TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN CƠ TIN HỌC

BÁO CÁO NGHIÊN CỨU KHOA HỌC

Dự đoán năng lượng tiêu thụ ngành luyện thép sử dụng mô hình hồi quy tuyến tính

Nhóm thực hiện: Nhóm 12 – K67A2 Toán tin
Tên thành viên: Đào Mạnh Đức – 22000085
Tạ Đăng Đức – 22000088
Nguyễn Huy Hoàng – 22000095
Giảng viên hướng dẫn: Lê Huy Hoàng

Hà Nội, 12/2024

MỤC LỤC:

TÓM TẮT BÀI NGHIÊN CỨU

Ngành công nghiệp thép đóng vai trò quan trọng trong nhiều lĩnh vực, từ xây dựng cơ sở hạ tầng, giao thông vận tải đến kiến trúc hiện đại. Tuy nhiên, việc dự đoán năng lượng tiêu thụ trong ngành công nghiệp thép vẫn là một thách thức lớn do ảnh hưởng của nhiều yếu tố như loại thép được sản xuất, quy trình sản xuất, và hiệu suất của các nhà máy. Đề tài "Dự đoán năng lượng tiêu thụ trong ngành công nghiệp thép bằng mô hình hồi quy tuyến tính" được thực hiện nhằm giải quyết vấn đề này.

Đề tài kết hợp mô hình hồi quy tuyến tính để dự đoán biến liên tục năng lượng tiêu thụ cụ thể trong đề tài này là điện năng. Đề tài xây dựng một mô hình dự đoán chính xác mức tiêu thụ năng lượng, từ đó hỗ trợ các nhà máy thép tối ưu hóa quá trình sản xuất, giảm thiểu lãng phí năng lượng và góp phần phát triển bền vững.

Cấu trúc của đề tài bao gồm các phần chính:

Phần 1: Tổng quan về vấn đề nghiên cứu, trình bày mục tiêu, phương pháp và phạm vi thực hiện.

Phần 2: Giới thiệu dữ liệu được sử dụng, bao gồm các yếu tố liên quan đến tiêu thụ năng lượng và cách tiền xử lý dữ liệu.

Phần 3: Trình bày lý thuyết mô hình hồi quy tuyến tính.

Phần 4: Trình bày kết quả thực nghiệm, đánh giá độ chính xác của mô hình.

PHẦN 1. TỔNG QUAN BÁO CÁO

Phần 1 sẽ trình bày một cách tổng quan đề tài về: đặt vấn đề, phương pháp thực hiện, đưa ra mục tiêu, phạm vi của để tài cũng như các tiêu chí đánh giá hệ thống v.v.

1.1. Đặt vấn đề

Trong bối cảnh sự phát triển mạnh mẽ của ngành công nghiệp thép, việc dự đoán nhu cầu năng lượng tiêu thụ là một yếu tố quan trọng để tối ưu hóa quy trình sản xuất, giảm thiểu chi phí và bảo vệ môi trường. Trong ngành sản xuất thép, năng lượng tiêu thụ chủ yếu đến từ điện năng và các nguồn năng lượng hóa thạch, do đó việc hiểu rõ và dự đoán chính xác mức tiêu thụ năng lượng có thể giúp các nhà sản xuất điều chỉnh quy trình sản xuất sao cho hiệu quả hơn. Tuy nhiên, việc dự đoán này thường xuyên đối mặt với những thách thức lớn về tính chính xác và khả năng thích ứng với các yếu tố thay đổi không thể đoán trước trong quy trình sản xuất.

Trong ngành sản xuất thép, năng lượng tiêu thụ trong sản xuất thép có sự biến động lớn phụ thuộc vào nhiều yếu tố như công suất, tỷ lệ sử dụng các nguyên liệu, và các yếu tố ngoại vi khác như thời tiết và nhu cầu thị trường. Các phương pháp truyền thống để dự đoán mức tiêu thụ năng lượng như mô hình toán học hoặc thống kê đều gặp phải những hạn chế nhất định trong việc dự đoán các thay đổi nhanh chóng và đột ngột trong nhu cầu năng lượng.

Một trong những phương pháp hiệu quả để giải quyết vấn đề này là sử dụng các mô hình hồi quy tuyến tính để phân tích và dự đoán mức tiêu thụ năng lượng dựa trên các yếu tố có sẵn trong dữ liệu, chẳng hạn như công suất tiêu thụ, mức độ thải CO2, và các yếu tố khác. Tuy nhiên, việc xây dựng mô hình hồi quy chính xác và có thể tái sử dụng trên nhiều trường hợp khác nhau vẫn còn là một thách thức lớn.

1.2. Phương pháp thực hiện

1. Nghiên cứu và phân tích dữ liệu

Tiến hành thu thập và phân tích dữ liệu tiêu thụ năng lượng của các nhà máy sản xuất thép, bao gồm các yếu tố như công suất tiêu thụ, mức độ thải CO2, và các yếu tố môi trường khác. Dữ liệu này sẽ được sử dụng làm cơ sở để xây dựng mô hình hồi quy tuyến tính.

2. Xây dựng mô hình hồi quy tuyến tính

Sử dụng phương pháp hồi quy tuyến tính để phát triển mô hình dự đoán mức tiêu thụ năng lượng. Các biến độc lập như mức độ thải CO2 và công suất tiêu thụ sẽ được đưa vào mô hình để tìm ra mối quan hệ với lượng năng lượng tiêu thụ. Mô hình sẽ được kiểm tra với bộ dữ liệu thực tế để đánh giá độ chính xác.

3. Kiểm tra và đánh giá mô hình

Tiến hành kiểm tra mô hình với bộ dữ liệu kiểm tra để đánh giá khả năng dự đoán chính xác của hệ thống. Các chỉ số như trung bình bình phương sai số

(MSE) và hệ số xác định (R^2) sẽ được sử dụng để đo lường hiệu quả của mô hình. So sánh mô hình hồi quy tuyến tính với mô hình phi tuyến như KNN.

1.3. Mục tiêu của đề tài

Mục tiêu chính của đề tài là xây dựng một mô hình dự đoán năng lượng tiêu thụ cụ thể là điện năng dựa trên các yếu tố như công suất, lượng khí CO2 thải ra, loại tải,… từ đó giúp các cơ quan, doanh nghiệp có thể tối ưu hóa việc sử dụng năng lượng, tiết kiệm chi phí và nâng cao hiệu quả hoạt động.

1.4. Các tiêu chí đánh giá

Mô hình dự đoán sẽ được đánh giá dựa trên các tiêu chí độ chính xác của dự đoán. So sánh kết quả dự đoán của mô hình với giá trị thực tế thông qua các chỉ số như:

- **Trung bình bình phương sai số (MSE)** để đo độ chênh lệch.
- **Hệ số xác định (R^2)** để đánh giá mức độ phù hợp của mô hình.

1.5. Kết luận phần

Phần này đã trình bày tổng quan về đề tài và các phương pháp thực hiện. Các phần tiếp theo sẽ đi vào chi tiết về mô hình dự báo, thiết kế và triển khai hệ thống cũng như các kết quả thử nghiệm thực tế.

PHẦN 2. MÔ TẢ VÀ XỬ LÝ DỮ LIỆU

Nội dung phần 2 sẽ trình bày tổng quan về tập dữ liệu được xử dụng trong mô hình và một số tiền xử lý và phân tích được thực hiện với tập dữ liệu trước khi tập dữ liệu được đưa vào mô hình.

2.1. Mô tả dữ liệu

Tập dữ liệu được sử dụng trong nghiên cứu này được thu thập từ DAEWOO Steel Co. Ltd tại Gwangyang, Hàn Quốc. Công ty này chuyên sản xuất các loại thép cuộn, tấm thép và tấm sắt. Thông tin về tiêu thụ điện năng được lưu trữ trong hệ thống điện toán đám mây của công ty.

Ngoài ra, dữ liệu về tiêu thụ năng lượng trong ngành công nghiệp được lấy từ website của Korea Electric Power Corporation (KEPCO). Trang web này cung cấp các báo cáo tiêu thụ năng lượng với góc nhìn theo ngày, tháng và năm, giúp tạo ra một cái nhìn tổng quan về mức tiêu thụ năng lượng của ngành.

date	Usage_kW	Lagging_C	Leading_C	CO2(tCO2	Lagging_C	Leading_C	NSM	WeekStatu	Day_of_we	Load_Type
1/1/2018 0:15	3.17	2.95	0	0	73.21	100	900	Weekday	Monday	Light_Load
1/1/2018 0:30	4	4.46	0	0	66.77	100	1800	Weekday	Monday	Light_Load
1/1/2018 0:45	3.24	3.28	0	0	70.28	100	2700	Weekday	Monday	Light_Load
1/1/2018 1:00	3.31	3.56	0	0	68.09	100	3600	Weekday	Monday	Light_Load
1/1/2018 1:15	3.82	4.5	0	0	64.72	100	4500	Weekday	Monday	Light_Load
1/1/2018 1:30	3.28	3.56	0	0	67.76	100	5400	Weekday	Monday	Light_Load

Figure 1. Mẫu dữ liệu trong tập dữ liệu

2.1.1. Lý thuyết về tập dữ liệu

Tập dữ liệu đóng vai trò quan trọng trong việc xây dựng và đánh giá hiệu quả của mô hình dự đoán. Một tập dữ liệu tiêu chuẩn bao gồm hai phần chính: đặc tính (features) và dữ liệu (observations).

- **Đặc tính (Features):** Đây là các thuộc tính hoặc biến số, được sử dụng làm đầu vào cho mô hình. Các đặc tính này phải phản ánh chính xác mối quan hệ giữa các yếu tố và biến mục tiêu.
- **Dữ liệu (Observations):** Đây là các giá trị cụ thể thu thập được cho từng đặc tính. Số lượng dữ liệu lớn và chất lượng cao là yếu tố quan trọng để mô hình có thể học được mối quan hệ giữa các đặc tính và biến mục tiêu.

2.1.2 Mô tả các đặc tính trong tập dữ liệu

Tập dữ liệu sử dụng trong nghiên cứu này bao gồm các đặc tính sau:

- **Date:** Đặc trưng này biểu thị ngày và giờ ghi nhận dữ liệu, được định dạng theo kiểu MM/DD/YYYY HH:MM:SS AM/PM, ví dụ: 1/1/2018 12:15:00 AM.
- **Usage_kWh:** Đây là biến mục tiêu, biểu diễn lượng điện năng tiêu thụ trong một khoảng thời gian cụ thể. Đơn vị tính là kilowat-giờ (kWh).

- **Lagging_Current_Reactive_Power_kVarh**: Biểu thị công suất phản kháng của dòng điện trễ. Công suất phản kháng xảy ra khi dòng điện và điện áp không đồng pha nhau, do các tải cảm kháng như động cơ điện, cuộn dây biến áp hoặc thiết bị tương tự. Nếu giá trị này quá cao, nó có thể dẫn đến tổn thất năng lượng. Đơn vị tính bằng kVarh.
- **Leading_Current_Reactive_Power_kVarh**: Biểu thị công suất phản kháng của dòng điện sớm. Công suất phản kháng sớm xảy ra khi dòng điện "sớm pha" hơn so với điện áp, thường xuất hiện do các tải dung kháng như tụ điện hoặc thiết bị điện tử. Nó có thể giúp cân bằng công suất phản kháng với công suất phản kháng của dòng điện trễ trong một hệ thống điện. Đơn vị tính bằng kVarh.
- **CO2(tCO2):** Biến này phản ánh mức độ khí thải carbon dioxide (tấn CO2). Đây là một chỉ số quan trọng trong việc đánh giá tác động môi trường của các hoạt động công nghiệp. Đơn vụ tính bằng ppm.
- **Lagging_Current_Power_Factor:** Biểu thị hệ số công suất của dòng điện trễ, là một giá trị không có đơn vị dao động từ 0 đến 1. Hệ số công suất là tỷ lệ giữa công suất thực (kW) và tổng công suất (kVA) trong một hệ thống điện. Khi dòng điện "trễ pha" so với điện áp (do tải cảm kháng như động cơ hoặc máy biến áp), hệ số công suất sẽ giảm, dẫn đến hiệu quả sử dụng điện thấp hơn. Đơn vị tính bằng %.
- **Leading_Current_Power_Factor:** Biểu thị hệ số công suất của dòng điện sớm, cũng là một giá trị không có đơn vị dao động từ 0 đến 1. Khi dòng điện "sớm pha" hơn so với điện áp (do tải dung kháng như tụ điện hoặc thiết bị điện tử), hệ số công suất cũng bị ảnh hưởng. Đơn vị tính bằng %.
- **NSM:** Thể hiện thời gian trong ngày, được chuẩn hóa dưới dạng giây kể từ nửa đêm (midnight). Giá trị trong cột này nằm trong khoảng từ 0 đến 86399. Ví dụ 0 tương ứng với 12:00:00 AM (nửa đêm) và 86399 tương ứng với 11:59:59 PM (gần cuối ngày).
- **WeekStatus:** Biểu thị trạng thái của ngày trong tuần, giúp xác định xem ngày đó thuộc ngày trong tuần (Weekday) hay ngày cuối tuần (Weekend).
- **Day_of_week:** Biểu thị tên ngày trong tuần dưới dạng văn bản, ví dụ: "Sunday", "Monday", "Tuesday",..., "Saturday".
- **Load_type:** Chỉ ra mức tải năng lượng tiêu thụ của nhà máy trong từng khoảng thời gian nhất định. Mức tải này được phân loại thành ba nhóm chính: Light Load (ít hoạt động sản xuất), Medium Load (hoạt động với công suất bình thường), và Maximum Load (hoạt động với công suất tối đa).

