Comparative Analysis of Electricity Load Forecasting Methods: Supervised Learning vs. Similarity-Based Approaches

Rina Razafimahefa

January 2026

Abstract

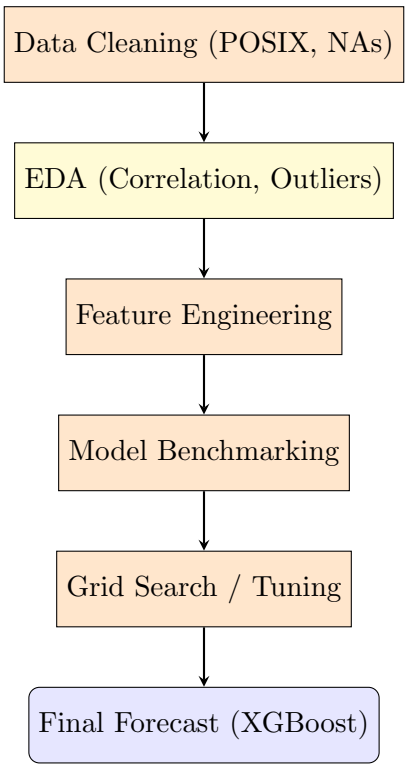
This report details the implementation of two different forecasting strategies for 15-minute interval electricity load data. First, an XGBoost model was optimized, achieving a validation RMSE of 10.71. Second, a custom R package, `WNNRina`, was developed to implement a Weighted Nearest Neighbors (WNN) algorithm. The results show that while WNN (RMSE 20.23) provides a robust non-parametric baseline, the inclusion of exogenous variables in XGBoost is essential for high-precision forecasting.

1 Part 1: Supervised Learning & Model Selection

The primary objective of this stage was to develop a high-precision forecasting engine for 15-minute interval electricity load. The methodology followed a systematic pipeline of data cleaning, exploratory analysis, and model benchmarking.

1.1 Forecasting Pipeline

The following flowchart summarizes the technical process adopted to move from raw data to the final 24-hour forecast:



1.2 Data Challenges and Pre-processing

Significant effort was invested in data sanitation. The initial dataset contained inconsistent date formats, requiring a robust parsing logic (see Appendix C) to unify all entries into a standard POSIXct format.

Furthermore, the dataset exhibited several "zero power" episodes. Given the building context and stable temperature readings during these periods, they were identified as sensor failures or grid outages. These points were treated as missing values and handled via linear interpolation to maintain signal continuity for autoregressive modeling.

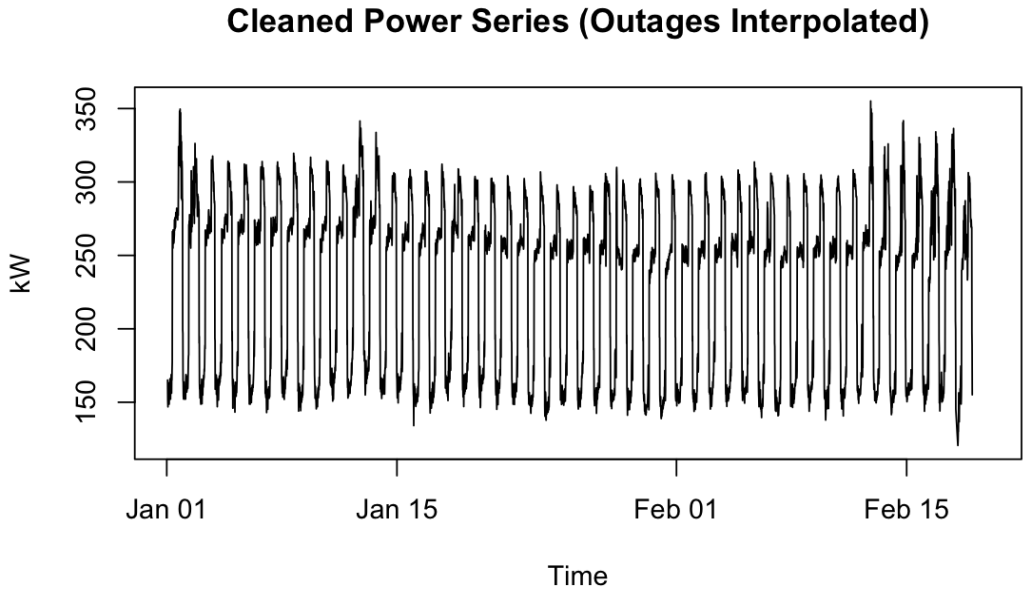


Figure 1: Full consumption series after interpolation of sensor outages.

