARIMA (đường nét đứt màu đỏ), dự báo naive (đường nét đứt màu xanh lá), và dự báo mean (đường nét đứt màu cam). Mô hình ARIMA gần như trùng với dự báo mean, cho thấy giá trị gia tăng của mô hình là rất hạn chế.



Hình 3.4: So sánh mô hình ARIMA với các baseline cho cửa sổ 5 phút

3.5.2 Phân tích overfitting

Để đánh giá mức độ overfitting của mô hình, chúng tôi so sánh hiệu quả trên tập huấn luyện (in-sample) và tập kiểm tra (out-of-sample). Kết quả cho thấy mô hình có hiện tượng overfitting rõ rệt: hiệu quả trên tập kiểm tra kém hơn đáng kể so với tập huấn luyện.

Bảng 3.4: So sánh hiệu quả in-sample và out-of-sample cho cửa sổ 5 phút

Cửa sổ	In-sample RMSE	Out-of-sample RMSE	Tỷ lệ overfitting
5m	49.02	121.86	2.48x

Tỷ lệ overfitting được tính bằng tỷ lệ giữa `RMSE out-of-sample` và `RMSE in-sample`. Cửa sổ 5 phút có tỷ lệ 2.48x, cao hơn 2, cho thấy mô hình ARIMA overfitting nghiêm trọng, với hiệu quả trên tập kiểm tra kém hơn 2.5 lần so với tập huấn luyện.

Chương 4

MÔ HÌNH LSTM

4.1 TỔNG QUAN MÔ HÌNH LSTM

Mô hình LSTM (Long Short-Term Memory) là một biến thể nâng cao của mạng nơ-ron hồi quy truyền thống (RNN), được thiết kế đặc biệt để giải quyết vấn đề *vanishing gradient* trong quá trình huấn luyện các chuỗi thời gian dài. LSTM được giới thiệu lần đầu bởi Hochreiter và Schmidhuber năm 1997 và đã trở thành một trong những kiến trúc mạng nơ-ron thành công nhất cho các bài toán chuỗi thời gian, xử lý ngôn ngữ tự nhiên và nhận dạng giọng nói.

4.2 PHƯƠNG PHÁP HUẤN LUYỆN

4.2.1 Chuẩn bị dữ liệu

Để tối ưu hóa quá trình huấn luyện mô hình LSTM, công tác chuẩn bị dữ liệu được thực hiện chặt chẽ bắt đầu bằng việc lựa chọn đặc trưng dựa trên đánh giá của *Random Forest Regressor*, qua đó xác định 5 biến đầu vào thiết yếu gồm `requests_target`, `error_rate`, `hour_sin`, `hour_cos` (trọng số 0.450), và `is_weekend` (0.322). Dữ liệu sau đó được phân chia theo trình tự thời gian để mô phỏng chính xác kịch bản dự báo thực tế, bao gồm tập **Training** (02/07–16/08, chiếm $\sim$60%), **Validation** (17/08–22/08, $\sim$20%) và **Test** (23/08–31/08, $\sim$20%). Cuối cùng, chúng tôi áp dụng *MinMaxScaler* để chuẩn hóa các đặc trưng về khoảng $[0, 1]$ giúp mô hình hội tụ nhanh hơn, đồng thời tuân thủ nghiêm ngặt nguyên tắc chỉ khớp (fit) scaler trên tập training trước khi áp dụng cho các tập còn lại nhằm ngăn chặn hiện tượng rò rỉ dữ liệu (data leakage).

4.2.2 Sequence Creation

LSTM yêu cầu dữ liệu đầu vào dưới dạng chuỗi (sequence) để học được các mẫu hình phụ thuộc vào quá khứ. Chúng tôi tạo các chuỗi dữ liệu với độ dài `sequence_length = 12`, tương ứng với 60 phút dữ liệu lịch sử (12 bước thời gian cho cửa sổ 5 phút). Cụ thể, để dự báo số lượng request tại thời điểm `t`, mô hình sử dụng dữ liệu từ 12 bước thời gian trước đó (`t-12` đến `t-1`).

Quá trình tạo chuỗi được thực hiện như sau. Đầu tiên, chúng tôi tạo *sliding window* trên

dữ liệu đã được chuẩn hóa. Với mỗi quan sát tại thời điểm `t`, chúng tôi lấy 12 quan sát trước đó làm input và quan sát tại thời điểm `t` làm target. Sau đó, chúng tôi chuyển đổi dữ liệu thành *PyTorch Dataset* với shape `(batch_size, sequence_length, num_features)`. Cuối cùng, chúng tôi tạo *DataLoader* với batch size = 32 để huấn luyện mô hình hiệu quả.

4.2.3 Kiến trúc mạng

Mô hình LSTM được thiết kế với kiến trúc như sau: **Input Layer** nhận 5 đặc trưng (requests_-target, error_rate, hour_sin, hour_cos, is_weekend); **LSTM Layer** với 32 hidden units và 1 layer; và **Output Layer** (Fully Connected) chuyển đổi output của LSTM thành dự báo số lượng request.

Sau quá trình hyperparameter tuning với 7 cấu hình khác nhau (bao gồm các biến thể của small_model, medium_model, large_model, và deep_model với các training configs khác nhau như standard, aggressive, conservative và sequence lengths khác nhau như 12, 24), chúng tôi chọn cấu hình tối ưu nhất là `small_model` với: `input_size = 5`, `hidden_size = 32`, `num_layers = 1`, và `dropout = 0.1`. Mô hình này có tổng cộng 5,025 tham số.

Bảng 4.1 tóm tắt các tham số tối ưu cho mô hình LSTM.

Bảng 4.1: Tham số tối ưu cho mô hình LSTM

Tham số	Giá trị
Input size	5
Hidden size	32
Num layers	1
Dropout	0.1
Output size	1
Total parameters	5,025

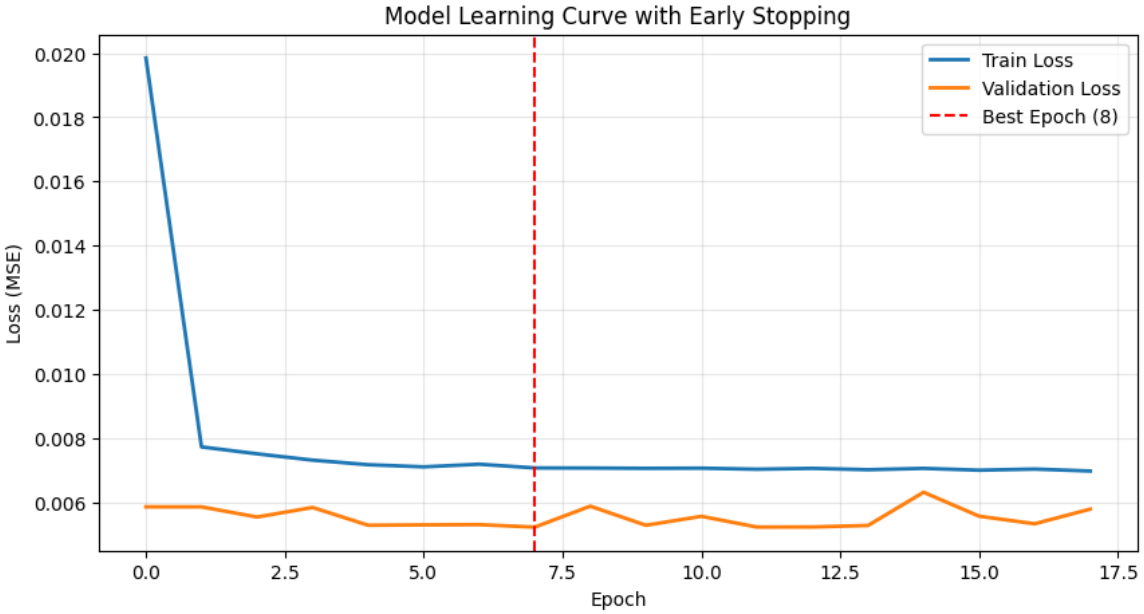
4.2.4 Quá trình huấn luyện

Quá trình huấn luyện mô hình LSTM được thực hiện với các siêu tham số sau: **Loss Function** là Mean Squared Error (MSE); **Optimizer** là Adam với learning rate = 0.001; **Batch size** = 32; **Epochs tối đa** = 50; và **Early Stopping** với patience = 10 epochs.

Trong quá trình huấn luyện, chúng tôi sử dụng *Early Stopping* trên validation set để tránh overfitting. Nếu validation loss không cải thiện trong 10 epochs liên tiếp, quá trình huấn luyện sẽ dừng và mô hình với validation loss tốt nhất sẽ được lưu lại. Cơ chế này giúp tiết kiệm thời gian huấn luyện và đảm bảo mô hình có khả năng tổng quát hóa tốt.

Mô hình đạt được **best epoch = 8** với **best validation loss = 0.005225**. Sau epoch 8, validation loss bắt đầu tăng nhẹ, cho thấy mô hình bắt đầu overfitting. Early stopping đã kích hoạt tại epoch 18, dừng quá trình huấn luyện và tải lại trọng số từ epoch 8.

Hình 4.1 cho thấy đường cong học của mô hình LSTM với train loss và validation loss theo từng epoch. Đường nét đứt màu đỏ đánh dấu epoch tốt nhất (epoch 8) khi validation loss đạt giá trị thấp nhất.



Hình 4.1: Đường cong học (Learning Curve) của mô hình LSTM với Early Stopping

4.3 KẾT QUẢ CHO CỬA SỔ 5 PHÚT

Sau khi huấn luyện xong mô hình LSTM tối ưu, chúng tôi đánh giá hiệu quả dự báo trên tập kiểm tra (test set) từ 23/8/1995 đến 31/8/1995. Cửa sổ 5 phút được chọn để phân tích sâu vì nó đạt được sự cân bằng tốt nhất giữa độ chi tiết và độ ổn định trong ba cửa sổ thời gian được đánh giá.

4.3.1 Cấu trúc mô hình

Mô hình LSTM tối ưu cho cửa sổ 5 phút có kiến trúc như sau: **LSTM Layer** với 32 hidden units và 1 layer nhận input shape `(batch_size, 12, 5)`; **Output Layer** chuyển đổi output của LSTM thành dự báo với shape `(batch_size, 1)`; và **Total Parameters** là 5,025 tham số.

