ĐẠI HỌC BÁCH KHOA HÀ NỘI

BÀI TẬP LỚN
CHUỖI THỜI GIAN

Mô hình LSTM và mô hình VARMAX trong dự báo chất lượng không khí

Phạm Thị Hoa - 20195874

Lê Thanh Thảo - 20195919

Phạm Thu Trang - 20195931

Ngành: Toán Tin

Giảng viên hướng dẫn: TS. Nguyễn Thị Ngọc Anh

Chữ kí của GVHD

Viện: Toán ứng dụng và Tin học

HÀ NỘI, 3/2023

Lời cảm ơn

Nhóm chúng em xin gửi lời cảm ơn sâu sắc tới **TS.Nguyễn Thị Ngọc Anh**, người giảng viên đã tận tình chỉ bảo, luôn theo dõi sát sao và giúp đỡ chúng em trong quá trình nghiên cứu.

Chúng em cũng xin gửi lời cảm ơn đến Viện Toán ứng dụng và Tin học, Trường Đại học Bách Khoa Hà Nội đã cung cấp những kiến thức để tạo điều kiện thuận lợi cho chúng em hoàn thành đồ án này.

Chúng em xin chân thành cảm ơn!

Tóm tắt nội dung luận văn

Dự báo chuỗi thời gian là một lĩnh vực đã được nghiên cứu từ lâu và được nhiều người quan tâm. Tuy nhiên, trong những năm gần đây, Machine Learning và cụ thể hơn là Deep Learning đang phát triển mạnh mẽ, đạt được nhiều thành tựu vượt trội và thể hiện được nhiều tiềm năng ứng dụng trong chuỗi thời gian. Vì vậy, trong bài tập lớn này, chúng em sẽ tìm hiểu và sử dụng mô hình LSTM và mô hình VARMAX trong bài toán dự đoán chất lượng không khí để có cái nhìn cụ thể hơn về các mô hình này.

Từ khóa: *Deep Learning, Long Short-Term Memory (LSTM), Forecasting, Vector Auto Regressive Moving Average with Exogenous variables (VARMAX), Time Series Data..*

Hà Nội, ngày 7 tháng 3 năm 2023

Sinh viên thực hiện

Phạm Thị Hoa

Lê Thanh Thảo

Phạm Thu Trang

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

Từ viết tắt	**Ý nghĩa**
LSTM	Long short-term memory
VARMAX	Vector Auto Regressive Moving Average with Exogenous variables
RNN	Recurrent Neural Network
ML	Machine Learning
DL	Deep Learning

CHƯƠNG 1. MỞ ĐẦU

1.1 Lời mở đầu

Con người có thể sống sót trong 30 ngày mà không cần ăn, 3 ngày không uống, nhưng sẽ chết chỉ sau 3 phút nếu không thở. Nhu cầu về không khí của con người cũng không đổi, chúng ta hít thở nó thường xuyên dù trong nhà hay ngoài nhà, thậm chí phải chấp nhận cả khi chất lượng không được như mong đợi. Mùi khó chịu khiến con người nhận ra chất lượng không khí không tốt, nhưng nhiều khi khứu giác lại không thể cảm nhận được những tác nhân gây hại đang tồn tại, như các hóa chất độc hại – chúng ảnh hưởng vô cùng tồi tệ tới sức khỏe.

Ngày này chất lượng không khí tại các thành phố lớn ngày càng đi xuống. Bởi vậy việc đo lường, dự đoán chất lượng không khí tại một khu vực nhất định là vô cùng cần thiết. Nó cho ta biết chỉ số ô nhiễm tại nơi đó là bao nhiêu, nếu chỉ số quá cao sẽ ảnh hưởng rất lớn tới sức khỏe con người.

1.2 Giới thiệu bài toán

- Bài toán: Nhóm chúng em sẽ sử dụng mô hình LSTM và mô hình VARMAX để dự đoán chất lượng không khí trong 24h tới dựa vào các thông số thời tiết của các ngày trước đó như: nhiệt độ, độ ẩm, hướng gió, áp suất khí quyển, lượng bụi mịn PM10, PM 2.5, nồng độ khí CO, S02, NO trong không khí,...
- Phạm vi nghiên cứu: từ ngày 14/4/2015 tới ngày 5/9/2019.

1.3 Phân công công việc trong nhóm

Sinh viên	Công việc
Phạm Thị Hoa	Tìm hiểu code và ứng dụng mô hình LSTM vào bài toán
Lê Thanh Thảo	Tìm hiểu các mô hình và làm báo cáo
Phạm Thu Trang	Tìm hiểu code và ứng dụng mô hình VARMAX vào bài toán

Bảng 1.1 Phân công công việc

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

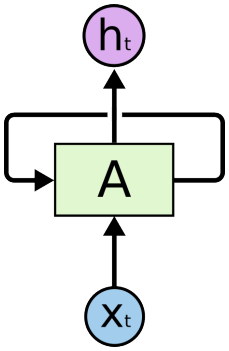
2.0.1 Mạng nơ-ron đệ quy (RNN)

Kiến trúc mạng nơ-ron đệ quy RNN

Con người không bắt đầu suy nghĩ của họ từ đầu tại tất cả các thời điểm. Cũng giống như việc đọc báo cáo đồ án này, chúng ta hiểu mỗi chữ ở đây dựa vào các chữ chúng ta đã hiểu trước đó chứ không phải đọc tới đâu ném hết đi tới đó, rồi bắt đầu suy nghĩ lại từ đầu tới chữ chúng ta đang đọc. Tức là tư duy đã có bộ nhớ để lưu lại những gì điễn ra trước đó.

Tuy nhiên các mô hình mạng nơ-ron truyền thống thì không thể làm được việc đó, đó có thể coi là một khuyết điểm chính của mạng nơ-ron truyền thống. Ví dụ, nếu chúng ta muốn phân loại các bối cảnh xảy ra ở tất cả các thời điểm trong một bộ phim, thì đúng là không thể hiểu được một tình huống trong phim mà lại phụ thuộc vào các tình huống trước đó nếu sử dụng các mạng nơ-ron truyền thống.

Mạng nơ-ron hồi quy (Recurrent Neural Network) sinh ra để giải quyết vấn đề đó. Mạng này chứa các vòng lặp bên trong cho phép thông tin có thể lưu lại được.

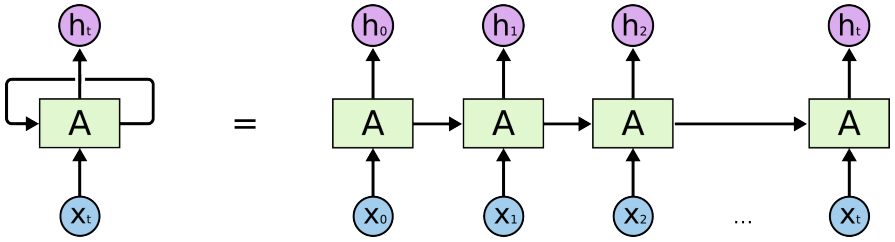


Hình 2.1 Recurrent Neural Networks have loops.

Hình vẽ trên mô tả một đoạn của mạng nơ-ron hồi quy $\mathcal{A}$ với đầu vào là

x_t và đầu ra là h_t. Một vòng lặp cho phép thông tin có thể được truyền từ bước này qua bước này qua bước khác của mạng nơ-ron.

Một mạng nơ-ron hồi quy có thể được coi là nhiều bản sao chép của cùng một mạng, trong đó mỗi đầu ra của mạng này là đầu vào của một mạng sao chép khác. Chuỗi lặp lại các mạng này chính là phân giải của mạng nơ-ron hồi quy, các vòng lặp khiến chúng tạo thành một chuỗi danh sách các mạng sao chép nhau.



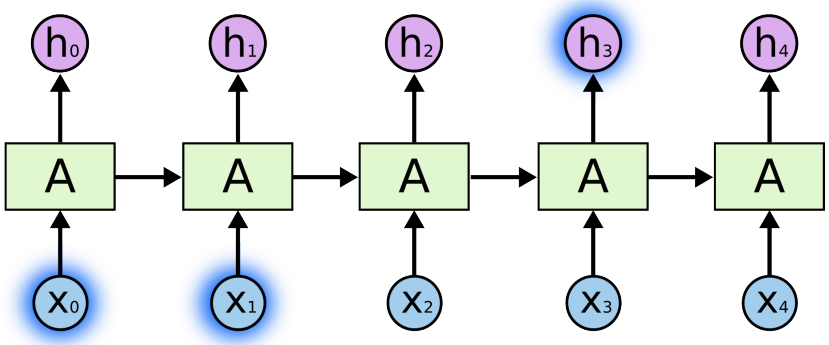
Hình 2.2 An unrolled recurrent neural network.

Trong vài năm gần đây, việc ứng dụng RNN đã đưa ra được nhiều kết quả không thể tin nổi trong nhiều lĩnh vực: nhận dạng giọng nói, mô hình hóa ngôn ngữ, dịch máy, mô tả ảnh,... Danh sách vẫn còn đang được mở rộng tiếp.

Vấn đề phụ thuộc xa

Một điểm nổi bật của RNN chính là ý tưởng kết nối các thông tin phía trước để dự đoán cho hiện tại. Việc này tương tự như ta sử dụng các cảnh trước của bộ phim để hiểu được cảnh hiện thời. Nếu mà RNN có thể làm được việc đó thì chúng sẽ cực kì hữu dụng, tuy nhiên liệu chúng có thể làm được không?

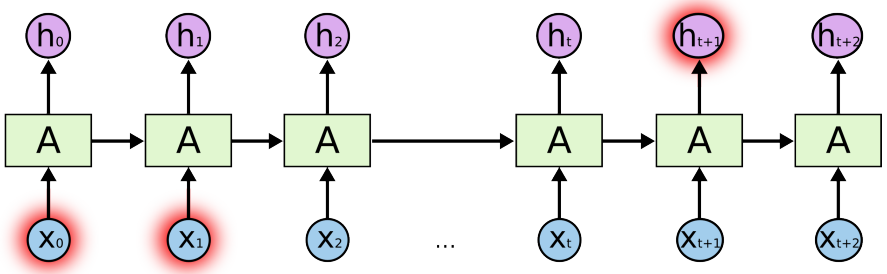
Đôi lúc ta chỉ cần xem lại thông tin vừa có thôi là đủ để biết được tình huống hiện tại. Ví dụ, ta có câu: *"các đám mây trên bầu trời"* thì ta chỉ cần đọc tới *"các đám mây trên bầu"* là đủ biết được chữ tiếp theo là *"trời"* rồi. Trong tình huống này, khoảng cách tới thông tin có được cần để dự đoán là nhỏ, nên RNN hoàn toàn có thể học được.



Hình 2.3 Mạng RNN phụ thuộc ngắn.

