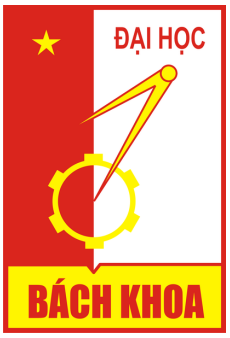
ĐẠI HỌC BÁCH KHOA HÀ NỘI
KHOA TOÁN - TIN



ĐỒ ÁN II

DỰ BÁO NHU CẦU TRONG LOGISTICS SỬ DỤNG ARIMA và LSTM

Chuyên ngành: Toán ứng dụng

Giảng viên hướng dẫn: TS. Nguyễn Thị Ngọc Anh
Sinh viên thực hiện: Trương Việt Dũng
MSSV: 20206278
Lớp: HTTT 01 - K65

Hà Nội, tháng 1 năm 2024

NHẬN XÉT CỦA GIẢNG VIÊN

1. Mục tiêu

(a)

(b)

(c)

2. Nội dung

(a)

(b)

(c)

3. Đánh giá kết quả đạt được

(a)

(b)

(c)

Hà Nội, ngày ... tháng ... năm 2024

Giảng viên hướng dẫn

TS. NGUYỄN THỊ NGỌC ANH

Lời Cảm Ơn

Báo cáo này được thực hiện và hoàn thành tại Đại học Bách Khoa Hà Nội, nằm trong nội dung học phần Đồ án II của kì học 2023-1.
Em xin được dành lời cảm ơn chân thành tới TS. Nguyễn Thị Ngọc Anh, là giảng viên đã trực tiếp hướng dẫn và gợi ý cho em đề tài rất thú vị này, đồng thời cô cũng đã giúp đỡ tận tình và có những góp ý, định hướng bổ ích để em hiểu hơn về đề tài, từ đó có thể hoàn thành báo cáo này một cách tốt nhất.

Hà Nội, tháng 01 năm 2024

Tác giả đồ án

Trương Việt Dũng

Tóm tắt nội dung đồ án

Trong đồ án này, em đã tiến hành nghiên cứu và thực nghiệm nhằm đưa ra một phương pháp để dự báo nhu cầu trong lĩnh vực Logistics trên bộ dữ liệu chuỗi thời gian sẵn có. Sau khi tìm hiểu và EDA dữ liệu, em đã lựa chọn 2 mô hình là ARIMA và LSTM để tiến hành nghiên cứu và thực nghiệm. Sau quá trình nghiên cứu em đã thực nghiệm thành công trên 2 mô hình và thu được kết quả tương đối khả quan. Trong suốt quá trình đó, em cũng đã làm thêm các bước bổ sung như chuẩn hóa dữ liệu bằng thuật toán Welford hay lọc những giá trị ngoại lai bằng phương pháp khoảng tứ phân vị để kết quả dự đoán được chính xác hơn và đưa ra những nhận xét và so sánh về 2 mô hình. Kết quả cuối cùng cho thấy mỗi mô hình đều có những điểm mạnh khi áp dụng vào bài toán thực tế, tuy nhiên vẫn sẽ cần cải thiện và nâng cấp thêm để có thể đạt được mục tiêu ứng dụng trong tương lai.

Mục lục

Chương 1

Giới thiệu

1.1 Động cơ phát triển đồ án

Logistics là một phần quan trọng trong Quản lý chuỗi cung ứng. Logistics quản lý luồng hàng hóa bắt đầu từ cung cấp nguyên vật liệu từ nhà cung cấp đến sản xuất và kết thúc bằng việc giao sản phẩm đến người dùng. Ngày nay, với sự phát triển kinh tế và sự gia tăng của thương mại điện tử đã dẫn đến một sự bùng nổ về khối lượng và lưu lượng trong ngành Logistics. Cùng với đó, mô hình kinh doanh đa dạng hóa, sự tăng cường về tốc độ và yêu cầu về sự linh hoạt trong chuỗi cung ứng đã đặt ra nhiều cơ hội và thách thức mới trong lĩnh vực này. Và chính những vấn đề đó đã đòi hỏi ngành logistics phải phát triển để bắt kịp tính thời đại và đáp ứng đủ yêu cầu trong sự vận động nhanh chóng này. Logistics thông minh đã chứng minh vai trò ngày càng quan trọng của mình so với logistics truyền thống thông qua việc xử lý được các bài toán mà logistic truyền thống gặp phải cũng như gia tăng hiệu quả làm việc cho doanh nghiệp. Chính vì vậy mà việc ứng dụng công nghệ mới cũng như các thuật toán học máy vào lĩnh vực này nhằm bắt kịp yêu cầu và sự phát triển của thời đại là điều vô cùng thiết thực, cấp bách và không cần bàn cãi.

Dự báo nhu cầu trong logistics đề cập đến việc dự đoán nhu cầu trong tương lai đối với hàng hóa, lưu trữ, vận chuyển và dịch vụ trong chuỗi cung ứng. Dự báo như cầu liên quan đến việc phân tích dữ liệu lịch sử về đơn đặt hàng, giao hàng, tỷ lệ và khách hàng trong các giai đoạn khác nhau, gắn xu hướng thị trường và

các yếu tố định kỳ khác để đưa ra dự báo sáng suốt về tăng trưởng hoặc giảm nhu cầu trong tương lai. Dự báo nhu cầu là một phần quan trọng trong quản lý chuỗi cung ứng và quản lý nguồn cung cấp của một doanh nghiệp. Dự báo này giúp các doanh nghiệp lập kế hoạch sản xuất, tồn kho, và tài chính hiệu quả.

Đồ án này sẽ tập trung dự báo nhu cầu dựa trên hai phương pháp là ARIMA và LSTM. Đầu vào của mô hình là dữ liệu chuỗi thời gian nhu cầu đặt hàng được ghi nhận theo ngày. Mục tiêu mong muốn đạt được sẽ là có góc nhìn chính xác hơn về dự báo nhu cầu, hiểu được những khó khăn, vướng mắc khi thực hiện bài toán dự báo, nắm rõ hai thuật toán mạnh mẽ được sử dụng trong các bài toán dự báo là ARIMA và LSTM và cuối cùng là thực nghiệm thành công trên hai mô hình sau đó so sánh và đưa mô hình có kết quả tốt vào ứng dụng trong thực tế.

1.2 Các nghiên cứu liên quan

Với ý nghĩa quan trọng của mình mà vấn đề dự báo nhu cầu trong logistics đã nhận được nhiều sự quan tâm nghiên cứu nhằm phát triển các phương pháp và công cụ giúp các công ty giảm chi phí, cải thiện dịch vụ khách hàng và tối ưu hóa mức tồn kho. Tiêu biểu phải kể đến "A new key performance indicator model for demand forecasting in inventory management considering supply chain reliability and seasonality" của Yasin Tadayonrad, Alassane Balle Ndiaye[2023]. Mục tiêu chính của bài báo bày là giới thiệu Chỉ báo hiệu suất chính được sử dụng trong quá trình dự báo nhu cầu nhằm mang lại kết quả hiệu quả hơn về mặt chi phí tồn kho. Các phương pháp dự báo khác nhau được sử dụng sao cho phù hợp trên dữ liệu nhu cầu trước đây, mục tiêu, hạn chế, v.v. Các tác giả đưa ra các góc nhìn của mình về các cách tiếp cận nhu cầu dụ báo bao gồm phân tích chuỗi thời gian, phân tích nguyên nhân và phân tích định tính cũng như các thuật toán học máy. Phân tích chuỗi thời gian là một trong những phương pháp phổ biến nhất được sử dụng trong dự báo nhu cầu. Nó liên quan đến việc phân tích dữ liệu lịch sử để xác định các mô hình và xu hướng có thể được sử dụng để dự đoán nhu cầu trong tương lai. Các phương pháp chuỗi thời gian bao gồm các mô hình Trung bình trượt (MA), Làm mịn theo cấp số nhân (ES) và các mô hình Trung bình trượt tích hợp tự hồi

quy (ARIMA). Đây cũng là cơ sở quan trọng để em xây dựng nên Đồ án này. Ngoài ra, các tác giả cũng đã chứng minh và rất mong muốn trong bất kỳ mô hình dự báo nào, Chỉ số hiệu suất chính (KPI) nên được dùng để đo lường hiệu suất của dự báo. Họ cũng đề xuất một cách tiếp cận mới để xác định mức tồn kho an toàn tốt nhất. Cách tiếp cận này xem xét các chỉ số độ tin cậy cung cấp mạng lưới hậu cần và tính thời vụ được xác định trong các mô hình nhu cầu lịch sử. Nhận thấy việc dự báo nhu cầu trong tương lai và xác định mức tồn kho an toàn là đặc biệt quan trọng trong việc lập kế hoạch chuỗi cung ứng. Bài báo đã cố gắng kết hợp hai quá trình thiết yếu và liên tiếp này thành một mô hình duy nhất. Việc tích hợp dự báo nhu cầu và xác định mức tồn kho an toàn là rất quan trọng để đạt được quản lý hàng tồn kho tối ưu và đáp ứng nhu cầu của khách hàng một cách hiệu quả. Bằng cách dự báo chính xác nhu cầu trong tương lai, các công ty có thể đưa ra quyết định sáng suốt về số lượng và thời gian bổ sung hàng tồn kho. Đồng thời, việc xác định mức tồn kho an toàn phù hợp giúp chống lại những bất ổn về nhu cầu và sự gián đoạn của chuỗi cung ứng. Từ những lập luận và chứng mình của tác giả, ta có thể thấy được dự báo nhu cầu luôn là một trong những thách thức cơ bản trong chuỗi cung ứng. Dự báo nhu cầu là một công cụ thiết yếu để tung ra các sản phẩm sắp ra mắt, lập kế hoạch sản xuất, xác định mức tồn kho cần thiết và tạo ra các phương thức phân phối tối ưu. Những sai lầm giữa ước tính thấp và cao có thể rất tốn kém. Dự báo nhu cầu nhiều hơn thực tế có thể dẫn đến sự gia tăng quá mức trong đầu tư vào sản xuất và tồn kho. Điều này có thể dẫn đến lãng phí tài chính và giảm lợi nhuận. Mặt khác của vấn đề là việc đánh giá thấp nhu cầu có thể gây ra hậu quả tiêu cực. Khi công ty dự báo thấp hơn mức thực tế, công ty không thể tăng tính di động, đầu tư vào công nghệ mới và lập kế hoạch cho tương lai. Như vậy, dự báo nhu cầu trong chuỗi cung ứng là một quá trình quan trọng giúp các công ty dự đoán nhu cầu trong tương lai đối với sản phẩm hoặc dịch vụ của họ. Điều này cho phép họ đưa ra quyết định sáng suốt về sản xuất, quản lý hàng tồn kho và hậu cần. [5]

Ngoài ra, bài báo "Commodity demand forecasting using modulated rank reduction for humanitarian logistics planning" của các tác giả Donovan Fuqua, Steven Hespeler[2022] cũng đưa đến một góc nhìn khác của dự báo nguyên vật liệu đầu vào khi xem xét nhu cầu nhiên liệu trong hai sự kiện khủng hoảng nhân đạo trong

khu vực và chuỗi cung ứng do Chính phủ Hoa Kỳ vận hành như một phần của Chiến dịch Ứng phó Thống nhất. Bài báo cho thấy hiệu quả của việc học sâu để dự đoán nhu cầu trực tuyến theo chuỗi ngắn, chuỗi có phương sai cao và dữ liệu chuỗi thời gian cũng như trình diễn cách tiếp cận thuật toán để dự đoán trực tuyến chuỗi ngắn cho các bộ dữ liệu có tính linh hoạt cao và đa biến. Điểm nổi bật của bài báo này đó là dự báo nhu cầu trong khủng hoảng nhân đạo có rất ít dữ liệu và có tính biến động cao. Tuy nhiên các tác giả vẫn xây dựng được một mô hình có hiệu suất tốt, trong đó thuật toán Long Short-Term Memory (LSTM) có vai trò quan trọng trong việc tạo nên mô hình. Mục tiêu của LSTM RNN là học hỏi từ cả ngắn hạn và dài hạn, mối tương quan thời gian trong dữ liệu đơn biến và đa biến. LSTM thường sử dụng chuỗi thời gian hoặc dữ liệu tuần tự để tạo ra các dự đoán trong tương lai. Dẫn chứng từ bài báo cho thấy LSTM có thể giải quyết những trở ngại phức tạp và có độ trễ thời gian dài trước đây là trở ngại cho việc đào tạo RNN trong thời gian thực thành công. Tương tự như tất cả các mạng nơ-ron, LSTM truyền dữ liệu qua nhiều bước bao gồm việc bắt đầu trọng số chuyển tiếp. Các dự đoán mô hình từ các trọng số đưa ra tính toán tổn thất soa sánh đến các giá trị thực tế. Cuối cùng, tìm kiếm gradient cập nhật các trọng số bị ảnh hưởng bởi sụ mất mát.[1] Đó là cơ sở để em quyết định lưa chọn thuật toán LSTM dể xây dựng và tối ưu nhằm tạo nên một mô hình dự đoán tốt.

