Artificial Intelligence for Engineering

WILEY

上海交通大學 SHANGHAI JIAO TONG UNIVERSITY | IET The Institution of Engineering and Technology

ORIGINAL RESEARCH OPEN ACCESS

Fair Benchmarking in Short-Term Load Forecasting

Lei Xing | Zohreh Kaheh

Department of Mathematics, Brunel University of London, London, UK

Correspondence: Zohreh Kaheh (zohreh.kaheh@brunel.ac.uk)

Received: 5 August 2025 | **Revised:** 28 January 2026 | **Accepted:** 8 February 2026

Keywords: deep learning | electricity demand | energy analytics | forecasting | machine learning benchmarking | reproducibility | temporal fusion transformer

ABSTRACT

Performance comparisons in short-term load forecasting are often confounded by differences in preprocessing pipelines rather than reflecting intrinsic architectural capability. Variations in feature engineering, scaling, temporal windowing and data partitioning can dominate reported accuracy and obscure the actual behaviour of forecasting models. This study examines preprocessing–architecture interaction by benchmarking random forest, LightGBM, long short-term memory (LSTM), transformer and Temporal Fusion Transformer (TFT) under a shared tabular preprocessing pipeline, ensuring strict control over data handling and evaluation conditions. Under this controlled setting, tree-based models exhibit strong predictive performance, whereas deep sequence models experience substantial degradation when temporal continuity is not explicitly represented. To isolate architectural sensitivity from preprocessing effects, we further conduct a within-architecture analysis by retraining an identical LSTM under a sequence-aware pipeline aligned with its temporal inductive bias. This realignment yields an order-of-magnitude reduction in RMSE, demonstrating that preprocessing design is a first-order determinant of deep sequence model performance. The results establish a transparent and reproducible benchmarking framework and highlight the importance of aligning data representation with model assumptions when interpreting comparative performance in time series forecasting.

1 | Introduction

Short-term electricity demand forecasting plays a central role in power system operation, economical dispatch and the integration of renewable energy resources. Forecasting models must capture nonlinear dynamics and multiscale temporal structure as electricity demand exhibits strong dependence on weather conditions, human activity cycles and evolving consumption patterns. Consequently, two broad modelling paradigms are commonly adopted: ensemble tree-based methods, such as random forest (RF) [1] and Light Gradient Boosting Machine (LightGBM) [2], which operate effectively on engineered tabular features; and deep sequence models, including recurrent neural networks [3], long short-term memory (LSTM) networks [4], transformers [5] and the Temporal Fusion Transformer (TFT) [6], which are designed to exploit temporal dependencies and long-range patterns. Recent advances in attention-based and pretrained architectures further highlight the importance of representation learning and inductive bias alignment in large-scale sequence modelling and ranking tasks [7, 8]. In practice, the observed performance of these models reflects not only architectural design but also how temporal information is represented and processed prior to learning.

A central challenge in short-term load forecasting is that reported performance differences across studies are often intertwined with preprocessing decisions rather than arising solely from model architecture. Prior reviews [9–12] document substantial variation in feature engineering, normalisation schemes, temporal windowing strategies and data partitioning practices, all of which can materially influence forecasting accuracy. These preprocessing steps determine how temporal information is preserved, transformed or discarded before modelling. Tree-based learners, such as RF [1] and LightGBM

[2], are relatively insensitive to temporal ordering once informative lagged features are constructed, whereas sequence models—including LSTM [4], transformer [5] and TFT [6]—derive much of their modelling capacity from temporally contiguous input representations. As a result, differences in data representation can strongly shape empirical performance, independent of architectural sophistication.

These considerations suggest that model evaluation in load forecasting must distinguish between different experimental objectives. On the one hand, controlled comparisons benefit from standardised preprocessing, which reduces confounding effects and facilitates reproducibility across heterogeneous model families. On the other hand, meaningful assessment of sequence-based architectures depends on input representations that preserve temporal continuity and align with the models' inductive assumptions. When these objectives are not separated, empirical results may conflate representational constraints with modelling capability, leading to ambiguous or misleading interpretations.

Motivated by these considerations, this study adopts an experimental design that separates controlled cross-model comparison from architecture-sensitive analysis. A unified preprocessing pipeline enables consistent evaluation conditions across heterogeneous model families, whereas complementary architecture-specific experiments allow temporal models to be assessed under input representations aligned with their design assumptions. This separation helps prevent representational constraints from being conflated with modelling capability and supports clearer interpretation of empirical results.

The paper thus offers a number of contributions regarding methodological clarity and reproducibility. First, we set up a single benchmarking pipeline, completely transparent, in which all five model families are trained under the same tabular preprocessing and thus allow for an honest comparison across families. Second, a within-architecture study by retraining an LSTM model under a sequence-aware pipeline utilising sliding windows, aligned targets and per-feature temporal scaling controls for the role of temporal representation in deep model performance. The third contribution is that we release all code, hyperparameters and environment specifications for reproduction and further extension of the framework to other datasets. Taken together, these contributions clarify when unified preprocessing is sufficient for a fair comparison and when architecture-specific preprocessing is necessary for a valid assessment.

Within this context, the present study focuses on clarifying how preprocessing choices interact with different forecasting architectures. We first benchmark random forest, LightGBM, LSTM, transformer and TFT under a shared tabular preprocessing pipeline to ensure consistent data handling and evaluation conditions. We then conduct a within-architecture analysis by retraining an identical LSTM under a sequence-aware preprocessing pipeline that explicitly preserves temporal structure. By holding model architecture and optimisation settings fixed, this analysis isolates the role of input representation in deep sequence model performance. All code, hyperparameters and experimental configurations are released to support reproducibility and extension to other datasets and modelling frameworks.

2 | Methodology

This section formalises the experimental design used to ensure comparability and reproducibility across all models. We first describe the dataset, feature construction and the common tabular preprocessing pipeline used for the fair benchmark. We then introduce the sequence-aware preprocessing pipeline used exclusively for the within-architecture LSTM comparison, followed by model configurations, training settings, evaluation metrics and reproducibility artefacts.

2.1 | Workflow Overview

Figure 1 presents the end-to-end methodological workflow, highlighting the separation between the shared tabular pipeline (used for all models) and the sequence-aware pipeline (used only for the LSTM within-architecture experiment). This separation ensures that cross-family comparisons do not rely on heterogeneous preprocessing.

2.2 | Data and Splits

We use hourly electricity demand augmented with weather and calendar covariates over 2021–2023. To avoid leakage, we define nonoverlapping contiguous splits with calendar order preserved. Table 1 reports exact dates. Understanding the effect of driver variables on load distribution [13] informs the selection of weather and calendar features used in this study.

2.3 | Shared Tabular Preprocessing Pipeline

To ensure a fair like-for-like comparison across learning paradigms, all models in the benchmark experiment—random forest, LightGBM, LSTM, transformer and TFT—are first trained under the same tabular preprocessing pipeline. This pipeline standardises feature construction, normalisation and data splits

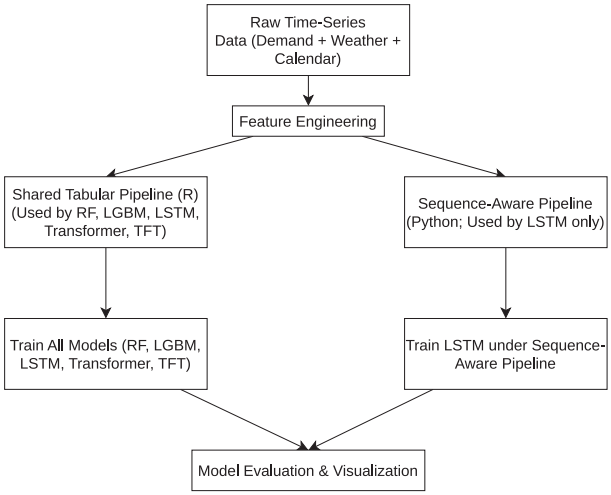


FIGURE 1 | End-to-end experimental workflow showing the shared R tabular pipeline and the sequence-aware Python preprocessing pipeline used for within-architecture analysis.