Nhưng trong nhiều tình huống ta buộc phải sử dụng nhiều ngữ cảnh hơn để suy luận. Ví dụ, dự đoán chữ cuối cùng trong đoạn: *"I grew up in France... I speak fluent French."*. Rõ ràng là các thông tin gần (*"I speak fluent"*) chỉ có phép ta biết được đằng sau nó sẽ là tên của một ngôn ngữ nào đó, còn không thể nào biết được đó là tiếng gì. Muốn biết là tiếng gì, thì ta cần phải có thêm ngữ cảnh *"I grew up in France"* nữa mới có thể suy luận được. Rõ ràng là khoảng cách thông tin lúc này có thể đã khá xa rồi.

Thật không may là với khoảng cách càng lớn dần thì RNN bắt đầu không thể nhớ và học được nữa.



Hình 2.4 Mạng RNN phụ thuộc dài.

Về mặt lý thuyết, rõ ràng là RNN có khả năng xử lý các phụ thuộc xa (long-term dependencies). Chúng ta có thể xem xét và cài đặt các tham số sao cho khéo là có thể giải quyết được vấn đề này. Tuy nhiên, đáng tiếc trong thực tế RNN có vẻ không thể học được các tham số đó. Vấn đề này đã được khám phá khá sâu bởi **Hochreiter (1991)** và **Bengio, et al (1994)** trong các bài báo của mình, họ đã tìm được nhưng lý do căn bản để giải thích tại sao RNN không thể học được.

Tuy nhiên, LSTM không vấp phải vấn đề đó.

2.1 LSTM

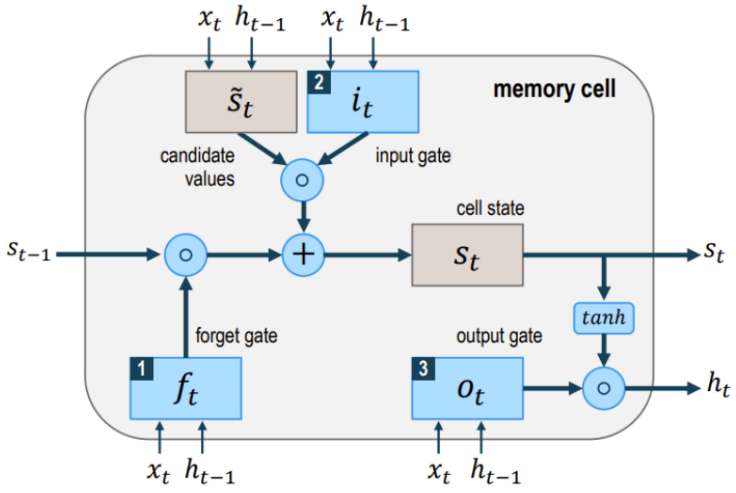
2.1.1 Mô hình Long Short Term Memory networks (LSTM)

Mạng bộ nhớ dài-ngắn (Long Short Term Memory networks), thường được gọi là LSTM - là một phiên bản mở rộng của mạng thần kinh hồi quy (RNN) nhân tạo được sử dụng trong lĩnh vực học sâu. LSTM được giới thiệu bởi **Hochreiter & Schmidhuber** vào năm 1997.

LSTM được đưa ra để giải quyết các vấn đề về phụ thuộc xa (long-term dependency) trong mạng RNN do bị ảnh hưởng bởi vấn đề mất gradient. Việc nhớ thông tin trong suốt thời gian dài là đặc tính mặc định của LSTM, chứ không cần phải huấn luyện để có thể nhớ được. Tức là ngay nội tại của nó đã có thể ghi nhớ được mà không cần bất kì can thiệp nào.

Một đơn vị LSTM thông thường bao gồm:

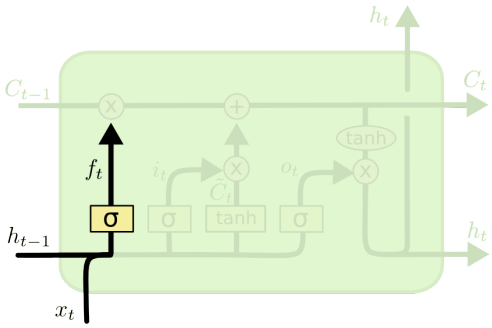
- **Tế bào (cell)**
- **Cổng quên (forget gate):** có nhiệm vụ loại bỏ những thông tin không cần thiết nhận được khỏi cell internal state.
- **Cổng vào (input gate):** có nhiệm vụ chọn lọc nhưng thông tin cần thiết nào được thêm vào cell internal state.
- **Cổng ra (output gate):** có nhiệm vụ xác định những thông tin nào từ cell internal state được sử dụng như đầu ra.



Hình 2.5 Sơ đồ biểu diễn kiến trúc bên trong của một tế bào LSTM

Kiến trúc bên trong LSTM

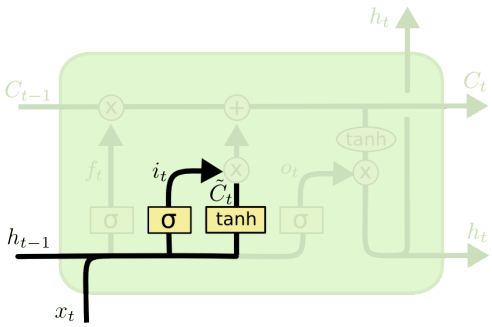
Bước đầu tiên của LSTM là quyết định xem thông tin nào cần bỏ đi từ trạng thái tế bào. Quyết định sẽ được tầng sigmoid đưa ra hay còn gọi là tầng cổng quên (forget gate layer). Nó lấy đầu vào là h_{t-1} và x_t rồi đưa ra kết quả nằm trong khoảng $[0, 1]$ cho mỗi giá trị trong trạng thái tế bào C_{t-1}. Đầu ra là 0 chỉ rằng toàn bộ thông tin sẽ bị bỏ đi, còn 1 thể hiện rằng nó giữ toàn bộ thông tin lại.



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

Hình 2.6 Tầng cổng quên.

Bước tiếp theo là quyết định xem thông tin mới nào ta sẽ lưu vào trạng thái tế bào. Việc này gồm 2 phần. Đầu tiên là sử dụng một tầng sigmoid - tầng cổng vào (input gate layer) để chọn giá trị cập nhật. Sau đó là tầng *tanh* sẽ tạo ra một vector giá trị mới $\tilde{C}_t$ mục đích là thêm vào cho trạng thái. Trong bước kế tiếp ta sẽ kết hợp 2 giá trị trên với nhau nhằm mục đích tạo ra một cập nhật trang thái.

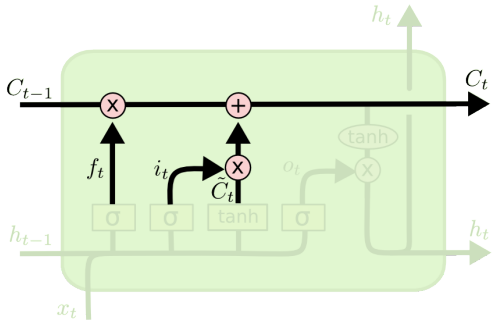


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Hình 2.7 Tầng cổng vào.

Giờ là lúc cập nhập trạng thái tế bào cũ C_{t-1} thành trạng thái mới C_t. Trong các bước trước đó đã quyết định các việc cần làm, nên bây giờ chỉ cần

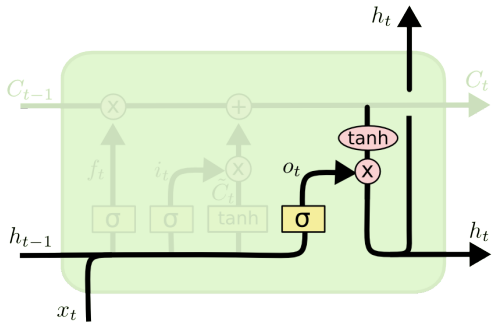
thực hiện.
Nhận trạng thái trước với f_t để loại bỏ được những thông tin đã quyết định quên. Sau đó cộng thêm $i_t * \tilde{C}_t$. Trạng thái mới thu được này phụ thuộc vào việc ta quyết định cập nhập mỗi giá trị trạng thái ra sao.



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

Hình 2.8 Cập nhật trạng thái tế bào.

Cuối cùng, ta cần xem xem đầu ra mong muốn là gì. Giá trị đầu ra sẽ dựa vào trạng thái tế bào và sẽ được tiếp tục sàng lọc. Trước tiên, ta chạy một tầng *sigmoid* để chọn phần nào của trạng thái tế bào ta muốn xuất ra. Sau đó, ta đưa nó trạng thái tế bảo qua một hàm *tanh* để co giá trị nó về khoảng $[-1, 1]$, và nhân nó với đầu ra của cổng sigmoid để được giá trị đầu ra ta mong muốn.



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

Hình 2.9 Cổng ra.

2.2 VARMAX

2.2.1 Giới thiệu mô hình VARMAX

Mô hình tự hồi quy trung bình trượt véc tơ với các biến hồi quy ngoại sinh (VARMAX). Đường tự hồi quy trung bình trượt véc tơ với các biến hồi quy

ngoại sinh (VARMAX) là một phần mở rộng của mô hình VARMA cũng bao gồm mô hình hóa các biến ngoại sinh. Nó là phiên bản đa biến của phương pháp ARMAX.

Các biến ngoại sinh còn được gọi là đồng biến và có thể được coi là các chuỗi đầu vào song song có các quan sát ở các bước đồng thời với chuỗi ban đầu. (Các) chuỗi chính được gọi là dữ liệu nội sinh để đối chiếu nó với (các) chuỗi ngoại sinh. Các quan sát đối với các biến ngoại sinh được đưa trực tiếp vào mô hình tại mỗi bước thời gian và không được mô hình hóa theo cách giống như chuỗi nội sinh chính (ví dụ như quy trình AR, MA, v.v.).

Mô hình VARMAX là một mô hình phù hợp cho chuỗi thời gian đa biến. Nó thêm thành phần trung bình cộng vào mô hình VAR và có thể cho phép các biến ngoại sinh. Bởi vậy, mô hình VARMAX gồm các thành phần:

- **VAR (Vector Autoregression):** thành phần tự hồi quy bao gồm tập hợp các độ trễ của biến hiện tại.
- **MA (Moving Average):** được hiểu là quá trình thay đổi hoặc dịch chuyển giá trị trung bình của một chuỗi theo thời gian.
- **X (Exogenous variable):** sử dụng các biến ngoại sinh.

Mô hình này có thể phù hợp với các quy trình phức tạp hơn nhiều mô hình chuỗi thời gian khác. Tuy nhiên, nó cũng có nhược điểm: thời gian train tương đối dài so với các mô hình đơn giản hơn và cần một lượng dữ liệu tương đối lớn để ước tính chính xác.

2.2.2 Mô hình VARMAX(p,q)

Mô hình toán học của VARMAX được định nghĩa như sau: Công thức tổng quát cho mô hình VARMAX:

$$X_t = \delta_t - \Sigma_{i=0}^{p} \phi_i X_{t-i} + \varepsilon_t + \Sigma_{i=1}^{q} \theta_i \varepsilon_{t-i}$$

hay $\phi(B)X_t = \delta_t + \theta(B)\varepsilon_t$
tại $\phi(B) = I_k - \Sigma_{i=1}^{p} \phi_i B^i$, $\theta(B) = I_k - \Sigma_{i=1}^{q} \theta_i B^i$
và δ_t có thể là một hằng số xác định (biến ngoại sinh).