Một nghiên cứu nữa có thể kể đến được nêu trong bài báo "Forecasting tourism demand with a novel robust decomposition and ensemble framework". Nghiên cứu này giới thiệu một khung dự báo nhu cầu du lịch mới, được củng cố bằng thuật toán phân rã phức tạp. Cách tiếp cận của tác giả ban đầu là tách dữ liệu gốc thành nhiều chuỗi phụ và sau đó chọn các mô hình dự báo dựa trên các thuộc tính dữ liệu tương ứng của chúng. Họ đã đánh giá khuôn khổ đề xuất bằng cách dự báo lượng khách du lịch hàng tháng đến Hồng Kông từ sáu quốc gia. Phương pháp này ban đầu áp dụng thuật toán phân rã mạnh mẽ để phân chia dữ liệu gốc thành các thành phần có thể hiểu được, chẳng hạn như xu hướng và tính thời vụ. Theo đó, các mô hình tuyến tính hoặc phi tuyến tính được chọn để lập mô hình, tùy thuộc vào thuộc tính của các thành phần được phân tách. Ví dụ: các thành phần tuyến tính được mô hình hóa bằng mô hình trung bình di chuyển tích hợp tự hồi quy (ARIMA) cấp độ, trong khi các thành phần phi tuyến tính yêu cầu ứng dụng mô

hình mạng bộ nhớ ngắn hạn (LSTM). [3] Như vậy, bài báo "Forecasting tourism demand with a novel robust decomposition and ensemble framework" cùng với các nghiên cứu liên quan về LSTM trong dự báo nhu cầu đóng vai trò quan trọng trong việc xây dựng cơ sở lý thuyết cho Đồ án này. Đồ án này sẽ áp dụng và mở rộng các phương pháp và kết quả từ những nghiên cứu này để cải thiện khả năng dự báo nhu cầu trong lĩnh vực Logistics.

1.3 Đối tượng và phạm vi nghiên cứu

Dữ liệu được dùng để nghiên cứu là dữ liệu về nhu cầu hàng hóa của 1 công ty Loggistics có các thuộc tính như loại sản phẩm, kho chứa sản phẩm, số lượng đặt hàng và thời gian đặt hàng được thu tập trong khoảng thời gian từ năm 2012 đến năm 2016 theo ngày. Tuy nhiên, phạm vi nghiên cứu sẽ chỉ dừng lại ở bài toán dự báo đơn biến vì vậy nên các thuộc tính như loại sản phẩm hay kho chứa sẽ được dùng để phân loại sản phẩm.

1.4 Mục tiêu của Đồ án

Đồ án này gồm có 4 mục tiêu quan trọng:

1. Xây dựng mô hình dự báo nhu cầu trong Logistics với dữ liệu chuỗi thời gian áp dụng thuật toán ARIMA.

2. Xây dựng mô hình dự báo nhu cầu trong Logistics với dữ liệu chuỗi thời gian áp dụng thuật toán LSTM.

3. Tối ưu hiệu suất của hai mô hình nghiên cứu.

4. So sánh hai mô hình và đưa ra kết luận.

1.5 Kết cấu Đồ án

Báo cáo đồ án II này sẽ được trình bày gồm 4 chương như sau:

- Chương 1: Giới thiệu về vấn đề nghiên cứu, đối tượng và phạm vi nghiên cứu,phương pháp nghiên cứu.
- Chương 2: Kiến thức cơ sở về quá trình khai phá, làm sạch dữ liệu, mô hình ARIMA và mô hình LSTM.
- Chương 3: Kết quả thực nghiệm trên mô hình ARIMA và LSTM.
- Chương 4: Kết luận và hướng phát triển đề tài.

Chương 2

Cơ sở lý thuyết

Hệ thống chuỗi cung ứng luôn gặp khó khăn trong việc dự báo nhu cầu trong tương lai của khách hàng. Việc ước tính chính xác nhu cầu trong tương lai cho phép các chuyên gia chuỗi cung ứng đưa ra quyết định tốt hơn trong việc lập kế hoạch và thực hiện các hành động hiệu quả trong quy trình vận hành. Dự báo nhu cầu hỗ trợ các nhà hoạch định chuỗi cung ứng trong việc trả lời "số lượng sản phẩm nên được gửi đến mỗi cửa hàng bán lẻ", "mức tồn kho tối ưu cho mỗi sản phẩm là bao nhiêu?", "số lượng giữ ở mỗi trung tâm phân phối?", "từ mỗi nhà máy sản xuất, nên sản xuất và vận chuyển bao nhiêu sản phẩm?", "ước tính lượng nguyên liệu thô cần mua từ nhà cung cấp trong những tháng tới là bao nhiêu?", cũng như nhiều vấn đề khác. Để trả lời những câu hỏi này, các học viên và nhà nghiên cứu đã thực hiện nghiên cứu sâu rộng trong những thập kỷ qua. Một trong những thông số quan trọng nhất là nhu cầu trong tương lai của khách hàng. Kết quả là, hệ thống chuỗi cung ứng sẽ được chuẩn bị phù hợp cho tương lai bằng cách sử dụng các mô hình dự báo hiệu quả. Những mô hình này cung cấp thông tin đáng tin cậy hơn cho quá trình lập kế hoạch chuỗi cung ứng. Trong phạm vi nghiên cứu của Đồ án này, thuật toán LSTM sẽ được tìm hiểu và ứng dụng để xây dựng mô hình dự báo nhu cầu trong tương lai. Ngoài ra, mô hình cũng sẽ được cải thiện và tối ưu nhằm cho ra một kết quả chính xác hơn và có thể ứng dụng được.

2.1 Phân tích và khám phá dữ liệu - EDA

2.1.1 Mục đích và vai trò của EDA

EDA là một bước quan trọng của kỹ thuật xử lý dữ liệu. Quá trình EDA giúp chúng ta có cái nhìn đầu tiên về dữ liệu, từ đó có cảm giác nhất định về những gì mình có trong tay trước khi có những chiến lược xây dựng mô hình.

Hiểu về dữ liệu cần bao gồm nhiều khía cạnh như nắm rõ mô tả dữ liệu (kích thước, dạng dữ liệu), thống kê mô tả các biến, xác định mối quan hệ giữa các biến, phát hiện các khuôn mẫu và xu hướng của dữ liệu, tìm ra các giá trị bất thường, ngoại lai của dữ liệu, đồng thời kiểm tra các giả thuyết ban đầu...

Như vậy, để có thể hiểu và có cái nhìn chính xác về dữ liệu, chúng ta cần đặt ra các câu hỏi nhằm tìm ra nhiều hướng khai thác và nhiều góc nhìn khác nhau về dữ liệu. Từ đó mà biết được mình cần tập trung vào khía cạnh nào, nên sử dụng kỹ thuật biến đổi, phân tích nào và xây dựng mô hình phù hợp. Tóm lại, vai trò của EDA sẽ là hỗ trợ làm sạch dữ liệu với các kỹ thuật xác định các giá trị thiếu, sai sót và các đặc điểm dữ liệu bất thường, nắm rõ đặc điểm cấu trúc và mô hình của tập dữ liệu, phát triển và kiểm chứng các giả thuyết và giả định, xác định các biến quan trọng nhất và mối tương quan giữa các biến, hiểu rõ cách các biến tương tác với nhau và ảnh hưởng của mỗi biến đối với kết quả phân tích.

2.1.2 Quá trình thực hiện EDA

Quá trình thực hiện EDA là một quá trình lặp đi lặp lại nhiều lần nhằm giúp chúng ta có cái nhìn chính xác và hiểu hơn vè dữ liệu. Quá trình này sẽ gồm các bước quan trọng sau:

- **Xác định kích thước dữ liêu:** Bước này giúp hình dung được dữ liệu có khoảng bao nhiêu mẫu và có bao nhiêu trường dữ liệu. Việc này rất quan trọng vì nó góp phần vào việc đưa ra chiến lược xây dựng mô hình. Ví dụ nếu dữ liệu quá ít, khả năng cao là không thể dùng Deep Learning để giải quyết mà cần dùng các phương pháp khác. Biết về kích thước dữ liệu cũng giúp xác

định kích thước tập huấn luyện (training data) và tập kiểm định (validation data) cũng như chuẩn bị bộ nhớ phù hợp.

- **Xác định ý nghĩa của từng trường dữ liệu:** Xác định được ý nghĩa của từng trường dữ liệu giúp ta có những cách xử lý và tạo đặc trưng phù hợp. Hiểu rõ ý nghĩa còn giúp loại bỏ đi những trường không cần thiết cũng như chuyển đổi dữ liệu qua dạng số để thuận lợi hơn cho quá trình xây dựng và huấn luyện mô hình.
- **Kiểm tra phân phối xác suất:** Bước này cung cấp các thông tin tổng quan về dữ liệu, chẳng hạn như số lượng mẫu, trung bình, độ lệch chuẩn,...
- **Trực quan hóa dữ liệu:** Bước này giúp hiểu rõ hơn về dữ liệu bằng cách hiển thị dữ liệu một cách trực quan.
- **Khám phá các mẫu:** Bước này tìm kiếm các mẫu trong dữ liệu. Các mẫu có thể bao gồm xu hướng, mùa vụ, và các biến động bất thường.

2.1.3 Sử dụng khoảng tứ phân vị để xác định giới hạn cảu các giá trị ngoại lai

Trước tiên, ta cùng nói qua một chút về các điểm dữ liệu ngoại lai, điểm ngoại lai là một điểm dữ liệu sai khác đáng kể so với các quan sát khác. Những dữ liệu như vậy có thể gây ra các vấn đề nghiêm trọng trong phân tích, dự báo.

Các ước lượng thống kê có khả năng ứng phó với các giá trị ngoại lai được gọi là ước lượng bền: số trung vị là một thống kê bền về xu hướng tập trung, trong khi trung bình thì không. Tuy nhiên số trung bình là ước lượng chính xác hơn trên tổng quát. Khoảng tứ phân vị (IQR) là một tiêu chuẩn thường được dùng để phát hiện các giá trị ngoại lai của dữ liệu, với Q1, Q3 lần lượt là khoảng tứ phân vị thứ nhất và thứ ba. Và IQR được xác định bởi công thức:

$$IQR = Q3 - Q1 \tag{2.1}$$

Và các điểm ngoại lai x được xác định như sau:

$$\text{x} > Q3 + 1.5 \times \text{IQR}, \tag{2.2}$$

$$\text{x} < Q1 - 1.5 \times \text{IQR}. \tag{2.3}$$

Đối với những điểm này, ta xử lý bằng cách thay thế chúng bằng giá trị gần nhất trong đoạn $[Q1 - 1.5 \times \text{IQR}, Q3 + 1.5 \times \text{IQR}]$.

2.1.4 Chuẩn hoá dữ liệu bằng thuật toán Welford

Trong thống kê, điểm chuẩn là số độ lệch chuẩn, mà theo đó giá trị của điểm thô (Giá trị được quan sát hoặc được ghi nhận trong dữ liệu) cao hơn hoặc thấp hơn giá trị trung bình của những gì đang được quan sát hoặc đo lường. Điểm thô trên trung bình có điểm tiêu chuẩn dương, trong khi những điểm dưới trung bình có điểm tiêu chuẩn âm.

Nó được tính bằng cách lấy điểm thô của từng cá nhân trừ đi giá trị trung bình của mẫu và sau đó chia hiệu đó cho độ lệch chuẩn của mẫu. Quá trình chuyển đổi điểm thô thành điểm tiêu chuẩn này được gọi là chuẩn hóa.

Điểm tiêu chuẩn thường được gọi là điểm z (Z-score); hai thuật ngữ có thể được sử dụng thay thế cho nhau. Các thuật ngữ tương đương khác được sử dụng bao gồm giá trị z (z-values), điểm số bình thường (normal scores), biến tiêu chuẩn hóa (standardized variables).

Việc tính toán điểm số z yêu cầu kiến thức về giá trị trung bình và độ lệch chuẩn của tổng thể hoàn chỉnh mà điểm dữ liệu thuộc về; nếu một người chỉ có một mẫu quan sát từ dân số, thì phép tính tương tự sử dụng giá trị trung bình mẫu và độ lệch chuẩn mẫu sẽ sinh ra thống kê t.