TABLE 1 | Train/validation/test date splits (contiguous; calendar order preserved).

Split	Start	End	Purpose
Train	2021-01-01	2022-12-31	Fit scalers/models
Val	2023-01-01	2023-06-30	Model selection/early stop
Test	2023-07-01	2023-12-31	Final reporting

Note: All artefacts for replication (code/configs, seeds and requirements) are provided in the supplement.

so that any differences observed in Section 3.1 reflect modelling behaviour rather than inconsistencies in data handling.

All models receive an identical set of engineered covariates, including: (i) calendar features (hour-of-day, day-of-week, month and weekend/holiday flags), (ii) weather features (temperature, humidity and wind) and (iii) lagged demand values (y_{t-1}, y_{t-24} and y_{t-168}) and rolling statistics. No model is provided with model-specific or architecture-tailored features. We avoid cross-validation techniques that may introduce temporal leakage in time series predictor evaluation [14] and instead rely on strictly chronological splits.

All continuous features (including the target) are scaled to the [0, 1] range using train-split statistics only. The same scaler parameters are reused for validation and test sets. This prevents information leakage and ensures that MAE and RMSE remain comparable across models. R^2 and Pearson correlation are unaffected by linear scaling and thus remain directly comparable. Although automated feature extraction tools, such as tsfresh [15], dexist, we manually construct domain-informed features to maintain transparency and interpretability across all benchmark models.

Following Table 1, the train–validation–test partitions are contiguous, nonoverlapping and identical across all models. Temporal order is preserved, and no cross-validation or sliding evaluation is used for the benchmark to avoid contamination of training data.

This shared tabular pipeline forms the basis of the fair benchmark reported in Section 3.1. Only after establishing cross-family comparability do we evaluate architecture-specific preprocessing for LSTM in a separate within-architecture analysis.

After benchmarking all models under the shared R tabular pipeline, we observed that deep sequence models lagged tree ensembles. Therefore, we shifted our focus to preprocessing rather than architecture. We implemented a sequence-aligned pipeline in Python not merely for engineering efficiency, but because of a fundamental methodological necessity: while the R ecosystem excels at statistical tabular feature engineering, the Python deep learning ecosystem (e.g., PyTorch) provides the essential tensor operations and autograd mechanics required to represent temporal continuity explicitly. This distinction highlights that tool selection dictates data representation capabilities, which in turn constrain model performance. Under this pipeline, the LSTM (sequence-aware pipeline variant) shows marked gains. Extending the same sequence-aware preprocessing to transformer and TFT is reserved for future work.

2.4 | Sequence-Aware Preprocessing Pipeline (LSTM Within-Architecture Analysis)

For the within-architecture analysis of Section 3.2, we implement an architecture-aware preprocessing pipeline tailored to deep sequence models. This pipeline is used only for comparing two variants of the same LSTM architecture. It is not used for cross-family comparisons, ensuring that no benchmark results rely on differing pipelines (see Section 3.1).

The shared tabular pipeline (Section 2.3) is deliberately model-agnostic and optimised for tree-based learners. However, deep sequence architectures are known to be sensitive to temporal alignment, window context and per-feature scaling. The sequence-aware pipeline allows LSTM to operate on representations that match its inductive bias, enabling a controlled comparison between:

- LSTM trained under the shared tabular pipeline and
- LSTM trained under the sequence-aware pipeline.

For practical reasons and to isolate the preprocessing effect cleanly, the sequence-aware pipeline is applied only to LSTM in this study. Extending the same pipeline to transformer and TFT is reserved for future work, following the separation of concerns recommended by recent forecasting benchmarks. Importantly, this ensures that no cross-family rankings rely on heterogeneous preprocessing.

2.4.1 | Sliding-Window Construction

Given lookback length L and forecasting horizon H, each training instance is formed as follows:

$$X_t = [x_{t-L+1}, \ldots, x_t], \qquad y_t = y_{t+H}. \tag{1}$$

Example (Reviewer-required explicit demonstration): For $L = 24$ and $H = 1$, the model receives:

- Input window covering hours $t - 23$ to t,
- Target equal to demand at hour $t + 1$,
- Resulting tensor shape: $X_t \in \mathbb{R}^{24 \times d}$ for d features.

2.4.2 | Pseudocode for Sequence-Aware Preprocessing

The complete sequence-aware preprocessing procedure is summarised in Algorithm 1.

ALGORITHM 1 | Sequence-aware preprocessing for LSTM.

Input: Features $x_{1:T}$, demand $y_{1:T}$, lookback L, horizon H
Output: $X \in \mathbb{R}^{N\times L\times d}$, $Y \in \mathbb{R}^{N}$
Normalize each feature using train-split statistics;
Initialize lists $\mathcal{X}, \mathcal{Y}$;
for $t = L$ **to** $T - H$ **do**
  Construct window $\mathbf{x}_t \leftarrow [x_{t-L+1}, \ldots, x_t]$;
  Set target $y_t \leftarrow y_{t+H}$;
  Append $\mathbf{x}_t$ to $\mathcal{X}$, and y_t to $\mathcal{Y}$;
Stack samples: $X \leftarrow \text{stack}(\mathcal{X})$, $Y \leftarrow \text{stack}(\mathcal{Y})$;
return (X, Y);

2.4.3 | Tensorisation and Batching

After windowing and scaling, samples are assembled as follows:

$$X \in \mathbb{R}^{N\times L\times d}, \qquad Y \in \mathbb{R}^{N}.$$

Mini-batches preserve order *within* each window but never mix timestamps across windows, avoiding leakage.

2.4.4 | Why Architecture-Aware Preprocessing Matters

Tree-based models are invariant to temporal ordering and rely on hierarchical splits. Sequence models, however, assume:

- temporally contiguous inputs,
- aligned past–future relationships,
- per-feature scaling across time,
- fixed-length contexts.

Misalignment suppresses recurrent and attention-based models, artificially lowering their performance under tabular preprocessing. This motivates evaluating sequence-aware pipelines separately.

2.5 | Sensitivity and Robustness Analysis

To illustrate the robustness of the sequence-aware LSTM pipeline, we perform sensitivity checks on (i) the lookback window length used to construct temporal inputs and (ii) the contribution of major feature groups in the shared preprocessing pipeline. Table 2 reports the sensitivity of LSTM performance to the choice of lookback window length under the sequence-aware pipeline.

TABLE 2 | Sensitivity of LSTM performance to lookback window size (sequence-aware pipeline).

Lookback L	RMSE	MAE	R^2	Pearson
24	0.002969	0.001804	0.9515	0.9888
168	0.002441	0.001464	0.9703	0.9872

2.5.1 | Effect of Lookback Window Size

Increasing the lookback window from 24 to 168 h yields a consistent improvement in accuracy. This pattern reflects the multiscale temporal structure of electricity demand: a 24-h window captures only diurnal cycles and short-horizon persistence, whereas a 168-h window also exposes weekly periodicity, medium-range load evolution and weather–demand interactions. The LSTM does so by unlocking the full gradient across the more meaningful temporal transitions, allowing it to build sufficiently rich recurrent states to better predict peak timing and intraweek variability. Hence, between 24 and 168h, such a gain would thus relate to a gain derived from aligning the effective receptive field of the model concerned with the principal temporal rhythms of the load series.