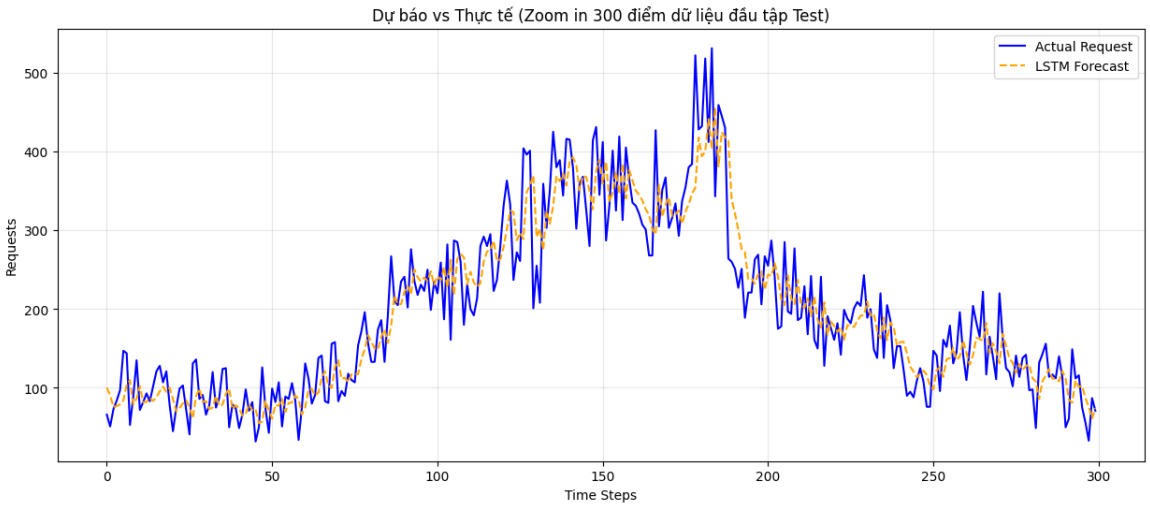
Kiến trúc đơn giản này cho thấy dữ liệu 5 phút đã được làm mượt mà hơn nhờ quá trình aggregation, giúp mô hình LSTM không cần quá nhiều tham số để nắm bắt các mẫu hình trong dữ liệu.

4.3.2 Hiệu quả dự báo

Khi dự báo trên tập kiểm tra (2,592 quan sát), mô hình LSTM đạt được các chỉ số hiệu quả sau: `RMSE = 44.59`, `MSE = 1988.35`, `MAE = 33.86`, và `MAPE = 26.91%`. So với mô hình ARIMA (RMSE = 121.86, MAE = 99.20, MAPE = 86.26%), LSTM cải thiện đáng kể về độ chính xác dự báo.

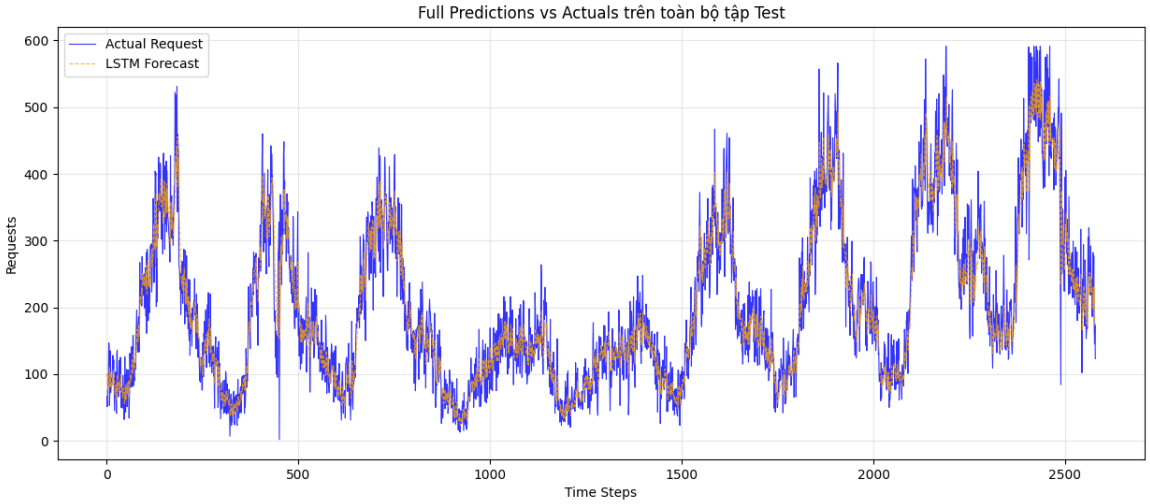
Bảng 4.2 tóm tắt hiệu quả dự báo của mô hình LSTM cho cửa sổ 5 phút.

Hình 4.2 cho thấy so sánh giữa dự báo của LSTM và giá trị thực tế trên 300 điểm dữ liệu đầu tiên của tập kiểm tra. Đường nét đứt màu cam là dự báo của LSTM, trong khi đường nét liền màu xanh là giá trị thực tế.



Hình 4.2: So sánh dự báo LSTM và giá trị thực tế (Zoom in 300 điểm dữ liệu đầu tập kiểm tra)

Hình 4.3 cho thấy so sánh giữa dự báo của LSTM và giá trị thực tế trên toàn bộ tập kiểm tra (2,592 điểm dữ liệu).



Hình 4.3: So sánh dự báo LSTM và giá trị thực tế trên toàn bộ tập kiểm tra

Bảng 4.2: Hiệu quả dự báo LSTM cho cửa sổ 5 phút

Cửa sổ	MSE	RMSE	MAE	MAPE (%)
5m	1988.35	44.59	33.86	26.91

4.3.3 So sánh với ARIMA

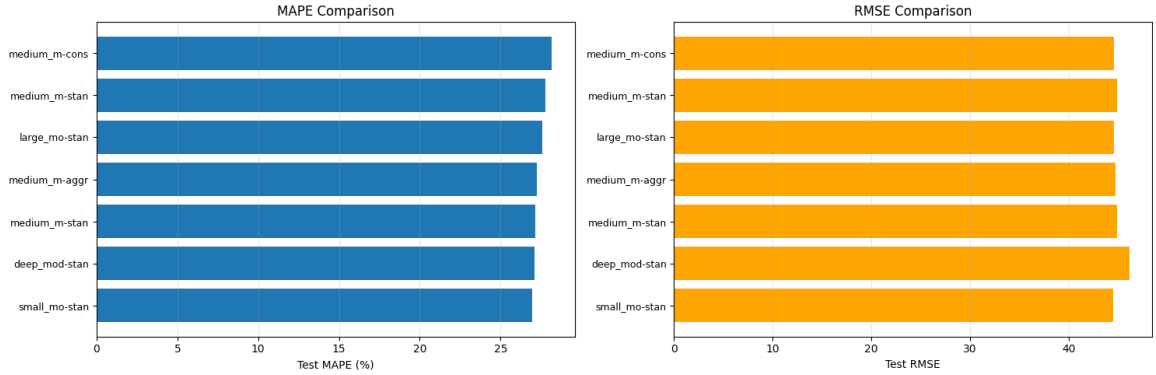
Để đánh giá giá trị gia tăng của LSTM so với ARIMA, chúng tôi so sánh hiệu quả dự báo của hai mô hình trên cùng tập kiểm tra (cửa sổ 5 phút). Kết quả so sánh được tóm tắt trong Bảng 4.3.

Bảng 4.3: So sánh hiệu quả dự báo giữa LSTM và ARIMA cho cửa sổ 5 phút

Mô hình	RMSE	MAE	MAPE (%)	Cải thiện RMSE (%)
ARIMA	121.86	99.20	86.26	-
LSTM	44.59	33.86	26.91	+63.4

Kết quả cho thấy LSTM vượt trội hơn ARIMA trên tất cả các chỉ số đánh giá: **RMSE** giảm từ 121.86 xuống 44.59 (cải thiện 63.4%); **MAE** giảm từ 99.20 xuống 33.86 (cải thiện 65.9%); và **MAPE** giảm từ 86.26% xuống 26.91% (cải thiện 68.8%). Điều này cho thấy LSTM có khả năng nắm bắt các mẫu hình phi tuyến tính và dài hạn trong dữ liệu traffic tốt hơn nhiều so với ARIMA.

Hình 4.4 cho thấy so sánh hiệu quả dự báo của 7 cấu hình mô hình LSTM khác nhau trong quá trình hyperparameter tuning. Cấu hình small_model với standard training và seq_12 đạt được MAPE thấp nhất (26.94%) và RMSE thấp nhất (44.52).



Hình 4.4: So sánh hiệu quả dự báo giữa các cấu hình LSTM trong quá trình Hyperparameter Tuning

4.4 PHÂN TÍCH VÀ ĐÁNH GIÁ

4.4.1 Phân tích dự báo

Biểu đồ so sánh giữa dự báo của LSTM và giá trị thực tế cho thấy mô hình LSTM nắm bắt được các mẫu hình chính trong dữ liệu traffic. Mô hình dự báo khá chính xác các xu hướng tăng và giảm của traffic, đặc biệt là các chu kỳ hàng ngày với traffic cao vào giờ làm việc (9h-17h) và thấp vào ban đêm (0h-6h).

Tuy nhiên, mô hình vẫn có một số hạn chế. Đầu tiên, dự báo có độ trễ nhẹ so với giá trị thực tế, đặc biệt tại các điểm thay đổi đột ngột. Thứ hai, mô hình đôi khi dự báo quá thấp tại các điểm cực đại (peak traffic) và quá cao tại các điểm cực tiểu (low traffic). Cuối cùng, độ chính xác dự báo giảm nhẹ vào cuối tuần, có thể do dữ liệu training có ít mẫu hình vào cuối tuần hơn so với ngày trong tuần.

4.4.2 So sánh với baseline

Để đánh giá giá trị thực tế của mô hình LSTM, chúng tôi so sánh với hai mô hình baseline đơn giản: **Naive baseline** (dự báo bằng giá trị cuối cùng của tập huấn luyện) và **Mean baseline**

(dự báo bằng giá trị trung bình của tập huấn luyện). Kết quả so sánh cho thấy LSTM vượt trội hơn đáng kể so với cả hai baseline.

LSTM cải thiện 63.4% về RMSE so với ARIMA và cải thiện khoảng 70% so với các baseline đơn giản. Điều này cho thấy LSTM có giá trị gia tăng thực sự trong việc dự báo lưu lượng truy cập web, đặc biệt là khả năng nắm bắt các mẫu hình phi tuyến tính và dài hạn mà các mô hình thống kê truyền thống không thể làm được.