2.2. Tiền xử lý dữ liệu

2.2.1. Kiểm tra dữ liệu bị thiếu

- Dữ liệu bị thiếu (missing data) xảy ra khi một hoặc nhiều giá trị trong tập dữ liệu không được ghi nhận. Quan sát các cột hoặc hàng trong tập dữ liệu để phát hiện các giá trị trống (ví dụ: "Null", "N/A")

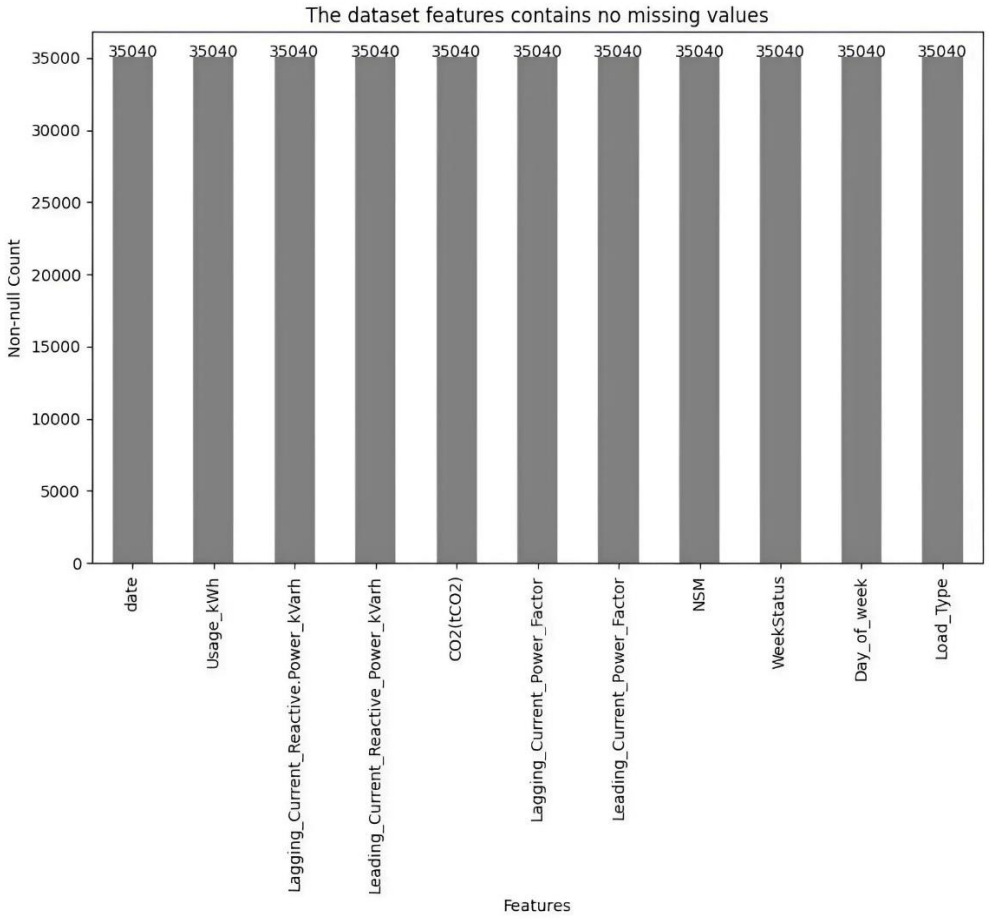


Figure 2. Số giá trị không phải giá trị null của mỗi cột trong tập dữ liệu

Theo biểu đồ, ta có thể thấy trong tập dữ liệu, các cột đều đủ giá trị

- Cách xử lý dữ liệu bị thiếu:
  1. Loại bỏ hàng hoặc cột chứa giá trị thiếu.
  2. Thay thế giá trị thiếu bằng giá trị trung bình(Mean) hoặc giá trị trung vị (Median).

2.2.2. Xử lý dữ liệu không định lượng

- Dữ liệu định lượng và dữ liệu định tính
  1. Dữ liệu định lượng là dữ liệu có thể đo lường hoặc biểu diễn bằng các con số.
     Ví dụ: Usage_kWh, Lagging_Current, Leading_Current, CO2, NSM trong tập dữ liệu.
  2. Dữ liệu định tính là dữ liệu biểu diễn các đặc điểm, tính chất, hoặc danh mục mà không mang giá trị số học. Các giá trị định tính thường được biểu diễn dưới dạng danh mục (categorical) hoặc chuỗi ký tự.

Ví dụ: Date, WeekStatus, Day_of_week, Load_type trong tập dữ liệu.

- Vấn đề với dữ liệu không định lượng (định tính) là không thể trực tiếp sử dụng làm đầu vào cho các mô hình học máy hoặc hồi quy, vì các thuật toán yêu cầu dữ liệu phải ở dạng số (numerical). Do đó, các cột dữ liệu không định lượng cần được chuyển đổi thành dạng số thông qua các phương pháp mã hóa dữ liệu.
- Các phương pháp xử lý

1. *Mã hóa nhị phân (Binary Encoding)*

Mã hóa nhị phân thường được áp dụng cho các cột dữ liệu định tính có nhiều danh mục riêng biệt (categorical variables). Với phương pháp này, mỗi danh mục sẽ được biểu diễn thành một cột riêng biệt, và các giá trị trong cột đó là nhị phân (0 hoặc 1).

Với tập dữ liệu của ta, ta sẽ áp dụng phương pháp này với các cột Load_Type và Day_of_week. Dưới đây là ví dụ áp dụng phương pháp này cho cột Load_type và với cột Day_of_week ta cũng áp dụng tương tự.

Load_Type	Light_Load	Medium_Load	Maximum_Load
Light_Load	1	0	0
Medium_Load	0	1	0
Maximum_Load	0	0	1

Figure 3. Cột Load_Type trước và sau khi sử dụng phương pháp mã hóa nhị phân

2. *Mã hóa danh mục (Categorical Encoding)*

Mã hóa danh mục được áp dụng cho các cột dữ liệu định tính nhị phân hoặc có rất ít danh mục riêng biệt. Phương pháp này gán các giá trị số nguyên (0, 1, 2,...) cho từng danh mục, tạo thành một cột số duy nhất.

Cột WeekStatus được áp dụng phương pháp này với cột WeekStatus duy nhất và giá trị Weekend được gán giá trị 1 và giá trị Weekday được gán giá trị 0.

3. *Tách dữ liệu chuỗi thời gian (Datetime Splitting)*

Tách dữ liệu chuỗi thời gian (như ngày, giờ) thành các thành phần riêng biệt có ý nghĩa. Điều này hữu ích khi dữ liệu thời gian cần được phân tích ở các mức chi tiết hơn như năm, tháng, hoặc giờ.

Cột Date sẽ được biến đổi như sau:

- Ban đầu là một chuỗi dạng 1/1/2018 12:15:00 AM.
- Chuyển đổi thành dạng datetime và tách thành các cột mới:
  - year: Năm (ví dụ: 2018).
  - month: Tháng (1).

- day: Ngày trong tháng (1).
- hour: Giờ trong ngày (12).
- minute: Phút trong giờ (15).

2.3. Phân tích tương quan

2.3.1. Lý thuyết về phân tích tương quan

Phân tích tương quan là một phương pháp thống kê quan trọng để đánh giá mức độ và hướng mối quan hệ giữa các biến trong một bộ dữ liệu. Mục tiêu của phân tích tương quan là xác định liệu một biến có ảnh hưởng đến biến khác hay không, và nếu có, mức độ ảnh hưởng là mạnh hay yếu, và liệu mối quan hệ này có phải là tuyến tính hay không.

Một trong những hệ số tương quan phổ biến là hệ số tương quan Spearman, được sử dụng khi mối quan hệ giữa các biến không cần thiết phải tuyến tính, hoặc khi dữ liệu không thỏa mãn các giả định của hệ số tương quan Pearson (như phân phối chuẩn). Hệ số tương quan Spearman đo lường mối quan hệ giữa các xếp hạng của hai biến, thay vì mối quan hệ giữa các giá trị thực tế của chúng.

Công thức tính hệ số tương quan Spearman là:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

Trong đó:

- d_i là độ lệch giữa các giá trị xếp hạng của hai biến tại quan sát thứ i
- n là tổng số quan sát trong dữ liệu.

Hệ số tương quan Spearman có giá trị nằm trong khoảng từ -1 đến 1:

- $\rho > 0$: Khi một biến tăng, biến kia có xu hướng tăng.
- $\rho < 0$: Khi một biến tăng, biến kia có xu hướng giảm.
- $\rho = 0$: Không có mối quan hệ.

2.3.2. Mục đích của phân tích tương quan

Phân tích tương quan giữa các biến độc lập và biến phụ thuộc là bước quan trọng trong việc xây dựng mô hình hồi quy tuyến tính. Mục đích của việc này là để:

- **Đánh giá mức độ mối quan hệ giữa các biến:** Khi xây dựng mô hình hồi quy, ta cần hiểu rõ mối quan hệ giữa các biến đầu vào (biến độc lập) và biến đầu ra (biến phụ thuộc). Phân tích tương quan giúp xác định những biến có mối quan hệ mạnh với biến phụ thuộc.
- **Giảm thiểu đa cộng tuyến:** Nếu hai hoặc nhiều biến độc lập có mối tương quan cao với nhau, điều này có thể gây ra hiện tượng đa cộng tuyến trong mô hình hồi quy tuyến tính. Khi đó, một số biến có thể không đóng góp nhiều thông tin mới, và việc loại bỏ hoặc kết hợp chúng có thể giúp cải thiện chất lượng mô hình.

- **Chọn lựa biến quan trọng:** Phân tích tương quan giúp xác định các biến nào có ảnh hưởng mạnh nhất đến biến mục tiêu, giúp tối ưu hóa mô hình hồi quy bằng cách chỉ sử dụng các biến quan trọng.

2.3.3. Áp dụng phân tích tương quan

Để hiểu rõ mối quan hệ giữa các biến trong bộ dữ liệu năng lượng của ngành thép, chúng ta đã sử dụng hệ số tương quan Spearman để phân tích mối quan hệ giữa các đặc trưng như Usage_kWh, Lagging_Current, Leading_Current, CO2(tCO2), và các đặc trưng khác như Load_Type WeekStatus, Day_of_week.

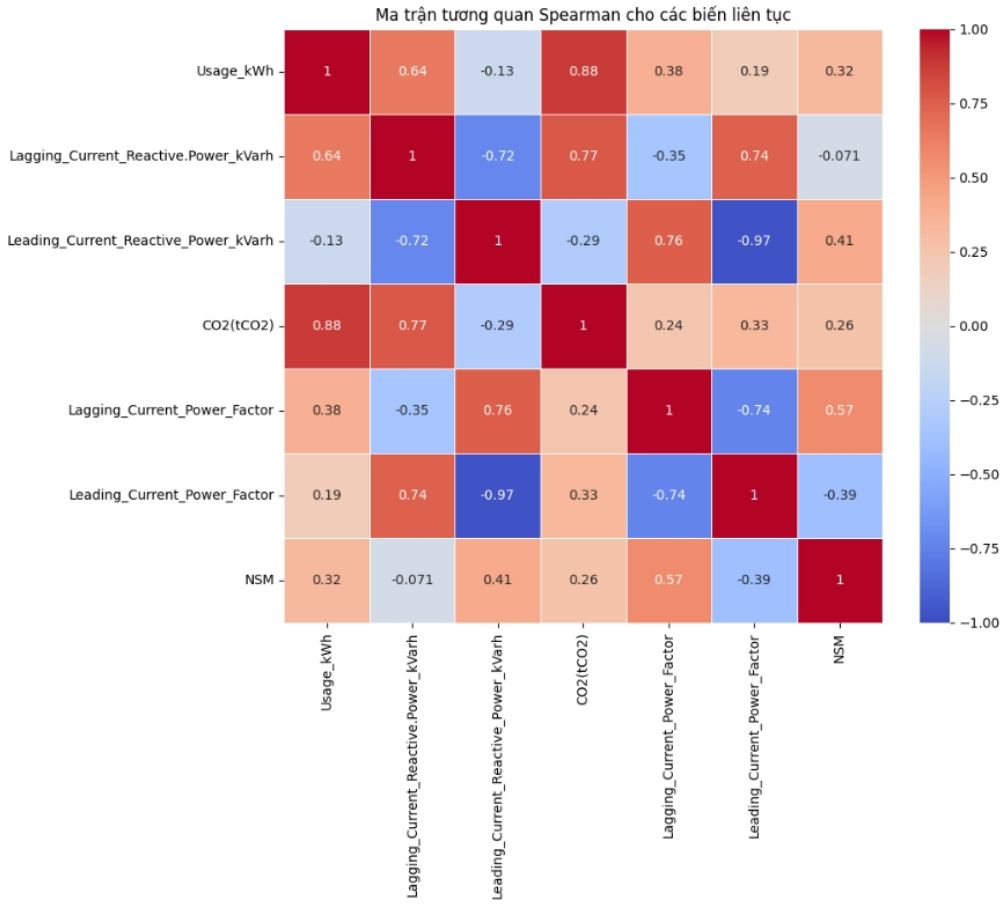


Figure 4. Ma trận tương quan Spearman cho các biến liên tục

	Biến nhị phân	Biến liên tục	Tương quan Spearman
0	WeekStatus	Usage_kWh	-0.313772
1	WeekStatus	Lagging_Current_Reactive.Power_kVarh	-0.376717
2	WeekStatus	Leading_Current_Reactive_Power_kVarh	0.277706
3	WeekStatus	CO2(tCO2)	-0.294912
4	WeekStatus	Lagging_Current_Power_Factor	0.141790
5	WeekStatus	Leading_Current_Power_Factor	-0.308554

Figure 5. Tương quan spearman với biến nhị phân

	Biến phân loại	Biến liên tục	Tương quan Spearman
0	Day_of_week	Usage_kWh	0.023589
1	Day_of_week	Lagging_Current_Reactive.Power_kVarh	0.029261
2	Day_of_week	Leading_Current_Reactive_Power_kVarh	-0.020040
3	Day_of_week	CO2(tCO2)	0.023800
4	Day_of_week	Lagging_Current_Power_Factor	-0.005033
5	Day_of_week	Leading_Current_Power_Factor	0.021608
6	Load_Type	Usage_kWh	0.447016
7	Load_Type	Lagging_Current_Reactive.Power_kVarh	0.154132
8	Load_Type	Leading_Current_Reactive_Power_kVarh	0.210400
9	Load_Type	CO2(tCO2)	0.456960
10	Load_Type	Lagging_Current_Power_Factor	0.437593
11	Load_Type	Leading_Current_Power_Factor	-0.159258

Figure 6. Tương quan spearman với biến phân loại

Qua kết quả trên, ta có thể các điểm đáng chú ý như sau:

- CO2 có tương quan mạnh nhất với biến mục tiêu Usage_kWh với hệ số là 0.88, và có thể xem xét về vấn đề đa cộng tuyến hoặc có thể sử dụng Ridge hoặc Lasso Regression để xử lý vấn đề này
- Các biến liên quan đến công suất phản kháng (Reactive Power) và hệ số công suất (Power Factor) có mối quan hệ phức tạp và đan xen với nhau
- NSM có mối tương quan không quá mạnh với các biến khác, phần lớn ở mức trung bình hoặc yếu
- Day_of_week có tương quan rất yếu với các biến khác (<0.03), có thể xem xét loại bỏ vì khả năng dự đoán thấp
- Load_Type cho thấy tương quan khá tốt với nhiều biến (0.4-0.5), nên được xem xét giữ lại trong mô hình.

2.4. Phân chia dữ liệu cho mô hình huấn luyện và đánh giá

2.4.1. Chia tập dữ liệu

Chia dữ liệu thành hai phần: huấn luyện và kiểm tra, giúp mô hình học các mối quan hệ và tổng quát hóa tốt trên dữ liệu chưa thấy.