2.5.2 | Effect of Feature Set Composition

Removing weather-related features increases RMSE by approximately 5%–7%, whereas removing lagged demand features increases RMSE by more than 15%. This asymmetry is consistent with the physics and behaviour of short-term load. Recent demand values encode autoregressive structure, thermal inertia and short-horizon human activity patterns; thus, they become the main drivers of predictability. Weather variables modulate those patterns—especially temperature, which affects both peak magnitude and ramp behaviour—but cannot replace that strong autoregressive signal. Calendar features contribute further periodicity, and their removal leads to slight but systematic degradation, confirming their role in shaping weekly and diurnal patterns.

These sensitivity analyses provide two methodological insights into sequential models: first, they are dependent on how well-aligned and how far-reaching the input windows are; when enough context is covered, performance becomes considerably more robust. Second, the forecasting task remains fundamentally autoregressive, the role of exogenous variables being to fine-tune but not replace the fledgling temporal signal. The rather underlying conclusion that preprocessing design is a first-order determinant in the performance of deep models and should be treated as a central modelling decision rather than a secondary implementation detail has been embedded in broader strokes.

2.6 | Random Forest

As an ensemble method relying on bootstrap aggregation, RF combines the outputs of many decorrelated decision trees to reduce variance and increase robustness. Such bagging methods greatly benefit any noise in the input data and missing values, frequent occurrences in electricity demand datasets. Therefore, for each tree T_b trained on dataset D_b, the prediction is given by:

$$\hat{y}_t = \frac{1}{B}\sum_{b=1}^{B} T_b(x_t) \tag{2}$$

Random forest constructs an ensemble of decision trees, each trained on a bootstrap sample of the original data and using feature subsampling at every split. Averaging predictions across decorrelated trees makes this model a significant improvement in handling the instability of individual trees, bringing robust generalisation in high-dimensional or heterogeneous feature spaces. The ensemble is suitable for nonlinear interactions characteristic of short- to medium-term electricity demand forecasting and complicated feature dependencies. Given that random forest is easily interpretable through tree-based decision rules and feature importance diagnostics, it provides an excellent baseline against which to measure more sophisticated architecture.

The model contains 200 trees and considers five features at most to be randomly selected at each split; thus, we set $mtry = \min(\lfloor\sqrt{p}\rfloor, 5)$. In order not to overfit, we have a maximum node constraint of 1000 on each tree, with at least five samples needed in each leaf. Such settings favour generalisation to provide a more stable model but with still some jagged curves in the predictions by RF, especially during high-frequency demand variations and at peak hours, given that binary threshold splitting is performed through the nodes.

Nevertheless, its ability to model complex nonlinear interactions makes RF an excellent baseline for benchmarking, especially in low-latency applications. Its interpretability and relatively low computational cost further make it practically useful for operational energy forecasts.

Decision trees in the RF model use binary splits with thresholds of feature variables. Each node answers a yes/no question, with the objective of reducing variance within the node and increasing the accuracy of predictions. An illustrative example of this splitting mechanism is shown in Figure 2.

2.7 | LightGBM

LightGBM is a gradient-boosted decision tree algorithm optimised for speed and scalability [2], sharing architectural similarities with XGBoost [16] but employing histogram-based splitting for improved efficiency. The model iteratively fits trees to the negative gradients of the loss function, allowing it to capture complex nonlinear relationships in large-scale high-dimensional datasets. It minimises the following regularised objective:

$$F_m(x) = F_{m-1}(x) + \eta \cdot h_m(x) \tag{3}$$

where $h_m(x)$ is the m-th regression tree and η denotes the learning rate. Loss minimisation is performed using histogram-based leaf-wise tree growth, which allows LightGBM to accelerate training by grouping continuous features into discrete bins, thereby improving both memory efficiency and speed.

2.7.1 | Boosted Trees Objective

The overall objective is to minimise:

$$L = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(T_k) \tag{4}$$

where $l(y_i, \hat{y}_i)$ is the prediction error and $\Omega(T_k)$ is a regularisation term penalising model complexity.

2.7.2 | Tree Splitting Criterion

$$\text{Split}(S) = \arg\max_{\text{threshold}} \text{gain}(S_{\text{left}}, S_{\text{right}})$$

where gain denotes the information gain and $S_{\text{left}}, S_{\text{right}}$ are data partitions from a candidate split.

For this research, LightGBM served a dual purpose: first, as a competitive baseline for machine learning-based electricity demand forecasting and, second, as an embedded feature selector due to its capability to rank variable importance. The histogram-based method and the leaf-wise splitting make it highly efficient in modelling the consumption pattern with time-related features and weather indicators. In addition, this model supports fast inference and scalability, which would ease practical deployment in high-frequency forecasting tasks.

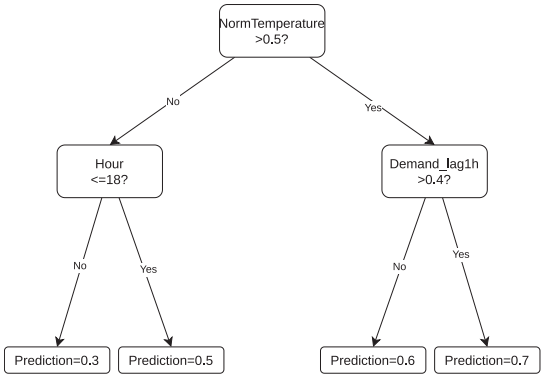


FIGURE 2 | Illustrative example of decision tree splits in Random Forest.

2.8 | Ensemble Approach (RF + LightGBM)

Although both random forest and LightGBM are tree-based learners, their inductive biases and sources of predictive strength are quite different. Random forest is a variance-reducing technique that uses bagging to average a forest of decorrelated trees. The stability emerges when their predictions are averaged across trees, even in feature spaces characterised by noise or irregularities. LightGBM, on the other hand, is a gradient boosting algorithm wherein trees are built in a sequential manner to correct the residual errors, thereby capturing detailed nonlinear interactions and structured deviations that, for example, bagging might underfit. In other

words, the strategy of error reduction in random forest—variance suppression—and, in contrast, the methodology of error reduction in LightGBM—bias refinement—sets a methodological basis for the two models' combination.

To retain interpretability and avoid introducing hyperparameters, we adopt a simple unweighted averaging ensemble:

$$\hat{y}_{\text{ens}} = \frac{1}{2}\left(\hat{y}_{\text{RF}} + \hat{y}_{\text{LGBM}}\right).$$

Preliminary experiments regarding weighted averaging and stacking yielded negligible improvements relative to the added complexity and would have required second-stage learners or additional cross-validation procedures incompatible with the unified benchmarking design. Therefore, simple averaging provides an intelligible way to combine the two models' complementary inductive biases without modifying the evaluation protocol.

It is not the ensemble developed herein to present a novel forecasting method but rather a diagnostic tool used to consider whether bagging-based stability combined with boosting-based residual correction leads to improved robustness across validation and test splits. This would relate to earlier work in energy forecasting, which also found that mixed ensemble methods do reduce downside risk and increase robustness. For instance, Shoko et al. [17], Makatjane and Mmelesi [18] and Makatjane and Shoko [19] highlight that combining distinct learning paradigms can improve accuracy by smoothing error distributions. Similarly, recent studies on hybrid frameworks [20] suggest that ensemble approaches are particularly advantageous when base learners exhibit complementary error profiles.

