CHALMERS
UNIVERSITY OF TECHNOLOGY

Learning to Optimize Voltage Level for Energy-Efficient Radios

Citation for the original published paper (version of record):
Nyberg, C., Stiebe, E., Lejon, T. et al (2026). Learning to Optimize Voltage Level for Energy-Efficient Radios. IEEE Transactions on Machine Learning in Communications and Networking, 4: 1199-1226. http://dx.doi.org/10.1109/TMLCN.2026.3711568

(article starts on next page)

IEEE Transactions on Machine Learning in Communications and Networking

Received 16 August 2025; revised 21 March 2026 and 26 May 2026; accepted 2 July 2026. Date of publication 9 July 2026; date of current version 15 July 2026.

The associate editor coordinating the review of this article and approving it for publication was Z. Chang.

Digital Object Identifier 10.1109/TMLCN.2026.3711568

Learning to Optimize Voltage Level for Energy-Efficient Radios

CECILIA NYBERG[1,2], ELIN STIEBE[1,2], THOMAS LEJON[3], LINUS ARONSSON[1,2], AND MORTEZA HAGHIR CHEHREGHANI[1,2]

[1]Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg 412 96, Sweden
[2]Department of Computer Science and Engineering, University of Gothenburg, Gothenburg 405 30, Sweden
[3]Department of Radio Systems and Technology, Ericsson, Stockholm 164 80, Sweden

CORRESPONDING AUTHOR: C. NYBERG (cecilia.nyberg@hotmail.se)

The work of Cecilia Nyberg and Elin Stiebe was supported by Ericsson AB. The work of Linus Aronsson and Morteza Haghir Chehreghani was supported in part by the Wallenberg AI, Autonomous Systems, and Software Program (WASP), funded by Knut and Alice Wallenberg Foundations.

ABSTRACT Reducing energy waste in cellular radio systems is critical for lowering the carbon footprint of wireless communication networks. In current deployments, radio power amplifiers (PAs) typically operate at a fixed high supply voltage regardless of physical resource block utilization (PRB-U), leading to unnecessary energy consumption. This work investigates a machine learning-based approach on real field data for dynamically switching the PA supply voltage between two hardware-supported levels, thereby adapting the available radio-frequency (RF) output power capacity to predicted traffic demand. While many prior approaches focus on network-level control, the proposed method performs voltage optimization at the level of individual radio units. Eight models are evaluated under two task formulations: 1) classification of voltage levels and 2) regression of PRB-U followed by mapping to voltage states. The results show that machine learning can achieve substantial energy savings, although effectiveness varies across radios. Underestimations, defined as instances where the predicted voltage is lower than required, are minimized more effectively by classification models than by regression approaches, resulting in a more reliable trade-off between energy efficiency and connectivity. A customized Feedforward Neural Network classifier and a Random Forest model perform best, outperforming both statistical baselines and more complex temporal architectures, achieving F1 scores of 0.35 and 0.29 respectively while reducing energy consumption by over 12%. In contrast, a baseline ARIMA model shows poor predictive performance with an F1 score of 0.09 despite similar energy savings. Explainability analysis using SHAP indicates that current PRB-U is the most influential feature, while the superior performance of nonlinear models suggests that linear utilization-based approaches are insufficient for reliable switching decisions. These results demonstrate the potential of learning-based PA voltage control as a new mechanism for improving the energy efficiency of cellular radio systems while maintaining service reliability.

INDEX TERMS Dynamic voltage scaling, energy efficiency, machine learning, explainable artificial intelligence, load forecasting, time series analysis.

I. INTRODUCTION

THE Paris Agreement is a significant milestone in climate policy making, adopted on December 12, 2015 [1]. This legally binding treaty, established under the United Nations, outlines in Article *2a* the goal of "holding the increase in the global average temperature to well below 2°C above pre-industrial levels and pursuing efforts to limit the increase to 1.5°C". January 2025 was the warmest on record, with temperature levels 1.75°C above pre-industrial levels [2].

Significant reductions in global energy consumption and emissions are required across all sectors [3] to meet the goals outlined in the Paris Agreement. One strategy to support this transition is the incorporation of machine learning

(ML)-based methods to enable more energy-efficient solutions. As many energy systems rely on sequential, sensor-collected data, time series forecasting can play a central role in enabling predictive control and dynamic adaptation.

Time series forecasting has advanced rapidly in recent years, supported by the increasing availability of data [4]. It is widely applied in domains such as healthcare, finance, and industrial production [4], and has also shown promise in energy efficiency applications [5]. A wide range of methods exist, from traditional statistical models to machine learning techniques [6].

In line with these developments, ML models have gained traction over traditional statistical approaches due to their ability to capture complex patterns and long-term dependencies in data [4]. Various neural architectures, such as Deep Neural Networks (DNNs), Graph Neural Networks, Convolutional Neural Networks (CNNs), and Transformers, have shown strong potential in this domain [7], [8]. Despite their success, deep learning models such as Transformers can face certain limitations in time series forecasting tasks, such as inefficiencies and diminishing returns when increasing the look-back window length. Furthermore, CNNs have shown strong results in time series tasks. For example, in one study of stock market forecasting, CNNs outperformed the Long Short-Term Memory model (LSTM) networks by 30% in accuracy [9].

Moreover, [10] conducted a comprehensive benchmarking study across a wide range of real-world tabular datasets and found that models like XGBoost and Random Forests consistently outperformed deep learning models on medium-sized datasets. Their results emphasize the strength of these classical approaches in scenarios where data is limited, noise is present, and the data is non-rotationally invariant, such that forming linear combinations of features may introduce misleading representations. Recently, [11] demonstrated the superior performance of tree ensemble methods, such as XGBoost and Random Forests, in multi-armed bandit problems as well.

Energy consumption forecasting using time-series data is inherently challenging due to its nonlinear, non-stationary, and seasonally dependent nature [12]. Deep learning methods have demonstrated strong performance improvements in large-scale energy forecasting tasks such as electric load demand and renewable generation prediction [5], [13].

Another area where time-series predictions for energy efficient operations have been applied is that of cellular base stations (BSs). Foundational frameworks such as the EARTH model establish a mapping between radiated radio-frequency (RF) output power and total supply power, demonstrating an approximately linear relationship between transmit power and BS power consumption [14]. This linear load-dependent abstraction has been widely adopted in subsequent work, including [15], [16], [17]. Power scaling with respect to component carriers has likewise been approximated using linear models [16], reinforcing the prevalence of load-based representations in radio energy analysis.

More recently, research has focused on refining BS power consumption modeling using both analytical and data-driven techniques. In [16], machine learning is combined with analytical modeling to characterize load- and configuration-dependent behavior across diverse 5G BS setups. In parallel, [15] investigates probabilistic forecasting of Physical Resource Block (PRB) utilization to improve energy efficiency in Open RAN deployments by reducing unnecessary capacity over-allocation while minimizing the risk of capacity shortfall.

While these studies enhance load modeling and forecasting accuracy, energy savings are typically realized indirectly through improved prediction combined with existing linear PRB-to-power mappings. Explicit hardware-level actuation mechanisms are generally not considered.

In particular, discrete adaptation of the power amplifier (PA) supply voltage under practical hardware cost constraints has not been systematically explored. Adjusting the PA supply voltage directly constrains the maximum available RF output power range and therefore influences DC input power consumption at the source. However, supporting multiple PA supply voltage levels increases hardware complexity, requiring additional DC-DC regulation stages, validation effort, and calibration overhead. It is therefore important to evaluate achievable energy savings under restricted actuation capability.

Furthermore, controlling the voltage from the radio is of increased interest in the context of Open RAN, where architectural disaggregation motivates energy-efficient control strategies at each level of the stack, rather than centralized network-wide optimization.

This paper addresses the above gap by developing a machine learning–based methodology for predicting the PA supply voltage required to satisfy upcoming downlink throughput demand at the radio unit level. The focus is on the PA, which converts DC input power to RF output power delivered at the antenna port. The objective is to enable dynamic switching between two predefined PA supply voltage levels such that RF output capability aligns with predicted PRB utilization while respecting hardware cost constraints. By adapting the supply voltage based on predicted load conditions, unnecessary DC power consumption can be reduced without compromising performance during peak demand.

Importantly, the proposed approach operates at the capability level rather than at the waveform level. In contrast to envelope tracking techniques, which continuously modulate the PA supply voltage at instantaneous signal timescales, the proposed method selects between discrete operating regions based on predicted load conditions while leaving the RF waveform unchanged.

A limitation of the proposed approach is that a discrete two-level voltage configuration cannot achieve the theoretical minimum energy consumption obtainable through continuous supply adaptation. The objective of this study is therefore not to approximate continuous tracking, but to evaluate whether substantial energy savings can still be achieved under a practical two-level hardware constraint.

The study proceeds in four main stages. First, it explores whether ML-based methods can provide benefits in the target use case. Second, it compares classification and regression models, including the selection of appropriate evaluation metrics for hyperparameter tuning. Third, it identifies the most effective model architecture based on predictive performance. Lastly, it investigates which input variables are most influential in determining voltage levels, using SHAP [18] for model interpretation.

These stages form the basis for the following research questions, which guide the investigation throughout the paper:

- **RQ1**: Are machine learning techniques useful for dynamic adaptation of voltage in radios?
- **RQ2**: What are the advantages and disadvantages of using classification and regression in the case of energy efficient radios?
- **RQ3**: Which machine learning models are most beneficial in terms of energy saving and optimal connectivity?
- **RQ4**: Which input variables are most influential for the top-performing models in determining voltage levels in energy-efficient radios?

II. PROBLEM FORMULATION

In practical radio deployments, the PA supply voltage is typically configured for worst-case operation, allowing the radio unit to support peak traffic demand at any time. While this supports reliable service delivery, it results in suboptimal energy efficiency during periods of low utilization.

The PA supply voltage directly influences both the maximum achievable RF output power and the PA efficiency η [14]. Therefore, operating at a higher supply voltage than necessary leads to unnecessary DC power consumption when the traffic load is low.

The objective of this work is to dynamically adapt the PA supply voltage such that sufficient RF output power is maintained to meet real-time traffic demand while minimizing the DC input power. The proposed framework is intended to be applicable across radio units with comparable hardware, and this paper presents and evaluates the approach across five radio units. An overview of the framework is shown in Figure 1. Sensor measurements, including PRB utilization, power consumption, and temporal context, are collected at the radio unit and used in a prediction layer. Two alternative machine learning approaches are considered. In the regression-based formulation, a model predicts the PRB utilization for the next time interval, from which the required RF output power $P_{\mathrm{RF,req}}$ is estimated. This requirement constrains the selection of the PA supply voltage. In the classification-based formulation, a model directly predicts the appropriate PA supply voltage level. In both cases, the voltage decision is applied through a DC–DC converter that adjusts the PA supply voltage between two hardware-supported levels. In parallel, the radio signal path follows the standard transmission chain, where PRB allocation drives baseband signal generation and IFFT/OFDM modulation before amplification by the power amplifier. The PA produces the RF output power $P_{\mathrm{RF,out}}$, which is delivered to the antenna. This path is not modified by the proposed approach but defines the RF output power requirement that the predicted load and resulting PA supply voltage must satisfy. The selected voltage level determines the PA efficiency $\eta(V_{PA})$ and the corresponding DC input power, which is computed as $P_{DC} = P_{\mathrm{RF,out}}/\eta(V_{PA})$ for energy evaluation. The inset in Figure 1 illustrates the PA efficiency curves for the two selectable supply voltage levels. The relationship between predicted PRB utilization and required RF output power is motivated in Subsection II-A.

From a hardware perspective, supporting multiple discrete PA supply voltage levels increases implementation cost. Designing DC–DC converters with multiple independent output voltage levels increases circuit complexity and component count [19]. This results in higher design complexity and implementation cost for each supported voltage level.

To balance energy efficiency improvements with hardware feasibility, this study considers a two-level PA supply voltage architecture, consisting of a high-voltage mode for peak load support and a low-voltage mode for reduced traffic conditions. The threshold separating the two voltage levels is described in Subsection II-B.

Under this practical constraint, we investigate the achievable energy savings and assess whether significant reductions in DC input power can be obtained without introducing additional hardware complexity.

The remainder of this section formally defines the modeling problem, including the target definition, the voltage control process, feature description, data characteristics, and the evaluation methodology.

A. TARGET DEFINITION

Prior studies have shown that the PA constitutes the dominant load-dependent component of base station power consumption [14]. The supply power has been shown to follow an affine dependence on the RF output power [20], indicating that dynamic power variations are primarily driven by transmit power.

More recently, radio-level AAU power models [16] decompose the total power consumption into static components and a transmit-dependent term:

$$P_{\mathrm{AAU}} = P_0 + P_{\mathrm{BB}} + P_{\mathrm{Tran}} + P_{\mathrm{PA}} + \frac{1}{\eta}\sum_{c=1}^{C} P_{\mathrm{TX},c}. \quad (1)$$

where η defines the efficiency of the PA.

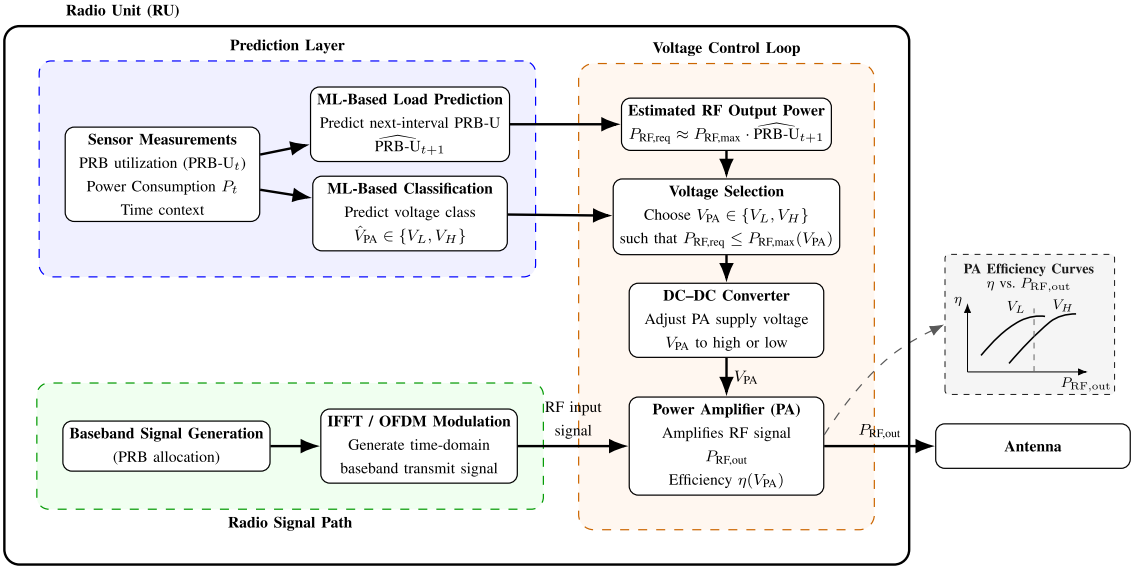


Fig. 1. System overview of the proposed ML-based two-level PA supply voltage control. Sensor measurements are used in a prediction layer to estimate future load or directly classify the PA supply voltage level. The selected voltage is applied through a DC–DC converter to the PA, which produces the RF output power $P_{\text{RF,out}}$ delivered to the antenna. The inset illustrates the PA efficiency curves for the two selectable supply voltage levels.

In this formulation, only the transmit term is traffic dependent. Reference [16] further state that the transmit power increases approximately linearly with the number of allocated PRBs.

A PRB consists of 12 subcarriers in the frequency domain and a fixed number of orthogonal frequency division multiplexing (OFDM) symbols in the time domain [21]. For the purposes of this study, the key insight is that PRB utilization (PRB-U) reflects the proportion of a radio's total data-handling capacity that is currently in use. Formally, PRB-U is,

$$\text{PRB-U} = \frac{N_{\text{PRB, allocated}}}{N_{\text{PRB, max}}} \tag{2}$$

Under the assumption of approximately constant transmit power spectral density per scheduled PRB at the system level, the aggregate RF output power can be modeled as proportional to the fraction of utilized PRBs. This assumption is consistent with OFDM, where the available system bandwidth is divided into multiple orthogonal subcarriers that transmit data symbols simultaneously [22]. In many OFDM system configurations, particularly when static resource allocation is assumed, each subcarrier is assigned approximately equal transmit power. Consequently, the total transmit power scales approximately with the number of active subcarriers and, by extension, with the number of scheduled PRBs. Similar assumptions are commonly adopted in OFDM-based studies where equal transmit power per subcarrier is assumed [23], [24], [25]. The aggregate RF output power can therefore be modeled as proportional to the fraction of utilized PRBs:

$$P_{\text{RF,out}} = P_{\text{RF,max}} \cdot \text{PRB-U}, \tag{3}$$

This formulation treats PRB utilization as a proxy for transmit load, which is consistent with prior Open RAN forecasting work [15], [17], where PRB utilization is used as a surrogate for transmit load.

