Gradient Boosting on Trees for Time Series Forecasting: Best Practices

Wojciech Wideł *

08.03.2026

1 Data Strategy

- **Stationarity (The Trend Trap):** Tree models cannot extrapolate trends. Always difference the target ($y_t - y_{t-1}$) or detrend data before training.
  - *Log Transform:* Apply `log1p(target)` when variance scales with the level (multiplicative seasonality) or data is highly skewed. This stabilizes variance and forces the model to learn *relative* percentage changes rather than absolute unit changes.
  - *Inversion:* Reconstruct the forecast by anchoring to the last known actual value y_t:
    - *Differenced Target:* $\hat{y}_{t+h} = y_t + \sum_{i=1}^{h} \widehat{\Delta y}_{t+i}$
    - *Log-Differenced Target:* $\hat{y}_{t+h} = y_t \times \exp\left(\sum_{i=1}^{h} \widehat{\Delta \log} \, y_{t+i}\right)$
- **Global Modeling:** Pool all products/locations into a single dataset. Use IDs (e.g., `store_id`) as categorical features to share statistical strength.
  - *Scale Sensitivity:* High-volume series dominate standard MAE/MSE loss. Use Log-Transformation or sample weights to ensure the model learns patterns from low-volume series as well.
- **Zero-Inflation (The Sparsity Problem):** Supply chain data often contains exact zeros (stockouts, no demand).
  - *Modeling Choice:* Standard regression objectives treat zeros as noise. For intermittent demand, use `tweedie` objective or a two-stage model (Classifier → Regression).
- **Avoid Leakage:** When creating lag features, simulate the "data availability" cutoff perfectly. A prediction for week t can only use data available strictly before t.

2 Forecasting Strategy

- **Recursive (Iterative):** Train a single model for $t+1$. Feed predictions back as inputs for subsequent steps $(t+2, \ldots, t+h)$.
  - *Pros:* Maximum data utilization; single model to maintain; flexible horizon.
  - *Cons:* Error accumulation (compounding errors); distribution shift (training on actuals, predicting on predictions).
- **Direct:** Train h separate models. Model i predicts step $t+i$ directly using actual history available at t.
  - *Pros:* No error accumulation; consistent train/test distributions.
  - *Cons:* High computational cost; ignores temporal dependencies between steps.
  - *Smoothness Constraint:* Direct forecasts can be "jagged" (unstable from week to week). Post-processing smoothing or hybrid approaches (DirRec) are recommended for inventory stability.
  - *Horizon Viability:* Training h models becomes computationally prohibitive and noisy for long horizons (e.g., $h > 26$). Recursive strategies are often preferred for long-term strategic planning.
- **Hybrid (DirRec):** Train h models, but pass predictions from model $i-1$ as a feature to model i.
  - *Pros:* Captures sequential dependencies while mitigating error propagation.

*With the help of `z.ai`.

3 Feature Engineering

- **Seasonality: Lags vs. Fourier:**
  - *Lag Features (e.g., `lag_52`):* Excellent for high-autocorrelation but drops initial observations. Trees cannot extrapolate if the lag value exceeds the training range.
  - *Fourier Terms:* Sine/Cosine pairs capture smooth periodic patterns. Since values are bounded in $[-1, 1]$ and repeat cyclically, trees do not extrapolate—they simply apply learned rules to recurring values. Best for stable seasonality.
- **Cyclic Encoding:** Do not use raw integers for time (e.g., 1-12). Use Sine/Cosine transformations ($\sin(2\pi\frac{\text{month}}{12})$) so that cyclic time points (Dec $\rightarrow$ Jan) are mathematically close.
- **Holiday Windows:** Binary flags are insufficient. Create features like `days_to_holiday` and `days_since_holiday` to capture sales ramps.
- **Rolling Statistics:** Generate `rolling_mean`, `rolling_std`, and `rolling_max`.
  - *Critical:* Apply `.shift(1)` before rolling calculations to ensure the window ends at $t-1$, preventing data leakage.
- **The Exogenous Trap:** Only include features (e.g., Price, Weather) if their future values are known or can be reliably forecasted. Using "actual future weather" creates unreplicable leakage in production.