2.9 | Long Short-Term Memory

The long short-term memory architecture [4] extended the conventional recurrent neural network by integrating an explicit memory cell and several gating mechanisms that control information flow concerning how the information is stored, updated and shown when required through time. Unlike in traditional ones, which overwrite the hidden state at every time step, the LSTM has a cell state that is distinct from the hidden state and can hold information over long periods. The evolution of the cell state is controlled by three interacting gates: a forget gate determines what information from the past is to be kept, an input gate selects what newly encoded information is to be stored and an output gate reveals relevant parts of the updated memory to the subsequent layer. Thanks to this internal gating structure, the model captures extended dependencies in time more easily than conventional RNNs; thus, LSTMs may hold particular relevance in forecasting short-term loads that involve complex interactions of daily and weekly periodicities.

At each time step t, the LSTM updates are defined as follows:

$$f_t = \sigma\left(W_f[h_{t-1}, x_t] + b_f\right) \quad \text{(Forget gate)} \tag{5}$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad \text{(Input gate)} \tag{6}$$

$$\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \quad \text{(Candidate cell state)} \tag{7}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \quad \text{(Cell state update)} \tag{8}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad \text{(Output gate)} \tag{9}$$

$$h_t = o_t \odot \tanh(C_t) \quad \text{(Updated hidden state)} \tag{10}$$

These gating operations jointly determine how information flows through the sequence, allowing the LSTM to represent both short-term fluctuations and long-term trends in electricity demand. This suggests that the LSTM-type architectures are able to represent periodicity, as in daily load cycles, or low-frequency variations caused by weather and seasonal influences. Extensions of RNN-based architectures to electricity price forecasting in markets with high renewable penetration [21] further demonstrate the versatility of recurrent models for energy system applications.

The current paper compares two LSTM architectures to advance knowledge about how the forecast behaviour varies with different temporal horizons. The first model uses a 24-h lookback window, thus representing a day-ahead configuration, whereas the second uses a 168-h window in which weekly dynamics are captured. Both architectures were coded in Python using the PyTorch framework, allowing for fast GPU training and easy revision of the model with different sequence lengths and depths. Recent applications of deep LSTM networks to electricity consumption forecasting [22] have demonstrated the importance of multiscale temporal modelling for capturing daily and weekly demand patterns. Hybrid architectures combining recurrent and convolutional components [23] have shown promise in modelling both short-term and long-term temporal patterns, though we focus on pure LSTM architectures in this study.

2.10 | Transformer

The transformer architecture [5] utilises an attention mechanism rather than recurrent computation to model sequential data. Since each time step can directly attend to all others, it can catch global temporal dependencies in one layer and hence does not introduce the bottlenecks due to iterative recurrence. Thanks to this parallel structure, efficient learning can be achieved over long input sequences, thus making transformers ideally suited for short-term load forecasting, where recent demand, calendar and weather features interact across several temporal scales. The central part is the scaled dot-product attention mechanism defined as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{11}$$

where Q, K and V represent the query, key and value matrices derived from input embeddings and d_k is the dimensionality of the keys.

To better capture temporal dependencies and improve representation learning, we utilise a multihead attention mechanism:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, ..., head_h)W^O \quad (12)$$

$$head_i = \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right) \quad (13)$$

where W_i^Q, W_i^K W_i^V are learnt projections and W^O is the output projection matrix.

Temporal order is preserved using sinusoidal positional encoding defined as follows:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \quad (14)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \quad (15)$$

where *pos* denotes the position index and d_{model} is the model's hidden size.

2.10.1 | Model Architecture and Configuration

To adapt the transformer to electricity demand forecasting, we design a lightweight encoder-only architecture with the following configuration:

- One encoder layer
- Two attention heads
- Hidden dimension size: 32
- Dropout: 0.1

The model projects the input features to a hidden dimension, applies positional encoding and passes them through the transformer encoder over the temporal dimension. The final output is generated via a fully connected decoder at the last time step. Our simplified model reduces layer depth and number of heads to improve memory efficiency [5]. Transformer-based models have been successfully applied to electricity demand prediction tasks [24], leveraging attention mechanisms to capture complex temporal dependencies. Extensions, such as Informer [25], address long sequence time-series forecasting through sparse attention mechanisms, though such modifications are beyond the scope of this benchmark. Similarly, Autoformer [26] incorporates decomposition with auto-correlation for long-term series forecasting. Recent work on efficient token representations [27] explores further optimisations for time series transformers.

2.11 | Temporal Fusion Transformer (TFT)

The Temporal Fusion Transformer (TFT) [6] is an attention-driven architecture designed specifically for multihorizon time series forecasting with built-in interpretability. By combining recurrent encoders with gating mechanisms and a multi-head attention layer, TFT integrates both short-term temporal patterns and long-range contextual information, whilst simultaneously providing transparent variable-level and time-level importance estimates.

The scaled dot-product attention used in the attention block is computed as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \quad (16)$$

where Q, K and V are the query, key and value matrices derived from input embeddings and d_k is the dimensionality of the keys.

2.11.1 | Implementation Details

Our implementation of TFT is closely aligned with the original specification by Lim et al., adapted for univariate hourly electricity forecasting. The model uses:

- `hidden_size = 64`, `dropout = 0.1` and `num_heads = 4`
- `lstm_layers = 1` and `quantiles = [0.1, 0.5, 0.9]`
- `embedding_dimension = 16` for both static and temporal inputs
- `features: NormSmoothedDemand`, `Temperature`, `Hour` and `Holiday`

The model is trained for 50 epochs using the Adam optimiser with MSE loss, on sequences of length 10 and a batch size of 500. This design balances interpretability, robustness and memory efficiency.

Indeed, based on a modular design, dynamic variable selection and interpretable attention mechanisms, TFT fits both long- and short-term dependencies. Under MV input scenarios, it reveals excellent forecasting performance; therefore, it should be seen as a good complement in our pipeline for forecasting electricity demand.

2.12 | Metrics and Reporting

All error-based metrics are computed on the min–max normalised target using statistics only from the training split, ensuring comparability across all models, avoiding scale-induced distortions and providing a consistent evaluation scale for the fair benchmark. This would ensure no metric inflation due to varied pipelines or feature treatments.

Following best practices for forecast accuracy measurement [28], we report four standard forecasting metrics:

- Mean absolute error (MAE),
- Root mean squared error (RMSE),
- Coefficient of determination (R^2),
- Pearson correlation.

MAE and RMSE quantify magnitude of residuals on the normalised [0, 1] scale, whereas R^2 and Pearson correlation measure shape fidelity and are invariant under affine scaling. This

separation clarifies whether differences arise from amplitude errors or trend-tracking behaviour.

Unified reporting policy. All benchmark metrics are computed using identical postprocessing and aggregation procedures, ensuring that cross-model differences reflect modelling behaviour rather than inconsistencies in normalisation or rescaling. For qualitative assessment, forecast plots are displayed in the original MW units to preserve interpretability, but these plots are not used for metric comparison.

This unified reporting framework enforces methodological transparency and aligns with recent recommendations for fair model comparison in load forecasting benchmarks.

2.13 | Reproducibility Details

Tables 3 and 4 report the complete hyperparameters and optimisation settings for all models. Random seeds, software versions and hardware are listed below. These artefacts, together with code/config files and requirements, enable exact replication.

Environment. Tested environment—Python 3.10.11; PyTorch 2.6.0 + cpu (CPU only) and R 4.5.0 (RStudio 2024.12.1 + 563). Compatible range—Python 3.10–3.11. Random seed(s): 123. We provide lockfiles (pip/conda and `renv`) and scripts for exact replication.

3 | Results and Analysis

This section examines the performance of all models under the unified tabular preprocessing pipeline, followed by an analysis of the LSTM architecture under the sequence-aware pipeline. Beyond reporting numerical differences, we discuss the methodological and practical implications of the findings, linking model behaviour to inductive biases, error decomposition and operational forecasting needs.