Hence, this study predicts PRB utilization as a time-series forecasting problem and estimates the corresponding RF output power using the known maximum transmit power of the considered radio.

B. VOLTAGE CONTROL FRAMEWORK

The voltage control investigated in this work targets the PA, which converts DC input power $P_{\text{DC,in}}$ into RF output power $P_{\text{RF,out}}$. In this study, $P_{\text{RF,out}}$ denotes the conducted RF output power at the antenna port, i.e., the power delivered from the PA to the antenna interface prior to radiation.

The mapping between conducted RF output power and the total DC supply power has been formalized in the EARTH energy efficiency framework [14], where the RF output power P_{out} is linked to the total supply power P_{in} through a load-dependent model.

In practice, direct control of RF output power is not possible, but it can be influenced indirectly through adjustments of the PA supply voltage using a DC–DC converter. Reducing the supply voltage lowers the maximum achievable RF output power while simultaneously reducing the DC input power by increasing the PA efficiency η in (1). Consequently, the voltage level must be selected such that the required $P_{\text{RF,out}}$ can be delivered while minimizing $P_{\text{DC,in}}$. Latency and hardware response constraints introduced by the DC–DC converter are assumed to be managed within the system design by delaying the RF signal to match the

Algorithm 1 PRB-U Voltage Level Adjustment

1: **Predict PRB-U:**
  If binary classification:
    Predict class $\hat{y} \in \{0, 1\}$
  Else if regression:
    Predict continuous utilization $\hat{y}_c$
    Map continuous prediction to binary class:

$$\hat{y} \leftarrow \begin{cases} 0, & \text{if } \hat{y}_c < 0.25 \\ 1, & \text{otherwise} \end{cases}$$

2: **Adjust the voltage level based prediction:**
  If $\hat{y} = 0$, decrease voltage by 50%
  Else no voltage change
3: **Estimate energy savings from $P_{\text{DC in}}$ based on voltage level**
  If $\hat{y} = 0$, estimate 15% reduction in $P_{\text{DC in}}$
  Else no change

converter delay. Furthermore, the transition time between voltage levels is designed to be only a small fraction of the symbol duration. Consequently, the delay is assumed to be minimized and negligible.

To provide intuition for this relationship and motivate the voltage threshold used in this study, we consider a simplified model in which the RF output power scales with the square of the supply voltage, assuming constant resistance. This approximation follows from the basic electrical relation $P = U^2/R$ [26], resulting in

$$P_{\text{RF,out}} \propto \frac{U^2}{R} \propto U^2. \tag{4}$$

Supporting multiple PA supply voltage levels increases hardware complexity due to the need for additional DC–DC converter outputs and switching circuitry. To limit hardware cost and complexity, this work therefore considers a two-level PA supply voltage architecture consisting of a *high* and a *low* voltage level.

The high level corresponds to standard operation, while the low level represents a 50% voltage reduction. Consequent with (4), reducing the voltage by half leads to approximately a quarter of the RF output power. Under the assumption of constant spectral power density, this corresponds to a decision threshold of 0.25 in PRB utilization for initiating voltage reduction.

Figure 2 shows histograms of the PRB-U values for each radio. The distributions indicate that the majority of PRB-U values for the five selected radios fall below the threshold, suggesting substantial potential for power savings.

The dynamic voltage adjustment process is structured into the following steps for each predictive model:

C. FEATURE DESCRIPTION

All data in this study is collected through field-deployed sensors in Ericsson's products. Figure 2 presents the PRB-U for five randomly selected radios of the same type but with different serial numbers (radio 1 to 5). The PRB-U values in Figure 2 exclude nighttime hours (00:00 to 05:00), during which the radios are assumed to be in sleep mode with zero utilization. The five radios exhibit overall low utilization, indicating a potential for energy savings.

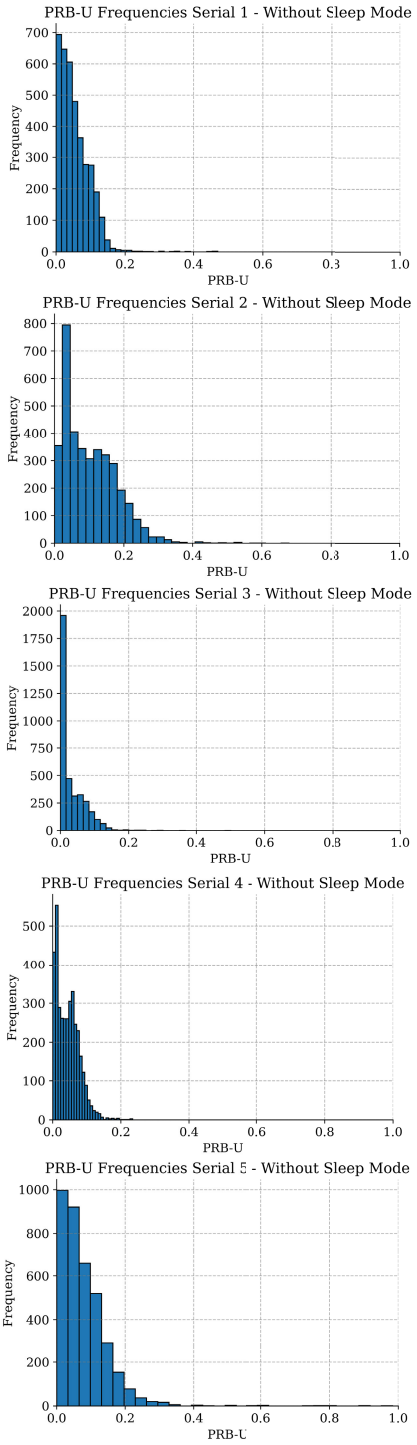


Fig. 2. Histograms of PRB-U values for each radio unit, divided into 30 segments each.

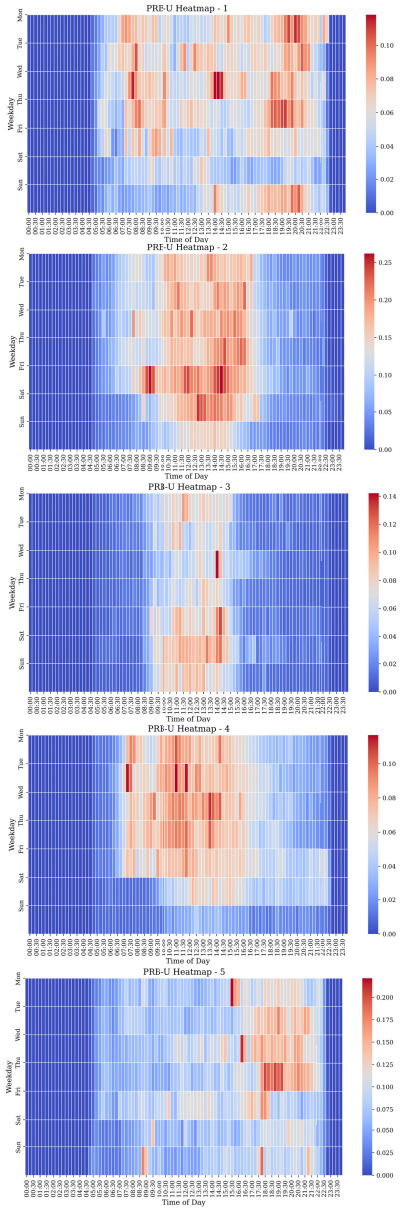


Fig. 3. Heatmaps of PRB-U intensity across weekdays and time of day.

Figure 3 shows how the PRB-U values of the instances in Figure 2 are distributed over time. Low-utilization instances are shown in blue, and high-utilization instances in red. The dataset used for training and evaluation was collected from two internal measurement systems at Ericsson. The training set spans from 2025-01-17 to 2025-03-06 and contains 23,520 raw data points, corresponding to 4,704 samples per radio. An independent test set was constructed using data from 2025-03-08 to 2025-03-22, resulting in 7,200 additional instances.

Each radio reports measurements at 15-minute intervals, corresponding to 96 samples per day. The dataset comprises both time-series and static features, where the time-series measurements capture the temporal evolution of operational variables relevant to power demand prediction. The training set represents approximately 47 days of observations per radio, while the test set covers roughly 14 days.

The list of features is as follows; *serialNumber*: which radio the data belongs to, *timestamp*: the date and time of the measurement, *PowerConsumption*: an array of $P_{\text{DC in}}$ consumed every 6 seconds during the 15-minute interval, *MinPowerConsumption*: the lowest value in the PowerConsumption array, *MaxPowerConsumption*: the highest value in the PowerConsumption array. The feature *PRB-U* is the target variable and is measured as the average utilization during the 15-minute interval, *voltage* is an array of voltages measured every 6 seconds and *current* is an array of current measurements every 6 seconds. The *PowerConsumption* array contains $\tilde{3}$-5 missing entries, which were addressed by forward-filling, assuming the last observed measurement remained valid until a new value was recorded.

Current and *Voltage* are excluded from the analysis. Current to avoid multicollinearity, as it is found to be highly correlated with power. Voltage is excluded since it is theoretically static and, in practice, exhibits very low variance both within and across the analyzed radios. Additionally, *SerialNumber* was excluded to encourage generalization and avoid overfitting to specific devices. The resulting dataset contains no identifying information, which supports the development of device-agnostic models. However, lagged values of the temporal features are still constructed separately for each device, meaning that some device-specific structure remains in the data.

Two main preprocessing actions are taken. First, the mean and median values are derived from the six-second interval array of the PowerConsumption feature. Second, three temporal features are extracted from the timestamp feature. Each is transformed using sine and cosine functions to account for its cyclic nature. The function is shown in (5), where x denotes the original variable, and the divisor corresponds to the maximum value of x, e.g., 24 for hours and 7 for weekdays. This transformation ensures that conceptually similar values, such as 23:00 and 01:00 in hours, receive similar representations, thereby capturing their inherent cyclical relationship.

$$x_{cos} = \cos\left(2\pi \frac{x}{max(x)}\right), \; x_{sin} = \sin\left(2\pi \frac{x}{max(x)}\right) \tag{5}$$

Table 1 lists all features used in the modelling process. The features *maxPowerConsumption*, *minPowerConsumption*, and *medianPowerConsumption*, along with the sinusoidal transformations of *weekday*, *hour*, and *minute*, are treated as non-temporal features (i.e., no lagged values included). In contrast, *meanPowerConsumption* and *PRB Utilization* (PRB-U) are incorporated with lagged values, where the lag length is determined through hyperparameter optimization, later described in Section III-A. The *meanPowerConsumption* is the only power-related feature for which lagged values are included, in order to reduce model

TABLE 1. Features after data preprocessing, all power statistics are derived from $P_{\text{DC in}}$.

Feature	Description
PRB Utilization	Utilization of broadcast capacity every 15 minutes
MaxPowerConsumption	Maximum power consumed during 15-minute interval
MinPowerConsumption	Minimum power consumed during 15-minute interval
meanPowerConsumption	Mean power consumed during 15-minute interval
medianPowerConsumption	Median power consumed during 15-minute interval
minute_sin	Sinusoidal transformation of minute
minute_cos	Cosine transformation of minute
hour_sin	Sinusoidal transformation of hour
hour_cos	Cosine transformation of hour
weekday_sin	Sinusoidal transformation of weekday
weekday_cos	Cosine transformation of weekday

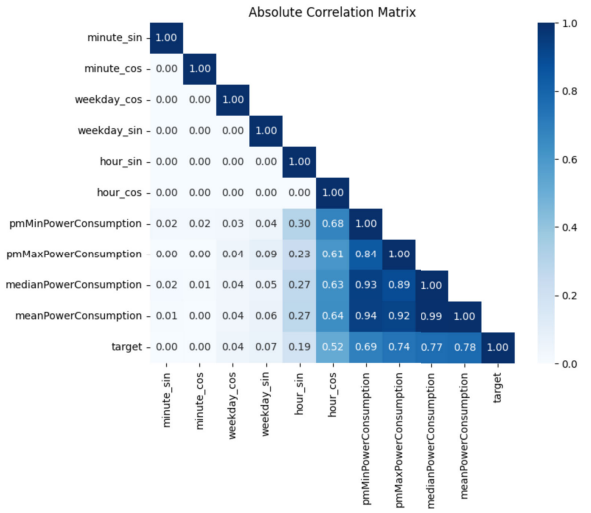


Fig. 4. Absolute Pearson correlation coefficients between the features and the target variable PRB-U.

complexity. Since all power-related features are derived from the same 6-second resolution data and therefore carry overlapping information, the inclusion of multiple lagged versions is considered redundant. Preliminary feature importance experiments using a *Temporal Fusion Transformer*, which is later introduced in Section III-A, indicate that *meanPowerConsumption* had the highest predictive importance, justifying its selection for temporal modelling.

Figure 4 shows the absolute Pearson correlation coefficients between the features listed in Table 1 and the target variable, PRB-U. All power-related variables showed strong correlations with the target variable and were therefore included. The hour feature in sine and cosine, showed

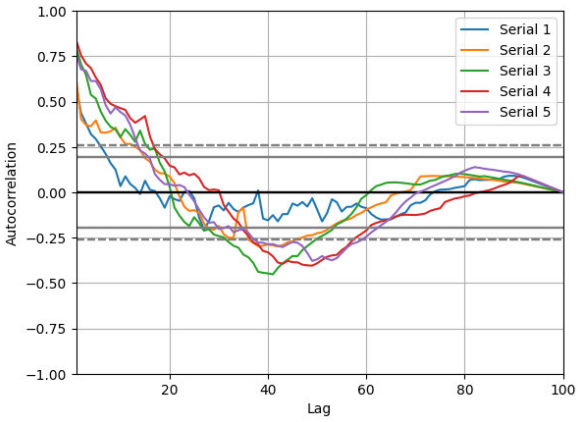


Fig. 5. Autocorrelation of PRB-U as a function of lag, where colors indicate different radio units.

TABLE 2. Class distribution in each data set.

Data Set	Positive	Negative	Negative to Positive
Training	158	18 272	116:1
Validation	66	4 059	62:1
Test	68	6 642	98:1

relatively high correlations. Other temporal features were less predictive, but they were kept based on unknowns about the radios' location. For example, if a cell site is located near a train station, it may have regular traffic surges at predictable minutes or weekdays.

Additionally, the autocorrelation function of the PRB-U time series is presented in Figure 5. For all radios, the autocorrelation is strongest at the smallest lag values, indicating pronounced short-term temporal dependence. This statistical property implies that the most recent PRB-U observation contains substantial predictive information for subsequent time steps in a one-step-ahead forecasting setting.

D. CLASS IMBALANCE

To examine class balance between high- and low-voltage states, the distribution of the target variable *PRB-U* was analyzed. As shown in Table 2, the dataset exhibits class imbalance, with the positive class consistently underrepresented across all data splits. In this formulation, the negative class corresponds to the lower voltage level, while the positive class denotes the higher voltage level. This predominance of low utilization was previously observed in the histograms shown in Figure 2.

E. EVALUATION METRICS

The performance of the models is evaluated using three main criteria: energy savings, number of underestimations, and F1 score. For regression models, additional metrics include mean squared error (MSE) and mean absolute error (MAE).

Energy savings are calculated as the difference between the static power curve (P_s) and the dynamic power curve

(P_{d}) over a given time interval $[t_1, t_2]$, where P is $P_{\mathrm{DC\,in}}$. This is approximated using a Riemann sum as specified in Equation (6). The derivation from power P to energy E is explained further in Appendix A.

$$E_{\mathrm{saved}} = \int_{t_1}^{t_2} \left(P_{\mathrm{s},t} - P_{\mathrm{d},t}\right)\, dt \;\approx \sum_{t=t_1}^{t_2} \left(P_{\mathrm{s},t} - P_{\mathrm{d},t}\right) \tag{6}$$

The number of false negatives, denoted as underestimations, is computed as the number of times the binary prediction $\hat{y}$ (0 or 1) is below the actual prediction y. Equation 7 shows the formal calculation used.

$$\text{Underestimations} = \sum_{i=t_1}^{t_2} \mathbf{1}_{\{\hat{y}_i < y_i\}} \tag{7}$$

The F1 score is defined as the harmonic mean of precision and recall [27], as:

$$\text{F1 Score} = 2 \cdot \frac{\text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}}. \tag{8}$$

The MSE and MAE for the regression models are given by (9) and (10).