Công thức:

Nếu biết giá trị trung bình của tổng thể và độ lệch chuẩn của tổng thể, điểm thô x được chuyển đổi thành điểm chuẩn bởi công thức sau:

$$z = \frac{x - \mu}{\sigma} \tag{2.4}$$

Trong đó: μ là giá trị trung bình của mẫu, σ là độ lệch chuẩn của tổng thể.

Giá trị tuyệt đối của z biểu thị khoảng cách giữa thô x và trung bình của mẫu theo đơn vị của độ lệch chuẩn. z âm khi điểm thô thấp hơn giá trị trung bình, dương khi cao hơn.

Việc tính z bằng công thức này yêu cầu sử dụng giá trị trung bình tổng thể và độ lệch chuẩn tổng thể, chứ không phải giá trị trung bình của mẫu nhỏ, cũng như độ lệch của mẫu nhỏ. Tuy nhiên, việc biết giá trị trung bình thực và độ lệch chuẩn của tổng thể thường là một kỳ vọng không thực tế, ngoại trừ các trường hợp như thử nghiệm tiêu chuẩn hóa, trong đó toàn bộ tổng thể được đo lường.

z-score có nhiều vài trò quan trọng trong phân tích và xử lý dữ liệu như tính khoảng dự đoán, kiểm soát quá trình, phân tích cụm và nhân rộng đa chiều,... và một vai trò không thể không kể đến của z-score đó là chuẩn hóa trong thống kê toán học. Trong thống kê toán học, một biến ngẫu nhiên X được chuẩn hóa bằng cách trừ đi giá trị kỳ vọng của nó và chia hiệu số cho độ lệch chuẩn của nó:

$$Z = \frac{X - \mathrm{E}[X]}{\sigma(X)} \tag{2.5}$$

z-score là thước đo độ biến động của một công cụ và là một phương pháp tốt để chuẩn hóa dữ liệu cho các bài toán dự báo nhu cầu trong tương lai. Tuy nhiên các số liệu trong tương lai có thể sai lệch nhiều so với thực tế dẫn đến việc độ lệch chuẩn và giá trị trung bình của dữ liệu cũng sẽ có nhiều sai khác. Chúng ta có thể giảm đi sự sai khác này bằng cách cập nhật thêm giá trị dự đoán mới liên tục trong quá trình dự báo để tính toán lại độ lệch chuẩn và giá trị trung bình. Đó chính là ý tưởng và cơ sở của thuật toán Welford.

Nhìn chung, thuật toán Welford tính toán giá trị chuẩn hóa tương tự như thuật toán Z-score, điểm khác biệt ở đây là Welford liên tục cập nhật thêm các điểm dự báo mới để tính toán lại giá trị cho độ lệch chuẩn và giá trị trung bình. Điều này giúp giảm sai lệch hơn so với việc sử dụng Z-score vì trong quá trình dự báo, các giá trị trong tương lai có thể biến động rất nhiều so với hiện tại.

Thuật toán Welford sẽ được trình bày lại như sau:

Algorithm 1 Welford

0: **function** VARIANCE(samples)
1: $M \leftarrow 0$
2: $S \leftarrow 0$
3: $N \leftarrow \text{length}(samples)$
3: **for** k **from** 1 **to** N **do**
4: $x \leftarrow \text{samples}[k]$
5: oldM $\leftarrow M$
6: $M \leftarrow M + \frac{x-M}{k}$
7: $S \leftarrow S + (x - M) \cdot (x - \text{oldM})$
7: **end for**
8: **return** $\frac{S}{N-1}$

2.1.5 Kiểm định tính dừng của dữ liệu chuỗi thời gian

Trong phân tích chuỗi thời gian, tính dừng của dữ liệu là một giả định quan trọng. Một chuỗi thời gian được coi là dừng nếu giá trị hiện tại của chuỗi không phụ thuộc vào giá trị trong quá khứ. Nói cách khác, một chuỗi dừng có xu hướng trung bình về giá trị trung bình trong dài hạn.

Tính dừng của dữ liệu chuỗi thời gian có ý nghĩa quan trọng trong phân tích hồi quy. Nếu dữ liệu chuỗi thời gian không dừng, thì các ước lượng hồi quy sẽ không hiệu quả và các kết quả hồi quy sẽ không đáng tin cậy.

Phương pháp kiểm định tính dừng được đề xuất sử dụng trong nghiên cứu này là kiểm định Dickey-Fuller mở rộng (ADF). Phương pháp này là phương pháp được sử dụng phổ biến nhất trong phân tích chuỗi thời gian.

Kiểm định ADF sử dụng mô hình hồi quy sau:

$$y_t = a + b_1 y_{t-1} + b_2 y_{t-2} + \cdots + b_m y_{t-m} + e_t. \tag{2.6}$$

Trong đó: y_t là biến cần kiểm định ở thời điểm t, a là hằng số, b_i là hệ số trễ của biến cần kiểm định, e_t là phần sai số ngẫu nhiên.

Để thực hiện kiểm định ADF, chúng ta cần tính giá trị của thống kê ADF. Thống

kê ADF có phân phối Student-t.

Nếu giá trị tuyệt đối của thống kê ADF lớn hơn giá trị tới hạn của phân phối Student-t ở mức ý nghĩa 5%, thì có thể kết luận rằng chuỗi thời gian có tính dừng.

Kiểm định tính dừng của dữ liệu chuỗi thời gian là một bước quan trọng trong phân tích chuỗi thời gian. Việc kiểm định tính dừng giúp đảm bảo rằng các ước lượng hồi quy là hiệu quả và các kết quả hồi quy là đáng tin cậy.

2.2 ARIMA

2.2.1 Giới thiệu mô hình ARIMA

Chúng ta biết rằng hầu hết các chuỗi thời gian đều có sự tương quan giữa giá trị trong quá khứ đến giá trị hiện tại. Mức độ tương quan càng lớn khi chuỗi càng gần thời điểm hiện tại. Chính vì thể mô hình ARIMA sẽ tìm cách đưa vào các biến trễ nhằm tạo ra một mô hình dự báo fitting tốt hơn giá trị của chuỗi.

Mô hình tư hôi quy tich hơp trung binh trươt (Autoregressive integrated moving average – ARIMA), hay còn được gọi là mô hình Box-Jenkins, là một phương pháp nghiên cứu độc lập thông qua việc dự đoán theo các chuỗi thời gian và thường được áp dụng cho dữ liệu chuỗi thời gian tự tương quan (autocorre-lated). ARIMA là sự kết hợp giữa các phương pháp tiếp cận Tự hồi quy (AR) và Trung bình trượt (MA) trong việc xây dựng mô hình tổng hợp của chuỗi thời gian. Mô hình này khá đơn giản, nhưng có thể cho kết quả tốt. Nó bao gồm các tham số để tính đến tính thời vụ, xu hướng dài hạn, tự hồi quy và trung bình trượt, từ đó xử lý tự tương quan được nhúng trong dữ liệu.

Mô hình ARIMA bao gồm bốn bước:

- Nhận dạng mô hình thử nghiệm: ta sẽ quan sát xem chuỗi thời gian có phải chuỗi dừng hay không, nếu chuỗi thời gian có tính mùa hay tính xu hướng thì ta sẽ thực hiện các bước để đưa chuỗi về dạng dừng.
- Ước lượng tham số: d là bậc tích hợp và p,q sẽ được xác định bằng biểu đồ tự tương quan, thông qua nghiên cứu chiều hướng biến đổi của hàm tương

quan toàn phần hay một phần. Cụ thể, p sẽ là bậc của đồ thị AR. Xét từ độ trễ đầu tiên, thanh nào nằm ngoài đường giới hạn và sau độ giảm một cách đáng kể sau một độ trễ thì hệ số tự tương quan riêng phần đó là p. Tương tự, q sẽ là bậc của MA.

- Kiểm định bằng chẩn đoán: Sau khi các tham số của mô hình tổng quát đã được xây dựng, ta sẽ kiểm tra mức độ chính xác và sự phù hợp của mô hình với dữ liệu đã lập, xem xét phần sai số có phải ngẫu nhiên thuần túy hay không. Nếu có thì mô hình đó thỏa mãn, nếu không thì ta sẽ phải thực hiện lại các bước trên.

- Dự báo: Ở bước cuối cùng này, khi mô hình phù hợp với dữ liệu đã tìm được, ta sẽ thực hiện dự báo tại thời điểm tiếp theo.

Mô hình ARIMA giúp dự báo với độ tin cậy cao hơn từ các phương pháp lập mô hình kinh tế lượng truyền thống, đặc biệt đối với dự báo ngắn hạn.

2.2.2 Mô hình ARIMA(p,d,q)

Mô hình ARIMA sẽ biểu diễn phương trình hồi quy tuyến tính đa biến của các biến đầu vào là 3 thành phần chính:

- 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:

$$y_t = c + \phi_1 y_{t-1} + \epsilon_t \tag{2.7}$$

  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:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} ... + \phi_p y_{t-p} + \epsilon_t \tag{2.8}$$

hay

$$y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} \tag{2.9}$$

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:

$$E(\epsilon_t) = 0 \tag{2.10}$$

$$\sigma(\epsilon_t) = \alpha \tag{2.11}$$

$$\rho(\epsilon_t, \epsilon_{t-s)} = 0, \forall t \geq s \tag{2.12}$$

Vế (2.10) 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 (2.11) là phương sai của chuỗi không đổi. Do kì vọng và phương sai không đỏi nên chúng ta gọi 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:

$$MA(q) = \mu + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} \tag{2.13}$$

- Intergrated: Là quá trình đồng tích hợp hoặc lấy sai phân. Yêu cầu chung của các thuật toán trong time series là chuỗi phải đảm bảo tính dừng. Hầu hết chuỗi đều tăng hoặc giảm theo thời gian. Do đó yếu tố tương quan giữa chúng chưa chắc là thực sự mà là do chúng cùng tương quan theo thời gian. Khi biến đổi sang chuỗi dừng, các nhân tố ảnh hưởng thời gian được loại bỏ

và chuỗi sẽ dễ dự báo hơn. Để tạo thành chuỗi dừng, một phương pháp đơn giản nhất là chúng ta sẽ lấy sai phân. Một số chuỗi tài chính còn qui đổi sang logarit hoặc lợi nhuận. Bậc của sai phân để tạo thành chuỗi dừng còn gọi là bậc của quá trình đồng tích hợp (order of intergrated). Quá trình sai phân bậc d của chuỗi được thực hiện như sau:

$$\text{Sai phân bậc } 1 : I(1) = \Delta(x_t) = x_t - x_{t-1}$$
$$\text{Sai phân bậc } 2 : I(d) = \Delta^d(x_t) = \underbrace{\Delta(\Delta(...\Delta(x_t)))}_{d\,times}$$

- Thông thường chuỗi sẽ dừng sau quá trình đồng tích hợp I(0) hoặc I(1). Rất ít chuỗi chúng ta phải lấy tới sai phân bậc 2. Một số trường hợp chúng ta sẽ cần biến đổi logarit hoặc căn bậc 2 để tạo thành chuỗi dừng. Phương trình ARIMA(p,d,q) có thể được biểu diễn dưới dạng:

$$\Delta x_t = \phi_1 \Delta x_{t-1} + \phi_2 \Delta x_{t-2} + ...\phi_p \Delta x_{t-p} + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + ... + \theta_q \epsilon_{t-q} \quad (2.14)$$

Trong đó Δ_{x_t} là giá trị sai phân bậc d và ϵ_t là các chuỗi nhiễu trắng.

2.2.3 Cách xác định hệ số p, d, q

Cách xác định hệ số sai phân d trong mô hình ARIMA

Mục đích của sai phân trong mô hình ARIMA là làm cho chuỗi thời gian dừng.

Thứ tự sai phân phù hợp là độ lệch tối thiểu cần thiết để cho một chuỗi thời gian dừng biến đổi xung quanh một giá trị kỳ vọng xác định và đồ thị của ACF đạt đến giá trị 0 khá nhanh.

Nếu tự tương quan dương với nhiều độ trễ (10 hoặc nhiều hơn), thì chuỗi đó cần phải sai phân thêm. Mặt khác, nếu chính tự tương quan với độ trễ 1 quá âm, thì chuỗi có thể bị sai phân quá mức.