3.1 | Benchmark Under a Common R Pipeline

Table 5 reports a like-for-like benchmark in which all models are trained and evaluated under the same R-based preprocessing pipeline defined in Sections 2.3 and 2.12. Metrics are computed on the normalised [0, 1] scale using train-only statistics to avoid leakage. Performance differences observed under this setting therefore reflect how different model architectures respond to an identical tabular representation rather than differences in feature engineering, scaling or data partitioning.

Under this tabular-oriented pipeline, tree-based learners exhibit strong predictive accuracy. This behaviour is consistent with

TABLE 3 | Hyperparameters and optimisation for tabular models.

Model	Main HPs	Optim.	Notes	Seed
RF	trees = 200; mtry = min($\lfloor\sqrt{p}\rfloor$, 12); nodesize = 10; maxnodes = 1500 and importance = TRUE	Bagging	Bootstrap and OOB	123
LightGBM	objective = `regression`; metric = `rmse`; boosting = `gbdt`; learning_rate = 0.05; num_leaves = 15; max_depth = 5; feature_fraction = 0.8; bagging_fraction = 0.8 and nrounds = 100	GBDT	Histogram and leaf-wise	123
Ensemble	Average of RF + LGBM (weights 1/2, 1/2)	—	Fixed weights 1/2	—

TABLE 4 | Hyperparameters and optimisation for sequence models.

Model	L	Hidden/Heads	Layers (lyrs)	Batch	LR	Epochs
LSTM	24/168	128/–	2	64	10^{-3}	50
Transformer	10	32/2	1	500	10^{-3}	50
TFT	24	64/4	1	32	10^{-3}	30

TABLE 5 | Fair benchmark under a common R pipeline (normalised scale).

Model (same R pipeline)	RMSE	MAE	R^2	Pearson
Random Forest	0.030	0.021	0.983	0.992
LightGBM	0.018	0.012	0.994	0.997
Ensemble (RF + LightGBM)	0.022	0.015	0.991	0.996
LSTM	0.140	0.086	0.626	0.791
Transformer	0.111	0.069	0.759	0.872
TFT	0.164	0.096	0.757	0.880

TABLE 6 | Within-architecture comparison for LSTM (normalised scale).

LSTM variant (same architecture)	RMSE	MAE	R^2	Pearson
LSTM (baseline, R pipeline)	0.140	0.086	0.626	0.791
LSTM (sequence-aware, Python pipeline)	0.002	0.002	0.970	0.987

their inductive biases: decision-tree models are naturally suited to feature-engineered inputs, such as lagged demand, rolling statistics and calendar covariates, and they remain robust when temporal structure is encoded implicitly through static features. LightGBM, in particular, benefits from gradient boosting, which refines nonlinear interactions among engineered predictors and yields lower residual error under this representation.

In contrast, deep sequence models—including LSTM, transformer and TFT—show substantially degraded performance when trained on the same tabular inputs. Without access to temporally contiguous input sequences, these architectures cannot fully exploit recurrent or attention-based mechanisms that constitute their primary modelling advantage. The resulting performance gap therefore reflects a mismatch between the input representation and the temporal assumptions of sequence models rather than an inherent limitation of the architectures themselves.

The results in Table 5 motivate the subsequent within-architecture analysis. By holding the model architecture fixed and modifying only the preprocessing strategy, we examine how aligning input representation with temporal inductive bias affects deep sequence model behaviour. The tabular benchmark reported here serves as a controlled reference point, establishing a baseline against which representation-sensitive effects can be interpreted without confounding from heterogeneous preprocessing.

3.2 | Within-Architecture Effects: LSTM (R vs. Python)

This subsection provides a controlled within-architecture illustration of how input representation affects deep sequence models. By holding the LSTM architecture and optimisation settings fixed and modifying only the preprocessing strategy, we isolate the impact of temporal alignment on model behaviour.

Under the shared R-based tabular pipeline, the LSTM receives flattened feature representations in which temporal continuity is encoded implicitly through lagged variables. Although such features capture short-term dependence, they treat lagged observations as independent inputs, preventing the model from exploiting its recurrent inductive bias. As a result, gradient propagation across time steps is limited, and the network behaves similarly to a shallow feedforward model, failing to learn richer autoregressive structure or multiscale temporal dependencies.

In contrast, under the Python sequence-aware pipeline, the same LSTM architecture is trained on fixed-length temporal windows with aligned past–future relationships, per-feature

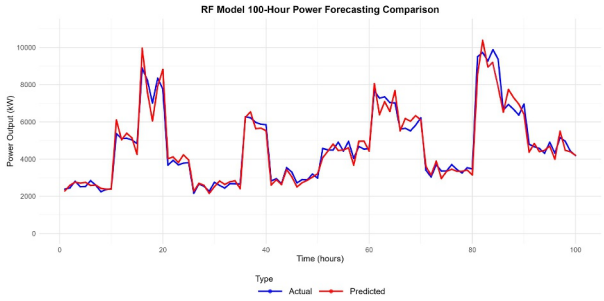


FIGURE 3 | Prediction versus actual—random forest.

temporal scaling and fully tensorised input sequences. This representation restores the recurrence dynamics the model is designed to exploit: gradients propagate across contiguous time steps, hidden states accumulate information over time and temporal patterns, such as diurnal cycles and load ramps, become explicitly learnable. The resulting reductions in RMSE and MAE therefore reflect alignment between input representation and architectural assumptions, rather than changes in model capacity or optimisation.

A quantitative comparison of the two LSTM variants is provided in Table 6. These results illustrate a methodological principle rather than a cross-model comparison: for deep sequence architectures, preprocessing and model design are tightly coupled and cannot be treated as independent choices. Misalignment between input representation and temporal inductive bias can severely suppress performance, whereas appropriate alignment recovers the expressive capacity of the architecture.

Accordingly, cross-family comparisons in this study are conducted exclusively under the unified tabular pipeline (Table 5), while architecture-sensitive effects are examined through controlled within-architecture analyses such as the LSTM experiment reported here. From a practical perspective, the findings underscore that the reliability of deep sequence models in short-term load forecasting depends critically on preprocessing design, particularly in operational settings where temporal structure may be simplified or distorted.

3.3 | Visual Comparison of Model Predictions

Figure 3 shows the prediction versus actual load for the random forest model, illustrating its strong short-term predictive performance under stable load conditions However, predictions look less adaptive and are generally much jagged when offers are set due to the partitioning mechanism of decision trees. This reflects the limited variability of the model concerning temporal continuity.

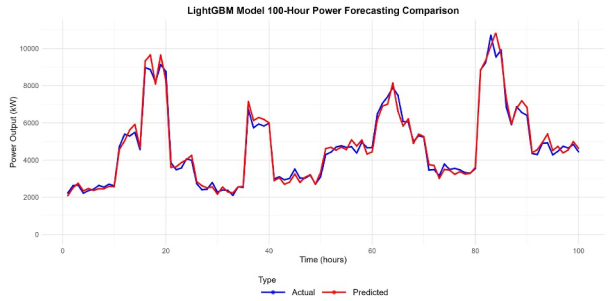


FIGURE 4 | Prediction versus actual—LightGBM.

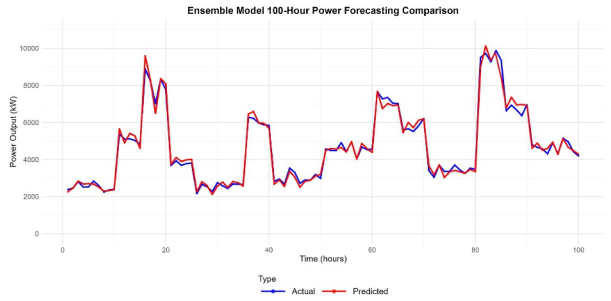


FIGURE 5 | Prediction versus actual—ensemble.