1.3 Exploratory Data Analysis (EDA)

A deep dive into the dataset revealed that while a correlation exists between outdoor temperature and power demand (see Figure 3), it is not the primary driver in the short term. The building exhibits thermal inertia, making temporal indices more influential for 15-minute granularity. Analysis of the daily load profile confirmed a strong dual-peak structure corresponding to occupancy schedules.

1.4 Feature Engineering

Based on the EDA, the following features were engineered to capture the building's dynamics:

- **Calendar Variables:** Hour of the day and Day of the week (`wday`) to capture human activity.
- **Dynamic Trend:** A linear `time_index` to account for long-term drifts.
- **Exogenous Driver:** External temperature.
- **Autoregressive Lags:** Values of the target variable (Y_{t-k}) to provide the most recent baseline.

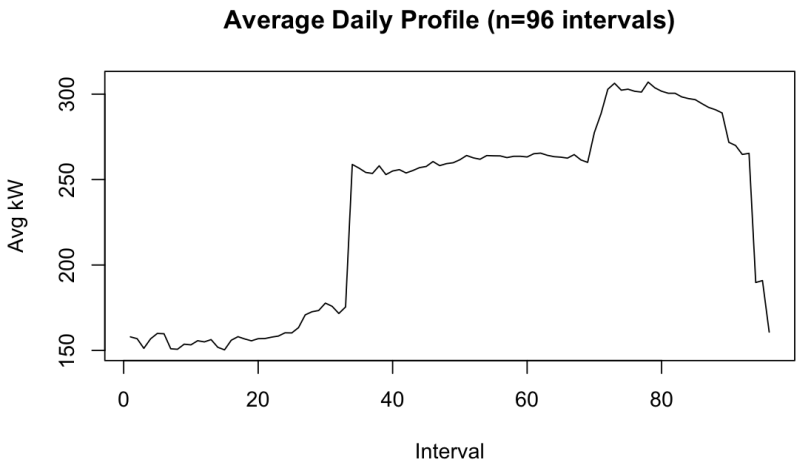


Figure 2: Aggregated daily load profile highlighting standard occupancy cycles.

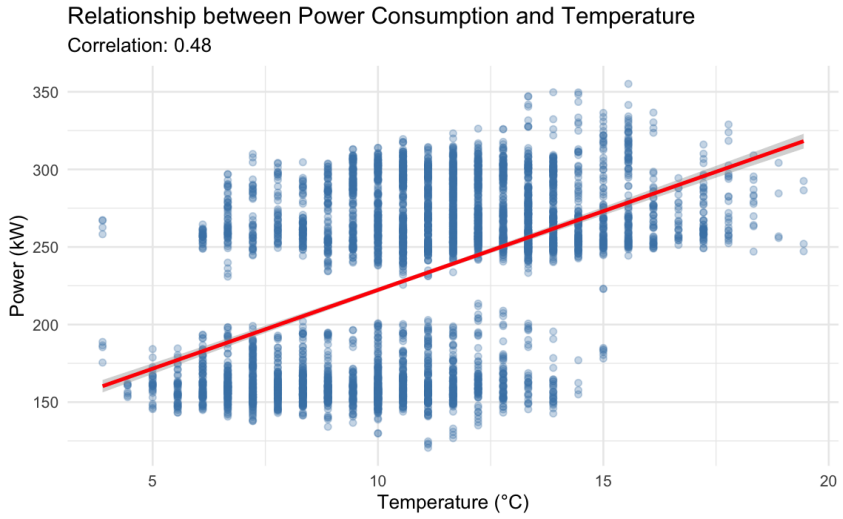


Figure 3: Power demand vs. outdoor temperature showing high variance at specific points.