- Phần Huấn Luyện (Training Set) được sử dụng để huấn luyện mô hình, tức là cho mô hình học cách tìm kiếm các mối quan hệ giữa các đặc trưng đầu vào và biến mục tiêu. Chiếm 60-80% dữ liệu. Phần này càng lớn, mô hình càng học được các đặc trưng sâu hơn.
- Phần Kiểm Tra (Test Set) được sử dụng để đánh giá hiệu quả của mô hình sau khi đã huấn luyện. Phần này giúp chúng ta kiểm tra xem mô hình có thể tổng quát hóa tốt trên dữ liệu chưa thấy trước đó hay không. Chiếm 20-40% dữ liệu. Giúp kiểm tra khả năng tổng quát hóa của mô hình.

2.4.2. Tại sao cần phải chia dữ liệu?

Chia dữ liệu thành hai phần là một phương pháp quan trọng trong học máy vì các lý do sau:

- **Tránh overfitting (quá khớp):** Nếu mô hình chỉ học trên toàn bộ dữ liệu mà không có tập kiểm tra, có thể dẫn đến việc mô hình học quá kỹ các đặc trưng của dữ liệu huấn luyện, và do đó không thể tổng quát tốt trên dữ liệu mới. Chia dữ liệu giúp kiểm tra khả năng tổng quát của mô hình.
- **Đánh giá độ chính xác của mô hình:** Kiểm tra mô hình trên dữ liệu chưa thấy, đánh giá hiệu suất khách quan.
- **Chẩn đoán và cải thiện mô hình:** Nếu mô hình kém hiệu quả trên tập kiểm tra, có thể điều chỉnh mô hình hoặc cải thiện quá trình tiền xử lý.

PHẦN 3. MÔ HÌNH HỒI QUY TUYẾN TÍNH

3.1. Khái quát về hồi quy:

Thuật ngữ "hồi quy" đã được Francis Galton sử dụng vào thế kỷ 19 để nhằm mô tả một hiện tượng sinh học. Trong các nghiên cứu về di truyền học Galton đã đưa ra khái niệm "hồi quy" này khi phân tích mối quan hệ giữa chiều cao của cha mẹ và con cái. Ông quan sát thấy rằng, con cái của những có nhân có chiều cao rất cao hoặc rất thấp thì thường có xu hướng có chiều cao gần với giá trị trung bình của quần thể hơn so với chiều cao của cha mẹ chúng. Hiện tượng này, được gọi là "hồi quy về giá trị trung bình" ("regression to the mean"), đã thu hút được nhiều quan tâm và đặt nền móng cho nhiều nghiên cứu thống kê sau này.

3.2. Hồi quy tuyến tính:

Hồi quy tuyến tính là một mô hình trong thống kê nhằm ước tính mối quan hệ tuyến tính giữa một biến phụ thuộc với một hoặc nhiều biến độc lập. Một mô hình có chính xác một biến độc lập thì được gọi là hồi quy tuyến tính đơn; còn một mô hình có hai hoặc nhiều biến độc lập thì được gọi là hồi quy tuyến tính bội. Trong bài báo cáo này, chúng tôi sẽ đề cập đến cả 2 mô hình.

3.2.1. Mô hình hồi quy tuyến tính đơn

1. Phương trình hồi quy tuyến tính đơn

Để bắt đầu, ta có một phương trình đơn giản như sau:

$$y = \beta_0 + \beta_1 x + u \quad (1)$$

Trong đó:

y: biến phụ thuộc (hoặc biến phản hồi).

x: biến độc lập (hoặc biến hồi quy).

β_0: hệ số chặn, nó sẽ cho ta biết giá trị của biến phụ thuộc khi biến độc lập bằng 0.

β_1: hệ số độ dốc, nó sẽ cho ta biết giá trị của biến phụ thuộc thay đổi như thế nào khi biến độc lập thay đổi 1 đơn vị.

u: sai số hoặc là nhiễu, đại diện cho các yếu tố khác có ảnh hưởng đến biến phụ thuộc.

2. Hàm hồi quy tổng thể (Population Regression Function - PRF)

Để hiểu rõ hơn về bản chất mối quan hệ giữa biến phụ thuộc và biến độc lập trong toàn bộ quần thể mà chúng ta đang quan tâm, chúng ta cần xem xét khái niệm về hàm hồi quy tổng thể (PRF). PRF mô tả giá trị kỳ vọng (trung bình) của biến phụ thuộc ứng với mỗi giá trị của biến độc lập.

Giả sử ta có 2 giả định sau:

- $E(u) = 0$: Kỳ vọng của sai số ngẫu nhiên bằng 0. Giả định này có nghĩa là yếu tố không được mô hình hóa có tác động trung bình lên y là bằng 0.

- $E(u|x) = E(u)$: sai số ngẫu nhiên u độc lập về kỳ vọng với x, nghĩa là sai số u không phụ thuộc vào giá trị x. Giả định này nhằm bảo đảm rằng mọi biến động không thể giải thích của y đều là ngẫu nhiên và không có mối liên hệ nào với x.

Với 2 giả định trên, ta thu được phương trình hàm hồi quy tổng thể (PRF):

$$E(y|x) = \beta_0 + \beta_1 x \quad (2)$$

3. Hàm hồi quy mẫu (Sample Regression Function - SRF)

Trong thực tế, không thể nghiên cứu toàn bộ tổng thể, do đó chỉ có thể đưa ra dạng của hàm hồi quy tổng thể chứ không thể xác định hàm này một cách chính xác. Để ước lượng hàm hồi quy tổng thể phải dựa vào một mẫu được rút ra ngẫu nhiên từ tổng thể. Hàm hồi quy được xây dựng dựa trên mẫu được gọi là hàm hồi quy mẫu (SRF). Và ra không thể ước lượng một cách chính xác PRF dựa trên mẫu ngẫu nhiên, và đối với các mẫu ngẫu nhiên khác nhau sẽ cho ra những hàm SRF khác nhau.

Giả sử ta có phương trình (2), và (x_i, y_i) là đại diện cho một cặp giá trị quan sát của biến độc lập x và biến phụ thuộc y trong tập dữ liệu; với i nằm trong khoảng tổng số quan sát. Khi đó, ta có được phương trình:

$$\widehat{y_i} = \widehat{\beta_0} + \widehat{\beta_1} x_i \quad (3)$$

Với:
$\widehat{y_i}$: là giá trị dự đoán của biến phụ thuộc cho quan sát thứ i.
$\widehat{\beta_0}$ và $\widehat{\beta_1}$: là các ước lượng của lần lượt hệ số chặn và hệ số góc.

Mục tiêu của chúng ta là tìm các ước lượng $\widehat{\beta_0}$ và $\widehat{\beta_1}$ sao cho SRF là một đại diện tốt nhất cho PRF. Một phương pháp phổ biến để thực hiện điều này chính là phương pháp bình phương tối thiểu.

4. Ước lượng bình phương nhỏ nhất thông thường (Ordinary Least Square - OLS)

Giả sử $\widehat{\beta_0}$ và $\widehat{\beta_1}$ là các ước lượng cần tìm cho lần lượt các hệ số β_0 và β_1 trong mô hình hồi quy tuyến tính. Ta sẽ đi tìm $\widehat{\beta_0}$ và $\widehat{\beta_1}$ sao cho các giá trị dự đoán $\widehat{y_i}$ càng gần với giá trị thực y_i càng tốt, tức là các phần dư giữa giá trị thực và giá trị dự đoán

$$e_i = y_i - \widehat{y_i} = y_i - (\widehat{\beta_0} + \widehat{\beta_1} x_i) \quad (4)$$

càng nhỏ càng tốt. Do các $e_i, i = \overline{1,n}$ có thể âm, có thể dương, vì thế ta sẽ đi tìm các ước lượng $\widehat{\beta_0}$ và $\widehat{\beta_1}$ sao cho tổng bình phương các phần dư là nhỏ nhất.

Vậy, ý tưởng của phương pháp OLS là đi tìm $\widehat{\beta_0}$ và $\widehat{\beta_1}$ để thỏa mãn hàm mục tiêu:

$$\min \sum_{i=1}^{n} e_i^2 = \min \sum_{i=1}^{n} [y_i - (\widehat{\beta_0} + \widehat{\beta_1} x_i)]^2 \quad (5)$$

Figure 7:Minh họa ý tưởng OLS (Nguồn: Analytics Yogi)

Áp dụng giải tích, ta có thể tính ra được phương trình (5) đạt cực tiểu với các ước lượng $\widehat{\beta_0}$ và $\widehat{\beta_1}$ lần lượt là:

$$\widehat{\beta_1} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$
$$\widehat{\beta_0} = \bar{y} - \widehat{\beta_1}\bar{x} \quad (6)$$

3.2.2. Mô hình hồi quy tuyến tính bội

Như đã thảo luận, mô hình hồi quy tuyến tính đơn cho phép chúng ta nghiên cứu mối quan hệ giữa một biến phụ thuộc và một biến độc lập duy nhất. Tuy nhiên, trong thực tế, biến phụ thuộc thường bị ảnh hưởng bởi nhiều yếu tố khác nhau. Do đó, chúng ta cần mở rộng mô hình hồi quy tuyến tính đơn để có thể xem xét tác động của nhiều biến độc lập cùng một lúc. Điều này dẫn đến mô hình hồi quy tuyến tính bội.

Cũng với cách tiếp cận như của mô hình hồi quy tuyến tính đơn, mô hình hồi quy tuyến tính bội cũng được chúng tối làm điều tương tự.

1. Hàm hồi quy tổng thể

PRF là một mô hình lý thuyết mô tả mối quan hệ kỳ vọng giữa biến phụ thuộc và biến độc lập trong tổng thể, với mô hình hồi quy tuyến tình bội, ta có phương trình hồi quy tổng thể kỳ vọng được biểu diễn như sau:

$$E(y_i) = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik}$$

Tuy nhiên trên thực tế, các quan sát của chúng ta luôn có sai số do các yếu tố ngẫu nhiên nên phương trình hồi quy tổng thể lại được biểu diễn là:

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik} + u_i$$

Viết chúng lại dưới dạng ma trận, ta thu được như sau:

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & \dots & x_{1k} \\ 1 & x_{21} & \dots & x_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \dots & x_{nk} \end{pmatrix} \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_k \end{pmatrix} + \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix}$$

hay

$$Y = X\beta + u \tag{7}$$

2. Hàm hồi quy mẫu

Sau khi tìm hiểu về PRF, chúng ta xem xét đến SRF, là ước lượng của PRF dựa trên dữ liệu mẫu, ta có phương trình hồi quy mẫu dự đoán là:

$$\widehat{y_i} = \widehat{\beta_0} + \widehat{\beta_1} x_{i1} + \cdots + \widehat{\beta_k} x_{ik}$$

Tuy nhiên, trong thực tế thì khi thu thập dữ liệu mẫu, ta sẽ thu được một phương trình SRF đầy đủ hơn, bao gồm cả phần dư e, do đó phương trình hồi quy mẫu sẽ được biểu diễn là:

$$\widehat{y_i} = \widehat{\beta_0} + \widehat{\beta_1} x_{i1} + \cdots + \widehat{\beta_k} x_{ik} + e_i$$

3. Ước lượng bình phương nhỏ nhất (OLS)

Cũng cùng ý tưởng với OLS của mô hình hồi quy tuyến tính đơn, với mô hình hồi quy tuyến tính bội, ta sẽ đi tìm các hệ số hồi quy ước lượng $\widehat{\beta_j}$ với ($j = \overline{0,k}$) sao cho hàm mục tiêu sau được thỏa mãn:

$$\min \sum_{i=1}^{n} e_i^2 = \min \sum_{i=1}^{n} (y_i - (\widehat{\beta_0} + \widehat{\beta_1} x_{i1} + \cdots + \widehat{\beta_k} x_{ik}))^2 \tag{8}$$

hay viết dưới dạng ma trận, ta thu được:

$$\min ||Y - X\hat{\beta}||^2$$

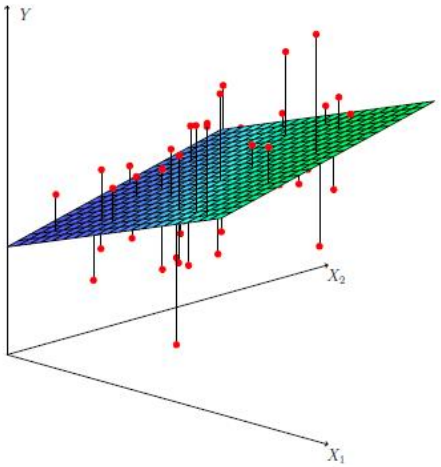


Figure 8: Minh họa ý tưởng OLS (Nguồn: An Introduction to Statistical Learning)

Áp dụng giải tích, ta có thể tính toán ra được các giá trị $\widehat{\beta_J}$ như sau:

$$\widehat{\beta_J} = (X^T X)^{-1} X^T Y \quad (9)$$

3.2.3. Độ phù hợp của mô hình

Sau khi ước lượng được các tham số của mô hình hồi quy bằng phương pháp OLS, một câu hỏi đặt ra là: mô hình này phù hợp với dữ liệu đến mức nào? Để trả lời câu hỏi này, chúng ta cần đánh giá độ phù hợp của mô hình, tức là xem xét mức độ mà mô hình có thể giải thích được sự biến thiên của biến phụ thuộc.

Trước hết, ta có 3 khái niệm sau:

- Tổng bình phương độ lệch (Total Sum of Square - TSS):
  - TSS đo lường tổng sự biến thiên của biến phụ thuộc y so với giá trị của nó.
  - Công thức:

$$TSS = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

- Tổng bình phương độ lêch được giải thích (Explained Sum of Squares - ESS):
  - ESS đo lường sự biến thiên của y được giải thích bởi mô hình.
  - Công thức:

$$ESS = \sum_{i=1}^{n} (\hat{y_i} - \bar{y})^2$$

- Tổng bình phương độ lệch phần dư (Residual Sum of Squares – RSS):
  - RSS đo lường sự biến thiên của y mà mô hình không thể giải thích, nó là sai số của mô hình.

- Công thức:

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Từ đó, qua một số phép biến đổi, ta có thể chứng minh mối quan hệ giữa 3 đại lượng là:

$$TSS = ESS + RSS \quad (10)$$

Để có một cái nhìn toàn diện hơn về độ phù hợp của mô hình, ta sẽ cùng thảo luận về 2 chỉ số: R-squared và Mean Squared Error.