As illustrated in Figure 4, the LightGBM predictions come out closer to the actual values of electricity demand. The gradient-boosting framework incorporates better the nonlinearity of interactions and calendar features, which smoothens the prediction and improves the seasonality fitting.

As shown in Figure 5, the ensemble model balances robustness and accuracy by reducing variance relative to individual models. This is where an ensemble model consisting of random forests and LightGBM strikes a balance between robustness and accuracy, significantly reducing the variance from the single models' output and improving their generalisation ability. Of course, there is some peak smoothing during these periods, particularly relating to high demand.

The LSTM prediction behaviour under the shared tabular pipeline is illustrated in Figure 6. Deep learning models, such as LSTM, transformer and TFT, present distinct prediction behaviours within a common tabular pipeline. The LSTM model captures the main demand structure and daily/weekly periodicity, but in high-variance regions—such as sharp peaks and troughs—it tends to lag and underestimate extremes, reflecting the mismatch between its sequential inductive bias and the tabular feature representation. The transformer model's prediction behaviour under the common tabular pipeline is shown in Figure 7. The transformer produces smoother trajectories and aligns well with actual values during stable periods, but still undersmooths some extreme spikes, suggesting that the shallow encoder used here prioritises global trends over very short-term jumps. Figure 8 illustrates the prediction versus actual load for the temporal fusion transformer. TFT shows the highest flexibility among the deep models in tracking sudden changes whilst preserving overall trends, though slight over and undershoots remain around the sharpest peaks. A dedicated within-

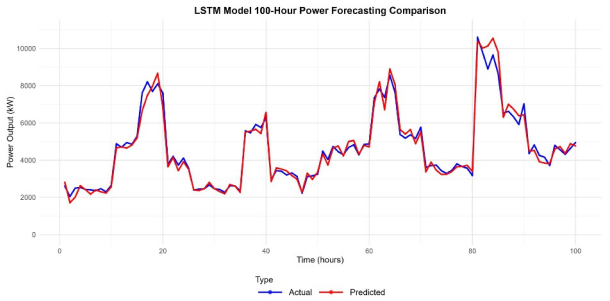


FIGURE 6 | Prediction versus actual—LSTM.

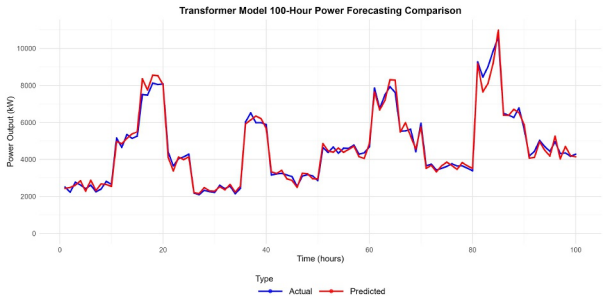


FIGURE 7 | Prediction versus actual—transformer.

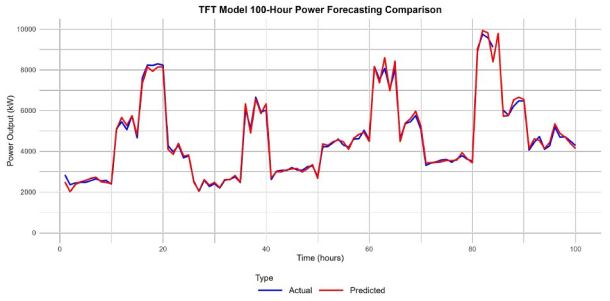
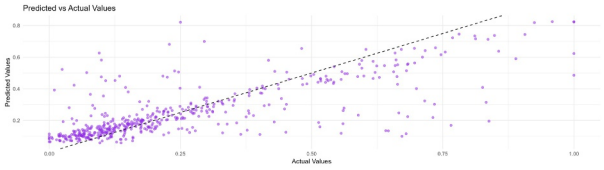


FIGURE 8 | Prediction versus actual—Temporal Fusion Trans former (TFT).

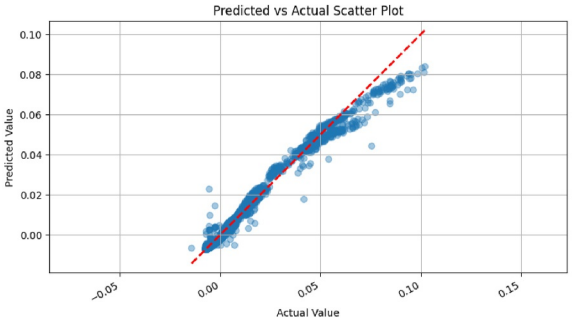
architecture analysis of LSTM under a sequence-aware pipeline is reported separately in Section 3.4 and Figure 9.

The transformer model exhibits smooth curve transitions and aligns well with actual values under stable demand. However, it was not able to capture sharp peaks and sudden flatness here. It usually delivers oversmooth outputs. The attention mechanism indicates a tendency to diffuse the focus over time; it favours the global trend but may impair it when it requires an abrupt local change.

The TFT model delivers better performance in tracking sudden spikes and seasonal fluctuations. It adapts fast to localised surges whilst keeping the overall trend intact. On visualisation, there is a slight overestimation at peak demand points, perhaps attributable to attention weight instability or sensitivity to noise in lagged covariates. Despite this, TFT outperforms other deep learning models in capturing complex temporal structures and dynamic dependencies.



(a) LSTM (shared tabular pipeline)



(b) LSTM (sequence-aware pipeline)

FIGURE 9 | Predicted versus actual scatter plots for the same LSTM architecture under two preprocessing pipelines.

3.4 | LSTM Pipeline Sensitivity Analysis (Within-Architecture)

To isolate the effect of preprocessing on recurrent sequence models, we conduct a within-architecture comparison using the same LSTM network trained under two distinct data pipelines: (i) the shared tabular pipeline used for the benchmark and (ii) the sequence-aware pipeline designed for recurrent architectures. This experiment is deliberately separated from the benchmark results to avoid drawing cross-family conclusions.

When trained on the tabular features used for all benchmark models, the LSTM exhibits performance comparable to the tree-based models. However, its scatter plot in Figure 9a shows a broad dispersion around the diagonal and consistent underestimation during moderate and high-demand periods. This behaviour is expected: without explicit temporal alignment or sliding windows, the recurrent network receives inputs that do not match its inductive bias, limiting its ability to track short-term dynamics.

Using the same LSTM architecture but training it on sequentially aligned windows (Figure 9b), the scatter plot becomes substantially tighter. The model now captures both local temporal transitions and high-variance regions more reliably. Since all hyperparameters—including hidden size, number of layers, dropout and optimisation settings—remain unchanged, the performance gain derives solely from the architecture-aligned preprocessing rather than architectural modification.

The contrast between the two scatter plots illustrates a methodological lesson:

- Tree-based models (RF and LightGBM) are largely insensitive to temporal representation details as long as lag features are constructed consistently.

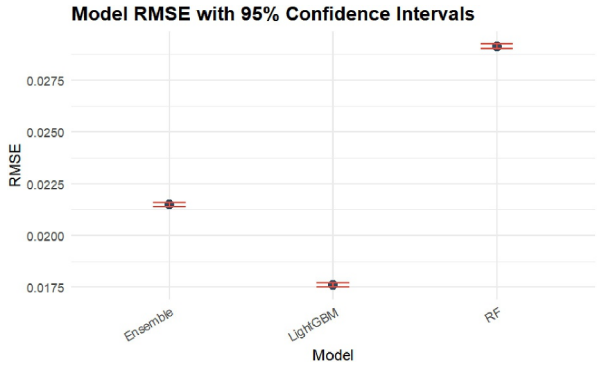


FIGURE 10 | RMSE with 95% confidence intervals for machine learning model.