1.5 Benchmark and Validation Strategy

Models were evaluated using a **Rolling Forecasting Origin** (Time Series Cross-Validation). This approach ensures that the RMSE (Table 1) reflects the model's stability across multiple 24-hour windows rather than a single test split. XGBoost consistently outperformed other architectures across all folds. Exhaustive results are provided in Appendix A and B.

1.6 Final Model Choice: XGBoost

XGBoost was selected for its superior ability to model non-linear relationships. Feature importance analysis (Figure 4) reveals:

- **time_index & wday:** Dominant predictors, reflecting daily/weekly occupancy routines.
- **temperature:** Secondary predictor; its impact is smoothed by building inertia.

This model was used to generate the final forecast provided in `RinaRazafimahefa.xlsx`.

Table 1: Top 4 Forecasting Models (Summary)

Model	RMSE	Status
XGBoost	**10.71**	**Champion**
NNAR	10.92	Runner-up
Random Forest	11.80	Top Tier
Prophet	12.94	Baseline

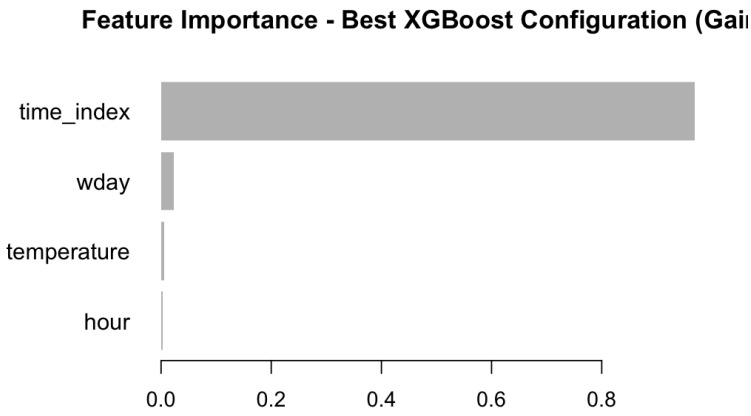


Figure 4: XGBoost Feature Importance (Gain).

2 Part 2: Similarity-Based Forecasting via WNN

The second phase involved implementing the Weighted Nearest Neighbors (WNN) algorithm, encapsulated in the `WNNRina` R package.

2.1 Algorithmic Implementation

The core engine, `predict_wnn`, extracts the most recent 96 observations as a query pattern and scans the history for the k most similar windows using Euclidean distance. We implemented a squared inverse distance weighting ($1/d^2$) to prioritize closer historical patterns.

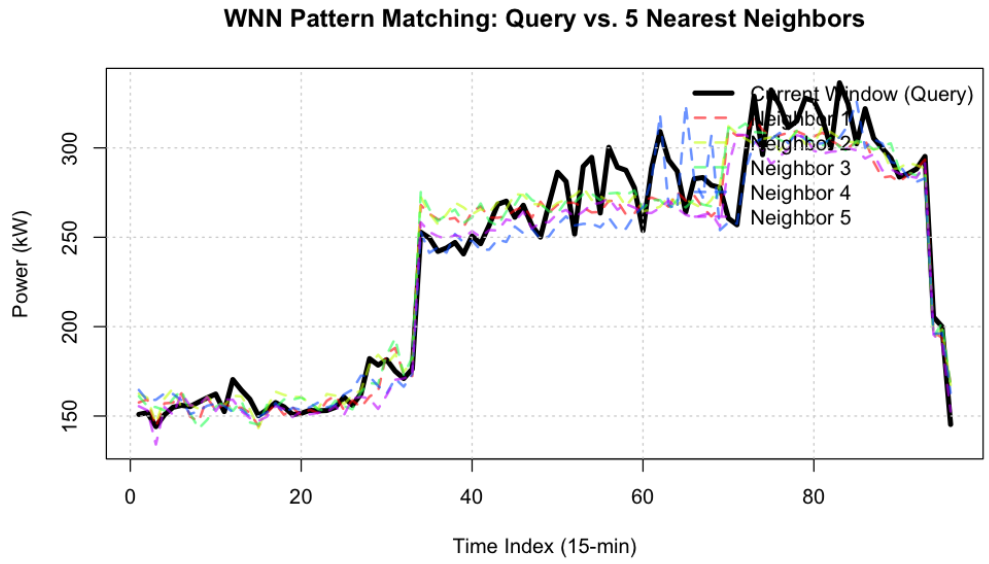


Figure 5: WNN Similarity Matching: Current Query vs. Top 5 Historical Neighbors.