- Hệ số xác định (R-squared):

Hệ số xác định (ký hiệu là R^2) là một thước đo quan trọng cho biết tỷ lệ phần trăm sự biến thiên của biến phụ thuộc được giải thích bởi mô hình. Nó được tính bằng công thức:

$$R^2 = \frac{ESS}{TSS} = 1 - \frac{RSS}{TSS}$$

R^2 có giá trị từ 0 đến 1, trong đó khi R^2 tiến càng gần 1 thì mô hình càng giải thích được nhiều sự biến thiên của biến phụ thuộc và do đó, độ phù hợp càng cao.

- Mean Squared Error (MSE):

MSE là một thước đo khác để đánh giá mức độ phù hợp của mô hình. MSE đo lường giá trị trung bình của bình phương các sai số giữa giá trị thực và giá trị dự đoán. Nó được tính bằng công thức:

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

Như đã đề cập trước đó, mục tiêu của hồi quy tuyến tính là đi tìm các giá trị ước lượng sao cho tổng bình phương phần dư là nhỏ nhất có thể; do vậy nếu MSE càng nhỏ thì độ phù hợp của mô hình càng cao, mô hình càng chính xác.

3.3. Ridge Regression và Lasso Regression

Trong các phần trước, chúng ta đã thảo luận về mô hình hồi quy tuyến tính và phương pháp ước lượng OLS. Mặc dù OLS là một phương pháp mạnh mẽ và được sử dụng rộng rãi, nhưng nó có thể gặp phải một số hạn chế, đặc biệt là khi chúng ta đối mặt với các mô hình phức tạp với nhiều biến độc lập. Trong những trường hợp này, mô hình có thể bị overfitting (quá khớp), dẫn đến việc mô hình

hoạt động tốt trên dữ liệu huấn luyện nhưng lại kém hiệu quả trên dữ liệu mới. Để khắc phục vấn đề này, chúng ta có thể sử dụng các phương pháp điều chuẩn (regularization).

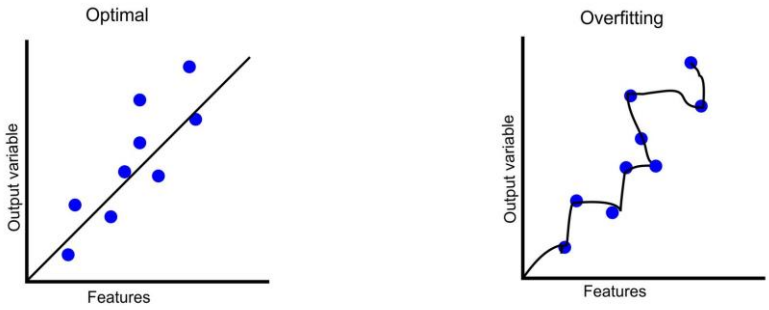


Figure 9: Minh họa overfitting (Nguồn freeCodecamp)

3.3.1. Khái quát về phương pháp điều chuẩn (Regularization)

Khái niệm: Regularization là một kỹ thuật để giảm độ phức tạp của mô hình và hạn chế overfitting bằng cách thêm một yếu tố phạt (penalty) vào hàm mục tiêu (ở đây là RSS) mà chúng ta muốn tối thiểu hóa.

Mục đích: Mục đích của regularization là để tạo ra một mô hình đơn giản hơn, có khả năng khái quát tốt hơn, tức là có thể hoạt động tốt trên dữ liệu mới.

Và hai trong số các phương pháp regularization thì phổ biến nhất là Lasso và Ridge.

3.3.2. Ridge Regression

Ridge Regression thêm vào hàm mục tiêu một yếu tố phạt bằng bình phương của các hệ số hồi quy, tức:

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y_i})^2 + \alpha \|\beta\|_2^2$$

Yếu tố phạt này sẽ có tác dụng co lại các hệ số hồi quy, làm giảm độ lớn của chúng và hạn chế overfitting, tuy nhiên, các hệ số vẫn sẽ khác 0. Để hiểu một cách đơn giản là nó sẽ làm giảm bớt sự phụ thuộc của đa cộng tuyến, tránh trường hợp chỉ một vài yếu tố mà quyết định cả mô hình, gây sai lệch cho khả năng giải thích của mô hình.

3.3.3. Lasso Regression

Lasso Regression thêm vào hàm mục tiêu một yếu tố phạt bằng trị tuyệt đối của các hệ số hồi quy, tức:

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y_i})^2 + \alpha \|\beta\|_1$$

Yếu tố phạt nàu sẽ có tác dụng co lại các hệ số hồi quy, và một số hệ số có thể bị co về 0. Điều này dẫn đến việc Lasso Regression có thể như là một cách lựa chọn những biến quan trọng và bỏ đi những biến ít quan trọng hơn.

3.3.4. Nhận xét

- Cả Lasso và Ridge đều là các phương pháp regularization, nhằm giảm overfitting bằng cách thêm các yếu tố phạt vào hàm mục tiêu.
- Tuy nhiên điểm khác biệt chính là cách chúng thêm yếu tố phạt; khi mà Ridge sử dụng bình phương còn Lasso thì trị tuyệt đối. Điều này khiến cho Lasso có khả năng chọn biến còn Ridge thì không.
- Đối với những loại hồi quy này thì chúng ta cần tinh chỉnh hệ số α để tìm ra một hệ số là tốt nhất cho từng bộ dữ liệu.
- Nếu dữ liệu bị quá khớp nặng, ta sẽ tăng hệ số α lên. Nếu mô hình không bị quá khớp thì ta có thể lựa chọn α gần 0, tức là phương trình hồi quy tương đương với hồi quy tuyến tính đa biến.
- α sẽ được lựa chọn dựa vào một kỹ thuật được gọi là k-fold cross validation.

PHẦN 4. KIỂM THỬ VÀ ĐÁNH GIÁ KẾT QUẢ

4.1. Kết quả khi chạy các mô hình

Sau khi chạy thực nghiệm sử dụng các mô hình Linear Regression, Ridge Regression, Lasso Regression và KNN ta thu được các chỉ số thể hiện độ chính xác của mô hình như sau :

```
LinearRegression:
  MSE = 19.8290
  R2 = 0.9823
  Time = 0.0765 seconds

Ridge:
  MSE = 19.8283
  R2 = 0.9823
  Time = 0.0045 seconds

Lasso:
  MSE = 22.2366
  R2 = 0.9801
  Time = 0.0256 seconds

KNN:
  MSE = 25.3257
  R2 = 0.9774
  Time = 0.0020 seconds
```

Nhóm chúng tôi cũng tiến hành chạy thực nghiệm mô hình trên một tập dữ liệu mới và cho ra kết quả như sau:

20 Dự đoán đầu tiên của mô hình LinearRegression:

	Timestamp	Actual	Predicted
0	2019-01-01 08:00:00	4.18	4.203545
1	2019-01-01 08:15:00	2.95	3.174519
2	2019-01-01 08:30:00	4.07	4.219509
3	2019-01-01 08:45:00	5.51	5.974764
4	2019-01-01 09:00:00	2.99	3.026050
5	2019-01-01 09:15:00	3.74	3.569799
6	2019-01-01 09:30:00	2.95	2.806548
7	2019-01-01 09:45:00	4.18	3.884658
8	2019-01-01 10:00:00	4.54	4.622816

20 Dự đoán đầu tiên của mô hình Ridge:

	Timestamp	Actual	Predicted
0	2019-01-01 08:00:00	4.18	4.929617
1	2019-01-01 08:15:00	2.95	3.043286
2	2019-01-01 08:30:00	4.07	4.064777
3	2019-01-01 08:45:00	5.51	4.501039
4	2019-01-01 09:00:00	2.99	3.044763
5	2019-01-01 09:15:00	3.74	4.203584
6	2019-01-01 09:30:00	2.95	2.990586
7	2019-01-01 09:45:00	4.18	4.080176
8	2019-01-01 10:00:00	4.54	4.942305

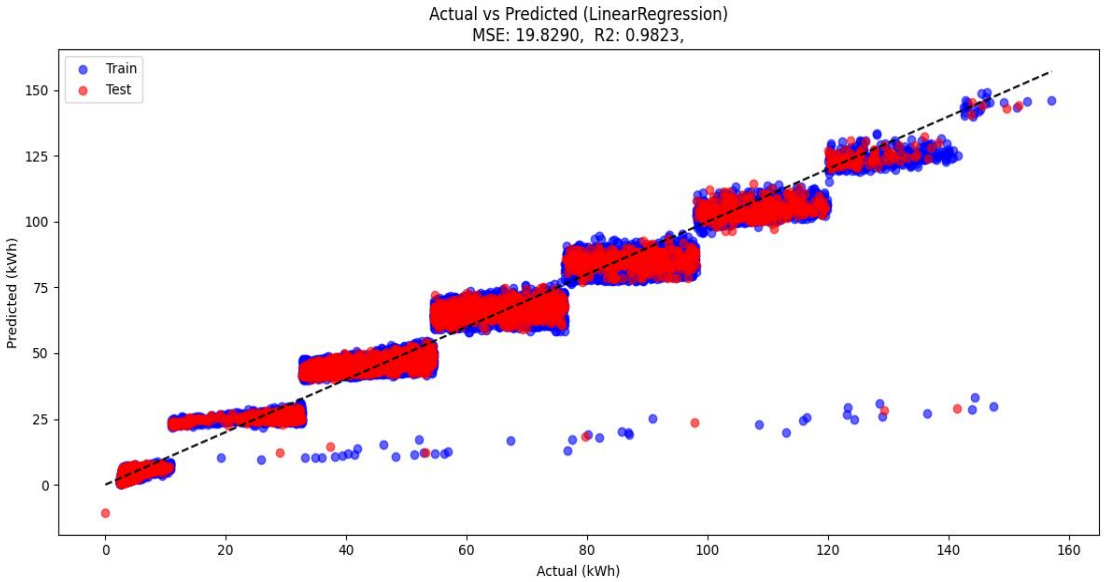
20 Dự đoán đầu tiên của mô hình Lasso:

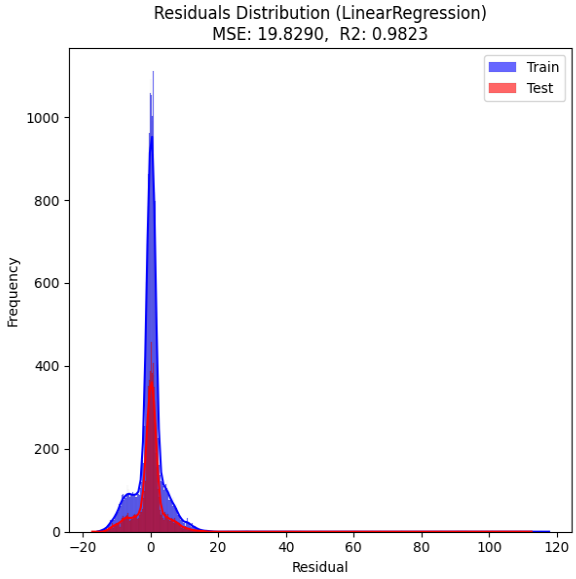
	Timestamp	Actual	Predicted
0	2019-01-01 08:00:00	4.18	4.130486
1	2019-01-01 08:15:00	2.95	3.537849
2	2019-01-01 08:30:00	4.07	4.756472
3	2019-01-01 08:45:00	5.51	6.683054
4	2019-01-01 09:00:00	2.99	3.560720
5	2019-01-01 09:15:00	3.74	4.266792
7	2019-01-01 09:45:00	4.18	5.181931
7	2019-01-01 09:45:00	4.18	5.181931
8	2019-01-01 10:00:00	4.54	5.223832

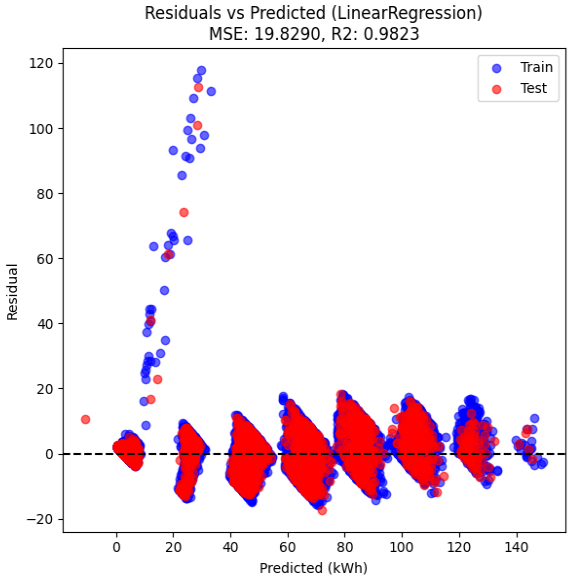
4.2. Biểu đồ đánh giá mô hình tuyến tính

Dưới đây là hình ảnh các biểu đồ dùng để đánh giá mô hình:

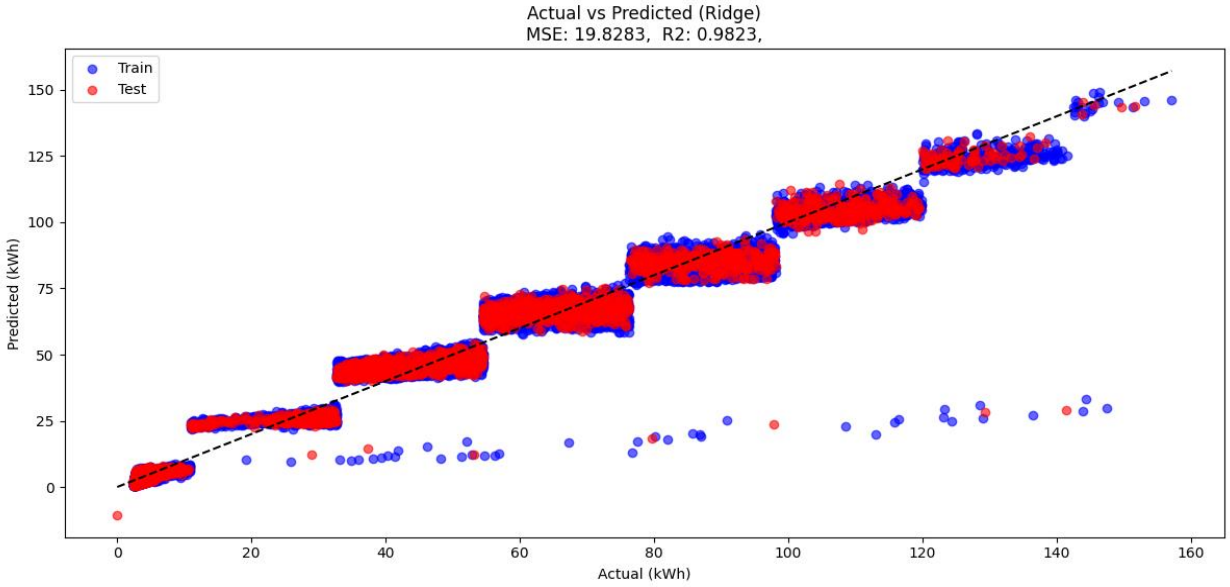
1. Biểu đồ của LinearRegression

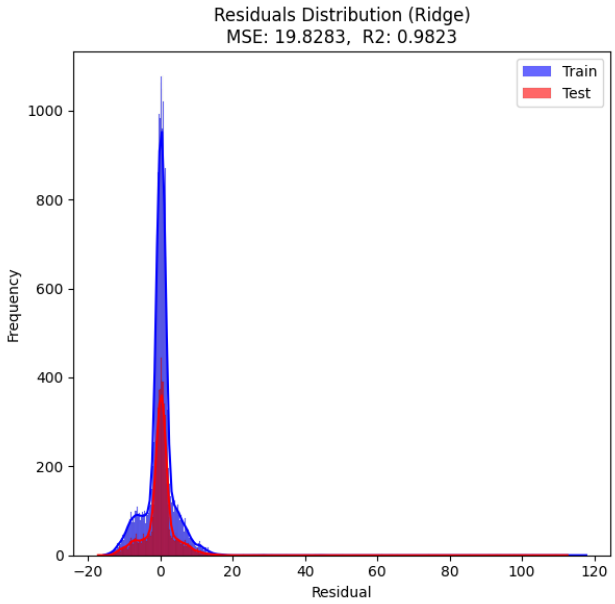


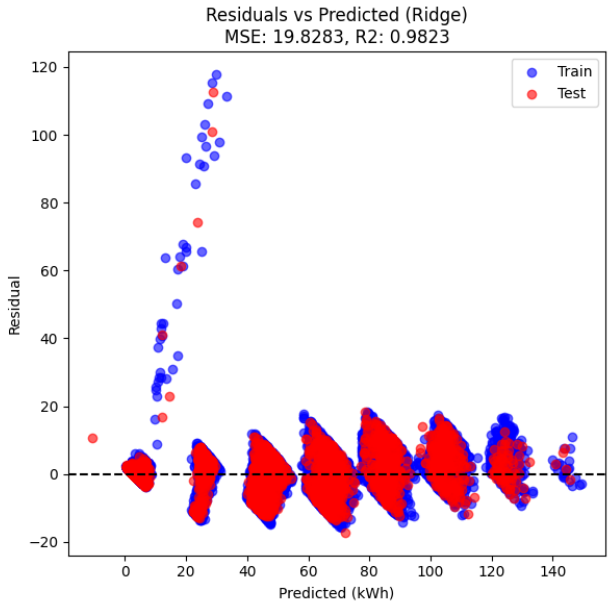




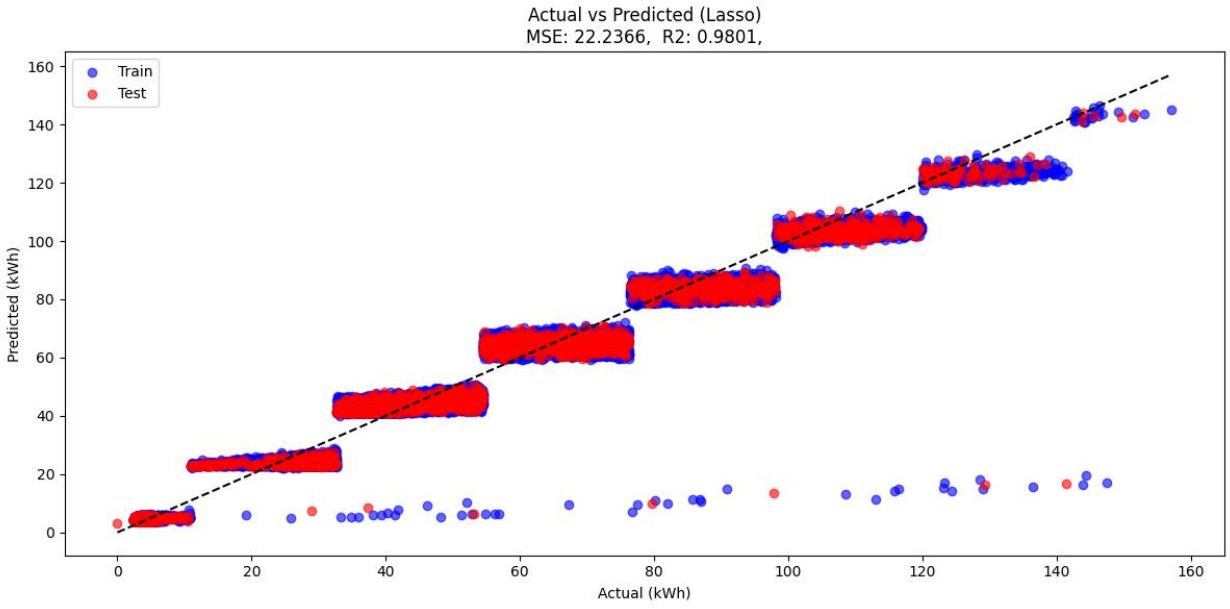
2. Biểu đồ của Ridge Regression

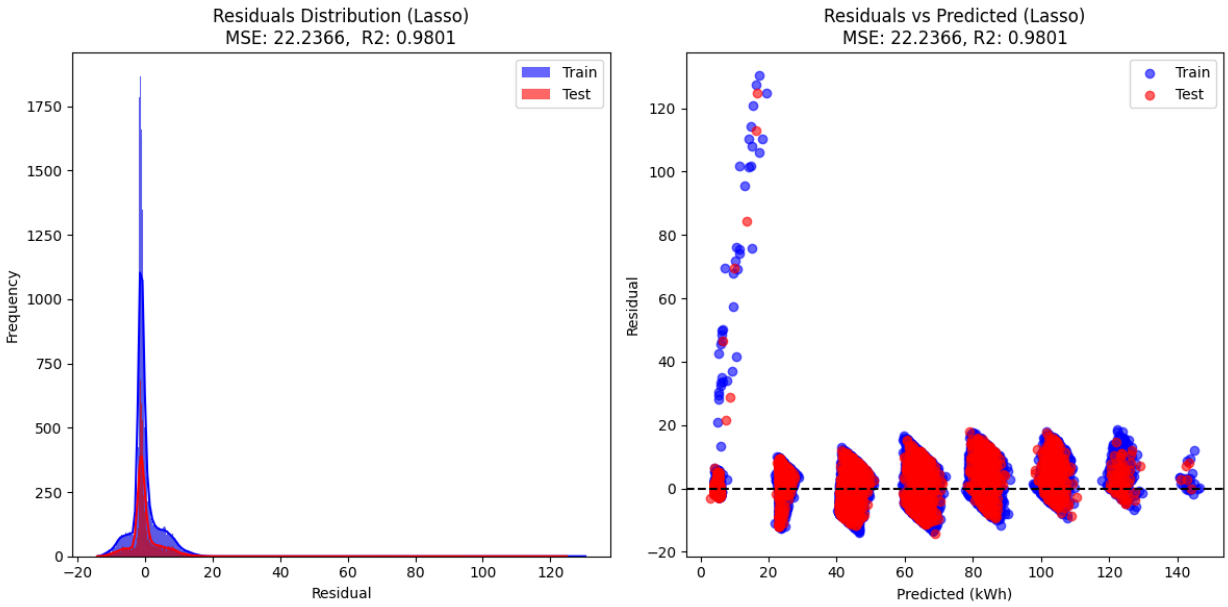






3. Biểu đồ của Lasso Regression





4. Nhận xét về các biểu đồ của các mô hình hồi quy:

Ta có thể thấy 3 mô hình Linear Regression, Ridge và Lasso đều cho ra biểu đồ có thể nói gần như là tương đồng. Vậy nên ta sẽ phân tích chi tiết cùng lúc biểu đồ của 3 mô hình:

1. Biểu đồ Actual vs Predicted

- Ưu điểm

Mối quan hệ tuyến tính tổng thể: Các điểm dữ liệu (cả tập huấn luyện và kiểm tra) có xu hướng bám khá sát đường chéo (đường y=x), cho thấy các mô hình có khả năng dự đoán giá trị khá chính xác.

Khả năng khái quát hóa tốt: Các điểm dữ liệu kiểm tra (màu đỏ) cũng nằm gần đường chéo và phân bố tương tự như dữ liệu huấn luyện (màu xanh lam), cho thấy các mô hình có khả năng khái quát hóa tốt đến dữ liệu mới

- Nhược điểm:

Các "bậc thang" rõ rệt: Thay vì một đám mây điểm liên tục, chúng ta thấy các nhóm điểm tạo thành các "bậc thang" ngang. Điều này cho thấy các mô hình đang dự đoán các giá trị rời rạc hoặc có các ngưỡng nhất định trong dữ liệu mà mô hình không nắm bắt được. Có thể có các yếu tố hoặc biến tiềm ẩn ảnh hưởng đến kết quả theo cách phân đoạn.

Một vài điểm ngoại lệ: Có một số ít điểm dữ liệu nằm khá xa đường chéo, đặc biệt là ở phần giá trị thực tế thấp. Đây có thể là những trường hợp mà mô hình dự đoán kém chính xác hơn.

2. Biểu đồ Residuals Distribution

- Ưu điểm:

Phần lớn phần dư tập trung gần 0: Đỉnh của biểu đồ mật độ tập trung gần giá trị 0, điều này là mong muốn vì nó cho thấy phần lớn các dự đoán của các mô hình không bị sai lệch quá nhiều.

- Nhược điểm:

Tính đối xứng: Phân phối của phần dư cũng không hoàn toàn đối xứng và có một số phần dư có tần suất nghiêng chút về một phía, như Linear Regression với Ridge thì có chút nhỉnh về phía bên trái, còn Lasso thì ngược lại. Điều này cho thấy mô hình có xu hướng dự đoán lớn hơn giá trị thực tế hoặc ngược lại trong một số trường hợp.

Không hoàn toàn tuân theo phân phối chuẩn: Hình dạng của phân phối của các mô hình không hoàn toàn giống hình chuông đối xứng của phân phối chuẩn. Điều này có thể vi phạm giả định về tính chuẩn của sai số trong mô hình hồi quy tuyến tính, ảnh hưởng đến tính hợp lệ của các kiểm định thống kê.

Sự khác biệt giữa dữ liệu huấn luyện và kiểm tra: Phân phối phần dư của dữ liệu kiểm tra (màu đỏ) có vẻ rộng hơn và ít tập trung hơn so với dữ liệu huấn luyện (màu xanh lam). Điều này có thể gợi ý rằng mô hình hoạt động tốt hơn trên dữ liệu huấn luyện so với dữ liệu chưa từng thấy, gợi ý về việc có khả năng mô hình có khả năng overfitting nhẹ.

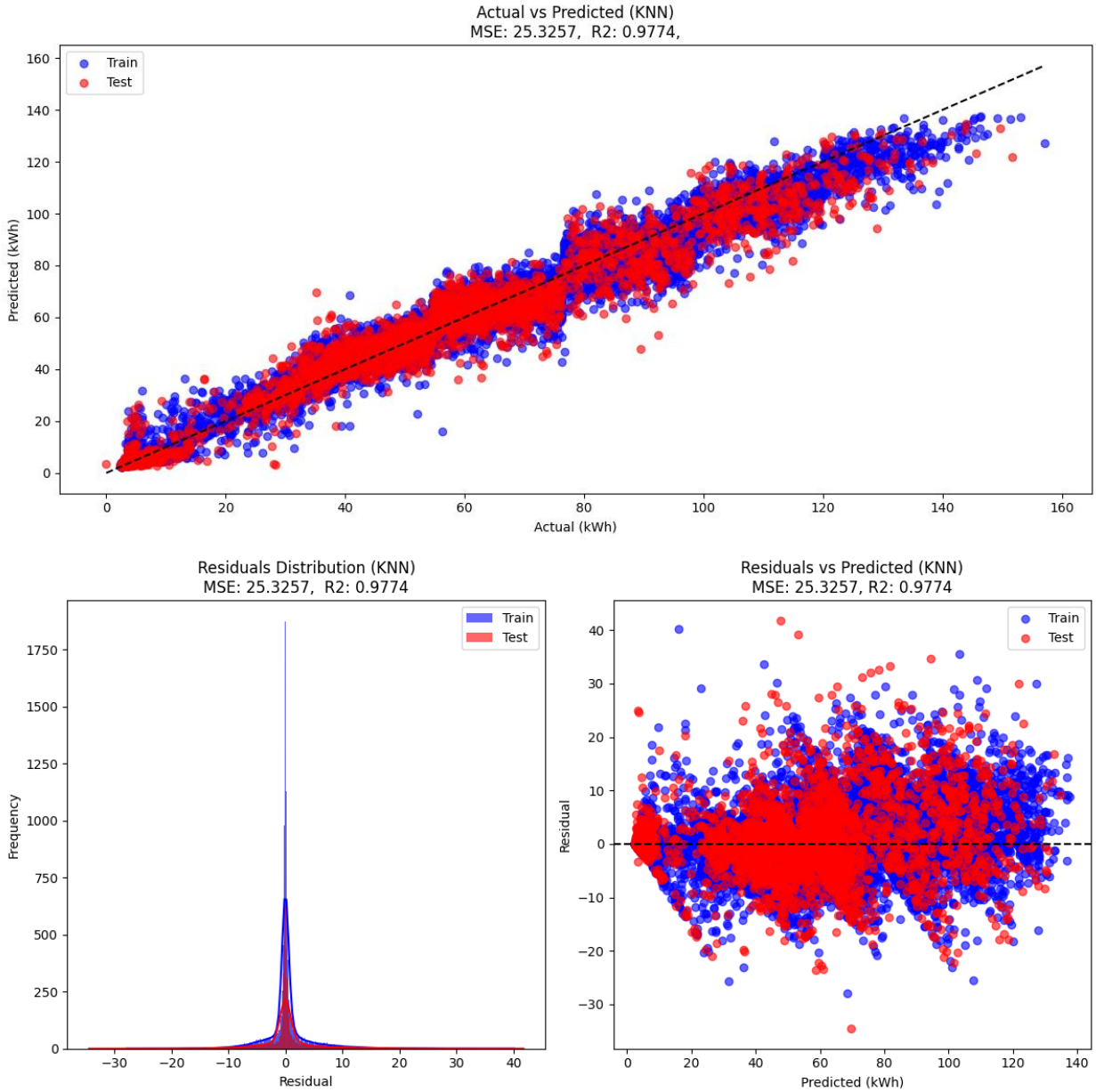
3. Biểu đồ Residuals vs Predicted:

Mô hình rõ ràng trong phần dư: Thay vì một đám mây điểm phân tán ngẫu nhiên xung quanh đường ngang y = 0, chúng ta thấy một nhóm các cụm điểm trải theo chiều dọc rõ ràng tạo thành chuỗi hình răng cưa. Điều này là một dấu hiệu mạnh mẽ cho thấy mối quan hệ giữa biến độc lập và biến phụ thuộc không hoàn toàn tuyến tính. Mô hình tuyến tính đang cố gắng "ép" một đường thẳng vào một mối quan hệ phức tạp hơn, gợi ý có thể có yếu tố phi tuyến tính trong mối quan hệ giữa đầu vào và đầu ra.