Trong trường hợp, chúng ta thực sự không thể quyết định giữa hai thứ tự sai lệch, thì hãy chọn thứ tự cho độ lệch chuẩn nhỏ nhất trong chuỗi sai lệch. Đầu

tiên, để xác định hệ số d cho mô hình ta cần xác định chuỗi đã cho dừng hay chưa, vì nếu chuỗi đã dừng thì ta không cần đến giá trị sai phân (d=0) còn chưa dừng thì ta cần phải đi xác định hệ số d.

Ở đây, Augmented Dickey Fuller test (adfuller()), trong gói statsmodels của python được sử dụng để kiểm định xem chuỗi đã dừng hay chưa.

Giả thiết không của kiểm định ADF là chuỗi thời gian không dừng. Vì vậy nếu giá trị p-value nhỏ hơn 0.05 thì bác bỏ giả thiết không và kết luận rằng chuỗi thời gian dừng.

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.

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.

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.

Algorithm 2 Xác định p và q

Require: Chuỗi dữ liệu thời gian *data*

1: Tính toán hàm tự tương quan (ACF): acf = autocorrelation_function($data$)
2: Tính toán hàm tự tương quan riêng biệt (PACF): pacf = partial_autocorrelation_function($data$)
3: Xác định ngưỡng cắt ACF: acf_cutoff = $\frac{1.96}{\sqrt{\text{len}(data)}}$
4: Xác định ngưỡng cắt PACF: pacf_cutoff = $\frac{1.96}{\sqrt{\text{len}(data)}}$
5: Tìm chỉ số đầu tiên mà giá trị ACF vượt quá ngưỡng cắt: $p = \arg\min\{i : \text{acf}[i] < \text{acf\_cutoff}\}$
6: Tìm chỉ số đầu tiên mà giá trị PACF vượt quá ngưỡng cắt: $q = \arg\min\{i : \text{pacf}[i] < \text{pacf\_cutoff}\}$
7: **return** p, q =0

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:

$$\phi(B)X_t = \theta(B)Z_t \tag{2.15}$$

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

$$\phi(B) = 1 - \phi_1 B - ... - \phi_p B^p \tag{2.16}$$

và

$$\theta(B) = 1 + \theta_1 B + ...\theta_q B^q \tag{2.17}$$

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.

$$X_t - \phi_1 X_{t-1} - ... - \phi_p X_{t-p} = Z_t + \theta_1 Z_{t-1} + ...\theta_q Z_{t-q} \tag{2.18}$$

Khi p=0 thì biểu thức (2.18) là một chuỗi MA(q) còn nếu q=0 thì biểu thức (2.18) là một chuỗi AR(p). Brockwell và Davis [5] 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:

$$V_0 = \gamma(0) \tag{2.19}$$

$$\theta_{m,m-k} = V_k^{-1}[\gamma(m-k) - \sum_{j=0}^{k-1} \theta_{m,m-j}] \tag{2.20}$$

$$V_m = \gamma(0) - \sum_{j=0}^{m-1} \theta_{m,m-j} V_j \tag{2.21}$$

Nếu quá trình được cho bởi phương trình (2.17) là khả nghịch, thì nó có thể được biểu diễn dưới dạng:

$$X_t = \sum_{j=0}^{\infty} \psi_{jt-j} \tag{2.22}$$

Biểu thức (2.17) và (2.22) có thể được biểu diễn:

$$\psi_0 = 1 \tag{2.23}$$

$$\psi_j = \theta_j + \sum_{i=1}^{min(j,p)} \theta_i \psi_{j-1}, j = 1, 2, ... \tag{2.24}$$

Theo quy ước ta có $\theta_j = 0$ với j > q và $\phi_i = 0$ với i > p của chuỗi ARMA(p,q). Ta thấy rằng $\psi_j \longrightarrow \theta_{m,j}$. Do đó biểu thức(2.24) được viết dưới dạng:

$$\theta_{m,j} = \theta_j + \sum_{i=1}^{min(j,p)} \phi_i \theta_{m,j-1}, j = 1, 2, ...p + q \tag{2.25}$$

Với j=q+1, q+2,...q+p trong biểu thức (2.25) một hệ p phương trình có thể được tạo ra và chúng có dạng:

$$\begin{bmatrix} \theta_{m,q+1} \\ \theta_{m,q+2} \\ \vdots \\ \theta_{m,q+p} \end{bmatrix} = \begin{bmatrix} \theta_{m,q} & \theta_{m,q-1} & \cdots & \theta_{m,q+1-p} \\ \theta_{m,q+1} & \theta_{m,q} & \cdots & \theta_{m,q+2-p} \\ \vdots & \vdots & \cdots & \vdots \\ \theta_{m,q+p-1} & \theta_{m,q+p-2} & \cdots & \theta_{m,q} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_p \end{bmatrix} \tag{2.26}$$

Các giá trị của $(\phi_1, \phi_2, ...\phi_p)$ có thể được xác định bởi biểu thức(2.26). Từ biểu thức (2.25), θ_j có thể được xác định bởi phương trình:

$$\theta_j = \theta_{m,j} - \sum_{i=1}^{min(j,p)} \phi_i \theta_{m,j-i}, j = 1, 2, ...q \tag{2.27}$$

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.

2.2.5 Xây dựng mô hình ARIMA

Khi đã xác định được các tham số p, d, q, ta tiến hành xây dựng mô hình theo lần lượt các bước:

Algorithm 3 prepare_features()

Require: * x: Dữ liệu đầu vào * d: Số bậc của differencing * p: Số bậc của AR * q: Số bậc của MA

Ensure: * $features$: Các features

0: **if** $d > 0$ **then**

1: $x \leftarrow difference(x, d)$

1: **end if**

2: $ar\_features \leftarrow None$

3: $ma\_features \leftarrow None$

3: **if** $q > 0$ **then**

4: $ar \leftarrow ARIMA(0, 0, p)$

5: $ar.fit\_predict(x)$

6: $eps \leftarrow ar.resid$

7: $eps[0] \leftarrow 0$

8: $ma\_features \leftarrow lag\_view(np.r_[np.zeros(q), eps], q)$

8: **end if**

8: **if** $p > 0$ **then**

9: $ar\_features \leftarrow lag\_view(np.r_[np.zeros(p), x], p)[0]$

9: **end if**

9: **if** $ar\_features \neq None$ and $ma\_features \neq None$ **then**

10: $n \leftarrow min(len(ar\_features), len(ma\_features))$

11: $ar\_features \leftarrow ar\_features[: n]$

12: $ma\_features \leftarrow ma\_features[: n]$

13: $features \leftarrow np.hstack((ar\_features, ma\_features))$

13: **else if** $ma\_features \neq None$ **then**

14: $features \leftarrow ma\_features$

14: **else**

15: $features \leftarrow ar\_features$

15: **end if**

return $features$ =0

Algorithm 4 Dựng mô hình

0: **procedure** FIT(X)

1: **Input:** X - Ma trận đặc trưng

2: **Output:** model - Mô hình ARIMA

3: // Ước lượng các tham số của mô hình ARIMA

4: $features, x \leftarrow prepare_features(x)fit$(features, x)

return $features$

Algorithm 5 Huấn luyện mô hình

0: **procedure** FIT_PREDICT(X)

1: **Input:** X - Ma trận đặc trưng

2: **Output:** forecasts - Dự đoán

3: $features =$ fit(x)

4: **return** predict(x, prepared=($features$))

4: **end procedure**=0

Algorithm 6 Dự đoán kết quả

0: **procedure** PREDICT(model, steps)

1: **Input:** model - Mô hình ARIMA, steps - Số bước dự đoán

2: **Output:** predictions - Dự đoán tương lai

3: $features \leftarrow kwargs.get('prepared', None)$

3: **if** $features$ is None **then**

4: $features, \leftarrow prepare\_features(x)$

4: **end if**

5: $y = predict(features)$

6: $resid \leftarrow x - y$

7: **return** return_output(y)

7: **end procedure**=0

Algorithm 7 Trả về kết quả

procedure RETURN_OUTPUT(predictions)
Input: predictions - Dự đoán
Output: output - Đầu ra kết quả
    if d $>$ 0 **then**
$output \leftarrow undo\_difference(output, d)$
    end if
return output
end procedure=0

Algorithm 8 Mô hình ARIMA

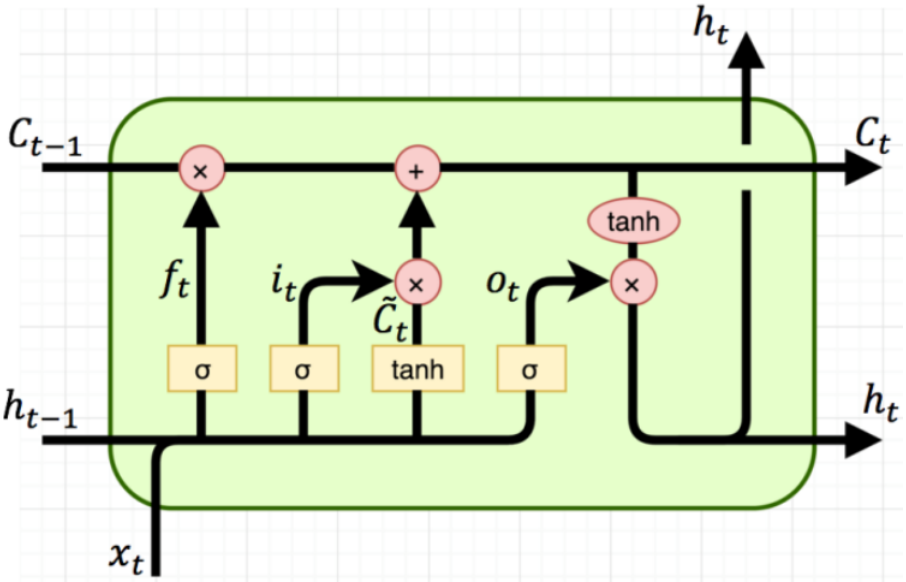
procedure ARIMA(data)
Input: data - Dữ liệu chuỗi thời gian
Output: output - Đầu ra kết quả dự đoán
$(features, x) \leftarrow prepare\_features(x)$
$y \leftarrow predict(features)$
$y \leftarrow [n]$ is zeros
$i = 0$
while $i < n$ **do**
    $feat \leftarrow [y[-(p+n)+i : -n+i], q]$
    $y[det(x)+1] \leftarrow predict(feat[None, :])$
    $i \leftarrow i+1$
end while
return $return\_output(y)$
end procedure=0

2.3 Thuật toán Mạng trí nhớ ngắn hạn định hướng dài hạn (LSTM)

Mạng trí nhớ ngắn hạn định hướng dài hạn LSTM là một kiến trúc đặc biệt của RNN có khả năng học được sự phụ thuộc trong dài hạn được giới thiệu bởi Hochreiter và Schmidhuber (1997). LSTM đã khắc phục được rất nhiều hạn chế chủa RNN về triệt tiêu đạo hàm. Tuy nhiên cấu trúc có phần phức tạp hơn mặc dù vẫn giữ được tư tưởng chính của RNN là sao chép các kiến trúc theo dạng chuỗi. Không giống như các feed-forward neural networks, LSTM có các kết nối phản hồi. Nó có thể xử lý không chỉ các điểm dữ liệu đơn lẻ (chẳng hạn như hình ảnh) mà còn toàn bộ chuỗi dữ liệu (chẳng hạn như speech hoặc video).

2.3.1 Cấu trúc mạng của LSTM

Kiến trúc module trong mạng LSTM chứa 4 tầng ẩn (3 sigmoid và 1 tanh) tương tác với nhau theo một cấu trúc đặc biệt.

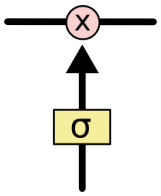


Hình 2.1: Mô hình LSTM [4]

Ý tưởng chính của LSTM là thành phần ô trạng thái (cell state), được thể hiện qua đường chạy ngang trên cùng của hình vẽ và các cổng để kiểm soát lưu thông thông tin. Cụ thể hơn, cell state là một dạng giống như băng truyền. Nó chạy

xuyên suốt tất cả các mắt xích (các nút mạng) và chỉ tương tác tuyến tính đôi chút. Vì vậy mà các thông tin có thể dễ dàng truyền đi thông suốt mà không sợ bị thay đổi. Cell state ghi nhớ các giá trị trong khoảng thời gian tùy ý và các cổng input gate, output gate và forget gate tham gia điều chỉnh các luồng thông tin input và output.

LSTM có khả năng bỏ đi hoặc thêm vào các thông tin cần thiết cho trạng thái tế báo, chúng được điều chỉnh cẩn thận bởi các nhóm được gọi là cổng (gate). Các cổng là nơi sàng lọc thông tin đi qua nó, chúng được kết hợp bởi một tầng mạng sigmoid và một phép nhân.



