Chương 3.

MÔ HÌNH DỰ ĐOÁN CHUỖI THỜI GIAN ĐƠN BIẾN (part 2)

PhD. Nguyễn Thị Khánh Tiên

email: tiennntk@ut.edu.vn

Mô hình Machine Learning cho chuỗi thời gian

Nội dung chính

- Cách chuyển bài toán chuỗi thời gian thành bài toán supervised learning.
- Tạo đặc trưng:
  - Lag features
  - Rolling statistics
  - Biến thời gian (month, quarter, year)
- Các mô hình phổ biến:
  - Linear Regression
  - Random Forest
  - XGBoost

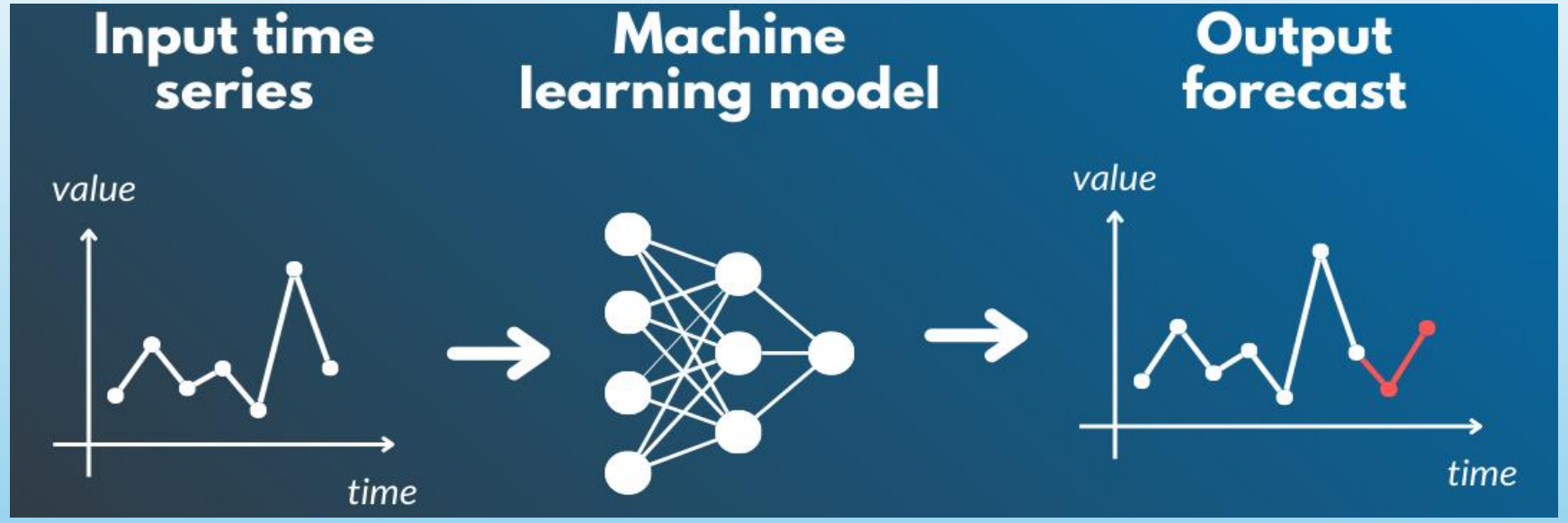
Phân tích

- Khả năng học quan hệ phi tuyến.
- Không yêu cầu chuỗi dừng.
- Phụ thuộc mạnh vào feature engineering.
- So sánh hiệu quả với mô hình thống kê.

Mục tiêu

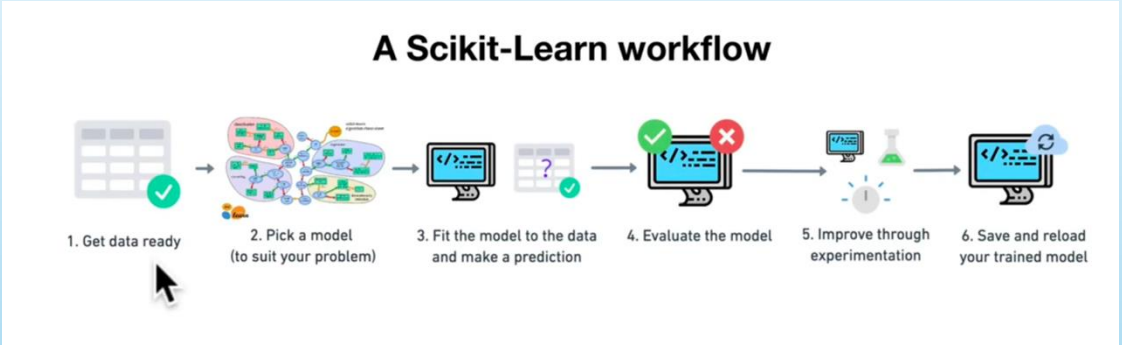
- Hiểu cách áp dụng ML cho bài toán time series.
- Biết ưu – nhược điểm của từng mô hình ML.
- Chuẩn bị nền tảng cho thực hành XGBoost cho time series.

Mô hình Machine Learning cho chuỗi thời gian



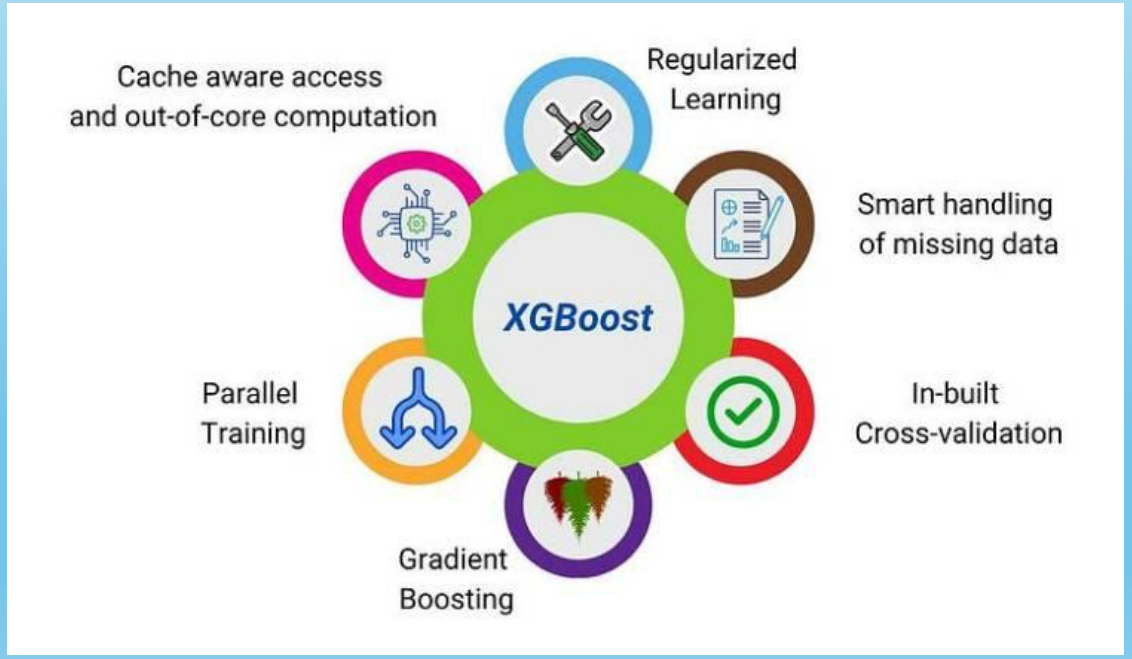
Sự khác nhau giữa mô hình truyền thống và mô hình ML

