LINEAR

Trong **phương trình hồi quy tuyến tính**, chúng ta có các biến sau:

- y: Đây là biến phụ thuộc (dependent variable), đại diện cho giá trị chúng ta muốn dự đoán hoặc giải thích dựa trên các biến độc lập khác.

- x: Đây là biến độc lập (independent variable), đại diện cho các yếu tố hoặc biến số khác mà chúng ta sử dụng để dự đoán hoặc giải thích giá trị của biến phụ thuộc y.

- β_o: Đây là hệ số chặn (intercept coefficient), đại diện cho giá trị y khi x = 0, tức là giá trị của y khi không có biến độc lập x. Hệ số chặn cho biết tại điểm xuất phát, giá trị của y là bao nhiêu.

- β_1: Đây là hệ số hồi quy (regression coefficient) cho biến x, đại diện cho sự thay đổi trung bình trong y khi x tăng lên một đơn vị. Hệ số hồi quy cho biết mức độ tác động của biến x lên biến y.

- ε: Đây là sai số ngẫu nhiên (random error), đại diện cho những yếu tố không xác định hoặc không đo được mà ảnh hưởng đến giá trị của biến phụ thuộc y mà không được giải thích bởi các biến độc lập x.

Phương trình hồi quy tuyến tính được sử dụng để mô hình hóa mối quan hệ tuyến tính giữa biến phụ thuộc và các biến giải thích, với mục tiêu dự đoán và giải thích giá trị của biến phụ thuộc dựa trên giá trị của các biến độc lập.

VECM

Lý thuyết

- Mô hình (VAR) và (VECM) là hai mô hình thống kê đa biến đều sử dụng chuỗi thời gian (time series) để mô hình hóa tương quan và tương tác giữa chúng, VECM có một tính chất đặc biệt và khác biệt so với VAR.
- **VECM là một mở rộng của VAR**, nhưng nó chủ yếu được sử dụng khi các biến time series có mối quan hệ cointegration. Một mối quan hệ cointegration xảy ra khi các biến không tương quan nhưng vẫn có một **mối quan hệ kết hợp dài hạn**.

Từ đó, đọc ý (2) "Nên đó là điều mà ở VECM sẽ phân tích và nghiên cứu đến giữa các biến các kết quả (Y) trong HTĐB.

- Giải thích về "hệ thống đa biến" vì điều kiện cần ít nhất 1 mối quan hệ cointegration => tối thiểu 2 biến.
- Biến kết quả (Y) vì ko có biến X trong phương trình.

VECM cho phép phân tích cả mối quan hệ ngắn hạn và dài hạn giữa các biến kết quả trong một hệ thống đa biến cointegrated:

- Mô hình VAR tập trung vào mô hình hóa tương quan ngắn hạn giữa các biến và từ đoán đưa ra dự đoán trong tương lai.
- VECM cung cấp thông tin về tác động cân bằng dài hạn và phân tích được mối quan hệ cointegration.

Do đó, VECM là một công cụ phù hợp hơn VAR khi muốn phân tích cả mối quan hệ ngắn hạn và dài hạn giữa các biến kết quả trong một hệ thống đa biến có mối quan hệ cointegration.

Phương trình VECM trong dạng chính quy **mô tả mối quan hệ tương quan dài hạn giữa các biến kết quả và sự điều chỉnh tự động của hệ thống đa biến khi có sự sai lệch ngắn hạn** như sau:

- ΔY_t: vectơ các biến kết quả tại thời điểm t, chứa thông tin về sự biến đổi của các biến kết quả trong mô hình.
- α: vectơ hệ số chặn (intercept).
- Y_{t-1}: vectơ các biến kết quả tại thời điểm (t-1). trong đó ρ là **độ trễ (lag)** của mô hình (số lượng thời điểm trễ), thể hiện mức độ ảnh hưởng của biến trễ lên biến kết quả trong mô hình VECM.
    - Bằng cách sử dụng giá trị lag, chúng ta có thể khám phá và mô hình hóa mối quan hệ giữa các biến kết quả trong quá khứ và tác động của chúng lên giá trị của các biến tại thời điểm hiện tại.

