Minor project
Big Data & Small Data

Smart Energy Twin

Arnhem, version 2021
Master Engineering Systems

PREFACE

This report presents the outcomes of our minor project within the Master Engineering Systems program at HAN University, conducted under the Smart Energy Twin initiative. Our team, comprising Melika Mirmohammad, Pavel Petrovski, Khashayar Amir Hosseini, and Mohammad Eghbalighahyazi, developed a machine learning and neural network based forecasting models to predict energy consumption and heat demand in a multi-source residential system for eight households. Addressing the critical need for energy efficiency, our project transforms raw data into precise predictions to optimize resource management. Under the supervision of Aishwarya Aswal and in collaboration with our client, Trung Nguyen, we navigated technical and collaborative challenges through weekly progress updates. This report details our methodology, from data processing to model evaluation, and evaluates the performance of several algorithms, including, Random Forest Regressor, Neural Networks, and XGBoost. Our aim was to deliver a robust forecasting tools that meets academic rigor and provides practical value for sustainable energy management.

MP-Smart Energy Twin

Arnhem, June 14, 2025

Project title:

Smart Energy Twin

Name of the group:	**MP-Smart Energy Twin**
Name of students:	Melika Mirmohammad Pavel Petrovski Khashayar Amir Hosseini Mohammad Eghbalighahyazi
Company:	HAN University of Applied Science
HAN Supervisor:	Aishwarya Aswal
Client:	Trung Nguyen
Date:	16th June, 2025

Summery

This project aimed to develop reliable and interpretable machine learning models for forecasting both electricity consumption and heat demand across eight residential households in the Smart Energy Twin project. Using a range of algorithms—including Random Forest, XGBoost, and CNN-GRU—we explored various feature engineering techniques like lagged variables, rolling windows, cyclical encodings, and temperature interactions.

For electricity consumption, models trained on minutely data achieved high accuracy, with XGBoost delivering the best forecasting performance when enhanced with lag and rolling features (Test RMSE ≈ 1769 W, R^2 ≈ 0.86). For heat demand, both hourly and daily models were created. The daily model, based on delta predictions, achieved the lowest RMSE (~0.39 kW) and highest generalizability.

Our experiments show that data preprocessing, feature engineering, and thoughtful hyperparameter tuning are critical for accurate energy forecasting. These results support the broader goal of improving household-level energy management using machine learning and digital twin frameworks.

Table of Contents

List of Figures

List of tables

1 INTRODUCTION

1.1 Background

Managing energy efficiently in today's world has become more important than ever, especially with growing demands and the push for sustainable living. One of the emerging solutions in this area is the use of Smart Energy Twins—essentially, digital versions of real-world energy systems. These virtual models can mirror the actual behavior of a system, helping us monitor, predict, and even optimize how energy is used. When combined with modern machine learning techniques, these digital twins become even more powerful, offering smart ways to improve how energy is managed (Farahani, Akbari, Asgari, & Lee, 2024).

In residential areas, especially homes, predicting electricity and heat demand accurately can be a challenge. People's energy usage patterns are often unpredictable and influenced by many factors like weather, occupancy, and lifestyle. This is where machine learning plays a crucial role. Recent studies have shown that machine learning models, when trained on real consumption data, can forecast residential electricity demand with impressive accuracy (Li, Wang, Liu, & Zhang, 2023). These models can analyze past data and learn patterns that aren't immediately obvious.

What's even better is that some approaches focus on making these models transparent and easy to understand. For example, combining fuzzy systems with optimization techniques allows not only precise predictions but also clear explanations of what factors are affecting energy consumption (Mousavi, Pour, & Dehghani, 2024). This is especially useful when energy providers or homeowners want to make informed decisions based on the predictions.

Additionally, when it comes to managing energy at the appliance level, short-term forecasting becomes crucial. Techniques like sequence-to-sequence learning, particularly with LSTM networks, have shown great success in predicting when and how much energy individual appliances will consume (Ahmed, Hussain, Kim, & Han, 2021). This level of detail can help households fine-tune their usage and avoid unnecessary waste.

In this project, we aim to explore and identify the most effective machine learning approaches for modeling and forecasting the energy demand of eight residential houses. After processing the raw data provided by the client, we will apply various machine learning techniques to predict electricity or heat consumption. The goal is to improve energy management strategies, reduce costs, and promote more sustainable energy usage.

1.2 Problem definition

HAN University and its partners in the Smart Energy Twin project lack a reliable and evaluated forecasting model to predict heat demand and electricity consumption within a multi-source energy system serving residential households. Such a model is needed to uncover the correlations between diverse energy data—characterized by stable and variable components—enabling optimization of energy management and improved cost efficiency.

1.3 Project Objective(s)

The primary objective of this project is to develop two reliable and evaluated machine learning models to forecast hourly heat demand and electricity consumption for residential households within a multi-source energy system, providing daily predictions. Data engineering to preprocess raw measurement data will form the foundation of the models, enabling the discovery of relationships within diverse energy data to support optimized energy management and cost efficiency.

1.4 Research Question

Main question:

How can a reliable and evaluated machine learning models be developed to forecast hourly heat demand or electricity consumption for residential households in a multi-source energy system, delivering daily predictions by making use of correlations within energy data?

Sub questions:

1. Is the currently available dataset sufficient to develop reliable forecasting models for heat demand and electricity consumption?
2. What data processing techniques are most effective for transforming raw energy measurements into a format suitable for accurate forecasting machine learning model?
3. Which machine learning model (e.g., time-series models, regression, or neural networks) is best suited for predicting hourly heat demand or electricity consumption in this residential energy system?
4. What validation strategies and performance metrics should be used to ensure model reliability and generalizability?

1.5 Approach

Our approach focuses on forecasting residential electricity and heat demand using machine learning models, with XGBoost, Random Forest and CNN as well as the neural network chosen for their ability to handle non-linear data and temporal dependencies and making sure we have implemented enough algorithms to achieve the best results possible.

- We used both minute-level and hourly-averaged data to compare how data granularity affects performance. For hourly models, lag and rolling features were introduced to compensate for the loss of temporal resolution.
- To ensure realistic evaluation, we applied time-based train/validation/test splits and used GridSearchCV for hyperparameter tuning. Instead of exhaustive tuning, parameter ranges were narrowed based on RMSE trend analysis to reduce computation time.
- By analyzing feature importance and model errors, we ensured that added features—like short-term history and temperature—contributed meaningfully to the predictions. This strategy balances accuracy, interpretability, and practicality for deployment in energy-aware systems.

1.6 Outline of the minor project report

In this study, a summary of relevant reviewed literature is presented in Chapter 2, which explains the use of different machine learning methods for prediction of energy demand. The methodology, including data cleaning, pre-processing, and demonstration of the models used for the prediction of heat demand and electricity consumption, is explained in Chapter 3. Chapter 4 presents the corresponding results for different models with different sets of features. The results for the prediction models are compared in Chapter 5, and the report is concluded in Chapter 6.

2 LITERATURE SURVEY

Residential buildings consume substantial energy for both heating and electricity – space heating and hot water alone account for a large fraction of household energy use (Neele Kemper, 2023), while residential electricity use contributes roughly 27% of total power consumption (Alisha Banga, 2024). Managing this demand is key to energy efficiency and decarbonization objectives. Efficient energy management systems (e.g. smart thermostats, heat pumps, home energy management) require accurate short-term forecasts of heat and power demand to optimize operation and integrate renewable sources (Neele Kemper, 2023). Accurate hourly forecasting of both heat and electricity loads thus helps balance supply and demand, enhance grid stability, and reduce operational costs; indeed, even a 1% improvement in load forecast accuracy can save thousands of megawatts and millions of dollars (Alisha Banga, 2024) .In recent years, numerous data-driven modeling approaches have been proposed for this task. This section summarizes academic studies that apply machine learning methods – including linear regression, decision trees, neural networks, and ensemble models like XGBoost, to forecasting hourly and daily heat demand and electricity consumption in residential and multi-source energy systems.

Electricity Demand Forecasting

Electricity usage in households fluctuates with occupancy, appliance use, and environmental conditions. Cyclic daily/weekly patterns coexist with random spikes, making demand partially linear and largely nonlinear. (Alisha Banga, 2024) Many recent studies employ machine learning to model these patterns at an hourly scale. Linear regression is often included as a baseline or part of a hybrid model – for instance, Banga and Sharma (2024) combine a trend-capturing linear model with XGBoost to better represent both seasonal structure and random fluctuations in a household load dataset (Alisha Banga, 2024). Decision-tree-based methods have shown strong capability in modeling the nonlinear aspects of residential loads. Moreover, boosting algorithms have frequently

achieved top accuracy: in a recent residential load forecasting study, the XGBoost model outperformed all other machine learning and even deep learning models tested (Alisha Banga, 2024). According to studies in (Alisha Banga, 2024) and (C. Monzani, 2024), these methods effectively model weather impacts, outliers, and nonlinear interactions, although they may struggle with trend extrapolation. Deep learning also plays a major role.
Majumdar (2025) emphasized the importance of time-series feature engineering using lag and rolling window statistics for improving ML model performance. Lag features (e.g., previous hour/day values) and rolling statistics (e.g., rolling means, min/max) help in capturing short-term trends and autocorrelation in time-series data. These techniques are widely applied in electricity forecasting scenarios to structure time-dependent data into a supervised learning framework that can be used by models like XGBoost, random forests, or neural networks. (Majumdar, 25)
(SAJJAD, 2020) studied the effectiveness of hybrid deep learning models, specifically a CNN-GRU architecture, for short-term residential electricity load forecasting. They emphasized the limitations of traditional models such as linear regression, support vector regression (SVR), and decision trees when dealing with complex, non-linear, and noisy time-series energy data. Their CNN-GRU hybrid model extracted spatial and temporal patterns by combining the feature extraction capability of CNNs with the sequence modeling strength of GRUs. The model achieved significantly lower error metrics (MSE = 0.09, RMSE = 0.31, MAE = 0.24) on the AEP and IHEPC datasets, outperforming LSTM, CNN-LSTM, and SVR approaches.

Heat Demand Forecasting in Residential Buildings:
Heat demand in residential buildings is strongly driven by weather conditions, especially outdoor air temperature (Łukasz Guz, 2025). Traditional prediction methods (e.g. physics-based heat balance or simple time-series models) often struggle to capture occupant behavior and nonlinear effects, motivating the use of machine learning (Neele Kemper, 2023). Simple regressions have been used as benchmarks – for example, a third-degree polynomial fit to outside temperature explained about 74% of the variance in hourly heating energy demand for retrofitted homes (Łukasz Guz, 2025). However, linear or polynomial models are limited by the assumption of a fixed functional form. They may fail to capture complex temporal patterns during peak demand or sudden weather change. (Neele Kemper, 2023). Another study explored short-term thermal energy demand forecasting for a solar district heating system using machine learning techniques including decision trees, support vector machines, and artificial neural networks. They demonstrated that incorporating multiple inputs—such as outdoor air temperature, solar radiation, time of day, and occupancy proxies (weekend indicators and working hours)—enhanced the models' performance compared to a baseline piecewise linear regression. The models showed up to 23.9% improvement in RMSE over linear regression, with results remaining consistent even when tested on weather forecasts with 6-hr and 48-hr horizons (Etienne Saloux, 2018). A variety of machine learning techniques have been tested for short-term heat load forecasting in the residential sector. Kemper (2025) compared time-series models, online learning, and offline machine learning in a case of 66 residential units; they found offline ML models (trained on historical batches) achieved the best accuracy (Neele Kemper, 2023). In particular, tree-based ensemble methods have shown strong performance – e.g. random forest regression yielded low error and proved robust to noise and outliers in heating data. Recent work on district heating systems (serving residential communities) highlights the value of advanced machine learning in forecasting heat loads. Extreme gradient boosting methods (XGBoost) in particular have stood out for their combination of high predictive accuracy and computational efficiency (C. Monzani, 2024). For example, one study integrated a metaheuristic optimizer with XGBoost and achieved a coefficient of determination $R^2 \approx 0.94$ in building heating load prediction. (Łukasz Guz, 2025). Several studies report that gradient-boosted trees (XGBoost) not only reduce prediction error but also train quickly, which is advantageous for online updates in heat networks (C. Monzani, 2024).
Artificial neural networks are also widely applied to thermal load forecasting. a properly trained deep neural network could outperform other approaches (including support vector machines, shallow neural nets, and decision trees) in 24–72 hour heat demand forecasting for a district heating case. In general, boosting and neural network models tend to handle the nonlinear relationships and complex usage patterns of heat demand better than simpler techniques.

3 METHODS

This chapter outlines the methodology for forecasting electricity and heat demand, focusing on daily hourly power consumption predictions for eight residential households in Wolfheze, Netherlands, using data from the Smart Energy Twin project.

3.1 Electricity consumption forecasting

The forecasting model predicts hourly power consumption for residential households using Machine Learning (Random Forest and XGBoost) and Neural Network (CNN-GRU) approaches, based on minutely-sampled sensor data.

3.1.1 Data

Two datasets provide minutely-sampled sensor readings:.

- **wolfheze_reading_electrical_meters_2_for_weeks.csv**: Covers 2 weeks (2025- 02-05 to 2025-02-19), with labels: reading, reading_date, device_fk, parameter. Includes measurements like voltage, current, power per phase, and total imported/exported energy
- **device_data_woningen.csv**: Spans 4 months (2024-12-19 to 2025-04-22), with labels: reading, reading_date, device_fk, parameter, device_description, parameter_description, power_phase_description.

3.1.2 Data Processing

Datasets were checked for completeness. The 2-week dataset is fully sampled, but the 4-month dataset has gaps after 2025-04-19, so it was truncated to 2025-04-19 12:42:07. Initial visualizations (time-series plots of power per phase) identified daily/weekly patterns and anomalies. Key processing steps:

- **Data correction**: Power_phase_2 and power_phase_3 sensors were reversed until 2025- 04-10, requiring sign correction for earlier data
- **Temperature extraction**: Hourly temperature data for Wolfheze (latitude 52.0056717◦N, longitude 5.7923944◦E) was sourced from the Open-Meteo archive API (Open-Meteo, 2025) and mapped to minutely timestamps.
- **Device and Parameters**: Only data for houses (device_fk=1) was used. Total power was computed as the sum of power_phase_1, power_phase_2, and power_phase_3 (parameters 7, 8, 9), as total_import_kwh (parameter 11) was unavailable in the 4-month dataset.
- **Data normalization**: Temperature and total_power were standardized using y = x−μ σ , where μ is the mean and σ is the standard deviation, to ensure equal feature contribution.

$$y = \frac{x - \mu}{\sigma} \ (1) \ where:$$

x is the original value of the feature
μ The mean of the feature across the entire dataset
σ The standard deviation of the feature across the dataset
y The standardized value

3.1.3 Feature Engineering

Features were engineered to enhance model performance:

- **Cyclical Encoding**: Hour of day was encoded as $hours\_\sin = \sin(2\pi * \frac{hour}{24})$, $hours\_\cos = \cos(2\pi * \frac{hour}{24})$ to capture daily cycles.
- **Workday Flag**: Binary flag (1 for weekdays, 0 for weekends) to reflect weekly patterns, for RFR and CNN&GRU.
- **One-hot encoding:** Days of the week are numerically encoded for XGBoost algorithm.
- **Outlier Handling**: Winsorization capped temperature and total_power at the 1st and 99th percentiles to mitigate extreme values, for the Random Forest Regressor, while for the other algorithms all power values were retained to preserve real spikes.

3.1.4 Metrics

The model is assessed using the following metrics:

- RMSE (Root Mean Squared Error)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_{actuali} - y_{predictedi}\right)^2}$$

Quantifies the average magnitude of errors in the predicted power, penalizing larger errors more due to the squaring operation.

- R^2 Score:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(y_{actuali} - y_{predictedi}\right)^2}{\sum_{i=1}^{n}(y_{actuali} - \bar{y}_{actual})^2} \; where \; \bar{y}_{actual} \; is \; the \; mean$$

Measures the proportion of variance

- *MAE (Mean Absolute Error)*

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_{actuali} - y_{predictedi}\right|$$

Measures the average absolute difference between predicted and actual power, providing a linear error metric that treats all deviations equally.

3.1.5 Random Forest Regressor

The Random Forest (RF) model is an ensemble learning technique based on decision trees, designed for regression task. It operates by constructing multiple decision trees during training, each on a random subset of the data and features, and averages their predictions to improve accuracy and reduce overfitting. For each tree, a random sample is drawn from the training set. At each node, a random subset of features is considered for the best split, enhancing diversity. For regression, the final prediction is the mean of the outputs from all trees:

$$\hat{y} = \frac{1}{T}\sum_{t=1}^{T} y_t \quad (2)$$

Where T is the number of trees and y_t is the prediction from tree t.

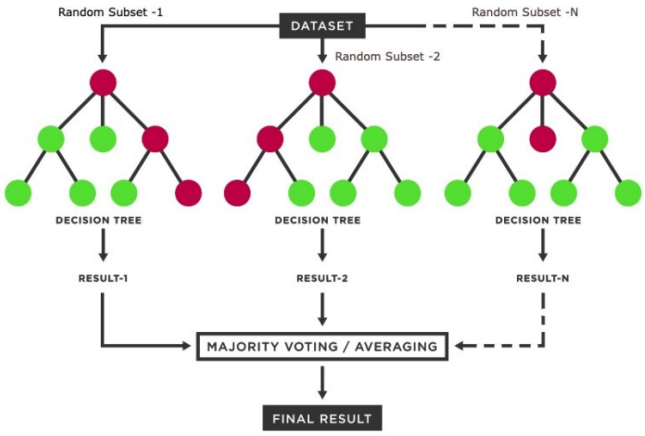


Figure 1 Random Forest

The RF is able to handle nonlinearity and is robust to outliers, making it suitable for the dataset's variability.

- **Data Split**: Employed an 80/20 train-test split using scikit-learn (Scikit-Learn, 2025) *train_test_split* with random state of 42 to ensure reproducibility. The training set (80%) was used for model fitting and hyperparameter tuning via 3-fold cross-validation, while the test set (20%) provided an unbiased evaluation of generalization performance.
- **Input features:** hour_sin, hour_cos, is_workday and temperature, and total_power as the sum of the three phases as output.

- **Initial Model Architecture**

  The initial RFR model, used default hyperparameters:

Table 1 Initial RFR Model Parameters

n_estimators	100
Max_depth	None
Min_samples_split	2
Max_features	auto
Min_samples_leaf	1

- **Optimization**

  Hyperparameter optimization was performed using GridSearchCV with 3-fold cross-validation, minimizing root mean squared error. The parameter grid is listed below and The best model was retrained on the full training set.

Table 2 RFR optimization Parameters

n_estimators	[100, 200, 300]
Max_depth	[None, 10, 20, 30]
Min_samples_split	[2, 5, 10]
Max_features	['sqrt', 'log2', None]
Min_samples_leaf	[1, 2, 4]

3.1.6 CNN & GRU

The Convolutional Neural Network-Gated Recurrent Unit (CNN-GRU) model combines convolutional and recurrent neural network layers for time series regression, specifically predicting average hourly total power. The CNN extracts short-term spatial patterns from the input sequence, while the GRU captures long-term temporal dependencies. The model processes 1440-minute (24-hour) sequences to predict the next 60-minute average total power, leveraging nonlinearity to handle spikes (e.g., from appliance usage). Predictions are generated by averaging outputs from a single dense layer across the sequence processing pipeline.

- **Convolutional Neural Network (CNN)**

  CNNs are adept at extracting spatial features from structured data, adapted here for 1D time series. They use convolution operations to detect local patterns, such as short-term power usage spikes. (SAJJAD, 2020).

Two convolutional layers with 16 and 8 filters (kernel size 2, ReLU activation, 0.3 dropout) process 1D sequences. Convolution output is: $f_i = \sum_{j=0}^{k-1} w_j * x_{i+j} + b$

Where k is the kernel size and b is the bias.

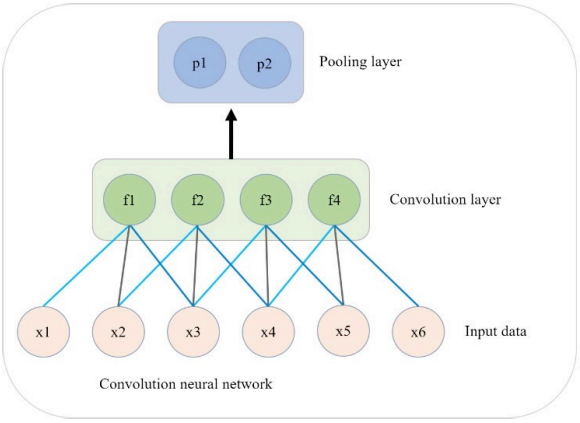


Figure 2 Basic CNN architecture

- **Gate Recurrent Unit (GRU)**
  Two GRU layers (16 and 8 units, 0.3 dropout) model temporal dynamics.
  GRUs use two gates - update (z_t) and reset (r_t) - to control information flow. The equations are :

$$z_t = \theta(W_z[h_{t-1}\, x_t] + b_z)\ Update\ gate$$

$$r_t = \theta(W_r[h_{t-1}\, x_t] + b_r)\ Reset\ gate:$$

$$\widehat{h_t} = tanh(W_h[r_t h_{t-1}\, x_t] + b_h)$$

$$Final\ hidden\ state:\ h_t = (1 - z_t)h_{t-1} + z_t\widehat{h_t}$$

Where x_t is input, at time t, h_{t-1} is the previous hidden state, h_t is the new hidden state, W and b are weights and biases, and θ is the sigmoid function. The update gate z_t determines how much past information to retain, while the reset gate $r_t$controls how much past state to forget.

