**Kỹ thuật tính năng cho chuỗi thời gian: Tạo công thức dữ liệu hoàn hảo**

**1. Tính năng Lag: Khai thác sức mạnh của quá khứ**

Một trong những kỹ thuật đơn giản và mạnh mẽ nhất trong kỹ thuật xây dựng đặc điểm chuỗi thời gian là tạo ra **các đặc điểm trễ** .

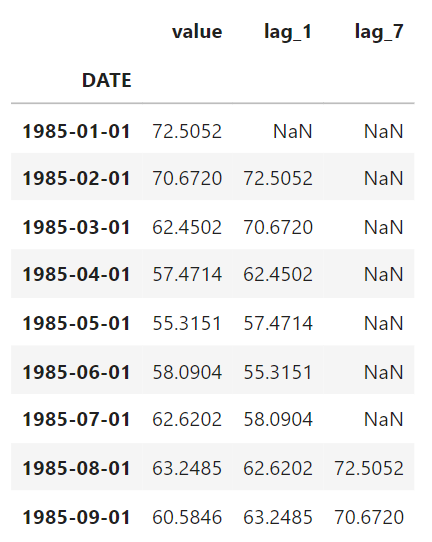
Những điều này nắm bắt **các giá trị trong quá khứ của biến mục tiêu để dự đoán các giá trị trong tương lai.** Xét cho cùng, xu hướng của ngày hôm qua thường đưa ra chỉ báo mạnh mẽ về hướng đi của ngày mai.

Ví dụ, bạn có thể kết hợp sản lượng năng lượng **từ những tháng hoặc năm trước** vì các tính năng có thể nâng cao đáng kể khả năng hiểu biết của mô hình về các mô hình lịch sử.

df['lag\_1'] = df['value'].shift(1) # Create a lag feature for the previous time step

df['lag\_7'] = df['value'].shift(7) # Create a lag feature for 7 months ago

*Ý tưởng này rất đơn giản: bằng cách dịch chuyển dữ liệu chuỗi thời gian, chúng tôi cung cấp cho mô hình khả năng truy cập vào****các quan sát trong quá khứ****, giúp mô hình tìm hiểu các mối phụ thuộc về thời gian trong dữ liệu.*



Hình 4: Ví dụ về tính năng trễ | Hình ảnh của tác giả.

Ví dụ, khi bạn dịch chuyển dữ liệu để tạo một tính năng trễ (ví dụ, **lag\_1**), các giá trị từ **một bước** trước hàng hiện tại được sử dụng làm tính năng. Tuy nhiên, đối với hàng dữ liệu đầu tiên, không có hàng "trước" nào để tham chiếu, vì vậy giá trị trong **lag\_1**cột trở thành **NaN**.

**2. Thống kê lăn: Làm mịn nhiễu**

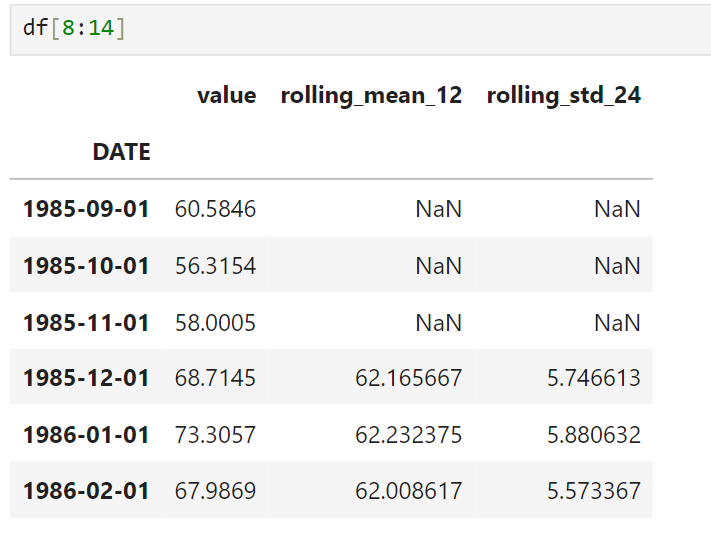
Dữ liệu chuỗi thời gian **thường có nhiễu** , khiến các mô hình khó nắm bắt được các mẫu thực tế hơn. Đây là nơi chúng tôi sử dụng **thống kê lăn.**

Bằng cách tính toán **các giá trị trung bình động** hoặc các số liệu khác (ví dụ: độ lệch chuẩn động), chúng ta có thể làm phẳng **các biến động ngắn hạn** và làm nổi bật các xu hướng cơ bản.

Ví dụ, để **làm mịn dữ liệu và quan sát xu hướng rộng hơn** , bạn có thể tính toán giá trị trung bình lăn trong một khoảng thời gian cụ thể, chẳng hạn như 12 tháng:

df['rolling\_mean\_12'] = df['value'].rolling(window=12).mean() # 12 months rolling mean

df['rolling\_std\_24'] = df['value'].rolling(window=12).std() # 12 months rolling standard deviation



Hình 5: Ví dụ về giá trị trung bình lăn và độ lệch chuẩn | Hình ảnh của tác giả.