Hình 2.2: Tầng Sigmoid.

Tầng sigmoid sẽ cho đầu ra là một số trong khoản [0, 1], mô tả có bao nhiêu thông tin có thể được thông qua. Khi đầu ra là 0 thì có nghĩa là không cho thông tin nào qua cả, còn khi là 1 thì có nghĩa là cho tất cả các thông tin đi qua nó.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.28}$$

Algorithm 9 Hàm Sigmoid

```
function SIGMOID(x)
return 1/(1 + exp(−x))
```

Tổng quan về cách các cổng trong LSTM hoạt động:

- **Input gate:** Phát hiện ra giá trị nào từ đầu vào sẽ được sử dụng để sửa đổi bộ nhớ. Hàm Sigmoid quyết định giá trị nào sẽ cho qua 0 hoặc 1. Và hàm tanh đưa ra trọng số cho các giá trị được truyền, quyết định mức độ quan trọng của chúng trong khoảng từ -1 đến 1.

- **Forget gate:** Nó khám phá các chi tiết cần loại bỏ khỏi khối. Một hàm sigmoid quyết định nó. Nó xem xét trạng thái trước đó (ht-1) và đầu vào nội dung (Xt) và xuất ra một số giữa 0 (bỏ qua điều này) và 1 (giữ nguyên điều này) cho mỗi số trong trạng thái ô Ct-1.

- **Output gate:** Đầu vào và bộ nhớ của khối được sử dụng để quyết định đầu ra. Hàm Sigmoid quyết định giá trị nào cho qua 0 hoặc 1. Và hàm tanh quyết định giá trị nào cho qua 0, 1. Và hàm tanh đưa ra trọng số cho các giá trị được truyền, quyết định mức độ quan trọng của chúng trong khoảng từ -1 đến 1 và nhân lên với đầu ra là sigmoid.

Chúng ta giả sử rằng có h nút ẩn, mỗi minibatch có kích thước n và kích thước đầu vào là d. Như vậy, đầu vào là $\mathbf{X}_t \in \mathbb{R}^{n \times d}$ và trạng thái ẩn của bước thời gian trước đó là $\mathbf{H}_{t-1} \in \mathbb{R}^{n \times h}$. Tương tự, các cổng được định nghĩa như sau: cổng đầu vào là $\mathbf{I}_t \in \mathbb{R}^{n \times h}$, cổng quên là $\mathbf{F}_t \in \mathbb{R}^{n \times h}$, và cổng đầu ra là $\mathbf{O}_t \in \mathbb{R}^{n \times h}$. Chúng được tính như sau:

$$\mathbf{I}_t = \sigma\left(\mathbf{X}_t \mathbf{W}_{xi} + \mathbf{H}_{t-1} \mathbf{W}_{hi} + \mathbf{b}_i\right), \tag{2.29}$$

$$\mathbf{F}_t = \sigma\left(\mathbf{X}_t \mathbf{W}_{xf} + \mathbf{H}_{t-1} \mathbf{W}_{hf} + \mathbf{b}_f\right), \tag{2.30}$$

$$\mathbf{O}_t = \sigma\left(\mathbf{X}_t \mathbf{W}_{xo} + \mathbf{H}_{t-1} \mathbf{W}_{ho} + \mathbf{b}_o\right), \tag{2.31}$$

trong đó $\mathbf{W}_{xi}, \mathbf{W}_{xf}, \mathbf{W}_{xo} \in \mathbb{R}^{d \times h}$ và $\mathbf{W}_{hi}, \mathbf{W}_{hf}, \mathbf{W}_{ho} \in \mathbb{R}^{h \times h}$ là các trọng số và $\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_o \in \mathbb{R}^{1 \times h}$ là các hệ số điều chỉnh.

Tiếp theo, chúng ta sẽ thiết kế một ô nhớ. Vì ta vẫn chưa chỉ định tác động của các cổng khác nhau, nên đầu tiên ta sẽ giới thiệu ô nhớ tiềm năng $\tilde{\mathbf{C}}_t \in \mathbb{R}^{n \times h}$. Các phép tính toán cũng tương tự như ba cổng mô tả ở trên, ngoài trừ việc ở đây ta sử dụng hàm kích hoạt tanh với miền giá trị nằm trong khoảng $[-1, 1]$. Điều này dẫn đến phương trình sau tại bước thời gian t.

$$\tilde{\mathbf{C}}_t = \tanh\left(\mathbf{X}_t \mathbf{W}_{xc} + \mathbf{H}_{t-1} \mathbf{W}_{hc} + \mathbf{b}_c\right). \tag{2.32}$$

ở đây $\mathbf{W}_{xc} \in \mathbb{R}^{d \times h}$ và $\mathbf{W}_{hc} \in \mathbb{R}^{h \times h}$ là các tham số trọng số và $\mathbf{b}_c \in \mathbb{R}^{1 \times h}$ là một hệ số điều chỉnh.

Trong LSTM, chúng ta có hai tham số, $\mathbf{I}_t$ điều chỉnh lượng dữ liệu mới được lấy vào thông qua $\tilde{\mathbf{C}}_t$ và tham số quên $\mathbf{F}_t$ chỉ định lượng thông tin cũ cần giữ lại trong ô nhớ $\mathbf{C}_{t-1} \in \mathbb{R}^{n \times h}$. Sử dụng cùng một phép nhân theo từng điểm (pointwise) như trước đây, chúng ta đi đến phương trình cập nhật như sau.

$$\mathbf{C}_t = \mathbf{F}_t \odot \mathbf{C}_{t-1} + \mathbf{I}_t \odot \tilde{\mathbf{C}}_t. \tag{2.33}$$

Nếu giá trị ở cổng quên luôn xấp xỉ bằng 1 và cổng đầu vào luôn xấp xỉ bằng 0 , thì giá trị ô nhớ trong quá khứ $\mathbf{C}_{t-1}$ sẽ được lưu lại qua thời gian và truyền tới bước thời gian hiện tại.

Cuối cùng, chúng ta cần phải xác định cách tính trạng thái ẩn $\mathbf{H}_t \in \mathbb{R}^{n \times h}$. Đây là nơi cổng đầu ra được sử dụng. Trong LSTM, đây chỉ đơn giản là một phiên bản có kiểm soát của hàm kích hoạt tanh trong ô nhớ. Điều này đảm bảo rằng các giá trị của $\mathbf{H}_t$ luôn nằm trong khoảng $(-1, 1)$. Bất cứ khi nào giá trị của cổng đầu ra là 1 , thực chất chúng ta đang đưa toàn bộ thông tin trong ô nhớ tới bộ dự đoán. Ngược lại, khi giá trị của cổng đầu ra là 0 , chúng ta giữ lại tất cả các thông tin trong ô nhớ và không xử lý gì thêm.

$$\mathbf{H}_t = \mathbf{O}_t \odot \tanh\left(\mathbf{C}_t\right) \tag{2.34}$$

Quá trình trên được gọi là quá trình tiến của LSTM.

Algorithm 10 lstm_forward

```
function LSTM_FORWARD(inputs, prev_h, prev_c, Wx, Wh, b)
for i = 0 to num_cols do
    a[i] ← prev_h[i] · Wh[i] + inputs[i] · Wx[i] + b[i]
end for
for i = 0 to num_cols do
    i[i] ← sigmoid(a[i])
end for
for i = 0 to num_cols do
    f[i] ← sigmoid(a[H + i])
end for
for i = 0 to num_cols do
    o[i] ← sigmoid(a[2H + i])
end for
for i = 0 to num_cols do
    g[i] ← tanh(a[3H + i])
end for
next_c ← f × prev_c + i × g
next_h ← o × tanh(next_c)
cache ← inputs, prev_h, prev_c, Wx, Wh, b, a, i, f, o, g, next_c
return next_h, next_c, cache
```

2.3.2 Lan truyền ngược trong LSTM

Ô LSTM trả về một vectơ đầu ra (h_t) và trạng thái LSTM được cập nhật (C_t). Để đạt được dự đoán, chúng tôi thêm một lớp đầu ra ánh xạ đầu ra LSTM với một dự đoán. Điều này có thể được áp dụng bất cứ khi nào chúng ta muốn đạt được một dự đoán. Trong các trường hợp chúng ta sẽ xem xét, chúng ta sẽ có một perceptron một lớp đơn giản không có chức năng kích hoạt để đầu ra là không giới hạn. Chúng tôi ánh xạ đến một đầu ra vô hướng vì đây là tất cả những gì cần thiết cho các vấn đề đơn giản của chúng tôi. Do đó, lớp cuối cùng này trở thành sản phẩm bên trong giữa vectơ trọng số và đầu ra của LSTM (với thêm 1 được

thêm vào trước để hoạt động như một sai lệch).

Các hoạt động của lan truyền ngược hoạt động để truyền tín hiệu huấn luyện (tức là gradient của hàm mất mát) đến các tham số được cập nhật. Sự lan truyền này về cơ bản là đảo ngược hướng của chuyển tiếp lấy một sản phẩm của các gradient dọc theo đường dẫn từ mất mát đến các tham số được cập nhật. Trong trường hợp RNN, điều này yêu cầu lan truyền các gradient "ngược theo thời gian".

Để thấy điều này rõ ràng nhất, chúng ta hãy viết hoạt động chuyển tiếp của LSTM dưới dạng đồ thị tính toán. Sau đó, chúng ta có thể thấy đường dẫn từ trọng lượng đến đầu ra bị đảo ngược trong quá trình huấn luyện. Việc tính toán cập nhật trọng lượng thực chất là một ứng dụng của quy tắc dây chuyền; tuy nhiên, nhìn vào nó qua lăng kính của đồ thị tính toán sẽ cho phép chúng ta thấy cách tính các cập nhật này một cách trực quan.

Algorithm 11 LSTMBackward

function LSTMBackward(dh_states, cache)

$do \leftarrow dh\_states * tanh(next\_c)$

$dtanh\_next\_c \leftarrow dh\_states * o$

$dnext\_c \leftarrow dtanh\_next_c * (1 - tanh(next\_c)^2)$

$df \leftarrow dnext\_c * prev\_c$

$dprev\_c \leftarrow \_c * f$

$di \leftarrow dnext\_c * g$

$dg \leftarrow dnext\_c * i$

$da \leftarrow np.hstack([d(i,f,o) * (i,f,o) * (1 - (i,f,o)), dg * (1 - g^2)])$

$dWx \leftarrow inputs^\intercal * da$

$dWh \leftarrow prev\_h^\intercal * da$

$db \leftarrow sum(da, axis = 0)$

$dinputs \leftarrow da * Wx^\intercal$

$dprev\_h \leftarrow da * Wh^\intercal$

return $dinputs, dprev\_h, dWx, dWh, db$

Chúng ta có thể thấy trong biểu đồ rằng trong quá trình chuyển tiếp trọng số ô, (W_c, W_f, W_i, W_o) có liên quan ở mọi bước thời gian. Do đó, có nhiều tuyến đường từ mất mát (L) đến trọng lượng. Để đạt được gradient thực sự, chúng ta cần tính toán gradient dọc theo mỗi con đường từ mất mát đến trọng lượng được đề cập và

sau đó tính tổng tất cả các gradient kết quả. Độ dốc dọc theo đường dẫn có thể được tính toán thông qua quy tắc chuỗi bằng cách lấy sản phẩm dọc theo đường dẫn.

Chúng ta hãy bắt đầu với một ma trận trọng số đơn để làm ví dụ minh họa. Xem xét ma trận trọng số W_o, chúng ta đạt được gradient sau đây.

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial W_o} = & \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial h_T} \frac{\partial h_T}{\partial W_o} \\ & + \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial h_T} \frac{\partial h_T}{\partial h_{T-1}} \frac{\partial h_{T-1}}{\partial W_o} \\ & + \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial h_T} \frac{\partial h_T}{\partial C_T} \frac{\partial C_T}{\partial h_{T-1}} \frac{\partial h_{T-1}}{\partial W_o} \\ & + \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial h_T} \frac{\partial h_T}{\partial h_{T-1}} \frac{\partial h_{T-1}}{\partial h_{T-2}} \frac{\partial h_{T-2}}{\partial W_o} \\ & + ...\end{aligned} \tag{2.35}$$

2.3.3 Xây dựng mô hình LSTM:

Xây dựng hàm xử lý xuôi (forward pass):

- forwardpass(inputs, prevh, prevc, Wx, Wh, b, Why, by):

  + Thực hiện lstmforward cho mọi đầu vào trong chuỗi.