Tiêu chí	AR / ARIMA	Machine Learning
Giả định	Tính dừng	Không bắt buộc
Quan hệ	Tuyến tính	Phi tuyến
Feature	Lag thủ công	Feature engineering linh hoạt
Đa biến	Hạn chế	Rất tốt



Các mô hình Machine Learning phổ biến

Mô hình	Phi tuyến	Đa biến	Giải thích
Linear Regression	x	v	Rất tốt
Decision Tree	v	v	Tốt
Random Forest	vv	v	Trung bình
XGBoost	vvv	v	Khó hơn



Chuẩn bị dữ liệu cho ML Time Series

Trước khi có thể sử dụng máy học, các bài toán dự báo *chuỗi thời gian phải được định hình lại thành các bài toán học có giám sát*. Từ một chuỗi dữ liệu đơn lẻ thành các cặp chuỗi đầu vào và đầu ra bao gồm các bước sau:

1. **Xác định chuỗi thời gian ban đầu**

   Bài toán xuất phát từ một chuỗi đơn biến:

   $$\{y_1, y_2, \dots, y_T\}$$

   trong đó y_t là giá trị quan sát tại thời điểm t.

2. **Xác định mục tiêu dự báo**

   Dự báo một bước:

   $$y_t = f(y_{t-1}, \dots, y_{t-p})$$

   Dự báo nhiều bước:

   $$(y_t, \dots, y_{t+h}) = f(y_{t-1}, \dots, y_{t-p})$$

   Trong đó:

   p: số độ trễ (lag)

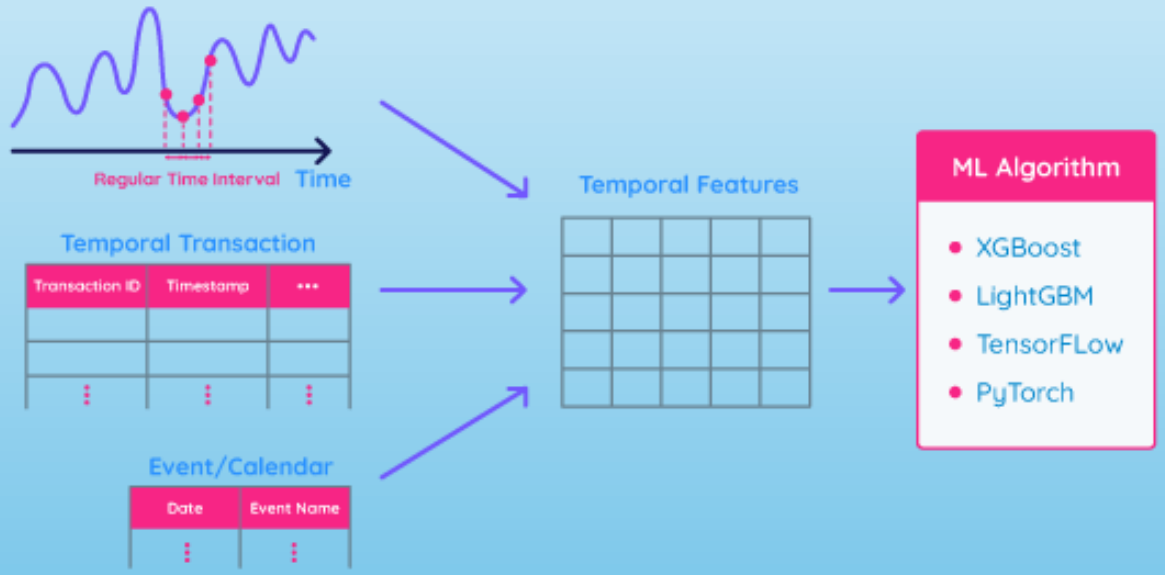
   h: số bước dự báo

3. **Tạo các đặc trưng trễ (lag feature)**

   Sử dụng các giá trị trong quá khứ của chuỗi để làm đầu vào:

   $$X_t = [y_{t-1}, y_{t-2}, \dots, y_{t-p}]$$

   Các đặc trưng này đại diện cho **thông tin lịch sử** của chuỗi.



Chuẩn bị dữ liệu cho ML Time Series

4. Áp dụng sliding window

Sử dụng cửa sổ trượt để tạo nhiều mẫu huấn luyện:

$$(X_t, y_t), (X_{t+1}, y_{t+1}), \dots$$

Mỗi cửa sổ tương ứng với một cặp:

Đầu vào (input sequence): dữ liệu quá khứ

Đầu ra (output/label): giá trị (hoặc chuỗi) tương lai

5. Hình thành tập dữ liệu học có giám sát

Sau khi áp dụng sliding window, thu được:

$$\mathcal{D} = \left\{\left(X^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$$

Trong đó:

$$X^{(i)} \in \mathbb{R}^p$$

$y^{(i)} \in \mathbb{R}$)hoặc $\mathbb{R}^h$với multi-step)

6. Loại bỏ các quan sát không đầy đủ

Các mẫu ở đầu chuỗi không đủ dữ liệu quá khứ sẽ bị loại bỏ để đảm bảo:

Không sử dụng thông tin tương lai

Mỗi mẫu có đủ số lag yêu cầu

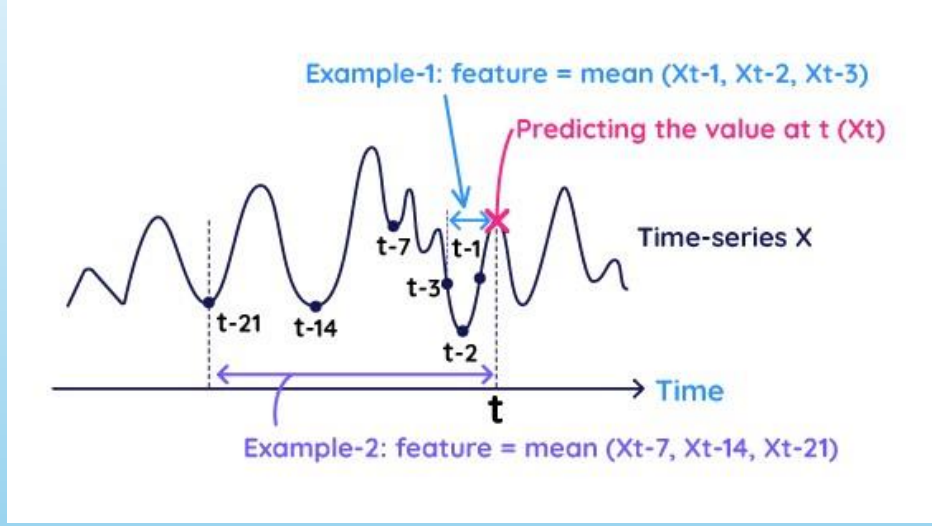
7. Chia dữ liệu train/test theo thời gian