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \tag{9}$$

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |\hat{y}_i - y_i| \tag{10}$$

III. MACHINE LEARNING FRAMEWORK

This section presents the machine learning framework used to evaluate the proposed system. It begins with an overview of the predictive models in Section III-A. Section III-B then describes how these models are adapted for the task. The hyperparameter tuning strategy is presented in Section III-C, and Section III-D describes the complexity and inference cost analysis. Finally, Section III-E introduces the SHAP framework used for model interpretability.

A. MODELS

In this study, several machine learning models are evaluated. Two tree-based models are considered: *Random Forest* [28] and *Extreme Gradient Boosting (XGBoost)* [29]. Additional models include *K-Nearest Neighbors* (KNN) [30] and *Support Vector Machine (SVM)* [31]. The neural network models include a *Feedforward Neural Network (FFNN)*, *Long Short-Term Memory (LSTM)* [32], *Convolutional Neural Network (CNN)* [33], and the *Temporal Fusion Transformer (TFT)* [13], a state-of-the-art architecture designed for multi-horizon time-series forecasting.

The *Temporal Fusion Transformer* (TFT) is a transformer based model designed specifically for time series forecasting [13]. It processes different types of input, including static metadata and time varying known and unknown variables, separately, which improves its ability to effectively leverage structured time series data. The model architecture integrates LSTMs and Gated Residual Networks (GRNs), enabling dynamic feature selection and adaptive complexity based on input relevance and data characteristics.

Additionally, we include an *Autoregressive Integrated Moving Average (ARIMA)* model [34] as a classical time series baseline, along with several moving average benchmarks. ARIMA is widely adopted in traffic and resource utilization forecasting studies and serves as a standard linear reference model for short-horizon prediction tasks [35]. The moving average benchmarks, denoted as *Naive_mean_t*, where t indicates the number of past observations included in the mean, provide simple persistence-based references. Such naive predictors are commonly used to establish lower performance bounds in one-step-ahead forecasting settings. As illustrated by the autocorrelation analysis in Figure 5, the PRB-U time series exhibits strong short-term persistence. This further motivates the inclusion of classical linear baselines such as ARIMA and simple moving-average predictors for one-step-ahead forecasting.

It should be noted that the proposed task focuses on one-step-ahead prediction. While recent state-of-the-art deep learning models, such as the TFT, have demonstrated strong performance in long-horizon time series forecasting, short-horizon prediction problems are often well-approximated by recent observations. Therefore, both classical linear baselines and advanced deep temporal models are included to provide a comprehensive evaluation across methodological paradigms.

B. TASK SPECIFIC ADAPTATION OF MODELS

We adapt the architectures of the FFNN, LSTM, and CNN models to suit the specific characteristics of the task, rather than relying on standard implementations. In particular, the LSTM and CNN models are implemented in a hybrid structure with separate processing streams for temporal and non-temporal features, referred to as HybridLSTM and HybridCNN. Processing all features jointly through temporal layers would introduce unnecessary model complexity and increase the number of recurrent parameters. Moreover, mixing non-sequential features into temporal processing may dilute the temporal inductive bias of the model, as such layers are designed to capture structured temporal correlations. By separating temporal and non-temporal feature streams, the architecture preserves the intended modeling capacity of each component while reducing unnecessary computational overhead. This consideration is particularly important for the deployment in radio units, where computational resources are constrained.

Given these deployment constraints, the feature set was deliberately kept compact. While additional engineered temporal features could be introduced, they would increase input dimensionality without clear empirical benefit, particularly given the one-step forecasting objective and the rapidly decaying autocorrelation structure shown in Figure 5. Consequently, lag length was selected as the primary tunable temporal feature.

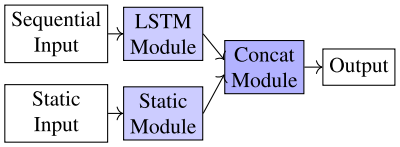


Fig. 6. Overview of the HybridLSTM architecture with two separate input streams.

The FFNN consists of a single input stream containing all features, as it does not explicitly model temporal dependencies. The number of hidden layers was treated as a hyperparameter to adapt model capacity to the chosen lag length. Larger lag values increase input dimensionality and temporal context, potentially requiring greater representational capacity, whereas shorter inputs risk overparameterization if the network is too deep. Residual connections were included as a tunable hyperparameter, as they are known to facilitate optimization and improve gradient flow in deeper networks [36]. Since network depth was itself tuned, residual connections may provide benefits when deeper architectures are selected, while offering limited advantage for shallower configurations. They were therefore treated as a tunable architectural component.

Batch normalization (BN) was applied after each linear transformation to improve training stability and convergence [37]. BN is widely adopted as a standard architectural component in modern neural networks and primarily influences optimization dynamics rather than model capacity. For this reason, BN was fixed across all configurations. Preliminary experiments indicated that removing BN reduced training stability without improving generalization. Additionally, BN adds negligible parameter overhead relative to the dense layers while consistently improving optimization stability. In contrast, dropout directly controls regularization strength and was therefore treated as a tunable hyperparameter (range 0–0.5). The full hyperparameter space for all models is described in Appendix C, and detailed descriptions of the FFNNs architecture can be found in Appendix B.

Figure 6 illustrates the LSTM architecture. The input is partitioned into sequential and static feature streams. The sequential features pass through one to four stacked LSTM layers, with depth treated as a hyperparameter. The static features pass through two fully connected layers with ReLU activation functions, whose dimensionalities are independently tuned. The outputs of both branches are concatenated and fed to the final prediction layer. Detailed descriptions of the LSTM module, static module, and concatenation module are provided in Appendix B.

Figure 7 illustrates the structure of the HybridCNN model. As in the LSTM architecture, the input is divided into sequential and static feature streams. However, the two sequential signals, PRB utilization and mean power consumption, are processed in separate branches. This separation allows each temporal signal to be modeled with independently tuned convolutional hyperparameters,

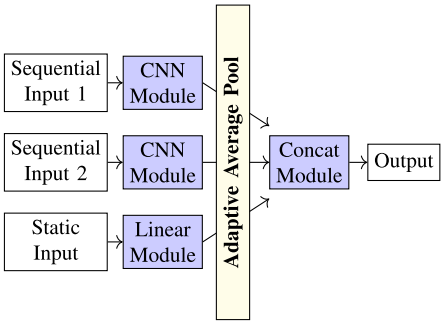


Fig. 7. Overview of the HybridCNN architecture with three separate input streams.

TABLE 3. Hyperparameter optimization setups used in optuna, Including the selection criterion, Optimization objective, and number of trials.

Notation	Selection Criterion	Optimization Function	Optuna Trials
Classification	F1 score	F1 score	100
Regression F1	F1 score	MSE	100
Regression MSE	MSE	MSE	100

reflecting their potentially different statistical characteristics. The static features are processed in a dedicated fully connected branch. In the LSTM, both temporal signals are processed jointly because recurrent units learn shared temporal dynamics efficiently [32]. In the CNN, convolutional filters are more sensitive to signal scale and local structure, so separating branches allows independent receptive field and channel allocation.

Detailed architectural configurations of the CNN are provided in Appendix B.

C. HYPERPARAMETER OPTIMIZATION

Hyperparameter optimization is performed using Optuna [38], an open-source framework for automated hyperparameter search. Optuna employs Bayesian optimization methods such as the Tree-structured Parzen Estimator (TPE) [39] to guide the search over the parameter space. In this work, the *TPESampler* is used for sampling candidate configurations, combined with the *MedianPruner* to terminate unpromising trials early.

For each model, three distinct sets of optimal hyperparameters are obtained, corresponding to the three Optuna setups described in Table 3. Each setup–model combination is optimized over 100 trials. The F1 score is used as the selection criterion to guide the hyperparameter search in two setups, as it prioritizes the correct identification of *true positives* over *true negatives*. This choice is particularly relevant given the strong class imbalance in the data: only approximately 1% of the instances belong to the positive class. In the third setup, MSE is used as the selection criterion instead, following a standard regression-oriented approach.

TABLE 4. Dominant prediction-time complexity of the evaluated models. The expressions indicate proportional scaling with the main architecture-dependent factors and omit lower-order operations such as activation Functions, Normalization, Dropout, Pooling, and element-wise additions.

Model	Dominant prediction-time cost
KNN	$\propto N_{\text{train}} d$
SVC/SVR	$\propto N_{\text{SV}} C_K$; $\propto N_{\text{SV}} d$ for linear/RBF kernels
RF	$\propto T_{\text{RF}} D_{\max}$
XGBoost	$\propto T_{\text{XGB}} D_{\max}$
FFNN	$\propto d_{\text{in}} H + B H^2$
HybridLSTM	$\propto \tau(F_{\text{seq}} H + K H^2)$
HybridCNN	$\propto \tau_1 C_1 k_1 + \tau_2 C_2 k_2$
TFT	$\propto T_{\text{seq}} H^2 + T_{\text{seq}}^2 H + d_s H$

The hyperparameter search space for each model can be seen in Appendix C. The optimal hyperparameter setup for each model can be seen in Appendix D.

D. MODEL COMPLEXITY AND INFERENCE

All models in this study are trained offline. Therefore, the relevant deployment cost is the prediction-time overhead rather than the training complexity. First, the dominant operations performed during inference are considered: distance computations for KNN [40], kernel evaluations for SVC/SVR [31], tree traversal for RF and XGBoost [28], [29], and forward-pass operations for neural-network-based models [32], [41], [42]. The resulting dominant prediction-time complexities are summarized in Table 4, where N_{train} is the number of stored training samples, d is the input feature dimension, N_{SV} is the number of support vectors, and C_K is the cost of one kernel evaluation. For the tree-based models, T_{RF} and T_{XGB} denote the number of trees in the Random Forest and XGBoost models, respectively, and $D_{\max}$ is the maximum tree depth. For the FFNN, d_{in} is the input dimension, H is the hidden size, and B is the number of hidden-to-hidden blocks. For the HybridLSTM, τ is the lag length, F_{seq} is the number of sequential features per time step, H is the hidden size, and K is the number of recurrent layers. For the HybridCNN, τ_i, C_i, and k_i denote the input sequence length, number of output channels, and kernel size of convolutional branch i. For the TFT, T_{seq} denotes the number of temporal input steps, including the encoder/history window and decoder/forecast horizon, H is the hidden dimension, and d_s is the number of static features.

For models using a flattened input vector, such as the FFNN, the input dimension is given by $d_{\text{in}} = 11 + 2 \cdot \text{lag}$, where the constant term corresponds to static features and the lag-dependent term corresponds to flattened sequential features. Since the FFNN consists of one input projection followed by hidden blocks of equal width, the dominant cost comes from the input-to-hidden and hidden-to-hidden transformations.

For the recurrent models, the HybridLSTM cost scales linearly with the sequence length, since the recurrent computation is repeated for each time step. An LSTM layer computes four gate transformations at each time step: input, forget, cell, and output gates [32]. Therefore, the dominant LSTM cost depends on the sequence length, sequential feature dimension, hidden size, and number of recurrent layers.

For the HybridCNN, the dominant inference cost comes from the two one-dimensional convolutional branches. Since each branch uses one input channel, this cost scales with the input sequence length, the number of output channels, and the kernel size. The static and concatenation branches consist of small fully connected transformations and therefore contribute lower-order terms. The complexity derivations for the HybridLSTM and HybridCNN are further detailed in Appendix E.

For the TFT-based model, the dominant inference cost comes from the recurrent encoder/decoder, gated residual networks, and temporal self-attention [13]. The recurrent and feed-forward components scale with the number of temporal steps and the hidden dimension, while the masked multi-head self-attention introduces a quadratic dependence on the temporal sequence length due to pairwise interactions across time steps. Thus, the TFT complexity is represented using the dominant recurrent/feed-forward and attention terms.

The expressions in Table 4 characterize the dominant prediction-time scaling of each model family. For KNN and SVC/SVR, inference cost scales linearly with the number of stored training samples or support vectors, while tree-based models scale with the number and depth of the trees. For the neural-network models, the scaling depends on how temporal information is processed: the FFNN flattens lagged inputs into the input dimension, the HybridLSTM applies recurrent computations at each time step, the HybridCNN applies convolutional operations over the temporal input, and the TFT additionally includes a self-attention term with quadratic dependence on the sequence length.

The complexity terms describe asymptotic scaling and do not fully capture implementation-level overhead. Therefore, architecture-specific model descriptors and single-sample inference latency are also reported. The model descriptors correspond to stored samples for KNN, support vectors for SVC/SVR, trees and nodes for tree-based models, and trainable parameters for neural-network-based models. Inference latency is measured on 100 test samples over 10 repetitions and reported as the mean and standard deviation of the per-sample prediction time. All measurements are performed on the CPU under identical hardware and software conditions.

E. SHAP ANALYSIS

SHapley Additive exPlanations (SHAP) is a unified, model-agnostic framework for interpreting predictions by assigning each feature a SHAP value, which quantifies its contribution to the model output based on principles from game theory [18], [43]. These values provide an interpretable approximation of the original model and support both local and global interpretability [18].

SHAP analysis is applied to two models, one tree-based and one neural network-based. The two models are selected

because of their strong performance with respect to the trade-off in minimizing restarts and maintaining energy savings. The analysis is conducted to identify which input features influence their predictions and to understand how the models approach the trade-off between minimizing restarts and maintaining energy savings.

`TreeExplainer` is used for the feature importance analysis of the tree-based model, while `GradientExplainer` is applied to the neural-network-based model. GradientExplainer is chosen over `DeepExplainer` due to the presence of batch normalization layers, which are not well supported by the latter.

IV. EXPERIMENTAL RESULTS

This section begins by outlining the experimental setup in Section IV-A. The predictive performance of the models is then presented in Section IV-B, followed by complexity and inference results in Section IV-C and a SHAP-based interpretability analysis in Section IV-D. Finally, Section IV-E introduces a method for managing the trade-off between energy savings and underestimations.

A. SETUP

The models are trained and evaluated on five randomly selected radios. The data is aggregated across all five units, without any identifiers indicating which specific radio each data point originates from. This decision was made to increase the generalizability of the algorithms, and avoid problems such as including new radio units never seen by the models before.

Final evaluations are conducted using the best hyperparameters identified through Optuna for each model–setup combination (see Table 3). Performance is assessed using the F1 score, the number of underestimations, and the estimated energy savings. For the regression models, MSE and MAE are also included.

All tuned hyperparameters and their corresponding search ranges are provided in Appendix C, while the final selected values for each model are listed in Appendix D.

B. PREDICTION RESULTS

Table 5 shows the evaluation results for all machine learning models and selected baseline models, based on test data aggregated across all radios. The model names in Table 5 include suffixes indicating the task formulation and tuning objective. Models labeled with _c are trained as classifiers. Models labeled with _F1_r are trained as regressors and tuned using the F1 score, while models labeled with _r are also regressors but tuned using MSE. This notation is used consistently across several result tables.

The naive mean baseline models were implemented with up to 96 steps back in time (24 hours), four top-performers are included in Table 5. *Naive_mean_48* is included because it achieved the highest energy savings. *Naive_mean_3* for having the lowest MSE, *Naive_mean_2* for achieving the highest F1 score, and *Naive_mean_1*, since it resulted in the smallest number of underestimations. As previously described in Section II-E, the last number indicates how many past values are included in the mean.

TABLE 5. Evaluation results for all models, including benchmarks and machine learning methods, on the test dataset.