  + Tính toán điểm số (scores) và xác suất (pro) cho các lớp đầu ra.

  + Trả về pro, các trạng thái ẩn và cache.

Algorithm 12 Forward Pass

0: **function** FORWARD_PASS(inputs, prev_h, prev_c, Wx, Wh, b, Why, by)
1: $h\_states, h\_cache \leftarrow$ lstm_forward(inputs, prev_h, prev_c, Wx, Wh, b)
2: scores $\leftarrow h\_states \cdot Why^T + by$
3: pro $\leftarrow$ softmax(scores)
4: **return** pro, h_states, h_cache

Tính toán mất mát (loss):

- calculateloss(pro, targets):

  + Tính toán mất mát dựa trên xác suất dự đoán và các nhãn mục tiêu.

Xây dựng hàm xử lý ngược (backward pass):

Algorithm 13 Tính toán Loss

function CALCULATE_LOSS(pro, targets)

seq_length $\leftarrow$ length(targets) correct_logprobs $\leftarrow$ $-\log(\text{pro}[\text{range}(\text{seq\_length}), \text{argmax}(\text{targets}, \text{axis} = 1)])$

loss $\leftarrow \frac{\sum \text{correct\_logprobs}}{\text{seq\_length}}$

return loss

- backwardpass(pro, targets, hcache, inputs, Wx, Wh, b, Why, by, learningrate):

  + Tính toán đạo hàm của các tham số và trạng thái ẩn.

  + Cập nhật các tham số mô hình theo tỉ lệ học (learningrate).

Algorithm 14 Backward Pass

function BACKWARD_PASS(pro, targets, h_cache, inputs, Wx, Wh, b, Why, by)

dscores $\leftarrow$ pro

dscores[range(length(targets)), argmax(targets, axis = 1)]$-$ = 1

$dWhy \leftarrow$ transpose(dscores) $\cdot$ h_states

$dby \leftarrow$ sum(dscores, axis = 0)

dh_states $\leftarrow$ dscores $\cdot$ Why

$dinputs$, $d$prev_h $\leftarrow$ lstm_backward(dh_states, h_cache)

for param, $d$param in zip($[Wx, Wh, Why, b, by]$, $[dWx, dWh, dWhy, db, dby]$) **do**

param$-$ = learning_rate $\cdot$ $d$param

end for

return $dinputs$, $d$prev_h

Mô hình sẽ được huấn luyện theo các bước:

+ Lặp lại quá trình huấn luyện trong nhiều vòng lặp (epochs).

+ Trong mỗi vòng lặp:

  + Thực hiện forwardpass cho mọi thời điểm trong chuỗi đầu vào.

  + Tính toán mất mát.

  + Thực hiện backwardpass để cập nhật các tham số.

Algorithm 15 Train Model

function TRAIN_MODEL(inputs, targets, prev_h, prev_c, Wx, Wh, b, Why, by)
  for epoch ← 1 to epochs **do**
prev_h ← zeros_like(prev_h)
prev_c ← zeros_like(prev_c)
    for t ← 1 to length(inputs) **do**
x ← inputs[t]
target ← targets[t]
pro, h_states, h_cache ← forward_pass(x, prev_h, prev_c, Wx, Wh, b, Why, by)
loss ← calculate_loss(pro, target)
$dinputs$, $d$prev_h ← backward_pass(pro, target, $h\_cache, x, Wx, Wh, b, Why, by$)
prev_h ← h_states[length(inputs)]
    end for
  end for

Cuối cùng thực hiện trả về kết quả dự đoán.

Algorithm 16 Dự đoán

function DU_DOAN(x, prev_h, prev_c, Wx, Wh, b, Why, by)
pro, h_states, h_cache ← forward_pass(x, prev_h, prev_c, Wx, Wh, b, Why, by)
prediction ← argmax(probabilities)
return prediction

2.4 Sai số phần trăm tuyệt đối trung bình (MAPE)

2.4.1 MAPE

Một trong những thước đo phổ biến nhất về độ chính xác của dự đoán mô hình, sai số phần trăm tuyệt đối trung bình (MAPE) là tỷ lệ phần trăm tương đương với sai số tuyệt đối trung bình (MAE). Sai số phần trăm tuyệt đối trung bình đo lường mức độ sai số trung bình do một mô hình tạo ra hoặc mức độ sai lệch trung bình của các dự đoán. Mặc dù việc hiểu số liệu này và cách tính toán số liệu này là quan trọng nhưng việc hiểu được điểm mạnh và hạn chế của số liệu này khi sử dụng số liệu này trong sản xuất cũng rất quan trọng.

Một mô hình học máy có thể chỉ tốt bằng dữ liệu được sử dụng để huấn luyện nó, nhưng việc đánh giá mô hình tổng thể dựa trên các số liệu hiệu suất được sử dụng trong sản xuất. Nếu dữ liệu không nhất quán – bị sai lệch, có nhiều ngoại lệ hoặc số 0 và số nan – chỉ số hiệu suất sẽ chịu trách nhiệm hiểu tính hiệu quả của mô hình và nắm bắt những vấn đề này. Để đo lường hiệu suất của các dự đoán của mô hình so với thực tế cơ bản của nó (còn được gọi là thực tế) và hiểu sâu hơn về cách mô hình của bạn tác động đến hành vi của người dùng, lợi nhuận và các chỉ số hiệu suất chính (KPI) khác, trước tiên phải chọn một chỉ số hiệu suất sao cho phù hợp với trường hợp sử dụng. Và MAPE được cho là phù hợp để đánh giá hiệu suất của

2.4.2 Tính toán chỉ số MAPE

MAPE phản ánh giá trị dự báo sai khác bao nhiêu phàn trăm so với giá trị trung bình và được tính theo công thức sau:

$$\text{MAPE} = \frac{1}{n}\sum_{i=0}^{n}\frac{|y_i - \hat{y_i}|}{y_i}. \tag{2.36}$$

Trong đó 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. [2]

Chương 3

Thực nghiệm và đánh giá kết quả

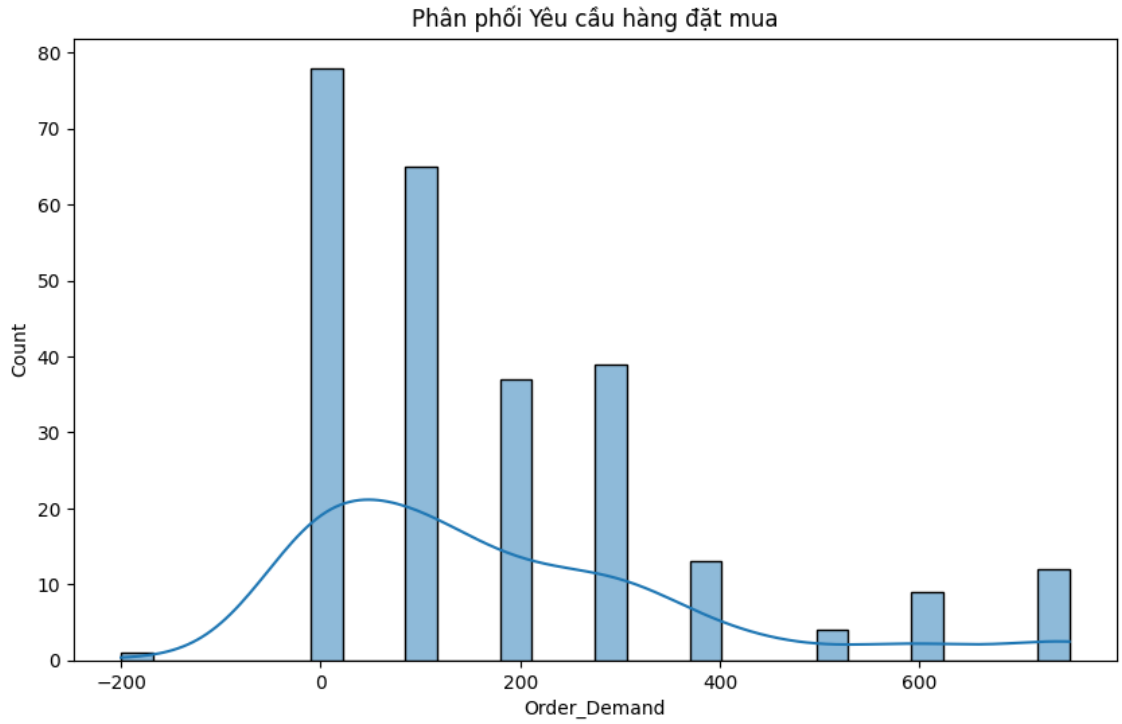
3.1 Dữ liệu và kịch bản chạy chương trình

Dữ liệu được sử dụng để nghiên cứu là dữ liệu về nhu cầu đặt hàng của một công ty Logistics trong khoảng thời gian từ năm 2012 đến năm 2016. Dữ liệu có hơn 1 triệu dòng và 5 cột là Productcode, Warehouse, ProductCategory, Date và OrderDemand. Các cột Productcode, Warehouse, ProductCategory thể hiện cho mã sản phẩm, kho lưu trữ và loại sản phẩm. Còn cột Date và cột OrderDemand lần lượt thể hiện thời gian và nhu cầu đặt hàng. Các sản phẩm khác nhau, ở các kho khác nhau và thuộc các loại khác nhau sẽ có những tính chất khác nhau. Do đó chúng ta sẽ chia dữ liệu theo từng loại dựa trên 3 cột Productcode, Warehouse, ProductCategory. Dữ liệu có Productcode, Warehouse và ProductCategory giống nhau sẽ được coi là một loại, từ đó ta có thể nghiên cứu và dự báo nhu cầu của từng loại trong tương lai. Sau đó ta sẽ lấy ra một loại để chạy mô hình.

Sau khi đã tách được dữ liệu dùng để nghiên cứu, chúng ta sẽ nhìn tổng thể bộ dữ liệu mới để có những hướng phân tích với bộ dữ liệu. Nhận thấy mỗi tháng chỉ có vài điểm dữ liệu chứ không có hết tất cả các ngày trong tháng nên các điểm dữ liệu sẽ được gộp lại theo tuần để quá trình phân tích và nghiên cứu được thuận lợi hơn. Bộ dữ liệu sau khi được gộp lại theo tuần sẽ có 258 điểm dữ liệu. Tiến hành thống kê mô tả bằng hàm describe() trong python ta có thể thấy được những thông số cơ bản của bộ dữ liệu như trung bình là 194,57 , độ lệch chuẩn là 241,52, giá trị nhỏ nhất là -200, lớn nhất là 1400, 25% dữ liệu có giá trị nhỏ hơn hoặc bằng

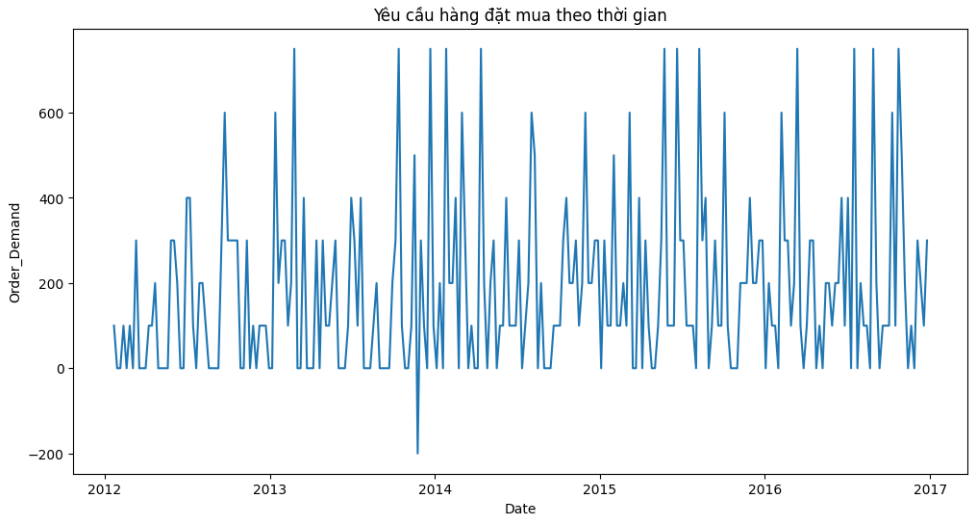
0, 50% dữ liệu có giá trị nhỏ hơn hoặc bằng 100 và 75% giá trị có giá trị nhỏ hơn hoặc bằng 300.