**rolling(window=12)**: Điều này tạo ra một cửa sổ lăn gồm 12 bước thời gian (tháng trong trường hợp này) để tính toán giá trị trung bình lăn. Tương tự với độ lệch chuẩn.

Với các số liệu thống kê liên tục này, bạn sẽ có được cái nhìn ổn định hơn về dữ liệu, giúp mô hình của bạn dễ dàng nắm bắt các xu hướng dài hạn hơn.

**3. Tính năng Ngày-Giờ: Trích xuất thông tin chi tiết về thời gian**

Dữ liệu chuỗi thời gian không chỉ liên quan đến các giá trị thay đổi theo thời gian, vì điều quan trọng là phải biết **thời điểm** những thay đổi đó xảy ra.

**Các tính năng ngày giờ** như giờ trong ngày, ngày trong tuần hoặc tháng trong năm có thể cung cấp thông tin chi tiết quan trọng, đặc biệt nếu dữ liệu của bạn thể hiện các mô hình theo mùa.

Bạn có thể dễ dàng trích xuất các tính năng như vậy bằng Pandas:

df['hour'] = df.index.hour

df['day\_of\_week'] = df.index.dayofweek

df['month'] = df.index.month

df['year'] = df.index.year

**Đối với tập dữ liệu được sử dụng ở đây, việc trích xuất tháng trong năm** sẽ hợp lý hơn vì dữ liệu hàng tháng thường thể hiện các mô hình theo mùa (ví dụ: mức tiêu thụ năng lượng cao hơn vào mùa đông hoặc mùa hè).

Các tính năng này **cho phép mô hình của bạn nhận ra các mô hình tuần hoàn** và tác động theo mùa, giúp đưa ra dự đoán đáng tin cậy hơn.

**4. Sự khác biệt: Giải quyết xu hướng và tính thời vụ**

Trong một số trường hợp, dữ liệu chuỗi thời gian của bạn có thể thể hiện xu hướng mạnh hoặc tính theo mùa, **khiến dữ liệu không ổn định.**

Để xử lý điều này, bạn có thể tạo **các tính năng khác biệt** , **nắm bắt các thay đổi** giữa các bước thời gian liên tiếp thay vì các giá trị tuyệt đối. Điều này có thể giúp mô hình của bạn tập trung vào các thay đổi theo thời gian thay vì mức độ của chính chuỗi:

df['diff\_1'] = df['value'].diff(1) # Difference between consecutive observations

df['diff\_7'] = df['value'].diff(7) # Difference between 7 months observations

Những đặc điểm khác biệt này **có thể giúp ổn định giá trị trung bình của chuỗi** , đặc biệt là khi xu hướng hoặc biến động theo mùa chiếm ưu thế.

**5. Các tính năng tuần hoàn: Bao quanh tính tuần hoàn**

Khi xử lý **dữ liệu theo mùa** , cách biểu diễn thời gian bằng số đơn giản (ví dụ: tháng hoặc ngày) có thể không đáp ứng được kỳ vọng.

Ví dụ, sự khác biệt giữa tháng 12 (tháng 12) và tháng 1 (tháng 1) thực tế chỉ là một tháng, nhưng về mặt số học, nó có vẻ lớn hơn nhiều. Để giải quyết vấn đề này, bạn có thể tạo **các tính năng tuần hoàn** bằng cách chuyển đổi các tính năng liên quan đến thời gian thành các hàm sin và cos:

df['sin\_month'] = np.sin(2 \* np.pi \* df['month'] / 12)

df['cos\_month'] = np.cos(2 \* np.pi \* df['month'] / 12)

Với điều này, bạn cho phép mô hình của mình hiểu rõ hơn các mối quan hệ tuần hoàn trong dữ liệu, chẳng hạn như đỉnh và đáy theo mùa.

**6. Các tính năng bên ngoài: Thêm ngữ cảnh vào dữ liệu của bạn**

Cuối cùng, đừng quên rằng dữ liệu chuỗi thời gian của bạn không tồn tại trong chân không!

Thông thường, **việc kết hợp dữ liệu bên ngoài,** chẳng hạn như điều kiện thời tiết, ngày lễ hoặc chỉ số kinh tế, có thể cải thiện đáng kể hiệu suất của mô hình.

Ví dụ, trong dữ liệu của chúng tôi, **việc bao gồm dữ liệu nhiệt độ** như một tính năng hoặc số giờ ban ngày có thể mang lại cho mô hình của bạn lợi thế đáng kể (nếu dữ liệu này khả dụng).

df['avg\_temperature'] = external\_weather\_data['avg\_temp']

df['max\_temperature'] = external\_weather\_data['max\_temp']

df['sunlight\_hours'] = external\_weather\_data['sunlight\_hours']

Các tính năng bên ngoài này **cung cấp bối cảnh** giúp mô hình của bạn hiểu được các yếu tố bên ngoài ảnh hưởng đến dữ liệu chuỗi thời gian của bạn.