Model	F1 Score	Energy Saving (%)	# Underestimations
rf_c	0.29	12.43	42
rf_F1_r	0.15	12.79	62
rf_r	0.18	12.81	61
xgb_c	0.18	12.34	50
xgb_F1_r	0.20	12.76	59
xgb_r	0.19	12.79	60
knc_c	0.10	12.78	64
knr_F1_r	0.13	12.70	61
knr_r	0.00	12.82	68
svc_c	0.16	12.39	53
svr_F1_r	0.28	12.76	55
svr_r	0.03	12.81	67
ffnn_c	**0.35**	12.41	34
ffnn_F1_r	0.07	12.78	65
ffnn_r	0.03	12.80	67
lstm_c	0.23	12.65	54
lstm_F1_r	0.20	12.75	59
lstm_r	0.03	12.81	67
cnn_c	0.19	12.67	57
cnn_F1_r	0.10	12.78	64
cnn_r	0.06	12.72	65
tft_c	0.20	11.78	**31**
tft_F1_r	0.16	12.74	60
tft_r	0.00	12.82	68
ARIMA	0.09	12.74	64
naive_mean_1	0.19	12.60	55
naive_mean_2	0.20	12.67	56
naive_mean_3	0.20	12.70	57
naive_mean_48	0.00	**12.83**	68

The worst case scenario for underestimations across the five radios is 68 while the optimal energy saving is 12.83%. As shown in Table 5, the FFNN classifier achieves the highest F1 score at 0.35 and records the second-lowest number of underestimations. The lowest energy saving is observed for the TFT classifier, at 11.78%, while all other models save at least 12.34% of the previously utilized kWh. In contrast, the TFT classifier achieves the best result in terms of minimizing underestimations. The highest energy savings, at 12.83%, is achieved by the *naive_mean_48* benchmark model, closely followed by the KNR and TFT regression models. However, these models also produce the maximum possible number of underestimations and an F1 score of 0, effectively never predicting the positive class correctly.

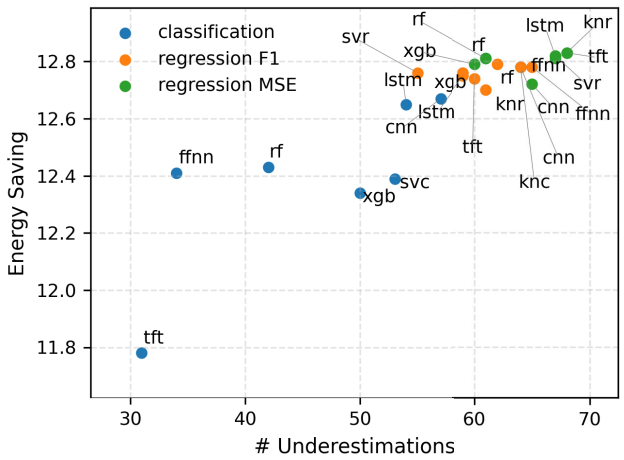


Fig. 8. Scatter summary of all evaluated models showing energy saving versus the number of underestimations across different training setups.

Figure 8 summarizes the percentage energy saved and number of underestimations. The figure is based on the statistics presented in Table 5. The scatter plot shows a distinct separation between the classification models and the regression-based approaches, classification algorithms performing better in terms of underestimations while regression models have a higher energy saving.

The bar charts in Figure 9 illustrate the performance of the three machine learning model setups across the three shared evaluation metrics: energy savings, number of underestimations, and F1 score. The first bar chart in Figure 9, shows a consistent trend in energy savings across all models. Notably, the only model that fails to achieve at least 12% energy savings is the TFT classifier.

The second bar chart in Figure 9 highlights a more varied performance in terms of underestimations. The classifier versions of the models generally perform best, followed by the regression model selected based on F1 score. In contrast, the regression model chosen using the more conventional MSE criterion exhibits the weakest performance.

The third bar chart in Figure 9 illustrates model performance in terms of F1 score across the different setups, which shows the greatest variation among the three evaluation metrics. The figure shows that the classification variant achieves the highest F1 score in five of the eight model types. The regression variant optimized for MSE performs the worst in almost all models, except for RF, where it ranks second. The regression variant optimized for F1 score generally achieves intermediate performance.

The regression models are evaluated on two additional metrics: MSE and MAE. The results are presented in Table 6. The RF regressor selected based on MSE (row 1) achieves the best performance on both metrics, closely followed by the same architecture selected based on F1 score (row 2). Additionally, the XGB model, also based

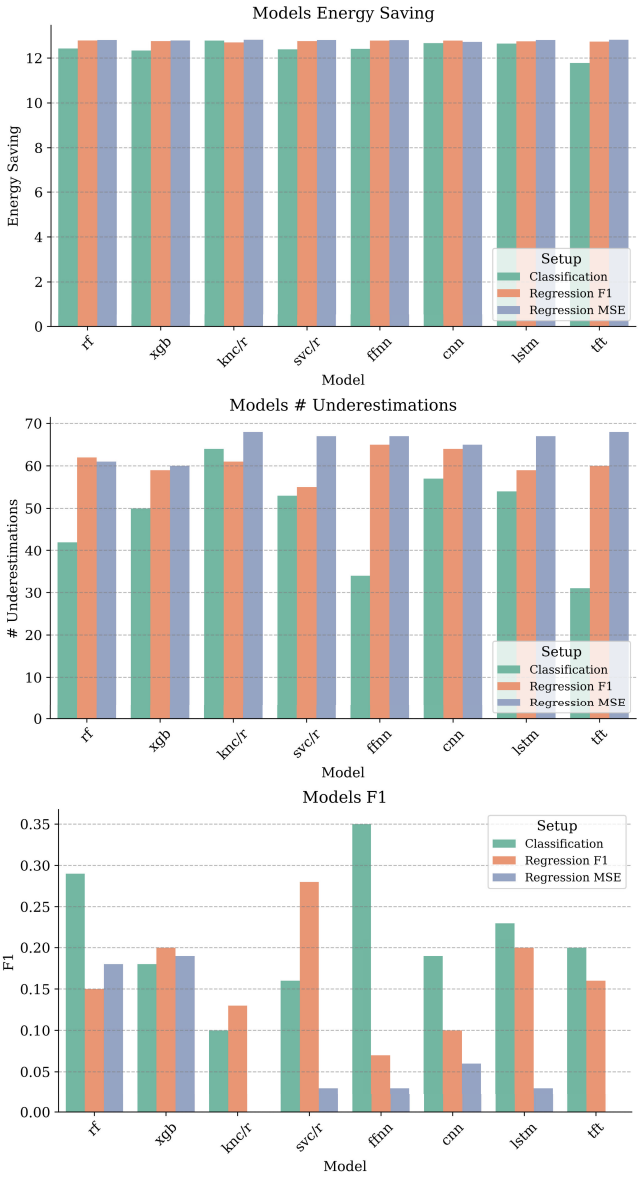


Fig. 9. Grouped bar charts of model performance in energy savings, underestimations and F1 score.

on decision trees, produces comparable results. The tree architectures are hence dominating in these metrics. In contrast, the TFT model selected using MSE shows the weakest performance on both MSE and MAE.

Selecting an optimal model for implementing dynamic voltage control in radios requires balancing the trade-off between maximizing energy savings and minimizing underestimations of voltage requirements. While greater energy savings can be achieved, this often increases the risk of failing to meet sudden spikes in power demand. Further research is needed to quantify the impact of underestimations relative to energy savings and to develop principled methods for weighting these factors in decision-making.

TABLE 6. Results from regression-specific evaluations.

model	MSE	MAE
rf_r	**0.00066**	**0.01407**
rf_F1_r	0.00068	0.01428
svr_r	0.00072	0.01451
svr_F1_r	0.00070	0.01506
knr_r	0.00073	0.01494
knr_F1_r	0.00108	0.01762
xgb_r	0.00069	0.01458
xgb_F1_r	0.00070	0.01460
ffnn_r	0.00073	0.01677
ffnn_F1_r	0.00080	0.01730
lstm_r	0.00081	0.01680
lstm_F1_r	0.00085	0.01779
cnn_r	0.00078	0.01628
cnn_F1_r	0.00095	0.02105
tft_r	0.00495	0.05320
tft_F1_r	0.00107	0.02001
ARIMA	0.00107	0.01971
naive_mean_1	0.00129	0.01945
naive_mean_2	0.00118	0.01936
naive_mean_3	0.00118	0.01988
naive_mean_48	0.00390	0.04822

As demonstrated in Figure 8, the RF and FFNN classifiers demonstrate particularly favorable performance in terms of this trade-off. Both achieve energy savings of approximately 12.4% while generating fewer underestimations than most other models. Moreover, these classifiers offer flexibility through adjustable decision thresholds, as discussed in Section IV-E.

The superior performance of RF and FFNN models can be explained by both the structural characteristics of the task and the properties of the dataset. The voltage switching problem is inherently threshold-based and operates on structured tabular inputs with limited short-term temporal dependence. RFs are well suited to such problems, as decision trees partition the feature space through threshold-based splits that naturally align with the binary switching objective. Moreover, tree ensembles are known to perform strongly on moderate-sized tabular datasets and are robust to noise and class imbalance.

The FFNN similarly benefits from the low-dimensional engineered feature representation. Since temporal dependencies are explicitly captured through lag features, the task reduces to learning a mapping in structured tabular space, where moderate-capacity feedforward networks often generalize effectively.

In contrast, CNN and LSTM architectures introduce stronger inductive biases and higher parameterization tailored to sequential feature extraction and adaptive memory modeling. Given the one-step prediction horizon, explicit lag encoding, and limited minority-class examples, these mechanisms may not provide additional benefit and may even hinder stable learning of rare high-voltage events.

The ARIMA and naive mean baselines perform noticeably worse than the learning-based models in predicting high PRB-U instances. Although the autocorrelation analysis in Figure 5 indicates strong correlation at short lag values, these methods rely on relatively simple linear or averaging dynamics. Consequently, they are unable to capture the more complex relationships between the input features and PRB-U dynamics, which limits their performance on this task.

Furthermore, the TFT introduces substantially greater model complexity. Given the moderate dataset size and the relatively simple temporal structure of this task, the additional capacity does not translate into improved performance and may instead lead to less stable training.

Overall, the classification-based models outperformed the regression-based models for the voltage switching task. A key reason is the alignment between the learning objective and the deployment objective. The control mechanism is inherently binary, selecting between two discrete PA voltage levels based on whether the predicted demand exceeds a threshold. Classification models directly optimize decision boundary separation, whereas regression models minimize continuous prediction error (e.g., MSE), which does not explicitly prioritize correct threshold decisions.

In addition, the dataset exhibits strong class imbalance, with only a small fraction of instances corresponding to high PRB-U values. In the classification formulation, this imbalance was explicitly addressed through class-weighted loss functions, which encouraged the models to prioritize the correct identification of rare high-demand instances. In contrast, the regression models optimized symmetric error objectives such as MSE, which penalize over- and underestimations equally and do not explicitly account for the rarity of high-voltage events.

This difference is particularly relevant because the voltage switching problem is inherently cost-asymmetric: misclassifying high-demand instances as low voltage may risk service degradation, whereas unnecessary high-voltage activation primarily incurs a marginal energy overhead.

Given the discrete actuation constraint of the two-level voltage system, directly modeling the binary decision proved more effective than estimating a continuous voltage trajectory. Additionally, while regression models may achieve lower point-wise voltage prediction error, the evaluation metric of interest in this application is switching accuracy and energy-performance trade-off, not continuous voltage estimation.

The performance of the models was also evaluated individually for each radio. Table 7 presents the potential energy savings and number of underestimations per radio under the assumption of perfect predictions. The energy savings represent the theoretical maximum achievable on this dataset, while the underestimation counts reflect the worst possible outcome for that metric.

TABLE 7. Upper range of metrics for each unit.

Radio	Energy Saving (%)	# Underestimations
1	12.87	3
2	13.04	30
3	12.75	2
4	13.01	0
5	12.42	33

The results in Table 7 indicate that the effectiveness of the proposed approach varies across radio units. The achievable energy savings range from 12.42% to 13.04%, while the number of restarts varies between 0 and 33. These differences can largely be explained by the distribution of PRB-U values across radios. Radios 1, 3, and 4 contain few instances in the higher PRB-U range, whereas radios 2 and 5 exhibit a greater concentration of such instances. It is likely that other units in the broader population may display even more extreme behaviors.

A detailed per-radio evaluation (reported in Appendix F) shows that all radios benefit from machine learning in terms of energy savings. However, the models struggle to capture sudden spikes in demand. In particular, none of the models detect the rare peaks in radios 1 and 3, while radio 4 exhibits no peaks at all.

In contrast, radios 2 and 5 achieve both energy savings and relatively low underestimation counts (Tables 31 and 34), indicating greater suitability for ML-based control. These results suggest that machine learning methods may be effective for certain radios but not universally beneficial, particularly under the two-voltage-level setting with the lower level fixed at 50%.

This behavior can be explained by differences in the traffic distributions across radios. For radios that never exceed the switching threshold, the optimal policy reduces to a constant low-voltage configuration, and no decision boundary needs to be learned. In such cases, machine learning offers no additional benefit beyond a static rule.

For radios with only very few above-threshold instances, the situation is still challenging. Although a decision boundary exists, the severe class imbalance limits the number of minority-class samples available for training. This reduces the statistical reliability of boundary estimation and increases uncertainty in predicting rare high-demand events. Consequently, the model's potential benefit is constrained by the limited learning signal.

In contrast, radios exhibiting more frequent transitions across the threshold provide informative samples near the decision boundary and meaningful switching opportunity, enabling ML models to learn discriminative patterns and achieve measurable gains under the two-level voltage constraint.

TABLE 8. Inference-related descriptors and average single-sample inference latency for the classical machine-learning models. Latency is reported as mean ± standard deviation, With median in parentheses.

Model	Descriptor	Value	Details	Latency [ms/sample]
Classification				
XGB	Tree nodes	16,349	–	0.0868 ± 0.2089 (0.0647)
SVC	Support vectors	1,285	RBF, $d = 153$	0.1030 ± 0.0077 (0.1035)
RF	Tree nodes	27,053	–	1.7010 ± 0.0327 (1.6943)
KNN	Stored samples	18,430	$d = 13$	0.3947 ± 0.0247 (0.3925)
Regression F1				
XGB	Tree nodes	22,032	–	0.0833 ± 0.1153 (0.0713)
SVR	Support vectors	7,708	Linear, $d = 203$	0.1540 ± 0.0046 (0.1533)
RF	Tree nodes	214,290	–	1.1189 ± 0.0277 (1.1140)
KNN	Stored samples	18,430	$d = 13$	0.2296 ± 0.0196 (0.2277)
Regression MSE				
XGB	Tree nodes	2,412	–	0.1249 ± 0.4128 (0.0613)
SVR	Support vectors	14,600	RBF, $d = 145$	0.8633 ± 0.0366 (0.8838)
RF	Tree nodes	227,663	–	1.8202 ± 0.0618 (1.8069)
KNN	Stored samples	18,428	$d = 13$	0.2176 ± 0.0184 (0.2153)

C. COMPLEXITY AND INFERENCE

Table 8 reports the main inference-related descriptors and single-sample inference latencies for the final classical machine-learning models. Since the dominant prediction-time cost differs between model families, the descriptors are defined as tree nodes for tree-based models, support vectors for SVC/SVR, and stored samples for KNN. The Details column reports the kernel type for SVC/SVR and the input feature dimension d where relevant.

Among the classical models, XGB achieved the lowest median latency across all three tasks, although its mean latency showed higher variability due to occasional timing outliers. RF had the highest latency in all three tasks, consistent with the need to traverse a larger ensemble of decision trees during each prediction. The Regression MSE SVR also showed higher latency than the other support-vector models, consistent with its larger number of support vectors and the use of an RBF kernel.

Table 9 reports the parameter count and single-sample forward-pass latency for the neural-network models. The FFNN models achieved the lowest latency across all three tasks, which is expected since they apply a fixed sequence of dense transformations to a flattened input vector. The HybridCNN models had slightly higher latency than the FFNN models despite having relatively small parameter counts, reflecting the additional latency from convolution, pooling, and reshaping operations. The HybridLSTM models were slower than the FFNN and HybridCNN models because their gated recurrent computations are repeated over the input sequence. The TFT models had the highest latency, consistent with their larger architectures and the use of recurrent components, gated residual networks, and temporal attention.

The FFNN models also achieved lower latency than the classical models, indicating that they provide a favorable trade-off between predictive performance and inference cost among the evaluated approaches.

Computational complexity, architecture-specific descriptors, and measured latency capture different aspects of

TABLE 9. Parameter count and average single-sample forward-pass latency forneural-network models. Latency is reported as mean ± standard deviation,with median in parentheses.