Xét thấy dữ liệu có nhiều điểm nhiễu, ta tiến hành xác định và thay thế những điểm ngoại lai. Dữ liệu sau khi được tiền xử lý qua các bước trên sẽ có phân phối như biểu đồ sau:



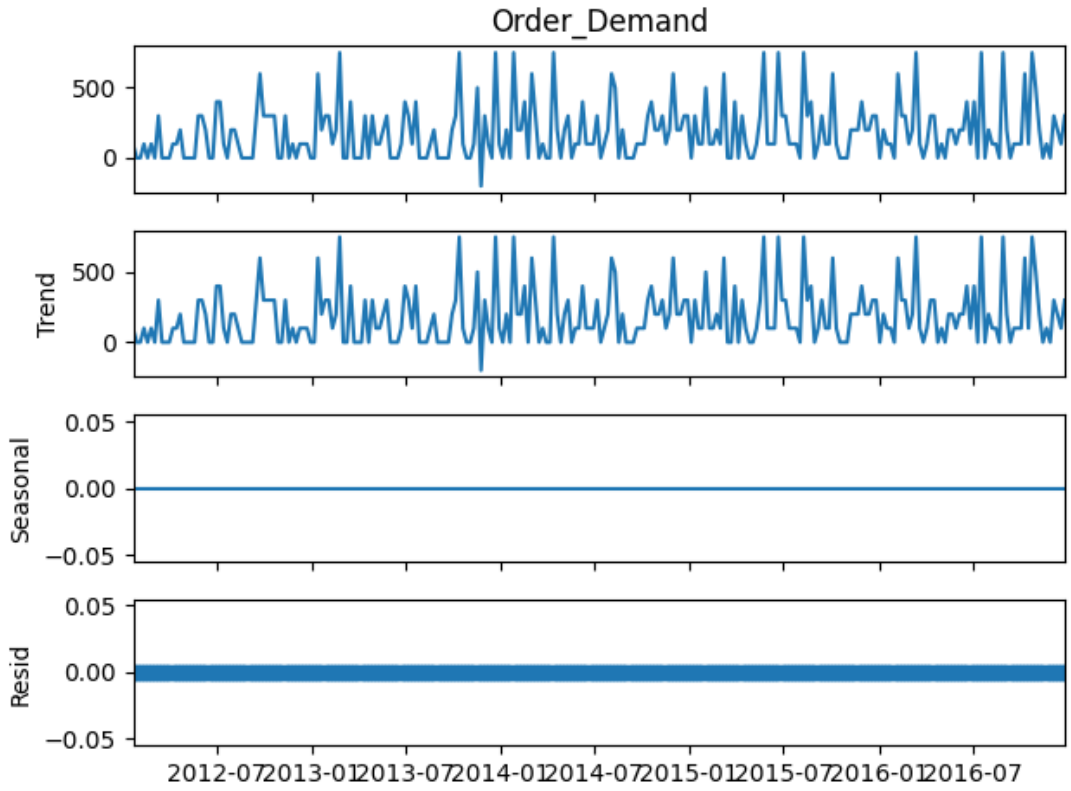
Hình 3.1: Phân phối yêu cầu hàng đặt mua

Dữ liệu được thể hiện dưới dạng biểu đồ theo thời gian:



Hình 3.2: Yêu cầu hàng đặt mua theo thời gian

Bước tiếp theo của quá trình EDA, dữ liệu sẽ được phân rã theo thời gian để kiểm tra tính xu hướng và tính mùa vụ.



Hình 3.3: Phân rã chuỗi thời gian

Dữ liệu sẽ được chạy trên hai kịch bản (kịch bản thứ nhất là chạy dữ liệu với mô hình ARIMA, kịch bản thứ hai là chạy dữ liệu với mô hình LSTM). Ở cả hai kịch bản, dữ liệu đều sẽ được chia thành tập huấn luyện và kiểm tra với tỷ lệ 80% dùng để huấn luyện và 20% dùng để kiểm tra, đánh giá. Ta tính toán các ước lượng cho tham số và dùng grid search để tìm kiếm bộ tham số phù hợp nhất cho mỗi mô hình.

3.2 Kết quả

Kết quả dự đoán được thể hiện bởi đồ thị và được đánh giá bằng sai số phần trăm tuyệt đối trung bình (MAPE). Dưới đây là kết quả của mỗi mô hình với các bộ tham số khác nhau:

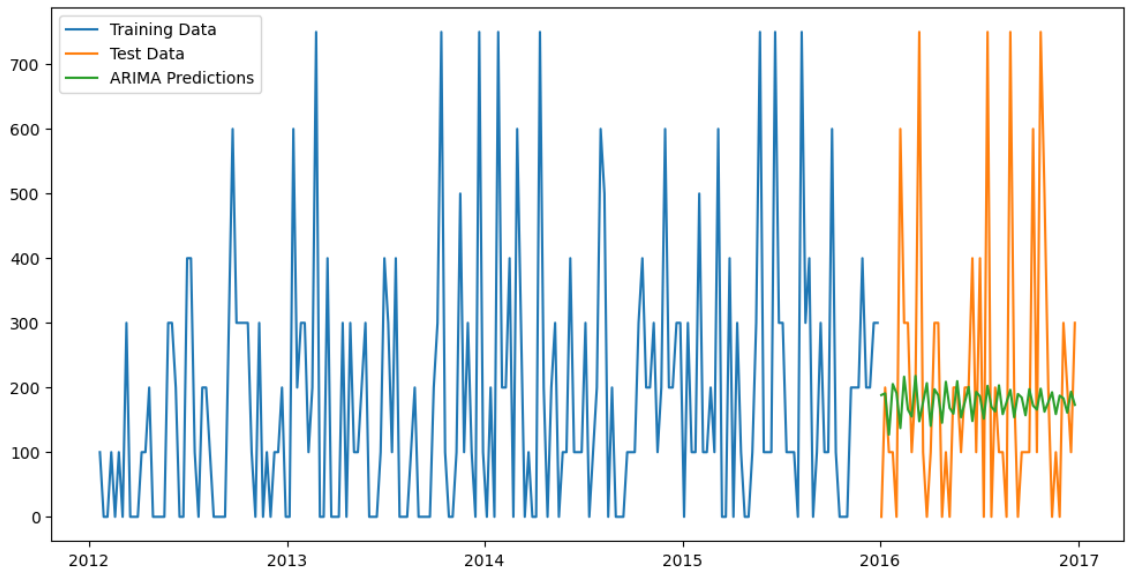
p / q	0	1	2	3	4	5
0	73,90	73,93	73,95	73,94	73,67	74,67
1	73,77	73,56	73,46	74,04	73,89	74,33
2	73,56	73,34	73,78	74,05	74,11	**73,01**
3	73,98	73,77	74,01	73,72	74,35	74,23
4	74,12	74,12	74,36	73,27	73,43	73,46
5	74,28	74,27	74,27	74,11	75,92	74,04

Hình 3.4: Kết quả dự đoán mô hình ARIMA

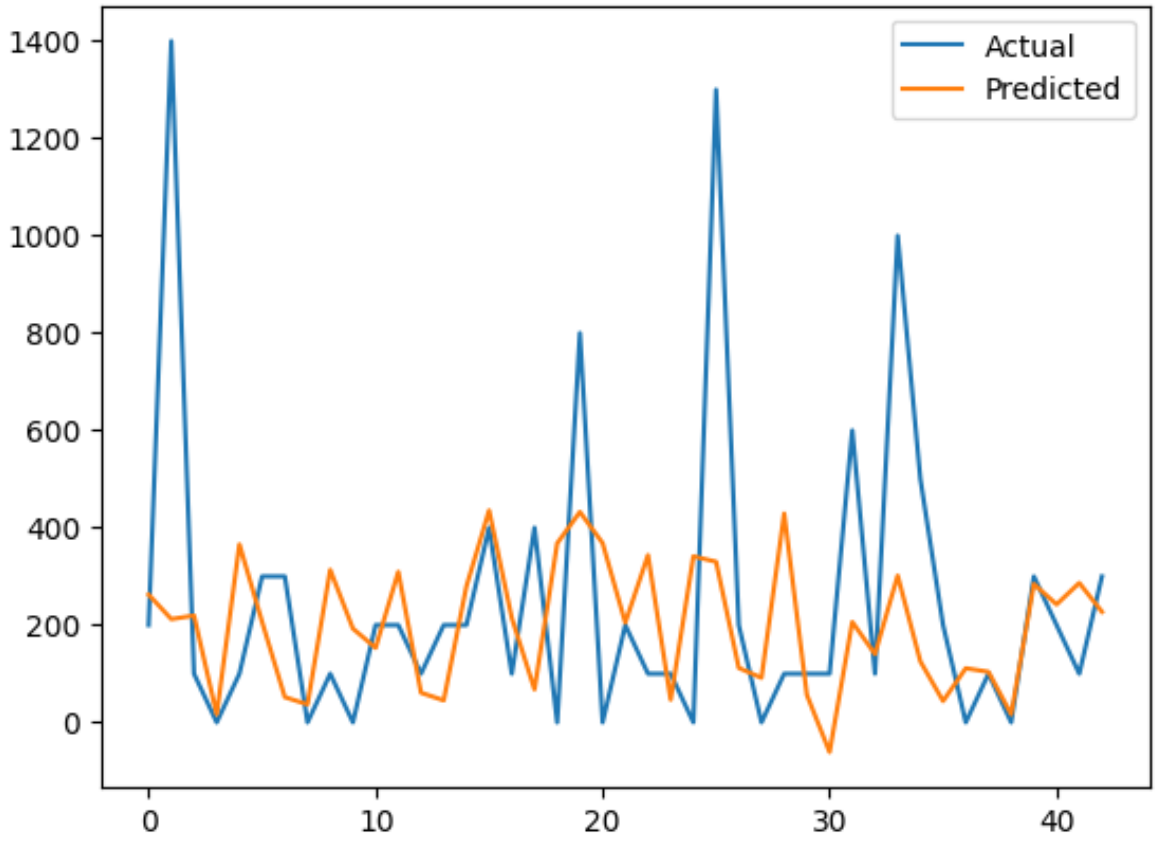
Look_back	epochs	Batch_size	Kết quả
3	100	2	79,01
3	100	5	77,17
3	200	2	77,18
3	200	5	77,94
6	100	2	94,77
6	100	5	80,74
6	200	2	120,7
6	200	5	126,33
9	100	2	90,88
9	100	5	88,12
9	200	2	**77,1**
9	200	5	105,32
12	100	2	103,74
12	100	5	134,25
12	200	2	119,12
12	200	5	138,87

Hình 3.5: Kết quả dự đoán mô hình LSTM

Kết quả chạy mô hình cho thấy MAPE khi chạy dữ liệu với ARIMA cho kết quả tốt nhất là 73% với p, q là 5 và 2(d = 0 do chuỗi thời gian đã dừng), còn LSTM là 77%. Tốc đọ chạy của mỗi mô hình trên bộ tham số này lần lượt là 27 giây và 2 phút.



Hình 3.6: Kết quả dự đoán mô hình ARIMA(5,0,2)



Hình 3.7: Kết quả dự đoán mô hình LSTM

3.3 Nhận xét

Qua biểu đồ so sánh và chỉ số sai số phần trăm tuyệt đối trung bình (MAPE) ta có thể thấy được:

- Trong kịch bản thứ nhất khi sử dụng thuật toán ARIMA mô hình cho ra kết quả dự đoán trên tập test gần với giá trị trung bình hơn, tuy nhiên chưa mô phỏng được dáng điệu của đồ thị trên tập test.
- Trong kịch bản thứ hai khi sử dụng thuật toán LSTM, mô hình cho ra kết quả dự đoán sai lệch nhiều hơn với giá trị trung bình tuy nhiên phần nào cho thấy được dáng điệu của đồ thị trên tập test.

Nhìn chung, cả hai mô hình vẫn cho kết quả sai số tương đối lớn nên chưa có tính áp dụng trong thực tiễn. Nguyên nhân này đến từ việc dữ liệu có rất nhiều điểm nhiễu và biến động rất lớn, không ổn định. Ngoài ra, do phạm vi của bài toán mới chỉ dừng lại ở bài toán dự báo đơn biến nên chắc chắn sẽ có những hạn chế so với việc nghiên cứu và áp dụng bài toán đa biến.

Qua kết quả so sánh hiệu suất của hai mô hình, ta thấy được mô hình ARIMA sẽ phù hợp hơn LSTM trong việc giải quyết bài toán này. Nguyên nhân của điều này là do phạm vi của bài toán chỉ dừng lại ở dự báo bài toán dự báo đơn biến trong khi LSTM sẽ cho thấy khả năng mạnh mẽ hơn khi áp dụng vào bài toán dự báo đa biến với nhiều chuỗi thời gian phức tạp. Tuy vậy, LSTM vẫn cho hiệu suất không thua kém nhiều so với mô hình ARIMA. Đó là cơ sở để có thế lựa chọn LSTM cho những bài toán dự báo đa biến phức tạp trong tương lai.