Các "bậc thang" tương ứng: Mô hình phần dư này phản ánh trực tiếp các "bậc thang" được thấy trong biểu đồ Actual vs Predicted. Khi giá trị dự đoán thay đổi qua các "bậc", phần dư cũng thay đổi theo một mô hình nhất quán.

4.3. So sánh với mô hình phi tuyến KNN

Ta thử nghiệm dự đoán trên một mô hình phi tuyến để so sánh hiệu suất giữa các mô hình hồi quy và mô hình phi tuyến. Cụ thể ở đây ta sử dụng mô hình KNN để dự đoán biến Usage_kWh.



1. Nhận xét về biểu đồ của mô hình KNN:

1. Biểu đồ Actual vs Predicted:

- Ưu điểm:

Mối quan hệ bám sát đường chéo tốt hơn: So với biểu đồ tương ứng của mô hình hồi quy tuyến tính, các điểm dữ liệu (cả tập huấn luyện và kiểm tra) trong biểu đồ này bám sát đường chéo (đường y=x) tốt hơn đáng kể. Điều này cho thấy mô hình KNN có khả năng dự đoán giá trị chính xác hơn, đặc biệt là ở các vùng giá trị mà mô hình tuyến tính gặp khó khăn (thể hiện qua các "bậc thang").

Loại bỏ các "bậc thang": Các "bậc thang" rõ rệt được thấy trong biểu đồ của mô hình hồi quy tuyến tính đã biến mất. Điều này cho thấy KNN, là một mô hình phi tham số, có thể linh hoạt hơn trong việc nắm bắt mối quan hệ không tuyến tính giữa biến đầu vào và đầu ra.

Khả năng khái quát hóa khá tốt: Các điểm dữ liệu kiểm tra (màu đỏ) phân bố tương đối đồng đều và bám sát đường chéo tương tự như dữ liệu huấn

luyện (màu xanh lam), cho thấy mô hình KNN có khả năng khái quát hóa khá tốt đến dữ liệu mới.

- Nhược điểm:

Độ phân tán lớn hơn ở một số vùng: Mặc dù bám sát đường chéo tốt hơn, nhưng có vẻ như độ phân tán của các điểm dữ liệu quanh đường chéo lớn hơn ở các vùng giá trị thực tế thấp và cao. Điều này tương ứng với quan sát về phương sai của phần dư lớn hơn ở các vùng giá trị dự đoán tương ứng trong biểu đồ "Residuals vs Predicted".

Một vài điểm ngoại lệ: Vẫn còn một số ít điểm dữ liệu nằm khá xa đường chéo, cho thấy có những trường hợp mà mô hình KNN vẫn dự đoán chưa chính xác.

Xu hướng "under-predict" ở giá trị cao và "over-predict" ở giá trị thấp: Quan sát kỹ hơn, có vẻ như ở phần cuối của biểu đồ (giá trị thực tế cao), các điểm dữ liệu có xu hướng nằm dưới đường chéo (mô hình under-predict). Ngược lại, ở phần đầu biểu đồ (giá trị thực tế thấp), các điểm có xu hướng nằm trên đường chéo (mô hình over-predict). Điều này có thể là đặc trưng của mô hình KNN khi cố gắng "trung bình hóa" các giá trị láng giềng.

2. Biểu đồ Residuals Distribution

- Ưu điểm:

Tập trung gần 0: Cả phân phối phần dư của tập huấn luyện (màu xanh lam) và tập kiểm tra (màu đỏ) đều có đỉnh tập trung rất gần giá trị 0. Điều này cho thấy phần lớn các dự đoán của mô hình không bị sai lệch quá nhiều.

Hình dạng gần với phân phối chuẩn hơn: So với biểu đồ phân phối phần dư của mô hình hồi quy tuyến tính, biểu đồ này có hình dạng gần với hình chuông đối xứng của phân phối chuẩn hơn, đặc biệt là đối với tập huấn luyện. Điều này cho thấy mô hình KNN có thể đáp ứng tốt hơn giả định về tính chuẩn của sai số so với mô hình tuyến tính.

- Nhược điểm:

Đuôi dày hơn: Cả hai phân phối đều có đuôi dày hơn so với phân phối chuẩn lý tưởng. Điều này có nghĩa là có nhiều giá trị phần dư lớn (cả dương và âm) hơn dự kiến nếu sai số tuân theo phân phối chuẩn hoàn toàn.

Sự khác biệt giữa dữ liệu huấn luyện và kiểm tra: Phân phối phần dư của tập kiểm tra (màu đỏ) có vẻ rộng hơn và "bẹt" hơn so với phân phối của tập huấn luyện (màu xanh lam). Điều này cho thấy phương sai của sai số trên tập kiểm tra lớn hơn, và mô hình có thể đang hoạt động kém ổn định hơn trên dữ liệu mới. Đây là một dấu hiệu tiềm ẩn của overfitting nhẹ.

Các giá trị ngoại lệ: Có một vài điểm dữ liệu có phần dư khá lớn (ở cả hai phía dương và âm) trong cả hai tập dữ liệu, đặc biệt là đối với tập kiểm tra.

3. Biểu đồ Residuals vs Predicted

- Ưu điểm:

Tính ngẫu nhiên được cải thiện: So với biểu đồ tương ứng của mô hình hồi quy tuyến tính, các điểm phần dư ở đây có vẻ phân tán ngẫu nhiên hơn

xung quanh đường ngang 0. Mô hình hình sin hoặc răng cưa rõ ràng đã biến mất, cho thấy mô hình KNN đã giải quyết được phần lớn vấn đề không tuyến tính.

- Nhược điểm:

Sự khác biệt giữa dữ liệu huấn luyện và kiểm tra: Các điểm phần dư của tập kiểm tra (màu đỏ) có xu hướng phân tán rộng hơn so với tập huấn luyện (màu xanh lam) trên toàn bộ phạm vi giá trị dự đoán, một lần nữa củng cố dấu hiệu về overfitting nhẹ.

Các giá trị ngoại lệ: Có một vài điểm phần dư nằm khá xa đường ngang 0, đặc biệt là đối với tập kiểm tra. Đây là những trường hợp mà mô hình KNN có sai số dự đoán lớn.

2. So sánh giữa mô hình KNN và mô hình hồi quy tuyến tính:

- Cải thiện về tính tuyến tính: Mô hình KNN đã giải quyết được vấn đề không tuyến tính rõ ràng trong phần dư mà mô hình hồi quy tuyến tính gặp phải. Các phần dư phân tán ngẫu nhiên hơn, cho thấy mô hình phù hợp hơn với mối quan hệ phức tạp trong dữ liệu.
- Phân phối phần dư gần chuẩn hơn (trên tập kiểm tra): Phân phối phần dư của tập kiểm tra có thể nói là gần tương đồng với phân phối phần dư của tập huấn luyện trong mô hình KNN, thêm vào đó phân phối phần dư lúc này cũng có hình dạng gần với phân phối chuẩn hơn so với mô hình tuyến tính.
- Vấn đề tiềm ẩn về overfitting: Tuy nhiên, sự khác biệt trong phân phối phần dư và độ phân tán của phần dư giữa tập huấn luyện và tập kiểm tra cho thấy mô hình KNN có thể đang bị overfitting nhẹ.

KẾT LUẬN

1. Đánh giá mô hình

Mô hình hồi quy tuyến tính đã thể hiện là một điểm khởi đầu hợp lý trong việc mô hình hóa mối quan hệ giữa các biến trong bài toán này, thể hiện qua giá trị R-squared tương đối cao. Tuy nhiên, các phân tích sâu hơn về phần dư cho thấy mô hình này có những hạn chế đáng kể và không phải là lựa chọn tối ưu để nắm bắt đầy đủ sự phức tạp của dữ liệu.

Mô hình có các hạn chế như sau:

- **Giả định tuyến tính bị vi phạm**: Biểu đồ "Residuals vs Predicted" cho thấy mối quan hệ phức tạp (hình sin hoặc răng cưa) giữa các biến, vượt xa khả năng mô hình tuyến tính, dẫn đến sai số dự đoán có hệ thống.
- **Phân phối phần dư không chuẩn**: Phần dư lệch dương và đuôi dài, vi phạm giả định chuẩn, làm suy yếu kiểm định thống kê và khoảng tin cậy.
- **Khả năng dự đoán hạn chế**: Biểu đồ "Actual vs Predicted" cho thấy mô hình khó dự đoán chính xác ở một số vùng giá trị, thể hiện qua sự phân tán lớn và "bậc thang".
- **Tiềm ẩn overfitting**: Sự khác biệt trong phân phối phần dư giữa tập huấn luyện và kiểm tra cho thấy mô hình chưa khái quát hóa tốt.
- **Thiếu giải thích yếu tố tiềm ẩn**: Các "bậc thang" gợi ý có yếu tố hoặc biến phân loại quan trọng mà mô hình không nắm bắt được.

2. Kết luận chung

Mô hình hồi quy tuyến tính là một điểm khởi đầu hữu ích, nhưng những hạn chế về giả định tuyến tính và phân phối phần dư không chuẩn cho thấy nó không phải là lựa chọn tốt nhất cho bài toán này. Các nỗ lực cải thiện trong tương lai nên tập trung vào việc sử dụng các mô hình phi tuyến tính và khám phá các kỹ thuật feature engineering để nắm bắt đầy đủ sự phức tạp của dữ liệu và cải thiện khả năng dự đoán.

3. Những cải thiện trong tương lai

Để cải thiện khả năng mô hình hóa và dự đoán trong bài toán này, cần xem xét các hướng đi sau:

1. Sử dụng các mô hình hồi quy phi tuyến tính: Do hạn chế lớn nhất là giả định tuyến tính bị vi phạm, việc chuyển sang các mô hình có thể nắm bắt các mối quan hệ phi tuyến là rất quan trọng. Các lựa chọn có thể bao gồm:

- Các mô hình dựa trên cây (ví dụ: Random Forest, Gradient Boosting): Các mô hình này có thể học các mối quan hệ phức tạp và tương tác giữa các biến.
- Mạng nơ-ron: Một lựa chọn mạnh mẽ cho các mối quan hệ phi tuyến phức tạp.

2. Điều chỉnh mô hình (nếu vẫn sử dụng mô hình tuyến tính): Nếu vì một lý do nào đó vẫn muốn sử dụng mô hình tuyến tính, có thể thử các kỹ thuật như thêm các biến giả để mô hình hóa các yếu tố phân loại tiềm ẩn hoặc sử dụng các phương pháp hồi quy mạnh mẽ hơn ít nhạy cảm với các vi phạm giả định. Tuy nhiên, việc này có thể không giải quyết triệt để vấn đề không tuyến tính.

TÀI LIỆU THAM KHẢO

1. UCI Machine Learning Repository.
https://archive.ics.uci.edu/dataset/851/steel+industry+energy+consumption

2. Power con sumption prediction for steel industry.
https://www.researchgate.net/publication/372416665_Power_consumption_prediction_for_steel_industry

3. Spearman's rank correlation coefficient.
https://datatab.net/tutorial/spearman-correlation

4. Bài giảng Kinh tế lượng – Trần Minh Nguyệt (2011).

5. An Introduction to Statistical Learning with Applications in R (Second Edition) – Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani (2021).

6. Introduction to Linear Regression Analysis (Fifth Edition) – Douglas C.Montgomery, Elizabeth A.Peck, G.Geofrey Vining.

7. Francis Galton and regression to the mean.
https://sites.ualberta.ca/~dwiens/stat575/misc%20resources/regression%20to%20the%20mean.pdf

PHỤ LỤC

Phụ lục 1: Giải thích phương trình (2) phần 3

Từ phương trình (1) ta có ở trên:

$$y = \beta_0 + \beta_1 x + u$$

Lấy kỳ vọng cả 2 vế, ta thu được:

$$E(y|x) = E(\beta_0 + \beta_1 x + u|x)$$

$\Rightarrow E(y|x) = E(\beta_0|x) + E(\beta_1 x|x) + E(u|x)$

Vì β_0 và $\beta_1 x$ sẽ là hằng số khi biết x:

$\Rightarrow E(y|x) = \beta_0 + \beta_1 x + E(u|x)$

$\Rightarrow E(y|x) = \beta_0 + \beta_1 x + E(u)$ (giả định)

$\Rightarrow E(y|x) = \beta_0 + \beta_1 x$

Phụ lục 2: Giải thích phương trình (6) phần 3

Ta đặt: $\sum_{i=1}^{n}[y_i - (\widehat{\beta_0} + \widehat{\beta_1} x_i)]^2 = f(\widehat{\beta_0}, \widehat{\beta_1})$

Từ đó ta được:

$\frac{\partial(f(\widehat{\beta_0},\widehat{\beta_1}))}{\partial\widehat{\beta_0}} = \sum 2\left(y_i - \widehat{\beta_0} - \widehat{\beta_1} x_i\right)(-1) = 0$ (*)

$\frac{\partial(f(\widehat{\beta_0},\widehat{\beta_1}))}{\partial\widehat{\beta_1}} = \sum 2\left(y_i - \widehat{\beta_0} - \widehat{\beta_1} x_i\right)(-x_i) = 0$ (**)

Rút gọn (*):

$\Rightarrow \sum y_i = n\widehat{\beta_0} + \widehat{\beta_1}\sum x_i$

$\Rightarrow \frac{\sum y_i}{n} = \widehat{\beta_0} + \widehat{\beta_1}\frac{\sum x_i}{n}$

$\Rightarrow \widehat{\beta_0} = \bar{y} - \widehat{\beta_1}\bar{x}$ (***)

Rút gọn (**):

$\Rightarrow \sum x_i y_i = \widehat{\beta_0}\sum x_i + \widehat{\beta_1}\sum {x_i}^2$

Thay vào (***):

$\Rightarrow \sum x_i y_i = (\bar{y} - \widehat{\beta_1}\bar{x}) \sum x_i + \widehat{\beta_1} \sum {x_i}^2$