Các thành phần của VARMAX:

- Auto regression: Kí hiệu là AR. Đây là thành phần tự hồi quy bao gồm tập hợp các độ trễ của biến hiện tại. Mô hình AR có thể được biểu diễn như sau:

$$y_t = c + \phi_1 y_{t-1} + \epsilon_t \tag{2.2.1}$$

Trong đó: y_t là giá trị tại thời gian t, c là hằng số, Φ_1 là hệ số còn ϵ_t là nhiễu trắng với $\epsilon_t \sim N(0, \sigma^2)$.

Độ trễ bậc p chính là giá trị lùi về quá khứ p bước thời gian của chuỗi. Độ trễ dài hoặc ngắn trong quá trình AR phụ thuộc vào tham số trễ p. Mô hình AR(p) của chuỗi y_t được biểu diễn như sau:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} ... + \phi_p y_{t-p} + \epsilon_t \tag{2.2.2}$$

hay

$$y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} \tag{2.2.3}$$

Trong đó Φ_i là hệ số tương ứng của mỗi giá trị $y_t - i$.

- Moving average: Quá trình trung bình trượt được hiểu là quá trình dịch chuyển hoặc thay đổi giá trị trung bình của chuỗi theo thời gian. Do chuỗi của chúng ta được giả định là dừng nên quá trình thay đổi trung bình dường như là 1 chuỗi nhiễu trắng. Quá trình moving average sẽ tìm mối liên hệ về mặt tuyến tính giữa các phần tử ngẫu nhiên ϵ_t chuỗi này là 1 chuỗi nhiễu trắng có các tính chất:

$$E(\epsilon_t) = 0 \tag{2.2.4}$$

$$\sigma(\epsilon_t) = \alpha \tag{2.2.5}$$

$$\rho(\epsilon_t, \epsilon_{t-s)} = 0, \forall t \geq s \tag{2.2.6}$$

Vế (2.2.4) có nghĩa rằng kì vọng của chuỗi dừng bằng 0 để đảm bảo chuỗi dừng không có sự thay đổi về trung bình theo thời gian. Vế (2.2.5) là phương sai của chuỗi không đổi. Do kì vọng và phương sai không đỏi nên chúng tagọi phân phối của nhiễu trắng là phân phối xác định và được kí hiệu $\epsilon_t \sim \mathcal{WN}(\ 0,\ \sigma^2)$. Nhiễu trắng là một thành phần ngẫu nhiên thể hiện cho yếu tố không thể dự báo của model và không có tính quy luật. Quá trình trung bình trượt được biểu diễn theo nhiễu trắng như sau:

$$MA(q) = \mu + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} \tag{2.2.7}$$

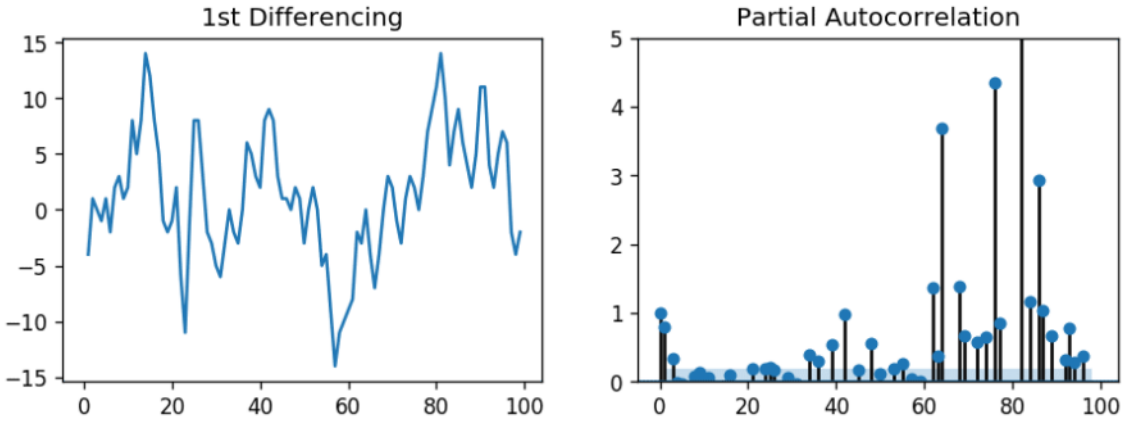
2.2.3 Cách xác định hệ số p, q

Cách xác định hệ số p của AR

Để kiểm tra hệ số q của mô hình AR ta kiểm tra biểu đồ tự tương quan 1 phần (PACF).

Tự tương quan một phần có thể được hình dung như mối tương quan giữa chuỗi và độ trễ của nó, sau khi loại trừ các đóng góp từ độ trễ trung gian. Vì vậy, PACF truyền tải mối tương quan thuần túy giữa độ trễ và chuỗi. Bằng cách đó, ta sẽ biết liệu độ trễ đó có cần thiết trong điều kiện AR hay không.

Bất kỳ sự tự tương quan nào trong một chuỗi dừng đều có thể được điều chỉnh bằng cách thêm đủ các thuật ngữ AR. Vì vậy, ban đầu ta coi thứ tự của số hạng AR bằng với càng nhiều độ trễ vượt qua giới hạn ý nghĩa trong biểu đồ PACF.

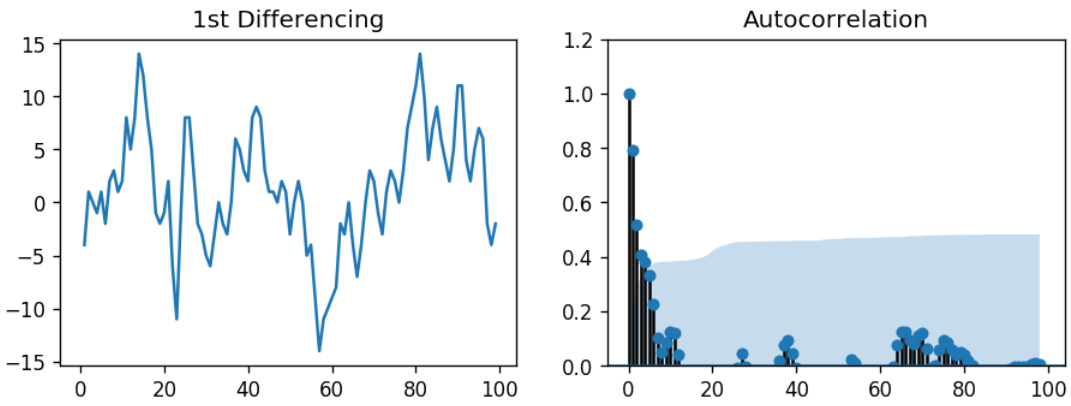


Hình 2.10 Ví dụ cách xác định p

Có thể quan sát thấy rằng độ trễ PACF bằng 1 là khá đáng kể vì nằm trên đường ý nghĩa (vùng màu xanh lam). Vì vậy ta chọn p bằng 1.

Cách xác định hệ số q của MA

Cũng giống như cách xem xét biểu đồ PACF cho hệ số của AR, chúng ta có thể xem biểu đồ ACF để biết hệ số của MA.



Hình 2.11 Ví dụ cách xác định q

Biểu đồ ACF cho ta biết cần phải thực hiện quá trình MA bao nhiêu lần để loại bỏ sự tự tương quan trong chuỗi thời gian dừng. Ở đây ta thấy nhiều là nằm trên đường ý nghĩa, ta có thể chọn giá trị q là 1 hoặc 2.

2.2.4 Ước lượng tham số mô hình sử dụng thuật toán đổi mới

Giả sử ta có một chuỗi ARMA(p,q) dừng với các quan sát $X_t, t = 1, 2, ..., N$. Mô hình ARMA(p,q) đưa ra bởi X_t:

$$\phi(B)X_t = \theta(B)Z_t \tag{2.2.8}$$

Với $\phi(B)$ và $\theta(B)$ là các đa thức bậc p và q

$$\phi(B) = 1 - \phi_1 B - ... - \phi_p B^p \tag{2.2.9}$$

và

$$\theta(B) = 1 + \theta_1 B + ...\theta_q B^q \tag{2.2.10}$$

Với B là toán tử lùi xác định bởi $(B^j X_t = X_{t-j}, B^j Z_t = Z_{t-j}, j = 0, \pm 1, ...)$, Z_t là chuỗi nhiễu trắng với kỳ vọng bằng 0 và phương sai là σ^2.

$$X_t - \phi_1 X_{t-1} - ... - \phi_p X_{t-p} = Z_t + \theta_1 Z_{t-1} + ...\theta_q Z_{t-q} \tag{2.2.11}$$

Khi p=0 thì biểu thức (2.2.11) là một chuỗi MA(q) còn nếu q=0 thì biểu thức (2.2.11) là một chuỗi AR(p). Brockwell và Davis (1987) đã đưa ra một thuật toán để ước tính các tham số AR và MA cho một mô hình ARMA (p, q). Theo

đó, với hàm tự tương phương sai đã biết(ACVF) $\gamma(.)$ thì các ước lượng đổi mới $\theta_{1,1}, \theta_{2,2}, \theta_{2,1}, \theta_{3,3}, \theta_{3,2}, ...$ có thể có được theo các biểu thức:

$$V_0 = \gamma(0) \tag{2.2.12}$$

$$\theta_{m,m-k} = V_k^{-1}[\gamma(m-k) - \sum_{j=0}^{k-1} \theta_{m,m-j}] \tag{2.2.13}$$

$$V_m = \gamma(0) - \sum_{j=0}^{m-1} \theta_{m,m-j} V_j \tag{2.2.14}$$

Nếu quá trình được cho bởi phương trình (2.2.10) là khả nghịch, thì nó có thể được biểu diễn dưới dạng:

$$X_t = \sum_{j=0}^{\infty} \psi_j W_{t-j} \tag{2.2.15}$$

Biểu thức (2.2.10) và (2.2.15) có thể được biểu diễn:

$$\psi_0 = 1 \tag{2.2.16}$$

$$\psi_j = \theta_j + \sum_{i=1}^{min(j,p)} \theta_i \psi_{j-1}, j = 1, 2, ... \tag{2.2.17}$$

Theo quy ước ta có $\theta_j = 0$ với j > q và $\phi_i = 0$ với i > p của chuỗi ARMA(p,q). Brockwell và Davis (1987) còn chứng minh thêm rằng $\psi_j \longrightarrow \theta_{m,j}$. Do đó biểu thức(2.2.17) được viết dưới dạng:

$$\theta_{m,j} = \theta_j + \sum_{i=1}^{min(j,p)} \phi_i \theta_{m,j-1}, j = 1, 2, ...p + q \tag{2.2.18}$$

Với j=q+1, q+2,...q+p trong biểu thức (2.2.18) một hệ p phương trình có thể được tạo ra và chúng có dạng:

$$\begin{bmatrix} \theta_{m,q+1} \\ \theta_{m,q+2} \\ \vdots \\ \theta_{m,q+p} \end{bmatrix} = \begin{bmatrix} \theta_{m,q} & \theta_{m,q-1} & \cdots & \theta_{m,q+1-p} \\ \theta_{m,q+1} & \theta_{m,q} & \cdots & \theta_{m,q+2-p} \\ \vdots & \vdots & \cdots & \vdots \\ \theta_{m,q+p-1} & \theta_{m,q+p-2} & \cdots & \theta_{m,q} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_p \end{bmatrix} \tag{2.2.19}$$

Các giá trị của $(\phi_1, \phi_2, ...\phi_p)$ có thể được xác định bởi biểu thức(2.2.19). Từ biểu thức (2.2.18), θ_j có thể được xác định bởi phương trình:

$$\theta_j = \theta_{m,j} - \sum_{i=1}^{min(j,p)} \phi_i \theta_{m,j-i}, j = 1, 2, ...q \tag{2.2.20}$$

Các kỹ thuật ở trên có thể được sử dụng để xác định tham số của AR và MA $\phi_1, \phi_2, ...\phi_p, \theta_1, \theta_2, ..., \theta_q$ trong mô hình ARMA.