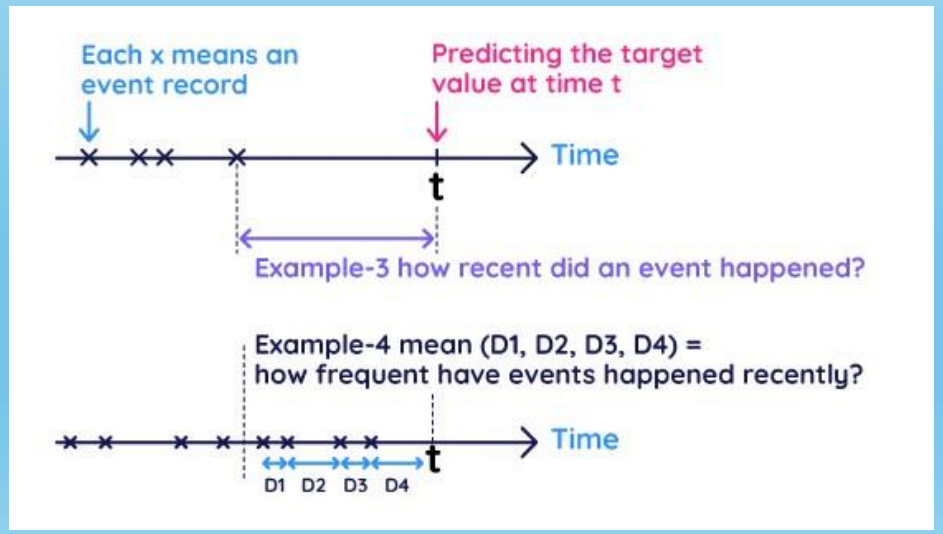
Chia tập dữ liệu theo thứ tự thời gian:

Train: các quan sát quá khứ

Test: các quan sát tương lai

Không xáo trộn (no shuffle) để tránh rò rỉ thông tin.





Cách chuyển đổi chuỗi thời gian thành bài toán học có giám sát trong Python

Chuẩn bị chuỗi dữ liệu:

```
import pandas as pd
df = pd.DataFrame({'y': series})
```

Tạo lag features

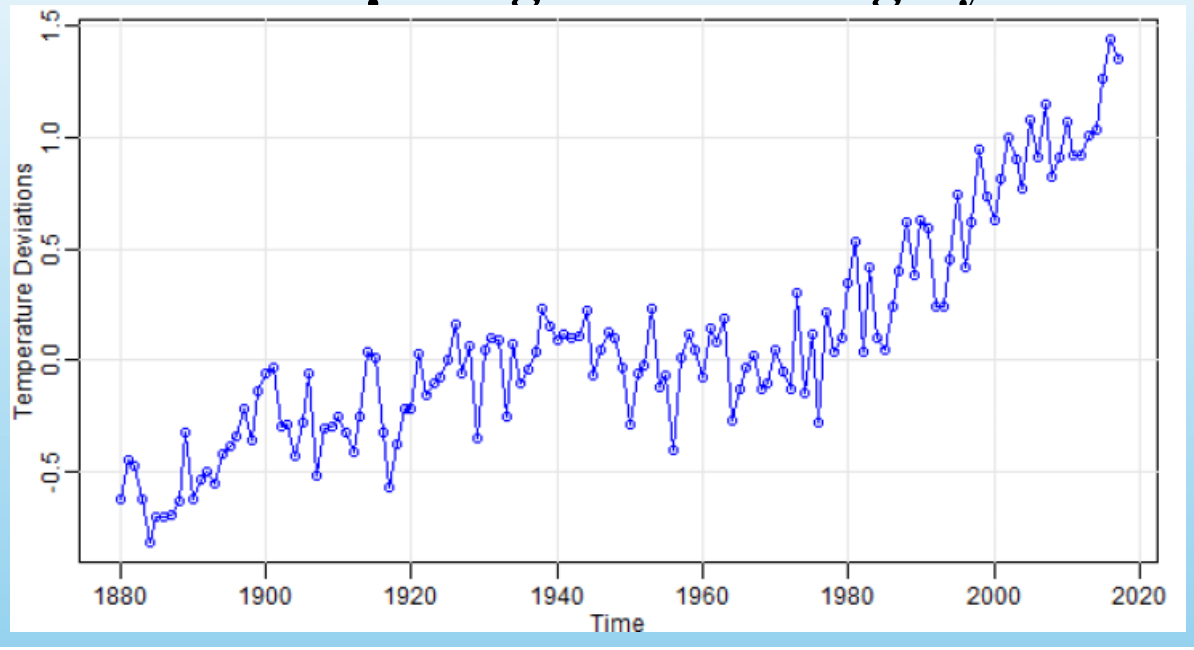
```
p = 3
for i in range(1, p+1):
    df[f'lag_{i}'] = df['y'].shift(i)
df = df.dropna()
```

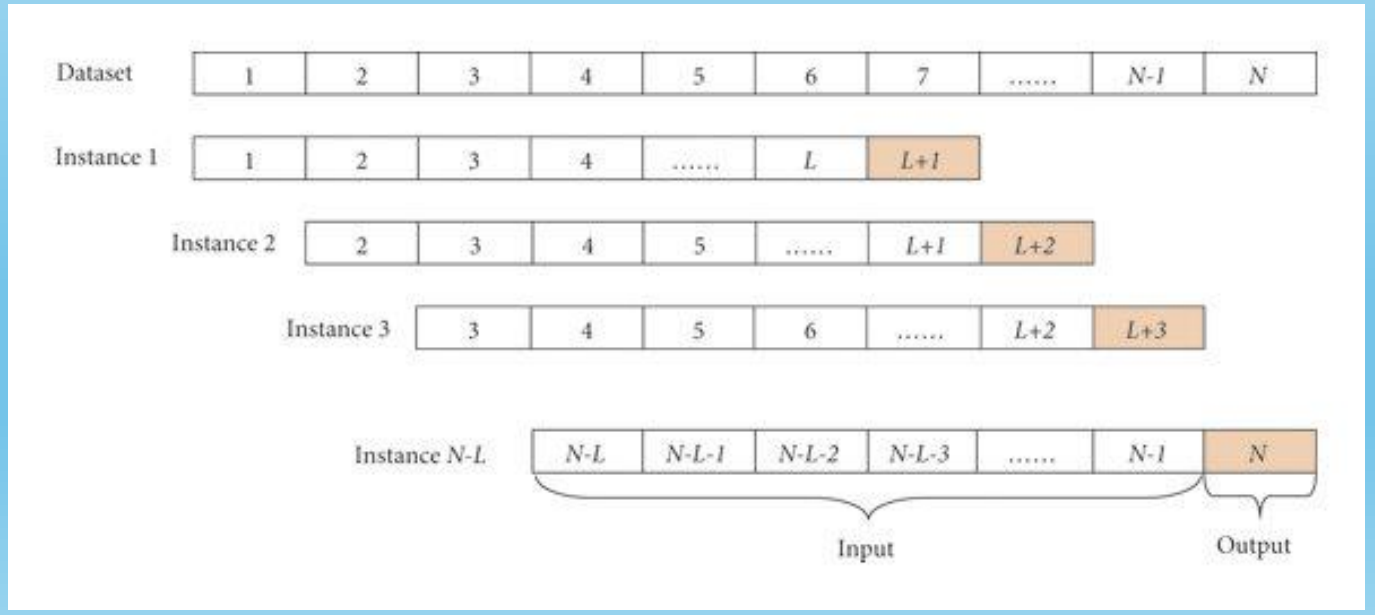
Chia X-y

```
X = df[[f'lag_{i}' for i in range(1, p+1)]]
y = df['y']
```