Model	Params	Latency [ms/sample]
Classification		
FFNN	79.7k	0.0370 ± 0.0051 (0.0359)
CNN	23.9k	0.0841 ± 0.0050 (0.0834)
LSTM	24.5k	0.1909 ± 0.0282 (0.1836)
TFT	5.50M	7.9885 ± 0.5845 (7.8063)
Regression F1		
FFNN	41.7k	0.0167 ± 0.0016 (0.0166)
CNN	3.0k	0.0777 ± 0.0035 (0.0771)
LSTM	337.0k	0.3812 ± 0.0379 (0.3695)
TFT	1.38M	3.7104 ± 0.1365 (3.6743)
Regression MSE		
FFNN	6.2k	0.0152 ± 0.0016 (0.0148)
CNN	9.7k	0.0783 ± 0.0082 (0.0769)
LSTM	26.2k	0.2782 ± 0.0197 (0.2737)
TFT	132.6k	2.6015 ± 0.2751 (2.5075)

inference cost. Therefore, both the dominant prediction-time complexity in Section III-D and the single-sample latency reported in this section are relevant. The latency results should mainly be interpreted as a relative comparison under identical CPU-based experimental conditions, since absolute inference time may differ on deployment hardware. Nevertheless, all evaluated models show low single-sample latency relative to the voltage-control interval considered in this study. For deployment, model complexity, memory requirements, and inference cost should be considered when selecting the final model architecture.

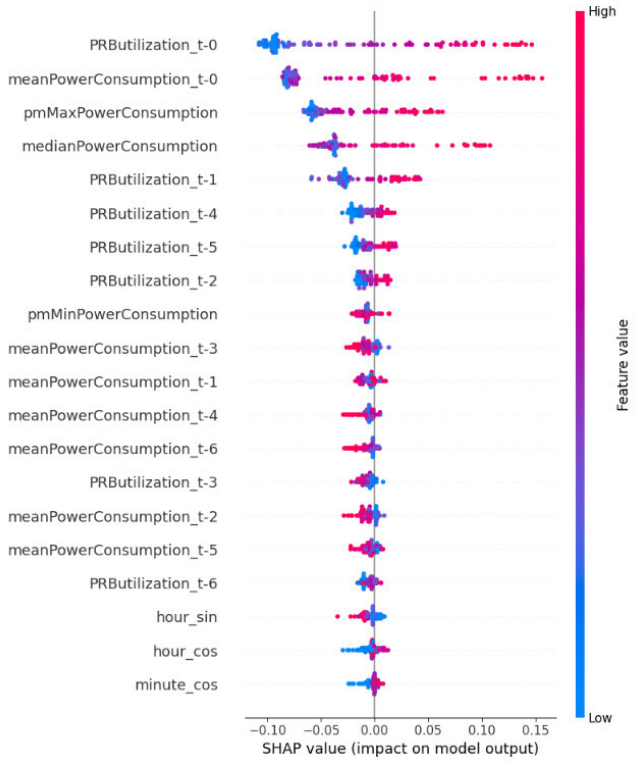


Fig. 10. SHAP values from the RF model for class 1 (high voltage).

D. EXPLAINABILITY

An explainability analysis is conducted on two representative models to identify the features that most strongly influence the predictions. Among the eight evaluated models, the RF and the FFNN classifiers are selected for this purpose. The RF classifier is chosen as it achieved the lowest number of restarts among the statistical models (see Table 5), indicating stability within this category. The FFNN classifier is included as a representative neural network–based model. Although the TFT achieved a slightly lower number of restarts, its energy-saving performance was substantially lower. Therefore, the FFNN is considered a more balanced model in terms of the trade-off between stability and energy efficiency.

Figure 10 presents the global SHAP summary plot for the RF classifier, illustrating the distribution of feature contributions to the prediction of class 1. The most influential feature is *PRB-U* at the current time step (t–0). Lower values of PRB-U are associated with negative SHAP values, reducing the probability of predicting class 1, whereas higher values yield positive SHAP contributions, increasing the likelihood of a high-level classification. The SHAP magnitudes for this feature range approximately from −0.10 to + 0.15 in probability space, indicating that instantaneous resource utilization exerts a substantial influence on the model output.

The dominance of PRB-U at the current time step is consistent with the pronounced short-term temporal dependency observed in the radio resource utilization data, as illustrated by the autocorrelation plot in Figure 5. Since the task involves predicting PRB-U at the subsequent time step, the most recent system state therefore contains substantial predictive information.

However, while short-term temporal dependency provides a strong predictive signal, it is insufficient to fully explain the superior performance of the RF classifier. Linear time-series baselines, including ARIMA and moving-average models, achieved substantially weaker performance in terms of the energy-saving versus underestimation trade-off. This indicates that the task cannot be adequately solved through a simple persistence or linear autoregressive mechanism alone. Although current PRB utilization constitutes the dominant feature, the consistent performance gap between the RF model and linear approaches suggests that more complex feature interactions and threshold effects play an important role in accurately identifying switching decisions.

The second most influential feature, mean power consumption at the current time step, exhibits SHAP values within a similar range (approximately −0.10 to + 0.15) as the PRB-U. Unlike PRB-U, however, both low and high values of mean power consumption are associated with positive and negative SHAP contributions depending on context. This indicates that its relationship with the predicted class is not strictly monotonic. Instead, the contribution of instantaneous power consumption depends on the surround-

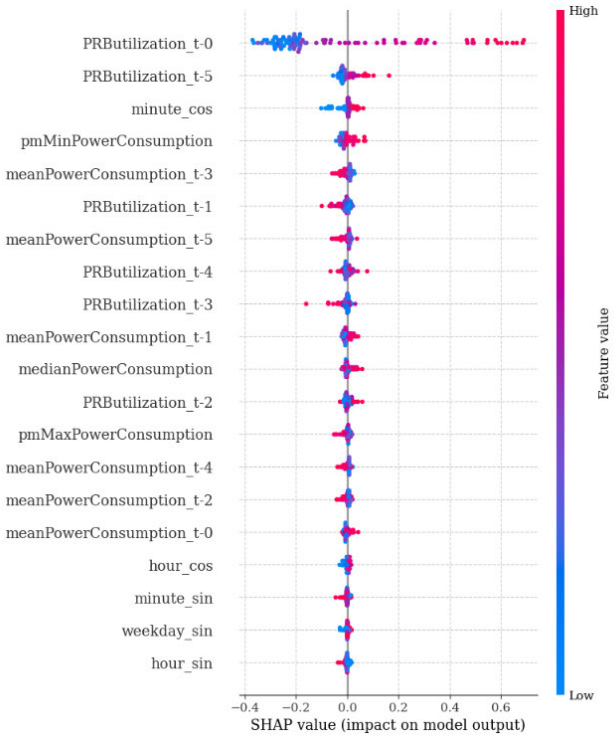


Fig. 11. SHAP values from the FFNN model for class 1 (high voltage).

ing feature configuration, further supporting the presence of nonlinear interactions captured by the RF model.

The remaining high-importance features are likewise associated primarily with the current time step, reinforcing the importance of recent temporal information in this prediction task. In contrast, static calendar-based features such as hour, minute, and weekday display comparatively small SHAP magnitudes. This suggests that the RF model relies primarily on short-term dynamic patterns in system load and power behavior rather than static temporal indicators.

For the FFNN model, the SHAP summary plot for class 1 predictions is presented in Figure 11. Similar to the RF classifier, the most influential feature is *PRB-U* at the current time step (t–0). However, the magnitude of its contribution is substantially larger, ranging approximately from −0.4 to + 0.6 in probability space. Lower values of PRB-U decrease the predicted probability of class 1, whereas higher values increase it. The wider SHAP range indicates that the FFNN places stronger emphasis on the instantaneous utilization level when forming predictions.

In contrast to the RF model, one static temporal feature, specifically the cosine encoding of the minute, exhibits moderately higher SHAP magnitudes in the FFNN (approximately −0.1 to + 0.1). This suggests that the FFNN incorporates certain periodic temporal patterns to a greater extent than the RF. Nevertheless, the remaining static features display comparatively small SHAP values, indicating that dynamic temporal variables remain the dominant contributors to the model output.

The larger SHAP magnitude assigned to PRB-U at t–0 suggests that this feature exerts a more concentrated influence in the FFNN than in the RF model, where importance is distributed more evenly across multiple features. This indicates that the FFNN relies more heavily on the instantaneous utilization level, whereas the RF integrates information from a broader set of temporal inputs. Such concentration of attribution may increase the FFNN's sensitivity to variations in PRB-U, while the more distributed attribution pattern of the RF could contribute to greater stability. However, a definitive assessment of robustness would require dedicated sensitivity analysis beyond SHAP-based attribution.

The SHAP analysis further suggests that several features exhibit consistently small attribution magnitudes across both models. This indicates that the feature space could potentially be reduced without substantial loss in predictive performance. However, a systematic feature selection study would be required to determine the minimal feature subset, which is left for future work.

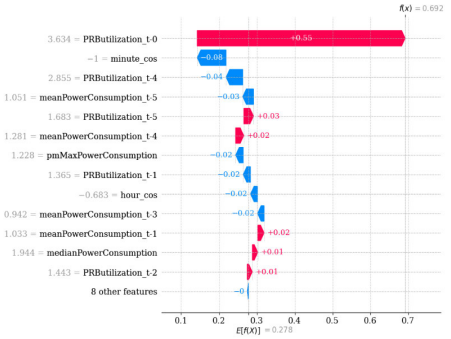
Figure 12 presents local SHAP explanations for individual predictions from both the RF and FFNN models. Subfigures (a) and (b) correspond to an instance correctly classified as class 1, while (c) and (d) illustrate a positive instance that is misclassified as class 0 (i.e., a false negative).

For the correctly classified instance, the FFNN assigns the largest contribution to PRB-U at the current time step, indicating strong reliance on instantaneous utilization. In contrast, the RF model distributes attribution more evenly across several features, with mean power consumption at the current time step receiving the largest contribution. This reflects the broader feature integration observed in the global SHAP analysis.

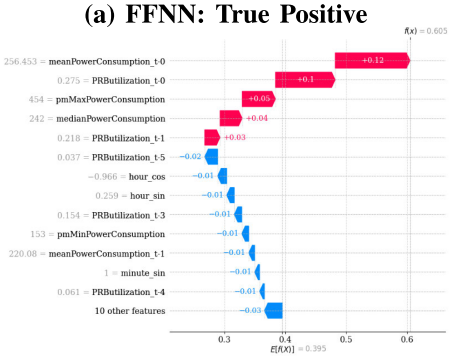
In the false negative case, both models exhibit a more distributed attribution pattern compared to the correctly classified instance, with multiple features contributing to the final prediction. This broader distribution suggests that no single feature provides a sufficiently strong signal to drive the prediction toward class 1. Despite incorporating information from several inputs, the models fail to assign a high probability to the positive class. The FFNN predicts a probability of 19%, whereas the RF assigns 46%, indicating that the RF is closer to the decision boundary in this case.

These observations are consistent with the global SHAP results: the FFNN demonstrates stronger reliance on PRB-U at the current time step, while the RF exhibits a more distributed attribution structure.

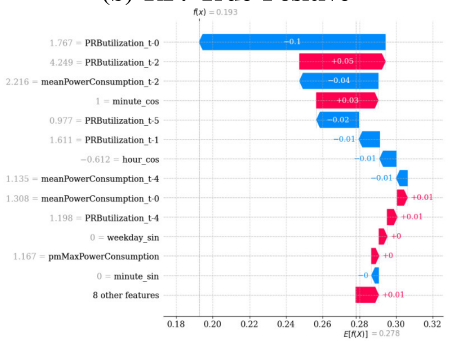
Figure 13 shows some SHAP explanations for the negative class. The subfigures (a) and (b) illustrate correctly classified negative instances, while (c) and (d) show instances that are mistakenly classified as positive (i.e., false positives). PRB-U at the current timestep remains the most influential feature for the FFNN in both cases. For the RF model, PRB-U at the current timestep is the most



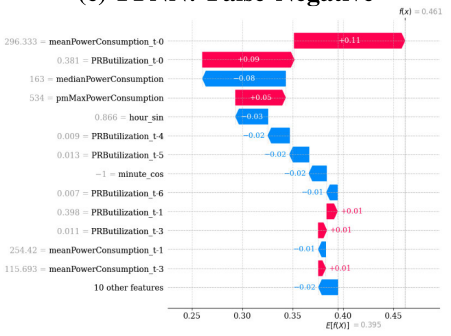
(a) FFNN: True Positive



(b) RF: True Positive

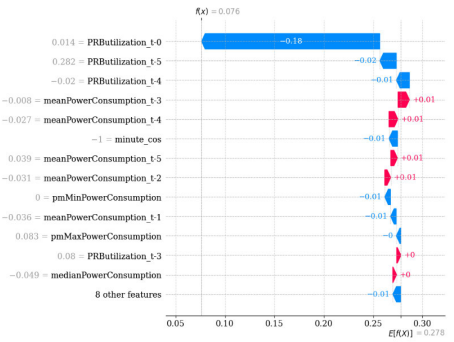


(c) FFNN: False Negative

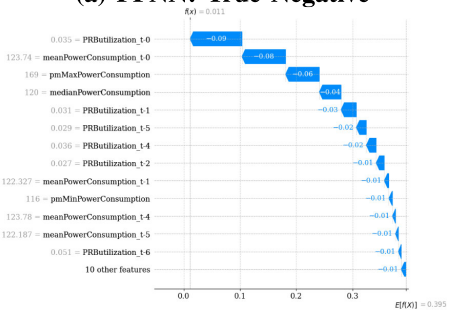


(d) RF: False Negative

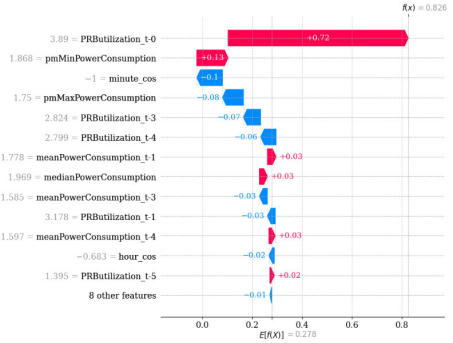
Fig. 12. SHAP explanations for positive samples. True positives and false negatives are shown for both models.



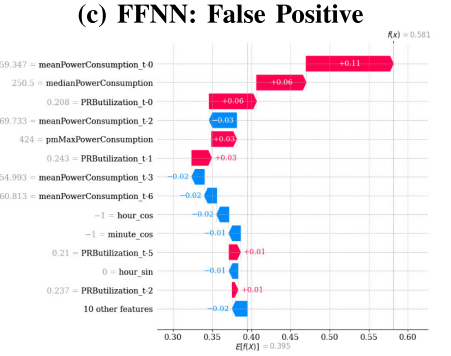
(a) FFNN: True Negative



(b) RF: True Negative



(c) FFNN: False Positive



(d) RF: False Positive

Fig. 13. SHAP explanations for negative samples. True negatives and false positives are shown for both models.

important feature in the true negative case, whereas mean power consumption at the current timestep dominates in the false positive case. For the false positive, the FFNN assigns a high probability of 83% to the positive class, while the RF model assigns a lower probability of 58%.