CHƯƠNG 3. CÀI ĐẶT VÀ ĐÁNH GIÁ KẾT QUẢ

3.1 Mô tả dữ liệu

Để thực hiện dự đoán trên hai mô hình LSTM và VARMAX, chúng em sẽ thực nghiệm trên dữ liệu về chỉ số môi trường không khí.

- Nguồn thu thập: Được cung cấp bởi giảng viên bộ môn
- Thời gian của dữ liệu: 142 ngày từ 7:00 14/4/2019 đến 6:00 5/9/2019 tính theo từng giờ.
- Kích thước dữ liệu: 556KB
- Giá trị cần dự đoán: mô hình dự đoán tất cả các giá trị trong bộ dữ liệu

	Barometer	Temp	NO	PM-10	RH	Radiation	WindDir	SO2	NOx	NO2	...	CO	PM-1	O3	Wind Spd	TSP	time	WinDir1	Temp1	ASKQ	Wind Spd (sai)
0	1010.280000	25.550000	18.57	18.12	98.55	0.01	188.880000	19.17	68.232	78.45	...	461.82	3.43	-17.17	1.480000	31.02	14/4/2019 07:00:00	NaN	NaN	NaN	NaN
1	1011.210000	25.170000	11.61	17.25	99.62	32.83	194.240000	13.49	48.492	58.17	...	595.40	3.43	-3.70	1.550000	31.13	14/4/2019 08:00:00	NaN	NaN	NaN	NaN
2	1012.360000	25.860000	8.79	12.04	97.46	311.73	108.600000	6.68	30.564	34.38	...	553.42	2.43	3.09	1.240000	21.29	14/4/2019 09:00:00	NaN	NaN	NaN	NaN
3	1012.660000	27.670000	11.33	10.91	88.45	532.91	119.740000	5.77	30.996	31.16	...	446.55	2.14	8.60	1.780000	19.24	14/4/2019 10:00:00	NaN	NaN	NaN	NaN
4	1012.780000	29.140000	5.50	8.91	77.99	610.51	116.880000	5.72	23.676	28.68	...	435.10	1.87	33.08	1.560000	18.48	14/4/2019 11:00:00	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3409	1006.197202	24.618810	2.31	12.86	62.53	NaN	177.290079	70.69	2.448	0.29	...	122.13	2.98	91.97	0.994953	23.62	5/9/2019 2:00	103.96	30.69	1002.97	2.46
3410	1004.847083	27.431410	2.30	10.30	62.61	NaN	131.708012	72.00	2.400	0.23	...	114.50	3.01	90.53	1.789547	18.27	5/9/2019 3:00	84.25	30.64	1002.61	2.57
3411	1004.401372	24.187061	2.47	10.48	62.80	NaN	148.633002	73.61	2.520	0.17	...	110.68	2.98	88.60	1.083361	17.82	5/9/2019 4:00	74.20	30.63	1002.26	3.08
3412	1009.252225	30.123856	2.48	11.45	63.54	NaN	211.588451	73.22	2.388	NaN	...	122.13	3.07	88.27	1.401751	18.58	5/9/2019 5:00	91.78	30.46	1002.10	2.75
3413	1004.019334	27.841429	2.55	14.54	70.41	NaN	109.182176	77.79	4.584	3.27	...	232.82	3.70	76.07	1.305240	24.03	5/9/2019 6:00	249.09	29.01	1002.14	1.58

3414 rows × 22 columns

Hình 3.1 Dữ liệu môi trường không khí từ 7:00 14/4/2019 đến 6:00 5/9/2019

Mô tả chi tiết tên các cột :

- Barometer: hay còn được gọi với thuật ngữ quen thuộc là áp kế, là tính năng dùng để đo áp suất khí quyển ở vị trí hiện tại của đồng hồ. Đơn vị đo áp suất phổ biến hiện nay là (hPa) = hectopascal, ngoài ra còn 2 đơn vị được sử dụng nhiều là kilopascal (kPa) và atmosphere (atm).

- Temp : là nhiệt độ đơn vị độ C

- NO : Nitơ monoxide, hay còn gọi là nitric oxide (công thức hóa học: NO) là chất khí không màu, không bền trong không khí vì bị oxy oxy hóa ở nhiệt độ thường tạo ra nitơ dioxide là chất khí màu nâu đỏ. NO được tạo ra từ năng lượng sấm sét

- NO2 : Nitrogen dioxide (NO 2) là một chất góp phần chính trong việc hình thành sương mù và là tiền thân của nhiều chất ô nhiễm thứ cấp có hại, bao gồm ozone và các chất dạng hạt. Nó phản ứng mạnh với các hóa chất khác và là một chất oxy hóa mạnh. Nitơ đioxit là một chất khí có màu đỏ cam đậm.

- NOx : là chất gây ô nhiễm chính trong khí quyển, là tác nhân của mưa axit, sương khói quang hóa và tích tụ ozone. Các oxit chủ yếu là nitơ monoxit (NO) và nitơ dioxit (NO2) cả hai đều ăn mòn và nguy hiểm cho sức khỏe.

- PM-10 : PM-10 là những hạt rất nhỏ được tìm thấy trong bụi và khói . Chúng có đường kính 10 micromet (0,01 mm) hoặc nhỏ hơn. Các hạt PM-10 là một chất gây ô nhiễm không khí phổ biến.

- RH : Độ ẩm tương đối (RH) là thước đo lượng hơi nước trong hỗn hợp nước-không khí so với lượng tối đa có thể. RH là tỷ lệ giữa tỷ lệ độ ẩm của hỗn hợp nước-không khí cụ thể so với tỷ lệ độ ẩm bão hòa ở một nhiệt độ nhất định.

- Radition : Bức xạ khí quyển là dòng năng lượng điện từ giữa mặt trời và bề mặt trái đất khi nó bị ảnh hưởng bởi các đám mây và khí trong bầu khí quyển của trái đất. Nó bao gồm cả bức xạ mặt trời (ánh sáng mặt trời) và bức xạ nhiệt.

- SO2 : Lưu huỳnh dioxit là chất gây ô nhiễm không khí phản ứng, không màu, có mùi nồng . Loại khí này có thể là mối đe dọa đối với sức khỏe con người, sức khỏe động vật và đời sống thực vật. Các nguồn phát thải sulfur dioxide chính là từ quá trình đốt cháy nhiên liệu hóa thạch và hoạt động núi lửa tự nhiên.

- PM-2-5: là những hạt bụi li ti có trong không khí với kích thước 2,5 micron trở xuống (so với sợi tóc con người thì nó nhỏ hơn khoảng 30 lần). Bụi mịn pm2. 5 được hình thành từ các chất như nitơ, carbon và các hợp chất kim loại khác. Nó chất gây ô nhiễm không khí gây lo ngại cho sức khỏe của mọi người khi nồng độ trong không khí cao.

- CO: Cacbon monoxit là một chất khí vô hình, không hề có màu sắc, vô vị, không hề có mùi, và nó là một chất độc nguy hiểm với độc tính cao. Nó là sản phẩm chính của quá trình đốt cháy nhiên liệu trong môi trường không đủ oxy.

- Là những hạt bụi dạng lỏng, hoặc rắn trôi nổi ngoài không khí. PM1 được coi là đặc biệt nguy hiểm do kích thước cực nhỏ của nó. Đường kính của hạt càng nhỏ thì nó càng gây ra nhiều tác hại.

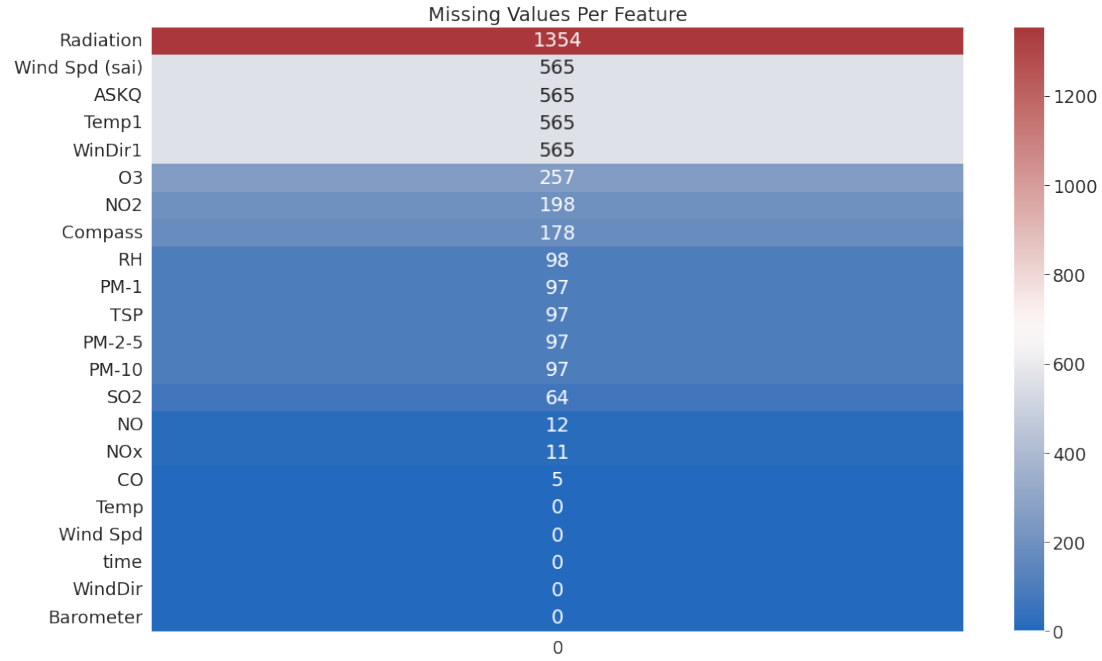
- O3 : là một chất khí màu xanh lam nhạt, có mùi hăng đặc biệt. Nó là một dạng thù hình của oxy kém bền hơn nhiều so với dạng nguyên tử O2, bị phá vỡ trong bầu khí quyển thấp hơn thành O2.