Chia train/test theo thời gian

```
split = int(len(df) * 0.8)
X_train, X_test = X[:split], X[split:]
y_train, y_test = y[:split], y[split:]
```





https://machinelearningmastery.com/convert-time-series-supervised-learning-problem-python/

Mô hình Linear Regression

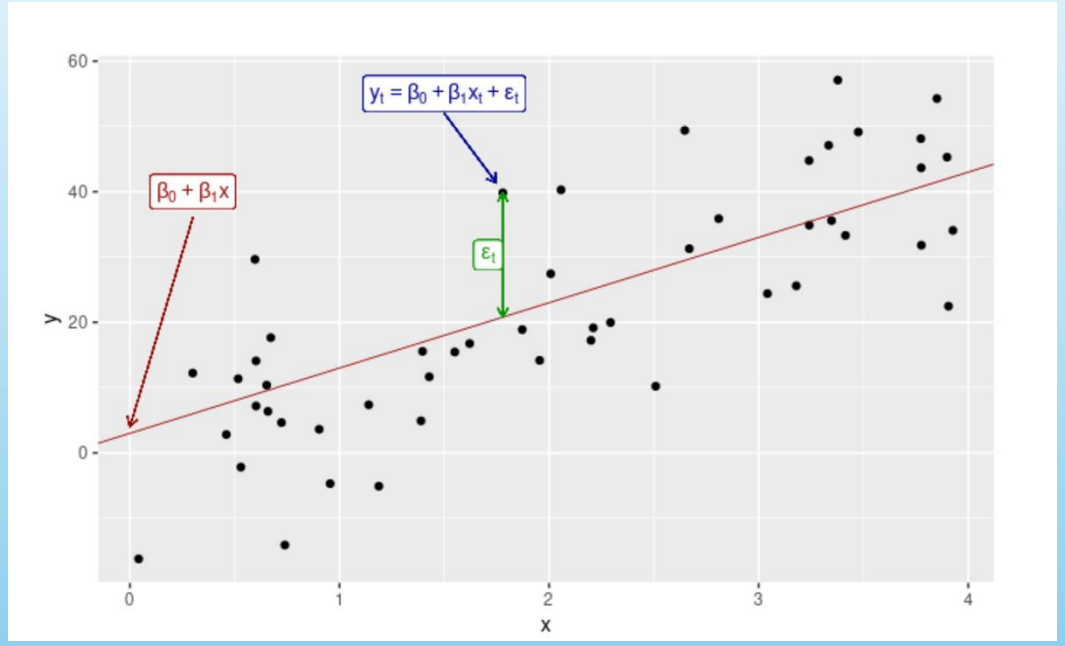
Mô hình Linear Regression (baseline)

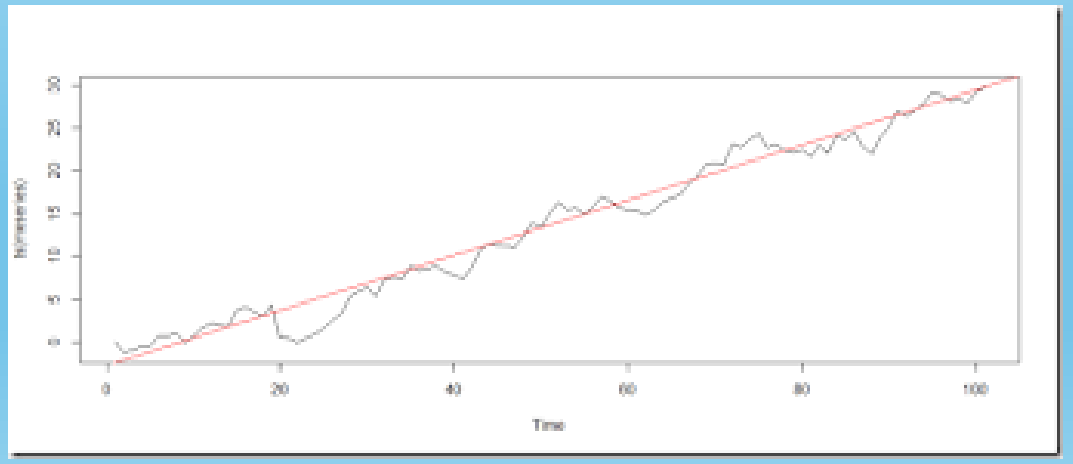
$$y_t = w_0 + \sum_{i=1}^{p} w_i \, y_{t-i}$$

mô hình **Linear Regression (Hồi quy tuyến tính)** thường được chọn làm "baseline" (mô hình cơ sở) vì tính đơn giản, tốc độ xử lý nhanh và khả năng giải thích cao

Các phương pháp tiếp cận chính:

- **Mô hình hóa xu hướng (Trend Modeling):** Sử dụng một chỉ số thời gian tăng dần (1,2,3,..) làm biến độc lập để dự báo giá trị.
- **Sử dụng biến trễ (Lag features):** Đây là cách phổ biến nhất.
- **Mô hình hóa tính thời vụ (Seasonality):** Tạo các biến giả (dummy variables) cho ngày trong tuần, tháng trong năm hoặc các khung giờ để bắt kịp các quy luật lặp lại.





Mô hình Linear Regression

Quy trình triển khai cơ bản (sử dụng thư viện ***scikit-learn***)

1. **Tạo đặc trưng:** Chuyển chuỗi thời gian thành bảng dữ liệu có các cột là lag_1, lag_2, hoặc time_index.
2. **Chia dữ liệu (Train/Test Split):** Không được xáo trộn (shuffle) dữ liệu. Bạn phải dùng dữ liệu cũ để huấn luyện và dữ liệu mới hơn để kiểm thử để đảm bảo tính thực tế của dự báo.
3. **Huấn luyện:** Sử dụng hàm LinearRegression() để khớp mô hình.
4. **Đánh giá:** Tính toán các chỉ số như MAE (Mean Absolute Error) hoặc RMSE (Root Mean Squared Error).

https://www.geeksforgeeks.org/machine-learning/step-by-step-guide-to-modeling-time-series-data-using-linear-regression/

Ưu điểm của hồi quy tuyến tính đối với dữ liệu chuỗi thời gian:

•**Khả năng giải thích** : Hồi quy tuyến tính cung cấp các hệ số có thể được giải thích để hiểu tác động của các biến số đến dự báo.

•**Tính đơn giản** : Dễ triển khai và dễ hiểu, giúp nó dễ tiếp cận cho các nỗ lực mô hình hóa ban đầu.

•**Hiệu quả tính toán:** Thời gian huấn luyện và dự đoán nhìn chung nhanh hơn so với các mô hình phức tạp hơn.