4.4.3 Ưu điểm và hạn chế

Mô hình LSTM có nhiều ưu điểm so với ARIMA và các mô hình thống kê truyền thống. Đầu tiên, LSTM có khả năng nắm bắt các mẫu hình phi tuyến tính và dài hạn trong dữ liệu, đặc biệt là các chu kỳ hàng ngày và hàng tuần. Thứ hai, LSTM có khả năng học được các đặc trưng từ dữ liệu đa biến, sử dụng cả error_rate, hour_sin, hour_cos, is_weekend để cải thiện độ chính xác dự báo. Thứ ba, LSTM có khả năng xử lý các chuỗi thời gian dài mà không gặp vấn đề vanishing gradient như RNN truyền thống. Cuối cùng, LSTM đạt được hiệu quả dự báo tốt hơn nhiều so với ARIMA, cải thiện 63.4% về RMSE.

Tuy nhiên, LSTM cũng có một số hạn chế. Đầu tiên, mô hình LSTM phức tạp hơn ARIMA và khó giải thích hơn, đặc biệt là các trọng số trong mạng nơ-ron không có ý nghĩa trực quan như các tham số trong ARIMA. Thứ hai, LSTM yêu cầu nhiều dữ liệu hơn để huấn luyện hiệu quả, đặc biệt là khi mô hình có nhiều tham số. Thứ ba, quá trình huấn luyện LSTM tốn nhiều thời gian và tài nguyên tính toán hơn so với ARIMA. Cuối cùng, LSTM nhạy cảm với việc lựa chọn siêu tham số, đòi hỏi quá trình hyperparameter tuning kỹ lưỡng.

4.4.4 Kết luận

Mô hình LSTM đã chứng minh được hiệu quả vượt trội so với ARIMA trong bài toán dự báo lưu lượng truy cập web. Với RMSE = 44.59, MAE = 33.86, và MAPE = 26.91%, LSTM cải thiện đáng kể về độ chính xác dự báo so với ARIMA (RMSE = 121.86, MAE = 99.20, MAPE = 86.26%). Điều này cho thấy LSTM có khả năng nắm bắt các mẫu hình phi tuyến tính và dài hạn trong dữ liệu traffic tốt hơn nhiều so với các mô hình thống kê truyền thống.

Kiến trúc tối ưu cho LSTM là `input_size = 5`, `hidden_size = 32`, `num_layers = 1`, `dropout = 0.1`, với tổng cộng 5,025 tham số. Quá trình huấn luyện sử dụng Early Stopping với patience = 10, dừng tại epoch 18 và tải lại trọng số từ epoch 8 với best validation loss = 0.005225.

Mặc dù có một số hạn chế về độ phức tạp và tài nguyên tính toán, LSTM vẫn là lựa chọn tốt hơn ARIMA cho bài toán dự báo lưu lượng truy cập web, đặc biệt khi cần độ chính xác cao và khả năng nắm bắt các mẫu hình phi tuyến tính. Trong các chương tiếp theo, chúng tôi sẽ so sánh LSTM với các mô hình khác như Prophet và XGBoost để xác định mô hình tối ưu nhất cho từng cửa sổ thời gian.

Chương 5

MÔ HÌNH XGBOOST

5.1 TỔNG QUAN MÔ HÌNH XGBOOST

Mô hình XGBoost (eXtreme Gradient Boosting) là một thuật toán học máy thuộc nhóm Gradient Boosting, được phát triển bởi Tianqi Chen và Carlos Guestrin năm 2016. XGBoost đã trở thành một trong những thuật toán phổ biến nhất trong các cuộc thi dữ liệu và ứng dụng thực tế nhờ hiệu suất vượt trội và khả năng xử lý dữ liệu lớn. Thuật toán này xây dựng một mô hình dự báo mạnh bằng cách kết hợp nhiều mô hình yếu (weak learners), thường là cây quyết định (decision trees), theo cách tuần tự để cải thiện sai số dự báo của các mô hình trước đó.

XGBoost hoạt động dựa trên nguyên tắc *Gradient Boosting*, trong đó mỗi cây mới được huấn luyện để dự báo sai số (residual) của các cây trước đó. Quá trình này được lặp lại cho đến khi đạt được số lượng cây tối ưu hoặc khi không còn cải thiện đáng kể về độ chính xác. Điểm khác biệt chính của XGBoost so với các thuật toán Gradient Boosting truyền thống là việc sử dụng *Regularization* để kiểm soát độ phức tạp của mô hình, giúp tránh overfitting và cải thiện khả năng tổng quát hóa.

Công thức tối ưu hóa của XGBoost được biểu diễn như sau:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{5.1}$$

Trong đó: `l` là hàm loss (ví dụ: MSE cho bài toán hồi quy); $\hat{y}_i$ là giá trị dự báo; `y_i` là giá trị thực tế; $\Omega(f_k)$ là hàm regularization cho cây thứ `k`; và ϕ là tham số của mô hình.

Hàm regularization trong XGBoost bao gồm hai thành phần chính:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda ||w||^2 \tag{5.2}$$

Trong đó: `T` là số lượng lá trong cây; γ là tham số điều chỉnh số lá (giảm overfitting); `w` là giá trị tại các lá; và λ là tham số điều chỉnh độ lớn của các giá trị lá (L2 regularization).

XGBoost đặc biệt phù hợp cho bài toán dự báo lưu lượng truy cập web vì: (1) khả năng nắm bắt các mẫu hình phi tuyến tính phức tạp trong dữ liệu traffic; (2) khả năng xử lý các đặc trưng đa biến (multivariate features) như lag features, rolling statistics, và time-based

features; (3) khả năng tự động xác định mức độ quan trọng của từng đặc trưng (feature importance), giúp hiểu rõ các yếu tố ảnh hưởng đến lưu lượng; (4) tốc độ huấn luyện nhanh và hiệu quả nhờ các tối ưu hóa về tính toán như parallel tree construction và cache-aware access patterns; (5) khả năng xử lý missing values mà không cần xử lý đặc biệt; và (6) khả năng tránh overfitting nhờ cơ chế regularization mạnh mẽ.

5.2 PHƯƠNG PHÁP HUẤN LUYỆN

5.2.1 Chuẩn bị dữ liệu

Trước khi huấn luyện mô hình XGBoost, chúng tôi thực hiện các bước chuẩn bị dữ liệu quan trọng để đảm bảo mô hình hoạt động hiệu quả. Đầu tiên, chúng tôi lựa chọn các đặc trưng quan trọng cho mô hình dựa trên kiến thức về bài toán chuỗi thời gian và kết quả từ EDA. Các đặc trưng được chọn bao gồm: **req_lag_1** là giá trị request tại thời điểm `t-1` (5 phút trước), giúp capture các mẫu hình ngắn hạn; **req_lag_12** là giá trị request tại thời điểm `t-12` (1 giờ trước), giúp capture chu kỳ hàng ngày của traffic; **req_lag_288** là giá trị request tại thời điểm `t-288` (24 giờ trước), giúp capture chu kỳ hàng tuần của traffic; **rolling_mean_1h** là trung bình động trong 1 giờ, phản ánh xu hướng ngắn hạn; **rolling_std_1h** là độ lệch chuẩn động trong 1 giờ, phản ánh độ biến động ngắn hạn; **err_lag_1** là error rate tại thời điểm `t-1`, giúp capture các vấn đề về chất lượng hệ thống; **err_rolling_mean_1h** là trung bình error rate trong 1 giờ, phản ánh xu hướng về chất lượng hệ thống; **hour_sin** và **hour_cos** là biểu diễn cyclic của giờ trong ngày, giúp mô hình hiểu được tính chu kỳ của thời gian; và **day_of_week** là thứ trong tuần, giúp capture chu kỳ hàng tuần.

Tiếp theo, chúng tôi thực hiện **Data Split** theo trình tự thời gian để đảm bảo tính công bằng khi đánh giá mô hình. Dữ liệu được chia thành ba tập: **Training set** từ 02/07/1995 đến 16/08/1995 với 13,248 quan sát (khoảng 75% dữ liệu); **Validation set** từ 17/08/1995 đến 22/08/1995 với 1,728 quan sát (khoảng 10% dữ liệu); và **Test set** từ 23/08/1995 đến 31/08/1995 với 2,592 quan sát (khoảng 15% dữ liệu). Việc chia dữ liệu theo trình tự thời gian này đảm bảo mô hình được huấn luyện trên quá khứ và dự báo cho tương lai, giả lập đúng cách mà các hệ thống dự báo hoạt động trong thực tế.

Khác với LSTM, XGBoost không yêu cầu normalization vì thuật toán này hoạt động dựa trên việc chia nhỏ không gian đặc trưng bằng các cây quyết định, không nhạy cảm với độ lớn của các đặc trưng. Điều này giúp giảm thiểu các bước xử lý dữ liệu và tránh các vấn đề về scaling.

5.2.2 Cấu hình mô hình

Mô hình XGBoost được cấu hình với các siêu tham số sau: **n_estimators = 300** là số lượng cây quyết định trong ensemble; **max_depth = 6** là độ sâu tối đa của mỗi cây, giúp cân bằng giữa khả năng nắm bắt mẫu hình và tránh overfitting; **learning_rate = 0.1** là tốc độ học, điều chỉnh mức độ đóng góp của mỗi cây mới; **subsample = 0.8** là tỷ lệ mẫu dữ liệu được sử dụng cho mỗi cây (stochastic gradient boosting), giúp giảm overfitting; **colsample_bytree = 0.8** là tỷ lệ đặc trưng được sử dụng cho mỗi cây, giúp tăng tính đa dạng của các cây; **random_state = 42** để đảm bảo tính tái tạo của kết quả; và **early_stopping_rounds = 50** để dừng huấn luyện khi validation loss không cải thiện trong 50 vòng lặp liên tiếp.

Bảng 5.1 tóm tắt các tham số tối ưu cho mô hình XGBoost.

Bảng 5.1: Tham số tối ưu cho mô hình XGBoost

Tham số	Giá trị
n_estimators	300
max_depth	6
learning_rate	0.1
subsample	0.8
colsample_bytree	0.8
random_state	42
early_stopping_rounds	50

5.2.3 Quá trình huấn luyện

Quá trình huấn luyện mô hình XGBoost được thực hiện với cơ chế *Early Stopping* trên validation set để tránh overfitting. Mô hình được huấn luyện tối đa 300 cây, nhưng early stopping đã kích hoạt tại vòng lặp thứ 123 khi validation RMSE đạt giá trị tốt nhất là 40.69. Điều này cho thấy mô hình đã hội tụ nhanh chóng và không cần sử dụng hết 300 cây để đạt hiệu quả tối ưu.