- Π: ma trận hệ số cointegration, đại diện cho tương quan dài hạn giữa các biến kết quả trong hệ thống đa biến. Ma trận này thể hiện sự điều chỉnh tự động của các biến kết quả về một trạng thái cân bằng dài hạn sau khi có sự sai lệch ngắn hạn.
- $\theta_1, \theta_2, \ldots \theta_p$ là ma trận hệ số của các biến sai phân (differences) lagged, đại diện cho mối quan hệ ngắn hạn giữa các biến kết quả và các biến sai phân lagged của chúng
- ε_t: vectơ nhiễu (error term) tại thời điểm t, đại diện cho các yếu tố ngẫu nhiên không được mô hình hóa bởi các biến đầu vào.

Để ước lượng và phân tích VECM:

- Đảm bảo stationary (tính ổn định) của chuỗi dữ liệu: **trong hệ thống đa biến. Một biến non-stationary không thể tham gia vào một mô hình cointegration trực tiếp, vì nó có thể dẫn đến kết quả không đáng tin cậy.**
    - Sử dụng kiểm định Augmented Dickey-Fuller.
    - Nếu non-stationary, thực hiện kỹ thuật first difference (sai phân bậc 1): **để biến đổi nó thành một biến stationary.**
    - **Sau khi áp dụng kỹ thuật sai phân bậc 1, ta kiểm tra lại tính trừu tượng của biến đã được biến đổi. Nếu biến sau sai phân bậc 1 trở thành stationary, ta có thể sử dụng nó trong mô hình VECM.**
- Xác định số lượng lag order (số lượng bậc tự do):
    - Lag order của VECM = số lag tối ưu từ Underlying VAR model trừ 1:

- Lý do là trong mô hình VAR cơ bản, các biến được xác định dựa trên biến gốc (original variables), trong khi trong mô hình VECM, các biến được xác định dưới dạng sai phân cấp 1 (first difference form). Sai phân cấp 1 được thực hiện để đạt được tính dừng của chuỗi dữ liệu và giải quyết vấn đề tính dừng khi áp dụng mô hình VECM.
- Khi áp dụng sai phân cấp 1 cho các biến, chúng ta loại bỏ xu hướng tăng/giảm tuyến tính trong chuỗi dữ liệu. Điều này dẫn đến việc mất đi một bậc tự do trong mô hình VECM, nên số lượng lag order trong VECM sẽ ít hơn một so với số lượng lag order trong VAR cơ bản.
- Vì vậy, nếu mô hình VAR cơ bản có p lags, thì mô hình VECM tương ứng sẽ có (p-1) lags.
    - Lag order của VECM >0: **số lượng lag order bằng 0, tức là không xem xét bất kỳ giá trị trước đó nào để dự đoán giá trị hiện tại, thì VECM không sẽ không có ý nghĩa**
- Xác định số lượng cointegrating relationships (mối quan hệ cointegration):
    - Sử dụng phương pháp kiểm định Johansen: **dựa trên ý tưởng rằng nếu có mối quan hệ cointegration trong hệ thống đa biến, các biến sẽ di chuyển cùng nhau dài hạn mặc dù có thể có sự dao động ngắn hạn.**
    - Phương pháp này cho phép chúng ta xác định số lượng mối quan hệ cointegration bằng cách kiểm tra giả thuyết số lượng mối quan hệ cointegration có thể có từ 0 đến một giới hạn tối đa, với mức độ tin cậy trong phương pháp kiểm định Johansen được xác định dựa trên mức độ p-value mà chúng ta chọn, thường là 5% hoặc 1%.

Code

Đoạn code trên thực hiện việc khôi phục giá trị dự đoán từ dạng sai phân cấp 1 về dạng ban đầu trên tập test. Quá trình khôi phục giá trị được thực hiện như sau:

- **test_forecast_restored = test_diff.copy():** Tạo một bản sao của tập dữ liệu sai phân cấp 1 (test_diff) để lưu trữ giá trị dự đoán đã khôi phục.
- **test_forecast_restored.loc[-1] = train_data.iloc[-2]:** Gán giá trị cuối cùng của tập train_data (trước khi tạo dự đoán) vào hàng đầu tiên của tập dữ liệu đã khôi phục (test_forecast_restored). Điều này đảm bảo rằng giá trị dự đoán được bắt đầu từ giá trị cuối cùng của tập train_data.
- **test_forecast_restored['Open'] = test_diff['Open'].cumsum() + train_data['Open'].iloc[-1]**: Áp dụng phép cộng tích lũy (cumulative sum) cho cột 'Open' của tập dữ liệu sai phân cấp 1 (test_diff). Kết quả của phép cộng tích lũy được cộng thêm giá trị cuối cùng của tập train_data để khôi phục giá trị ban đầu.
- Tương tự, quá trình khôi phục giá trị ban đầu cũng được thực hiện cho các cột 'High', 'Low', 'Close', và 'Volume' trong tập dữ liệu sai phân cấp 1 (test_diff).
- Kết quả cuối cùng là test_forecast_restored, một bản sao của tập dữ liệu ban đầu với giá trị dự đoán đã được khôi phục từ dạng sai phân cấp 1 trên tập test.