Nhược điểm của hồi quy tuyến tính đối với dữ liệu chuỗi thời gian:

•**Giả định về tính tuyến tính** : Hồi quy tuyến tính giả định mối quan hệ tuyến tính giữa các biến, điều này không phải lúc nào cũng đúng đối với các mô hình chuỗi thời gian phức tạp.

•**Tính linh hoạt hạn chế** : Nó có thể gặp khó khăn trong việc nắm bắt các mối quan hệ phi tuyến tính và các biến đổi theo mùa phức tạp nếu không sử dụng kỹ thuật xử lý đặc trưng.

•**Độ nhạy cảm với các giá trị ngoại lệ** : Các giá trị ngoại lệ có thể ảnh hưởng đáng kể đến đường hồi quy và dự đoán của mô hình.

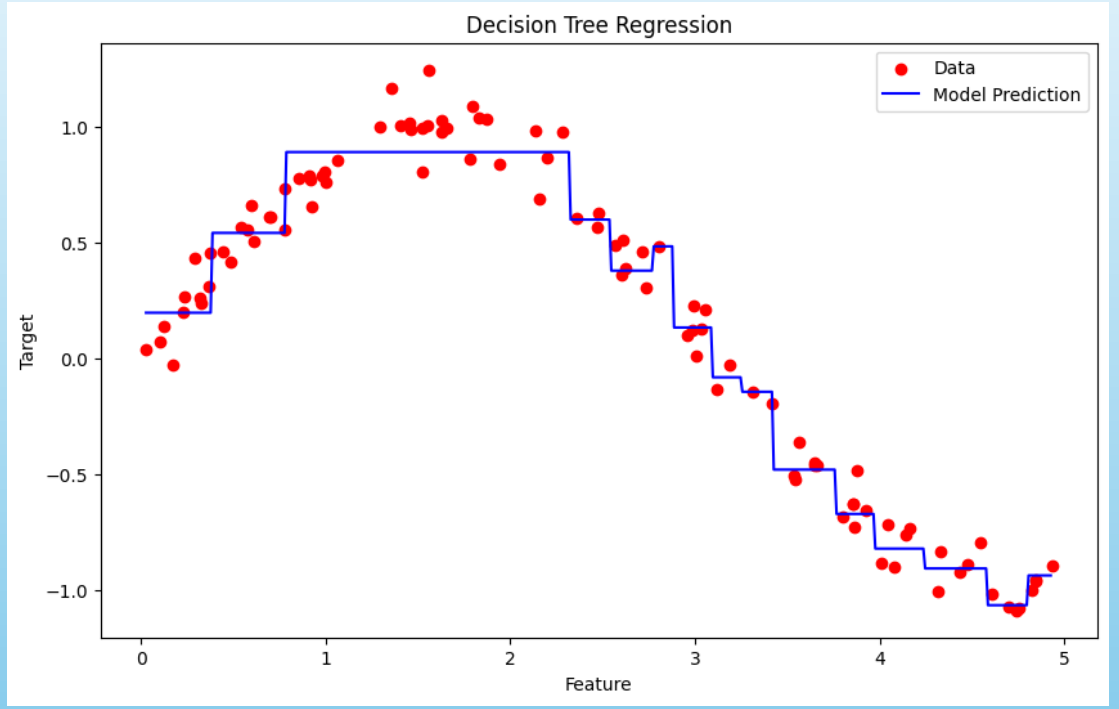
Mô hình Decision Tree Regression

Mô hình **Decision Tree Regression (DTR)** trong dự báo chuỗi thời gian là phương pháp sử dụng cấu trúc cây quyết định để đưa ra giá trị dự báo liên tục dựa trên các dữ liệu lịch sử.

- Chia dữ liệu theo ngưỡng
- Học được **phi tuyến**
- Không cần chuẩn hoá dữ liệu

Cách hoạt động đối với chuỗi thời gian. Dữ liệu chuỗi thời gian vốn là một chuỗi các giá trị theo thứ tự. Để sử dụng DTR cần phải chuyển đổi dữ liệu này thành dạng bảng thông qua kỹ thuật **trích xuất đặc trưng**:

- **Đặc trưng trễ (Lag features):** Sử dụng các giá trị ở quá khứ (t-1, t-2,...) làm đầu vào để dự báo giá trị tại thời điểm t.
- **Đặc trưng thời gian:** Trích xuất các thuộc tính như thứ trong tuần, tháng, quý, hoặc các sự kiện đặc biệt (lễ, Tết).
- **Thống kê cửa sổ trượt (Rolling windows):** Tính giá trị trung bình, độ lệch chuẩn trong một khoảng thời gian gần nhất.



Mô hình Decision Tree Regression

Ưu điểm:

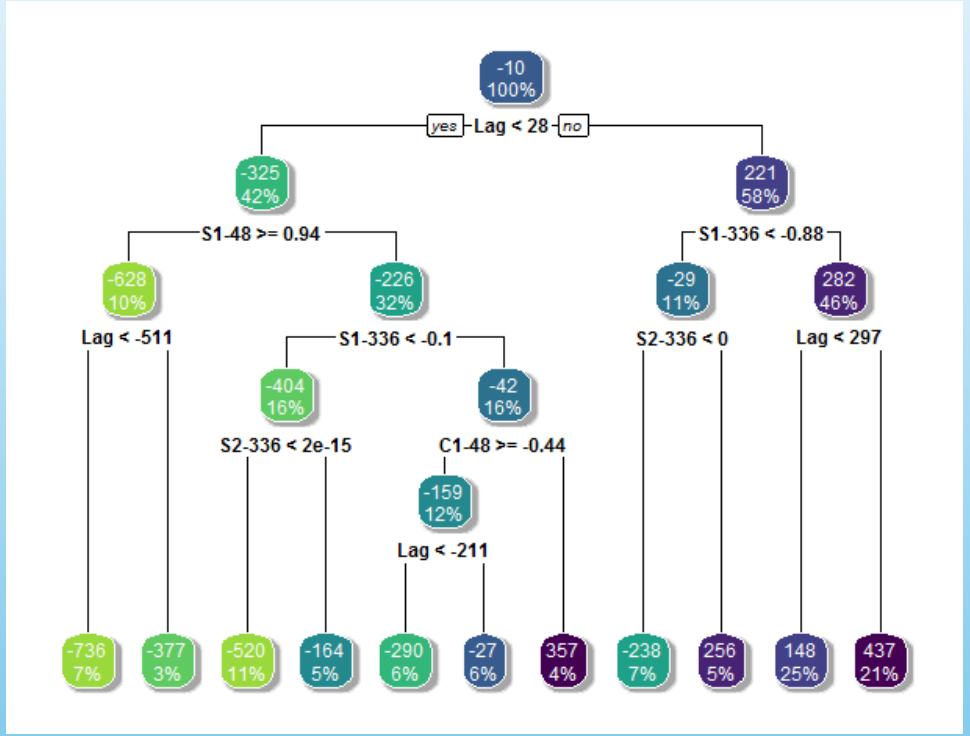
- Dễ giải thích: Có thể trực quan hóa quá trình ra quyết định qua cấu trúc cây.
- Không yêu cầu chuẩn hóa: Hiệu quả với cả dữ liệu số và dữ liệu phân loại mà không cần xử lý phức tạp.
- Tốc độ nhanh: Quá trình huấn luyện và dự báo diễn ra nhanh chóng.