E. EXPLORING THE TRADE-OFF BETWEEN ENERGY SAVINGS AND UNDERESTIMATIONS

To analyze the trade-off between energy savings and underestimations, we conduct a threshold-shifting evaluation using the classification models' probability outputs. The

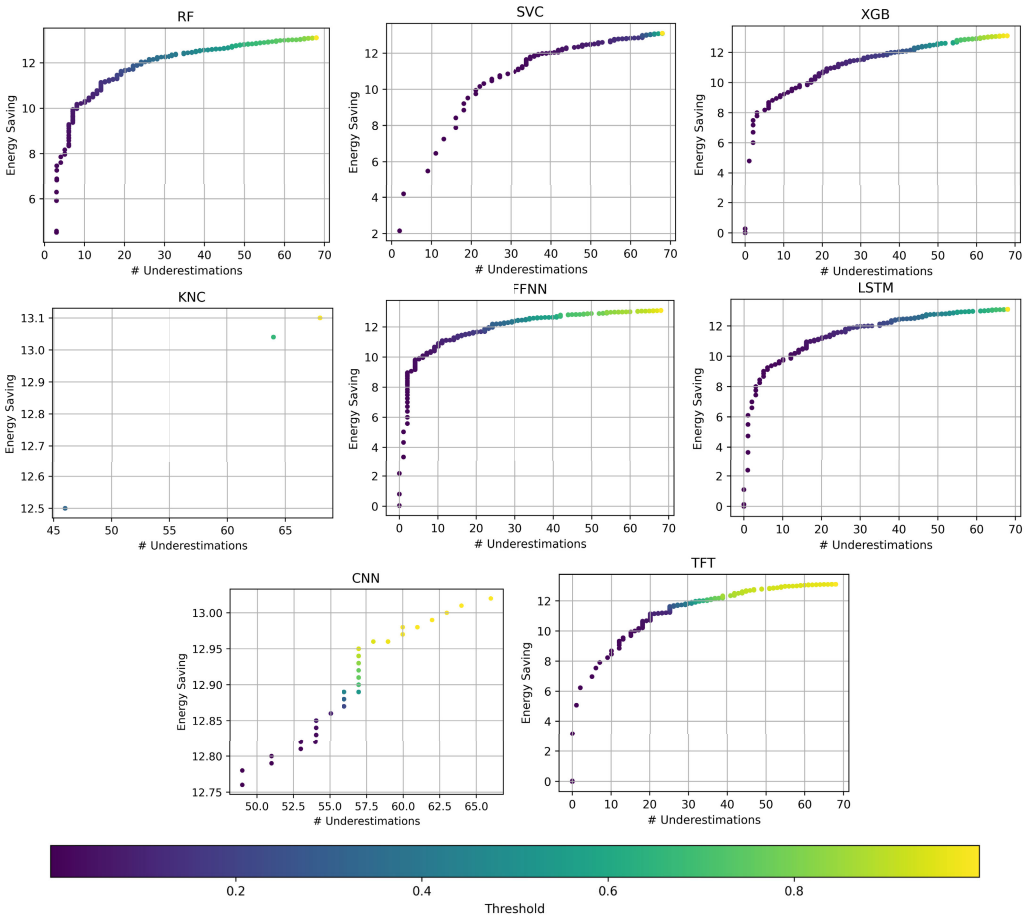


Fig. 14. **Model performance trade-offs under varying thresholds. Color encodes threshold level.**

classification threshold is varied from 0.01 to 0.99 to observe how different operating points affect model behavior. This enables exploration of how decision thresholds can be adapted to deployment-specific constraints, such as acceptable restart frequency or energy-saving targets.

Figure 14 illustrates the resulting trade-off curves for each model. This approach of varying the classification threshold demonstrates the explicit trade-off between the two evaluation metrics across models. The results indicate that some models are more sensitive to threshold tuning than others. For example, the SVC model exhibits a substantial increase in both restarts and energy savings even for small threshold shifts.

The FFNN model demonstrates the steepest improvement curve with a low number of restarts, achieving only 5 underestimations at an energy saving of 10%. It is followed by the HybridLSTM and RF models, both of which manage to achieve approximately 10% energy savings with no more than 10 restarts in total.

Although machine learning is found to be relevant for the dynamic voltage scenario in radios, an important question before implementation is how many underestimations are acceptable for a given customer and radio unit. Threshold shifting provides a simple mechanism to adjust this trade-off, enabling the same model to operate at different points on the energy–reliability spectrum depending on the deployment requirements.

In practice, this allows the decision threshold to be tuned according to operational priorities, for example prioritizing fewer restarts at the cost of reduced energy savings, or conversely allowing more aggressive energy reduction.

F. LIMITATIONS

It should be noted that ML models, particularly deep learning models, typically perform better when trained on large datasets. The dataset in this paper comprises approximately data 18,000 instances, which is moderate in size for structured time-series data. However, the dataset is highly imbalanced, with few instances in the positive class. This imbalance may have limited the models' ability to learn patterns associated with rare high-utilization events. As a result, the performance of the ML-based models presented in this work might improve if trained on a larger and more balanced dataset.

Additionally, the dataset consists of aggregated data from all five radio units, with no identifiers indicating the origin of

each data point. Since usage patterns vary across individual radios, it is uncertain whether patterns learned from one radio can be generalized to others. Improved results might be achieved by training separate models per radio unit or by incorporating features that distinguish between different radio types or configurations.

From a generalizability perspective, this study evaluates the proposed voltage-control approach using data from five radio units. Although the overall approach is intended as a general framework for reducing energy consumption by adapting the voltage level to predicted traffic demand, the trained models themselves may not generalize to radio units not included in the training data, particularly if these units exhibit traffic patterns that differ substantially from those observed in the training set. In practical deployment, models would therefore likely need to be trained or adapted using data from the target radio units, or from a broader and more representative set of radio units. While certain recurring patterns may be common, such as peaks being preceded by upward trends and dips by downward trends, as shown in Fig. 3, traffic patterns are expected to vary significantly across deployment scenarios. Furthermore, the data was collected only during winter and early spring, so seasonal variation in PRB utilization may affect the results. Broader temporal and deployment coverage is therefore needed to assess generalizability further.

In addition, some deployment contexts may exhibit distinct traffic behavior, such as rural units along train tracks, where passing trains can cause short but intense traffic bursts. Such patterns may affect both model generalizability and the achievable energy-saving benefit, since the gain from voltage adaptation depends on the temporal PRB-utilization pattern of the radio unit. Since the proposed approach is evaluated on a limited set of radio units, assessing its generalizability and benefit under substantially different deployment scenarios is left for future research.

Finally, the computational energy required for model inference is not explicitly measured. While inference latency is reported as an indicator of computational overhead, the processor-level energy cost in deployment is not measured.

V. CONCLUSION

This paper investigates machine learning approaches for dynamically controlling the supply voltage of the PA in radio units, enabling adaptation of the available RF output power capacity between two hardware-supported voltage levels.

The results show that machine learning is a promising strategy for voltage control over time (RQ1). However, dynamic voltage adaptation introduces a trade-off: while energy savings can be substantial, underestimations risk causing connectivity disturbances. Additionally, performance varies across radio units—some benefit less from ML and may perform as well or better with simple benchmark models or static voltage settings. This is particularly evident for radios with very few high-voltage occurrences, where limited peak examples make it difficult for models to learn the patterns preceding high-demand events.

Of the explored approaches, classification outperforms regression in minimizing underestimations (RQ2). For regression models, tuning based on F1 score rather than MSE is more effective at reducing underestimations without compromising energy savings. This highlights a mismatch between standard regression training objectives and the operational goal of avoiding missed high-demand peaks, as well as the absence of asymmetric loss terms penalizing missed high-demand events.

The RF classifier and a custom FFNN emerge as strong candidates for the task (RQ3). The RF achieves 42 underestimations with 12.43% energy savings, while the FFNN has 34 underestimations and saves 12.41% energy. These results suggest that capturing nonlinear relationships in the data is beneficial, however more complex temporal architectures provide limited additional value for this task. The FFNN also showed low inference latency, making it a favorable candidate when both predictive performance and computational cost are considered.

Explainability analysis reveals that both models rely primarily on current PRB-U values (RQ4). The RF distributes importance more evenly across the features while the FFNN shows stronger dependence on a single dominant input. This is consistent with the strong temporal correlation between current and next-step PRB-U values. However, the inferior performance of purely linear models suggests that a linear dependence on this feature alone is insufficient.

A key advantage of the classification setup is access to prediction probabilities, which allows threshold tuning to balance energy savings and underestimations. With this approach, the FFNN classifier achieves only 5 underestimations with 10% energy savings.

Overall, this work contributes to the development of energy-aware radio systems by demonstrating that machine learning can effectively control PA voltage levels to save energy in radio units. The study highlights a previously underexplored optimization dimension in radio systems: adapting the hardware-supported PA voltage level to match predicted traffic demand. Even with only two voltage levels, the results indicate that energy savings of approximately 12% are achievable without introducing unacceptable reliability risks.

FUTURE WORK

Future work should investigate the real-world impact of voltage underestimations on service quality to guide model design and deployment strategies. In addition, exploring scenarios with more than two voltage levels may further improve the achievable trade-off between energy efficiency and hardware design cost.

APPENDIX

The first Appendix, A, details the calculation of energy savings from power changes. The following Section B details

the architectures of FFNN, LSTM, and CNN. Following, C details the full search spaces for each model and D details the optimal space after tuning the models.

A. ENERGY SAVING CALCULATION

This appendix explains how the saved energy is calculated.

We consider two power curves defined over discrete time steps and aim to compute the difference in total energy consumption between them. The static power curve represents the power consumed under a constant, high voltage throughout the interval, while the dynamic power curve represents the power when the voltage varies over time. Power is calculated as the product of current and voltage, as shown in Equation 11:

$$P = I * V \tag{11}$$

The power at time t for the static and dynamic case can therefore be expressed as:

$$P_{\text{static},t} = V_{\text{static}} \cdot I_{static,t} \tag{12}$$

$$P_{\text{dynamic},t} = V_{\text{dynamic},t} \cdot I_{dynamic,t} \tag{13}$$

where $V_{\text{dynamic},t} \leq V_{\text{static}}$ for all discrete time steps $t \in \{1, 2, \ldots, T\}$.

To relate energy and power, we use the fundamental definition of instantaneous power (P) as the rate at which work (dW) (or energy) is done [44]:

$$P = \frac{dW}{dt} \tag{14}$$

Rearranging this gives the energy consumed during a specific time step i, where P_i is the power during the time step, and Δt is the time step duration. Here, P_i is assumed to be constant over the duration Δ t.

In the continuous case, energy is defined as the integral of power over time. Thus, over a time interval $[t_1, t_2]$, the energy consumed along a power curve is given by:

$$W = \lim_{\Delta t \to 0} \sum_{i=t_1}^{t_2} P_i \cdot \Delta t = \int_{t_1}^{t_2} P(t)\, dt \tag{15}$$

To compute the *saved energy*, we take the difference between the static (unoptimized) power curve, $P_{\text{static}}(t)$, and the dynamically adjusted power curve, $P_{\text{dynamic}}(t)$, over a time interval $[t_1, t_2]$:

$$E_{\text{saved}} = \int_{t_1}^{t_2} \left(P_{\text{static}}(t) - P_{\text{dynamic}}(t)\right)\, dt \tag{16}$$

Since the system is sampled at discrete time steps, this integral is approximated using a Riemann sum:

$$E_{\text{saved}} \approx \sum_{t=t_1}^{t_2} \left(P_{\text{static},t} - P_{\text{dynamic},t}\right) \cdot \Delta t \tag{17}$$

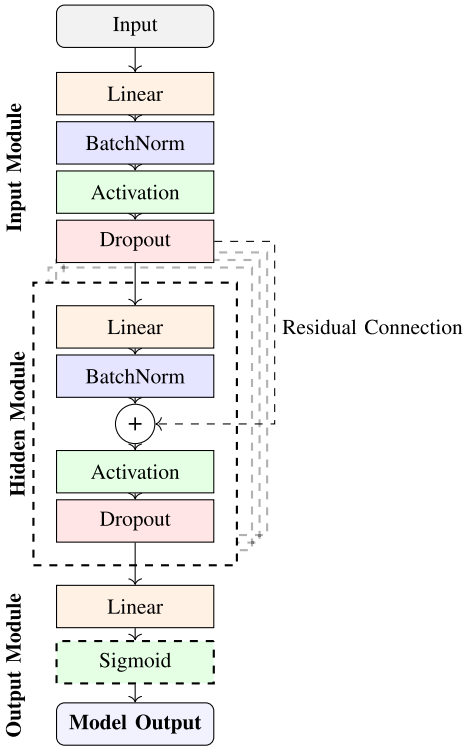


Fig. 15. FFNN architecture with stacked hidden layers.

TABLE 10. Scalers and their abbreviations.

Scaler	Abbreviation
EncoderNormalizer	encnorm
MinMaxScaler	minmax
RobustScaler	robust
StandardScaler	standard

B. MODEL ARCHITECTURES

This appendix presents the detailed architectures of the neural network models employed in the study.

Fig. 15 illustrates the architecture of the adjusted FFNN.

Fig. 16 shows the architecture of the LSTM, and Fig. 17 the CNN.

C. HYPERPARAMETER SEARCH SPACE

This appendix specifies the hyperparameter space which was investigated for each model. Table 10 shows abbreviations used for those including scaler options. Any cell marked with - indicates that the other task formulations have that parameter but it is not available in that specific one.

D. COMMON PARAMETERS

Several hyperparameters were shared across multiple models. Among these were the choice of scaler and the lag in the data. Scalers were adopted from the `Scikit-learn` library and the lag was set to be up to one day.

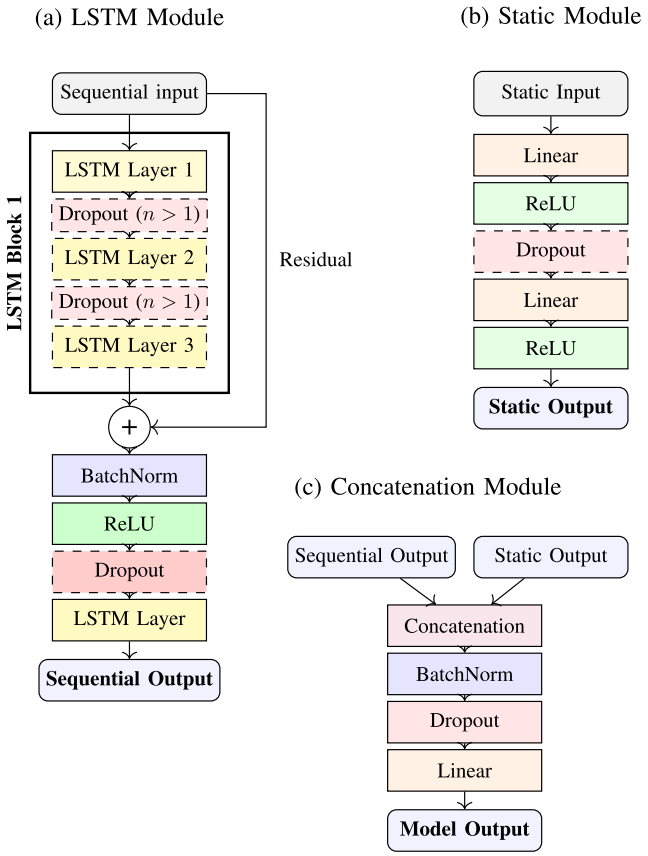


Fig. 16. Overview of the components of the HybridLSTM architecture.

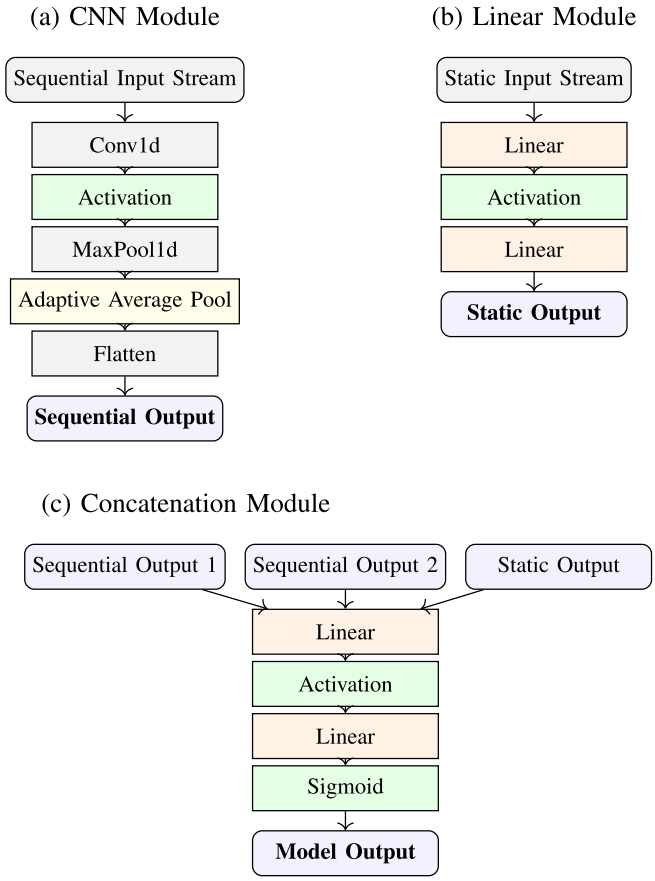


Fig. 17. Overview of the components of the HybridCNN architecture.

TABLE 11. Hyperparameter search space for data preprocessing.

Parameter	Type	Range
`scaler`	Categorical	`standard`, `minmax`, `robust`
`lag_value`	Integer	(1, 96)

For the neural network models, additional common hyperparameters included the optimizer, learning rate, and number of training epochs. All except the number of epochs were included as tunable parameters within the Optuna optimization process. Epochs was set to 125 across models.

E. STATISTICAL MODELS

The search spaces for the hyperparameters related to data preprocessing, which are shared across most non-neural models, are presented in Table 11. Where scaler is the scaler used for the data and lag the amount of look back the model gets. The hyperparameter search spaces specific to the different models are displayed in the following tables.