$\Rightarrow \sum x_i y_i = \bar{y} \sum x_i - \widehat{\beta_1}\bar{x} \sum x_i + \widehat{\beta_1} \sum {x_i}^2$

$\Rightarrow \sum x_i y_i - \bar{y} \sum x_i = \widehat{\beta_1}(\sum {x_i}^2 - \bar{x} \sum x_i)$

$\Rightarrow \widehat{\beta_1} = \frac{\sum x_i y_i - \bar{y} \sum x_i}{\sum {x_i}^2 - \bar{x} \sum x_i} = \frac{\sum x_i y_i - (\sum x_i \sum y_i)/n}{\sum {x_i}^2 - (\sum x_i)^2/n} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$

Phụ lục 3: Giải thích phương trình (9) phần 3

Ở trước đấy, ta đã viết được hàm mục tiêu dưới dạng ma trận như sau:

$$||Y - X\hat{\beta}||^2$$

Khi này: ta thu được:

$||Y - X\hat{\beta}||^2 = ||X\hat{\beta} - Y||^2 = \left(X\hat{\beta} - Y\right)^T (X\hat{\beta} - Y)$

$= Y^T Y - Y^T X\hat{\beta} - \hat{\beta}^T X^T Y + \hat{\beta}^T X^T X\hat{\beta}$

Bởi vì hàm trên là hàm lồi, nên nghiệm tối ưu sẽ là:

$\frac{\partial}{\partial \hat{\beta}}(Y^T Y - Y^T X\hat{\beta} - \hat{\beta}^T X^T Y + \hat{\beta}^T X^T X\hat{\beta}) = -2X^T Y + 2X^T X\hat{\beta} = 0$

$\Rightarrow X^T X\hat{\beta} = X^T Y$

$\Rightarrow \hat{\beta} = (X^T X)^{-1} X^T Y$

Phụ lục 4: Giải thích phương trình (10) phần 3

Ta có: $TSS = \sum(y_i - \bar{y})^2 = \sum(y_i - \hat{y_i} + \hat{y_i} - \bar{y})^2$

$= \sum(y_i - \hat{y_i})^2 + 2\sum(y_i - \hat{y_i})(\hat{y_i} - \bar{y}) + \sum(\hat{y_i} - \bar{y})^2$

$= RSS + ESS + 2\sum(y_i - \hat{y_i})(\hat{y_i} - \bar{y})$

Xét: $\sum(y_i - \hat{y_i})(\hat{y_i} - \bar{y}) = \sum(y_i - \widehat{\beta_0} - \widehat{\beta_1}x_i)(\widehat{\beta_0} + \widehat{\beta_1}x_i - \bar{y})$

$= \sum(y_i - \bar{y} + \widehat{\beta_1}\bar{x} - \widehat{\beta_1}x_i)(\bar{y} - \widehat{\beta_1}\bar{x} + \widehat{\beta_1}x_i - \bar{y})$

$= \sum((y_i - \bar{y}) - \widehat{\beta_1}(x_i - \bar{x}))\widehat{\beta_1}(x_i - \bar{x})$

$$= \sum(\widehat{\beta_1}(x_i - \bar{x})(y_i - \bar{y}) - \widehat{\beta_1}^2(x_i - \bar{x})^2)$$

$$= \widehat{\beta_1}\sum(x_i - \bar{x})(y_i - \bar{y}) - \widehat{\beta_1}^2\sum(x_i - \bar{x})^2$$

Mà:

$$\widehat{\beta_1} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Thay vào phương trình trên:

⇨ $\sum(y_i - \hat{y}_i)(\hat{y}_i - \bar{y}) = 0$

⇨ $TSS = RSS + ESS$

Phụ lục 5: Tiền xử lý dữ liệu

```
def load_and_preprocess_data(file_path):

    data = pd.read_csv(file_path)

    data['date'] = pd.to_datetime(data['date'], format='%d/%m/%Y %H:%M')

    data['year'] = data['date'].dt.year
    data['month'] = data['date'].dt.month
    data['day'] = data['date'].dt.day
    data['hour'] = data['date'].dt.hour
    data['minute'] = data['date'].dt.minute

    data = data.drop('date', axis=1).dropna()

    data = pd.get_dummies(data, columns=['WeekStatus', 'Day_of_week', 'Load_Type'],
    dtype=int)

    return data
```

❖ **Thư viện sử dụng:**

```
import pandas as pd
```

Pandas là công cụ hữu dụng được chuyên dùng để phục vụ các công việc tiền xử lý dữ liệu thông thường như loại bỏ các giá trị null, tách các biến ra thành các cột riêng biệt, tạo biến giả,….

❖ **Giải thích:**

```
data = pd.read_csv(file_path)
```

* *Chức năng*: Tải dữ liệu từ file CSV được chỉ định bởi đường dẫn file_path vào một DataFrame của pandas và đặt tên nó là data.

* *Giải thích qua về DataFrame:*

DataFrame là một cấu trúc dữ liệu hai chiều được cung cấp bởi thư viện pandas trong Python. Ta có thể hình dung DataFrame như một bảng dữ liệu mặc định trong Excel với:

- Hàng (rows) và cột (columns) (như trong Excel).
- Dữ liệu trong DataFrame có thể chứa các loại dữ liệu khác nhau (số nguyên, số thực, chuỗi, ngày tháng, v.v.) trong các cột khác nhau.

```
data['date'] = pd.to_datetime(data['date'], format='%d/%m/%Y %H:%M')
```

* *Chức năng:* Chuyển đổi kiểu dữ liệu của cột date trong DataFrame 'data' từ kiểu chuỗi (string) sang kiểu datetime có định dạng %d/%m/%y %H:%M (ví dụ 31/12/2024 10:59) để có thể dễ dàng thao tác với các giá trị thời gian.

Lưu ý: Do cột date đã có format %d/%m/%y %H:%M từ trước nên ta chỉ cần hiểu đơn giản là chuyển đổi kiểu dữ liệu của cột date từ string sang datetime với format có sẵn

```
data['year'] = data['date'].dt.year
data['month'] = data['date'].dt.month
data['day'] = data['date'].dt.day
data['hour'] = data['date'].dt.hour
data['minute'] = data['date'].dt.minute
```

* *Chức năng:* Trích xuất các dữ liệu từ cột date (datetime) và tạo ra các cột mới tương ứng gắn với các dữ liệu đó. Nói cách khác, các cột mới như year, month, day, hour, minute được thêm vào DataFrame và chứa các giá trị tương ứng.

* *Cách hoạt động:*

Ta có dữ liệu gốc như sau:

date
1/1/2018 0:15
1/1/2018 0:30
1/1/2018 0:45
1/1/2018 1:00
1/1/2018 1:15
1/1/2018 1:30
1/1/2018 1:45
1/1/2018 2:00
1/1/2018 2:15
1/1/2018 2:30
1/1/2018 2:45
1/1/2018 3:00
1/1/2018 3:15
1/1/2018 3:30
1/1/2018 3:45
1/1/2018 4:00

Sau khi trích xuất dữ liệu ra các cột mới tương ứng, ta được bảng dữ liệu mới:

year	month	day	hour	minute
2018	1	1	0	15
2018	1	1	0	30
2018	1	1	0	45
2018	1	1	1	0
2018	1	1	1	15
2018	1	1	1	30
2018	1	1	1	45
2018	1	1	2	0
2018	1	1	2	15
2018	1	1	2	30
2018	1	1	2	45
2018	1	1	3	0
2018	1	1	3	15
2018	1	1	3	30
2018	1	1	3	45
2018	1	1	4	0

```
data = data.drop('date', axis=1).dropna()
```

* *Chức năng:* Loại bỏ cột date vì sau khi trích xuất các thông tin cần thiết, cột này bị thừa và không còn hữu dụng, cùng lúc đó loại bỏ các dòng có giá trị bị thiếu (null hoặc NaN).

Trong file dữ liệu trên, do không có chứa bất kỳ dữ liệu nào bị null nên dòng code này có thể bỏ qua. Nhưng đây vẫn là một bước làm quan trọng không thể bỏ qua trong việc tiền xử lý dữ liệu nói chung nhằm đảm bảo tuân thủ đầy đủ

các quy tắc khi làm việc với dữ liệu, tránh những sai sót trong quá trình huấn luyện dữ liệu, gây ảnh hưởng xấu đến kết quả dự đoán đầu ra.

```
data = pd.get_dummies(data, columns=['WeekStatus', 'Day_of_week', 'Load_Type'],
dtype=int)
```

* *Chức năng:* Chuyển đổi các cột có kiểu dữ liệu string bao gồm WeekStatus, Day_of_week, và Load_Type thành các cột nhị phân (cột chỉ mang hai giá trị 0 hoặc 1).

* *Tại sao cần thực hiện?*

- Các mô hình như Linear Regression, Ridge, hoặc KNN yêu cầu tất cả dữ liệu đầu vào là số. Nếu ta để nguyên dữ liệu dạng chuỗi (như "Weekday", "Weekend"), mô hình sẽ không xử lý được.
- Tạo thông tin rõ ràng cho mô hình, tránh nhầm lẫn do thứ tự: Thay vì chuyển đổi trực tiếp các giá trị của biến độc lập thành số nguyên (vd. “Monday” → 0, “Tuesday” → 1, “Wednesday” → 2,…), phương pháp này đảm bảo không áp đặt bất kỳ thứ tự nào cho các biến độc lập, đảm bảo cân bằng vai trò của từng biến.

⇨ Giúp mô hình học được mối quan hệ giữa từng biến độc lập và biến đầu ra (Usage_kWh) mà không làm mất đi thông tin.

* *Cách hoạt động:* Mỗi giá trị duy nhất trong một cột sẽ được chuyển thành một cột riêng, với giá trị là 0 hoặc 1, đại diện cho sự có mặt hoặc vắng mặt của giá trị đó.

Ví dụ: Cột "Load_Type" chứa ba giá trị "Light_Load", "Medium_Load" và “Maximum Load”. Dưới đây là dữ liệu cụ thể:

132	Tuesday	Light_Load
133	Tuesday	Light_Load
134	Tuesday	Medium_Load
135	Tuesday	Medium_Load
136	Tuesday	Medium_Load
137	Tuesday	Medium_Load
138	Tuesday	Maximum_Load
139	Tuesday	Maximum_Load

pd.get_dummies sẽ chuyển chúng thành ba cột lần lượt là: Load_Type_Light_Load,Load_Type_Maximum_LoadvàLoad_Type_Medium_Load. Sau đó gắn các giá trị 0(FALSE) và 1(TRUE) tương ứng

1	0	0
1	0	0
0	0	1
0	0	1
0	0	1
0	0	1
0	1	0
0	1	0

```
return data
```

* *Chức năng:* Trả về DataFrame 'data' sau khi đã thực hiện toàn bộ các bước tiền xử lý dữ liệu, sẵn sàng cho bước xử lý dữ liệu tiếp theo.

Phụ lục 6: Phân tách và chuẩn hóa

```
def split_and_scale_data(data):


    X = data.drop(['Usage_kWh'], axis=1)
    y = data['Usage_kWh']



    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=43)


    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)


    return X_train_scaled, X_test_scaled, y_train, y_test
```

❖ **Thư viện sử dụng:**

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

❖ **Giải thích:**

```
X = data.drop(['Usage_kWh'], axis=1)
```

* *Chức năng:* Tạo một biến X lưu trữ DataFrame 'data' đã loại bỏ cột 'Usage_kWh để lấy tập hợp các biến độc lập hay các đặc trưng (features).

Kết quả là biến X chứa toàn bộ các đặc trưng đầu vào dùng để dự đoán (trừ Usage_kWh, vì đây là biến mục tiêu).

```
y = data['Usage_kWh']
```

* *Chức năng:* Tạo biến Y chứa mỗi data cột Usage_kWh làm biến mục tiêu (target variable).

Biến này là một mảng (hoặc Series) chứa giá trị thực tế cần dự đoán.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=43)
```

* *Chức năng:*
Chia dữ liệu thành hai tập:

- Tập huấn luyện (dùng để huấn luyện mô hình).
- Tập kiểm tra (dùng để đánh giá mô hình).

* *Cách hoạt động:*
train_test_split nhận vào:

X: tập hợp đặc trưng đầu vào.
y: biến mục tiêu.

test_size=0.2: 20% dữ liệu sẽ được dùng làm tập kiểm tra, 80% còn lại dùng làm tập huấn luyện

Lý do:
Tập huấn luyện (80%):

- Cần càng nhiều dữ liệu càng tốt để mô hình học được các mối quan hệ giữa biến độc lập và biến mục tiêu.
- Nếu dùng quá ít dữ liệu huấn luyện, mô hình sẽ không có đủ thông tin để học, dẫn đến kết quả dự đoán trên tập huấn luyện lẫn tập kiểm tra đều không chính xác (underfitting).
- Ngược lại, nếu nhiều quá thì mô hình trong quá trình huấn luyện dần về sau sẽ bắt đầu có xu hướng ghi nhớ, học thuộc dữ liệu thay vì học hỏi, dẫn đến kết quả trên tập huấn luyện rất chuẩn xác, gần khớp với giá trị thực tế, nhưng kết quả trên tập kiểm tra thì bị lệch đi rất nhiều (overfitting).

Tập kiểm tra (20%):

- Cần đủ dữ liệu để có thể đánh giá độ chính xác của mô hình một cách đáng tin cậy.

- Nếu dùng quá ít dữ liệu kiểm tra, kết quả đánh giá sẽ chưa đủ thuyết phục để làm cơ sở đánh giá hiệu năng mô hình. Ngược lại, nếu quá nhiều thì như đã đề cập ở trên, mô hình không có đủ dữ liệu để huấn luyện.

So với hơn 35 nghìn dòng (record) trong file dữ liệu mà chúng tôi có, thì tỷ lệ 80:20 là hợp lý.

- random_state=43: Số 43 thực chất chỉ là số ngẫu nhiên, ta có thể bất kì số nguyên nào cũng được. Mục đích cuối cùng của tham số chỉ là đảm bảo kết quả chia dữ liệu giữa hai tập huấn luyện và kiểm tra là như nhau sau mỗi lần chạy code, đảm bảo kết quả đầu ra không bị thay đổi do xáo trộn dữ liệu.

Kết quả:
X_train và y_train: tập đầu vào và đầu ra của dữ liệu huấn luyện.
X_test và y_test: tập đầu vào và đầu ra của dữ liệu kiểm tra.