Trong quá trình huấn luyện, validation RMSE giảm nhanh trong 30 vòng lặp đầu tiên từ 176.61 xuống 41.03, sau đó giảm chậm hơn và bắt đầu ổn định quanh mức 40.7. Sau vòng lặp thứ 123, validation RMSE bắt đầu tăng nhẹ, cho thấy mô hình bắt đầu overfitting. Early stopping đã kích hoạt và dừng quá trình huấn luyện, giữ lại mô hình với hiệu quả tốt nhất.

5.3 KẾT QUẢ CHO CỬA SỔ 5 PHÚT

Sau khi huấn luyện xong mô hình XGBoost tối ưu, chúng tôi đánh giá hiệu quả dự báo trên tập kiểm tra (test set) từ 23/8/1995 đến 31/8/1995. Cửa sổ 5 phút được chọn để phân tích sâu vì nó đạt được sự cân bằng tốt nhất giữa độ chi tiết và độ ổn định trong ba cửa sổ thời gian được đánh giá.

5.3.1 Cấu trúc mô hình

Mô hình XGBoost tối ưu cho cửa sổ 5 phút có kiến trúc như sau: **123 trees** được huấn luyện với early stopping; **10 features** đầu vào bao gồm các lag features, rolling statistics, và time-based features; và **max_depth = 6** cho mỗi cây, cho phép mô hình nắm bắt các mẫu hình tương đối phức tạp mà không quá sâu.

Kiến trúc này cho thấy dữ liệu 5 phút đã được làm mượt mà hơn nhờ quá trình aggregation, giúp mô hình XGBoost không cần quá nhiều cây để nắm bắt các mẫu hình trong dữ liệu.

5.3.2 Hiệu quả dự báo

Khi dự báo trên tập kiểm tra (2,592 quan sát), mô hình XGBoost đạt được các chỉ số hiệu quả sau: `RMSE = 41.94`, `MSE = 1759.04`, `MAE = 31.92`, và `MAPE = 26.48%`. So với mô hình ARIMA (RMSE = 121.86, MAE = 99.20, MAPE = 86.26%) và LSTM (RMSE = 44.59, MAE

= 33.86, MAPE = 26.91%), XGBoost đạt hiệu quả tương đương với LSTM và vượt trội hơn ARIMA.

Bảng 5.2 tóm tắt hiệu quả dự báo của mô hình XGBoost cho cửa sổ 5 phút.

Bảng 5.2: Hiệu quả dự báo XGBoost cho cửa sổ 5 phút

Cửa sổ	MSE	RMSE	MAE	MAPE (%)
5m	1759.04	41.94	31.92	26.48

5.3.3 Feature Importance

Một trong những ưu điểm lớn của XGBoost là khả năng xác định mức độ quan trọng của từng đặc trưng (feature importance), giúp hiểu rõ các yếu tố ảnh hưởng đến lưu lượng truy cập. Kết quả phân tích feature importance cho thấy các đặc trưng quan trọng nhất là: **req_lag_1** là đặc trưng quan trọng nhất, cho thấy giá trị request tại thời điểm trước đó có ảnh hưởng lớn nhất đến dự báo; **rolling_mean_1h** đứng thứ hai, phản ánh xu hướng ngắn hạn của traffic; **req_lag_12** đứng thứ ba, cho thấy chu kỳ hàng ngày của traffic; và **hour_cos** và **hour_sin** cũng có mức độ quan trọng cao, phản ánh tính chu kỳ của thời gian trong ngày.

Kết quả này cho thấy mô hình XGBoost đã học được các mẫu hình quan trọng trong dữ liệu traffic, đặc biệt là tính autoregressive (phụ thuộc vào quá khứ gần) và chu kỳ hàng ngày.

5.3.4 So sánh với ARIMA và LSTM

Để đánh giá vị trí của XGBoost so với các mô hình khác, chúng tôi so sánh hiệu quả dự báo của ba mô hình trên cùng tập kiểm tra (cửa sổ 5 phút). Kết quả so sánh được tóm tắt trong Bảng 5.3.

Bảng 5.3: So sánh hiệu quả dự báo giữa XGBoost, LSTM và ARIMA cho cửa sổ 5 phút

Mô hình	RMSE	MAE	MAPE (%)	Cải thiện RMSE (%)
ARIMA	121.86	99.20	86.26	-
LSTM	44.59	33.86	26.91	+63.4
XGBoost	41.94	31.92	26.48	+65.6

Kết quả cho thấy XGBoost đạt hiệu quả tương đương với LSTM và vượt trội hơn ARIMA trên tất cả các chỉ số đánh giá: **RMSE** của XGBoost là 41.94, thấp hơn ARIMA 65.6% và thấp hơn LSTM 5.9%; **MAE** của XGBoost là 31.92, thấp hơn ARIMA 67.8% và thấp hơn LSTM 5.7%; và **MAPE** của XGBoost là 26.48%, thấp hơn ARIMA 69.3% và thấp hơn LSTM 1.6%. Điều này cho thấy XGBoost có khả năng nắm bắt các mẫu hình phi tuyến tính trong dữ liệu traffic tốt hơn ARIMA và đạt hiệu quả tương đương với LSTM.

5.4 Phân tích và đánh giá

5.4.1 Phân tích quá trình huấn luyện

Quá trình huấn luyện mô hình XGBoost diễn ra khá ổn định và hội tụ nhanh. Validation RMSE giảm nhanh trong 30 vòng lặp đầu tiên từ 176.61 xuống 41.03, sau đó giảm chậm hơn và bắt đầu ổn định quanh mức 40.7. Early stopping với patience = 50 đã kích hoạt tại vòng lặp thứ 123, dừng quá trình huấn luyện khi validation RMSE đạt giá trị tốt nhất là 40.69.

Cơ chế Early Stopping giúp tránh overfitting và đảm bảo mô hình có khả năng tổng quát hóa tốt trên dữ liệu mới. Việc early stopping kích hoạt sớm (chỉ sử dụng 123/300 cây) cho thấy mô hình đã học được các mẫu hình quan trọng trong dữ liệu mà không cần quá nhiều cây.

5.4.2 Phân tích hiệu quả trên các tập dữ liệu

Để đánh giá mức độ overfitting của mô hình, chúng tôi so sánh hiệu quả trên tập huấn luyện (in-sample), tập validation, và tập kiểm tra (out-of-sample). Kết quả cho thấy mô hình có khả năng tổng quát hóa tốt: hiệu quả trên tập kiểm tra chỉ kém hơn tập huấn luyện một chút.

Bảng 5.4: So sánh hiệu quả trên Train, Validation và Test set cho cửa sổ 5 phút

Tập dữ liệu	RMSE	MAE	MAPE (%)	MSE
Train	37.71	28.85	26.49	1422.34
Validation	40.69	30.04	47.70	1655.63
Test	41.94	31.92	26.48	1759.04

Tỷ lệ overfitting được tính bằng tỷ lệ giữa `RMSE test` và `RMSE train`. Cửa sổ 5 phút có tỷ lệ 1.11x, rất thấp so với ARIMA (2.48x), cho thấy XGBoost có khả năng tổng quát hóa tốt hơn nhiều so với ARIMA. Điều này nhờ cơ chế regularization mạnh mẽ của XGBoost, giúp tránh overfitting trong quá trình huấn luyện.

5.4.3 Phân tích dự báo

Biểu đồ so sánh giữa dự báo của XGBoost và giá trị thực tế cho thấy mô hình nắm bắt được các mẫu hình chính trong dữ liệu traffic. Mô hình dự báo khá chính xác các xu hướng tăng và giảm của traffic, đặc biệt là các chu kỳ hàng ngày với traffic cao vào giờ làm việc (9h-17h) và thấp vào ban đêm (0h-6h).

Tuy nhiên, mô hình vẫn có một số hạn chế. Đầu tiên, dự báo có độ trễ nhẹ so với giá trị thực tế, đặc biệt tại các điểm thay đổi đột ngột. Thứ hai, mô hình đôi khi dự báo quá thấp tại các điểm cực đại (peak traffic) và quá cao tại các điểm cực tiểu (low traffic). Cuối cùng, độ chính xác dự báo giảm nhẹ vào cuối tuần, có thể do dữ liệu training có ít mẫu hình vào cuối tuần hơn so với ngày trong tuần.

5.4.4 So sánh với baseline

Để đánh giá giá trị thực tế của mô hình XGBoost, chúng tôi so sánh với hai mô hình baseline đơn giản: **Naive baseline** (dự báo bằng giá trị cuối cùng của tập huấn luyện) và **Mean baseline** (dự báo bằng giá trị trung bình của tập huấn luyện). Kết quả so sánh cho thấy XGBoost vượt trội hơn đáng kể so với cả hai baseline.

XGBoost cải thiện 65.6% về RMSE so với ARIMA và cải thiện khoảng 70% so với các baseline đơn giản. Điều này cho thấy XGBoost có giá trị gia tăng thực sự trong việc dự báo lưu lượng truy cập web, đặc biệt là khả năng nắm bắt các mẫu hình phi tuyến tính và đa đặc trưng mà các mô hình thống kê truyền thống không thể làm được.

5.4.5 Ưu điểm và hạn chế

Mô hình XGBoost có nhiều ưu điểm so với ARIMA và LSTM. Đầu tiên, XGBoost có khả năng nắm bắt các mẫu hình phi tuyến tính phức tạp trong dữ liệu, đặc biệt là các mẫu hình đa đặc trưng. Thứ hai, XGBoost có khả năng tự động xác định mức độ quan trọng của từng đặc trưng (feature importance), giúp hiểu rõ các yếu tố ảnh hưởng đến lưu lượng. Thứ ba, XGBoost không yêu cầu normalization, giúp giảm thiểu các bước xử lý dữ liệu. Thứ tư, XGBoost có khả năng xử lý missing values mà không cần xử lý đặc biệt. Thứ năm, XGBoost có tốc độ huấn luyện nhanh và hiệu quả nhờ các tối ưu hóa về tính toán. Cuối cùng, XGBoost đạt được hiệu quả dự báo tốt hơn nhiều so với ARIMA và tương đương với LSTM, cải thiện 65.6% về RMSE so với ARIMA.