- Sequence models, such as LSTM, require temporally aligned input windows; otherwise, their capacity to model short-term structure is severely constrained.

This experiment does not compare LSTM against transformer or TFT; instead, it isolates how preprocessing design interacts with recurrent architectures. The results highlight that model comparison across families must be done under a controlled pipeline, whereas architecture-specific preprocessing should be evaluated separately.

The LSTM pipeline comparison emphasises that deep recurrent models are highly sensitive to the temporal structure of input data, whereas tabular models are more robust to feature representation. This reinforces the need for pipeline transparency and justifies the strict separation between the benchmark analysis and the within-architecture study.

3.5 | Statistical Significance Testing

To assess whether the performance differences among the tree-based models in Table 5 are statistically meaningful, we test the following hypotheses on the 1-h-ahead RMSE under the shared tabular pipeline:

- H_0: There is no significant difference in average RMSE between random forest, LightGBM and ensemble.
- H_1: LightGBM achieves a significantly lower RMSE than random forest and ensemble.

H_1 is supported. The RMSE distributions with 95% confidence intervals are summarised in Figure 10. LightGBM consistently delivers the best predictive performance among machine learning models, with statistical significance in RMSE and strong feature interpretability.

3.6 | Feature Attribution via SHAP

To interpret the internal decision process of the LightGBM model, we employed SHAP (SHapley Additive exPlanations) [29] using R-based implementations. SHAP assigns each

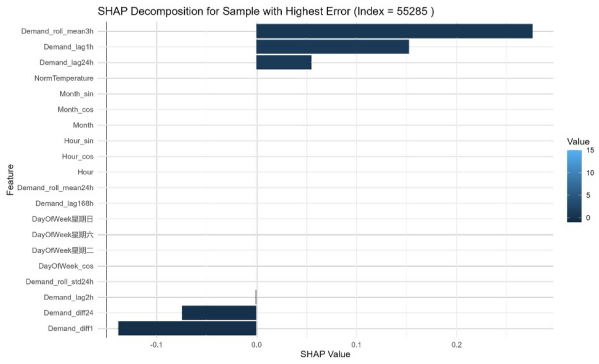


FIGURE 11 | Global SHAP summary plot for LightGBM.

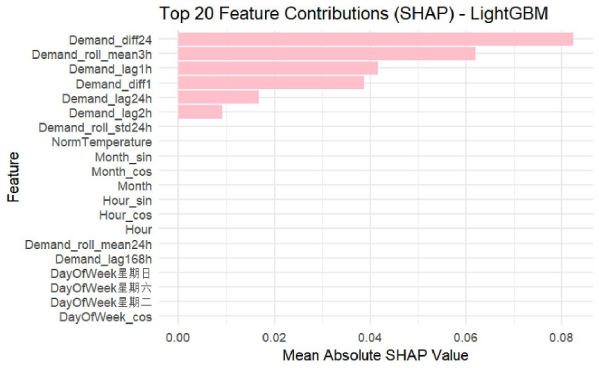


FIGURE 12 | Top 20 SHAP feature contributions for LightGBM.

prediction a set of feature attributions, helping to uncover which inputs drive the output in different contexts.

Figure 11 provides a global SHAP summary plot, revealing the marginal contribution of each feature across all predictions. It shows that short-term rolling averages (e.g., Demand_roll_mean3h), lagged demand values (e.g., Demand_lag1h and Demand_lag24h), and normalised temperature are the three most important features—these results are generally aligned with expectations.

Further, this is complemented by the top-20 ranked bar plot of mean absolute SHAP values in Figure 12. Both figures reveal that, when the temporal lag features and weather metrics are the primary drivers of model output, it has good interpretability and trust in the model's predictions.

4 | Discussion

The findings elaborately explain how the interaction among model architecture, preprocessing strategy and deployment context induces the shaping of forecasting performance across learning paradigms. Under a tabular preprocessing representation, classical models, such as LightGBM and random forest, show good short-term predictive performance with relatively little tuning. Their robustness emanates from an ensemble structure that handles redundancy and irregular feature distributions well. The earlier results are quite consistent with those observed in the earlier literature on load forecasting, denoting the stability of tree-based learners when it comes to feature-engineered representations.

On the other hand, the initial performance of deep learning models—LSTM and TFT—was more dependent on alignment through preprocessing than architectural capacity. Earlier experiments utilised the same feature-engineering pipeline, which was optimised for tabular learners; thus, they limited sequence models' ability to exploit temporal continuity. Indeed, further redesigning the LSTM preprocessing to explicitly preserve temporal structure using sequential windowing and scale-aware normalisation resulted in significant forecast accuracy improvement. That change illustrates the sensitivity of deep sequence models to input representation rather than indicating any shortcoming in the architectures themselves.

These highlights make it clear that model effectiveness relies on more than algorithmic complexity or the number of layers in the model: there must be coherence between architectural assumptions and data representation. A temporal model will need representations that keep order, scale consistency and contextual continuity. A tabular model will perform best with an explicit feature construction and aggregation. Therefore, preprocessing should be considered part of the model design rather than a separate implementation detail.

Further analysis of TFT exemplifies this issue. Attention-based architectures allow for flexible modelling of complex temporal dependencies, but performance can be brittle regarding scaling input, noise characteristics and feature stability. Again, this illustrates the trade-off between expressive power and robustness and highlights the importance of matching model design to data characteristics and preprocessing choices.

All results jointly present short-term load forecasting as a system-level codesign problem of model architecture, data representation and computational infrastructure. In industrial practice, tree-based ensembles give robust performances in tabular pipelines for day-ahead planning, whereas sequence-aligned deep models are preferable for capturing more dynamic temporal patterns when appropriate preprocessing is made possible. All this, of course, provides a natural motivation for hybrid or ensemble methods that exploit complementary inductive biases.

5 | Conclusion and Future Research

This work presents a transparent and reproducible benchmarking study of machine learning and deep learning methods for short-term electricity demand forecasting, including an understanding of the role of preprocessing in determining comparative performance.

Under a common preprocessing pipeline, RF, LightGBM, ensemble models, LSTM, transformer and TFT are benchmarked, and in this respect, a controlled reference point is given due to limited confounding from data handling and reporting choices. Herein, tree-based models represent an adequately stable performance under feature engineered tabular representations, whereas the architecture of deep learning remains highly sensitive to the representation of temporal information.

Inner-architecture investigation of LSTM shows, indeed, that alignment of input representation to temporal inductive bias alters predictive behaviour dramatically, thus giving evidence to claim the importance of architecture-aware preprocessing for time series forecasting.

Beyond the contribution of individual model results, this represents a more elaborate methodological contribution. The experiments show that preprocessing should not be viewed as a one-off straw-man preliminary, but rather as a live component of model design intimately involving architectural assumptions. Therefore, benchmarking results need to be appraised with an equivalent degree of care regarding representational constraints as numeric performance metrics.

Future research should consider:

- Extending the same sequence-aware preprocessing to transformer and TFT models would enable an architecture-consistent comparison under temporally aligned input representations.
- Quantile-aware or demand-weighted loss functions are incorporated to improve capturing peak behaviour and uncertainty. Probabilistic deep learning approaches, such as DeepAR [30], offer complementary frameworks for explicit uncertainty quantification in multihorizon forecasting.
- Hybrid stacking strategies are explored to combine the stability of machine learning models with the expressiveness of deep sequence models.
- Developing scalable low-latency deployment pipelines for real-time forecasting in utility applications.