- Wind Spd : Tốc độ gió hay còn gọi là tốc độ luồng gió, là một đại lượng khí quyển cơ bản do không khí di chuyển từ nơi có áp suất cao đến áp suất thấp.

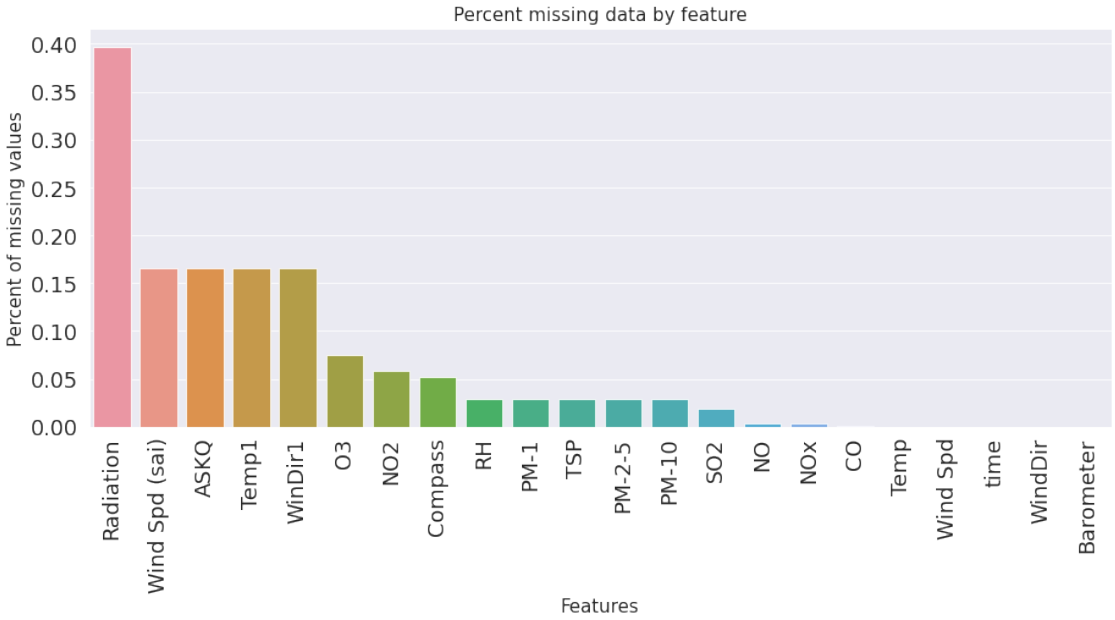
- TSP: Tổng bụi lơ lửng (TSP) là tổng các hạt bụi có đường kính khí động học nhỏ hơn hoặc bằng 100 micromet.

3.2 Xử lý dữ liệu

Vấn đề của bộ dữ liệu : có khá nhiều dữ liệu bị khuyết thiếu.



Hình 3.2 Số lượng dữ liệu khuyết thiếu của từng cột



Hình 3.3 Tỷ lệ dữ liệu khuyết thiếu của từng cột

Nhận thấy tỷ lệ dữ liệu khuyết thiếu của từng cột cũng không cao lắm nên có thể giữ lại và sẽ sử dụng Linear-Regression để dự đoán các giá trị còn thiếu như sau:

```
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()

lst = ['CO', 'NOx','NO','SO2','PM-10','PM-2-5','TSP','PM-1','RH','Compass', 'NO2'
    ,'O3','WinDir1','Temp1','ASKQ','Wind Spd (sai)','Radiation']
data = df[['Barometer','Wind Spd','Temp','WindDir']]
for a in lst:
    data = pd.concat([data, df[a]], axis=1)
    x_train = data[data[a].notnull()].drop(columns=a)
    y_train = data[data[a].notnull()][a]
    x_test = data[data[a].isnull()].drop(columns=a)
    y_test = data[data[a].isnull()][a]
    linreg.fit(x_train, y_train)
    predicted = linreg.predict(x_test)
    data[a][data[a].isnull()] = predicted
    df[a][df[a].isnull()] = predicted
    print("done")
```

Trong đó:

- **lst** là 1 mảng chứa tên các cột bị thiếu dữ liệu được sắp xếp theo số lượng tăng dần.
- **data** chứa các cột có đủ dữ liệu.

Thực hiện:

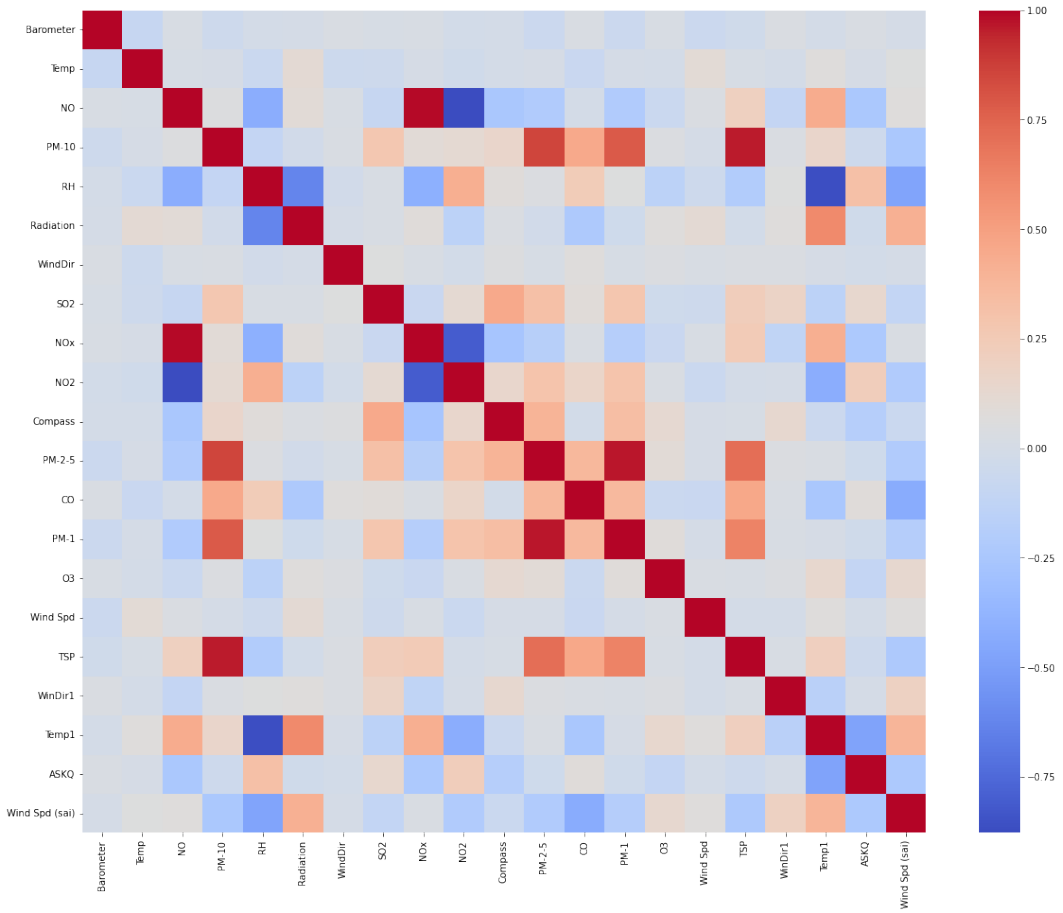
- Ta sử dụng vòng lặp for để dự đoán các cột có số lượng dữ liệu bị thiếu ít hơn trước.
- Trong vòng lặp for, ta thêm 1 cột khuyết thiếu dữ liệu vào **data**
- Chia dữ liệu thành tập train và test:
  - train: chứa các dòng dữ liệu đầy đủ.
  - test: chứa các dòng dữ liệu bị khuyết thiếu.
- Sử dụng mô hình dự đoán Linear-Regression để đưa ra kết quả dự đoán của tập test và tiếp tục quá trình nếu còn thỏa mãn vòng lặp.

Kết quả thu được:

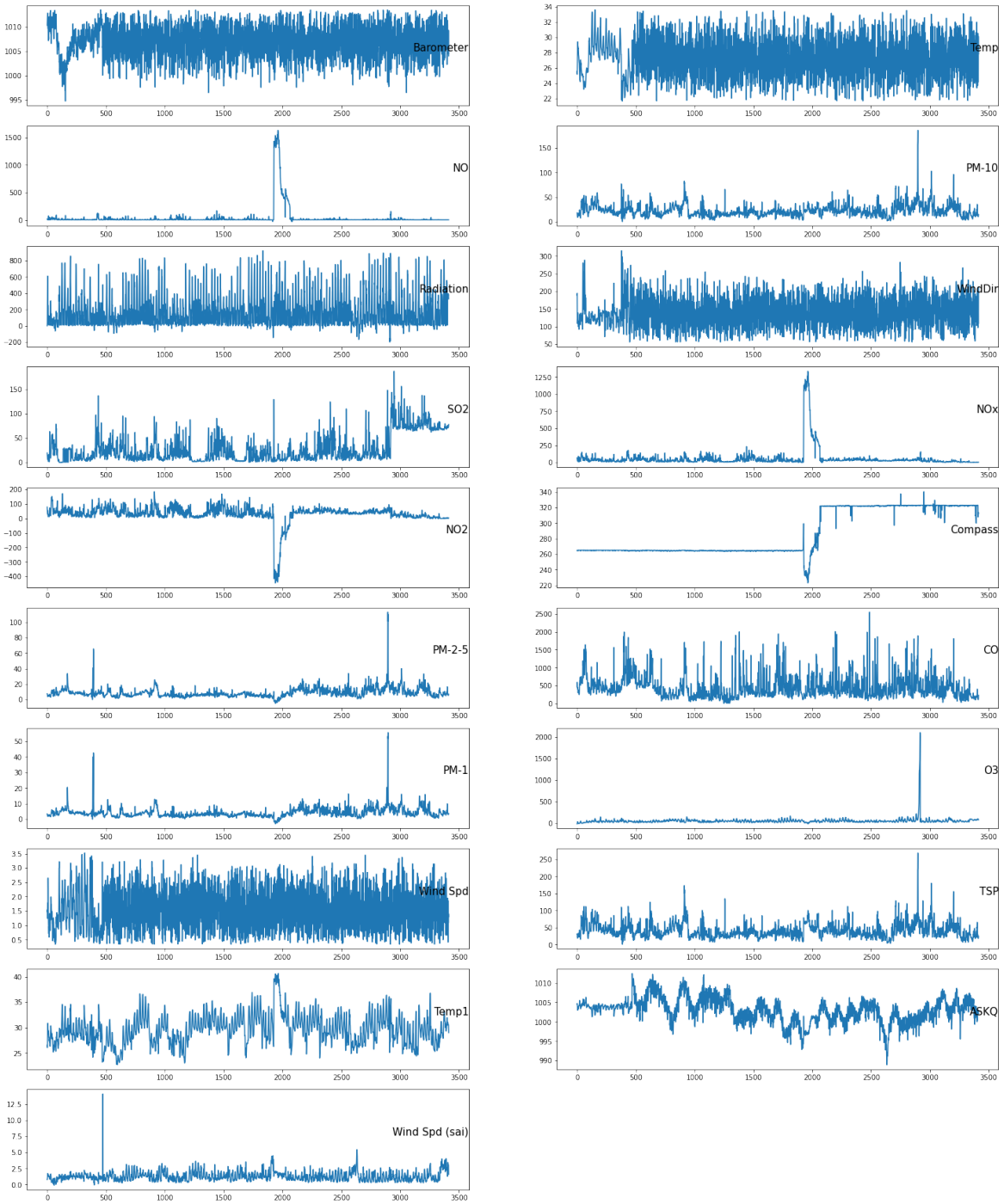
```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3414 entries, 0 to 3413
Data columns (total 22 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Barometer       3414 non-null   float64
 1   Temp            3414 non-null   float64
 2   NO              3414 non-null   float64
 3   PM-10           3414 non-null   float64
 4   RH              3414 non-null   float64
 5   Radiation       3414 non-null   float64
 6   WindDir         3414 non-null   float64
 7   SO2             3414 non-null   float64
 8   NOx             3414 non-null   float64
 9   NO2             3414 non-null   float64
 10  Compass         3414 non-null   float64
 11  PM-2-5          3414 non-null   float64
 12  CO              3414 non-null   float64
 13  PM-1            3414 non-null   float64
 14  O3              3414 non-null   float64
 15  Wind Spd        3414 non-null   float64
 16  TSP             3414 non-null   float64
 17  time            3414 non-null   object
 18  WinDir1         3414 non-null   float64
 19  Temp1           3414 non-null   float64
 20  ASKQ            3414 non-null   float64
 21  Wind Spd (sai)  3414 non-null   float64
dtypes: float64(21), object(1)
memory usage: 586.9+ KB
```