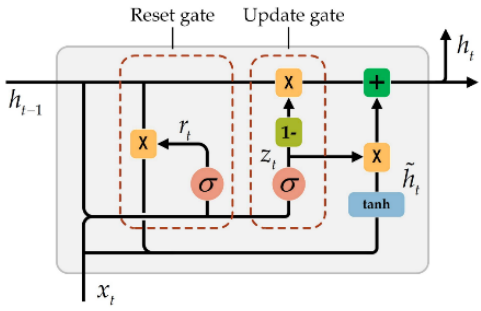


Figure 3 Basic GRU architecture

A Dense(1) layer outputs predictions. The model uses a 70/15/15 (train/validation/test) split, Adam optimizer (learning rate 0.001), batch size 64, and early stopping (patience 2)..

- **Sequence Generation and Model Input**

The CNN-GRU architecture processes input sequences of 1440 minutes, each with five features (hour_sin, hour_cos, is_workday, temperature, total_power). A sliding window generates sequences, targeting the next 60-minute mean power. This window slides forward by one minute, ensuring comprehensive coverage and enabling the model to learn daily power patterns. The CNN initially processes these sequences to extract spatial features, which are then fed into the GRU layers, culminating in a Dense(1) layer for the final prediction.

- **Model Architecture**

The model architecture combined two CNN layers for spatial feature extraction and two GRU layers for temporal modeling, and a single Dense layer for hourly predictions.

Table 3CNN-GRU Model Architecture

Layer	Type	Filters/Units	Kernel Size	Activation	Dropout
1	Convolutional	16	2	ReLU	0.3
2	Convolutional	8	2	ReLU	0.3
3	GRU	16	-	-	0.3
4	GRU	8	-	-	0.3
5	Dense	1	-	-	

- **Optimization**

The optimization employed a random search with 8 runs over a limited hyperparameter grid, using 3-fold cross-validation on an 80/20 train-test split.

Architecture: Fixed CNN filters (16, 8), kernel sizes (2), GRU units (16, 8), and dropout (0.3) were used, with L2 regularization (λ\lambdaλ) applied to all layers.

Training: Utilized the Adam optimizer, with early stopping on validation loss to prevent overfitting, limiting epochs to 15 per fold.

Evaluation: Selected the best hyperparameters based on the lowest mean validation MSE

Table 4 CNN&GRU Hyperparameter optimization

Hyperparameter	Values Tested
Learning Rate α	[0.0005, 0.001]
Regularization λ	[0.001, 0.01]
Batch Size	[128, 256]

3.1.7 XGBoost Regressor

This part explains the methodology employed for forecasting electricity demand using an XGBoost regression model. The process involved several critical stages, from initial data preparation and visualization to advanced feature engineering, hyperparameter tuning, and model evaluation. The entire analytical pipeline was structured to ensure robustness and accuracy in predicting future electricity consumption.
After Preprocessing and cleaning the data we continue with splitting the data as follows:

Note : No Need for Normalization Using XGBoost

A key aspect of XGBoost's methodology is its inherent robustness to feature scaling:

- **Tree-Based Model Nature**: XGBoost is an ensemble of decision trees. Decision trees operate by partitioning data based on feature thresholds, and the performance of these thresholds is not affected

by the scale or distribution of numerical features. Therefore, unlike models sensitive to feature magnitudes (e.g., K-Nearest Neighbors, Support Vector Machines with RBF kernels, neural networks), XGBoost generally does not require feature normalization or standardization. This property simplifies the preprocessing pipeline and makes the model less sensitive to outliers in feature scales.

1. Splitting the Data into 3 Random Datasets and Training the Base Dataset

To ensure a thorough and robust evaluation, the dataset was split into three distinct subsets:

- **Base Dataset (Training)**: The primary dataset used for training the XGBoost model.
- **Validation Dataset**: Used for hyperparameter tuning and early stopping during training to prevent overfitting.
- **Test Dataset**: An unseen dataset used for the final evaluation of the model's generalization performance.

2. Hyperparameter Tuning Using Grid Search

Optimal model performance often relies on well-tuned hyperparameters:

- **Grid Search Cross-Validation**: A systematic search approach was employed to find the best combination of hyperparameters for the XGBoost model. This involved defining a grid of hyperparameter values and evaluating the model's performance (e.g., using RMSE) for every possible combination within the grid. Cross-validation was used within the grid search to ensure that the chosen hyperparameters generalize well to unseen data.

3. Tuned Model Evaluation

After identifying the optimal hyperparameters, the model was re-trained with these settings and thoroughly evaluated:

4. Lag and Rolling Features

To incorporate temporal dependencies and capture trends, advanced feature engineering techniques were applied:

- **Lag Features**: Previous values of the total_power (and potentially other relevant features like temperature) were added as new features. For example, total_power_lag_1 would represent the total_power from the previous time step. This helps the model capture autocorrelation and short-term trends.
- **Rolling Features**: Statistical measures (mean, standard deviation) over a rolling window (last 3 hours, last 1 hours) of total_power (and other features) were created. These features help the model capture longer-term trends and volatility.

5. Feature Importance

Understanding the contribution of each feature to the model's predictions is crucial for interpretability:

- **XGBoost Feature Importance**: XGBoost models inherently provide a measure of feature importance (e.g., 'gain', 'weight', 'cover'). These scores indicate how useful or valuable each feature was in the construction of the boosted decision trees. Visualizing these scores helps identify the most influential factors in electricity demand.

6. Tuning Using Grid Search and Evaluation (with Lag and Rolling Features)

The process of hyperparameter tuning and evaluation was repeated after incorporating the newly engineered lag and rolling features:

- **Re-tuning**: Given the new set of features, the Grid Search Cross-Validation (as described in point 8) was re-executed to find the optimal hyperparameters for the model with the enriched feature set.
- **Re-evaluation**: The model with the newly tuned hyperparameters and engineered features was re-evaluated on the test set using the same performance metrics (MSE, RMSE, MAE, MAPE, R2). This step ensures that the added features genuinely improve predictive performance and that the model remains robust.

7. Aggregating the Mean of 60 Samples for Hourly Sampling

The original dataset likely contains data sampled at a higher frequency (e.g., minute-level). To align with an hourly forecasting objective and reduce noise:

- **Resampling to Hourly Mean**: The total_power and other relevant numerical features were resampled to an hourly frequency by calculating the mean of 60 minute-level samples within each hour. This reduces granularity and aligns the data to the desired prediction interval.

8. Doing All Above for the New Dataset (Hourly)

Steps 2 through 12 were meticulously repeated for the new hourly sampled dataset. This ensures that the entire analytical pipeline, from visualization and preprocessing to feature engineering, tuning, and evaluation, is applied consistently to the hourly data, optimizing the model specifically for hourly predictions.

9. Saving the Model

Once the final XGBoost model (trained on the hourly data with optimal hyperparameters and features) demonstrated satisfactory performance:

- **Model Serialization**: The trained XGBoost model object was saved to disk using a suitable serialization method (e.g., joblib or pickle). This allows for the reusability of the trained model without the need for re-training, facilitating future predictions and deployment.

10. Predicting the Last 24 Hours

The ultimate goal of the project was to forecast future electricity demand:

- **Future Data Preparation**: A dataset for the next 24 hours was extracted from the dataset, including all necessary features (hour, day of week, cyclical components, and appropriate lag/rolling features derived from the most recent historical data).
- **Forecasting**: The saved XGBoost model was loaded and used to predict the total_power for these future 24 hours.
- **Visualization of Forecast**: The predicted electricity demand for the next 24 hours was plotted alongside the most recent 48 hours of actual data. This visualization provides a clear comparison and allows for immediate assessment of the forecast's quality and alignment with recent trends. The plot would typically show hour_timestamp on the x-axis and total_power on the y-axis, distinguishing between 'Actual' and 'Predicted' data points.

This systematic and detailed methodology ensures that the electricity demand forecasting model is built on a solid foundation, capable of providing reliable and interpretable predictions.

3.2 Heat Demand of Residential Buildings

3.2.1 Cleaning the Raw Heat Demand Data:

To analyze four months of data from 17 January 2025 to 19 April 2025, it was first necessary to clean the dataset. The raw data contained readings from various sources, including heat meters, electricity meters, and solar panels. Initially, the heat meter data was isolated from the rest. This dataset included both heating (number_1) and cooling (number_2) values, as confirmed by comparison with other cleaned data available from this project. The cooling data was subsequently removed.

The heating data consisted of two components: space heating and domestic hot water. These were identified and separated based on supplementary data and household coding information from other related projects, then combined to calculate the total heat demand. Additionally, temperature data for the same period was incorporated using the same method previously applied to the electricity data.

Further preprocessing steps were required due to missing values and potential outliers in the dataset. These issues were addressed by removing the missing entries and performing outlier checks. The cleaned data was then organized by selecting relevant features such as temperature, day of the month (1–31), day of the week (1–7), weekend indicator, non-working hour flag, and month. All of these steps were implemented using Python.

It is important to note that the data preparation process was initially performed on a two-week subset before extending the same procedure to the full four-month dataset. Since the two-week period falls within the four-month range, this report focuses on the comprehensive four-month dataset, which inherently includes the shorter timeframe.

3.2.2 Model Training:

3.2.2.1 XGBoost Model

In the next phase, the cleaned and structured data was used to train a model capable of predicting heat demand on both an hourly and daily basis. The initial focus was on hourly forecasting. Among various machine learning techniques, XGBoost was selected due to its proven effectiveness in heat load prediction. This model is particularly advantageous because it can integrate a wide range of input features (e.g., weather conditions and time-based factors) and capture short-term fluctuations in heat demand without relying on an explicit physical model of the system. By including additional predictors such as hour of day, day of week, and month (to reflect occupant behavior patterns), the XGBoost model significantly improved prediction accuracy over simpler models. (Maciej Bujalski, 2021). Another recent work in 2025 optimized an XGBoost model using Bayesian hyperparameter tuning for residential building energy usage: the optimized XGB model (called TPE-XGB)

outperformed other state-of-the-art models, improving prediction accuracy for heating load by ~3.4% and for cooling load by ~10% compared to previous best models (Alawi, 2024)

XGBoost is an implementation of the gradient boosted decision tree algorithm, designed for efficiency and high predictive performance. Conceptually, it builds an ensemble of many shallow trees (weak learners) to create a strong predictive model. The model starts with an initial prediction (which could be a simple average) and then adds decision trees one by one. Each new tree is trained to fit the residual errors (the difference between the current model prediction and the true values), thereby sequentially correcting errors made by the existing ensemble. This additive model can be described as: the predicted output is the sum of predictions from all trees built so far. By combining multiple trees, XGBoost captures complex nonlinear relationships between input features (like temperatures, occupancy schedules, etc.) and the target heat demand. (Maciej Bujalski, 2021)

To train our model for predicting heat demand (with total heat as the target variable), we first imported several Python libraries:

pandas, numpy (for data manipulation), matplotlib, seaborn (for statistical data visualization), sklearn.model_selection (for data splitting), sklearn.metrics (for evaluation metrics), xgboost, and itertools (for iterations).

3.2.2.2 Artificial Neural Network

Neural Network Architecture

A range of neural network architectures were tested, from single-layer networks (e.g., 5, 10, 20, 30 neurons) to multi-layer configurations (e.g., [10, 5], [20, 10], [30, 15], [15, 10, 5]). Each network was configured with a training ratio of 70%, validation ratio of 15%, and 1000 epochs, with early stopping after 20 validation failures to prevent overfitting. The best architecture was selected based on the lowest validation RMSE, and results were visualized to compare RSME across architectures.

Neural Network Ensemble Size

To improve robustness, ensembles of neural networks were tested with sizes of 1, 3, 5, 10, 15, and 20 networks, each using the best architecture from the previous step. Initial weights were randomized for diversity, and predictions were averaged across the ensemble. Performance metrics were computed for validation and test sets, and the optimal ensemble size was determined by minimizing validation RMSE. Results were plotted to illustrate the effect of ensemble size on RSME.

Neural Network Learning Rate

Learning rates of 0.001, 0.01, 0.05, 0.1, 0.2, and 0.5 were tested for the neural network with the best architecture. Each network was trained and evaluated, with the optimal learning rate selected based on validation RMSE. Semilog plots illustrated the relationship between learning rate and RSME.

3.2.2.3 Hourly predicted models

First Hourly Model:

For the first model, we used the following features: hour, hour_squared, day_of_week, week_end, non_working_hour, temperature, temp_squared, day

Initially, we attempted to train the model using only the main features, but this approach failed to capture the data patterns effectively. Therefore, we introduced non-linear features such as temp_squared and hour_squared to help the model better capture complex trends in the data.

The dataset was then split into three subsets: 75% for training, 15% for validation, and 15% for testing. This split was applied consistently across all models.

We trained the model using XGBoost, which relies on several hyperparameters that govern model complexity, training speed, and generalization performance:

- Number of estimators: This controls how many trees are added to the model to improve prediction accuracy. Based on prior studies (e.g., Heat Load Forecasting with XGBoost), we considered 300–400 estimators, as 300–500 is typically sufficient for datasets of moderate complexity.

- Max depth: Defines the maximum depth of each decision tree. We selected a range of 5–7 to avoid underfitting or overfitting. Shallow trees (depth ≤ 3) may underfit non-linear relationships (between temperature and heating), whereas very deep trees (depth >10) tend to overfit noise in the meter data.
- Learning rate: Controls the steps to reach the convergence.
- Subsample: Refers to the fraction of training data used to build each tree. Since heat meter data often includes noise, using a lower learning rate and subsample values helps reduce overfitting. We experimented with subsample values of [0.6, 0.7, 0.8], as values between 0.5 and 0.8 are common in practice. Values below 0.5 may filter out too much signal, while values above 0.9 can reduce the effects of regularization.

We defined a loop across ranges for each hyperparameter and selected the best configuration based on the lowest RMSE score.

The choice of RMSE (Root Mean Square Error) over R^2 as the primary evaluation metric is intentional and supported by domain-specific considerations. In building energy forecasting, RMSE is preferred because: RMSE offers clear interpretability by quantifying errors in physical units of energy, it is sensitive to large errors which are critical to detect and avoid, and it aligns with energy performance assessment standards that require low prediction error for models to be considered useful. Additionally, focusing on RMSE dovetails with model optimization, since algorithms like XGBoost explicitly minimize squared error. Meanwhile, R^2, though useful as a descriptive statistic, can be misleading or less relevant in this domain – it focuses only on variance explained and can mask issues like bias or the real magnitude of errors. As a result, researchers and practitioners in building energy modeling lean on RMSE (and related error metrics) as their primary yardstick for model performance (Avina, 2022).This ensures that the models selected are not just statistically "good" in an abstract sense, but practically accurate and reliable for forecasting heat demand in buildings(Sharma, 2019). If one looked at R^2 alone, they might undervalue the model, but RMSE/CV(RMSE) revealed it was performing well for practical purposes. This underscores why experts caution against over-relying on R^2 in energy modeling – one publication bluntly states that "R^2 should be ignored" and only error-based criteria like CV(RMSE) should determine acceptability of a model's fit.

So, we evaluated model performance on both the validation and test sets using three metrics: RMSE, CV(RMSE), and R^2.

We followed the same modeling pipeline for other models, with the only variation being in feature selection and hyperparameter ranges.

Finally, we Plotted actual vs. predicted heat demand on an hourly basis, Included feature importance graphs to analyze the influence of each feature on the model, and we plotted the scatter plot of residuals to assess their distribution.

Second Hourly Model:

For the next model, we included the month feature but excluded the date variable. To compensate for the absence of day, we emphasized non-linear time-based features. The selected features integrate month, time-of-day, and weather-related variables, reflecting typical patterns in heat usage behavior. The features of this model were mentioned in table 16. They also include some feature engineering: 'is_morning 'and 'is_evening 'are binary features indicating peak heating periods due to occupant routines such as showering, cooking, and returning home. Binary encoding simplifies pattern recognition for the model. 'hour_sin' and 'hour_cos 'capture the cyclical structure of time, accounting for the fact that hour 23 is close to hour 0. The model was trained using the following hyperparameters:

- learning_rate = 0.05: A small step size that helps prevent overfitting.
- n_estimators = 150: A moderate number of trees to balance complexity and generalization. A higher learning rate allows for fewer estimators.
- max_depth = 4: Limits the depth of each tree to reduce the risk of overfitting on noisy hourly data.
- subsample = 0.9: Introduces randomness to reduce variance.
- colsample_bytree = 0.8: Trains each tree on a subset of features to increase model robustness.

The same training and validation and plots procedure as the previous model were followed.

Third model:

In this model, only ambient temperature, hour of day and weekend binary feature were considered as the original features to avoid making a date-dependent model.

To enhance model performance, several feature engineering techniques were applied to capture temporal, interaction, and nonlinear relationships in the data:

- Cyclic Encoding of Hour: The hour of day was transformed using sine and cosine
- Interaction Terms: An interaction feature was computed by multiplying ambient temperature by the weekend indicator to model potential differences in heating behavior on weekends. Hour-specific temperature interactions were also generated for each hour (0 to 23) by multiplying a binary hour indicator with ambient temperature.
- Nonlinear Transformations: To capture nonlinear effects, ambient temperature was squared and cubed, providing polynomial representations of temperature influences.
- Lag Features: Three lag features for ambient temperature (lag-1, lag-2, and lag-3) were created to incorporate temporal dependencies, assuming the data was chronologically ordered.
- Moving Averages: A three-hour moving average of ambient temperature was computed to smooth short-term fluctuations and highlight trends.

These engineered features were combined with the original features, resulting in an expanded feature set.

Fourth Hourly Model:

For the fourth model (third model of XGBoost), we removed the month feature and attempted to predict heat demand using the remaining features. However, the model underperformed. The lack of critical variables such as solar radiation, building inertia, occupant density, and window size hindered accuracy. To compensate for this, we introduced lagged and rolling features to enhance model memory and pattern recognition: (hour, day_of_week, week_end, non_working_hour, temperature, is_morning, is_evening, hour_sin, hour_cos, heat_lag_1, heat_roll_2, temp_x_hour)

- heat_lag_1: Previous hour's heat demand; captures autocorrelation and heat_roll_2: Two-hour rolling average; smooths short-term noise and captures local trends.
- temp_x_hour: Product of temperature and hour; models varying temperature effects over time.

These features improve the model's ability to recognize daily heating cycles (morning/evening peaks), account for temperature's hourly impact, and incorporate short-term memory of recent usage patterns.

For this model, we first used the hyperparameter ranges from the previous approach and then expanded them slightly through trial and error to enhance performance. The hyperparameter grid used was:

- learning_rate: [0.02, 0.03, 0.04, 0.05]
- max_depth: [3, 4, 5]
- n_estimators: [200, 300]
- subsample: [0.7, 0.8]
- colsample_bytree: [0.8, 1.0]

A loop was implemented to search across the defined hyperparameter space and select the configuration that minimized RMSE on the validation set. then model Evaluation and Visualization was done based on three metrics: RMSE, CV(RMSE), R^2. In final step, A plot of actual vs. predicted hourly heat demand for the first 100 test samples, and Feature importance plots to assess which variables contributed most to prediction accuracy were visualized. In addition to the boxplot and heatmap illustrating the impact of two hyperparameters (learning rate and max depth) on RMSE, showing how variations in their values influence model performance.

3.2.2.4 Daily Prediction model

For daily heat demand prediction, we initially trained a model to predict the absolute value of daily heat demand using the primary feature set (without lag or rolling features).

However, this approach failed to achieve satisfactory results, even after optimizing the hyperparameters and testing various parameter ranges. We then incorporated lag features and some non-linear terms such as temperature^3 and lagged temperature, but this model also performed poorly.

As a result, we shifted our modeling objective to predicting the day-over-day change in heat demand (heat_delta) instead of predicting absolute values directly. The model predicts the change in heat demand from the previous day, and the final heat prediction is reconstructed as:

Final heat prediction = heat_lag_1 + predicted_heat_delta

To implement this strategy, we first converted the hourly heat data into a daily time series. This involved fixing the year to 2025 and creating a datetime column using the year, month, day, and hour fields. These were previously separated during data cleaning for clarity. The data was then sorted chronologically and grouped by date.

Then we engineered the following daily features:

- temp_mean, temp_min, temp_max, day_of_week,
- week_end, non_working_hour, heat_mean
- temp_mean: The average temperature across all 24 hours of the day.
- temp_min / temp_max: The lowest and highest hourly temperatures recorded during the same day.
- heat_mean: The average heat demand for each day based on all hourly values.

These features help the model understand day-to-day temperature dynamics, which are crucial for accurate daily forecasting.

Next, we applied additional feature engineering to capture lag and trend-based behaviors, including: (heat_lag_1, heat_delta, heat_delta_lag_1, temp_ma_3)

- heat_lag_1: Yesterday's average heat demand.
- heat_delta: The change in demand from today to tomorrow — our prediction target.
- heat_delta_lag_1: Captures yesterday's change in heat demand, providing trend insight (e.g., consecutive increases).
- temp_ma_3: A 3-day moving average of temperature, shifted to exclude current data. This reflects how heating systems respond to gradual temperature trends rather than daily fluctuations.

These engineered features allow the model to learn from previous demand and weather trends to improve daily forecasts. We followed the same modeling pipeline used in the third hourly model: Splitting the dataset into 75% training, 15% validation, and 15% testing subsets and training an XGBoost model and tuning hyperparameters via grid search (looping over ranges). Hyperparameter tuning was done by minimizing RMSE on the validation set, using nearly the same ranges as the hourly model, with slight extensions based on trial and error. The evaluation and visualization procedures were the same as those used for the previous hourly heat demand model.

4 RESULTS

This chapter outlines, the results of the different models used to forecast electricity consumption and heat demand by the 8 residential households.