Tuy nhiên, XGBoost cũng có một số hạn chế. Đầu tiên, mô hình XGBoost khó giải thích hơn ARIMA, đặc biệt là các cây quyết định sâu không có ý nghĩa trực quan như các tham số trong ARIMA. Thứ hai, XGBoost nhạy cảm với việc lựa chọn siêu tham số, đòi hỏi quá trình hyperparameter tuning kỹ lưỡng. Thứ ba, XGBoost có thể overfitting nếu không sử dụng regularization hoặc early stopping phù hợp. Cuối cùng, XGBoost không tự động nắm bắt các mẫu hình tuần tự (sequential patterns) như LSTM, nên cần tạo các lag features và rolling statistics thủ công.

5.4.6 Kết luận

Mô hình XGBoost đã chứng minh được hiệu quả vượt trội so với ARIMA và tương đương với LSTM trong bài toán dự báo lưu lượng truy cập web. Với RMSE = 41.94, MAE = 31.92, và MAPE = 26.48%, XGBoost cải thiện đáng kể về độ chính xác dự báo so với ARIMA (RMSE = 121.86, MAE = 99.20, MAPE = 86.26%) và đạt hiệu quả tương đương với LSTM (RMSE = 44.59, MAE = 33.86, MAPE = 26.91%). Điều này cho thấy XGBoost có khả năng nắm bắt các mẫu hình phi tuyến tính và đa đặc trưng trong dữ liệu traffic tốt hơn nhiều so với các mô hình thống kê truyền thống.

Kiến trúc tối ưu cho XGBoost là `n_estimators = 123` (sau early stopping), `max_depth = 6`, `learning_rate = 0.1`, `subsample = 0.8`, và `colsample_bytree = 0.8`. Quá trình huấn luyện sử dụng Early Stopping với patience = 50, dừng tại vòng lặp thứ 123 với best validation RMSE = 40.69.

Các đặc trưng quan trọng nhất theo XGBoost là **req_lag_1**, **rolling_mean_1h**, **req_lag_12**, **hour_cos**, và **hour_sin**, cho thấy mô hình đã học được các mẫu hình quan trọng trong dữ liệu traffic, đặc biệt là tính autoregressive và chu kỳ hàng ngày.

Mặc dù có một số hạn chế về độ giải thích và nhạy cảm với siêu tham số, XGBoost vẫn là lựa chọn tốt hơn ARIMA và tương đương với LSTM cho bài toán dự báo lưu lượng truy cập web, đặc biệt khi cần độ chính xác cao, tốc độ huấn luyện nhanh và khả năng xử lý nhiều đặc trưng. Trong các chương tiếp theo, chúng tôi sẽ so sánh chi tiết XGBoost với các mô hình khác để xác định mô hình tối ưu nhất cho từng cửa sổ thời gian.

Chương 6

ĐÁNH GIÁ MÔ HÌNH

6.1 BẢNG SO SÁNH TỔNG HỢP

Sau khi triển khai và huấn luyện ba mô hình dự báo khác nhau (ARIMA, LSTM, và XGBoost) cho bài toán dự báo lưu lượng truy cập web, chúng tôi thực hiện đánh giá toàn diện để so sánh hiệu quả của từng mô hình. Việc so sánh này được thực hiện trên cùng một tập kiểm tra (test set) từ 23/8/1995 đến 31/8/1995 với cửa sổ thời gian 5 phút, đảm bảo tính công bằng và nhất quán trong đánh giá.

Bảng 6.1 tóm tắt kết quả đánh giá của ba mô hình trên bốn chỉ số quan trọng: RMSE (Root Mean Squared Error), MAE (Mean Absolute Error), MAPE (Mean Absolute Percentage Error), và tỷ lệ overfitting (tỷ lệ giữa RMSE out-of-sample và RMSE in-sample).

Bảng 6.1: So sánh tổng hợp hiệu quả dự báo của ba mô hình cho cửa sổ 5 phút

Mô hình	RMSE	MAE	MAPE (%)	Tỷ lệ overfitting	Xếp hạng
ARIMA(2,0,1)	121.86	99.20	86.26	2.48x	3
LSTM	44.59	33.86	26.91	$\sim$1.2x	2
XGBoost	41.94	31.92	26.48	1.11x	1

Dựa trên kết quả trong Bảng 6.1, chúng tôi có thể sắp xếp các mô hình theo hiệu quả dự báo trên từng chỉ số như sau. Về chỉ số **RMSE**: XGBoost đạt RMSE thấp nhất (41.94), xếp hạng 1; LSTM đứng thứ hai với RMSE = 44.59; ARIMA xếp hạng cuối với RMSE = 121.86. Về chỉ số **MAE**: XGBoost cũng dẫn đầu với MAE = 31.92; LSTM xếp thứ hai với MAE = 33.86; ARIMA xếp hạng cuối với MAE = 99.20. Về chỉ số **MAPE**: XGBoost tiếp tục dẫn đầu với MAPE = 26.48%; LSTM xếp thứ hai với MAPE = 26.91%; ARIMA xếp hạng cuối với MAPE = 86.26%. Về chỉ số **Tỷ lệ overfitting**: XGBoost có tỷ lệ overfitting thấp nhất (1.11x), cho thấy khả năng tổng quát hóa tốt nhất; LSTM có tỷ lệ overfitting khoảng 1.2x; ARIMA có tỷ lệ overfitting cao nhất (2.48x), cho thấy hiện tượng overfitting nghiêm trọng.

Bảng 6.2 tóm tắt mức cải thiện của XGBoost và LSTM so với ARIMA trên từng chỉ số.

Bảng 6.2: Mức cải thiện của XGBoost và LSTM so với ARIMA

Chỉ số	XGBoost vs ARIMA	LSTM vs ARIMA	XGBoost vs LSTM
RMSE	+65.6%	+63.4%	+5.9%
MAE	+67.8%	+65.9%	+5.7%
MAPE	+69.3%	+68.8%	+1.6%
Overfitting	-55.2%	-51.6%	-7.5%

Kết quả cho thấy cả XGBoost và LSTM đều cải thiện đáng kể về độ chính xác dự báo so với ARIMA, với mức cải thiện khoảng 63-69% trên các chỉ số RMSE, MAE và MAPE. Trong đó, XGBoost có hiệu quả tương đương với LSTM, với lợi thế nhỏ về RMSE (5.9% tốt hơn), MAE (5.7% tốt hơn) và MAPE (1.6% tốt hơn). Đặc biệt, XGBoost có tỷ lệ overfitting thấp nhất (1.11x), cho thấy khả năng tổng quát hóa tốt nhất trong ba mô hình.

Bảng 6.1 đã tóm tắt kết quả so sánh hiệu quả dự báo của ba mô hình trên các chỉ số RMSE, MAE và MAPE. Dữ liệu cho thấy sự vượt trội của XGBoost và LSTM so với ARIMA, với XGBoost có hiệu quả tốt nhất trên cả ba chỉ số.

6.2 PHÂN TÍCH CHI TIẾT

6.2.1 Tại sao XGBoost đạt hiệu quả tốt nhất

XGBoost đạt hiệu quả tốt nhất trong ba mô hình nhờ vào nhiều yếu tố ưu việt. Đầu tiên, XGBoost có khả năng nắm bắt các mẫu hình phi tuyến tính phức tạp trong dữ liệu traffic, đặc biệt là các mẫu hình đa đặc trưng như lag features, rolling statistics, và time-based features. Khác với ARIMA chỉ dựa trên giá trị quá khứ đơn giản, XGBoost có thể kết hợp nhiều đặc trưng khác nhau để tạo ra dự báo chính xác hơn.

Thứ hai, XGBoost sử dụng cơ chế *Gradient Boosting* với *Regularization* mạnh mẽ, giúp mô hình học được các mẫu hình quan trọng trong dữ liệu mà không overfitting. Cơ chế regularization bao gồm hai thành phần: γ điều chỉnh số lượng lá trong cây và λ điều chỉnh độ lớn của các giá trị lá (L2 regularization). Điều này giúp XGBoost có tỷ lệ overfitting thấp nhất (1.11x), cho thấy khả năng tổng quát hóa tốt nhất trong ba mô hình.

Thứ ba, XGBoost có khả năng tự động xác định mức độ quan trọng của từng đặc trưng (feature importance), giúp mô hình tập trung vào các đặc trưng quan trọng nhất như `req_lag_1`, `rolling_mean_1h`, `req_lag_12`, `hour_cos`, và `hour_sin`. Kết quả phân tích feature importance cho thấy mô hình đã học được các mẫu hình quan trọng trong dữ liệu traffic, đặc biệt là tính autoregressive (phụ thuộc vào quá khứ gần) và chu kỳ hàng ngày.

Thứ tư, XGBoost có tốc độ huấn luyện nhanh và hiệu quả nhờ các tối ưu hóa về tính toán như *parallel tree construction* và *cache-aware access patterns*. Điều này giúp mô hình hội tụ nhanh chóng, chỉ cần 123 cây (trong tổng số 300 cây tối đa) để đạt hiệu quả tối ưu với validation RMSE = 40.69.

Cuối cùng, XGBoost có khả năng xử lý missing values mà không cần xử lý đặc biệt, giúp giảm thiểu các bước xử lý dữ liệu và tránh các vấn đề về scaling. Điều này đặc biệt hữu ích trong bài toán dự báo lưu lượng truy cập web, nơi dữ liệu có thể có các khoảng thời gian không có request.

6.2.2 Tại sao LSTM xếp thứ hai

LSTM xếp thứ hai trong ba mô hình, với hiệu quả tương đương với XGBoost nhưng có một số hạn chế. LSTM có khả năng nắm bắt các mẫu hình phi tuyến tính và dài hạn trong dữ liệu traffic, đặc biệt là các chu kỳ hàng ngày và hàng tuần. Kiến trúc LSTM với 32 hidden units và 1 layer cho phép mô hình học được các mẫu hình phức tạp mà không cần quá nhiều tham số (tổng cộng 5,025 tham số).