```
scaler = StandardScaler()
```

* *Chức năng:* Tạo đối tượng StandardScaler để chuẩn hóa dữ liệu.

* *Cách hoạt động:*
StandardScaler chuẩn hóa dữ liệu bằng cách áp dụng công thức:

$$X_{scaled} = \frac{X - \mu}{\sigma}$$

Trong đó:
X: giá trị ban đầu của dữ liệu
μ: trung bình mẫu của cột tương ứng
σ: độ lệch chuẩn của cột tương ứng

Kết quả là các cột dữ liệu của biến độc lập có phân phối chuẩn với trung bình 0 và phương sai 1.

Lý do:

Một số cột có giá trị lớn (NSM với giá trị từ nghìn lên đến vài chục nghìn) trong khi một số cột có giá trị nhỏ (Lagging_Current_Reactive.Power_kVarh,…). Mô hình có thể ưu tiên các cột có giá trị lớn hơn nếu không chuẩn hóa.

Chuẩn hóa giúp các thuật toán như hồi quy tuyến tính hoặc KNN hoạt động hiệu quả hơn vì dữ liệu đã được đưa về cùng một thang đo.

```
X_train_scaled = scaler.fit_transform(X_train)
```

* *Chức năng:* Chuẩn hóa tập dữ liệu huấn luyện.

* *Cách hoạt động:*

- Tính trung bình và độ lệch chuẩn của X_train.

– Áp dụng chuẩn hóa trên các dữ liệu trong X_train.

Kết quả là X_train_scaled – tập đầu vào huấn luyện đã được chuẩn hóa.

```
X_test_scaled = scaler.transform(X_test)
```

* *Chức năng:* Áp dụng phép chuẩn hóa lên tập kiểm tra (X_test).

* *Cách hoạt động:*

– Sử dụng trung bình và độ lệch chuẩn đã tính từ X_train (không tính lại).

– Chuẩn hóa các dữ liệu của X_test

Kết quả là X_test_scaled - tập đầu vào kiểm tra đã được chuẩn hóa

```
return X_train_scaled, X_test_scaled, y_train, y_test
```

* *Chức năng:* Trả về các tập dữ liệu đã được chuẩn hóa, chuẩn bị cho bước huấn luyện mô hình và dự đoán dữ liệu.

Phụ lục 7: Huấn luyện mô hình và dự đoán

```
def train_and_evaluate_models(X_train, X_test, y_train, y_test):
    models = {
        "LinearRegression": LinearRegression(),
        "Ridge": Ridge(),
        "Lasso": Lasso(),
        "KNN": KNeighborsRegressor()
    }

    results = {}

    for name, model in models.items():

        start_time = time.time()
        model.fit(X_train, y_train)
        end_time = time.time()
        elapsed_time = end_time - start_time

        y_pred = model.predict(X_test)

        mse = mean_squared_error(y_test, y_pred)
        r2 = r2_score(y_test, y_pred)

        results[name] = {
            "MSE": mse,
            "R2": r2,
            "Time": elapsed_time
        }

        print(f"{name}:")
        print(f"  MSE = {mse:.4f}")
        print(f"  R2 = {r2:.4f}")
        print(f"  Time = {elapsed_time:.4f} seconds\n")

    return models, results
```

❖ **Thư viện sử dụng:**

```
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

❖ **Giải thích:**

```
models = {
    "LinearRegression": LinearRegression(),
    "Ridge": Ridge(),
    "Lasso": Lasso(),
    "KNN": KNeighborsRegressor()
}

results = {}
```

* *Chức năng:*

- Tạo một cấu trúc dữ liệu(cụ thể ở đây là dictionary) để chứa các mô hình cần sử dụng để huấn luyện và đánh giá bao gồm:
  - LinearRegression: Hồi quy tuyến tính thông thường.
  - Ridge: Hồi quy tuyến tính với điều chuẩn Ridge (L2 Regularization).
  - Lasso: Hồi quy tuyến tính với điều chuẩn Lasso (L1 Regularization).
  - KNeighborsRegressor: Hồi quy dựa trên thuật toán K-Nearest Neighbors (KNN).
- Khởi tạo thêm một dictionary 'result' để lưu trữ các kết quả đánh giá (MSE, R^2, thời gian chạy) cho từng mô hình mà ta sẽ thêm vào sau.

```
start_time = time.time()
model.fit(X_train, y_train)
end_time = time.time()
elapsed_time = end_time - start_time
```

* *Chức năng:* Tính tổng thời gian huấn luyện của từng mô hình

* *Cách hoạt động:*

- start_time: Lưu thời điểm bắt đầu huấn luyện mô hình.
- model.fit(X_train, y_train): Huấn luyện mô hình với tập dữ liệu huấn luyện (X_train, y_train).
- end_time: Lưu thời điểm hoàn thành huấn luyện mô hình.
- elapsed_time: Tính tổng thời gian huấn luyện mô hình bằng cách lấy hiệu giữa thời gian kết thúc và bắt đầu.

```
y_pred = model.predict(X_test)
```

* *Chức năng:* Dự đoán giá trị mục tiêu (y_pred) trên tập dữ liệu kiểm tra (X_test) sau khi mô hình đã được huấn luyện.

Ta sẽ đối chiếu y_pred này với y_test(y thực tế) để đánh giá hiệu suất, khả năng dự đoán của mô hình

```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

* *Chức năng:* Tính toán các hệ số đánh giá hiệu suất mô hình (MSE, R2) dựa trên y_pred và y_test

```
print(f"{name}:")
print(f"  MSE = {mse:.4f}")
print(f"  R2 = {r2:.4f}")
print(f"  Time = {elapsed_time:.4f} seconds\n")
```

* *Chức năng:* Lưu kết quả đánh giá của các mô hình vào dictionary 'results' kèm theo các thông số cần thiết

```
return models, results
```

* *Chức năng:* Trả về hai dictionary:
  - models: Tất cả các mô hình đã được huấn luyện.
  - results: Kết quả đánh giá của từng mô hình.

Phụ lục 8: Vẽ biểu đồ minh họa dự đoán

```
def plot_actual_vs_predicted(y_train, y_test, y_pred_train, y_pred_test, model_name, mse, r2):
    plt.figure(figsize=(12, 6))

    plt.scatter(y_train, y_pred_train, alpha=0.6, color='blue', label='Train')

    plt.scatter(y_test, y_pred_test, alpha=0.6, color='red', label='Test')

    plt.plot([min(min(y_train), min(y_test)), max(max(y_train), max(y_test))],
             [min(min(y_train), min(y_test)), max(max(y_train), max(y_test))],
             color='black', linestyle='--')

    plt.title(f'Actual vs Predicted ({model_name})\nMSE: {mse:.4f},  R2: {r2:.4f},')
    plt.xlabel('Actual (kWh)')
    plt.ylabel('Predicted (kWh)')
    plt.legend()
    plt.tight_layout()
    plt.show()
```

```
def plot_residuals(y_train, y_test, y_pred_train, y_pred_test, model_name, mse, r2):
    residuals_train = y_train - y_pred_train
    residuals_test = y_test - y_pred_test

    plt.figure(figsize=(12, 6))

    plt.subplot(1, 2, 1)
    sns.histplot(residuals_train, kde=True, color='blue', label='Train', alpha=0.6)
    sns.histplot(residuals_test, kde=True, color='red', label='Test', alpha=0.6)
    plt.title(f'Residuals Distribution ({model_name})\nMSE: {mse:.4f},  R2: {r2:.4f}')
    plt.xlabel('Residual')
    plt.ylabel('Frequency')
    plt.legend()

    plt.subplot(1, 2, 2)
    plt.scatter(y_pred_train, residuals_train, alpha=0.6, color='blue', label='Train')
    plt.scatter(y_pred_test, residuals_test, alpha=0.6, color='red', label='Test')
    plt.axhline(y=0, color='black', linestyle='--')
    plt.title(f'Residuals vs Predicted ({model_name})\nMSE: {mse:.4f}, R2: {r2:.4f}')
    plt.xlabel('Predicted (kWh)')
    plt.ylabel('Residual')
    plt.legend()

    plt.tight_layout()
    plt.show()
```

❖ **Thư viện sử dụng:**

```
import matplotlib.pyplot as plt
import seaborn as sns
import time
```

❖ **Giải thích:**

- Hàm plot actual vs_predicted

Hàm này vẽ biểu đồ so sánh giá trị thực tế (Actual) và giá trị dự đoán (Predicted) của một mô hình. Nó hiển thị dữ liệu từ cả tập huấn luyện và tập kiểm tra, đồng thời lấy đường chéo y=x làm tham chiếu để đánh giá độ chính xác của mô hình.

```
plt.figure(figsize=(12, 6))
```

* *Chức năng:* Tạo một khung mới để vẽ biểu đồ với kích thước: chiều rộng là 12 đơn vị và chiều cao là 6 đơn vị.

```
plt.scatter(y_train, y_pred_train, alpha=0.6, color='blue', label='Train')
plt.scatter(y_test, y_pred_test, alpha=0.6, color='red', label='Test')
```

* *Chức năng:*

Vẽ các điểm dữ liệu cho tập huấn luyện:

- y_train: Giá trị thực tế từ tập huấn luyện.
- y_pred_train: Giá trị dự đoán từ mô hình cho tập huấn luyện.
- alpha=0.6: Đặt độ trong suốt của các điểm (giúp dễ nhìn hơn khi dữ liệu dày đặc).
- color='blue': Màu xanh cho các điểm dữ liệu của tập huấn luyện.
- label='Train': Gán nhãn "Train" cho tập dữ liệu huấn luyện trong chú giải.

Tương tự như vậy với các điểm dữ liệu cho tập kiểm tra, khác ở chỗ các điểm này có màu đỏ và gán nhãn "Test".

```
plt.plot([min(min(y_train), min(y_test)), max(max(y_train), max(y_test))],
         [min(min(y_train), min(y_test)), max(max(y_train), max(y_test))],
         color='black', linestyle='--')
```

* *Chức năng:* Vẽ đường chéo tham chiếu trên biểu đồ.

- Điểm đầu và điểm cuối của đường tham chiếu có tọa độ bằng nhau (x,x), tạo thành một đường thẳng với góc 45 độ y=x (đường lý tưởng khi giá trị thực tế bằng giá trị dự đoán).
- color='black': Màu đen cho đường tham chiếu.
- linestyle='--': Đường nét đứt để phân biệt với các điểm dữ liệu.

```
plt.title(f'Actual vs Predicted ({model_name})\nMSE: {mse:.4f},  R2: {r2:.4f},')
```

* *Chức năng:* Thêm tiêu đề cho từng biểu đồ kèm theo các thông số hiệu suất để người xem có phân biệt.

```
plt.xlabel('Actual (kWh)')
plt.ylabel('Predicted (kWh)')
```

* *Chức năng:* Gắn nhãn cho trục x và y

Trục x đại diện cho giá trị thực tế (đơn vị kWh).

Trục y đại diện cho giá trị dự đoán (đơn vị kWh).

```
plt.legend()
```

* *Chức năng:* Hiển thị chú thích cho biểu đồ, phân biệt dữ liệu của tập huấn luyện và tập kiểm tra dựa trên màu sắc và nhãn (label) ta đã định nghĩa ở trên.

```
plt.tight_layout()
```

* *Chức năng:* Tự động điều chỉnh bố cục của biểu đồ để tránh bị cắt chữ hoặc tràn ra ngoài khung vẽ.

```
plt.show()
```

* *Chức năng:* Hiển thị biểu đồ đã được vẽ lên màn hình

- Hàm plot_residuals

Hàm plot_residuals được dùng để vẽ 2 biểu đồ nhằm trực quan hóa phần dư (residuals), cụ thể đó là:

– Phân phối phần dư (Residuals Distribution): Kiểm tra tính đối xứng và thiên lệch của phần dư. Qua đó, kiểm tra giả định về phân phối chuẩn của sai số (errors) trong mô hình hồi quy
– Phần dư so với giá trị dự đoán (Residuals vs Predicted): Kiểm tra sự phân tán và mẫu hình của phần dư. Qua đó, kiểm tra xem mối quan hệ giữa các biến độc lập và biến phụ thuộc có thực sự là tuyến tính hay không.

```
residuals_train = y_train - y_pred_train
residuals_test = y_test - y_pred_test
```

* *Chức năng:* Tính phần dư cho cả tập huấn luyện (residuals_train) và tập kiểm tra (residuals_test) với công thức chung:

$$\text{Residual} = \text{Actual} - \text{Predicted}.$$

Trong đó:
+ y_train: Giá trị thực tế của tập huấn luyện.
+ y_pred_train: Giá trị dự đoán từ mô hình cho tập huấn luyện.
+ y_test: Giá trị thực tế của tập kiểm tra.
+ y_pred_test: Giá trị dự đoán từ mô hình cho tập kiểm tra.

```
plt.subplot(1, 2, 1)
```

* *Chức năng:* Chia khung vẽ thành 1 hàng, 2 cột và chọn vị trí cột đầu tiên (bên trái) để vẽ biểu đồ phân phối phần dư.

```
sns.histplot(residuals_train, kde=True, color='blue', label='Train', alpha=0.6)
sns.histplot(residuals_test, kde=True, color='red', label='Test', alpha=0.6)
```

* *Chức năng:* Vẽ biểu đồ phân phối (histogram) của phần dư cho tập huấn luyện và kiểm tra. Trong đó:

– sns.histplot: Hàm của thư viện Seaborn dùng để vẽ histogram kèm theo đường mật độ kernel (KDE).

- residuals_train và residuals_test: Dữ liệu phần dư cần vẽ.
- kde=True: Hiển thị đường mật độ kernel ước lượng (một dạng mượt của histogram).
- color: Màu sắc cho mỗi tập dữ liệu (blue cho tập huấn luyện và red cho tập kiểm tra).
- label: Gán nhãn để hiển thị trong chú thích.
- alpha=0.6: Đặt độ trong suốt cho biểu đồ

```
plt.subplot(1, 2, 2)
```

* *Chức năng:* Chọn vị trí còn lại trong khung vẽ (cột thứ hai) để vẽ biểu đồ phần dư so với giá trị dự đoán.

```
plt.scatter(y_pred_train, residuals_train, alpha=0.6, color='blue', label='Train')
plt.scatter(y_pred_test, residuals_test, alpha=0.6, color='red', label='Test')
```

* *Chức năng:* Vẽ biểu đồ phân tán (scatter plot) của phần dư so với giá trị dự đoán.

- y_pred_train và y_pred_test: Giá trị dự đoán từ mô hình.
- residuals_train và residuals_test: Phần dư tương ứng.
- alpha=0.6: Đặt độ trong suốt để tránh biểu đồ quá dày đặc.
- color: Màu xanh (blue) cho tập huấn luyện và đỏ (red) cho tập kiểm tra.
- label: Gán nhãn để hiển thị trong chú thích.

```
plt.axhline(y=0, color='black', linestyle='--')
```

* *Chức năng:* Thêm một đường ngang tại y=0 (phần dư bằng 0) giúp dễ dàng so sánh các phần dư dương và âm.