Nhược điểm:

- Dễ Overfitting: Mô hình dễ học thuộc lòng dữ liệu nhiễu trong quá khứ nếu không giới hạn chiều sâu cây.
- Không có khả năng ngoại suy: DTR không thể dự báo các giá trị nằm ngoài phạm vi dữ liệu huấn luyện (không bắt kịp xu hướng dài hạn).
- Độ ổn định thấp: Thay đổi nhỏ trong dữ liệu có thể dẫn đến cấu trúc cây thay đổi hoàn toàn.

Khi nào nên sử dụng

- Khi dữ liệu có các mối quan hệ phi tuyến tính phức tạp hoặc có tính mùa vụ rõ rệt.
- Thường được dùng làm mô hình cơ sở (baseline) hoặc thành phần cấu tạo nên các mô hình mạnh hơn như **Random Forest** hoặc **Gradient Boosting** để cải thiện độ chính xác.
- Phù hợp để xử lý các bài toán có nhiều dữ liệu khiếm khuyết trong chuỗi thời gian bằng cách tái tạo dữ liệu dựa trên mô hình hồi quy.



1) https://www.kaggle.com/code/robikscube/tutorial-time-series-forecasting-with-xgboost

2) https://medium.com/@rahul-mohan-data-portfolio/time-series-forecasting-with-python-machine-learning-decision-trees-gradient-boosting-2bd863fabe25

3) https://machinelearningmastery.com/forecasting-the-future-with-tree-based-models-for-time-series/

Mô hình Gradient Boosting / XGBoost

Mô hình **XGBoost (Extreme Gradient Boosting)** là một thuật toán học máy mạnh mẽ dựa trên kỹ thuật **Gradient Boosting**, cực kỳ hiệu quả cho các bài toán dự báo chuỗi thời gian (time series) khi dữ liệu được chuyển đổi về dạng học máy có giám sát

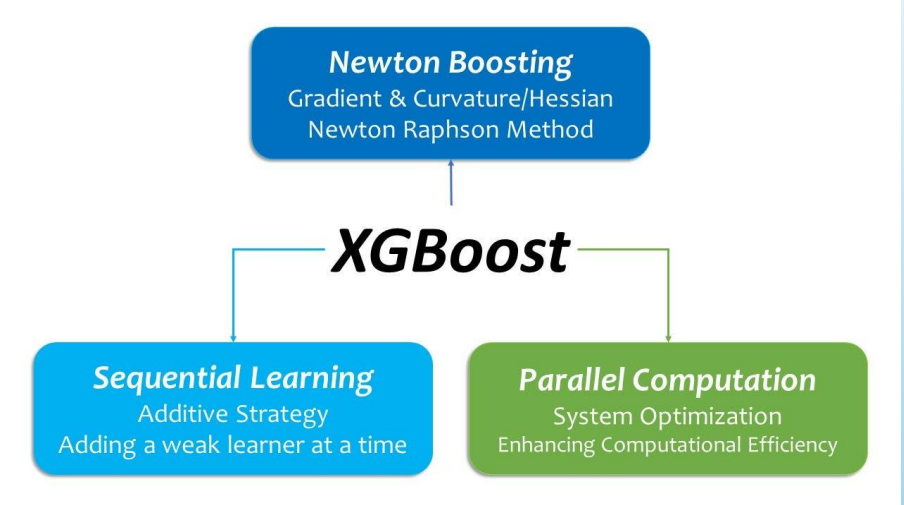
Các bước triển khai chính:

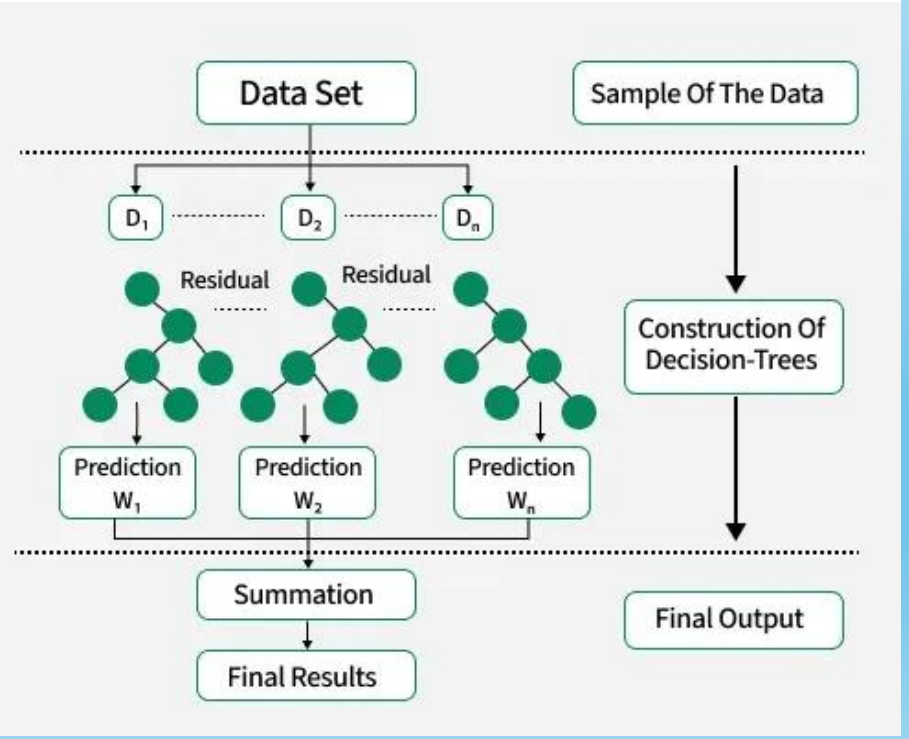
1. Biến đổi dữ liệu (Feature Engineering): Đây là bước quan trọng nhất để giúp XGBoost "hiểu" được yếu tố thời gian:

- **Lag features:**Sử dụng các giá trị trong quá khứ làm biến đầu vào.
- **Window statistics:** Tính toán trung bình trượt (rolling mean), giá trị lớn nhất/nhỏ nhất trong một khoảng thời gian.
- **Datetime features:** Trích xuất thông tin như giờ, ngày trong tuần, tháng, quý, năm từ cột thời gian

2. Huấn luyện và Đánh giá:

- **Walk-forward validation:** Không thể dùng cross-validation ngẫu nhiên thông thường vì sẽ làm lộ dữ liệu tương lai. Thay vào đó, cần dùng kỹ thuật kiểm chứng tịnh tiến (chia dữ liệu theo thứ tự thời gian).
- **Xử lý xu hướng (Trend):** XGBoost đôi khi gặp khó khăn với dữ liệu có xu hướng tăng/giảm mạnh vì nó không tự suy diễn (extrapolate) tốt bên ngoài phạm vi huấn luyện. Giải pháp thường là khử xu hướng (detrending) trước khi đưa vào mô hình.