4 Evaluation Metrics

- **Aggregation:** In global modelling, use scaled metrics, such as MASE, RMSSE, BIAS% or MAE%. Compute metrics per series, report the mean and the standard deviation.
- **BIAS (Mean Error):**

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

  Measures systematic over/under-forecasting.
  - *Sign:* Positive value = Over-forecast.
  - *Scaling:* BIAS% = BIAS/$\bar{y}_{train}$. Normalizes error by average volume, allowing fair comparison between high- and low-volume series in a global model.
- **MAE (Mean Absolute Error):**

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

  Average magnitude of errors. Robust to outliers.
  - *Theory:* Minimizing MAE estimates the **median**.
  - *Scaling:* MAE% = MAE/$\bar{y}_{train}$. Essential for global models to prevent high-volume items from masking poor performance on low-volume items.
  - *Bias Correction:* Because MAE targets the median, it systematically under-forecasts skewed demand (median $<$ mean). If total volume planning is required, apply a scaling factor or use RMSE.
- **RMSE (Root Mean Squared Error):**

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

  Penalizes large errors more heavily.
  - *Theory:* Minimizing RMSE estimates the **mean**. Sensitive to outliers (spikes).
  - *Context:* Preferred when large errors are disproportionately costly or if you need the model to predict average demand.

- **MASE (Mean Absolute Scaled Error):**

$$\text{MASE} = \frac{\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|}{\frac{1}{N-1}\sum_{t=2}^{N}|y_t - y_{t-1}|}$$

  Scales error by the in-sample naive forecast.

  - *Interpretation:* < 1.0 means better than naive; > 1.0 means worse.
  - *Edge Case:* If the training series is constant (zero variation), the denominator is 0. Handle by returning MAE or dropping the series.

- **RMSSE (Root Mean Squared Scaled Error):**

$$\text{RMSSE} = \frac{\sqrt{\frac{1}{h}\sum_{i=1}^{h}(y_i - \hat{y}_i)^2}}{\sqrt{\frac{1}{N-1}\sum_{t=2}^{N}(y_t - y_{t-1})^2}}$$

  Similar to MASE but based on squared error.

  - *Context:* Preferred when large errors are disproportionately costly (non-linear cost functions), bridging the gap between MASE (linear cost) and RMSE.

5 Model Configuration

- **Categorical Features:** Avoid One-Hot Encoding for high-cardinality IDs (e.g., Store ID). Use LightGBM's native categorical handling (`dtype="category"`) or regularized Target Encoding.
- **Objective Function:**
  - *Point Forecast:* Use `regression_l1` (MAE) for robustness, or `tweedie` for sales data with many exact zeros (intermittent demand).
  - *Uncertainty:* Use `quantile` objective with $\alpha = 0.05$ (pessimistic) and $\alpha = 0.95$ (optimistic) for inventory safety stock planning.
- **Regularization:** Keep `min_child_samples` (> 20) and `feature_fraction` (< 1.0) reasonable to prevent memorization of specific time steps.

6 Training & Tuning

- **The "Lower Learning Rate" Trick:** After hyperparameter search, manually lower the `learning_rate` (e.g., by 5x) and increase `num_boost_round` accordingly. Use early stopping to find the optimal iteration count.
- **Dataset Partitioning:** Set aside the last h periods as the Test Set (Final Exam). Do not use random K-Fold.
- **Cross-Validation:** Use `TimeSeriesSplit` or Rolling Origin Cross-Validation. Move the training cutoff forward to validate on multiple time windows.
- **The "Gap":** If there is a data latency in production (e.g., data available 1 week late), introduce a gap between the training end and validation start to simulate this reality.