4.1 Electricity Consumption

This chapter highlights the results of the developed models for electricity consumption forecasting.

4.1.1 Random Forest Regression

This section evaluates the performance of a Random Forest Regressor model trained to predict minutely total power consumption (in Watts) using the Wolfheze dataset, which includes power measurements and temperature data. The model was optimized using GridSearchCV, and its performance was assessed using R^2, RMSE and MAE metrics, alongside visualizations of data, predictions, residuals, feature importance, and hyperparameter sensitivity. The results demonstrate the model's ability to capture temporal and environmental patterns in power consumption. This chapter focuses on the results of the model for the larger data set as they are more informative. The results for the two weeks data are present in Appendix D

Figures 13 and 14 show the data looks like after the preprocessing steps. The total power is scaled from $\{-15000; 10000\}$ to $\{-3; 3\}$ and the temperature from $\{-5; 20\}$ to $\{-2; 3\}$.
The normalized data is split to 80% training and 20% test sets. No validation set is used, because the model is optimized using GridSearch, which utilizes k-fold type of validation.

Table 5 RFR Metrices Result (4 months dataset)

Set	RMSE (W)	R^2	MAE (W)
Train	1547.18 W	0.897	931.18 W
Test	1587.98 W	0.894	950.84 W

The training set R^2 score is 0.897 which is very close to the test set performance of R^2: 0.894. The root mean squared error for the training set is 1547.18 W which is just 40W less than the RMSE of the test set. The mean average square is also very close for both train and test set with 931.18 W and 950.84 W respectively. The close performance on both test and train sets confirms the models neither overfitting or underfitting and generalizes well on new data.

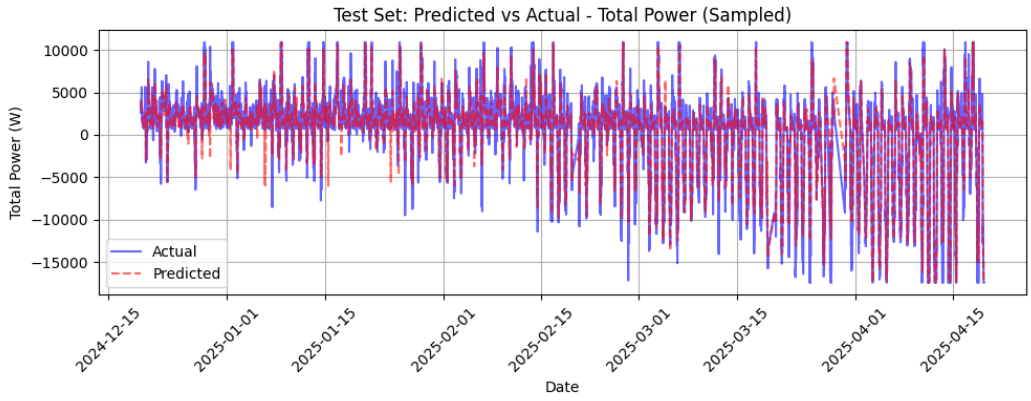


Figure 4 Test set predicted vs actual RFR

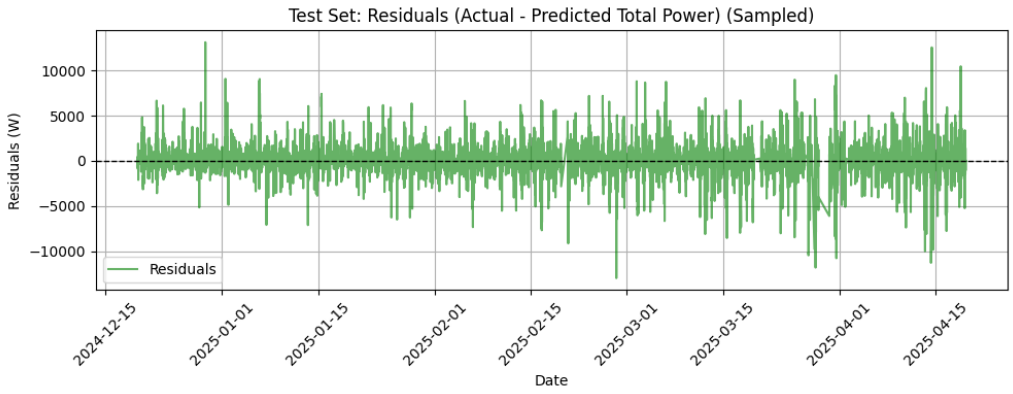


Figure 5 Test set residuals RFR

The training set predictions closely follow actual values, capturing daily and longer-term patterns. The test set shows similar alignment, with minor deviations, confirming the model's ability to generalize. These plots demonstrate effective modeling of temporal trends in power consumption. The plots next to each prediction, display residuals (actual minus predicted total_power) for the training and test sets. Test residuals are slightly larger, reflecting higher errors, but remain randomly distributed. The absence of clear temporal trends in residuals suggests the model captures major patterns.

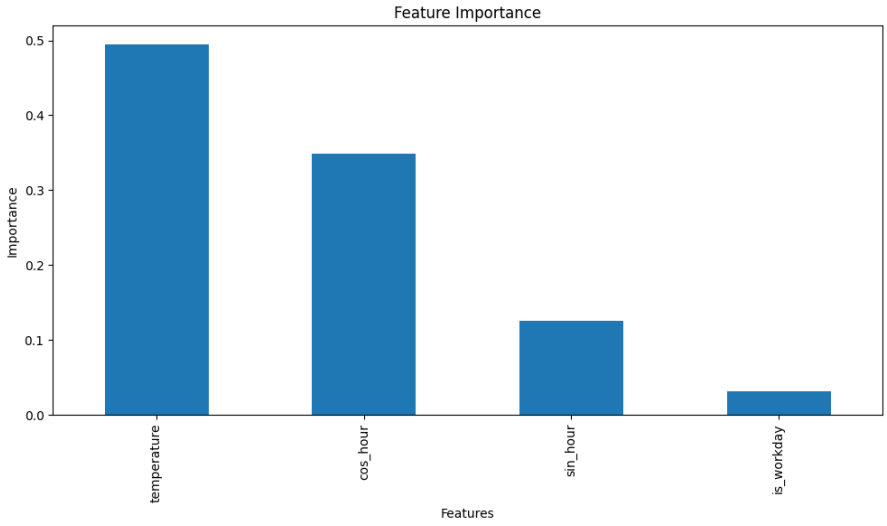


Figure 6 Feature Importance RFR

The figure above shows feature importance with temperature at almost 0.5 and cos of hour at 0.35 and sin of hour at 0.15. Temperature is the most influential impact, reflecting strong cycles in power consumption.

Hyperparameter optimization:

GridSearchCV identified the best parameters isted in table 4. With cross-validation RMSE of 0.327.

Table 6 Random Forest Hyperparameters after optimization

Hyperparameter	Value
'max_depth'	None
'max_features'	None
'min_samples_leaf'	1
'min_samples_split'	5
'n_estimators'	200

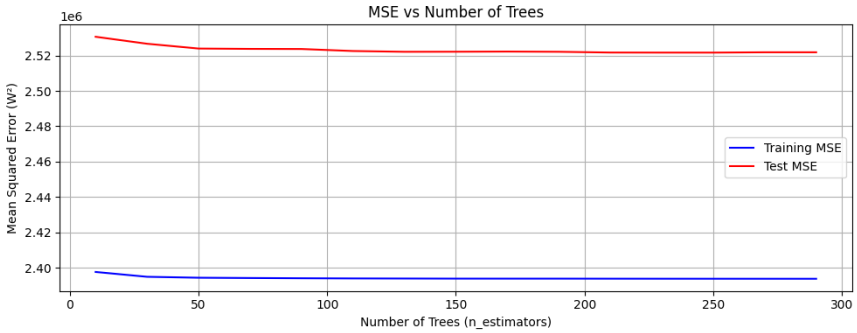


Figure 7 Number of trees reflecting the MSE

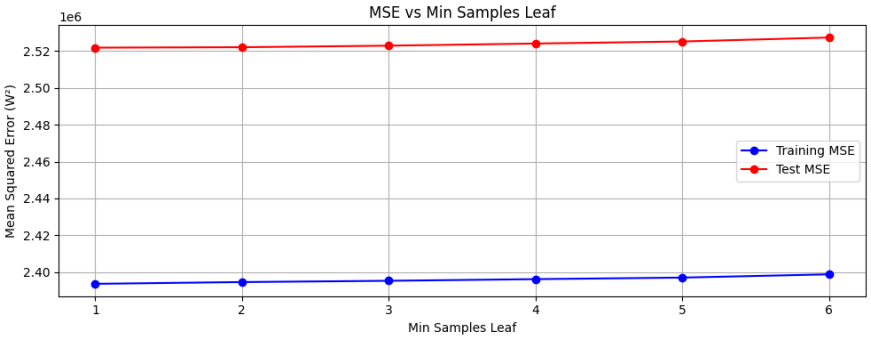


Figure 8 Samples Leaf vs MSE

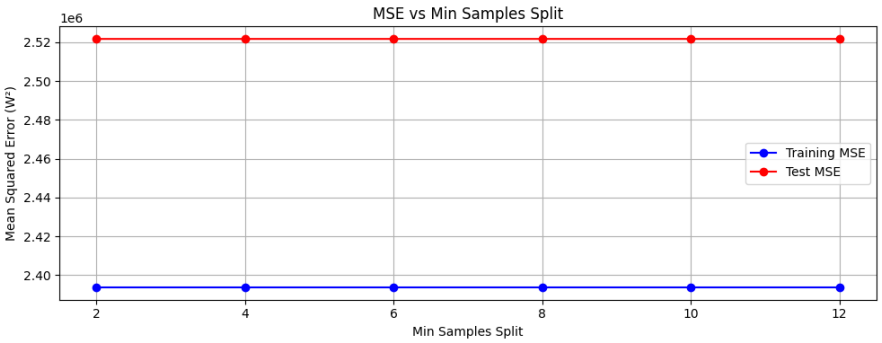


Figure 9 Samples Split vs MSE

From the figure above regarding the number of trees, it is visible that with the higher number, the MSE drops. After 200 trees, the difference in MSE is unsignificant, thus for the sake of computational complexity, 200 trees are chosen. With the increase of leaf samples, the model becomes more complex, but this does not bring any value, and the mean squared error is increasing, thus only 1 leaf sample is chosen. The samples split, does not have much of an influence for both train and test MSE, thus a value in the middle (e.g. 5) is chosen by the grid search.

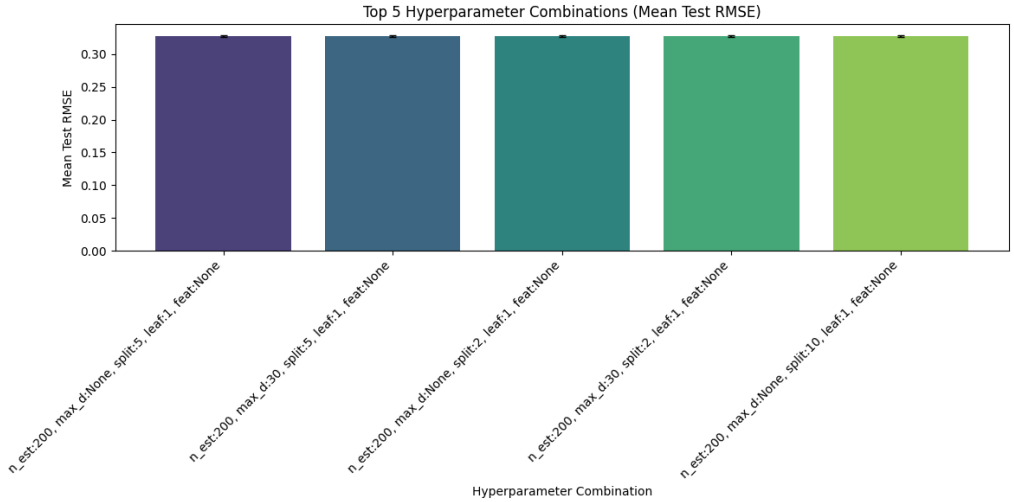


Figure 10 Top 5 Hyperparameters

From the GridSearch top 5 combinations of hyperparameters are plotted above. The combination in purple is chosen as it is the cheapest in terms of computational complexity.

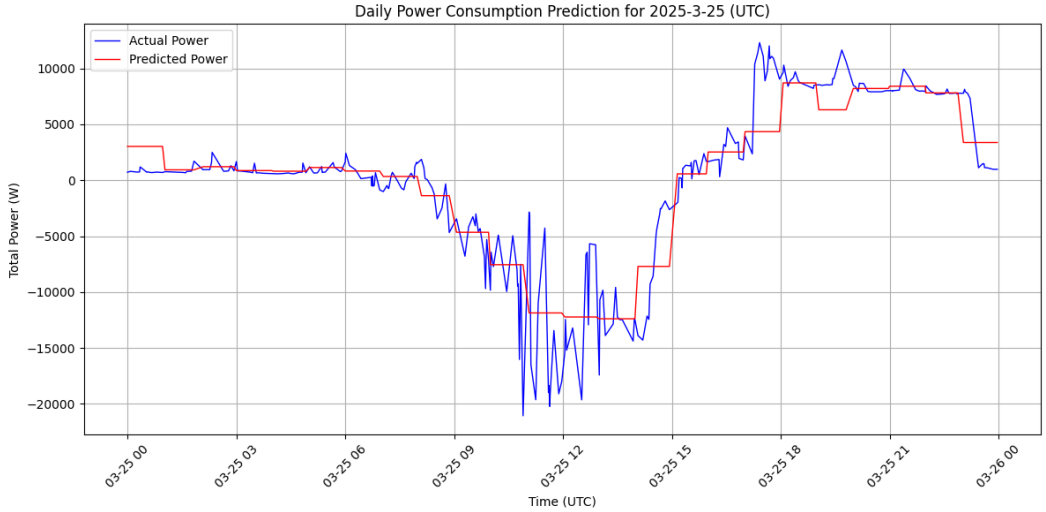


Figure 11 Daily Prediction of RFR for test set date

The Random Forest Regressor accurately predicts total power, with high R^2 (0.897train, 0. 894 test) and low errors (RMSE: 1547.18–1587.98W, MAE: 931.18–950.84 W). The model captures recurring cycles and temperature effects, with optimal hyperparameters balancing complexity and generalization. Limitations include the lack of forecasting (due to random train-test split and no lagged features) and potential for additional features.

4.1.2 CNN & GRU

This chapter presents the results of the preprocessing and training phases for the Convolutional Neural Network-Gated Recurrent Unit (CNN-GRU) model developed to forecast residential power consumption based on the dataset, spanning four months from December 19, 2024, to April 22, 2025.

Data Overview and Initial Visualization

Standardization was applied, transforming temperature and total_power to a mean of 0 and standard deviation of 1. After standardization, the total power is scaled from $\{-30000; 20000\}$ to $\{-65; 4\}$, while temperature is scaled from $\{-5; 20\}$ to $\{-2; 3\}$.

Optimization Results:

The optimization process utilized a sequence generation approach where data was slid every 5 minutes instead of every 1 minute, to subsample the dataset, therefore decreasing the training time. The optimization of the hybrid model was conducted via a random search with 8 runs over a hyperparameter grid. The process utilized 3-fold cross-validation on an 80/20 train-test split, with early stopping on validation MSE to prevent overfitting. The best configuration, identified from Run 4 with learning_rate=0.001, lambda=0.001 and batch_size=128 achieved an average validation loss of 0.1923, outperforming the other runs.

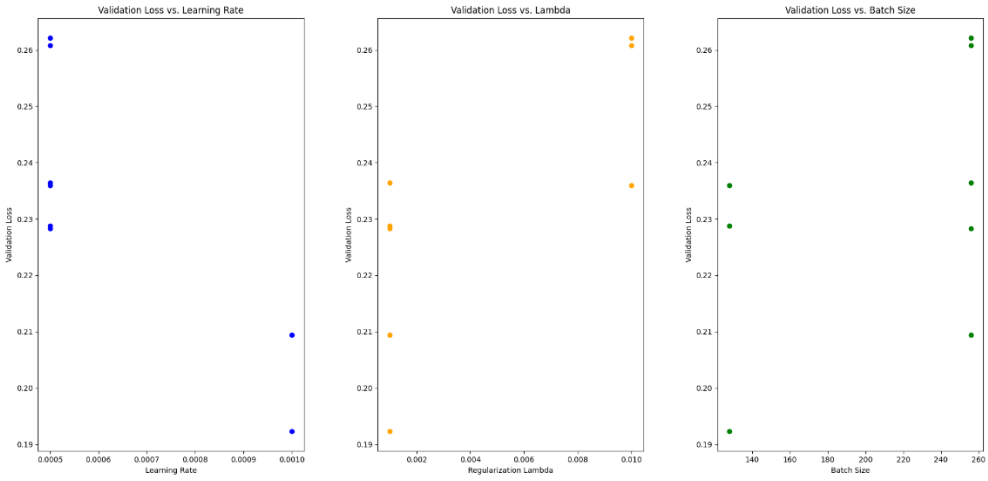


Figure 12 CNN & GRU Hyperparameters against Validation Loss

Neural Network Training Results

The CNN-GRU model was trained on a 70-15-15 chronological split of the preprocessed dataset. The model architecture, comprising two convolutional layers and two GRU layers processed 1 day sequences to predict the next 60-minute average total power. Training utilized a batch size of 128, a maximum of 15 epochs, and the Adam optimizer with a learning rate of 0.001, incorporating L2 regularization (lambda=0.001) and dropout (0.3) to mitigate overfitting. Early stopping with a patience of 2 was employed, monitoring validation loss to restore the best weights.

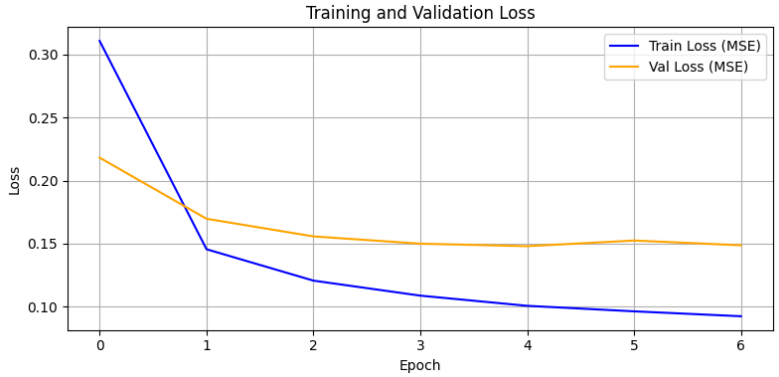


Figure 13 Train and validation cost function

The training process tracked the mean squared error (MSE) as the cost function. A plot of training and validation loss over epochs illustrated the model's convergence, with the training loss decreasing steadily and the validation loss providing insight into generalization. Early stopping halted training before the maximum 15 epochs, indicating the point of optimal performance. The model stopped at the 6th epoch to prevent overfitting.

Table 7 CNN & GRU Metrices Results

Set	RMSE (W)	R^2	MAE (W)
Train	1081.44	0.8946	708.15
Validation	1827.74	0.8792	1134.70
Test	3213.89	0.8385	1865.48

Actual versus predicted power plots were generated for the training, validation, and test sets, providing a visual assessment of model fit. These plots compared the true total power values against the model's predictions, highlighting the model's ability to capture trends and spikes. Metrics including MSE, RMSE, and MAE were computed for each set, quantifying prediction accuracy and error magnitude. The training set plot shows a close fit, while the validation and test sets reveal overfitting. While the test set has a very small mean squared error of just only 0.05, the validation set has 0.13 and the test set is 8 times bigger. The root mean squared error is quite low for the train as well, however the set has a 3 timer larger RMSE. Overall, the performance of the model is quite good on the training set, but it is not able to generalize the test and validation sets. This suggests that the model learns the data and it is overfitting.

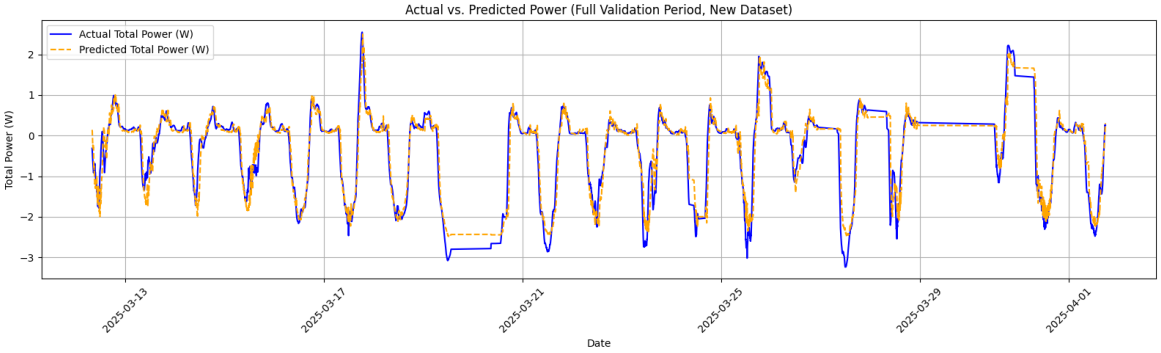


Figure 14Validation set actual vs predicted values

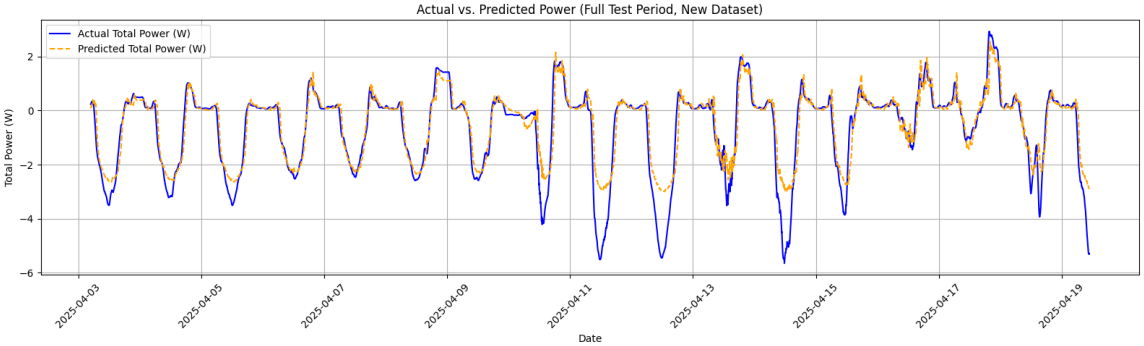


Figure 15 Test Set actual vs predicted

From the above graphs it is evident that the model is performing well on the train set and validation data, but misses a lot of the spikes on the test set.