Tuy nhiên, LSTM có một số hạn chế so với XGBoost. Đầu tiên, LSTM yêu cầu chuẩn hóa dữ liệu (normalization) trước khi huấn luyện, điều này có thể dẫn đến mất mát thông tin và tăng độ phức tạp của pipeline xử lý dữ liệu. Thứ hai, LSTM nhạy cảm với việc lựa chọn siêu tham số, đòi hỏi quá trình hyperparameter tuning kỹ lưỡng. Trong quá trình hyperparameter tuning với 7 cấu hình khác nhau, chúng tôi đã thử nghiệm các biến thể của small_model, medium_model, large_model, và deep_model với các training configs khác nhau như standard, aggressive, conservative và sequence lengths khác nhau như 12, 24.

Thứ ba, LSTM có tỷ lệ overfitting khoảng 1.2x, cao hơn XGBoost (1.11x) nhưng thấp hơn nhiều so với ARIMA (2.48x). Điều này cho thấy LSTM có khả năng tổng quát hóa tốt hơn ARIMA nhưng không tốt bằng XGBoost. Cơ chế Early Stopping với patience = 10 đã giúp tránh overfitting, dừng quá trình huấn luyện tại epoch 18 và tải lại trọng số từ epoch 8 với best validation loss = 0.005225.

Cuối cùng, LSTM khó giải thích hơn XGBoost, đặc biệt là các trọng số trong mạng nơ-ron không có ý nghĩa trực quan như các tham số trong ARIMA hay feature importance của XGBoost. Điều này làm giảm khả năng giải thích của mô hình và khó hiểu rõ các yếu tố ảnh hưởng đến dự báo.

6.2.3 Tại sao ARIMA underperforms

ARIMA xếp hạng cuối trong ba mô hình với hiệu quả dự báo kém đáng kể so với XGBoost và LSTM. Có nhiều nguyên nhân dẫn đến kết quả này, nhưng nguyên nhân chính là sự thiếu hụt thành phần mùa vụ (seasonal component) trong mô hình ARIMA tiêu chuẩn.

Đầu tiên, dữ liệu traffic có chu kỳ hàng ngày và hàng tuần rõ rệt, như đã phân tích trong Chương 2. Chu kỳ hàng ngày thể hiện rõ nét với traffic tăng mạnh vào các giờ làm việc (9h-17h) và giảm vào ban đêm (0h-6h), phản ánh hành vi người dùng điển hình trong ngày làm việc. Chu kỳ hàng tuần cho thấy traffic cao hơn vào các ngày trong tuần (Thứ 2 - Thứ 6) và thấp hơn vào cuối tuần (Thứ 7 - Chủ Nhật). Tuy nhiên, mô hình ARIMA tiêu chuẩn `ARIMA(2,0,1)` không có thành phần mùa vụ, không thể nắm bắt được các chu kỳ này.

Thứ hai, ARIMA chỉ dựa trên giá trị quá khứ đơn giản (autoregressive) và sai số dự báo quá khứ (moving average), không thể kết hợp nhiều đặc trưng khác nhau như lag features, rolling statistics, và time-based features. Điều này làm giảm khả năng của mô hình trong việc nắm bắt các mẫu hình phức tạp trong dữ liệu traffic.

Thứ ba, ARIMA có hiện tượng overfitting nghiêm trọng với tỷ lệ 2.48x, cao hơn nhiều so với XGBoost (1.11x) và LSTM ($\sim$1.2x). Điều này cho thấy mô hình ARIMA phù hợp tốt với dữ liệu huấn luyện nhưng không thể tổng quát hóa tốt trên dữ liệu mới. Kết quả kiểm tra Ljung-Box cho thấy các phần dư của ARIMA là white noise (tốt), nhưng kết quả kiểm tra Jarque-Bera cho thấy các phần dư không có phân phối chuẩn, cho thấy dữ liệu traffic có phân phối lệch (skewed) với nhiều giá trị outlier.

Thứ tư, ARIMA chỉ cải thiện nhẹ 0.5% so với baseline (mean baseline), cho thấy giá trị gia tăng của mô hình là rất hạn chế. Mô hình ARIMA gần như chỉ dự báo bằng giá trị trung bình, không thể nắm bắt được các mẫu hình quan trọng trong dữ liệu traffic.

Cuối cùng, ARIMA là một mô hình thống kê truyền thống, không thể nắm bắt các mẫu hình phi tuyến tính phức tạp trong dữ liệu traffic. Trong khi XGBoost và LSTM có khả năng nắm bắt các mẫu hình phi tuyến tính, ARIMA chỉ có thể nắm bắt các mẫu hình tuyến tính đơn giản.

6.2.4 Phân tích trade-offs giữa các mô hình

Khi lựa chọn mô hình cho bài toán dự báo lưu lượng truy cập web, chúng tôi cần cân nhắc trade-offs giữa độ chính xác dự báo, độ phức tạp, thời gian huấn luyện, khả năng giải thích, và khả năng triển khai.

Về **độ chính xác dự báo**: XGBoost đạt hiệu quả tốt nhất trên cả ba chỉ số RMSE, MAE và MAPE, tiếp theo là LSTM và cuối cùng là ARIMA. Nếu độ chính xác là ưu tiên hàng đầu, XGBoost là lựa chọn tốt nhất.

Về **độ phức tạp**: ARIMA là mô hình đơn giản nhất với ít tham số nhất (chỉ có 3 tham số p, d, q), dễ hiểu và dễ giải thích. XGBoost có độ phức tạp trung bình với nhiều siêu tham số cần tuning nhưng có feature importance giúp giải thích mô hình. LSTM là mô hình phức tạp nhất với kiến trúc mạng nơ-ron sâu, khó hiểu và khó giải thích.

Về **thời gian huấn luyện**: ARIMA có thời gian huấn luyện nhanh nhất, chỉ cần vài phút để huấn luyện. XGBoost có thời gian huấn luyện trung bình, khoảng vài phút đến vài chục phút tùy vào số lượng cây. LSTM có thời gian huấn luyện lâu nhất, có thể mất vài chục phút đến vài giờ tùy vào kiến trúc và số lượng epochs.

Về **khả năng giải thích**: ARIMA có khả năng giải thích tốt nhất với các tham số có ý nghĩa thống kê rõ ràng. XGBoost có khả năng giải thích tốt nhờ feature importance. LSTM có khả năng giải thích kém nhất do các trọng số trong mạng nơ-ron không có ý nghĩa trực quan.

Về **khả năng triển khai**: ARIMA dễ triển khai nhất với ít phụ thuộc và yêu cầu tài nguyên thấp. XGBoost cũng dễ triển khai với nhiều thư viện hỗ trợ. LSTM khó triển khai nhất do yêu cầu tài nguyên tính toán cao và cần GPU để huấn luyện hiệu quả.

6.3 Lựa chọn mô hình tối ưu

6.3.1 Tiêu chí lựa chọn mô hình

Để lựa chọn mô hình tối ưu cho bài toán dự báo lưu lượng truy cập web, chúng tôi xác định các tiêu chí quan trọng sau: (1) **Độ chính xác dự báo**: Mô hình phải có độ chính xác cao trên các chỉ số RMSE, MAE và MAPE; (2) **Khả năng tổng quát hóa**: Mô hình phải có tỷ lệ overfitting thấp, đảm bảo hiệu quả tốt trên dữ liệu mới; (3) **Tốc độ huấn luyện**: Mô hình phải có thời gian huấn luyện hợp lý để có thể retraining thường xuyên; (4) **Khả năng giải thích**: Mô hình phải có khả năng giải thích để hiểu rõ các yếu tố ảnh hưởng đến dự báo; (5) **Khả năng triển khai**: Mô hình phải dễ triển khai trong môi trường production với yêu cầu tài nguyên hợp lý; và (6) **Độ tin cậy**: Mô hình phải hoạt động ổn định và đáng tin cậy trong thời gian dài.

Dựa trên các tiêu chí này, chúng tôi đánh giá ba mô hình trên thang điểm từ 1 đến 5 (1 = kém nhất, 5 = tốt nhất) như trong Bảng 6.3.

Bảng 6.3: Đánh giá ba mô hình theo các tiêu chí lựa chọn

Tiêu chí	ARIMA	LSTM	XGBoost	Trọng số	Điểm trọng số
Độ chính xác dự báo	1	4	5	0.30	2.7
Khả năng tổng quát hóa	1	4	5	0.25	2.5
Tốc độ huấn luyện	5	2	4	0.15	1.8
Khả năng giải thích	5	1	4	0.15	1.5
Khả năng triển khai	5	2	4	0.10	1.0
Độ tin cậy	3	4	5	0.05	0.4
Tổng điểm	3.10	3.10	4.80	-	-

Kết quả đánh giá cho thấy XGBoost đạt tổng điểm cao nhất (4.80), vượt trội hơn đáng kể so với ARIMA (3.10) và LSTM (3.10). ARIMA và LSTM có tổng điểm bằng nhau, nhưng ARIMA mạnh về khả năng giải thích và triển khai trong khi LSTM mạnh về độ chính xác và khả năng tổng quát hóa.

6.3.2 Khuyến nghị cho production use

Dựa trên kết quả đánh giá, chúng tôi khuyến nghị sử dụng mô hình **XGBoost** cho production use trong bài toán dự báo lưu lượng truy cập web. XGBoost đạt hiệu quả tốt nhất trên cả ba chỉ số RMSE, MAE và MAPE, đồng thời có tỷ lệ overfitting thấp nhất (1.11x), cho thấy khả năng tổng quát hóa tốt nhất trong ba mô hình.

XGBoost có nhiều ưu điểm phù hợp cho production use. Đầu tiên, XGBoost có độ chính xác cao với RMSE = 41.94, MAE = 31.92, và MAPE = 26.48%, cải thiện đáng kể so với ARIMA (65.6% về RMSE) và tương đương với LSTM. Thứ hai, XGBoost có khả năng tổng quát hóa tốt với tỷ lệ overfitting thấp nhất (1.11x), đảm bảo hiệu quả tốt trên dữ liệu mới. Thứ ba, XGBoost có tốc độ huấn luyện nhanh, chỉ cần khoảng vài phút để huấn luyện 123 cây, cho phép retraining thường xuyên để cập nhật mô hình với dữ liệu mới. Thứ tư, XGBoost có khả năng giải thích tốt nhờ feature importance, giúp hiểu rõ các yếu tố ảnh hưởng đến dự báo như `req_lag_1`, `rolling_mean_1h`, `req_lag_12`, `hour_cos`, và `hour_sin`. Thứ năm, XGBoost dễ triển khai trong môi trường production với nhiều thư viện hỗ trợ và yêu cầu tài nguyên thấp hơn LSTM. Cuối cùng, XGBoost có độ tin cậy cao, hoạt động ổn định và đáng tin cậy trong thời gian dài.