1) RANDOM FOREST REGRESSOR

Table 12 gives the optimal parameters found through optuna for RF. *n_estimators* refers to the number of trees in the forest, *max_depth* to the depth that the trees are allowed, *min_samples_split* is the minimum number of samples needed for a split, and *min_samples_leaf* the minimum number of samples required to create a node [45].

2) EXTREME GRADIENT BOOSTING

Similarly to RF, *n_estimators* indicated the number of trees [45]. *learning_rate* decreases the contribution of each tree by its rate, *max_depth* is as in RF the max allowed depth of the trees, *subsample* is the percentage of samples used to train each individual tree. Finally, *colsample_bytree*

3) K-NEAREST NEIGHBORS

The unsupervised ML algorithm was given the search space as shown in Table 14. First, *n_neighbors* is the number of neighbors used to make a prediction, *weights* decides how much each neighbors' value contributes with [45]. *leaf_size* aids in memory and speed of the algorithm, and *p* decides the function for distances.

4) SUPPORT VECTOR MACHINE

The *C* parameter controls the strength of the regularization [45]. The kernel coefficient is calculated according to the parameter *gamma*, *scale*, *epsilon* controls the radius out from which there is no penalty added during training phase.

TABLE 12. Hyperparameter search space for RF.

Parameter	Type	Range
n_estimators	Integer (log-uniform)	(50, 300)
max_depth	Integer (log-uniform)	(4, 20)
min_samples_split	Integer	(2, 10)
min_samples_leaf	Integer	(1, 10)

TABLE 13. Hyperparameter search space for XGB.

Parameter	Type	Range
n_estimators	Integer (log-uniform)	(50, 300)
learning_rate	Float (log-uniform)	(0.01, 0.2)
max_depth	Integer	(3, 10)
subsample	Float	(0.7, 1.0)
colsample_bytree	Float	(0.7, 1.0)

TABLE 14. Hyperparameter search space for KNC/R.

Parameter	Type	Range
n_neighbors	Integer	(2, 30)
weights	Categorical	uniform, distance
algorithm	Categorical	auto, kd_tree
leaf_size	Integer	(20, 50)
p	Categorical	1, 2

TABLE 15. Hyperparameter search space for SVC/R.

Parameter	Type	Range
C	Float (log-uniform)	(0.01, 10)
gamma	Categorical	scale, auto, 0.001, 0.01, 0.1
epsilon	Float (log-uniform)	(0.001, 0.1)
kernel	Categorical	linear, rbf

TABLE 16. Hyperparameter search space for neural networks' common hyperparameters.

Parameter	Type	Range
scaler	Categorical	standard, minmax, robust
lag_value	Integer	(1, 96)
batch_size	Categorical	16, 32, 64, 128
optimizer	Categorical	adam, rmsprop, sgd
lr	Float (log-uniform)	$(1 \times 10^{-5}, 1 \times 10^{-2})$
grad_clip	Boolean	True, False
hidden_size	Integer (step = 32)	(32, 256)
use_sigmoid	Boolean	True, False

Lastly, *kernel* is used to calculate the similarity or difference between data points. Selecting its function is crucial for creating a functional model.

F. NEURAL NETWORKS

The common search spaces are presented in Table. 16. *Scaler* describes the way data is transformed for the ML models, *lag* value defines the lag in the data and hence the information back in time that the models get. *Batch_size* is the number of samples used at a time during training, *optimizer* controls the change of the model during training phase. The learning rate *lr* is closely connected to the optimizer in that it chooses the amount of change that is done in every training loop. The parameter *grad_clip* can be activated to decrease the change of exploding gradients. *hidden_size* is the number of neurons in the hidden layers and *use_sigmoid* defines if a last sigmoid activation function is used in the model.

TABLE 17. Hyperparameter search space for FFNN.

Parameter	Type	Range
num_layers	Integer	(1, 4)
dropout_rate	Float (step = 0.1)	(0.0, 0.5)
activation_func	Categorical	relu, leaky_relu, elu, gelu
use_residuals	Boolean	True, False

TABLE 18. Hyperparameter search space for CNN.

Parameter	Type	Range
branch_out_channels1	Integer (step=8)	(8, 128)
branch_out_channels2	Integer (step=8)	(8, 128)
kernel_size1	Integer (step=1)	(1, min(10, lag))
kernel_size2	Integer (step=1)	(1, min(10, lag))
stride1	Integer (step=1)	(1, 7)
stride2	Integer (step=1)	(1, 7)
padding_type1	Categorical	same, valid
padding_type2	Categorical	same, valid
ad_avg_pool1	Integer (step=8)	(8, 128)
ad_avg_pool2	Integer (step=8)	(8, 128)
max_pooling_kernel1	Integer (step=1)	(1, 10)
max_pooling_kernel2	Integer (step=1)	(1, 10)
static_hidden	Integer (step=8)	(8, 128)
out_static	Integer (step=8)	(8, 128)
conc_hidden	Integer (step=8)	(8, 128)
out_to_concat	Integer (step=8)	(8, 128)
activation_seq1	Categorical	relu, leaky_relu, sigmoid, tanh
activation_seq2	Categorical	relu, leaky_relu, sigmoid, tanh
activation_concat	Categorical	relu, leaky_relu, sigmoid, tanh
activation_static	Categorical	relu, leaky_relu, sigmoid, tanh

TABLE 19. Hyperparameter search space for LSTM.

Parameter	Type	Range
lstm_layers	Integer	(1, 3)
lstm_dropout	Float (step = 0.1)	(0.0, 0.5
static_hidden_size_1	Integer (step = 16)	(16, 128)
static_hidden_size_2	Integer (step = 8)	(8, 64)

TABLE 20. Hyperparameter search space for the dataset configuration.

Parameter	Type	Range
max_encoder_length	Integer	(7, 60)
encoder_random_ratio	Float	(0.5, 1.0)
max_prediction_length	Integer	(3, 30)
add_relative_time_idx	Boolean	True, False
add_target_scales	Boolean	True, False
add_encoder_length	Categorical	''auto'', True, False
randomize_length	Boolean	True, False
target_normalizer	Categorical	auto, TorchNormalizer, GroupNormalizer encnorm
scaler	Categorical	StandardScaler, RobustScaler encnorm

TABLE 21. Hyperparameter search space for TFT.

Parameter	Type	Range
hidden_size	Categorical	8, 16, 32, 64, 128, 256
lstm_layers	Integer	(1, 3)
dropout	Float	(0.05, 0.4)
attention_head_size	Integer	(1, 8)
hidden_continuous_size	Categorical	4, 8, 16, 32
reduce_on _plateau_patience	Integer (log-uniform)	(5, 15)
share_single _variable_networks	Boolean	True, False
causal_attention	Boolean	True, False
learning_rate	Float (log-uniform)	(10^{-4}, 10^{-2})
batch_size	Categorical	32, 64, 128

TABLE 22. Optimal hyperparameters RF.

Hyperparameter	C	R(F1)	R(MSE)
lag_value	6	96	96
max_depth	19	17	15
n_estimators	165	92	155
min_samples_leaf	10	3	7
min_samples_split	3	9	4

1) FEED FORWARD NEURAL NETWORK

Num_layers is the number of hidden layers in the neural net, *dropout_rate* prevents overfitting by setting a neuron's output to 0 with the specified rate. The *activation_func* controls the activation function used and *use_residuals* controls whether residuals are used.

TABLE 23. Optimal hyperparameters XGB.

Hyperparameter	C	R(F1)	R(MSE)
lag_value	79	94	6
max_depth	10	6	4
n_estimators	199	254	88
subsample	0.77	0.97	0.98
colsample_bytree	0.72	0.78	0.83
learning_rate	0.019	0.057	0.12

TABLE 24. Optimal hyperparameters KNC/R.

Hyperparameter	C	R(F1)	R(MSE)
lag_value	1	1	1
algorithm	auto	auto	kd_tree
leaf_size	34	23	35
n_neighbors	3	2	29
p	2	1	2
weights	uniform	distance	distance
scaler	standard	robust	robust

TABLE 25. Optimal hyperparameters SVC/R.

Hyperparameter	C	R(F1)	R(MSE)
lag_value	71	96	67
C	6.99	1.09	0.11
gamma	auto	scale	scale
kernel	rbf	linear	rbf
epsilon	-	0.0097	0.0010
scaler	robust	minmax	minmax

TABLE 26. Optimal hyperparameters FFNN.

Hyperparameter	C	R(F1)	R(MSE)
lag_value	5	74	25
activation_func	relu	leaky_relu	elu
dropout_rate	0.3	0.2	0.4
gradient_clip	True	True	True
hidden_size	192	256	96
lr	0.000010	0.00027	0.00067
num_layers	3	1	1
use_residuals	False	True	True
use_sigmoid	False	False	False
batch_size	64	64	128
scaler	robust	robust	robust

2) CONVOLUTIONAL NEURAL NETWORK

Table 18 shows the hyperparameter space for the hybrid CNN model. 1 and 2 in the end of the first 12 parameters indicates whether it is applied to the first or second convolutional block. The channels created by the convolution is controlled by *branch_out_channels*, *kernel_size* is the size of the kernel convolving over the sequence. Kernel size is defined to never be larger than the input size, lag, as it is impossible to convolve a larger kernel than the input.

TABLE 27. Optimal hyperparameters LSTM.

Hyperparameter	C	R(F1)	R(MSE)
gradient_clip	True	False	False
hidden_size	32	128	32
lag_value	5	22	17
seq_feat_size	2	2	2
lr	0.000034	0.000064	0.000055
lstm_dropout	0.3	0.5	0.5
lstm_layers	2	2	2
scaler	minmax	standard	minmax
static_hidden_size_1	64	128	64
static_hidden_size_2	32	24	56
batch_size	64	64	64
optimizer	adam	adam	adam

TABLE 28. Optimal hyperparameters CNN.

Hyperparameter	C	R(F1)	R(MSE)
activation_concat	tanh	relu	leaky_relu
activation_seq1	relu	relu	tanh
activation_seq2	leaky_relu	tanh	leaky_relu
activation_static	relu	sigmoid	relu
ad_avg_pool1	40	64	40
ad_avg_pool2	64	8	96
branch_out_channels1	64	16	96
branch_out_channels2	64	96	32
conc_hidden	128	8	48
gradient_clip	True	True	True
kernel_size1	10	7	3
kernel_size2	7	3	2
lag_value	85	96	95
max_pooling_kernel1	8	3	4
max_pooling_kernel2	2	10	6
out_static	88	16	112
out_to_concat	48	40	56
padding_type1	valid	valid	same
padding_type2	same	same	same
static_hidden	40	56	8
stride1	1	7	3
stride2	3	2	1
optimizer	nan	nan	adam
lr	0.00045	0.00010	0.00050
scaler	minmax	standard	robust

The *stride* defines the jump after each convolution, *padding* parameter is the potential extra input added to the beginning and end of the sequence. The parameter *ad_avg_pool* controls the size of the output of the AdaptiveAvgPool1d layer, which takes a mean and creates an output of the specified size. The *max_pooling* parameter controls the kernel_size parameter of the MaxPool1d layer [46].

Static_hidden is the neuron size of the non-sequential hidden layer and *out_static* of its output layer. *Conc_hidden* is the neuron size of the concatenation later. *Out_to_concat* controls the resize using a AdaptiveAvgPool1d layer of each individual output before entering the concatenation layer. The *activation* parameters control the activation function for each block as indicated by its suffix.

TABLE 29. Optimal hyperparameters TFT.

Hyperparameter	C	R(F1)	R(MSE)
lag_value	49	26	43
add_encoder_length	-	-	False
add_relative_time_idx	False	True	False
add_target_scales	-	-	True
attention_head_size	2	8	5
causal_attention	False	False	True
dropout	0.25	0.15	0.08
encoder_random_ratio	0.75	0.53	0.90
epochs	-	-	170
hidden_continuous_size	32.00	4	32
hidden_size	32.00	128	32
lstm_layers	1.00	3	2
max_prediction_length	19.00	8	3
randomize_length	True	False	True
reduce_on_plateau_patience	9.00	7	5
share_single_variable_networks	False	True	True
target_normalizer	-	-	encnorm
batch_size	-	-	128
learning_rate	0.00	0.01	0.00
scaler	encnorm	standard	standard

3) LONG SHORT MEMORY

Table 19 shows the individual tuning parameters for the hybrid LSTM model. *lstm_layers* represents the number of hidden layers included in each LSTM module in the model. The *lstm_dropout* parameter determines the dropout rate applied between hidden layers inside each LSTM module; it is only used if the number of hidden layers is greater than one. The *static_hidden_size_1* and *static_hidden_size_2* parameters control the sizes of the first and second linear layers in the static branch of the LSTM.

4) TEMPORAL FUSION TRANSFORMER

The search space for the TFT is presented in Table 21. Since the TFT was implemented using the `pytorch-forecasting` library, the data was converted to a PyTorch Forecasting time series dataset. While tuning the TFT model, the parameters for the time series dataset were tuned as well. Since the Pytorch Forecasting library has different namings and configurations than regular neural networks implemented in torch, the TFT did not share the same common parameters as the other neural networks.

The search space for the dataset configuration is presented in Table 20.

max_encoder_length and *max_prediction_length* define the maximum number of time steps included in the encoder and decoder, respectively. The length of the

TABLE 30. Full evaluation radio 1.

model	F1	Energy Saving (%)	# Underestimations
rf_c	0.0	12.84	3
rf_F1	0.0	12.87	3
rf_r	0.0	12.87	3
xgb_c	0.0	12.83	3
xgb_F1	0.0	12.87	3
xgb_r	0.0	12.85	3
knc_c	0.0	12.84	3
knr_F1	0.0	12.86	3
knr_r	0.0	12.87	3
svc_c	0.0	12.81	3
svr_F1	0.0	12.84	3
svr_r	0.0	12.84	3
ffnn_c	0.0	12.73	3
ffnn_F1	0.0	12.87	3
ffnn_r	0.0	12.87	3
lstm_c	0.0	12.87	3
lstm_F1	0.0	12.84	3
lstm_r	0.0	12.87	3
cnn_c	0.00	12.84	3
cnn_F1	0.0	12.84	3
cnn_r	0.0	12.87	3
tft_c	0.0	12.81	3
tft_F1	0.0	12.87	3
tft_r	0.0	12.87	3
arima_r	0.0	12.85	3
naive_mean_1	0.0	12.82	3
naive_mean_2	0.0	12.86	3
naive_mean_3	0.0	12.87	3
naive_mean_48	0.0	12.87	3

TABLE 31. Full evaluation radio 2.

model	F1	Energy Saving (%)	# Underestimations
rf_c	0.21	11.73	18
rf_F1	0.11	12.94	28
rf_r	0.11	12.96	28
xgb_c	0.21	11.47	16
xgb_F1	0.18	12.83	26
xgb_r	0.05	12.89	29
knc_c	0.05	12.91	29
knr_F1	0.08	12.72	28
knr_r	0.0	13.02	30
svc_c	0.18	11.63	19
svr_F1	0.15	12.87	27
svr_r	0.0	12.99	30
ffnn_c	**0.31**	12.03	15
ffnn_F1	0.05	12.86	29
ffnn_r	0.06	12.95	29
lstm_c	0.19	12.41	23
lstm_F1	0.18	12.84	26
lstm_r	0.06	12.98	29
cnn_c	0.14	12.61	26
cnn_F1	0.05	12.90	29
cnn_r	0.11	12.63	27
tft_c	0.16	9.81	**11**
tft_F1	0.0	12.85	30
tft_r	0.0	13.02	30
arima_r	0.0	12.80	30
naive_mean_1	0.1	12.57	27
naive_mean_2	0.07	12.61	28
naive_mean_3	0.11	12.67	27
naive_mean_48	0.0	**13.04**	30

encoder is the TFT's equivalent of the *lag_value*. *encoder_random_ratio* controls the fraction of *max_encoder_length* to select as the minimum encoder length. The Boolean flag *add_relative_time_idx* controls whether to include relative time indices for sequences. These indices range from encoder length to prediction length. *add_target_scales* decides if target scales for static real features should be added, and *add_encoder_length* if encoder length should be added to static real variables. *randomize_length* enables random variation in encoder lengths per sample. The *target_normalizer* and *scaler* parameters define the normalization strategies applied to the target and input features, respectively.