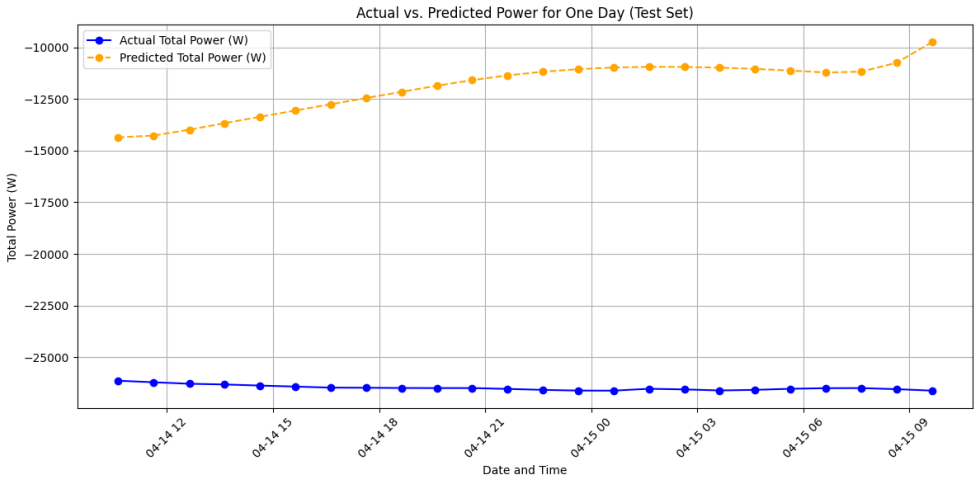


Figure 16 One day prediction of CNN & GRU

The trained model was saved as cnn_gru_model_paper_aligned_new_v6_optimized.keras, ensuring reproducibility and enabling further analysis or deployment.

4.1.3 XGBoost Regression

This section evaluates the performance of a XGBoost Regressor model trained to predict minutely total power consumption (in watts) using the 4-month dataset, which includes power measurements and temperature data. The model was optimized using GridSearchCV, and its performance was assessed using R^2, RMSE and MAE metrics, alongside visualizations of data, predictions, residuals, feature importance, and hyperparameter sensitivity. The results demonstrate the model's ability to capture temporal and environmental patterns in power consumption.

Dataset Visualizations:

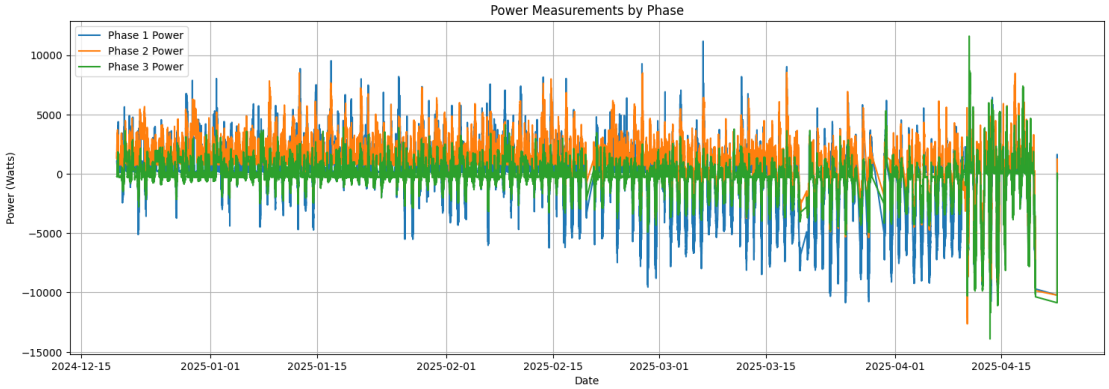


Figure 17: Power phase graphs after preprocessing the signs

Figure 17 shows the power consumption across all three phases after sign correction**.** As instructed by the client, the values for Phase 2 and Phase 3 were sign-inverted prior to April 10, 2025, to correct a wiring issue. After preprocessing, the power signals align more consistently across phases, with Phase 1 and Phase 3 exhibiting similar magnitude and direction, while Phase 2 maintains stable positive values. This preprocessing ensures accurate downstream analysis and model training.

XGBoost and Random Forest Regressor are both tree-based models and do not require feature normalization or scaling. Unlike linear models or neural networks, they split features based on thresholds rather than distances, making them inherently insensitive to the scale or distribution of input features. As a result, normalization has little to no effect on their performance.

4.1.3.1 Model 1: Baseline XGBoost Using Minute-Level Data

The first model was trained using 4 months of minute-sampled data. The feature set included:

- Temperature

- Day of the week (one-hot encoded)
- Hour of the day (encoded as sine/cosine)

The dataset had 165,486 rows and 9 columns, meaning 165,486 training samples and 9 input features per sample.

Table 8: Model 1 Hyperparameters before tunning

Parameter	colsample_bytree	learning_rate	max_depth	n_estimators	subsample	random_state
Value	0.8	0.05	10	150	0.8	42

Performance metrics (pre-tuning):

Table 9 : Model 1 results

Set	**RMSE (W)**	**R^2**	**MAE (W)**
Train	1694.69	0.889	1095.83
Val	1750.51	0.880	1118.90
Test	1734.24	0.884	1113.46

These scores demonstrate strong generalization, with relatively small performance drop from training to validation and test sets. The model shows stable error levels across all splits even before hyperparameter optimization.

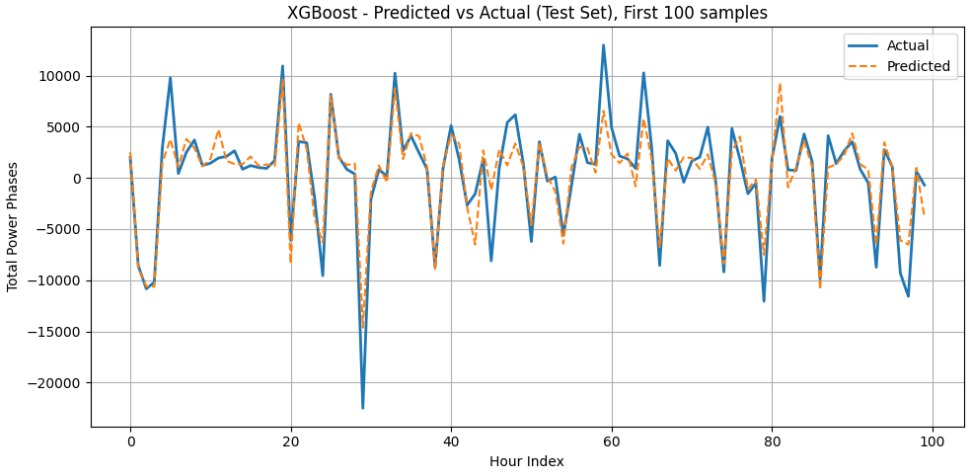


Figure 18 : Predicted vs. Actual total power consumption for the first 100 samples of the test set using the baseline XGBoost model.

Figure 18 illustrates the predicted vs. actual total power consumption over the first 100 samples of the test set using the baseline XGBoost model. The orange dashed line represents the model's predictions, while the solid blue line shows the ground truth.

We observe that the model captures the overall trend and short-term fluctuations in power demand reasonably well. Despite some deviation at the extreme peaks and troughs, the predicted values closely follow the actual signal across most time steps. This indicates the model's ability to generalize effectively to unseen data and justify its reported R^2 of 0.884 on the test set.

Hyperparameter tunning:

Following a comprehensive GridSearchCV procedure, the XGBoost model was tuned using 5 key hyperparameters.

Table 10: Model 1 Hyperparameters after tunning

Parameter	colsample_bytree	learning_rate	max_depth	n_estimators	subsample	random_state
Value	1.0	0.2	10	200	1.0	42

This tuned model improved performance across all data splits:

Table 11: Model 1 results + Hyperparameter tunning

Set	RMSE (W)	R^2	MAE (W)
Train	1477.40	0.916	871.30
Val	1548.26	0.906	897.69
Test	1523.48	0.911	894.51

Compared to the baseline model, the tuned version significantly reduced error and increased the model's ability to generalize across unseen data.

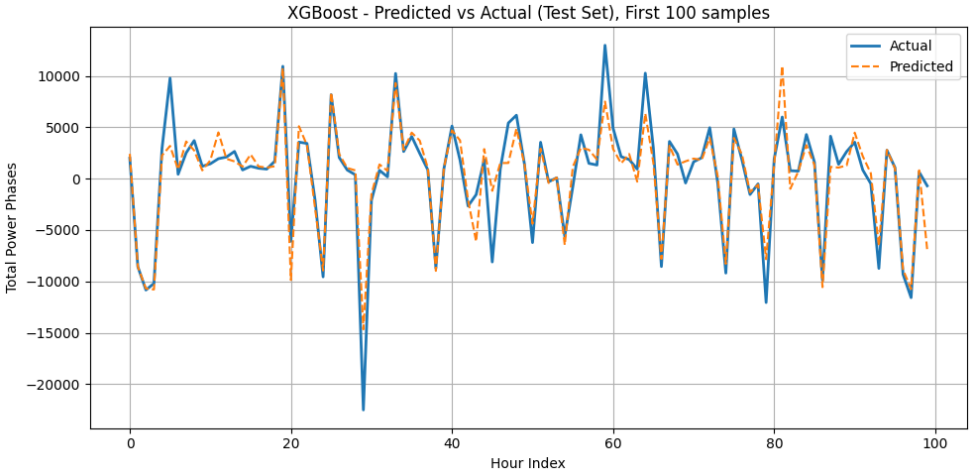


Figure 19 :Predicted vs. Actual power consumption (first 100 samples) on the test set using the tuned XGBoost model.

The predictions closely follow the real values with reduced deviation around peaks and dips, indicating a clear improvement in model responsiveness and accuracy.

4.1.3.2 Model 2: XGBoost with Hourly Averaged Data

In this version, we changed the sampling approach by replacing minute-level data with the average of 60 samples per hour. This results in a dataset with 2789 rows × 9 columns, drastically reducing the size and temporal resolution.
Feature set: Same as previous models — including temperature, hour (sin/cos), and day of the week (one-hot encoded).
Sampling change: Removed minute-wise entries, retaining only one aggregated value per hour.
Hyperparameters after grid search:

Table 12 : Model 2 Hyperparameters

Parameter	colsample_bytree	learning_rate	max_depth	n_estimators	subsample	random_state
Value	1.0	0.05	3	100	0.8	42

Performance Metrics:

Table 13 : Model 2 results + Hyperparameter tunning

Set	RMSE	R^2	MAE
Train	2838.23	0.675	1881.63
Val	3106.06	0.590	2031.51
Test	3121.58	0.599	2127.76

This performance indicates a clear drop in model accuracy compared to minute-level sampling. The model struggles to capture high-frequency variations, resulting in lower R^2 and higher RMSE and MAE values across all data splits.

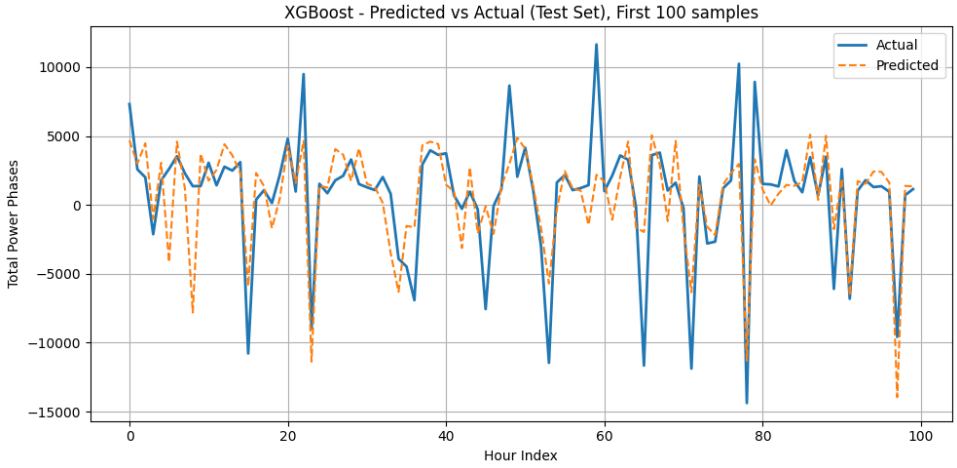
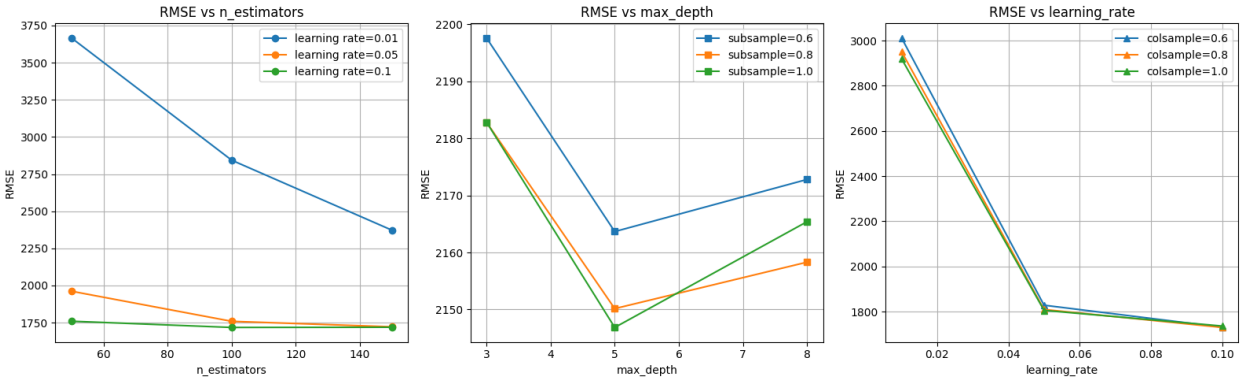


Figure 20 : Predicted vs. Actual total power consumption for the first 100 test samples using XGBoost with hourly-averaged data.

The predictions follow the general trend but fail to capture rapid fluctuations and sharp troughs, demonstrating the limitations of lower-resolution sampling in high-variance environments.

4.1.3.3 Model 3: Lag and Rolling Features with Hourly Sampling

To improve upon the performance drop observed in Model 2 (due to switching from minute-level to hourly-averaged sampling), Model 3 incorporates additional temporal features including lag and rolling statistics. These features give the model temporal awareness and compensate for the reduced resolution.
Feature set used (15 columns total):
['temprature', 'hour_sin', 'hour_cos','power_mean_last_3h', 'power_std_last_3h', 'temp_mean_last_3h', 'power_lag_1h', 'power_lag_2h', 'temp_lag_1h', + one-hot encoded day_of_week columns]
Dataset size: 2786 rows × 15 columns
Hyperparameter Exploration:
Before tuning, we investigated the relationship between RMSE and key parameters using grid search results:



- Learning Rate & Colsample:
  The learning rate has a significant influence on RMSE, while colsample_bytree shows only minor sensitivity. Since RMSE stabilizes around a learning rate of 0.1 or higher, and changes little with colsample values, we chose 0.1 as a reliable and efficient learning rate.
- Max Depth & Subsample:
  Changes in max depth led to relatively small RMSE differences (around 20 W), and subsample has negligible impact across configurations. To balance accuracy and computation time, we selected a shallower tree (max_depth = 5) with a subsample of 0.8, which offers efficient training with stable results.
- n_estimators & Learning Rate Interaction:

RMSE is highly sensitive to the combination of estimators and learning rate. While a higher learning rate like 0.1 performs well, it leads to higher RMSE when combined with more estimators.

As a result, we opted for a lower learning rate of 0.05 paired with 100 estimators, which reduced overfitting risk while maintaining accuracy and reducing training time.These plots guided the narrowing of grid search ranges, helping reduce training time while maintaining performance.

After investigating these numbers, the hyperparameter tunning using grid search become:

Table 14: Model 3 Hyperparameters

Parameter	colsample_bytree	learning_rate	max_depth	n_estimators	subsample	random_state
Value	0.6	0.05	**5**	100	0.8	42

Model performance:

Table 15 : Model 3 performace

Set	**RMSE**	**R^2**	**MAE**
Train	1295.38	0.935	838.06
Val	1510.69	0.892	1014.27
Test	1768.77	0.863	1042.24

- An RMSE of ~1768.77 Watts translates to roughly 1.77 kWh per hour, which is reasonable considering residential systems may peak well above 10–12 kWh depending on equipment.
- MAE (1042.24 W or ~1.04 kWh) indicates the average error is within acceptable forecasting limits, especially without future external inputs like weather forecasts.
- R^2 score of 0.863 confirms the model explains ~86% of variance in unseen data — a strong performance considering only short-range historical features are used.

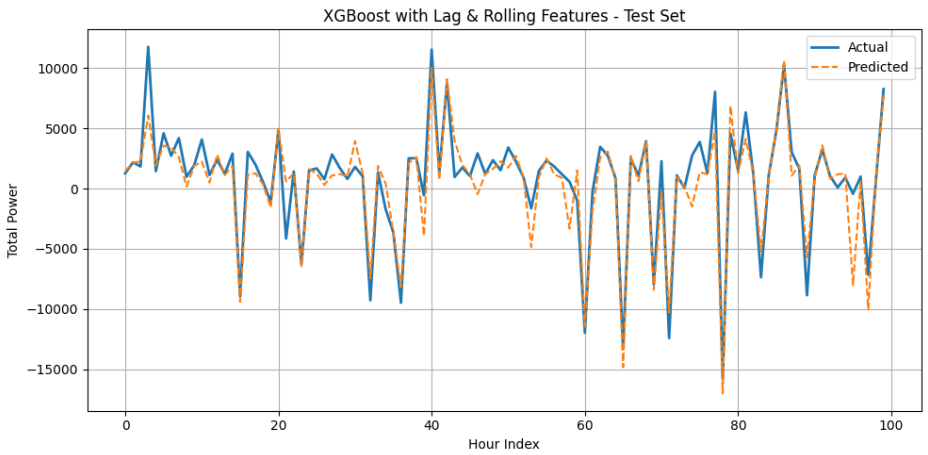


Figure 21 : Model 3 predicted vs actual data

Also to investigate that the features we add has how much influence on the model we used feature importance method:

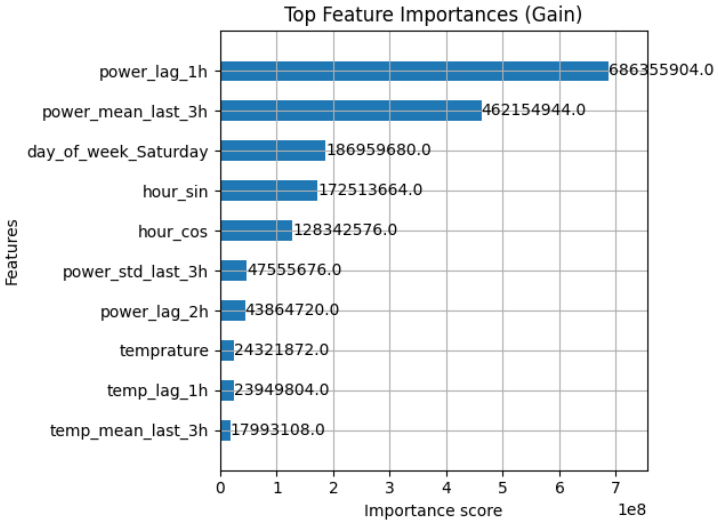


Figure 22 : Feature Importance using Model 3 with Lag and rolling features

Feature importance analysis shows the most influential variables were:

- power_lag_1h and power_mean_last_3h — confirming the value of giving the model access to past behavior.
- Temporal features like hour_sin and day_of_week had secondary importance, which supports daily and weekly cyclicity in power demand.

This validates the decision to use lag and rolling statistics to reintroduce short-term dynamics lost during hourly down sampling.
Model 3 demonstrates that even with a smaller, hourly-aggregated dataset, model performance can be restored and even improved by smart feature engineering. The addition of lag and rolling window features proved critical, both statistically (via importance ranking) and in predictive accuracy. RMSE and MAE values are within an interpretable and reasonable energy range, and the model provides a reliable tool for forecasting residential electricity demand at an hourly level.

4.1.3.4 24-Hour Forecast Evaluation (Test Set)

To assess the model's generalization ability on unseen data, a random 24-hour window was selected from the test set. The model's predictions were compared to actual power consumption during this period.

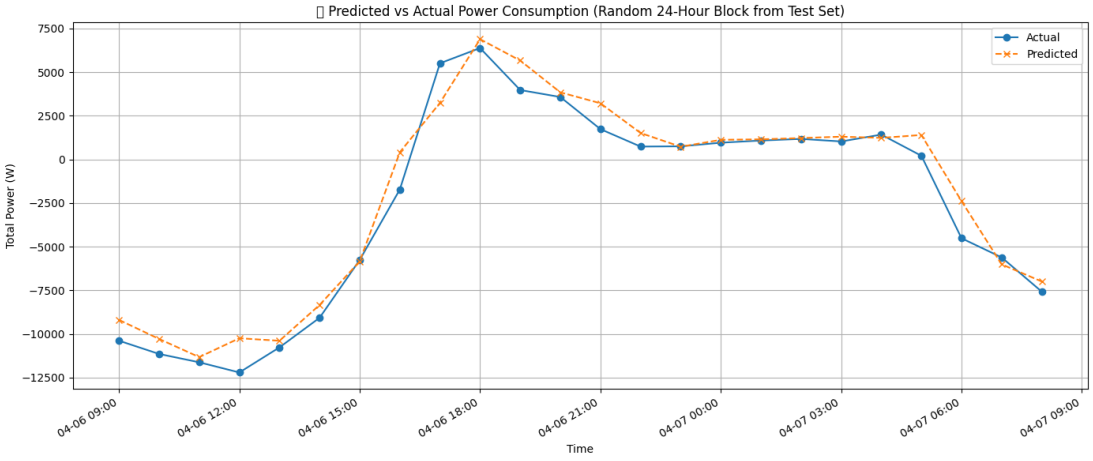


*Figure 23 : Actual vs. predicting a random 24 -*hours *in test dataset*

As shown in Figure 23, the model successfully captures the overall consumption pattern:

- It accurately predicts the rise in consumption during late afternoon and evening**.**
- It also follows the decline during nighttime and early morning hours.

- Some deviation is observed in peak magnitude, particularly around 18:00–21:00, likely due to volatility not reflected in lagged inputs.

Visually, the model tracks the dynamics of the system well, preserving peak positions**,** trend direction, and level shifts**.**

This confirms the model's capacity to deliver robust short-term predictions, which is essential for real-time energy planning and demand response applications.

4.2 Heat Demand

This section presents the results of the hourly and daily heat demand prediction model developed for residential buildings.

The model is based on an XGBoost regressor and Neural Network trained on time series features derived from four months of hourly data. The goal of the model is to predict hourly and daily heating energy demand, leveraging temporal patterns, weather conditions, and recent consumption history.

4.2.1 Hourly Heat Demand Models

4.2.1.1 Hourly Heat Demand - Model 1

The first model was developed using a relatively simple feature set with minimal preprocessing to assess whether a straightforward structure could yield reliable results. The feature set included:

Time features: hour, hour_squared, day_of_week, dayCalendar indicators: week_end, non_working_hour and Temperature-related features: temperature, temp_squared . These features were selected to capture non-linear hourly patterns, weekday/weekend effects, and basic thermal relationships with outdoor temperature.

Model Performance:

After hyperparameter tuning using the validation set, the best parameters identified were:

Best Hyperparameters: {n_estimators: 400, max_depth: 7, learning_rate: 0.02, subsample: 0.8}

Table 16 Metrices Results-Model 1

Metrics	Training	Validation	Test
RSME (kW)	3.56	9.8389	10.8179
CVRSME	1.12%	3.09 %	3.39 %
R^2	0.98	0.87	0.85

The R^2 values of 0.88 (validation set) and 0.86 (test set) indicate that the model explains 85% of the variation in hourly heat demand, good performance for a relatively basic model. The RMSE values (~10 kW) show that, on average, the model's predictions deviate from the actual values by about 10 kilowatts, which is reasonable given the complexity of the task. CVRMSE values below 4% further suggest that the prediction errors are low relative to the average demand, confirming that the model generalizes well.

4.2.1.2 Prediction Visualization and Feature Importance:

Figure 24 shows the actual vs. predicted hourly heat demand over the first 100 hours of the test set. The predicted values closely follow the actual values, indicating that the model captures short-term variability effectively. Although minor under- and over-estimations are present, there is no evident systematic bias.

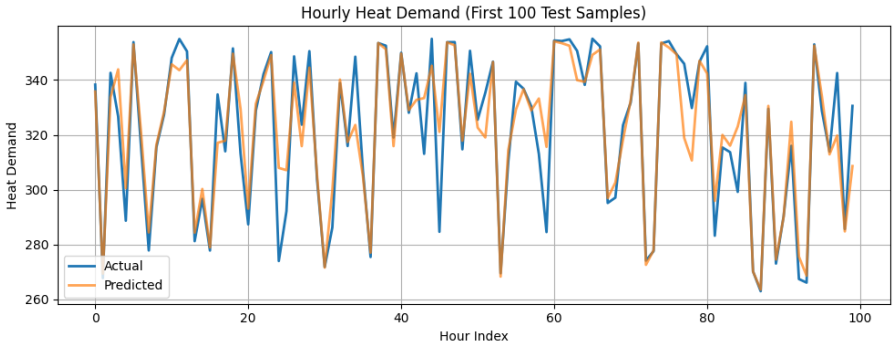


Figure 24 Actual vs Predicted Hourly Heat Demand-Model 1

Figure 25 shows the bar chart of feature importance. The top three most important features were:

- day: the day of the month can implicitly capture long-term patterns, particularly when the dataset spans multiple months. Heating demand often increases as the month progresses during colder periods, and early-month behavior may differ from mid- or late-month usage due to evolving weather conditions. Thus, this feature is valuable for representing seasonal trends. It contributes approximately 35% of the model's decision-making weight, meaning that in about 35% of the splits made by XGBoost's decision trees, the day feature was selected to reduce prediction error.
- temperature: As expected, this was the second most important feature (~32% importance), since heat demand is closely tied to outdoor temperature fluctuations.
- day_of_week: This variable captured behavioral differences between weekdays and weekends, contributing meaningfully to the model's predictions.

Additional features such as hour, non_working_hour, and hour_squared showed moderate importance, confirming that time-of-day patterns also influence heat demand. Engineered features like temp_squared helped capture the non-linear relationships between temperature and heating needs.

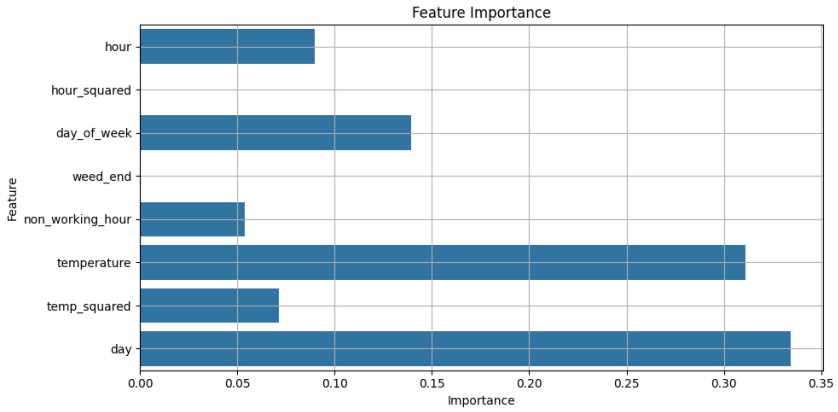


Figure 25 Feature Importance of Hourly Heat Demand-Model 1

Residual Analysis:

Figure 26 presents the residual plot, showing the difference between actual and predicted values. Residuals are generally centered around the zero line, indicating that predictions are on average unbiased. However, a moderate increase in residual spread is observed in the mid-range of predicted values (approximately 300–330 kW), suggesting prediction errors vary slightly across different demand levels. Importantly, the residuals show no clear structural patterns, which implies that the model's functional form is appropriate and that no major systematic errors remain unaccounted for.

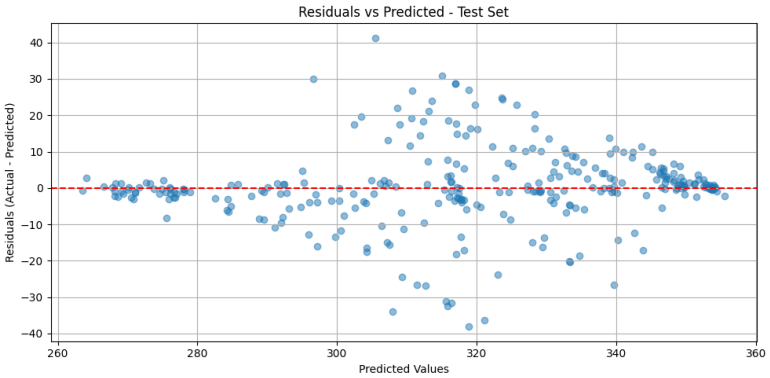


Figure 26 Residuals of Hourly Heat Demand-Model 1

4.2.1.3 Hourly Heat Demand – Model 2

In this second hourly heat demand prediction model, we added the month feature to examine its impact on model performance. The model was trained using XGBoost, with additional engineered features such as hour_sin, hour_cos, is_morning, and is_evening to better capture cyclical and behavioral temporal patterns.

Table 17 Metrices Result- Model 2

Metrics	Training	Validation	Test
RSME(kW)	5.79	6.95	6.89
CVRSME	1.81%	2.19 %	2.17 %
R^2	0.95	0.93	0.94

Compared to the first model (which had a test R^2 of 0.857 and an RMSE of approximately 10.8), this model shows significant improvements in both error reduction and variance explanation. The slightly higher R^2 on the test set relative to the validation set is likely due to random data splitting, differences in distribution between subsets or the presence of cleaner or more predictable patterns in the test data may explain this discrepancy.

The figure below shows the actual vs. predicted hourly heat demand for a sample of test instances (first 100 samples). The predicted values closely track the actual demand trends, showing that the model captures both sharp fluctuations and general patterns with relatively high accuracy. This reflects strong learning of temporal structure and low prediction noise.

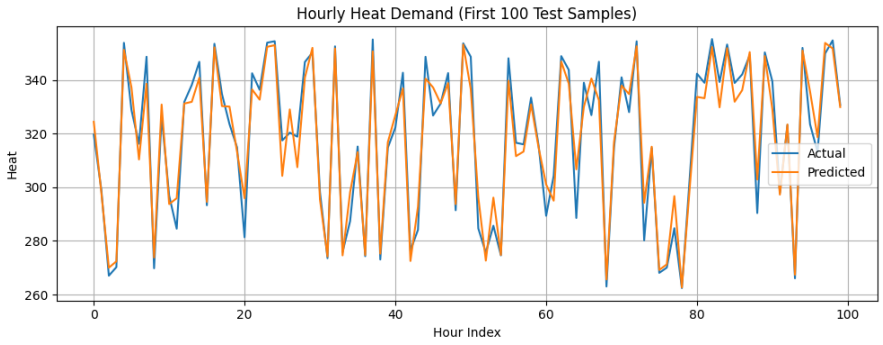


Figure 27 Actual vs Predicted Hourly Heat Demand -Model 2

The feature importance plot for this model reveals that the month feature overwhelmingly dominates the model's decision-making, accounting for over 85% of total importance. This indicates that seasonal variation from January to April plays a more prominent role in determining heat demand than even temperature. Features like temperature, hour, and non_working_hour contribute only marginally. While this dominance suggests the model is effectively leveraging seasonal trends, it also implies that the model may be learning temporal progression rather than physical drivers such as temperature or occupancy. Consequently, although the model

performs well for short-term forecasting within the same time range, its generalization to unseen months or future years may be limited.

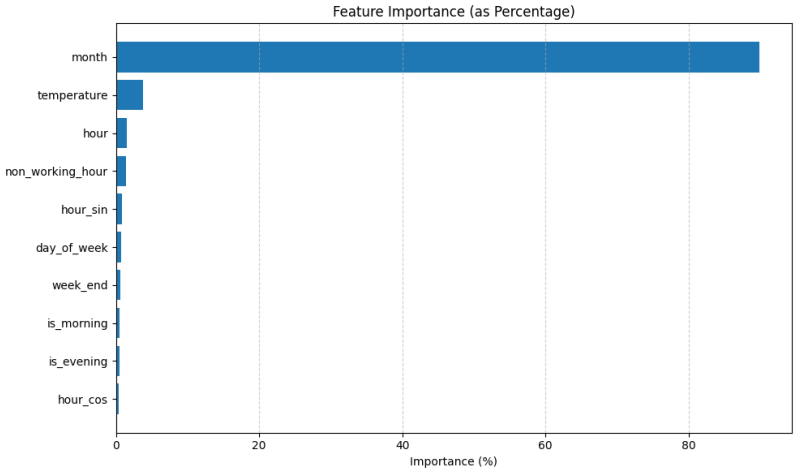


Figure 28 Feature Importance of Hourly Heat Demand-Model 2

The residual plot (Figure 29) supports these findings. Residuals are distributed around zero, indicating minimal bias and no strong structural error. Most residuals fall within the range of approximately –15 to +15, which is relatively small compared to the full heat demand range (approximately 260–350 kW). This further confirms that the model fits the data well, with low bias and low variance, and generalizes effectively within the observed period.

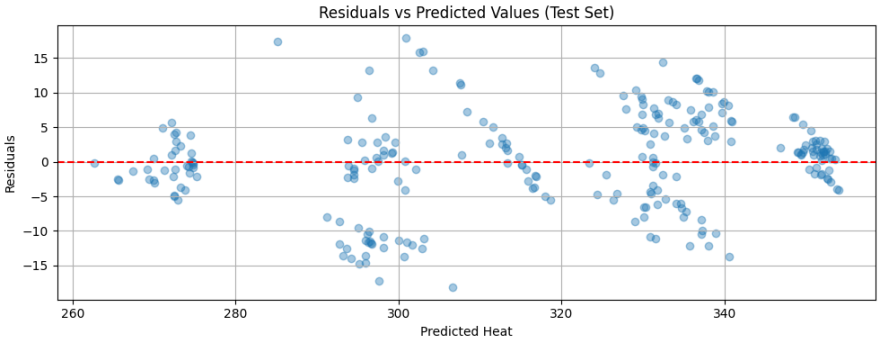


Figure 29 Residuals of Hourly Heat Demand-Model 2

4.2.1.4 Hourly Heat Demand-Model 3

The previous models provided appropriate accuracy, but they are highly dependent on date. Since we have data for only 4 months, the temperature pattern might change in the next year. As a result, the model should be independent of date (month number and day of month).
In (Etienne Saloux, 2018), several models were developed for prediction of heat demand using hour, temperature, and binary weekend features, including Artificial Neural Network (ANN), which showed good prediction accuracy. However, that article used data spanning 2.5 years with 10-minute intervals.

In this section, an ANN model was developed for heat demand prediction. Then, the model hyperparameters, including network architecture, ensemble size, and learning rate, were investigated to optimize the network.
The effects of hyperparameters on the network performance are depicted in Figure 30 in terms of RMSE.

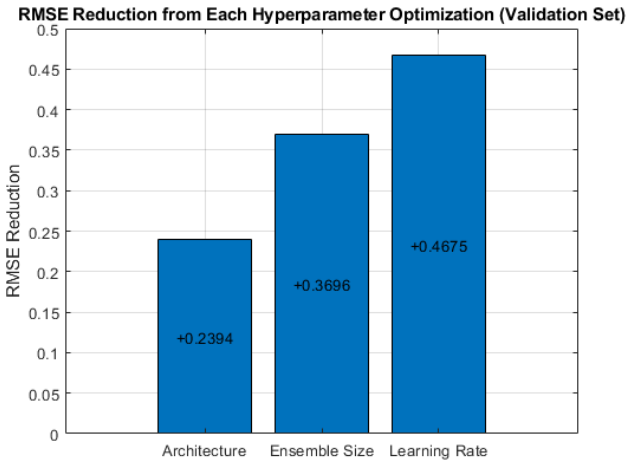


Figure 30 The effect of hyper parameters on the network performance-Model 3

According to Figure 30, learning rate is the most influential hyperparameter on the model performance. The RMSE variations against the hyperparameter changes are shown in Figure 31.

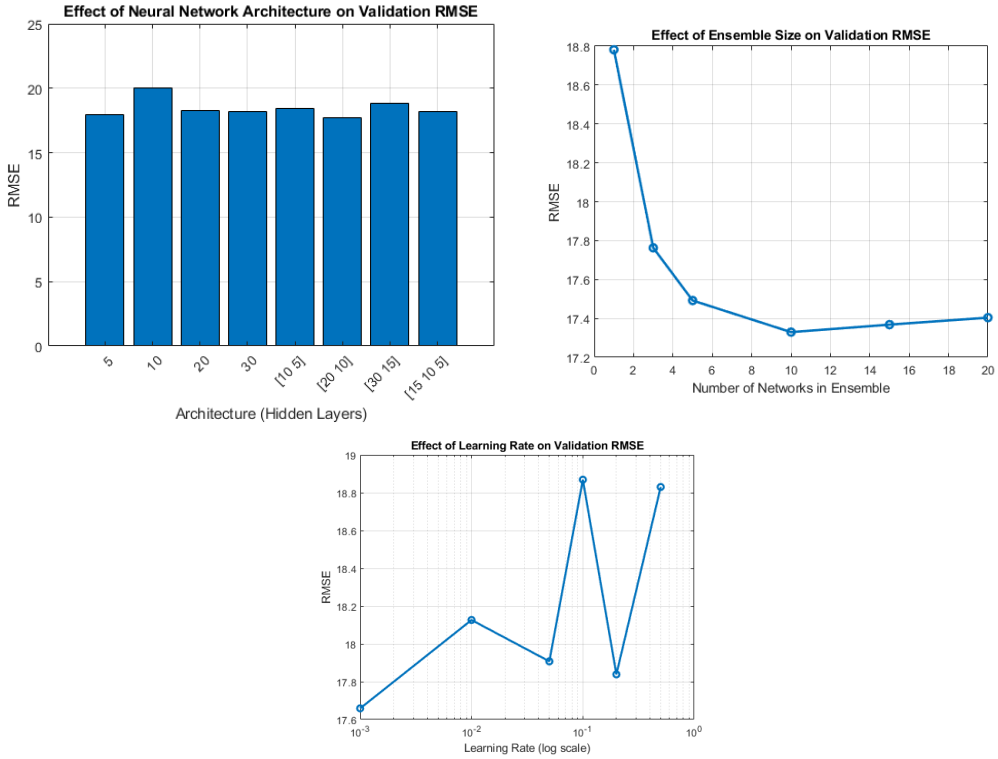


Figure 31 Hyper Parameter variation effects on RSME-Model 3

the best set-up for the network was obtained as:

- Neural Network Architecture: 2 hidden layers with 20 and 10 neurons, respectively
- Ensemble Size: 10 (Val RMSE = 17.3293)
- Learning Rate: 0.0010 (Val RMSE = 17.6590)

Therefore, the ANN network was developed in accordance with the mentioned values. The predicted and actual heat demand for the test set, as well as the residuals, are illustrated in Figure 35.

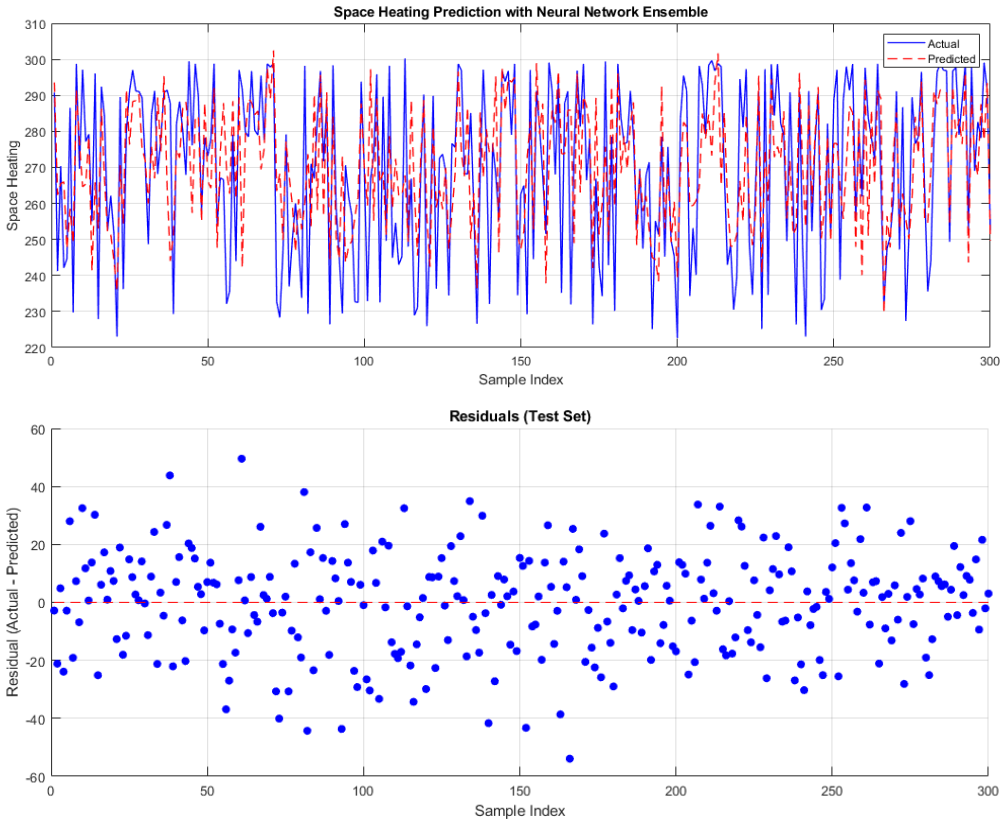


Figure 32 predicted heat demand and residual for the test set- Model 3

The results for the training, validation and test set are represented as:

Tabel 1 ANN metrics result

Metrics	Training	Validation	Test
RSME (kw)	15.0517	17.3416	17.5504
CVRSME	5.5 %	6.3%	6.5%
R^2	0.6033	0.4928	0.4440

The model accuracy apparently reduces in comparison with the previous models, but it can obtain more robust results for different years. It is obvious that the training data are insufficient for training the neural network to obtain accurate results. However, according to the ASHRAE standard for HVAC applications (Etienne Saloux, 2018), for hourly heat demand prediction, models with CV-RMSE below 15% are acceptable.

4.2.1.5 Hourly Heat Demand-Model 4

The fourth Model is designed based on behavioral (e.g., non-working hours), weather-related features, and Lag & roll features. The primary focus is on learning short-term patterns to produce accurate hourly forecasts.

The optimal hyperparameters were identified through a grid search:

learning_rate = 0.04, max_depth = 3, n_estimators = 300, subsample = 0.7, colsample_bytree = 1.0

Table 18 Metrices Results-Model 4

Metrics	Training	Validation	Test
RSME (kW)	1.09	7.19	7.68
CVRSME	0.34%	2.33%	2.54%
R^2	0.99	0.941	0.940

The model explains over 94% of the variation in hourly heat demand and exhibits nearly identical performance on both validation and test sets, indicating neither overfitting nor underfitting. With an RMSE in the range of 7.2–7.7 kW, the model's predictions deviate minimally from actual values. The CVRMSE values (~2.3–2.5%) confirm the model's high precision relative to the scale of heat demand.

Prediction Visualization and Feature Importance:

Figure 33 shows the actual vs. predicted heat demand for the first 100 hours of the test set. The model effectively captures the overall shape and periodic structure of hourly heating cycles. It tracks peaks and troughs well, although some amplification is observed, particularly around sharp demand changes this shows occasional overestimation of variability. There is also a segment (around hour 45–60) where the model underpredicts demand, likely due to reliance on heat_lag_1 and heat_roll_2, which smooth past patterns. When a new spike is not reflected in the prior 1–2 hours, the model may fail to anticipate it. This highlights a limitation of using short-term memory features in rapidly changing conditions. The shallow tree depth (max_depth = 3) may also contribute, as deeper trees were prone to overfitting during experimentation. Despite these limitations, the model demonstrates strong alignment with actual demand and exhibits high accuracy in short-term hourly forecasting.

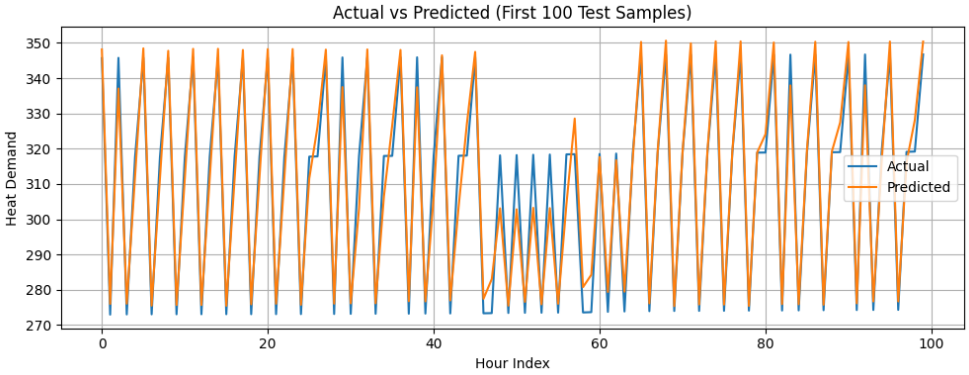


Figure 33 Actual vs Predicted Hourly Heat Demand-Model 4

From figure 34 the feature importance ranking (based on gain contribution) is as follows:

- heat_lag_1: Most dominant, contributing ~45% of total gain.
- temperature: Approximately 32%, underscoring the role of weather in heating behavior.
- heat_roll_2: Around 17%, showing the value of smoothed recent demand.

Time-based features such as hour, hour_sin, and is_morning contributed less, indicating that recent historical demand and temperature were more informative than time-of-day patterns. This reliance is consistent with typical residential heating behavior, which is heavily influenced by both prior consumption and outdoor temperature.

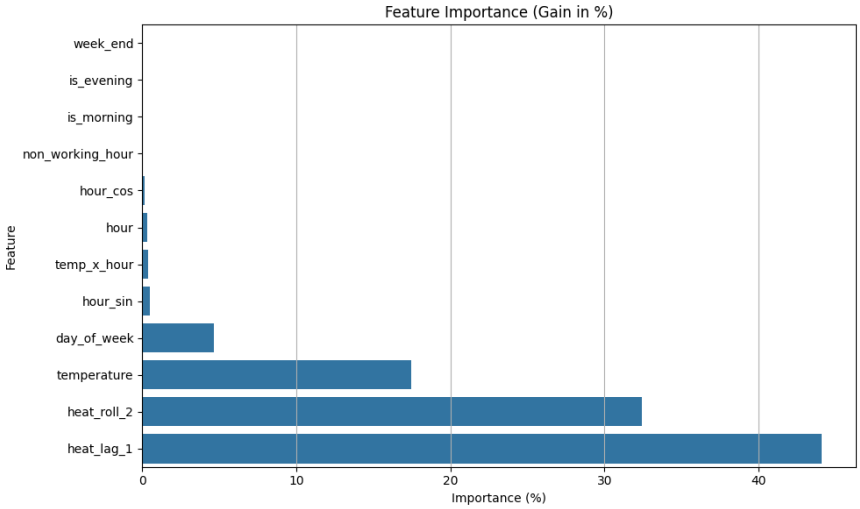


Figure 34 Feature Importance of Hourly Heat Demand-Model 4

Hyperparameter Analysis:

Figure 35 displays the heat map of RMSE across different values of learning_rate and max_depth. The lowest RMSE (7.907) occurred at learning_rate = 0.05 and max_depth = 3, confirming the optimal configuration found during grid search. As tree depth increased (e.g., depth = 5) or learning rate decreased, RMSE rose, indicating higher variance and potential overfitting or failure to converge.

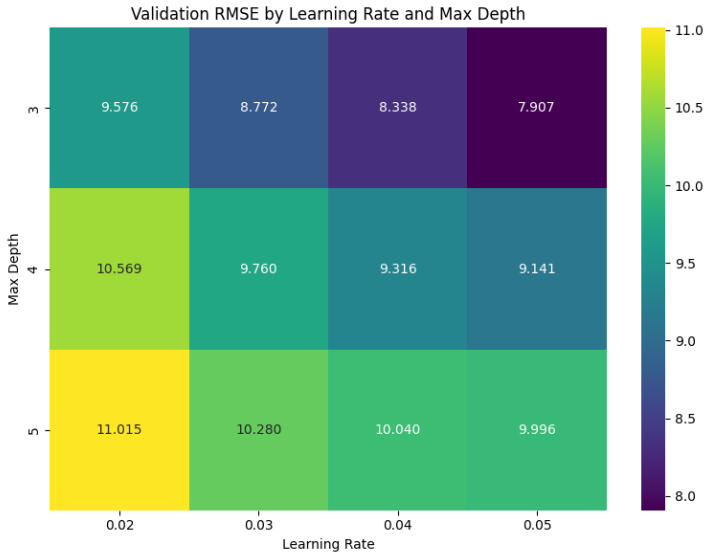


Figure 35 Heatmap: RMSE by Learning Rate and Max Depth-Hourly Heat Demand-Model 4

The figure 36 shows the box plot illustrating the impact of learning rate and maximum tree depth on RMSE values. The plot reveals that RMSE steadily decreases as the learning rate increases from 0.02 to 0.05. Shallow trees (with max_depth = 3) performed best, producing the lowest median RMSE and fewer outliers. In contrast, deeper trees introduced greater variance and higher prediction error, as also observed in the heat map, supporting the selection of max_depth = 3. These results indicate that a moderate learning rate combined with shallow trees yields more accurate and stable performance in hourly heat demand prediction. The dots above the boxes represent outlier RMSE values, corresponding to individual model runs (specific hyperparameter combinations) where RMSE was unusually high compared to others with the same tree depth.

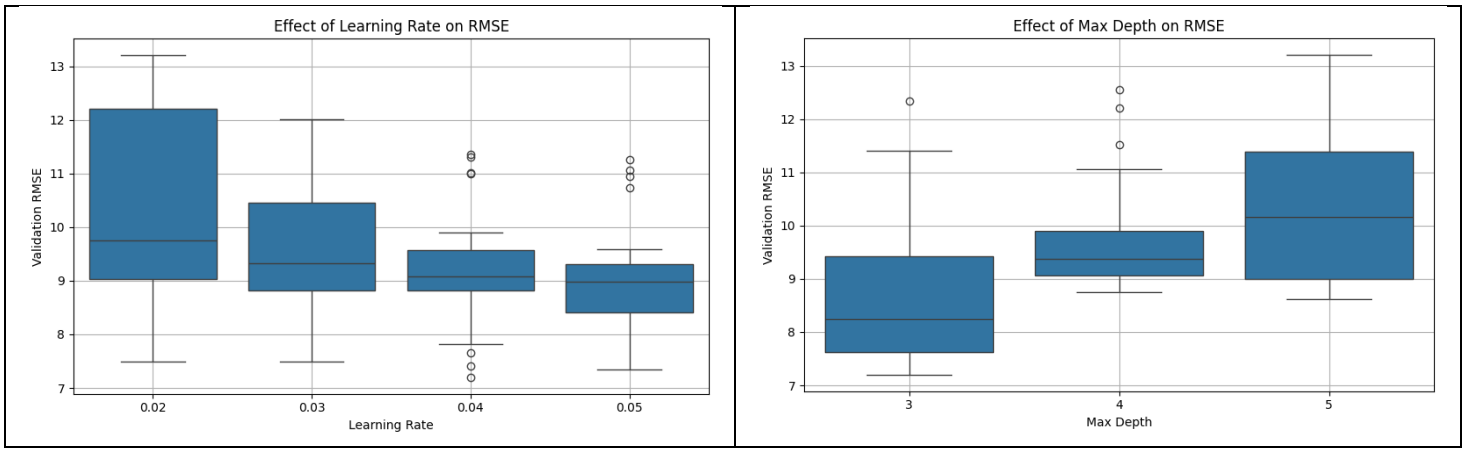


Figure 36 Boxplots: Learning Rate & Max Depth- Hourly Heat Demand-Model 4

4.2.2 Daily Heat Demand Model:

This model aims to predict the daily heat demand of a residential building using weather data (e.g., daily temperature), calendar-based variables (such as weekend indicators), and engineered features (including 3-day temperature averages and the previous day's heat demand). As discussed in the methodology section, the model was trained to predict the change in daily heat demand (heat_delta), and the final heat demand value was reconstructed by adding this predicted change to the previous day's demand.

Table 19 Metrices Results Daily Heat Demand Model

Metrics	Training	Validation	Test
RSME (kW)	0.36	0.44	0.39
CVRSME	0.115%	0.128%	0.112%
R^2	0.97	0.887	0.875

The model explains 88.7% of the variance in the validation data. The RMSE of 0.448 indicates that, on average, predictions deviate by approximately 0.469 kW, a very low error magnitude. The small CV-RMSE value further shows that the prediction error is minimal relative to the scale of daily heat demand.

On unseen data, the model explains 87.5% of the variance, and the RMSE and CV-RMSE values remain low, confirming that the model maintains strong predictive accuracy. The average prediction error is less than 0.1% of the daily heat demand, which demonstrates a high level of precision. Given that our primary evaluation metric is RMSE, we conclude that the model generalizes well, showing consistent performance on both validation and test sets.

The reason the RMSE is lower on the test set than on the validation set is that RMSE solely measures absolute prediction error, regardless of data variability. In contrast, R^2 evaluates model performance relative to a baseline model that always predicts the mean. If the test set contains less variability in actual demand, R^2 may appear lower even when RMSE is low. This suggests the model predicts accurately but cannot fully capture the variation in a less diverse test dataset.

Prediction Visualization and Feature Importance:

The following plot displays the actual vs. predicted daily heat demand over the test period. As shown, the model closely follows the true demand trend, successfully capturing short-term patterns. While the overall direction is well preserved, slight overestimations are observed during days 9–10. Despite this, the prediction remains accurate, with low residual error and a strong alignment to the actual values.

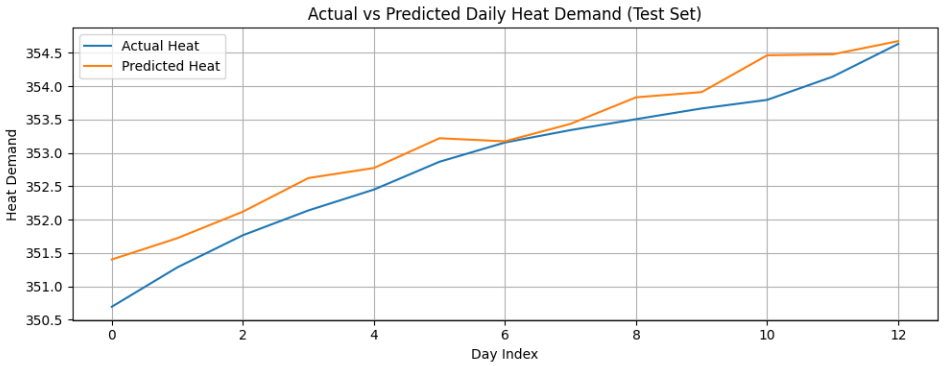


Figure 37 Actual vs Predicted Daily Heat Demand Model

The figure below shows the feature importance bar chart. As presented, the daily maximum temperature (temp_max) is by far the most influential feature, reducing prediction error more than any other variable. The average temperature (temp_mean) also plays a significant role, contributing approximately 0.04 to the model's gain. In contrast, features such as temp_min, day_of_week, and non_working_hour have similar but relatively minor contributions. Other variables, including temp_ma_3 and week_end, exhibit negligible impact.This shows that the model relies primarily on high daytime temperatures to make predictions, which is consistent with real-world behavior where heating demand decreases during warmer daytime periods.

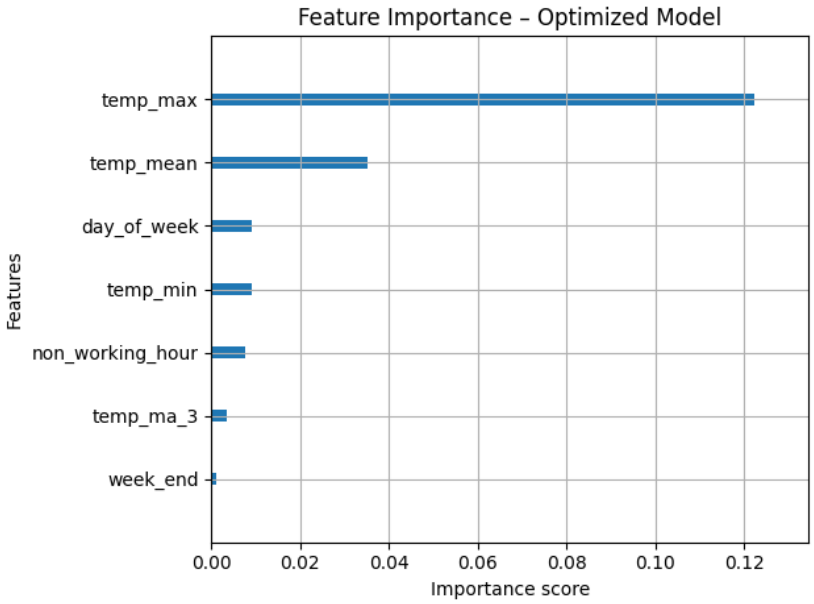


Figure 38 Feature Importance of Daily Heat Demand Model

The following figure presents the residuals of the model. As shown, the residuals are small (within ±0.7 kW) and randomly distributed, which indicates a generally well-calibrated model. However, the residuals are often slightly negative, meaning the model tends to overpredict heat demand (i.e., predicted values are frequently higher than the actual values, as observed in the plot). This suggests a minor overprediction bias, particularly on high-demand days. The model could be further improved by enhancing its ability to capture these extreme cases and reduce this small, systematic error.

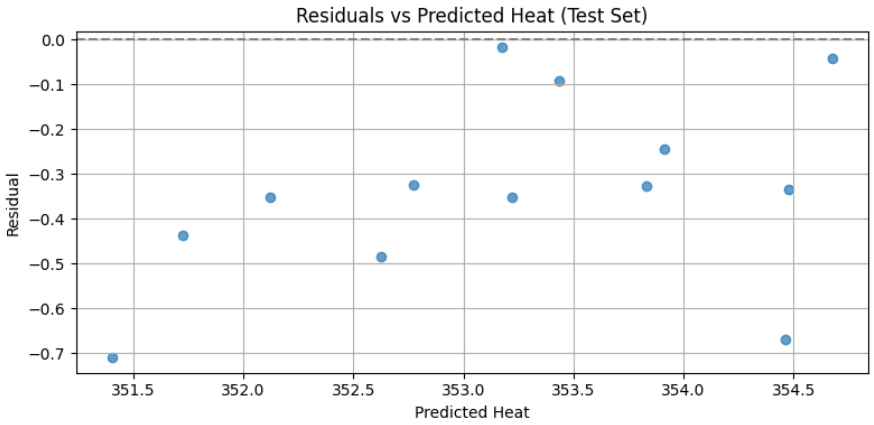


Figure 39 Residuals of Daily Heat Demand Model

As shown in the figure 40, we can observe how the choice of hyperparameters influences the RMSE and how these parameters were tuned. For clarity and focus, we present only two hyperparameters—learning rate and maximum tree depth—through a heat map and a box plot.

The best RMSE (the lowest error) was achieved with the following configuration: learning_rate = 0.04, max_depth = 5, n_estimators = 300, subsample = 0.8, colsample_bytree = 1.0

These settings produced the most accurate results. The figure shows that the model's performance steadily improves as both the learning rate and tree depth increase—up to a point. The combination of deeper trees and larger learning steps allowed the model to capture complex demand patterns more effectively.

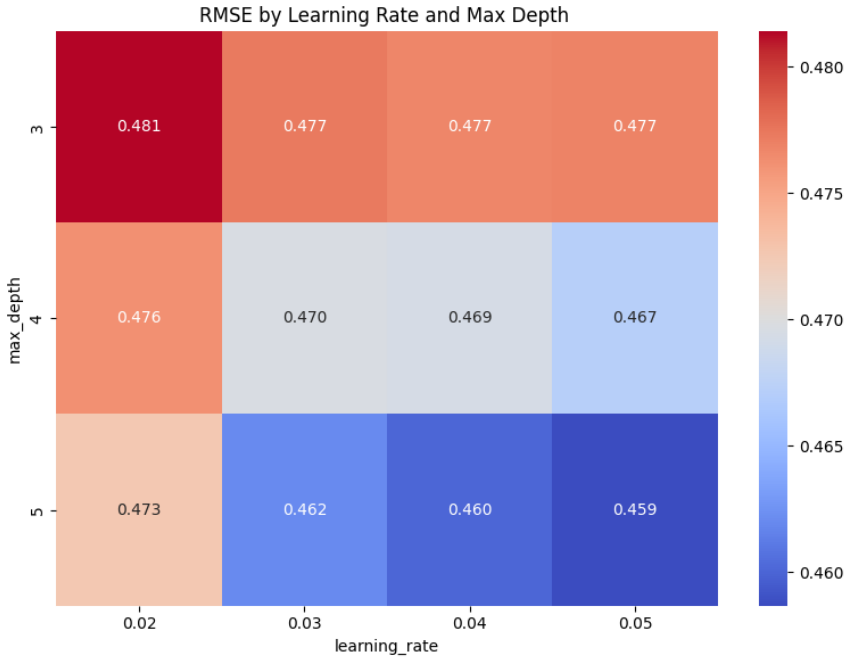


Figure 40 Heat map: RMSE by Learning Rate & Max Depth of Daily Heat Demand Model

The box plot illustrating the influence of these hyperparameters is shown below. Each box represents the distribution of RMSE scores for a specific combination of learning rate and maximum tree depth. Higher learning rates (0.05) result in lower RMSE values. The box plot for maximum depth shows that deeper trees (particularly those with depth = 5) consistently reduce RMSE, indicating a better model fit. The dots above the boxes represent outlier RMSE values, related to individual model runs (specific sets of hyperparameters) where the RMSE was unusually high compared to other runs at the same tree depth.

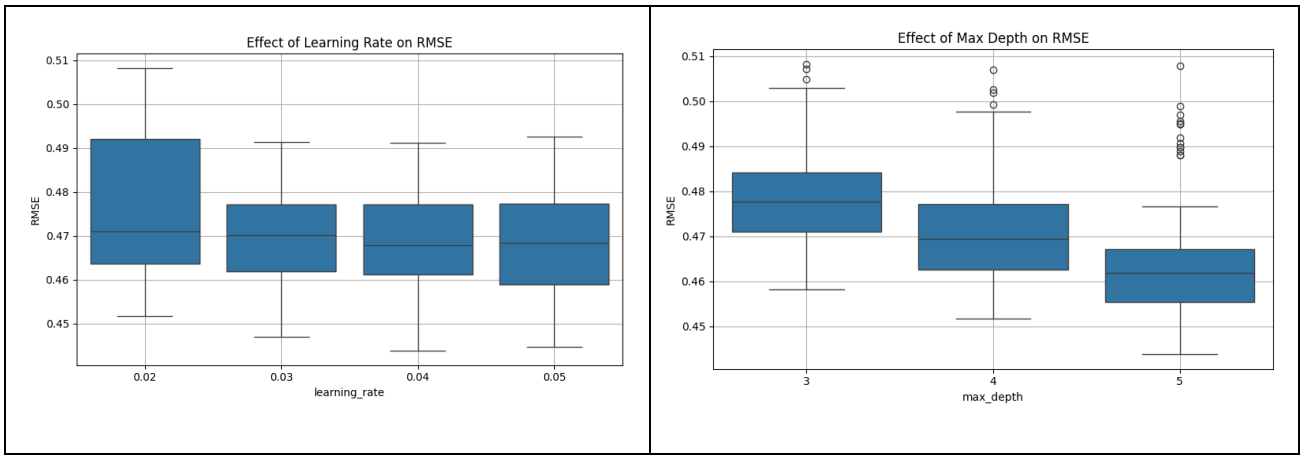


Figure 41 Boxplots: Effect of Hyperparameters of Daily Heat Demand Model

5 DISCUSSION

5.1 Electrical Consumption

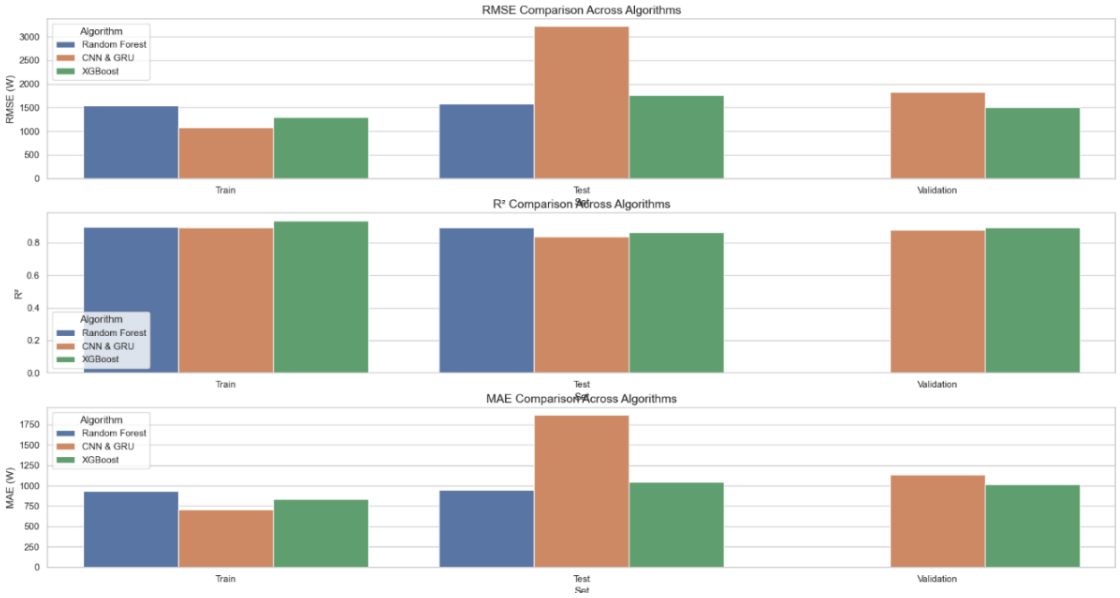


Figure 42 Algorithm Comparison for electrical consumption

The bar plot (Figure 52) compares Random Forest Regressor, CNN & GRU, and XGBoost for forecasting residential electrical consumption using RMSE, R^2, and MAE across train, validation (where applicable), and test sets. Random Forest showed stable performance (RMSE: 1547.18 W train, 1587.98 W test; R^2: 0.897 train, 0.894 test; MAE: 931.18 W train, 950.84 W test), indicating robust generalization. CNN & GRU excelled on training (RMSE: 1081.44 W; MAE: 708.15 W) but underperformed on test (RMSE: 3213.89 W; R^2: 0.8385; MAE: 1865.48 W) and validation (RMSE: 1827.74 W; MAE: 1134.70 W), suggesting overfitting. XGBoost achieved strong training results (RMSE: 1295.38 W; R^2: 0.935; MAE: 838.06 W) but moderate test performance (RMSE: 1768.77 W; R^2: 0.863; MAE: 1042.24 W), with validation metrics (RMSE: 1510.69 W; R^2: 0.892; MAE: 1014.27 W) indicating good generalization. Random Forest offers the best balance of accuracy and stability, while CNN & GRU requires optimization to enhance test performance.

5.2 Heat Demand

Table 20 Outline of Heat Demand Results

Models	First Model (Hourly)	Second Model (Hourly)	Third Model (Hourly)	Fourth Model (Hourly)	Daily Model
RMSE (kw)	10.81	6.89	17.55	7.68	0.39
R^2	0.85	0.94	0.44	0.94	0.87

The predictive models of heat demand created in this research offer an extensive investigation of various feature engineering methods and modelling strategies to make predictions of both hourly and daily household heating demands. The performance of each model evaluated using RMSE and R^2 metrics.

The initial model included date features so the model could evaluate whether simple time and temperature structure could accurately model behavior. Although this worked well, it exposed one of the model's fundamental limitations: that the dates were interpreted basically as sequential markers with no appreciation of season. This will not matter with small datasets, but with bigger, multi-season datasets, dates should ideally be represented in a form that conveys their seasonality, otherwise the model be misled.

To enhance accuracy and realism, the month feature was added to the second model, and the date feature was removed. However, the model became overly reliant on the season's changing, making it less useful for predictions outside the intended timeframe. The third model responded by applying a neural network based on temperature and hour. However, this model found it difficult to achieve the good accuracy. The decline in R2 score and rise in RMSE highlight the difficulty of generalizing complex patterns from a limited dataset specially using algorithms like neural networks. However, this model acknowledges that although performance may not be sufficient, this method is more appropriate for changing usage and climate conditions. The fourth model showed how accuracy and generalizability could be balanced by using lag and rolling features to incorporate short-term memory. It achieved high accuracy without overfitting to the calendar. However, some drawbacks were identified, including the model's inability to adjust to reflect changes that are not represented in recent history.

The daily model took a different approach by predicting the change in daily heat demand rather than absolute values. Despite its high accuracy and low error, it was still dependent on the demand from the previous day. Because of this, the model is very good at identifying short-term trends, but it is also a little reliant on demand behavior continuity. In summary, the models show the trade-offs between accuracy, generalizability, and different features. Each model brings its strengths depending on forecasting goals, whether high short-term precision or broader applicability over different conditions is more critical.

6 CONCLUSIONS AND RECOMMENDATIONS

This chapter outlines the best performing model for each of the goals, electrical consumption forecasting and heat demand forecasting.

6.1 Electrical consumption

- While the Random Forest Regressor showed best results in terms of metrics comparison (RMSE: 1547.18 W train, 1587.98 W test; R^2: 0.897 train, 0.894 test; MAE: 931.18 W train, 950.84 W test), due to it's random data set split and the lack of lag and roll features, it is not very suitable for forecasting, it is rather a learning algorithm.
- On the other hand, XGBoost has lag and roll features and provides the opportunity for more accurate forecasting predictions, while still keeping performance metrics close to the RFR results, for training set: (RMSE: 1295.38 W; R^2: 0.935; MAE: 838.06 W) and test set metrics values (RMSE: 1768.77 W; R^2: 0.863; MAE: 1042.24 W).
- The hybrid neural network model has promising results on the training set (RMSE: 1081.44 W; MAE: 708.15 W), however due the chronological data split, the model overfits (RMSE: 3213.89 W; R^2: 0.8385; MAE: 1865.48 W) and does not generalize of unseen data well enough. However, with a proper optimization, this model could be quite efficient in a longer run with bigger datasets.

6.2 Heat Demand

- The model used moth as a feature and the model using rolling features demonstrated superior performance ($R^2 = 0.94$).
- While incorporating month features improved accuracy, it created over-reliance on seasonal patterns, highlighting the challenge of balancing precision with generalizability across different timeframes and climate conditions.
- The daily model is able to predict the daily heat demand by ($R^2 = 0.87$).
- The neural network approach (third model) struggled with limited data, evidenced by its significantly lower performance metrics ($R^2 = 0.44$), demonstrating that complex algorithms require substantial training data to be effective.

- Incorporating lag and rolling features (fourth model) successfully balanced accuracy with generalizability by integrating short-term memory components, though this approach remains limited in adapting to unprecedented changes.

- The research reveals fundamental trade-offs between accuracy, generalizability, and feature dependency across all models, emphasizing that model selection should align with specific forecasting objectives—whether prioritizing immediate precision or long-term adaptability.

Recommendations:

- Due to the complexity of the models, all of them are relatively slow on training. Therefore, a machine with proper resources and good GPU would be verry efficient for optimization of the models.
- CNN & GRU shows a promising foundation for a proper forecasting model, but further optimization is needed to address the overfitting nature of the algorithm.
- Lag and Roll features should be implemented in the Random Forest algorithm, to enhance forecasting abilities of the model.
- More data, would beneficial for the model training as they would be able to capture more patterns, therefore making better predictions.
- More weather-related data in the models features could bring some information about the non-linear relationship between weather data and electricity consumption, therefore improving the model's forecasting abilities.

7 REFERENCES

Ahmed, R., Hussain, F., Kim, J., & Han, K. (2021). *Short-term load forecasting for smart home appliances with sequence to sequence learning.*

Alawi, O. K. (2024). Optimizing building energy performance predictions: A comparative study of artificial intelligence models. *Building Engineering*.

Alisha Banga, S. C. (2024). Short-Term Electricity Consumption Forecasting at Residential Level Using Two-Phase Hybrid Machine Learning Model. *Electrica*.

C. Monzani, G. C. (2024). Modelling Hourly Thermal Energy Demand: A Machine Learning Approach of Residential District Heating Substations in Turin. *International Conference on Renewable Energies and Power Quality.* Bilbao.

Etienne Saloux, J. A. (2018). Forcasting District Heating Demand using Machine Learning Algorithms. *DHC.* Humburg.

Farahani, R., Akbari, M., Asgari, N., & Lee, C. (2024). *Advancements in Digital Twin Technology and Machine Learning for Next-Generation Energy Systems.* Energy Reports.

KNMI. (2025). Retrieved from The Royal Netherlands Meteorological Institute: https://www.knmi.nl/home

Li, X., Wang, J., Liu, J., & Zhang, Y. (2023). *A machine learning framework to estimate residential electricity demand.*

Łukasz Guz, O. G. (2025). Forecasting Heat Power Demand in Retrofitted Residential Buildings. *MDPI*.

Maciej Bujalski, P. M. (2021). Heat demand forecasting in District Heating Network using.

Majumdar, S. (25, 04). *Lag & Rolling Features: Time Series Forecasting with Python*. Retrieved from Medium: https://sarmita-majumdar.medium.com/lag-rolling-features-time-series-forecasting-with-python-0c3d6ee2b845

Mousavi, S., Pour, S., & Dehghani, M. (2024). *Energy Consumption Prediction in Residential Buildings—An Accurate and Interpretable Machine Learning Approach Combining Fuzzy Systems with Evolutionary Optimization.*

Neele Kemper, M. H. (2023). Forecasting of residential unit's heat demands:a comparison of machine learning techniques in a real-world case study. *Springer Nature*.

Open-Meteo. (2025). *Open-Meteo archive*. Retrieved from Open-Meteo: https://archive-api.open-meteo.com/v1/archive

SAJJAD, M. A. (2020). A Novel CNN-GRU-Based Hybrid Approach for Short-Term Residential Load Forecasting. *IEEE*.

Scikit-Learn. (2025). Retrieved from scikit-learn: https://scikit-learn.org/stable/

APPENDIX A NOMENCLATURE

Symbols

α	Learning rate coefficient	[-]
λ	Regularization coefficient	[-]

Acronyms

ANN	Artificial Neural Network
RFR	Random Forest Regressor
CNN	Convolutional Neural Network
GRU	Gate Recurrent Unit
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
MAE	Mean Average Error
XGBoost	Extreme Gradient Boosting

APPENDIX B SOFTWARE DEPENDENCIES

- Python 3.+ up to Python 3.12
- Pandas
- Scikt-Learn
- NumPy
- Joblib
- Matplotlib
- openmeteo-requests
- Seaborn
- Tensorflow
- Xgboost

All of the software libraries mentioned above are open-source.

APPENDIX C DATA MAPPINGS AND DATASET SNAPSHOTS

Table 21Parameter mapping

```
parameter_mapping = {
"voltage_phase_1": 1,
"voltage_phase_2": 2,
"voltage_phase_3": 3,
"current_phase_1": 4,
"current_phase_2": 5,
"current_phase_3": 6,
"power_phase_1": 7,
"power_phase_2": 8,
"power_phase_3": 9,
"total_system_power": 10,
"total_import_kwh": 11,
"total_export_kwh": 12,
"total_kwh": 13,
"l1_import_kwh": 14,
"l2_import_kwh": 15,
"l3_import_kwh": 16,
"l1_export_kwh": 17,
"l2_export_kwh": 18,
"l3_export_kwh": 19,
"l1_total_kwh": 20,
"l2_total_kwh": 21,
"l3_total_kwh": 22
}
```

Table 22 Device mapping

```
device_id,"device_manufacturer","device_name","device_baud_rate","device_databits","device_protocol",
"device_ipadress","description"
2,Solenco,Solenco,,,,"10.1.0.21",
3,Heat Pump,HeatPump,,,,"10.1.0.22",
4,WKO,WKO,,,,"10.1.0.23",
5,LightingCombine,LightingCombine,,,,"10.1.0.25",
1,woningen,woningen,,,,"10.1.0.20",
```

	A	B	C	D
1	reading	reading_date	device_fk	parameter
2	-2184.894775	10:02.1	1	7
3	0	10:02.1	5	12
4	2912.875	10:02.1	5	11
5	0	10:02.1	5	9
6	0	10:02.1	5	8
7	498.728394	10:02.1	5	7
8	0	10:02.1	4	12
9	5596.835938	10:02.1	4	11
10	0	10:02.1	4	9
11	0	10:02.1	4	8
12	1215.407104	10:02.1	4	7

Figure 43 Two weeks electrical data

	A	B	C	D	E	F	G	H
1	reading	reading_date	device_fk	parameter	device_de	parameter	power_phase_description	
2	1054.853882	12/19/2024 16:01	1	7	woningen	power_ph	power_phase_1	
3	224.91304	01:05.1	1	9	woningen	power_ph	power_phase_3	
4	-1754.675049	01:05.1	1	8	woningen	power_ph	power_phase_2	
5	1029.713135	02:04.1	1	7	woningen	power_ph	power_phase_1	
6	-1271.288574	02:04.1	1	8	woningen	power_ph	power_phase_2	
7	225.73732	02:04.1	1	9	woningen	power_ph	power_phase_3	
8	-2500.891602	38:02.9	1	8	woningen	power_ph	power_phase_2	
9	235.746552	38:02.9	1	9	woningen	power_ph	power_phase_3	
10	771.593018	38:02.9	1	7	woningen	power_ph	power_phase_1	
11	235.982056	39:02.2	1	9	woningen	power_ph	power_phase_3	
12	784.310608	39:02.2	1	7	woningen	power_ph	power_phase_1	

Figure 44 device_data_woningen.csv 4 months data headers

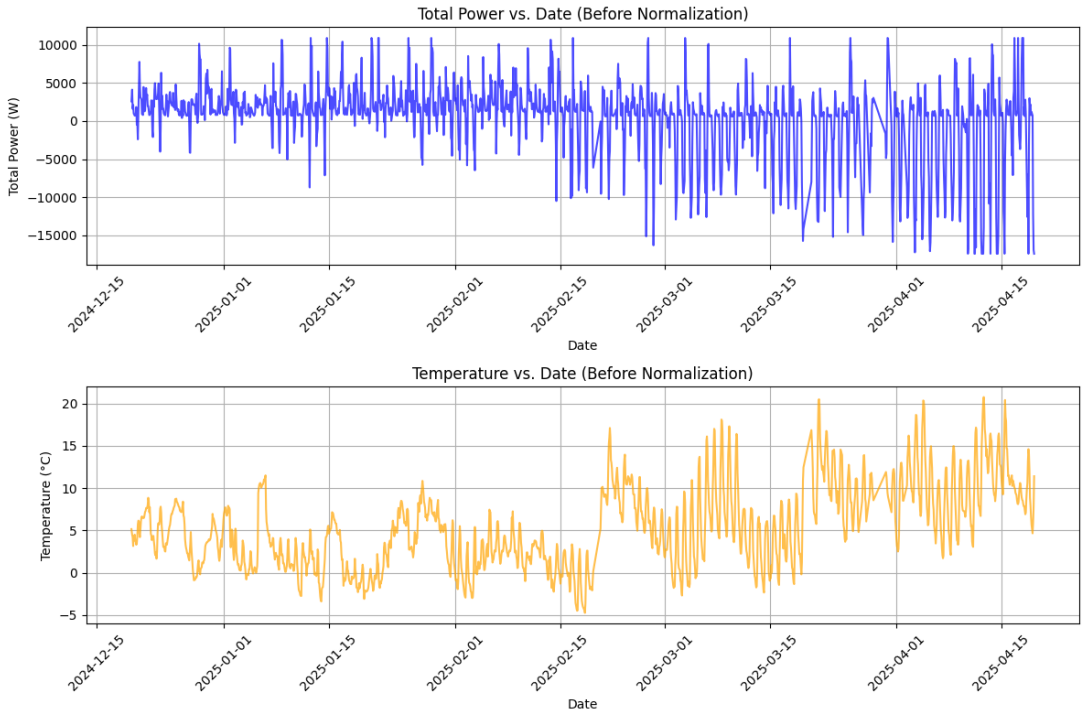


Figure 45 Power and Temperature before normalization (4 months) RFR

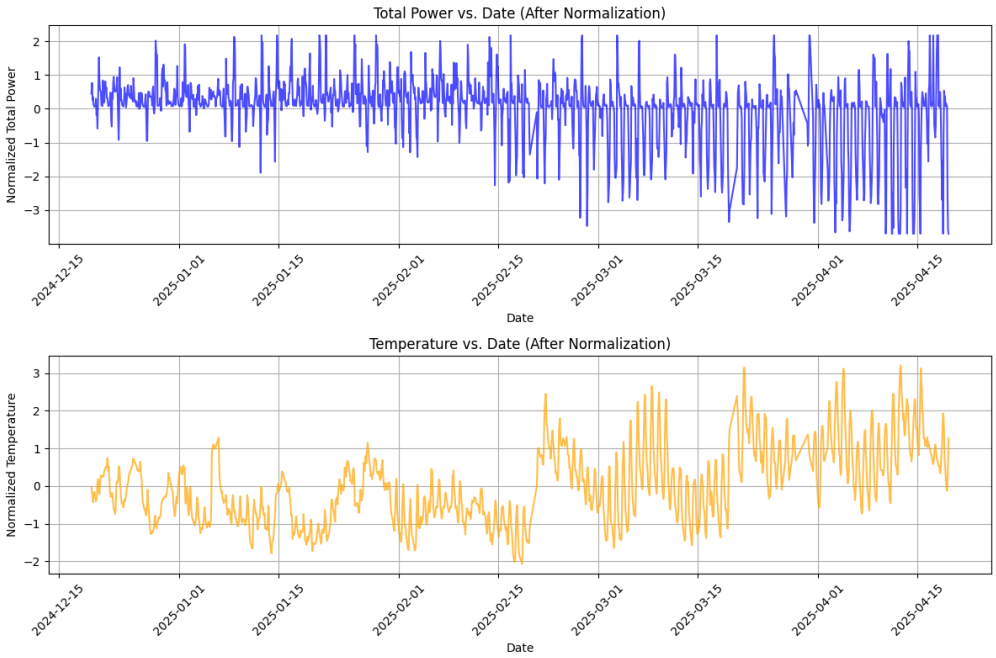


Figure 46 Power and Temperature after normalization (4 months) RFR

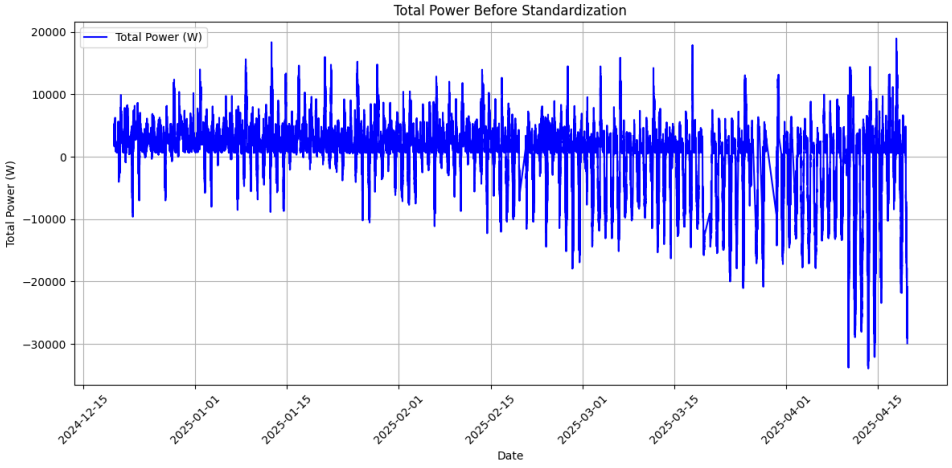


Figure 47 Total Power before normalization CNN&GRU

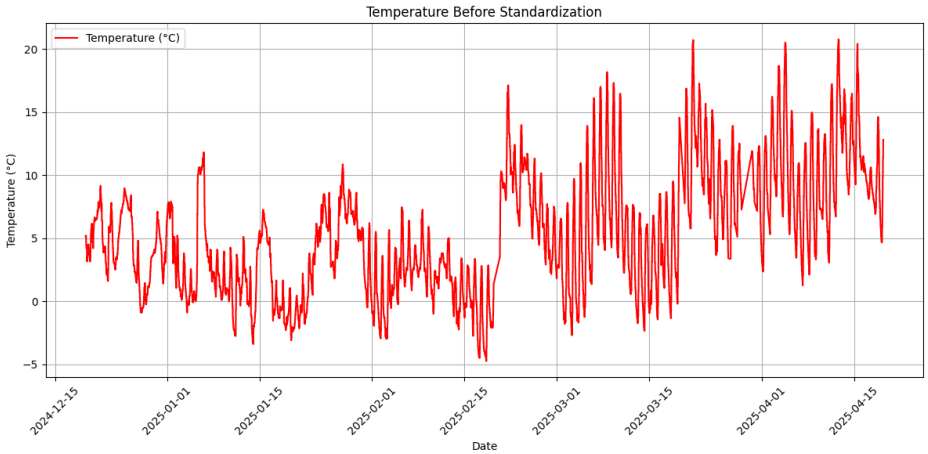


Figure 48 Temperature before normalization CNN&GRU

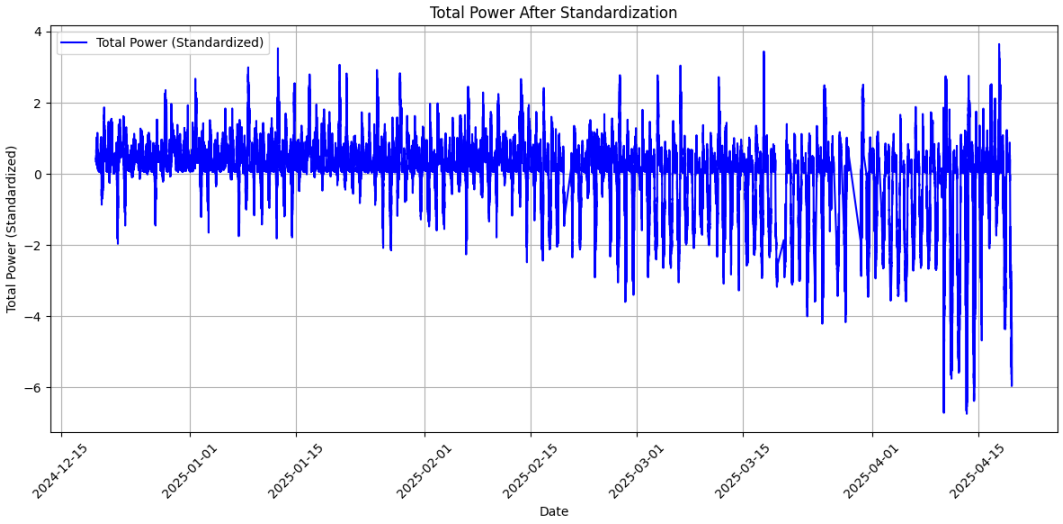


Figure 49 Total Power after normalization

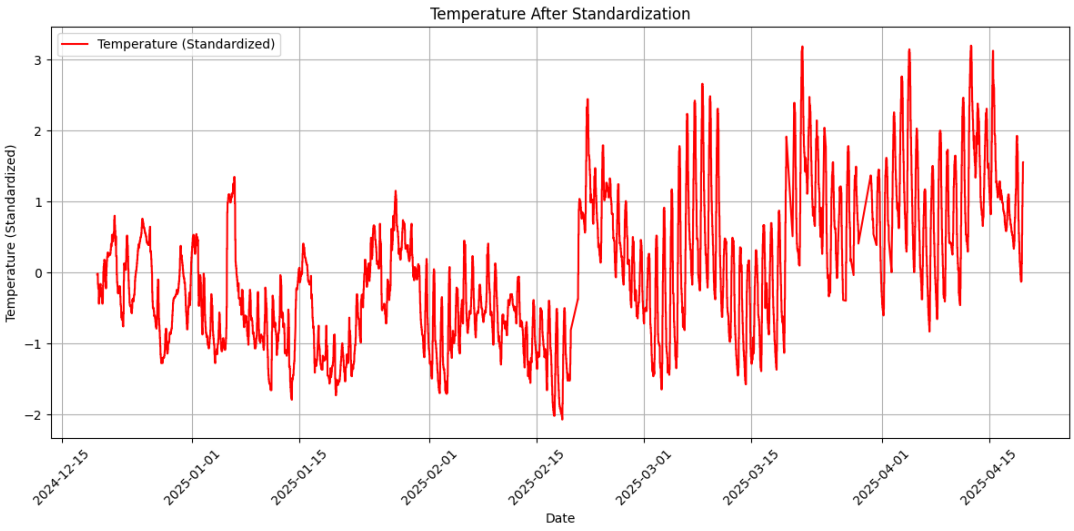


Figure 50 Temperature after normalization

Table 23 Features of Heat Demand Models

Models	Features
Model 1 (Hourly)	hour, hour_squared, day_of_week, week_end, non_working_hour, temperature, temp_squared, day
Model 2 (Hourly)	hour, day_of_week, week_end, non_working_hour, temperature, is_morning, is_evening, hour_sin, hour_cos, month
Model 3 (Hourly)	Hour, temperature, temperature lag, temperature^2, temperature^3, moving average of the temperature, non linear temperature term
Model 4 (Hourly)	hour,day_of_week, week_end, non_working_hour, temperature, is_morning, is_evening, hour_sin, hour_cos,heat_lag_1, heat_roll_2, temp_x_hour,
Daily Model	temp_mean', temp_min, temp_max, day_of_week, week_end, non_working_hour, temp_ma_3

APPENDIX D TWO WEEKS RANDOM FOREST REGRESSOR RESULTS

After checking for any missing values in the dataset, outliers are handled as the first 1st and 99th percentile are capped. The data then is standardized.

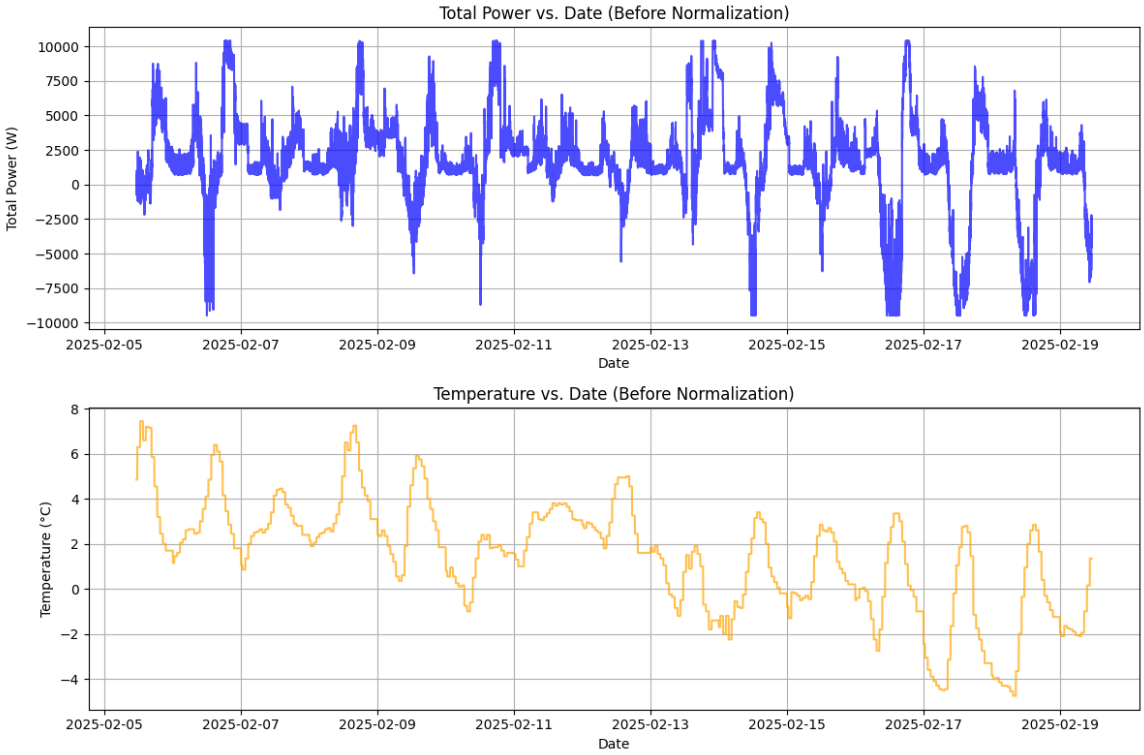


Figure 51 Two week data before normalization

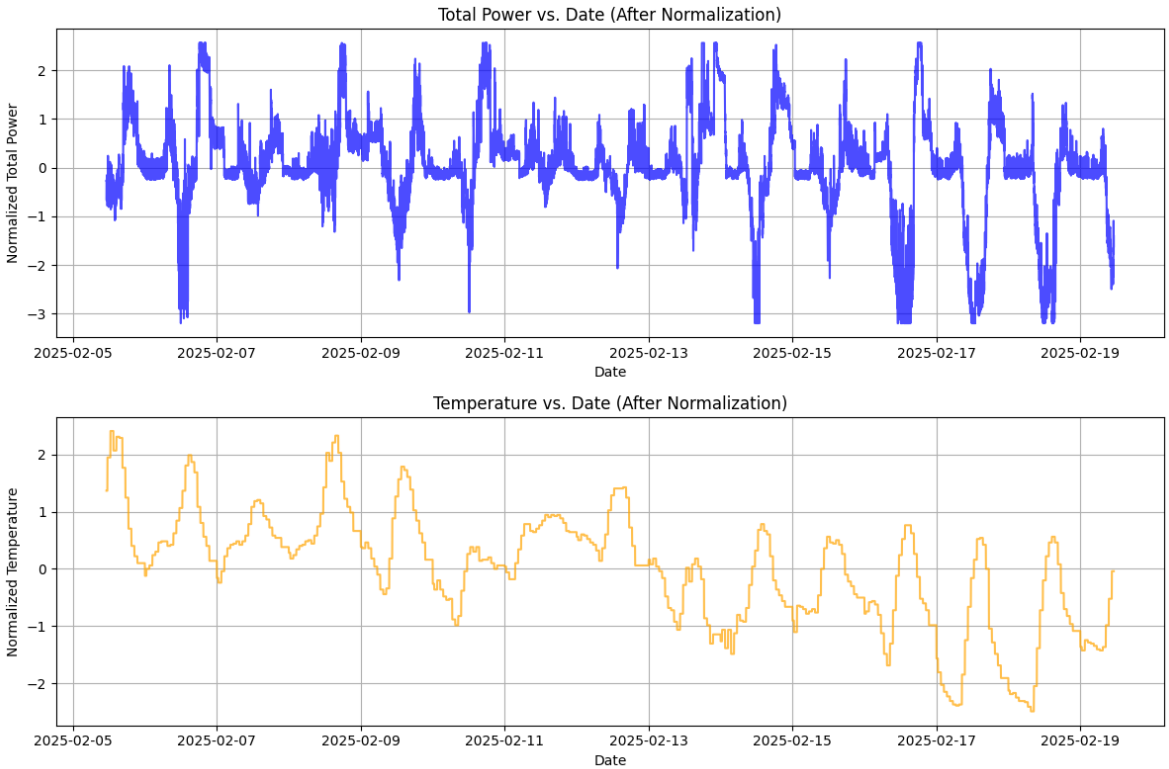


Figure 52 Two weeks data after normalization

The temperature and total sum of the phases power is standardized. This resulted in changing the size of the power from -10000 to 10000, to, -3 to 3. The temperature is scaled from -5 to 8, to, -3 to 3.

Then the model is trained with all default parameters and 100 decision trees.

Table 24 RFR Metrics Result (2 weeks data)

Training Set	Test Set
$R^2 = 0.919$	$R^2 = 0.910$
RMSE = 992.77 W	RMSE = 1015.29 W
MAE = 662.01 W	MAE = 679.68 W

The RFR achieved a high R^2 score of 0.919 for training set and 0.91 for test set. Root Mean Square error of 992.77 W for the training set and 1015.29W for the test data. The mean average error between the two sets is 662.01W and 679.68 W for train and test, respectively. The close metrics results between the two datasets, suggests that there is no underfitting or overfitting and the model is able to generalize well unseen data.

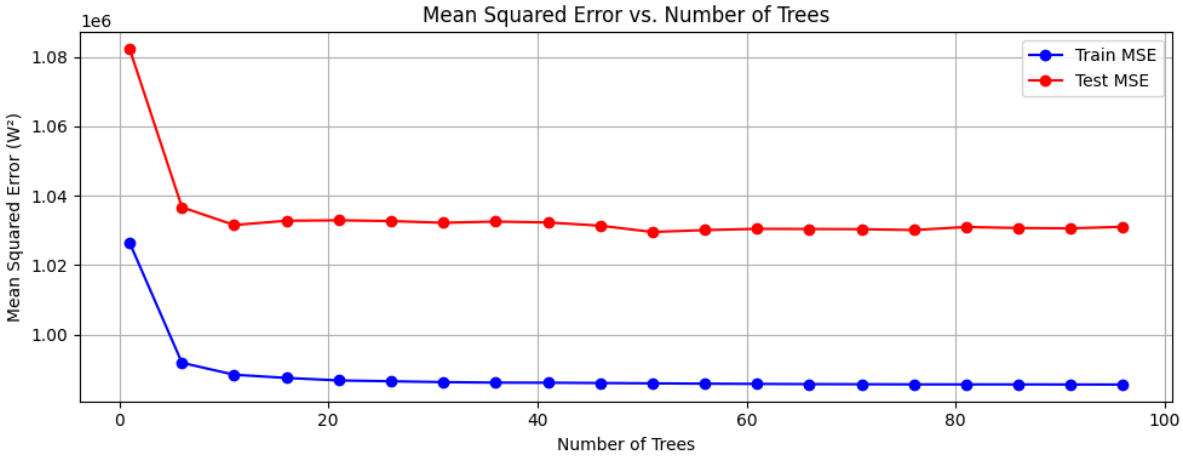


Figure 53 Number of trees vs MSE (Two Weeks)

The graph above shows how the number of trees in the Random Forest affects the Mean Squared Error. Usually, the higher the number of trees, the lower the error, however, it is important to keep the balance, because the higher number of trees decreases computational efficiency.

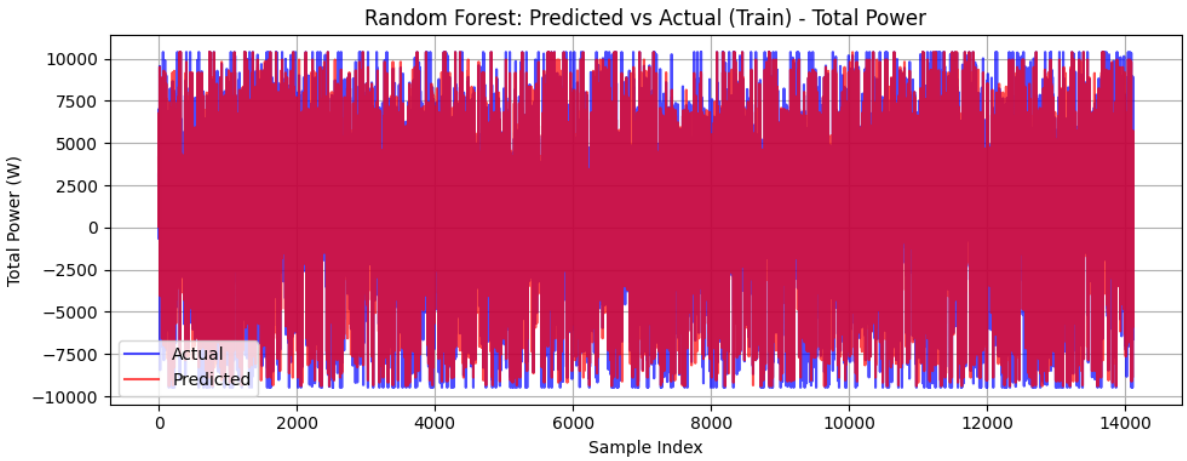


Figure 54 Train set predicted vs actual (Two Weeks)

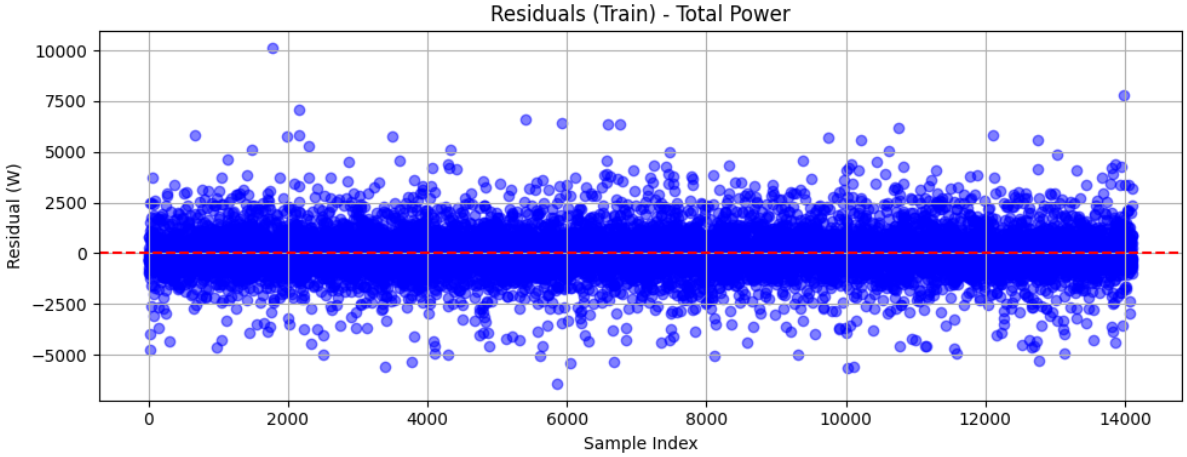


Figure 55 Residuals Train set (Two weeks)

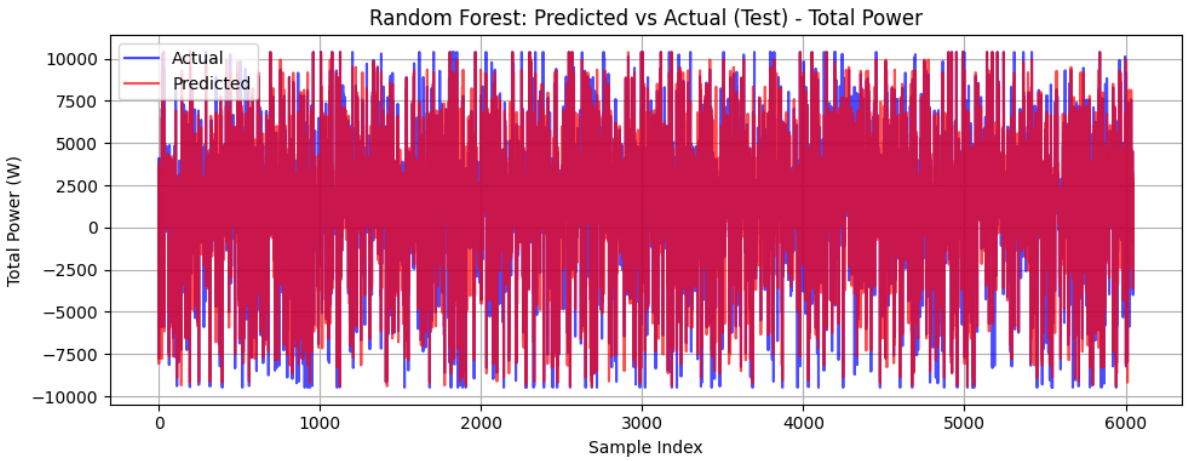


Figure 56 Test set predicted vs actual (Two Weeks)

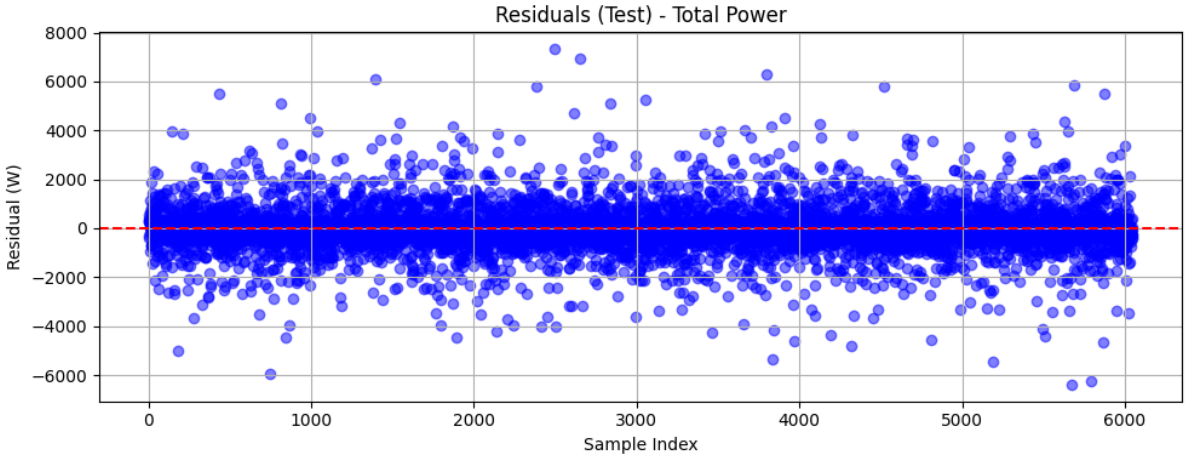


Figure 57 Test set residuals (Two Weeks)

The initial implementation of the RFR method shows very good results. High R^2 suggests that the model is capable of explaining the variance in the data. The RMSE is around 1000W, which given the fact that the data is sampled minutely, is calculated as

$$Energy\ error\ (kWh) = \frac{Power\ Error(W)}{60} * \frac{1}{1000} = \sim 0017\ (kWh)$$

The initial implementation of the Random Forest Regressor showed promising, results. However, the dataset of two weeks is too short to make proper predictions and conclusions, therefore without further optimization, the RFR model is taken to the larger dataset.