Hình 3.4 Thông tin dữ liệu sau khi xử lý



Hình 3.5 Heatmap



Hình 3.6 Trực quan hóa dữ liệu

3.3 Tiêu chuẩn đánh giá mô hình

3.3.1 Mean Squared Error (MSE)

MSE được hiểu là giá trị sai số bình phương trung bình hoặc là lỗi bình phương trung bình. Nó đề cập đến giá trị trung bình của chênh lệch bình phương giữa tham số dự đoán và tham số quan sát được và có công thức như sau:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.3.1}$$

với y_i là giá trị thực sự cần dự đoán, và $\hat{y}_i$ là giá trị mô hình dự đoán, n là kích thước của dữ liệu cần dự đoán.

3.3.2 Root Mean Squared Error (RMSE)

RMSE là thước đo mức độ hiệu quả của mô hình của bạn. Nó thực hiện điều này bằng cách đo sự khác biệt giữa các giá trị dự đoán và giá trị thực tế . R-MSE càng nhỏ tức là sai số càng bé thì mức độ ước lượng cho thấy độ tin cậy của mô hình có thể đạt cao nhất và có công thức tính là:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.3.2}$$

với y_i là giá trị thực sự cần dự đoán, và $\hat{y}_i$ là giá trị mô hình dự đoán, n là kích thước của dữ liệu cần dự đoán.

3.3.3 Sử dụng *model.plot_ diagnostics()*

Tạo lưới ô 2x2 chứa các kết quả đánh giá mô hình:

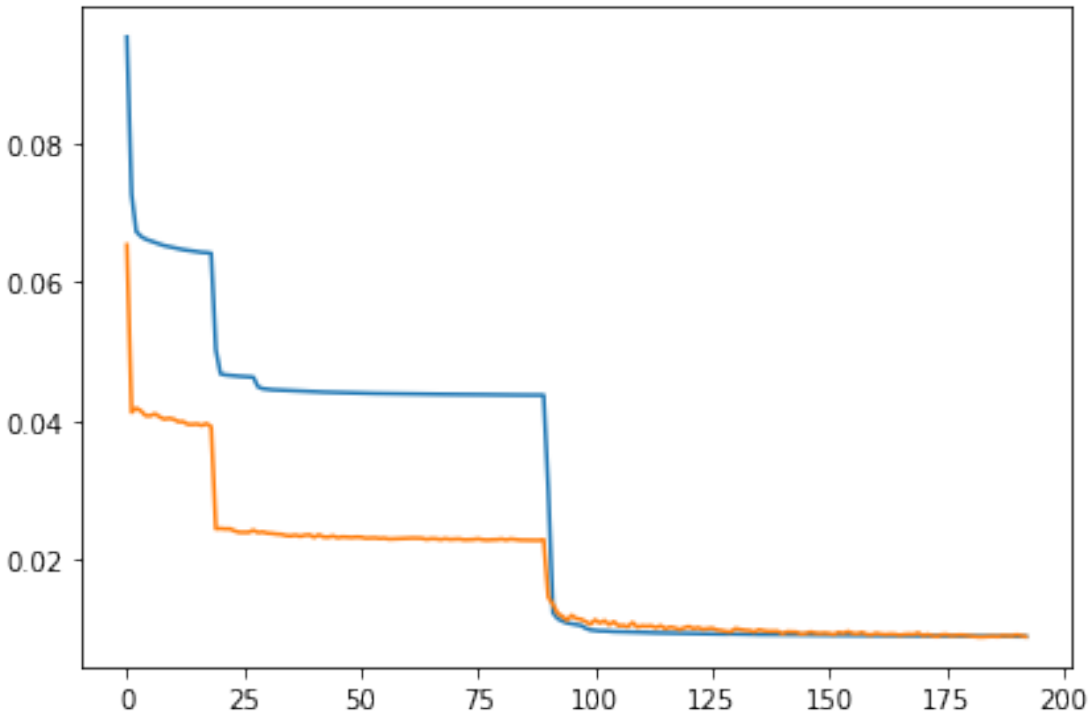
1. Standardized residuals
2. Histogram plus estimated density
3. Normal Q-Q
4. Correlogram

3.4 Kết quả

3.4.1 Mô hình LSTM

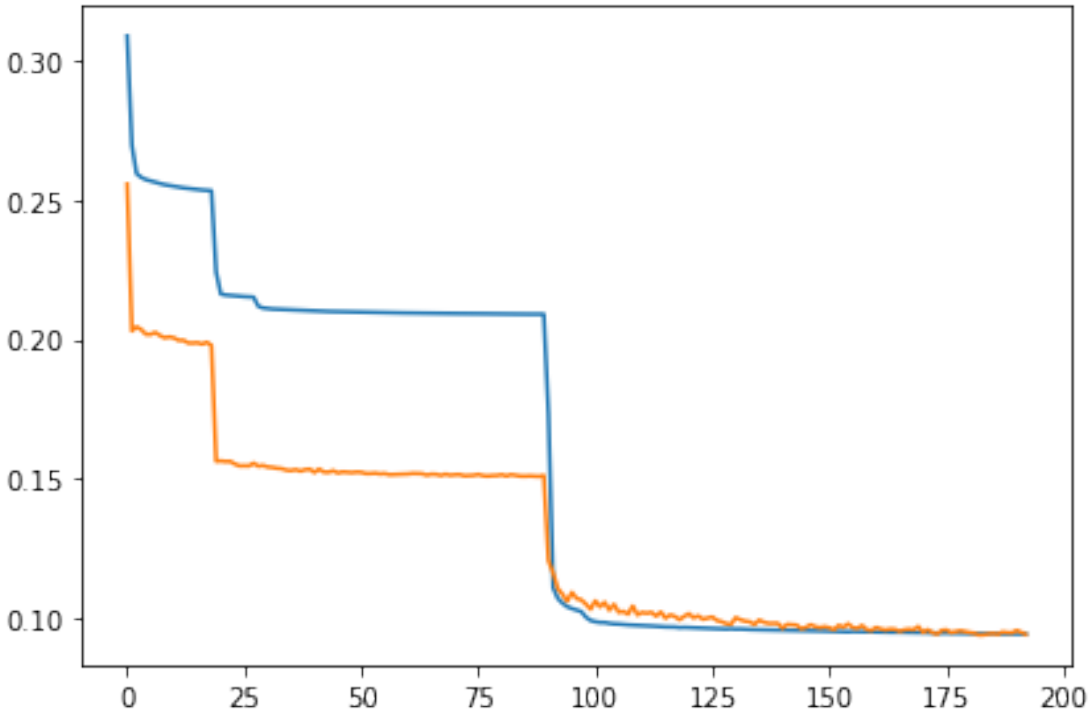
Đánh giá mô hình :

- Loss: được tính bằng MSE, LOSS của mô hình sau khi huấn luyện qua 200 epoch



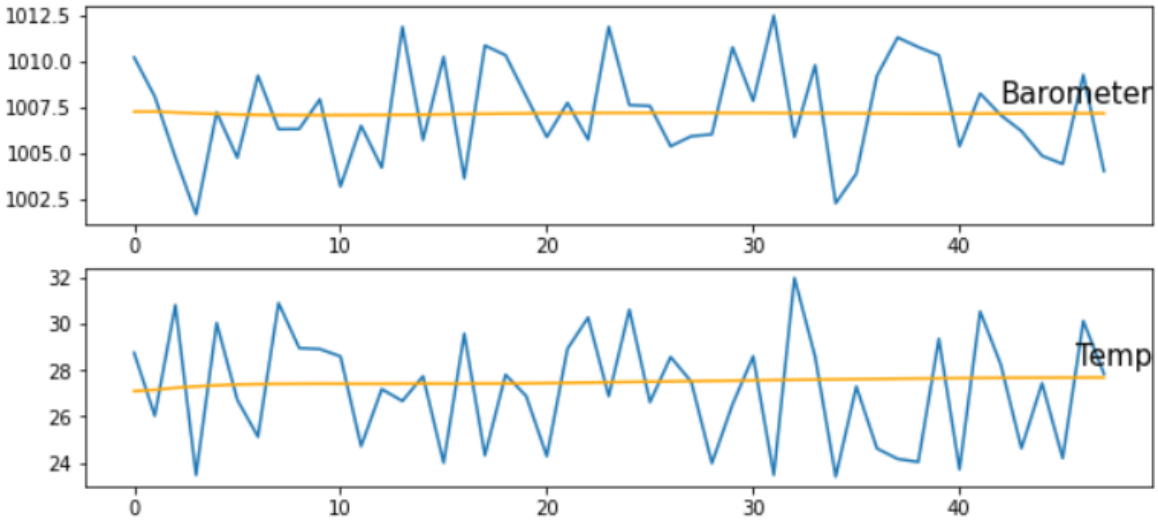
Hình 3.7 LOSS của mô hình LSTM

- RMSE của mô hình sau khi huấn luyện qua 200 epoch:

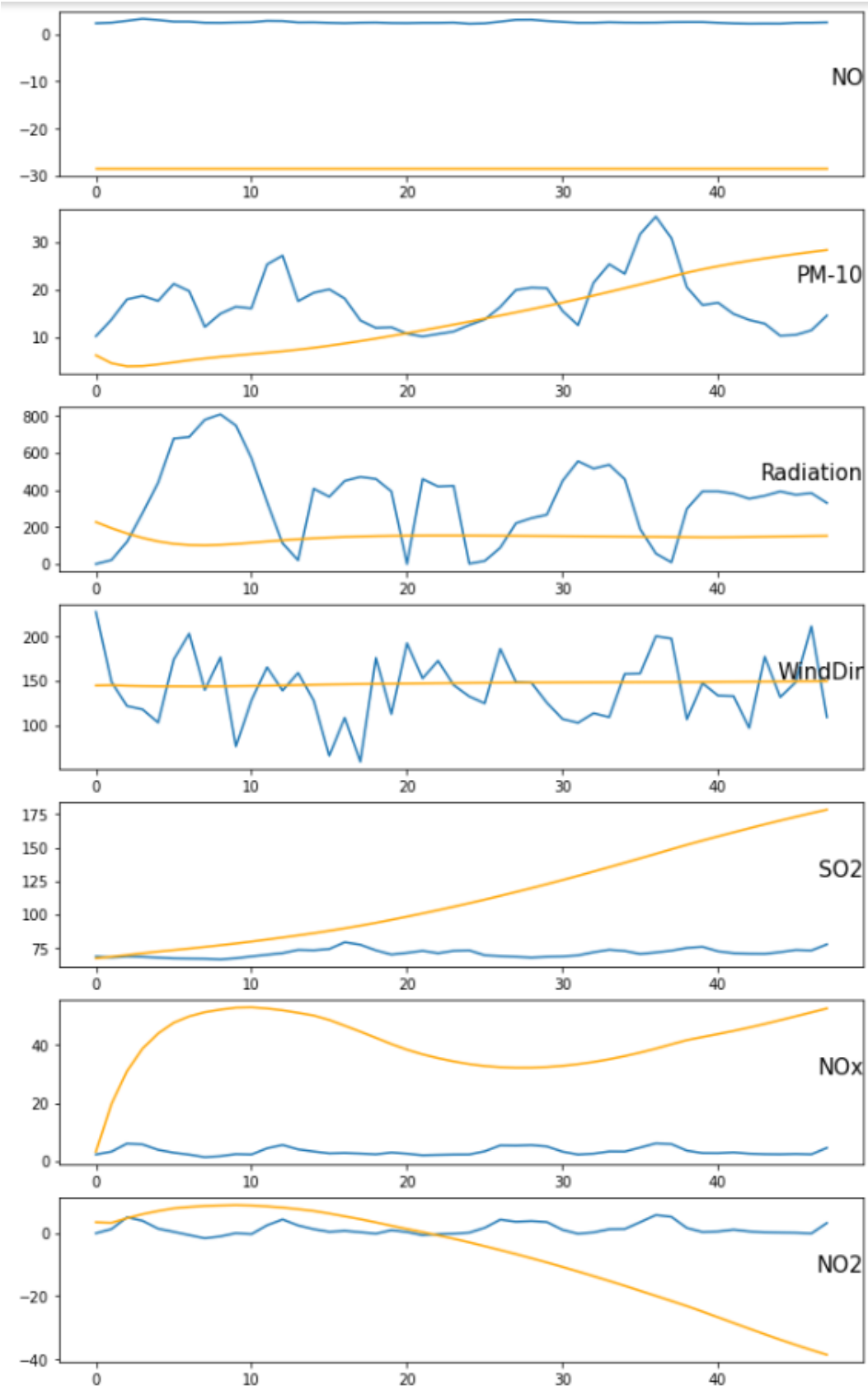


Hình 3.8 RMSE của mô hình LSTM

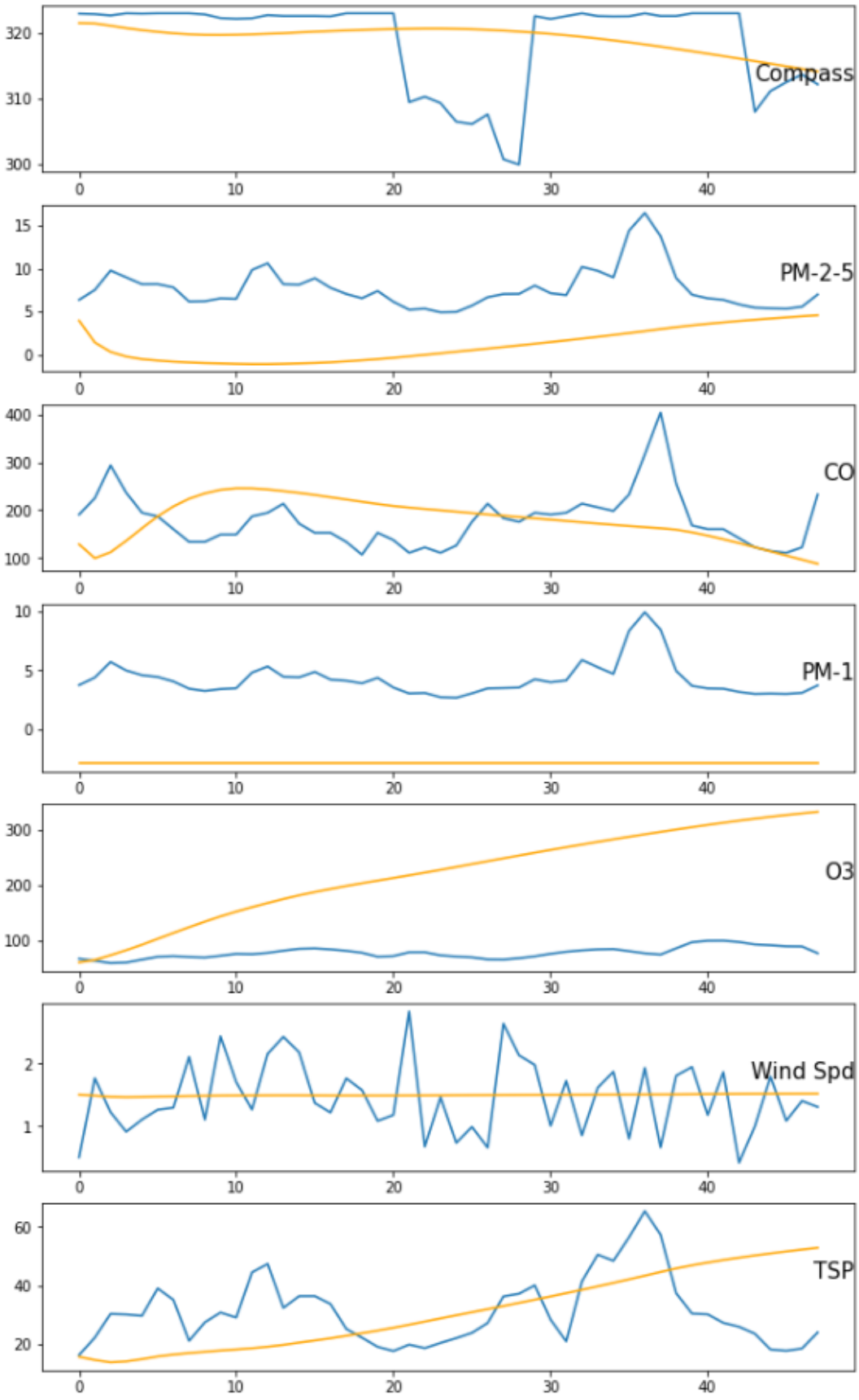
Kết quả dự đoán của mô hình với 48 giờ cuối của bộ dữ liệu so với giá trị thực trên tập dữ liệu:



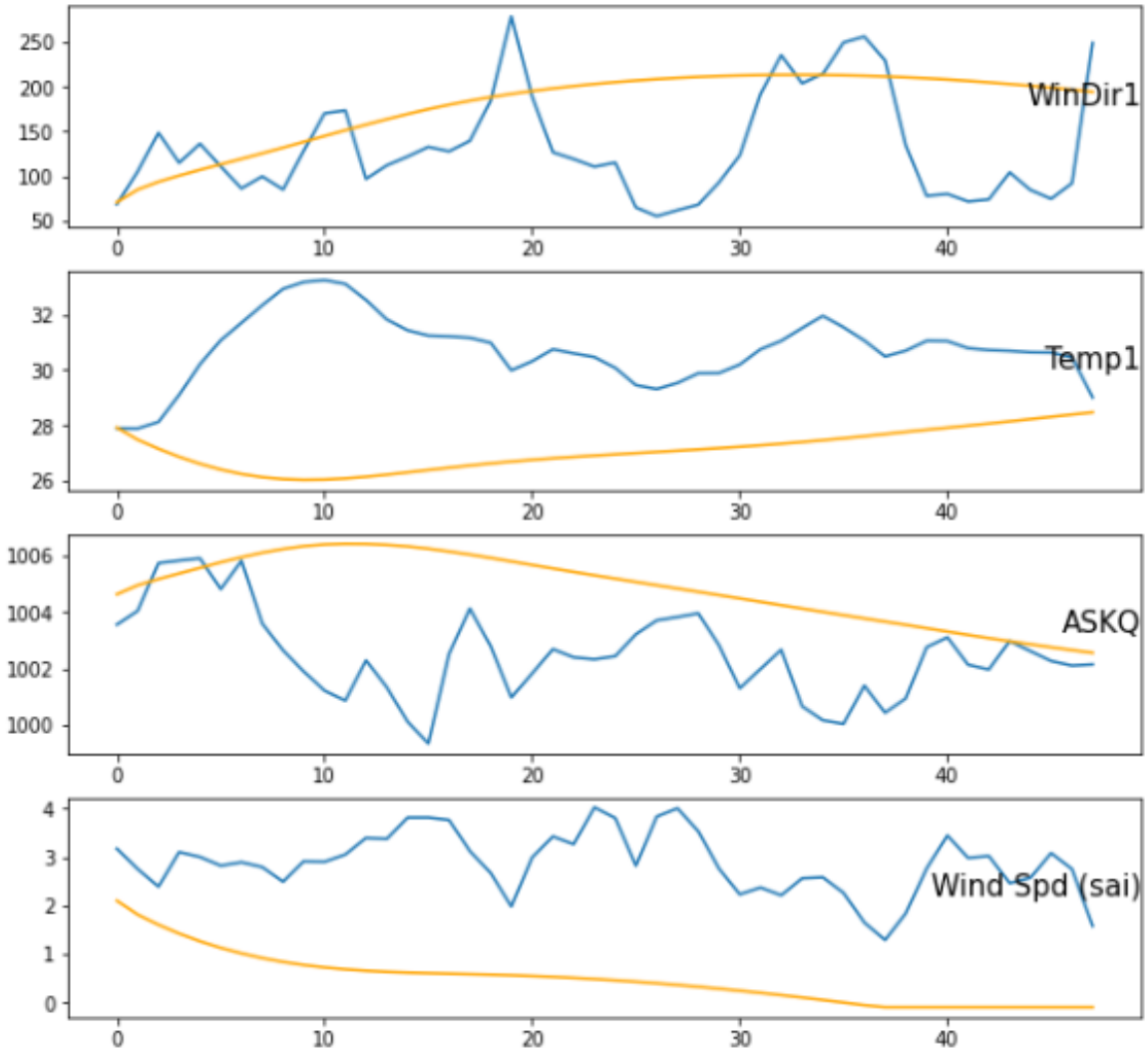
Hình 3.9 Barometer và Temp



Hình 3.10 NO, PM-10, Radition, WindDir, SO2, NOx và NO2



Hình 3.11 Compass, PM-2-5, CO, PM-1, O3, Wind Spd và TSP



Hình 3.12 WinDir1, Temp1, ASKQ, Wind-Spd(sai)