2.2 Software Engineering and Comparison

The package follows professional standards (`Roxygen2` documentation, HTML vignette). While WNN achieved an RMSE of 20.23, it serves as a robust non-parametric baseline that captures the typical daily profile without intensive training.

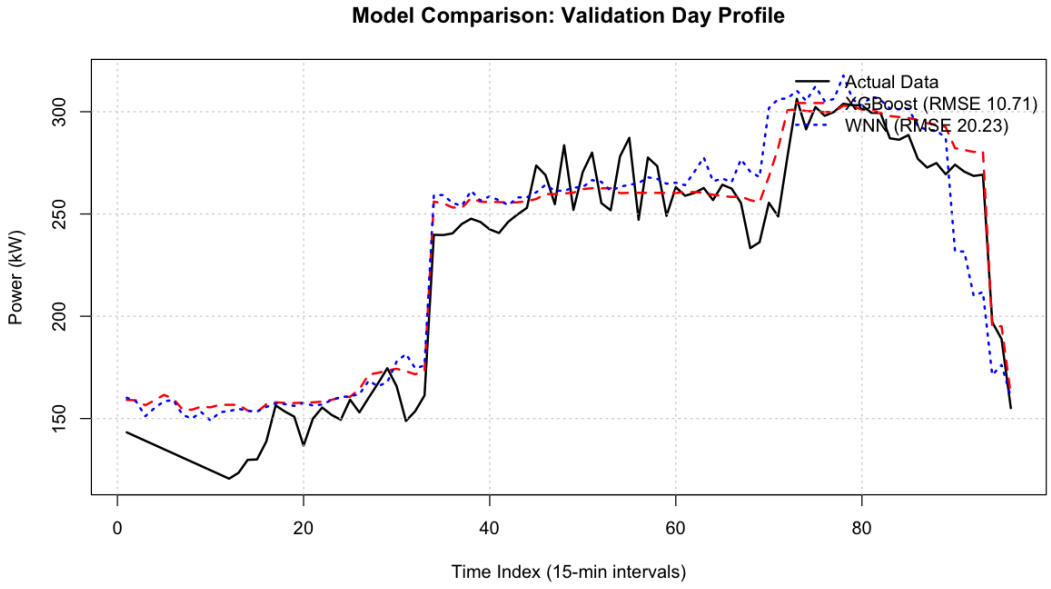


Figure 6: Final Model Comparison: XGBoost (red) vs. WNN (blue) vs. Actuals (black).

As shown in Figure 6, XGBoost tracks the actual load more precisely by leveraging temperature data, while WNN provides a smoothed average profile.

3 Conclusion

For operational 15-minute forecasting, XGBoost (RMSE 10.71) is the superior model. However, the WNN implementation remains a valuable diagnostic tool for understanding historical pattern

repetitions.

A Exhaustive Leaderboard

Rank	Model Architecture	Validation RMSE
1	**XGBoost (Optimized)**	**10.71**
2	NNAR	10.92
3	Random Forest	11.80
4	Prophet	12.94
5	SARIMAX (with Temp Lags)	13.40
6	HW Multiplicative	13.85
7	HW Additive	14.10
8	TBATS	14.15
9	Holt Linear Trend	14.90
10	auto.arima	15.30

B Modeling Framework

Category	Model	Parameters Tuned	Search Strategy
Baselines	SNaive / RW	None	RMSE scaling.
Smoothing	SES / Holt / HW	$\alpha, \beta, \gamma, \phi$	Automated (`ets`) + Grid search.
Stochastic	SARIMAX	$p, d, q+$ `xreg`	Temperature & Lags as regressor.
Machine Learning	**XGBoost**	**eta, depth, nrounds**	**Complete Grid Search.**
Distance (P2)	WNN	k, weighting	$1/d^2$ squared inverse distance.

C Data Pre-processing: Datetime Parsing Helper

```
parse_datetime_robust <- function(dt_vector) {
  parsed <- lubridate::parse_date_time(dt_vector,
                                       orders = c("mdy HMS", "dmy HMS", "ymd HMS"))
  return(parsed)
}
```