This work not only provides a careful cross-model benchmarking effort but also contributes more broadly methodological advice on architecture-aware pipeline design, informing future studies and operational practitioners on how to make better use of traditional and neural forecasting approaches within contemporary energy systems.

Author Contributions

Lei Xing: collecting the data, performing the numerical simulations, discussion of the results, writing the manuscript. **Zohreh Kaheh:** conceptualisation, methodology, validation, visualisation, writing – review and editing, software, formal analysis, project administration, supervision.

Funding

The authors have nothing to report.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

1. L. Breiman, "Random Forests," *Machine Learning* 45, no. 1 (2001): 5–32, https://doi.org/10.1023/a:1010933404324.

2. G. Ke, Q. Meng, T. Finley, et al., "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)* (Curran Associates, Inc., 2017), 3146–3154.

3. J. L. Elman, "Finding Structure in Time," *Cognitive Science* 14, no. 2 (1990): 179–211, https://doi.org/10.1207/s15516709cog1402_1.

4. S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation* 9, no. 8 (1997): 1735–1780, https://doi.org/10.1162/neco.1997.9.8.1735.

5. A. Vaswani, N. Shazeer, N. Parmar, et al., "Attention Is All You Need," in *Advances in Neural Information Processing Systems (NIPS)* (Curran Associates, Inc., 2017), 5998–6008.

6. B. Lim, S. Ö. Arik, N. Loeff, and T. Pfister, "Temporal Fusion Transformers for Interpretable Multi-Horizon Time Series Forecasting," *International Journal of Forecasting* 37, no. 4 (2021): 1748–1764, https://doi.org/10.1016/j.ijforecast.2021.03.012.

7. Y. Li, Z. Li, J. Qin, and T.-Y. Liu, "COLTR: Semi-Supervised Learning to Rank With Co-Training and Over-Parameterization for Web Search," *IEEE Transactions on Knowledge and Data Engineering* 35, no. 11 (2023): 10902–10915, https://xhyccc.github.io/12.pdf.

8. Y. Li, Z. Li, J. Qin, and T.-Y. Liu, "MPGraf: A Modular and Pre-Trained Graphformer for Learning to Rank at Web Scale," in *Proceedings of IEEE International Conference on Data Mining (ICDM)* (EEE, 2023), 415–424.

9. T. Hong and S. Fan, "Probabilistic Electric Load Forecasting: A Tutorial Review," *International Journal of Forecasting* 32, no. 3 (2016): 914–938, https://doi.org/10.1016/j.ijforecast.2015.11.011.

10. R. Weron, "Electricity Price Forecasting: A Review of the State-of-the-Art With a Look Into the Future," *International Journal of Forecasting* 30, no. 4 (2014): 1030–1081, https://doi.org/10.1016/j.ijforecast.2014.08.008.

11. J. Lago, F. De Ridder, and B. De Schutter, "Forecasting Day-Ahead Electricity Prices: A Review of State-of-the-Art Algorithms, Best Practices and an Open-Access Benchmark," *Applied Energy* 232 (2018): 312–333, https://doi.org/10.1016/j.apenergy.2021.116983.

12. S. B. Taieb, R. J. Hyndman, F. De Silva, and K. L. Kang, "Forecasting Competition in Electricity Load: The GEFCom2014 Benchmark," *International Journal of Forecasting* 32, no. 3 (2016): 1011–1021, https://doi.org/10.1016/j.ijforecast.2016.02.001.

13. Z. Kaheh and M. Shabanzadeh, "The Effect of Driver Variables on the Estimation of Bivariate Probability Density of Peak Loads in Long-Term Horizon," *Journal of Big Data* 8, no. 15 (2021): 15, https://doi.org/10.1186/s40537-020-00404-8.

14. C. Bergmeir and J. M. Benítez, "On the Use of Cross-Validation for Time Series Predictor Evaluation," *Information Sciences* 191 (2012): 192–213, https://doi.org/10.1016/j.ins.2011.12.028.

15. M. Christ, A. Braun, J. Neuffer, and A. W. Kempa-Liehr, "Time Series Feature Extraction on Basis of Scalable Hypothesis Tests (Tsfresh)," *Neurocomputing* 307 (2018): 72–77, https://doi.org/10.1016/j.neucom.2018.03.067.

16. T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)* (ACM, 2016), 785–794.

17. C. Shoko, C. Sigauke, and K. Makatjane, "An Application of Ensemble Stacking in Machine Learning to Predict Short-Term Electricity Demand in South Africa," *Statistics, Optimisation & Information Computing* 13, no. 6 (2025): 2412–2433, https://doi.org/10.19139/soic-2310-5070-2170.

18. K. Makatjane and K. Mmelesi, "An Improved Model Accuracy for Forecasting Risk Measures: Application of Ensemble Methods," *Journal*

of Applied Economics 27, no. 1 (2024): 2395775, https://doi.org/10.1080/15140326.2024.2395775.

19. K. Makatjane and C. Shoko, "A Deep Learning Forecasting of Downside Risk: Application of a Combined ESRNN-VAE," *Frontiers in Applied Mathematics and Statistics* 11 (2025): 1662252, https://doi.org/10.3389/fams.2025.1662252.

20. M. AlKandari and I. Ahmad, "Solar Power Generation Forecasting Using an Ensemble Approach Based on Deep Learning and Statistical Methods," *Applied Computing and Informatics* 20, no. 3/4 (2024): 231–250, https://doi.org/10.1016/j.aci.2019.11.002.

21. R. Kock, Z. Kaheh, and M. Shafie-Khah, "Developing Two RNN-Based Algorithms for Electricity Price Forecasting in Markets With High Penetration of Renewable Energy Resources," in *Proceedings of 2024 International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM)* (IEEE, 2024).

22. J. F. Torres, F. Martínez-Álvarez, and A. Troncoso, "A Deep LSTM Network for the Spanish Electricity Consumption Forecasting," *Neural Computing and Applications* 34, no. 13 (2022): 10533–10545, https://doi.org/10.1007/s00521-021-06773-2.

23. G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling Long- and Short-Term Temporal Patterns With Deep Neural Networks," in *Proceedings of 41st International ACM SIGIR Conference on Research and Development in Information Retrieval* (ACM, 2018), 95–104.

24. A. M. Mahmood, M. M. A. Zahra, W. Hamed, et al., "Electricity Demand Prediction by a Transformer-Based Model," *Majlesi Journal of Electrical Engineering* 16, no. 4 (2022): 97–102, https://oiccpress.com/mjee/article/view/4974.

25. H. Zhou, S. Zhang, J. Peng, et al., "Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting," in *Proceedings of AAAI Conference on Artificial Intelligence* (AAAI Press, 2021), 11106–11115.

26. H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition Transformers With Auto-Correlation for Long-Term Series Forecasting," in *Advances in Neural Information Processing Systems* (Curran Associates, Inc., 2021), 22419–22430.

27. Y. Nie, N. Nguyen, and J. P. Time, "A Time Series Is Worth 64 Words: Long-Term Forecasting With Transformers," *arXiv preprint arXiv:2211.14730* (2023), https://ar5iv.labs.arxiv.org/html/2211.14730.

28. R. J. Hyndman and A. B. Koehler, "Another Look at Measures of Forecast Accuracy," *International Journal of Forecasting* 22, no. 4 (2006): 679–688, https://doi.org/10.1016/j.ijforecast.2006.03.001.

29. S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in *Advances in Neural Information Processing Systems* (Curran Associates, Inc., 2017), 4765–4774.

30. D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "DeepAR: Probabilistic Forecasting With Autoregressive Recurrent Networks," *International Journal of Forecasting* 36, no. 3 (2020): 1181–1191, https://doi.org/10.1016/j.ijforecast.2019.07.001.