Mô hình Gradient Boosting / XGBoost

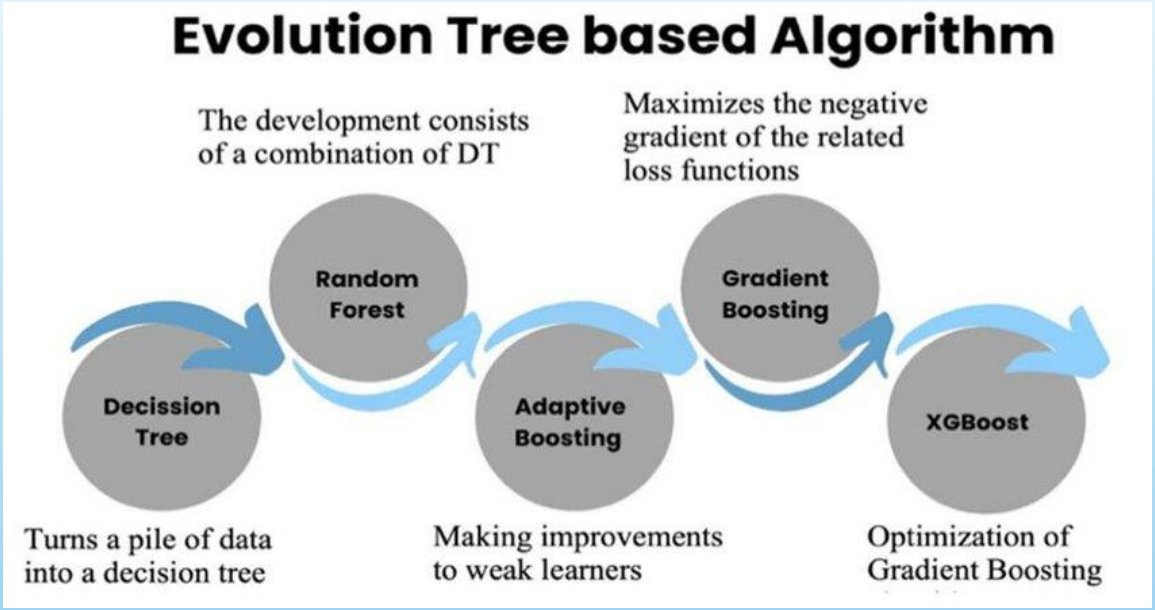
Ưu điểm:

- Độ chính xác cao: Thường cho kết quả vượt trội so với ARIMA hay Random Forest trên dữ liệu phức tạp.
- Tốc độ nhanh: Tối ưu hóa tính toán song song và xử lý được tập dữ liệu lớn.
- Xử lý dữ liệu khuyết: Tự động xử lý các giá trị thiếu trong tập dữ liệu thực tế.

Nhược điểm:

- Không tự động nắm bắt xu hướng: Cần xử lý dữ liệu (stationarity) kỹ lưỡng trước khi huấn luyện.
- Feature Engineering phức tạp: Đòi hỏi nhiều công sức tạo biến đặc trưng từ dữ liệu gốc.
- Dễ quá khớp (Overfitting): Cần điều chỉnh siêu tham số (tuning) cẩn thận.

https://xgboost.readthedocs.io/en/stable/



Mô hình XGBoost cho chuỗi thời gian nên sử dụng trong các trường hợp cụ thể sau để đạt hiệu quả tối ưu nhất :

1. Khi dữ liệu có nhiều biến bổ trợ (Exogenous Variables)

XGBoost vượt trội hơn các mô hình truyền thống (như ARIMA) khi bạn có nhiều yếu tố ngoại sinh ảnh hưởng đến kết quả.

•**Ví dụ:** Dự báo nhu cầu điện năng dựa trên nhiệt độ, độ ẩm, các ngày lễ và giá điện.

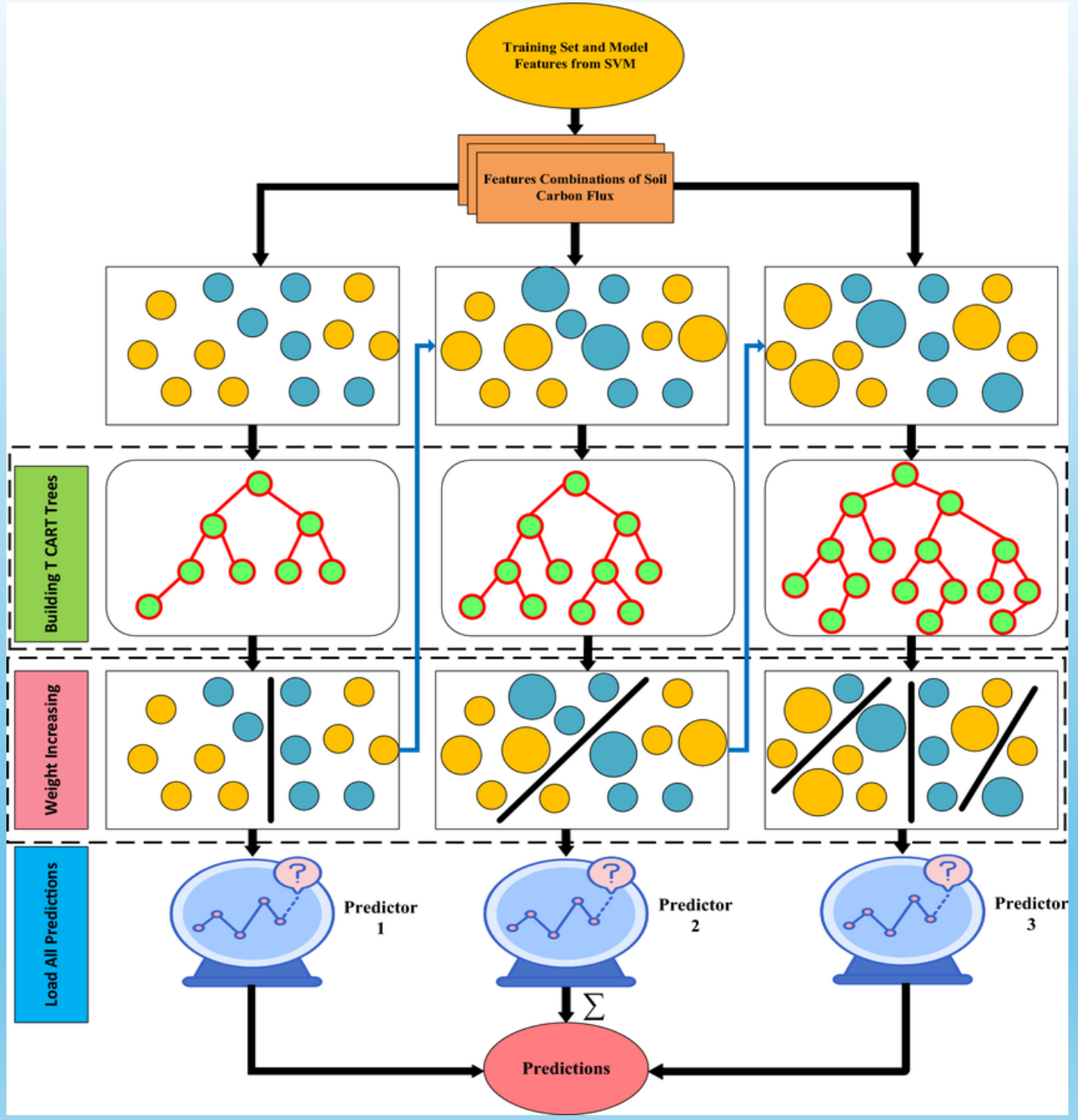
•**Lý do:** XGBoost xử lý cực tốt dữ liệu dạng bảng và có khả năng tự động nắm bắt các mối quan hệ phi tuyến giữa các biến này.