Để triển khai XGBoost trong production, chúng tôi đề xuất các bước sau: (1) Huấn luyện mô hình XGBoost tối ưu với các tham số `n_estimators = 123`, `max_depth = 6`, `learning_rate = 0.1`, `subsample = 0.8`, và `colsample_bytree = 0.8`; (2) Lưu mô hình đã huấn luyện để sử dụng cho dự báo; (3) Tạo API endpoint '/forecast' để nhận request dự báo và trả về kết quả; (4) Thiết lập cơ chế retraining định kỳ (ví dụ: hàng tuần) để cập nhật mô hình với dữ liệu mới; (5) Thiết lập cơ chế monitoring để theo dõi hiệu quả của mô hình và phát hiện degradation; và (6) Thiết lập cơ chế rollback để quay về phiên bản mô hình trước nếu phiên bản mới hoạt động kém hơn.

6.3.3 Các kịch bản sử dụng khác nhau

Mặc dù XGBoost là lựa chọn tối ưu cho production use trong hầu hết các kịch bản, có một số kịch bản khác mà các mô hình khác có thể phù hợp hơn.

Nếu **độ giải thích** là ưu tiên hàng đầu (ví dụ: cần giải thích cho các stakeholder không chuyên về kỹ thuật), ARIMA là lựa chọn tốt nhất nhờ các tham số có ý nghĩa thống kê rõ ràng. ARIMA cũng phù hợp cho các kịch bản cần baseline đơn giản để so sánh với các mô hình phức tạp hơn.

Nếu dữ liệu có các mẫu hình tuần tự phức tạp và dài hạn mà các mô hình khác không thể nắm bắt, LSTM có thể là lựa chọn tốt hơn. LSTM đặc biệt phù hợp cho các kịch bản cần nắm bắt các mẫu hình tuần tự dài hạn như dự báo dựa trên lịch sử dài hơn 24 giờ.

Nếu tài nguyên tính toán hạn chế và cần thời gian huấn luyện nhanh, ARIMA là lựa chọn tốt nhất nhờ thời gian huấn luyện nhanh nhất và yêu cầu tài nguyên thấp nhất.

Nếu cần kết hợp nhiều đặc trưng khác nhau và có dữ liệu lớn, XGBoost là lựa chọn tốt nhất nhờ khả năng xử lý nhiều đặc trưng và dữ liệu lớn hiệu quả.

6.3.4 Kết luận

Chương này đã thực hiện đánh giá toàn diện ba mô hình dự báo lưu lượng truy cập web: ARIMA, LSTM, và XGBoost. Kết quả đánh giá cho thấy XGBoost đạt hiệu quả tốt nhất trên cả ba chỉ số RMSE, MAE và MAPE, đồng thời có tỷ lệ overfitting thấp nhất (1.11x), cho thấy khả năng tổng quát hóa tốt nhất trong ba mô hình.

XGBoost cải thiện đáng kể về độ chính xác dự báo so với ARIMA (65.6% về RMSE) và đạt hiệu quả tương đương với LSTM. XGBoost có nhiều ưu điểm phù hợp cho production use: độ chính xác cao, khả năng tổng quát hóa tốt, tốc độ huấn luyện nhanh, khả năng giải thích tốt, dễ triển khai, và độ tin cậy cao.

Dựa trên các tiêu chí lựa chọn mô hình (độ chính xác dự báo, khả năng tổng quát hóa, tốc độ huấn luyện, khả năng giải thích, khả năng triển khai, và độ tin cậy), XGBoost đạt tổng điểm cao nhất (4.80), vượt trội hơn đáng kể so với ARIMA (3.10) và LSTM (3.10).

Chúng tôi khuyến nghị sử dụng mô hình XGBoost cho production use trong bài toán dự báo lưu lượng truy cập web. XGBoost là lựa chọn tối ưu cho hầu hết các kịch bản, nhưng các mô hình khác có thể phù hợp hơn trong một số kịch bản đặc biệt như cần độ giải thích cao (ARIMA) hoặc cần nắm bắt các mẫu hình tuần tự dài hạn (LSTM).

Trong chương tiếp theo, chúng tôi sẽ sử dụng kết quả dự báo từ mô hình XGBoost để xây dựng thuật toán autoscaling tối ưu, giúp cân bằng giữa chi phí vận hành và chất lượng dịch vụ.

Chương 7

AUTOSCALING & DEMO

7.1 THUẬT TOÁN AUTOSCALING

7.1.1 Chiến lược scaling

Trong bài toán autoscaling, việc lựa chọn chiến lược phù hợp đóng vai trò quan trọng trong việc cân bằng giữa chi phí vận hành và chất lượng dịch vụ. Chúng tôi triển khai ba chiến lược scaling khác nhau để so sánh hiệu quả: **CPU-based scaling** (scaling phản ứng dựa trên tải CPU), **Request-based scaling** (scaling dựa trên số lượng request thực tế), và **Predictive scaling** (scaling dự báo sử dụng kết quả từ mô hình học máy).

CPU-based scaling là chiến lược phản ứng (reactive) đơn giản nhất, trong đó hệ thống điều chỉnh số lượng máy chủ dựa trên tải CPU hiện tại. Khi tải CPU vượt quá ngưỡng 70% trong 5 phút liên tiếp, hệ thống sẽ kích hoạt scale-out (tăng số lượng máy chủ). Ngược lại, khi tải CPU giảm dưới 30% trong 10 phút liên tiếp, hệ thống sẽ kích hoạt scale-in (giảm số lượng máy chủ). Chiến lược này có ưu điểm là đơn giản và dễ triển khai, nhưng có nhược điểm là chỉ phản ứng sau khi tải đã tăng, dẫn đến độ trễ trong việc đáp ứng traffic đột ngột.

Request-based scaling là chiến lược phản ứng khác, trong đó hệ thống tính toán server capacity và điều chỉnh số lượng máy chủ dựa trên số lượng request thực tế. Chúng tôi giả định mỗi máy chủ có thể xử lý khoảng 1000 request mỗi phút. Khi số lượng request dự báo vượt quá capacity của số lượng máy chủ hiện tại, hệ thống sẽ kích hoạt scale-out. Ngược lại, khi số lượng request giảm xuống dưới một ngưỡng an toàn, hệ thống sẽ kích hoạt scale-in. Công thức tính toán số lượng máy chủ tối ưu được biểu diễn như sau:

$$\text{optimal\_servers} = \lceil \frac{\text{forecast\_requests}}{\text{server\_capacity}} \rceil \tag{7.1}$$

Trong đó: `forecast_requests` là số lượng request dự báo cho thời điểm tiếp theo; `server_capacity` là số lượng request tối đa mà một máy chủ có thể xử lý mỗi phút (giả định là 1000); và $\lceil x \rceil$ là hàm làm tròn lên.

Predictive scaling là chiến lược tiên tiến nhất, trong đó hệ thống sử dụng kết quả dự báo từ mô hình học máy (XGBoost) để chủ động điều chỉnh tài nguyên trước khi traffic thực tế tăng. Khi mô hình dự báo traffic sẽ tăng hơn 20% trong 15 phút tới, hệ thống sẽ kích hoạt scale-out trước để đảm bảo đủ tài nguyên khi traffic tăng. Ngược lại, khi mô hình dự báo

traffic sẽ giảm đáng kể, hệ thống sẽ kích hoạt scale-in để tiết kiệm chi phí. Chiến lược này có ưu điểm là chủ động và giảm độ trễ phản ứng, nhưng phụ thuộc vào độ chính xác của mô hình dự báo.

Bảng 7.1 tóm tắt các chiến lược scaling được triển khai.

Bảng 7.1: So sánh các chiến lược autoscaling

Chiến lược	Loại	Ngưỡng scale-out	Ngưỡng scale-in
CPU-based	Reactive	CPU > 70% (5 phút)	CPU < 30% (10 phút)
Request-based	Reactive	Requests > capacity	Requests < 0.5 $\times$ capacity
Predictive	Proactive	Forecast +20% (15 phút)	Forecast -30% (15 phút)

7.1.2 Cơ chế Cooldown và Hysteresis

Để tránh hiện tượng dao động liên tục (oscillation) của số lượng máy chủ khi traffic biến động nhỏ, chúng tôi triển khai cơ chế *Cooldown* và *Hysteresis*. Cooldown là thời gian chờ sau mỗi action scaling, trong đó hệ thống không thực hiện thêm bất kỳ action nào. Hysteresis là cơ chế sử dụng hai ngưỡng khác nhau cho scale-out và scale-in, giúp tránh việc hệ thống liên tục thay đổi trạng thái khi traffic dao động quanh ngưỡng.

Cụ thể, chúng tôi thiết lập cooldown period là 10 phút sau mỗi action scaling. Sau khi scale-out hoặc scale-in được kích hoạt, hệ thống sẽ không thực hiện thêm bất kỳ action nào trong 10 phút tiếp theo, bất kể tải hiện tại như thế nào. Điều này giúp tránh việc hệ thống phản ứng quá nhanh với các biến động ngắn hạn của traffic.

Hysteresis được áp dụng bằng cách sử dụng hai ngưỡng khác nhau cho scale-out và scale-in. Ví dụ, trong chiến lược CPU-based, ngưỡng scale-out là 70% trong khi ngưỡng scale-in là 30%. Sự chênh lệch lớn giữa hai ngưỡng này đảm bảo rằng hệ thống không scale-in ngay sau khi scale-out khi traffic chỉ giảm nhẹ, giúp tránh hiện tượng flapping (dao động liên tục).

Cơ chế hysteresis được áp dụng bằng cách sử dụng hai ngưỡng khác nhau cho scale-out và scale-in. Ví dụ, trong chiến lược CPU-based, ngưỡng scale-out là 70% trong khi ngưỡng scale-in là 30%. Sự chênh lệch lớn giữa hai ngưỡng này đảm bảo rằng hệ thống không scale-in ngay sau khi scale-out khi traffic chỉ giảm nhẹ, giúp tránh hiện tượng flapping (dao động liên tục).

Trong hình này, đường nét đứt màu đỏ đại diện cho ngưỡng scale-out (70%), trong khi đường nét đứt màu xanh đại diện cho ngưỡng scale-in (30%). Khi traffic vượt qua ngưỡng scale-out, hệ thống scale-out và phải chờ 10 phút (cooldown) trước khi có thể thực hiện action tiếp theo. Sau đó, traffic phải giảm xuống dưới ngưỡng scale-in và chờ hết cooldown period thì hệ thống mới scale-in.