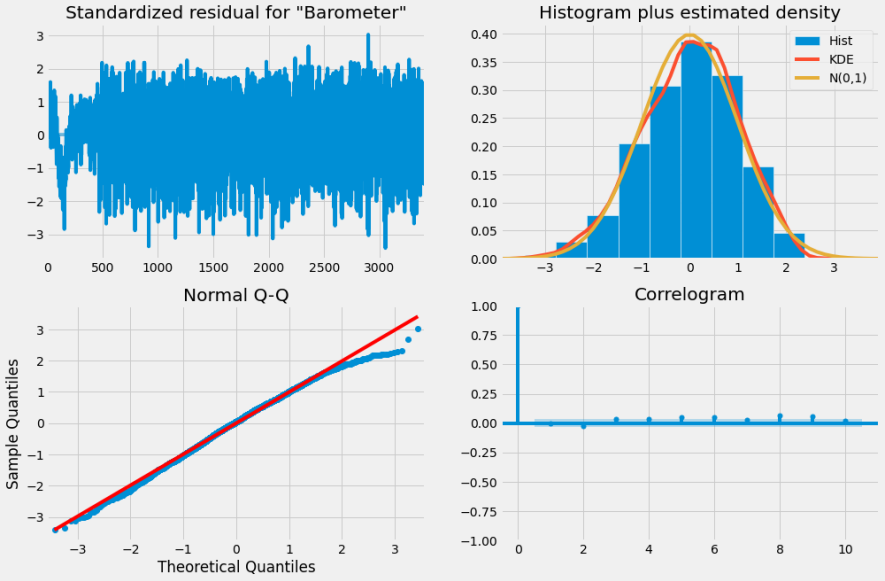
Nhận xét:

- Mô hình LSTM cho MSE và RMSE đều thấp. Sau 200, LOSS giảm dần về 0 và RMSE giảm còn 0.1.
- Kết quả dự đoán so với kết quả test không khớp nhau vẫn còn chênh lệch khá nhiều.
- Một số cột kết quả dự đoán cho giá trị là một đường thẳng.
- Mặc dù cả LOSS và RMSE đều thấp nhưng mô hình dự đoán kết quả lại không tốt.
- Mô hình không phù hợp với bộ dữ liệu.

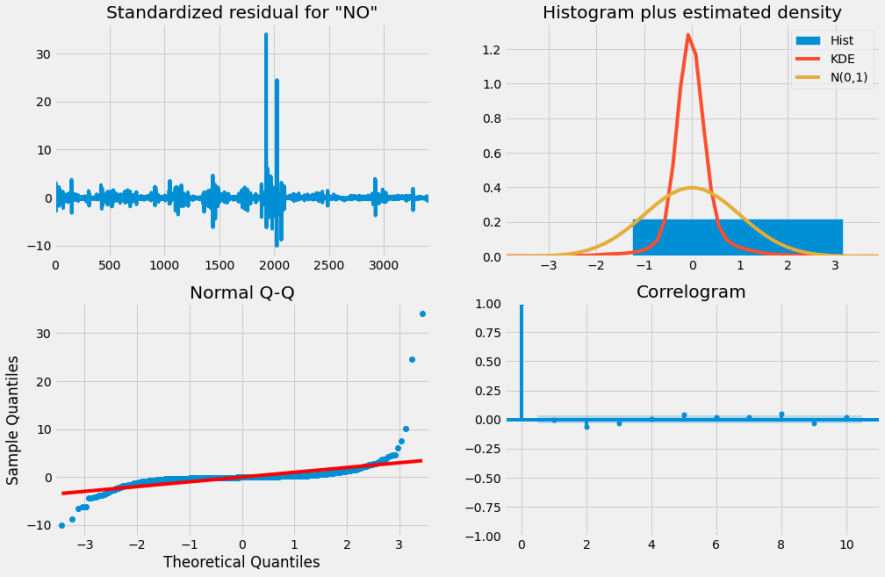
3.4.2 Mô hình VARMAX

1. Đánh giá mô hình:

   - Giá trị MSE của mô hình sau khi huấn luyện qua 1000 epoch (maxiter = 1000) là **1031.6635015800691**
   - Kết quả đánh giá theo một số cột:

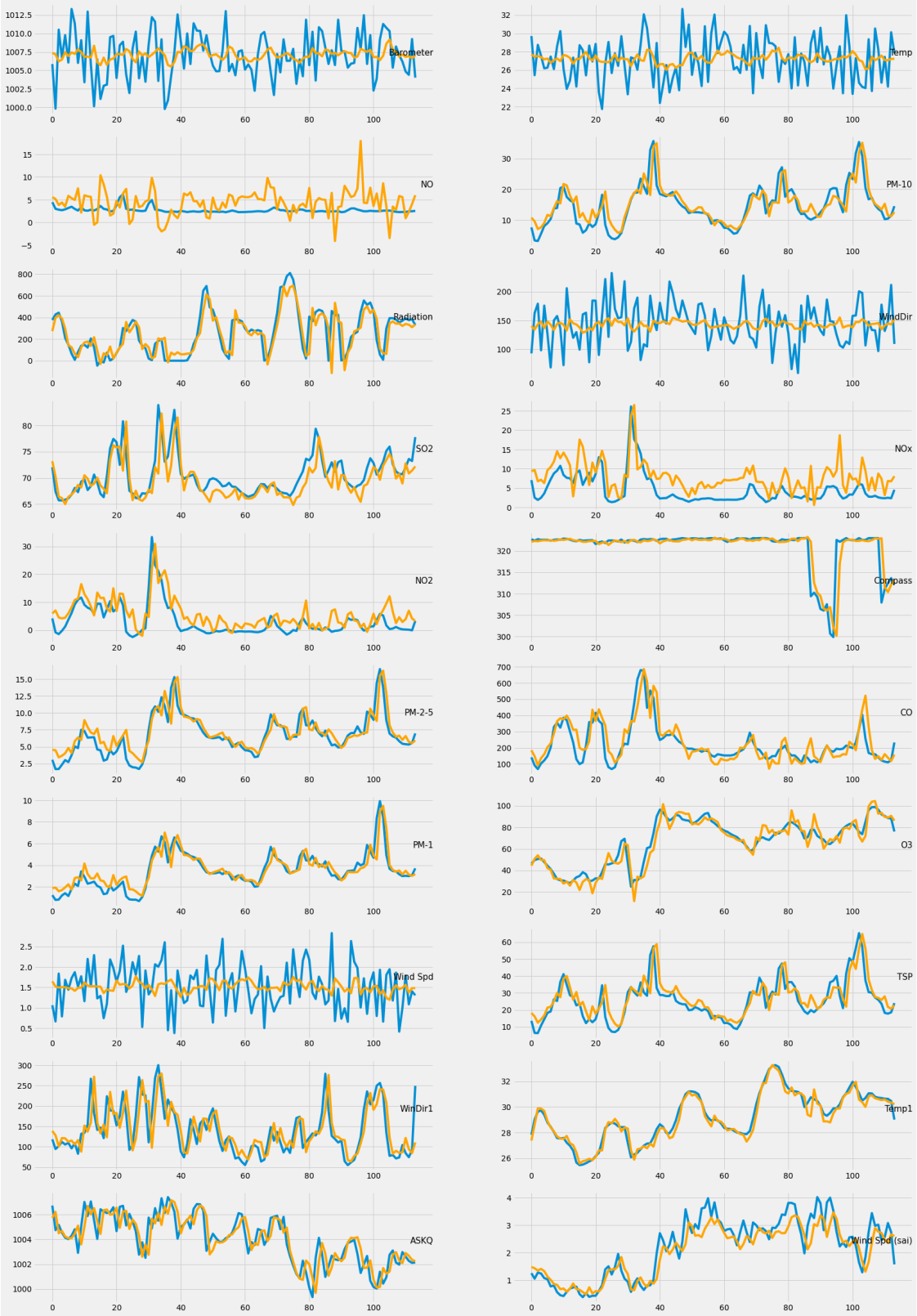


Hình 3.13 Barometer



Hình 3.14 NO

2. Kết quả dự đoán so với tập test



Hình 3.15 Kết quả dự đoán so với tập test

Nhận xét:

- Sau khi huấn luyện qua nhiều lần, thì giá trị MSE của mô hình VARMAX này vẫn khá cao.
- Tuy nhiên kết quả dự đoán so với kết quả test khá khớp với nhau trừ một số cột dữ liệu biến động mạnh.
- Mô hình vẫn cần cải thiện để phù hợp với bộ dữ liệu hơn nữa.

Nhận xét chung: Với các kết quả thu được từ hai mô hình đã thử nghiệm LSTM và VARMAX trên, nhận thấy mô hình VARMAX là phù hợp với bộ dữ liệu hơn.

CHƯƠNG 4. KẾT LUẬN

Trong phạm vi nội dung của bài tập lớn, một số nội dung mà nhóm chúng em đã đạt được:

- Thành công trong việc giới thiệu mô hình dự đoán chuỗi thời gian LSTM và mô hình VARMAX.
- Ứng dụng được vào bài toán dự đoán chất lượng không khí.

Với những kết quả đạt được, bài tập lớn có nhiều tiềm năng ứng dụng trong nhiều bài toán khác nhau về chuỗi thời gian. Một số hướng phát triển tiếp theo của đồ án:

- Cải thiện độ chính xác của mô hình và ứng dụng vào nhiều lĩnh vực khác nhau, ví dụ: dự báo giá chứng khoán, dự đoán doanh số bán hàng, v.v...
- Phát triển thêm các biến thể của LSTM như Gated Recurrent Unit (GRU), Depth Gated RNNs hay Clockwork RNNs

Tài liệu tham khảo

1. Joos Korstanje, *Advanced Forecasting with Python*.

2. B. Brockwell and R. Davis, "Time series: Theory and methods", Springer-Verlag, 1987.

3. S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural computation*, vol. 9, pp. 1735-80, 12 1997.

4. https://viblo.asia/p/machine-learning-thu-lam-nha-thien-van-du-bao-thoi-t

5. https://www.statsmodels.org/dev/examples/notebooks/generated/statespace_varmax

6. https://vi.wikipedia.org/wiki/Sai_s%E1%BB%91_to%C3%A0n_ph%C6%B0%C6%A1ng_trung_b%C3%ACnh