2. Khi mối quan hệ giữa các biến là phi tuyến phức tạp

Nếu chuỗi thời gian của bạn không tuân theo các quy luật tuyến tính đơn giản hoặc có các tương tác phức tạp mà mô hình thống kê khó nhận diện.

•**Ví dụ:** Dự báo lưu lượng truy cập web hoặc biến động thị trường chứng khoán vốn có tính bất định cao.

•**Lý do:** Cơ chế ensemble (kết hợp nhiều cây quyết định) giúp mô hình học được các ngưỡng điều kiện và sự kết hợp biến phức tạp.



Mô hình Gradient Boosting / XGBoost

3. Khi tập dữ liệu lớn và cần tốc độ huấn luyện nhanh

So với các mô hình Deep Learning như LSTM, XGBoost thường nhanh hơn đáng kể trong việc huấn luyện và tinh chỉnh tham số.

- **Tình huống:** Cần thử nghiệm nhanh nhiều giả thuyết hoặc triển khai mô hình trong môi trường có tài nguyên tính toán hạn chế hơn so với GPU cần cho Deep Learning.
- **Ưu thế:** Khả năng xử lý song song và tối ưu hóa bộ nhớ giúp XGBoost duy trì hiệu năng cao trên dữ liệu quy mô lớn.

4. Khi dữ liệu có nhiều giá trị khuyết hoặc nhiễu

XGBoost có cơ chế tự động học cách xử lý các giá trị thiếu (missing values) mà không cần bước tiền xử lý phức tạp như điền giá trị trung bình.

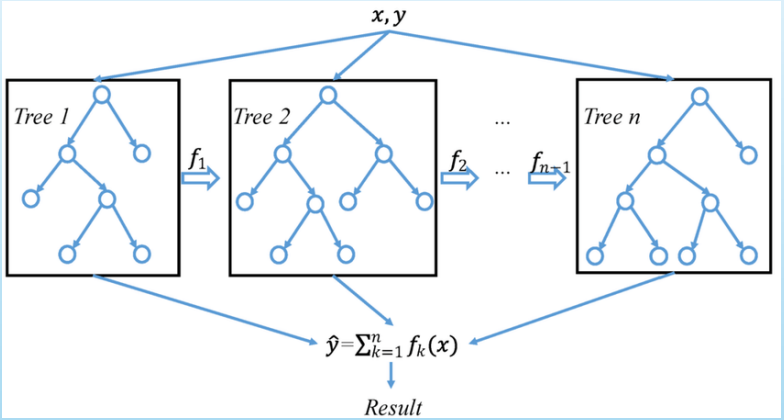
- **Ứng dụng thực tế:** Các bài toán dự báo trong thủy văn (lũ lụt) hoặc cảm biến công nghiệp thường xuyên gặp lỗi mất dữ liệu.

5. Dự báo ngắn hạn và trung hạn

XGBoost thường đạt độ chính xác rất cao cho các dự báo ngắn hạn nhờ khả năng nắm bắt nhanh các biến động gần nhất (thông qua Lag features).

Tuy nhiên XGBoost không tự suy diễn xu hướng (extrapolate) tốt như ARIMA nếu dữ liệu tương lai nằm hoàn toàn ngoài phạm vi giá trị của tập huấn luyện.

→ nên sử dụng XGBoost khi mục tiêu chính là **độ chính xác của dự báo** hơn là việc giải thích các hệ số thống kê, và khi có khả năng thực hiện kỹ thuật trích xuất biến (feature engineering) tốt.



How to Use XGBoost for Time Series Forecasting
https://machinelearningmastery.com/xgboost-for-time-series-forecasting/

[Tutorial] Time Series forecasting with *XGBoosthttps://www.kaggle.com/code/robikscube/tutorial-time-series-forecasting-with-xgboost*

Đánh giá mô hình

1. Đánh giá mô hình (Evaluation Metrics)

- Các chỉ số đo lường sai số giữa giá trị dự báo và giá trị thực tế:
  - **MAE (Mean Absolute Error):** Trung bình sai số tuyệt đối. Dễ hiểu vì cùng đơn vị với dữ liệu gốc.
  - **RMSE (Root Mean Squared Error):** Phạt nặng các sai số lớn. Thích hợp khi bạn muốn tránh các dự báo lệch quá xa.
  - **MAPE (Mean Absolute Percentage Error):** Đo lường sai số theo tỷ lệ phần trăm. Hữu ích khi so sánh các chuỗi có quy mô khác nhau.

2. Backtesting (Kiểm chứng ngược)

- Khác với dữ liệu bảng (Cross-validation ngẫu nhiên), backtesting chuỗi thời gian phải giữ nguyên trình tự thời gian:
  - **Time Series Split (Expanding Window):** Bắt đầu với một tập dữ liệu nhỏ để huấn luyện, sau đó mở rộng dần cửa sổ huấn luyện và dự báo cho điểm tiếp theo. Có thể sử dụng TimeSeriesSplit của Scikit-Learn.
  - **Sliding Window (Rolling Window):** Giữ kích thước tập huấn luyện cố định và "trượt" cửa sổ này dọc theo thời gian. Phương pháp này tốt khi dữ liệu cũ không còn phù hợp với xu hướng hiện tại.
  - **Walk-forward Validation:** Đây là "tiêu chuẩn vàng". Sau mỗi bước dự báo, dữ liệu thực tế sẽ được thêm vào tập huấn luyện để dự báo cho bước tiếp theo.

3. Các bước thực hiện chuẩn

- Chia dữ liệu: Chia thành tập Train (quá khứ) và Test (gần nhất). Lưu ý không bao giờ trộn (shuffle) dữ liệu.
- Chạy Backtest: Sử dụng Sliding hoặc Expanding window để tính toán các chỉ số đánh giá trên nhiều điểm thời gian khác nhau.
- Phân tích phần dư (Residual Analysis): Kiểm tra xem sai số có phải là nhiễu trắng (White Noise) hay không. Nếu phần dư còn quy luật (ví dụ: tự tương quan), mô hình vẫn chưa khai thác hết thông tin.

References:

- Time Series Forecasting as Supervised Learning
  → https://machinelearningmastery.com/time-series-forecasting-supervised-learning/
- How to Convert a Time Series to a Supervised Learning Problem in Python
  →https://machinelearningmastery.com/convert-time-series-supervised-learning-problem-python/
- A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning
  → https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/
- How to Use XGBoost for Time Series Forecasting
  →https://machinelearningmastery.com/xgboost-for-time-series-forecasting/
- [Tutorial] Time Series forecasting with XGBoost
  →https://www.kaggle.com/code/robikscube/tutorial-time-series-forecasting-with-xgboost
- How To Backtest Machine Learning Models for Time Series Forecasting
  →https://machinelearningmastery.com/backtest-machine-learning-models-time-series-forecasting/