Chương 4

Kết luận và hướng phát triển đề tài

4.1 Kết luận

- Dự báo nhu cầu trong Logistics là một vấn đề quan trọng trong thực tế và còn rất nhiều tiềm năng để nghiên cứu trong tương lai. Việc ứng dụng các thuật toán học máy và học sâu sẽ là xu hướng và chắc chắn sẽ mạng lại những kết quả tích cực đối với các doanh nghiệm trong việc lấp kế hoạch sản xuất, tồn kho và tài chính hiệu quả.

- Trong quá trình xây dựng và đánh giá 2 thuật toán dự báo nhu cầu trong Logistics LSTM và ARIMA, chúng ta đã thấy sức mạnh và đặc tính riêng biệt của từng mô hình.

- Mô hình ARIMA với cách tiếp cận thống kê và khả năng xử lý xu hướng và biến động có cấu trúc, cũng đã đem lại kết quả khả quan. Tuy nhiên, do đặc trưng của dữ liệu được sử dụng có nhiều điểm ngoại lai và biến động lớn nên mô hình chưa thể đem lại kết quả như kì vọng. Em cũng rút ra được rằng mô hình ARIMA thích hợp cho những tình huống khi có sự biến động có cấu trúc và dễ dàng diễn giải.

- Mô hình LSTM được kì vọng sẽ đem lại hiệu suất ấn tương, tuy nhiên do việc hạn chế phạm vi bài toán ở việc chỉ giải quyết bài toán dự báo đơn biến nên đã ảnh hưởng đến kết quả dự đoán của LSTM. Có thể thấy LSTM với khả năng học được các mối quan hệ phi tuyến tính và xử lý dữ liệu chuỗi thời

gian phức tạp sẽ phù hợp hơn cho bài toán dự báo đa biến với nhiều chuỗi và dữ liệu lớn.

- Kết quả dự báo của hai mô hình cho thấy mô hình ARIMA có hiệu suất tốt hơn mô hình LSTM. Nguyên nhân của điều này là do khi xét trong phạm vi bài toán dự báo đơn biến, LSTM không thể phát huy hết được ưu điểm cảu mình. LSTM sẽ phù hợp và cho kết quả tốt hơn khi gặp phải các bài toán có mối quan hệ phi tuyến tính và xử lý dữ liệu chuỗi thời gian phức tạp.
- Các thuật toán dự báo học máy và học sâu cho ta thấy chúng có khả năng dự báo đáng kinh ngạc. Tuy vậy chúng cũng đòi hỏi người thực hiện phải tính toán và cân nhắc nhiều điều trong việc thay đổi kiến trúc mô hình và tối ưu các tham số thì mới có thể đem lại hiệu quả tốt.
- Mặc dù hiệu quả mô hình chưa cao nhưng sẽ tạo ra cơ sở tiền đề để nghiên cứu và hoàn thiện trong tương lai.

4.2 Hướng phát triển của đề tài

- Mở rộng phạm vi bài toán thành dự báo dữ liệu đa biến với thuật toán LSTM.
- Tìm hiểu, nghiên cứu nhằm đánh giá, xây dựng lại kiến trúc mô hình LSTM như thêm hoặc giảm số lớp, số đơn vị trong mỗi lớp, ..., đồng thời cũng tối ưu các tham số để mô hình cho kết quả chính xác hơn.
- Tối ưu mô hình sao cho giảm độ phức tạp và đạt được hiệu quả cao hơn.
- Ứng dụng mô hình mới cho những bài toán thực tế ở lĩnh vực khác cũng như nghiên cứu tìm hiểu thêm các mô hình mới để có cái nhìn tổng quát và chính xác hơn trong việc lựa chọn mô hình cho các bài toán dự báo thực tế.

Tài liệu tham khảo

[1] Donovan Fuqua and Steven Hespeler. Commodity demand forecasting using modulated rank reduction for humanitarian logistics planning. *Expert Systems with Applications*, 206:117753, 2022.

[2] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[3] Xin Li, Xu Zhang, Chengyuan Zhang, and Shouyang Wang. Forecasting tourism demand with a novel robust decomposition and ensemble framework. *Expert Systems with Applications*, 236:121388, 09 2023.

[4] Kartika Resiandi, Yohei Murakami, and Arbi Haza Nasution. Neural network-based bilingual lexicon induction for indonesian ethnic languages. *Applied Sciences*, 13(15), 2023.

[5] Yasin Tadayonrad and Alassane Ballé Ndiaye. A new key performance indicator model for demand forecasting in inventory management considering supply chain reliability and seasonality. *Supply Chain Analytics*, 2023.

PHỤ LỤC

CODE PYTHON MÔ PHỎNG SỐ LIỆU

```
# Import cac thu vien can thiet
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from sklearn.preprocessing import StandardScaler

# Load Data
data = pd.read_csv('/content/drive/MyDrive/Book2.csv')

df = pd.DataFrame(data)

# Chuyen cot 'Date' sang dinh dang datetime
df['Date'] = pd.to_datetime(df['Date'], format='%m/%d/%Y')

# Dat cot 'Date' lam chi so cho DataFrame
df.set_index('Date', inplace=True)

# Hien thi 1 so hang dau cua dataframe
print(df.head())
# chia du lieu theo tuan
df = df.resample('W').sum()
#  Thng   k   m   t
print(df.describe())

#  Kim  tra  gi   tr   thiu
```

```
print(df.isnull().sum())
print(df['Order_Demand'].dtype)
if df['Order_Demand'].dtype != 'O':
    df['Order_Demand'] = df['Order_Demand'].astype(str)
#  X  lý  ct  'Order_Demand'     loi   b   cc  ký  t   khng  phi   s
df['Order_Demand'] = pd.to_numeric(df['Order_Demand'].str.replace(r'[^0-9-]', '',
    regex=True), errors='coerce')

# Tinh IQR cho cot 'Order_Demand'
Q1 = df['Order_Demand'].quantile(0.25)
Q3 = df['Order_Demand'].quantile(0.75)
IQR = Q3 - Q1

# Loc va thay the outlier
df['Order_Demand'] = df['Order_Demand'].apply(lambda x: max(x, Q1 - 1.5 * IQR) if x <
    Q1 - 1.5 * IQR else min(x, Q3 + 1.5 * IQR) if x > Q3 + 1.5 * IQR else x)
```

```
# EDA - Bieu o dang chuoi  thi  gian  ca  Order_Demand
plt.figure(figsize=(12, 6))
sns.lineplot(x='Date', y='Order_Demand', data=df)
plt.title('Yeu cau hang at mua theo thoi gian')
plt.show()

# EDA - Phan phoi cua Order_Demand
plt.figure(figsize=(10, 6))
sns.histplot(df['Order_Demand'], bins=30, kde=True)
plt.title('Phan phoi Yeu cau hang dat mua')
plt.show()
```

```
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# EDA - Phan tich chuoi thoi gian
# Phan ra chuoi thoi gian de kiem tra xu huong va mua vu
result = seasonal_decompose(df['Order_Demand'], model='additive', period=1)

result.plot()
plt.show()

```

```
# Hien  th   biu        chui   thi  gian
plt.plot(df['Order_Demand'])
plt.title(' Chui   Thi  Gian (Original)')
plt.show()

from statsmodels.tsa.stattools import adfuller

result_diff = adfuller(df['Order_Demand'].dropna())

# In ket qua kiem inh sau sai phan
print('ADF Statistic (After Differencing):', result_diff[0])
print('p-value:', result_diff[1])
print('Critical Values:', result_diff[4])

# Kiem tra ket qua va ua ra ket luan ve tinh dung
if result_diff[1] <= 0.05:
    print("Ket luan: Chuoi thoi gian sau sai phan la dung")
else:
    print("Ket luan: Chuoi thoi gian sau sai phan khong phai la dung")

# EDA - Kiem tra ham tuong quan tu hoi quy (ACF) va ham tuong quan rieng (PACF)
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
plot_acf(df['Order_Demand'], lags=50, ax=plt.gca())
plt.subplot(2, 1, 2)
plot_pacf(df['Order_Demand'], lags=50, ax=plt.gca())
plt.show()

from statsmodels.tsa.arima.model import ARIMA

# Chia du lieu thanh tap huan luyen v tap kiem tra
train_size = int(len(df) * 0.8)
train, test = df['Order_Demand'][:train_size], df['Order_Demand'][train_size:]

# Xay dung mo hinh ARIMA
model = ARIMA(train, order=(5, 0, 2))  # iu  chnh  order  da  trn  ACF  v  PACF
fit_model = model.fit()

# Du bao tren tap kiem tra
predictions = fit_model.forecast(steps=len(test))
```

```

# So sanh du bao voi thuc te
plt.figure(figsize=(12, 6))
plt.plot(train, label='Training Data')
plt.plot(test, label='Test Data')
plt.plot(predictions, label='ARIMA Predictions')
plt.legend()
plt.show()
```

```
from sklearn.metrics import mean_absolute_error
import numpy as np

# Tinh MAPE
mape = mean_absolute_error(test, predictions) / np.mean(test) * 100
print('Mean Absolute Percentage Error (MAPE):', mape)
```

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

from sklearn.metrics import mean_squared_error
from keras.models import Sequential
from keras.layers import LSTM, Dense

# Load  d   liu
data = pd.read_csv('/content/drive/MyDrive/Book2.csv')

df = pd.DataFrame(data)

#  Chuyn   ct  'Date' sang  nh   dng  datetime
df['Date'] = pd.to_datetime(df['Date'], format='%m/%d/%Y')

#   t    ct  'Date' lm  ch   s  cho DataFrame
df.set_index('Date', inplace=True)
#  Kim  tra  gi  tr  thiu
print(df.isnull().sum())
print(df['Order_Demand'].dtype)
if df['Order_Demand'].dtype != 'O':
    df['Order_Demand'] = df['Order_Demand'].astype(str)
```

```
# X lý ct 'Order_Demand' loi b cc ký t khng phi s
df['Order_Demand'] = pd.to_numeric(df['Order_Demand'].str.replace(r'[^0-9-]', '',
    regex=True), errors='coerce')
df = df.resample('W').sum()

# Welford algorithm
def update_welford(mean, var, count, new_value):
    count += 1
    delta = new_value - mean
    mean += delta / count
    delta2 = new_value - mean
    var += delta * delta2
    return mean, var, count

# Tnh mean v std bng Welford algorithm
count = 0
mean = 0
var = 0
for value in df['Order_Demand']:
    mean, var, count = update_welford(mean, var, count, value)

std = np.sqrt(var / count)

# Chun ha d liu
df['Order_Demand'] = (df['Order_Demand'] - mean) / std

# univariate lstm example
# Chia d liu thnh tp hun luyn v tp kim tra
train_size = int(len(df) * 0.8)
train, test = df['Order_Demand'][:train_size], df['Order_Demand'][train_size:]

# Hm to dataset cho m hnh LSTM
def create_dataset(dataset, look_back=1):
    X, Y = [], []
    for i in range(len(dataset) - look_back):
        a = dataset[i:(i + look_back)]
        X.append(a)
        Y.append(dataset[i + look_back])
    return np.array(X), np.array(Y)

```

```
# To dataset cho m hnh vi look_back
look_back = 9
X_train, y_train = create_dataset(train.values, look_back)
X_test, y_test = create_dataset(test.values, look_back)

# Reshape input m bo ph hp vi u vo ca m hnh LSTM ( s
    mu , s bc thi gian, s c trng )
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))

# Xy dng m hnh LSTM
model = Sequential()
model.add(LSTM(50, input_shape=(X_train.shape[1], 1)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')

# Hun luyn m hnh
model.fit(X_train, y_train, epochs=200, batch_size=2, verbose=2)

# D bo trn tp kim tra
y_pred = model.predict(X_test)

# o ngc qu trnh chun ha Z-score
y_pred_inv = (y_pred * std) + mean
y_test_inv = (y_test * std) + mean

# Hien thi du bao va thuc te tren bieu o
plt.plot(y_test_inv, label='Actual')
plt.plot(y_pred_inv, label='Predicted')
plt.legend()
plt.show()
```

```
# Tinh MAPE
from sklearn.metrics import mean_absolute_error
mape = mean_absolute_error(y_test_inv, y_pred_inv[:, 0]) / np.mean(y_test_inv) * 100
print('Mean Absolute Percentage Error (MAPE):', mape)
```