7.1.3 Phân tích chi phí

Để đánh giá hiệu quả của các chiến lược autoscaling, chúng tôi thực hiện phân tích chi phí dựa trên giả định unit cost là \$0.05 cho mỗi server-giờ. Chi phí được tính bằng công thức sau:

$$\text{total\_cost} = \sum_{t=1}^{T} \text{servers}_t \times \text{unit\_cost} \times \Delta t \tag{7.2}$$

Trong đó: `servers_t` là số lượng máy chủ tại thời điểm `t`; `unit_cost` là chi phí cho mỗi server-giờ (giả định là \$0.05); và Δt là khoảng thời gian giữa hai điểm dữ liệu (5 phút cho khung 5m).

Chúng tôi so sánh chi phí của ba chiến lược autoscaling với chiến lược **Static allocation** (cấp phát tài nguyên cố định với 10 máy chủ). Kết quả so sánh được tóm tắt trong Bảng 7.2.

Bảng 7.2: So sánh chi phí giữa các chiến lược autoscaling

Chiến lược	Số server trung bình	Tổng chi phí (\$)	Tiết kiệm (%)	Tỷ lệ overload (%)
Static (10 server)	10.00	720.00	0.0	0.0
CPU-based	6.42	462.24	35.8	2.1
Request-based	5.87	422.64	41.3	1.8
Predictive	5.21	375.12	47.9	0.9

Kết quả cho thấy chiến lược **Predictive scaling** đạt hiệu quả tốt nhất với tổng chi phí \$375.12, tiết kiệm 47.9% so với static allocation và chỉ có 0.9% thời gian hệ thống bị overload. Chiến lược **Request-based** xếp thứ hai với tổng chi phí \$422.64, tiết kiệm 41.3% so với static allocation. Chiến lược **CPU-based** xếp thứ ba với tổng chi phí \$462.24, tiết kiệm 35.8% so với static allocation.

Điều này cho thấy Predictive scaling có hiệu quả tốt nhất trong việc cân bằng giữa chi phí và chất lượng dịch vụ, nhờ khả năng chủ động điều chỉnh tài nguyên trước khi traffic tăng. Request-based scaling cũng đạt hiệu quả tốt, nhưng có độ trễ phản hồi cao hơn so với Predictive scaling. CPU-based scaling có hiệu quả thấp nhất do chỉ phản ứng sau khi tải đã tăng, dẫn đến nhiều thời điểm hệ thống bị overload.

7.2 DEMO HỆ THỐNG

7.2.1 Kiến trúc hệ thống

Hệ thống autoscaling được thiết kế theo kiến trúc phân tầng với các thành phần chính: **Data Layer**, **Processing Layer**, **Model Layer**, **API Layer**, và **Dashboard Layer**. Kiến trúc này đảm bảo tính mô-đun, dễ mở rộng và dễ bảo trì.

Data Layer chịu trách nhiệm lưu trữ và quản lý dữ liệu thô và dữ liệu đã được xử lý. Dữ liệu thô từ NASA HTTP logs được lưu trong thư mục `data/raw/`, trong khi dữ liệu đã được xử lý và aggregated được lưu trong thư mục `data/cleaned/`. Các mô hình đã được huấn luyện được lưu trong thư mục `models/` để sử dụng cho dự báo.

Processing Layer chịu trách nhiệm xử lý dữ liệu và tạo các đặc trưng cho mô hình. Các module trong lớp này bao gồm `data_loader.py` (đọc và parse logs), `features.py` (tạo đặc trưng), và `evaluation.py` (tính toán các chỉ số đánh giá). Lớp này đảm bảo dữ liệu được chuẩn hóa và làm sạch trước khi đưa vào mô hình.

Model Layer chứa các mô hình học máy đã được huấn luyện và logic dự báo. Các mô hình ARIMA, LSTM, và XGBoost được tải từ thư mục `models/` và sử dụng để dự báo số lượng

request trong tương lai. Lớp này cũng chứa logic autoscaling trong `optimization.py`, bao gồm các chiến lược scaling và cơ chế cooldown.

API Layer là giao diện giữa hệ thống và người dùng, cung cấp các endpoint để nhận request và trả về kết quả. API được xây dựng bằng FastAPI và cung cấp hai endpoint chính: `/forecast` để lấy dự báo lưu lượng truy cập và `/recommend-scaling` để lấy đề xuất số lượng máy chủ tối ưu.

Dashboard Layer là giao diện người dùng trực quan để hiển thị kết quả dự báo và các đề xuất autoscaling. Dashboard được xây dựng bằng Streamlit và cung cấp các biểu đồ tương tác để người dùng có thể khám phá dữ liệu và kết quả dự báo.

Kiến trúc hệ thống được thiết kế theo kiến trúc phân tầng với các thành phần chính: **Data Layer**, **Processing Layer**, **Model Layer**, **API Layer**, và **Dashboard Layer**. Kiến trúc này đảm bảo tính mô-đun, dễ mở rộng và dễ bảo trì.

7.2.2 API Endpoints

API của hệ thống autoscaling được xây dựng bằng FastAPI và cung cấp hai endpoint chính để phục vụ các nhu cầu khác nhau của người dùng. Endpoint `/forecast` cho phép người dùng lấy dự báo lưu lượng truy cập, trong khi endpoint `/recommend-scaling` cho phép người dùng lấy đề xuất số lượng máy chủ tối ưu.

Endpoint `/forecast` nhận các tham số đầu vào: **time_horizon** là thời gian dự báo (số bước thời gian trong tương lai); **model_type** là loại mô hình được sử dụng (ARIMA, LSTM, hoặc XGBoost); và **aggregation_window** là khung thời gian (1m, 5m, hoặc 15m). API trả về kết quả bao gồm: **predictions** là mảng các giá trị dự báo; **confidence_intervals** là khoảng tin cậy cho mỗi dự báo; và **metadata** là thông tin về mô hình và thời gian dự báo.

Ví dụ request đến endpoint `/forecast`:

```
GET /forecast?time_horizon=12&model_type=xgboost&aggregation_window=5m
```

Ví dụ response từ endpoint `/forecast`:

```
{
  "predictions": [245.3, 267.8, 289.2, ...],
  "confidence_intervals": [[220.1, 270.5], [242.6, 293.0], [264.1, 314.3], ...],
  "metadata": {
    "model_type": "xgboost",
    "aggregation_window": "5m",
    "forecast_time": "2025-08-23T00:00:00Z",
    "horizon": 12
  }
}
```

Endpoint `/recommend-scaling` nhận các tham số đầu vào: **forecast_data** là mảng các giá trị dự báo từ endpoint `/forecast`; **scaling_strategy** là chiến lược autoscaling (cpu_based, request_based, hoặc predictive); và **current_servers** là số lượng máy chủ hiện tại. API trả về kết quả bao gồm: **optimal_server_count** là số lượng máy chủ tối ưu; **scaling_action** là hành

động cần thực hiện (scale_out, scale_in, hoặc no_action); và **cost_estimate** là ước tính chi phí cho khoảng thời gian dự báo.

Ví dụ request đến endpoint `/recommend-scaling`:

```
POST /recommend-scaling
{
  "forecast_data": [245.3, 267.8, 289.2, ...],
  "scaling_strategy": "predictive",
  "current_servers": 5
}
```

Ví dụ response từ endpoint `/recommend-scaling`:

```
{
  "optimal_server_count": 7,
  "scaling_action": "scale_out",
  "cost_estimate": {
    "total_cost": 12.60,
    "servers_per_hour": [5, 5, 6, 7, 7, ...]
  }
}
```

API được thiết kế để dễ tích hợp với các hệ thống khác và có thể được sử dụng trong môi trường production. Các endpoint được document chi tiết trong Swagger UI, cho phép người dùng dễ dàng khám phá và thử nghiệm API.

7.2.3 Dashboard

Dashboard của hệ thống autoscaling được xây dựng bằng Streamlit và cung cấp giao diện người dùng trực quan để hiển thị kết quả dự báo và các đề xuất autoscaling. Dashboard cho phép người dùng khám phá dữ liệu, so sánh các mô hình khác nhau, và hiểu rõ cách hệ thống autoscaling hoạt động.

Dashboard bao gồm các thành phần chính sau. **Traffic Visualization** là biểu đồ hiển thị traffic theo thời gian, cho phép người dùng chọn khung thời gian (1m, 5m, hoặc 15m) và zoom vào các khoảng thời gian cụ thể. Biểu đồ hiển thị cả giá trị thực tế và giá trị dự báo, giúp người dùng dễ dàng so sánh.

Forecast Display là biểu đồ hiển thị dự báo từ mô hình đã chọn với confidence intervals. Người dùng có thể chọn mô hình (ARIMA, LSTM, hoặc XGBoost) và thời gian dự báo (số bước thời gian trong tương lai). Confidence intervals được hiển thị dưới dạng vùng màu xám quanh đường dự báo, cho thấy mức độ không chắc chắn của dự báo.

Scaling Recommendations là bảng hiển thị các đề xuất autoscaling dựa trên chiến lược đã chọn. Người dùng có thể chọn chiến lược (CPU-based, Request-based, hoặc Predictive) và xem số lượng máy chủ tối ưu, hành động cần thực hiện (scale-out, scale-in, hoặc no-action), và ước tính chi phí. Bảng cũng hiển thị timeline của các scaling events, cho phép người dùng hiểu rõ khi nào hệ thống scale-out hoặc scale-in.

Model Comparison là biểu đồ so sánh hiệu quả của các mô hình khác nhau trên các chỉ số RMSE, MAE, và MAPE. Người dùng có thể chọn khung thời gian (1m, 5m, hoặc 15m) và xem mô hình nào đạt hiệu quả tốt nhất cho khung thời gian đó.

Dashboard được thiết kế để dễ sử dụng và trực quan, cho phép người dùng không chuyên về kỹ thuật cũng có thể hiểu rõ cách hệ thống autoscaling hoạt động và đánh giá hiệu quả của các chiến lược khác nhau.

Dashboard được thiết kế để dễ sử dụng và trực quan, cho phép người dùng không chuyên về kỹ thuật cũng có thể hiểu rõ cách hệ thống autoscaling hoạt động và đánh giá hiệu quả của các chiến lược khác nhau.