- Dòng code test_forecast_restored = test_forecast_restored.sort_index().iloc[1:] được sử dụng để sắp xếp lại thứ tự các dòng trong tập dữ liệu test_forecast_restored và loại bỏ hàng đầu tiên.
- Cụ thể, các bước thực hiện là:
    - test_forecast_restored.sort_index(): Sắp xếp lại tập dữ liệu test_forecast_restored theo thứ tự của chỉ số (index). Điều này đảm bảo rằng các dòng trong tập dữ liệu được sắp xếp theo thứ tự tăng dần của thời gian.
    - test_forecast_restored.iloc[1:]: Chọn các hàng từ hàng thứ hai trở đi trong tập dữ liệu đã sắp xếp. Hàng đầu tiên được loại bỏ vì giá trị của nó có thể không phù hợp sau khi sắp xếp lại.
    - Kết quả cuối cùng là test_forecast_restored đã được sắp xếp lại và loại bỏ hàng đầu tiên, giúp đảm bảo rằng các giá trị dự đoán đã được khôi phục từ dạng sai phân cấp 1 trên tập test được đối chiếu theo thứ tự thời gian chính xác.

Tại sao sau khi khôi phục giá trị dự đoán thì dự liệu bị mất dòng đầu tiên?

- Khi thực hiện tích lũy (cumsum()) trên dữ liệu sai số (diff), dòng đầu tiên của dữ liệu sẽ bị mất do tính chất của phép tích lũy. Điều này xảy ra vì phép tích lũy tính toán giá trị mới bằng cách cộng dồn giá trị hiện tại với giá trị trước đó.
- Khi thực hiện tích lũy trên dữ liệu, phép tính đầu tiên chỉ sử dụng giá trị hiện tại và không có giá trị trước đó để cộng dồn. Do đó, kết quả của phép tính đầu tiên sẽ không được bảo toàn trong dữ liệu tích lũy.
- Lỗi NaN (Not a Number) xảy ra khi giá trị đầu tiên sau khi khôi phục được tính toán bằng cách cộng dồn dữ liệu sai số (diff) với giá trị cuối cùng của tập train. Nguyên nhân chính là vì dữ liệu sai số không chứa giá trị cho dòng đầu tiên.

Trong kết quả mô hình VAR, các chỉ số AIC (Akaike Information Criterion), BIC (Bayesian Information Criterion) và HQIC (Hannan-Quinn Information Criterion) được cung cấp để đánh giá và lựa chọn mô hình phù hợp.

Các chỉ số này được sử dụng để đánh giá sự phù hợp của các mô hình khác nhau dựa trên hiệu suất dự báo và độ phức tạp của mô hình. Mục tiêu là tìm mô hình có giá trị AIC, BIC, và HQIC thấp nhất, điều đó cho thấy mô hình đó có khả năng dự báo tốt hơn và đồng thời không quá phức tạp.

- AIC: AIC là một tiêu chí thông tin dựa trên lý thuyết thông tin Kullback-Leibler. Mô hình với giá trị AIC thấp hơn được coi là tốt hơn.
- BIC: BIC là một tiêu chí thông tin dựa trên lý thuyết thông tin Bayes. Giống như AIC, mô hình với giá trị BIC thấp hơn được ưu tiên.
- HQIC: HQIC là một phiên bản điều chỉnh của BIC, tập trung vào việc giảm thiểu sai số dự báo. Mô hình với giá trị HQIC thấp hơn được coi là tốt hơn.

Khi so sánh các mô hình VAR, việc lựa chọn mô hình phù hợp sẽ tùy thuộc vào ngữ cảnh và mục tiêu cụ thể của bạn. Thông thường, bạn nên xem xét các giá trị AIC, BIC và HQIC cùng nhau và chọn mô hình có giá trị thấp nhất trong các tiêu chí này.