The parameter setup for the TFT model is shown in Table 21. The *hidden_size* represents the number of neurons in each hidden layer. The *lstm_layers* parameter determines the number of LSTM layers to include in the model. The dropout rate is controlled by the *dropout* parameter. The *attention_head_size* represents the number of attention heads to include, and *hidden_continuous_size* specifies the hidden size for the layers processing continuous variables. *reduce_on_plateau_patience* controls the patience before reducing the learning rate by a factor of 10. The hyperparameter *share_single_variable_networks* determines whether the encoder and decoder share the single-variable networks. *causal_attention* controls whether the decoder is allowed to attend to future time steps. The *learning_rate* and *batch_size* parameters were described in Section C.

G. OPTIMAL HYPERPARAMETER SETTINGS

This appendix shows the optimal hyperparameters for each model.

H. ARCHITECTURE-SPECIFIC COMPLEXITY EXPRESSIONS

This appendix provides the architecture-specific expressions used to motivate the dominant complexity terms in Table 4. These expressions are approximate and omit lower-order operations such as activation functions, dropout, normalization, pooling, concatenation, and element-wise additions.

For the HybridLSTM, let τ denote the lag length, F_{seq} the sequential feature size per time step, H the LSTM hidden size, K the number of layers in the first LSTM block, S the

TABLE 32. Full evaluation radio 3.

model	F1	Energy Saving (%)	# Underestimations
rf_c	0.0	12.70	2
rf_F1	0.0	12.75	2
rf_r	0.0	12.75	2
xgb_c	0.0	12.70	2
xgb_F1	0.0	12.75	2
xgb_r	0.0	12.75	2
knc_c	0.0	12.75	2
knr_F1	0.0	12.73	2
knr_r	0.0	12.75	2
svc_c	0.0	12.67	2
svr_F1	0.0	12.75	2
svr_r	0.0	12.75	2
ffnn_c	0.0	12.70	2
ffnn_F1	0.0	12.75	2
ffnn_r	0.0	12.75	2
lstm_c	0.0	12.75	2
lstm_F1	0.0	12.75	2
lstm_r	0.0	12.75	2
cnn_c	0.0	12.75	2
cnn_F1	0.0	12.75	2
cnn_r	0.0	12.75	2
tft_c	0.0	12.67	2
tft_F1	0.0	12.75	2
tft_r	0.0	12.75	2
arima_r	0.0	12.75	2
naive_mean_1	0.0	12.72	2
naive_mean_2	0.0	12.75	2
naive_mean_3	0.0	12.75	2
naive_mean_48	0.0	12.75	2

TABLE 33. Full evaluation radio 4.

model	F1	Energy Saving (%)	# Underestimations
rf_c	0.0	13.01	0
rf_F1	0.0	13.01	0
rf_r	0.0	13.01	0
xgb_c	0.0	13.01	0
xgb_F1	0.0	13.01	0
xgb_r	0.0	13.01	0
knc_c	0.0	13.01	0
knr_F1	0.0	12.99	0
knr_r	0.0	13.01	0
svc_c	0.0	13.01	0
svr_F1	0.0	13.01	0
svr_r	0.0	13.01	0
ffnn_c	0.0	13.01	0
ffnn_F1	0.0	13.01	0
ffnn_r	0.0	13.01	0
lstm_c	0.0	13.01	0
lstm_F1	0.0	13.01	0
lstm_r	0.0	13.01	0
cnn_c	0.0	13.01	0
cnn_F1	0.0	13.01	0
cnn_r	0.0	13.01	0
tft_c	0.0	12.92	0
tft_F1	0.0	13.01	0
tft_r	0.0	13.01	0
arima_r	0.0	13.01	0
naive_mean_1	0.0	13.01	0
naive_mean_2	0.0	13.01	0
naive_mean_3	0.0	13.01	0
naive_mean_48	0.0	13.01	0

number of static input features, S_1 and S_2 the hidden sizes in the static branch, and d_{out} the output dimension. Since an LSTM computes four gate transformations at each time step, the approximate prediction-time cost is proportional to

$$\begin{aligned}\text{Cost}_{\text{LSTM}} \propto\ & \tau[4(F_{\text{seq}}H + H^2) + 4(K-1)(2H^2) \\ & + F_{\text{seq}}H + 4(2H^2)] + SS_1 \\ & + S_1S_2 + (H + S_2)d_{\text{out}}. \end{aligned} \tag{18}$$

The dominant recurrent term is therefore proportional to

$$\text{Cost}_{\text{LSTM}} \propto \tau(F_{\text{seq}}H + KH^2). \tag{19}$$

For the HybridCNN, let τ_i denote the input sequence length of convolutional branch i, C_i the number of output channels, and k_i the kernel size. Let S denote the static input dimension, H_s the static hidden size, S_{out} the static branch output size, R the resized branch output dimension, H_c the concatenation hidden size, and d_{out} the output dimension. The approximate prediction-time cost is proportional to

$$\begin{aligned}\text{Cost}_{\text{CNN}} \propto\ & \tau_1C_1k_1 + \tau_2C_2k_2 + SH_s + H_sS_{\text{out}} \\ & + 3RH_c + H_cd_{\text{out}}. \end{aligned} \tag{20}$$

The dominant convolutional term is therefore proportional to

$$\text{Cost}_{\text{CNN}} \propto \tau_1C_1k_1 + \tau_2C_2k_2. \tag{21}$$

I. EVALUATION PER RADIO

This appendix specifies the evaluation of the individual Radios. First, Table 30 shows the evaluation for radio 1. It is highly imbalanced and only has three possible underestimations and thus three instances in the positive class. None of the machine learning or benchmark methods were able to accurately predict any of these instances. The benchmark models performed comparably to the machine learning models on this test set. No model demonstrated superior performance; accordingly, no values are highlighted in bold.

Table 31 presents the evaluation results for radio 2. This dataset contains more instances of the positive class compared to radio 1. In this case, the machine learning models outperform the benchmarks in terms of underestimations. The *Naive_mean_48* model achieves the highest energy savings, although this is accompanied by the maximum

TABLE 34. Full evaluation radio 5.

model	F1	Energy Saving (%)	# Underestimations
rf_c	**0.47**	12.12	19
rf_F1	0.21	12.37	29
rf_r	0.26	12.42	28
xgb_c	0.14	12.02	29
xgb_F1	0.24	12.33	28
xgb_r	0.35	12.42	26
knc_c	0.16	12.38	30
knr_F1	0.21	12.25	28
knr_r	0.0	12.42	33
svc_c	0.15	12.12	29
svr_F1	0.43	12.35	23
svr_r	0.06	12.40	32
ffnn_c	0.46	11.73	**14**
ffnn_F1	0.11	12.40	31
ffnn_r	0.0	12.42	33
lstm_c	0.31	12.32	26
lstm_F1	0.24	12.33	28
lstm_r	0.0	12.42	33
cnn_c	0.27	12.19	26
cnn_F1	0.16	12.38	30
cnn_r	0.0	12.42	33
tft_c	0.33	11.40	15
tft_F1	0.31	12.21	25
tft_r	0.0	12.40	33
arima_r	0.19	12.33	29
naive_mean_1	0.3	11.97	23
naive_mean_2	0.36	12.18	23
naive_mean_3	0.31	12.23	25
naive_mean_48	0.0	12.42	33

number of underestimations. The second-best performers in terms of energy savings are two machine learning models: the TFT regressor optimized for MSE and the KNR model optimized for MSE, both achieving savings of 13.02%. However, similar to `naive_mean_48`, neither of these models successfully predicted any positive class instances. The model achieving the highest energy savings while also identifying some positive instances is the HybridLSTM optimized for MSE, which attained 12.98% savings and correctly predicted 1 out of 30 positive instances. The second-best in this regard is the Random Forest optimized for MSE, which achieved 12.96% savings with 28 out of 30 underestimations.

In terms of underestimations, the TFT classifier achieved the best performance with only 11 underestimations, correctly predicting approximately 60% of the positive class instances. The FFNN classifier had 15 underestimations and achieved 12.03% energy savings, compared to 9.81% for the TFT classifier.

Radio 3, similar to radio 1, contains few instances of the positive class—-two in total. None of the ML models or benchmarks successfully predict any of these positive instances. Notably, 17 out of 24 ML models achieve the maximum possible energy saving of 12.75%, as do several of the benchmarks. The results are provided in Table 32.

Table 33 presents the results for radio 4, which contains zero instances of the positive class. Consequently, the voltage can always be set to the lower level. All benchmark models and ML models, except two configurations, correctly maintain the voltage at the lower level throughout. This is evidenced by their achieving the maximum possible energy savings for radio 4, as shown in Table 33.

Lastly, Table 34 presents the results for radio 5, which has a similar proportion of positive instances as radio 2 (see Table 31). Several machine learning models, including the RF and XGB optimized for MSE, achieve the maximum possible energy savings of 12.42% while also correctly predicting some of the higher-utilization instances. The FFNN classifier demonstrates the best performance in terms of underestimations, closely followed by the TFT classifier. However, the FFNN classifier outperforms the TFT classifier across all evaluation metrics.

REFERENCES

[1] *The Paris Agreement, United Nations Framework Convention on Climate Change*, Bonn, Germany, 2015.

[2] Copernicus: January 2025 *Was the Warmest on Record Globally, Despite an Emerging La Niña*, Copernicus Climate Change Service, Reading, U.K., Feb. 2025.

[3] L. Clarke et al., "Energy systems," in *Climate Change 2022: Mitigation of Climate Change. Contribution of Working Group III to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*, P. R. Shukla et al., Eds., Cambridge, U.K. Cambridge Univ. Press, 2022, ch. 6.

[4] C. Zhang, L. Zhao, Z. Yin, and Z. Zhang, "Causalformer: Causal discovery-based transformer for multivariate time series forecasting," in *Proc. 16th Int. Congr. Image Signal Process., Biomed. Eng. Informat.* (CISP-BMEI), Oct. 2023, pp. 1–10.

[5] M. Tzelepi et al., "Deep learning for energy time-series analysis and forecasting," *CoRR*, vol. abs/2306.09129, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2306.09129

[6] B. Yildirim and M. Taskiran, "Optuna based optimized transformer model approach in Bitcoin time series analysis," in *Proc. 26th Int. Conf. Digit. Signal Process. Appl. (DSPA)*, Mar. 2024, pp. 1–6.

[7] L. Aronsson and A. Bengtsson, "Machine learning applied to traffic forecasting," M.S. thesis, Dept. Comput. Sci. Eng., Chalmers Univ. Technol., Univ. Gothenburg, Gothenburg,Sweden, 2019.

[8] Y. Zhang and J. Yan, "Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting," in *Proc. 11th Int. Conf. Learn. Represent.*, 2023.

[9] M. Kirisci and O. Cagcag Yolcu, "A new CNN-based model for financial time series: TAIEX and FTSE stocks forecasting," *Neural Process. Lett.*, vol. 54, no. 4, pp. 3357–3374, Aug. 2022.

[10] L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on typical tabular data?" in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 507–520.

[11] H. Nilsson, R. Johansson, N. Åkerblom, and M. H. Chehreghani, "Tree ensembles for contextual bandits," *Trans. Mach. Learn. Res.*, Oct. 2024. [Online]. Available: https://openreview.net/forum?id=59DCkSGw8S

[12] J. Guo, P. Lin, L. Zhang, Y. Pan, and Z. Xiao, "Dynamic adaptive encoder–decoder deep learning networks for multivariate time series forecasting of building energy consumption," *Appl. Energy*, vol. 350, Nov. 2023, Art. no. 121803.

[13] B. Lim, S. Ö. Arık, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," *Int. J. Forecasting*, vol. 37, no. 4, pp. 1748–1764, Oct. 2021.

[14] G. Auer et al., "How much energy is needed to run a wireless network?" *IEEE Wireless Commun.*, vol. 18, no. 5, pp. 40–49, Oct. 2011.
[15] V. Kasuluru, L. Blanco, C. J. Vaca-Rubio, and E. Zeydan, "On the impact of PRB load uncertainty forecasting for sustainable open RAN," in *Proc. IEEE 35th Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Sep. 2024, pp. 1–7.
[16] N. Piovesan, D. López-Pérez, A. De Domenico, X. Geng, H. Bao, and M. Debbah, "Machine learning and analytical power consumption models for 5G base stations," *IEEE Commun. Mag.*, vol. 60, no. 10, pp. 56–62, Oct. 2022.
[17] T. Song, D. Lopez, M. Meo, N. Piovesan, and D. Renga, "High altitude platform stations: The new network energy efficiency enabler in the 6G era," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2024, pp. 1–6.
[18] S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *CoRR*, vol. abs/1705.07874, 2017. [Online]. Available: http://arxiv.org/abs/1705.07874
[19] R. Aravind, C. Bharatiraja, R. Verma, and L. Mihet-Popa, "Performance analysis and overcoming cross-regulation challenges of a multi-input multi-output DC–DC converter for electric vehicles," *IEEE Access*, vol. 12, pp. 172742–172760, 2024.
[20] H. Holtkamp, G. Auer, V. Giannini, and H. Haas, "A parameterized base station power model," *IEEE Commun. Lett.*, vol. 17, no. 11, pp. 2033–2035, Nov. 2013.
[21] A. Ghosh, J. Zhang, J. G. Andrews, and R. Muhamed, *Fundamentals LTE*. London, U.K.: Pearson Education, 2010.
[22] H. N. Vu and H.-Y. Kong, "Joint subcarrier matching and power allocation in OFDM two-way relay systems," *J. Commun. Netw.*, vol. 14, no. 3, pp. 257–266, Jun. 2012.
[23] M. Bohge, J. Gross, and A. Wolisz, "The potential of dynamic power and sub-carrier assignments in multi-user OFDM-FDMAa cells," in *Proc. GLOBECOM IEEE Global Telecommun. Conf.*, Nov. 2005, pp. 2932–2936.
[24] K.-H. Park, Y.-C. Ko, and M.-S. Alouini, "On the diversity enhancement and power balancing of per-subcarrier transmit antenna selection in OFDM systems," *IEEE Trans. Veh. Technol.*, vol. 60, no. 5, pp. 2405–2410, Jun. 2011.
[25] F. A. Hla Myo Tun, S. B. Aye Thandar Phyo, and T. C. Zaw Min Naing, "Equal power distribution and dynamic subcarrier assignment in OFDM using minimum channel gain flow with robust optimization uncertain demand," 2010, *arXiv:1002.4045*.
[26] W. H. Hayt, J. E. Kemmerly, and S. M. Durbin, *Engineering Circuit Analysis*, 8th ed., New York, NY, USA: McGraw-Hill, 2022.
[27] L. A. Jeni, J. F. Cohn, and F. De La Torre, "Facing imbalanced data-recommendations for the use of performance metrics," in *Proc. Humaine Assoc. Conf. Affect. Comput. Intell. Interact.*, Sep. 2013, pp. 245–251.
[28] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001.
[29] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.
[30] P. Cunningham and S. Delany, "k-nearest neighbour classifiers—A tutorial," *ACM Comput. Surv.*, vol. 54, no. 6, pp. 128:1–128:25, Jul. 2022, doi: 10.1145/3459665.
[31] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, pp. 273–297, Sep. 1995.
[32] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
[33] Y. LeCun et al., "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989.
[34] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*, 5th ed., Hoboken, NJ, USA: Wiley, 2015.
[35] A. M. Nagib, H. Abou-Zeid, H. S. Hassanein, A. Bin Sediq, and G. Boudreau, "Deep learning-based forecasting of cellular network utilization at millisecond resolutions," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2021, pp. 1–6.
[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[37] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.
[38] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2019, pp. 2623–2631.
[39] J. Joy and M. P. Selvan, "A comprehensive study on the performance of different multi-class classification algorithms and hyperparameter tuning techniques using optuna," in *Proc. Int. Conf. Comput., Commun., Secur. Intell. Syst. (IC3SIS)*, Jun. 2022, pp. 1–5.
[40] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.
[41] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
[42] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.
[43] V. Vimbi, N. Shaffi, and M. Mahmud, "Interpreting artificial intelligence models: A systematic review on the application of LIME and SHAP in Alzheimer's disease detection," *Brain Informat.*, vol. 11, no. 1, p. 10, Dec. 2024.
[44] D. Halliday, R. Resnick, and J. Walker, *Fundamentals of Physics*, 10th ed., Hoboken, NJ, USA: Wiley, 2013.
[45] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.
[46] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, H. Wallach et al., Eds., vol. 32. Red Hook, NY, USA: Curran Associates, 2019, pp. 1–12. [Online]. Available: https://proceedings.neurips.cc/paperfiles/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf