TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

SHORT-TERM ELECTRICAL LOAD FORECASTING FOR BANESHWOR FEEDER USING MACHINE AND DEEP LEARNING MODELS

Submitted by

SUJIT KOIRALA

(PUL075MSPSE016)

A THESIS REPORT

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE IN POWER SYSTEM ENGINEERING

DEPARTMENT OF ELECTRICAL ENGINEERING

PULCHOWK CAMPUS, LALITPUR, NEPAL

JANUARY, 2026

COPYRIGHT

Certificate of Approval

The undersigned certify that they have read and recommended to the Institute of Engineering for acceptance, a THESIS entitled **Short-Term Electrical Load Fore casting for Baneshwor Feeder Using Machine and Deep Learning Models** submitted by Sujit Koirala (PUL075MSPSE016) as a partial requirement for the Master of Science in Power System Engineering. After review, we recommend its acceptance by the Institute of Engineering.

Asst. Prof. Amrit Dhakal
MSc. in Power System Engineering

Pulchowk Campus, Lalitpur
(Supervisor)

Deependra Neupane
Deputy Director, Electricity Regulatory Commission
Sano Gaucharan, Kathmandu, Nepal
(Supervisor)

Asst. Prof. Dr. Kamal Chapagain
Department of Electrical and Electronics Engineering
Kathmandu University (External Examiner)

Asst. Prof. Dr. Bishal Silwal
Program Coordinator
MSc. in Power System Engineering
Pulchowk Campus, Lalitpur

Assoc. Prof. Jeetendra Chaudhary
Head of Department
Department of Electrical Engineering
Pulchowk Campus, Lalitpur

Date: January, 2026

ABSTRACT

Predicting electricity demand a few hours ahead matters a lot for running power grids smoothly and for trading energy in markets. Because load patterns shift with weather, daily habits, and seasons, older statistical tools often miss the complex, nonlinear behavior in the data. This thesis tackles hour-ahead load prediction for the Baneshwor Feeder by testing several machine learning and deep learning methods side by side. The raw data—hourly megawatt readings together with temperature, solar radiation, and humidity—went through careful cleaning: gaps were filled, outliers removed, time-based features extracted, and cyclical hour/month values encoded as sine-cosine pairs. Six ML models (Linear Regression, Ridge, SVR, Random Forest, Gradient Boosting, XGBoost) and two recurrent neural networks (LSTM, GRU) were trained, tuned, and compared on the same test set. Performance was measured with MAE, RMSE, MAPE, and R^2. The GRU model—using lag inputs at 1, 3, 6, 12, 24, and 48 hours—came out on top (RMSE 0.289, R^2 0.879, MAPE 7.36%). LSTM followed closely (RMSE 0.314, R^2 0.857, MAPE 7.61%), while tuned XGBoost led the ML pack (RMSE 0.384, R^2 0.831, MAPE 12.69%). These error levels sit comfortably within the range utilities need for intraday market adjustments, showing that well-configured recurrent networks and boosted-tree ensembles can support real-world feeder-level forecasting and market participation.

Keywords: *short-term load forecasting, machine learning, deep learning, GRU, LSTM, XGBoost, electricity markets, day-ahead market, intraday market, power distribution systems*

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor and faculty members of the Department of Electrical Engineering for their valuable guidance, continuous support, and encouragement throughout the course of this project. Their technical insights and constructive feedback were instrumental in shaping this work.

I am also thankful to the Nepal Electricity Authority and relevant data-providing institutions for making the load and meteorological data available for this study. Their cooperation greatly contributed to the successful completion of the analysis.

Special thanks go to my friends and colleagues for their support, discussions, and motivation during the project period.

Finally, I would like to express my heartfelt appreciation to my family for their constant encouragement and support throughout my academic journey.

Sujit Koirala (PUL075MSPSE016)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVIATIONS

ANN	:	Artificial Neural Network
ARIMA	:	Autoregressive Integrated Moving Average
BS	:	Bikram Sambat (Nepali Calendar)
CNN	:	Convolutional Neural Network
DAM	:	Day-Ahead Market
DHM	:	Department of Hydrology and Meteorology
DL	:	Deep Learning
DWT	:	Discrete Wavelet Transform
GRU	:	Gated Recurrent Unit
IDM	:	Intraday Market
LSTM	:	Long Short-Term Memory
MAE	:	Mean Absolute Error
MAPE	:	Mean Absolute Percentage Error
ML	:	Machine Learning
MW	:	Megawatt
NEA	:	Nepal Electricity Authority
PCA	:	Principal Component Analysis
RF	:	Random Forest
RMSE	:	Root Mean Squared Error
RNN	:	Recurrent Neural Network

R^2	:	Coefficient of Determination
RTM	:	Real-Time Market
STLF	:	Short-Term Load Forecasting
SVR	:	Support Vector Regression
TCN	:	Temporal Convolutional Network
XGBoost	:	Extreme Gradient Boosting

CHAPTER ONE: INTRODUCTION

1.1. Background

This work looks at predicting electricity load one hour ahead for a single distribution feeder, relying on data-driven machine learning and deep learning methods. Hourly megawatt readings from the Baneshwor Feeder were combined with weather measurements and time markers to build and test several forecasting models. The goal was to find out which approach gives the most accurate and stable predictions—information that matters both for day-to-day grid operation and for trading energy in competitive markets.

Electricity use fluctuates constantly. People wake up and turn on lights, businesses open, air conditioners kick in during hot afternoons, and usage drops late at night. A system operator who can see even a few hours ahead can schedule generators more wisely, avoid expensive last-minute purchases, handle peak periods with less stress, and keep the supply steady [1, 2]. In energy markets, these forecasts let traders bid smarter and keep supply and demand in balance minute by minute [3, 4].

Why Short-Term Load Forecasting Matters: Good hour-ahead and day-ahead predictions help in several ways. On the *operations* side, dispatchers can commit just the right mix of generators, plan maintenance windows without risking outages, and keep enough reserves on hand. On the *money* side, forecast mistakes cost real cash—either from buying pricey peaking power at the last minute or from paying penalties for producing too much. For *market players*, bids and offers hinge on expected demand; better forecasts mean better margins and fewer surprises in the day-ahead and intraday auctions [3]. And as *wind and solar* grow, knowing what the net load will be becomes essential for keeping the lights on and avoiding waste [5].

Short-Term Load Forecasting (STLF) usually covers anything from one hour to

roughly a day ahead. At these timescales, dispatchers decide which generators to run, how much reserve to hold, and how power flows through the network. For electricity markets, hour-ahead predictions feed into intraday trading sessions, letting participants tweak their positions as new information comes in, while very-short-term forecasts help real-time balancing and ancillary services [6, 7]. Hour-ahead forecasting is especially handy because it gives utilities time to react to the latest weather changes and demand shifts. Traditionally, utilities leaned on simpler statistical tools—linear regression, ARIMA, exponential smoothing, Holt-Winters [8, 9]. These work fine for well-behaved patterns, but they struggle when the load curve gets noisy, nonlinear, or tangled up with many variables at once.

Machine learning models such as Random Forest, Support Vector Regression, and XGBoost have proven useful for energy forecasting [1]. They handle nonlinear relationships better than classic statistics, which makes them a natural choice for electricity load. Deep learning approaches—particularly recurrent networks like LSTM and GRU—can pick up on time dependencies that simpler models miss [2].

The Baneshwor Feeder of the Nepal Electricity Authority serves a mixed group of consumers in the Baneshwor region of Kathmandu Valley, which is shown in single line diagram shown in Figure 1.1. Its load pattern reflects residential lifestyles, commercial activity, seasonal tourism impacts, and local weather changes. Daily and weekly cycles are clearly visible, but there are also irregularities that simple models fail to capture. As power consumption continues to grow and diversify, the ability to forecast the feeder's short-term load accurately has become even more important [9]. This creates a strong motivation to investigate how modern ML and DL models can improve forecasting performance for this specific feeder.

1.2. Problem Statement

At present, demand estimation for the Baneshwor distribution circuit depends largely on heuristic approximations or elementary statistical procedures. Such methods fail to account for the interplay among thermal conditions, atmospheric

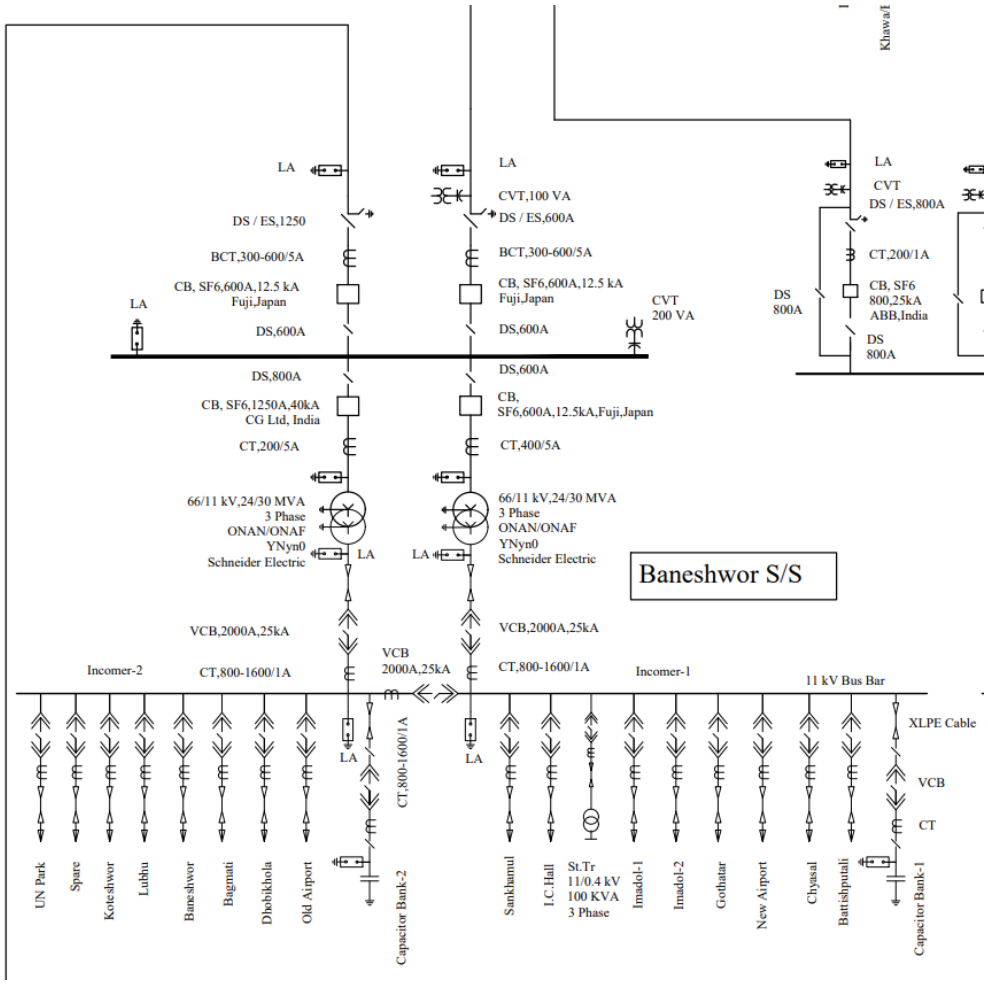

Figure 1.1 Substation & Transmission Line Network Baneshwor

moisture, precipitation, weekend effects, and exceptional events. Prediction discrepancies tend to accumulate during high-consumption intervals, abrupt meteorological shifts, and seasonal transitions. While a handful of investigations have addressed power forecasting within Kathmandu Valley, none have executed a rigorous side-by-side assessment of diverse ML and DL estimators with systematic hyperparameter optimization, nor situated the inquiry within an electricity-market context.

Unreliable forecasts impose dual penalties on infrastructure and finances. Schedulers may over-provision reserves (squandering capital) or under-provision (inviting supply deficits). Network losses can escalate, and in severe instances voltage instability or rotational outages may ensue. From a trading perspective, forecast

deviations translate into erroneous bids, onerous settlement penalties, and suboptimal procurement decisions [10]. As Nepal progresses toward a more competitive power landscape and regional interconnection, the stakes will only intensify. Hour-ahead predictions carry particular weight because grid participants must recalibrate commitments rapidly as circumstances evolve [11].

Despite the availability of archived consumption and climatological records, no systematic benchmarking of contemporary ML and DL approaches has yet been undertaken for this specific distribution line. Absent such a data-driven solution, operators forfeit access to algorithms capable of decoding the intricate dependencies embedded within the dataset.

This thesis addresses that void by assembling an end-to-end forecasting pipeline—training an assortment of ML and DL estimators, optimizing their configurations, quantifying their accuracy, and recommending the superior alternative for hour-ahead prediction on the Baneshwor line.

1.3. Objectives

The overarching objective is to design and validate ML and DL estimators for one-hour-ahead demand prediction on the Baneshwor distribution line, elevating forecast precision in alignment with intraday settlement schedules.

1. Compile and sanitize archived consumption records, meteorological observations, and temporal markers pertaining to the feeder.
2. Construct and benchmark multiple ML algorithms (SVR, Random Forest, XGBoost) and DL architectures (LSTM, GRU), gauging outcomes via RMSE, MAPE, MAE, and R^2.
3. Generate predictions applicable to both day-ahead scheduling and hour-ahead market clearing, supporting equilibrium maintenance and trading engagement.
4. Identify the top-performing estimator for operational deployment on this

distribution circuit, weighing accuracy against practical market constraints.

1.4. Scope and Limitations

This study covers only the Baneshwor Feeder under the Nepal Electricity Authority. The time focus is short-term: predicting load one hour ahead using historical hourly readings as the main input. That horizon matches what utilities need for intraday market adjustments—the most fast-paced part of energy trading [3]. With hour-ahead forecasts, dispatchers can tweak positions, fine-tune dispatch, and cut imbalance costs. The data set includes megawatt readings, temperature, humidity, rainfall, plus calendar flags for weekdays, weekends, and holidays.

On the algorithmic front, the investigation implements SVR, Random Forest, and XGBoost for classical learning, alongside LSTM and GRU for neural-network learning. Forecast quality is assessed via RMSE, MAPE, MAE, and R^2—the same metrics prevalent in power-market prediction literature, where precision governs a utility's ability to refine offers and sidestep deviation charges [10]. Note that predictive performance hinges substantially on the integrity and completeness of historical archives. Furthermore, this dissertation restricts attention to near-term horizons; intermediate and extended forecasting windows, renewable generation estimation, and wholesale price modeling lie beyond its purview.

Even with encouraging findings, multiple constraints surfaced during the work—chiefly concerning data characteristics, modeling premises, and analytical boundaries:

1. Data integrity reliance: Gaps, sensor malfunctions, or formatting irregularities within the archived consumption or meteorological records degrade algorithmic learning.

2. Difficulty with abrupt occurrences: Unplanned outages, public celebrations, extreme weather, and atypical usage surges resist prediction by any purely data-centric algorithm.

3. Computational burden for neural networks: LSTM and GRU demand greater training duration and hardware resources than classical estimators; outcomes may fluctuate with infrastructure configuration.

4. Omitted predictors: Macroeconomic activity, cultural events, industrial operating schedules, and spot-market prices remain absent from the dataset despite their potential explanatory value.

5. Feeder-specificity: The fitted algorithms are tailored to the Baneshwor distribution line; deployment on alternative circuits would necessitate fresh calibration.

6. Market-scope delimitation: The forecasts address hour-ahead consumption rather than offer formulation or price projection; integrated price-quantity optimization is deferred to subsequent research.

1.5. Report Organization

The dissertation is structured as follows. Chapter 1 introduces the concept of electrical demand, articulates why near-term forecasting matters for network operations and energy trading, and presents the research aims, boundaries, and constraints. Chapter 2 surveys antecedent scholarly work, identifies unresolved questions, and sketches the strategy employed herein; it additionally covers power-market fundamentals and the mathematical underpinnings of the ML/DL algorithms. Chapter 3 details the procedural pipeline—data acquisition, cleansing, model construction, training, validation, tuning, and assessment. Chapter 4 presents exploratory data insights, algorithmic configurations, parameter-search grids, and a comparative evaluation of outcomes pertinent to dispatch planning and hour-ahead trading. Chapter 5 concludes with summary findings, acknowledged constraints, market-related ramifications, and suggestions for follow-on inquiry.

CHAPTER TWO: LITERATURE REVIEW

This chapter examines prior scholarly contributions addressing short-term load prediction through ML and DL techniques. It analyzes the data sources, algorithmic approaches, and performance measurement strategies employed across power system forecasting studies, highlighting emerging patterns, strong points, and areas requiring further exploration. Recognizing these research voids clarifies the rationale for conducting this investigation. The review further establishes context for the algorithmic selections and procedural framework adopted herein.

2.1. Related Works

Extensive scholarly attention has been devoted to demand forecasting spanning short-term, medium-term, and long-term horizons. The majority of published work concentrates on near-term prediction windows.

[9] applied feed-forward neural networks to generate 24-hour demand predictions, drawing on dew point readings, dry bulb measurements, and atmospheric moisture content as model inputs. Their findings revealed that neural architectures effectively learn associations between meteorological conditions and consumption levels. In a parallel effort, [12] leveraged Meta's Prophet framework to produce near-term demand estimates, feeding in temporal markers, thermal readings, moisture levels, and meteorological projections. [13] investigated intermediate-horizon demand prediction via ensemble-based learning algorithms, benchmarking XGBoost and AdaBoost against conventional approaches such as SVR, tree-based classifiers, and Random Forest. Their outcomes underscored the strength of boosting strategies when dealing with intricate consumption behaviors. [1] evaluated five distinct learning algorithms and identified XGBoost as the top performer, relying on archived consumption records, climatological measurements, and public holiday flags. [14] examined three widely adopted ML techniques—SVM, Random

Forest, and LSTM—for demand estimation and introduced a combined forecasting strategy merging outputs from all three, establishing that blended approaches can surpass standalone model accuracy. Different from above studies, [15] performed a comparison between optimization methods (Particle Swarm Optimization, Dandelion Optimizer, Growth Optimizer) and machine learning models (SVR, ANN) for instantaneous peak electrical load forecasting. They found that ANN combined with Growth Optimizer outperformed other models and identified a strong positive correlation between GDP and peak load demand. [16] conducted a comprehensive evaluation of various machine learning algorithms for power load prediction, including Support Vector Machines, LSTM, ensemble classifiers, and Recurrent Neural Networks. They emphasized the importance of data preprocessing methods, feature selection strategies, and performance assessment metrics in achieving accurate forecasts, they demonstrated that ensemble methods and deep learning approaches consistently outperformed traditional statistical models.

Neural network architectures have gained substantial traction in consumption prediction tasks. [2] investigated deep learning frameworks alongside traditional ML approaches for estimating power demand within Kathmandu Valley, observing that LSTM networks delivered notably strong MAPE and RMSE scores. [8] carried out near-term demand estimation for the Gothatar distribution line using six predictor variables, concluding that recurrent architectures surpassed simpler baseline techniques like single exponential smoothing, double exponential smoothing, and the Holt-Winters procedure. [17] executed a thorough benchmarking exercise covering Random Forest, SVR, XGBoost, multi-layer perceptrons, LSTM, and one-dimensional convolutional networks, ultimately finding LSTM to yield the smallest prediction deviations across several accuracy measures. [18] assembled an extensive review of neural-network-driven near-term demand prediction spanning ten years of publications, noting that CNN-LSTM fusion designs have become prevalent owing to their capacity to jointly extract spatial and sequential features.

Combined architectural designs have become increasingly popular for sequential

data prediction. [19] put forward a blended deep learning system merging Gated Recurrent Units with Temporal Convolutional Networks and an attention layer for near-term consumption forecasting. The GRU component addressed extended temporal dependencies, whereas the TCN branch efficiently extracted recurring patterns. Their composite approach outperformed single-architecture baselines. [20] constructed a joint CNN-LSTM pipeline targeting individual dwelling demand prediction, with convolutional layers handling raw feature extraction and recurrent layers managing sequence modeling. This investigation confirmed the merit of uniting convolutional and recurrent building blocks when confronting the high variability present in residential consumption. Taking a different route, [21] introduced a parallel dual-channel configuration employing one-dimensional convolutions alongside Bidirectional LSTM for smart-grid demand estimation. Departing from conventional stacked CNN-LSTM layouts, their design processed spatial and temporal aspects through separate parallel pathways.

[22] model are also widely used in time-series forecasting task. [23] proposed a sparse transformer-based approach for electricity load forecasting that addressed the computational complexity limitations of standard transformer architectures, sparse attention mechanisms capture temporal dependencies more efficiently, achieving comparable accuracy to RNN-based state-of-the-art methods while being up to 5 times faster during inference. [24] developed a Time Augmented Transformer model for short-term electrical load forecasting, incorporating temporal features and self-attention mechanisms to capture complex dynamic non-linear sequence dependencies. Attention mechanism's capacity to capture complex dynamical patterns in multivariate data contributed to improved forecasting accuracy. [25] proposed a multivariate data slicing transformer neural network for load forecasting in power systems. The transformer model excelled in capturing spatiotemporal relationships by self-attention mechanisms. Their approach demonstrated superior performance in handling the intermittency and volatility characteristics, outperforming traditional statistical models and conventional machine learning methods.

Ensemble methods are also applied in electrical load forecasting task. [26] devel-

oped an enhanced stacked ensemble model combining Random Forest and XGBoost for renewable power and load forecasting, Random Forest first forecast the target variable, followed by XGBoost improving predictions through combination.

2.2. Electricity Markets and Load Forecasting

During the last several decades, power markets have transitioned from centralized monopoly structures toward competitive frameworks that unbundle generation, transmission, and distribution functions [4]. Within this market-oriented landscape, demand prediction has evolved from a purely operational exercise into a factor that shapes trading performance, running expenses, and financial outcomes for all participants [3].

2.2.1. Structure of Electricity Markets

Today's electricity markets run on several overlapping timeframes, each with its own purpose [7]. Knowing how these layers work shows why getting the right forecast at the right horizon really matters:

1. **Day-Ahead Market (DAM):** Market participants submit purchase and sale offers for every hour of the upcoming day, usually before a midday deadline. Clearing occurs when supply and demand curves intersect, establishing hourly prices and allocated volumes. Predictions spanning 24 to 48 hours ahead inform offer strategies, dispatch plans, and generator commitment decisions. Inaccurate day-ahead estimates lead either to costly last-minute procurement or to selling surplus energy at unfavorable rates [10].

2. **Intraday Market (IDM):** This trading venue allows participants to exchange energy for same-day delivery, frequently remaining open until an hour—or even fifteen minutes—before actual dispatch. When fresh meteorological or consumption updates arrive, traders revise their positions accordingly. Hour-ahead predictions prove essential here, enabling utilities to trim imbalance risk and respond to deviations the day-ahead estimate over-

looked. Reliable one-hour-out figures can determine intraday profitability and govern the magnitude of settlement charges a utility ultimately faces [11].

3. **Real-Time Balancing Market (RTM):** This platform operates around the clock to maintain instantaneous equilibrium between generation and consumption. System operators procure frequency-control services and spinning reserves, settling residual mismatches through this mechanism. Very-short-horizon predictions—spanning minutes to roughly an hour—help maintain stable operation and prevent costly emergency interventions. Because settlement prices in this market can surge dramatically or even turn negative, there exists a compelling economic driver for precise forecasting [27].

Figure 2.1 illustrates the relationship between forecasting horizons and electricity market operational timelines.

Market	Trading Horizon	Forecast Need	Key Use
Day-Ahead	D-1 (noon)	24–48 hours	Unit commitment, bidding
Intraday	Same day	1–12 hours	Position adjustment
Real-Time	Continuous	Minutes–1 hour	Balancing, reserves

Figure 2.1 Electricity Market Timeline and Forecasting Requirements

2.2.2. Role of Load Forecasting in Market Operations

Load forecasts serve several practical purposes in market operations [5, 10]:

Day-Ahead Bidding: Most trading happens the day before delivery. Utilities submit demand estimates by noon, and those numbers shape bids, generation schedules, and commitment plans. Underestimate load and you buy expensive power at the last minute; overestimate and you dump the extra at a loss. For big utilities running gigawatt portfolios, even small errors add up fast [10].

Intraday Adjustments: As delivery nears, fresher forecasts let traders tweak positions. Day-ahead guesses always have some error—weather changes, random

demand bumps—so hour-ahead predictions let utilities close the gap between what they contracted and what they actually need. Good hour-ahead numbers cut dependence on pricey real-time fixes.

Imbalance Cost Control: Any mismatch between forecasted and actual load ends up settled in the real-time market, often at unfavorable rates. During tight supply, imbalance prices can spike several times higher than day-ahead levels—or even go negative when there is too much power. Accurate hour-ahead forecasts catch the latest demand and weather shifts, giving utilities time to correct through intraday trades and dodge steep imbalance bills [11].

Grid Reliability: System operators rely on load forecasts to set reserves, manage congestion, and keep frequency stable. Better feeder-level predictions roll up into better system-wide visibility.

Economic Dispatch: Dispatch and scheduling algorithms eat load forecasts as their main input. Tighter forecasts mean leaner reserves, lower fuel bills, and better use of generation assets.

2.2.3. Relevance to Nepal's Power Sector

Nepal continues to operate under a vertically integrated utility structure governed by the Nepal Electricity Authority (NEA), yet transformation looms on the horizon. Cross-border power exchanges with India, expanding private-sector generation, and evolving regulatory frameworks signal a gradual shift toward market-like mechanisms. Establishing robust feeder-level prediction capabilities now is prudent for:

- Optimizing import scheduling and bilateral purchase agreements
- Preparing institutional readiness for potential wholesale market introduction
- Enabling demand-response program execution
- Facilitating smooth integration of distributed photovoltaic and storage assets

- Reducing technical and commercial losses via intelligent operations

The hour-ahead and day-ahead predictions generated within this dissertation align with such trading timescales, ensuring applicability to current grid management while anticipating future market participation requirements.

2.3. Theoretical Background of Forecasting Models

This section explains the math and inner workings of the models used in this study. Understanding how each algorithm learns patterns helps make sense of the forecasting results later on.

2.3.1. Machine Learning Models

2.3.1.1. Linear Regression

Linear Regression serves as the most elementary benchmark—it presumes a proportional connection linking predictors to the response variable. Although electricity demand rarely follows a purely linear pattern, this estimator supplies a convenient reference for gauging more sophisticated algorithms.

Mathematical Formulation:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n \tag{2.1}$$

where β_0 denotes the intercept (bias component), β_i represent the coefficients (weights) determined through ordinary least squares optimization, and x_i correspond to the predictor variables.

2.3.1.2. Ridge Regression

Ridge Regression [28] incorporates L2 penalty terms into the linear formulation, dampening overfitting tendencies and producing steadier coefficient values. This variant offers greater resilience than ordinary linear regression when the feature

set contains numerous intercorrelated variables—a condition present in the current dataset.

Mathematical Formulation:

$$\min_{\beta} \left(\|y - X\beta\|^2 + \alpha \|\beta\|^2 \right) \tag{2.2}$$

where α controls the strength of L2 regularization, $\|y - X\beta\|^2$ is the residual sum of squares, and $\|\beta\|^2$ is the L2 norm of the coefficient vector.

2.3.1.3. Support Vector Regression (SVR)

SVR [29] captures nonlinear associations by projecting input variables into an elevated feature space via kernel transformations. This technique performs effectively on intricate regression tasks when the sample count remains moderate.

Mathematical Formulation:

SVR finds a function $f(x)$ by solving the following optimization problem:

$$\min_{w,b,\xi,\xi^*} \left(\frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} (\xi_i + \xi_i^*) \right) \tag{2.3}$$

subject to the constraints:

$$y_i - (w \cdot x_i + b) \leq \varepsilon + \xi_i$$
$$(w \cdot x_i + b) - y_i \leq \varepsilon + \xi_i^*$$
$$\xi_i, \xi_i^* \geq 0$$

where w is the weight vector, b is the bias term, C is the regularization parameter controlling the trade-off between flatness and tolerance of deviations, ε defines the epsilon-insensitive tube, and ξ_i and ξ_i^* are slack variables for points outside the tube.

2.3.1.4. Random Forest Regressor

Random Forest [30] constitutes an ensemble strategy assembling numerous decision trees. Every tree draws from a randomly sampled portion of predictors and observations via bootstrap aggregation. The resulting model exhibits resilience, consistency, and competent handling of nonlinear structures.

Mathematical Formulation:

The Random Forest prediction is the average of all individual tree predictions:

$$\hat{y} = \frac{1}{T} \sum_{t=1}^{T} f_t(x) \tag{2.4}$$

where T is the total number of trees in the forest and $f_t(x)$ is the prediction of the t-th decision tree.

2.3.1.5. Gradient Boosting Regressor

Gradient Boosting [31] constructs trees in succession, where every subsequent tree targets the residual discrepancies left by its predecessors. This approach proves especially potent when applied to organized tabular datasets such as consumption records.

Mathematical Formulation:

At each boosting iteration m, the model is updated as:

$$F_m(x) = F_{m-1}(x) + \nu \cdot h_m(x) \tag{2.5}$$

where $F_{m-1}(x)$ is the ensemble prediction from the previous iteration, $h_m(x)$ is the new tree fitted to the negative gradient (pseudo-residuals), and ν is the learning rate (shrinkage factor) that controls the contribution of each tree.

2.3.1.6. XGBoost Regressor

XGBoost (Extreme Gradient Boosting) [32] represents a refined, regularized variant of gradient boosting engineered for computational efficiency, scalability, and elevated predictive accuracy. Within this investigation, it ranked among the top-performing ML algorithms.

Mathematical Formulation:

XGBoost minimizes the following regularized objective function:

$$\mathcal{L} = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{2.6}$$

where $l(y_i, \hat{y}_i)$ is the loss function measuring the difference between actual and predicted values, and $\Omega(f_k)$ is the regularization term for the k-th tree, defined as:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{2.7}$$

where T is the number of leaves in the tree, w_j is the weight (score) of the j-th leaf, γ controls the minimum loss reduction required to make a split, and λ is the L2 regularization term on leaf weights.

2.3.2. Deep Learning Models

To model the nonlinear structures and temporal correlations embedded in feeder demand data, two recurrent neural network variants were implemented and evaluated: LSTM and GRU. Both architectures received sliding-window sequences of consecutive hourly observations, enabling them to learn patterns across short and extended horizons. Optimization relied on the Adam algorithm [33], supplemented by early stopping to prevent excessive fitting to training samples.

2.3.2.1. Long Short-Term Memory (LSTM)

LSTM networks [34] maintain a persistent internal state across extended sequences through gating mechanisms that govern retention and discarding of information. This property renders them well suited to demand prediction, where current consumption levels hinge on events occurring hours or days earlier. In contrast to vanilla RNNs, LSTM cells sidestep the vanishing-gradient phenomenon.

Mathematical Formulation:

At each time step t, the LSTM computes the following:

Forget gate (determines what information to discard from cell state):

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.8}$$

Input gate (determines what new information to store):

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.9}$$

Candidate cell state (creates new candidate values):

$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{2.10}$$

Cell state update (combines old and new information):

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{2.11}$$

Output gate (determines what to output):

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.12}$$

Hidden state (final output at time step t):

$$h_t = o_t \odot \tanh(c_t) \tag{2.13}$$

where σ is the sigmoid activation function, tanh is the hyperbolic tangent activation function, $\odot$ denotes element-wise (Hadamard) product, $[h_{t-1}, x_t]$ represents concatenation of the previous hidden state and current input, W_f, W_i, W_c, and W_o are weight matrices for each gate, b_f, b_i, b_c, and b_o are bias vectors for each gate, c_t is the cell state that carries long-term memory, and h_t is the hidden state output.

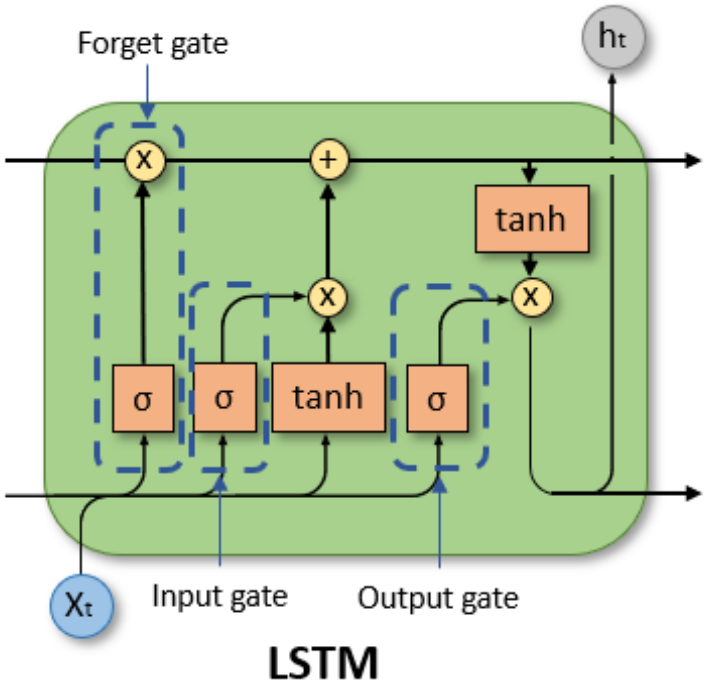

Figure 2.2 Architecture LSTM

2.3.2.2. Gated Recurrent Unit (GRU)

GRU [35] offers a more compact alternative to LSTM. It fuses the forget and input gating functions into a single update gate while unifying the cell state and hidden state into one vector. This streamlined design yields a reduced parameter count, quicker convergence, and frequently comparable predictive quality.

Mathematical Formulation:

At each time step t, the GRU computes the following:

Update gate (controls how much past information to keep):

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \tag{2.14}$$

Reset gate (determines how much past information to forget):

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \tag{2.15}$$

Candidate hidden state (computes new candidate activation):

$$\tilde{h}_t = \tanh(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h) \tag{2.16}$$

Final hidden state (interpolates between previous and candidate state):

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{2.17}$$

where σ is the sigmoid activation function, tanh is the hyperbolic tangent activation function, $\odot$ denotes element-wise (Hadamard) product, $[h_{t-1}, x_t]$ represents concatenation of the previous hidden state and current input, W_z, W_r, and W_h are weight matrices for the update gate, reset gate, and candidate state, b_z, b_r, and b_h are bias vectors, z_t controls the balance between old and new information, and r_t controls how much of the previous state influences the candidate.

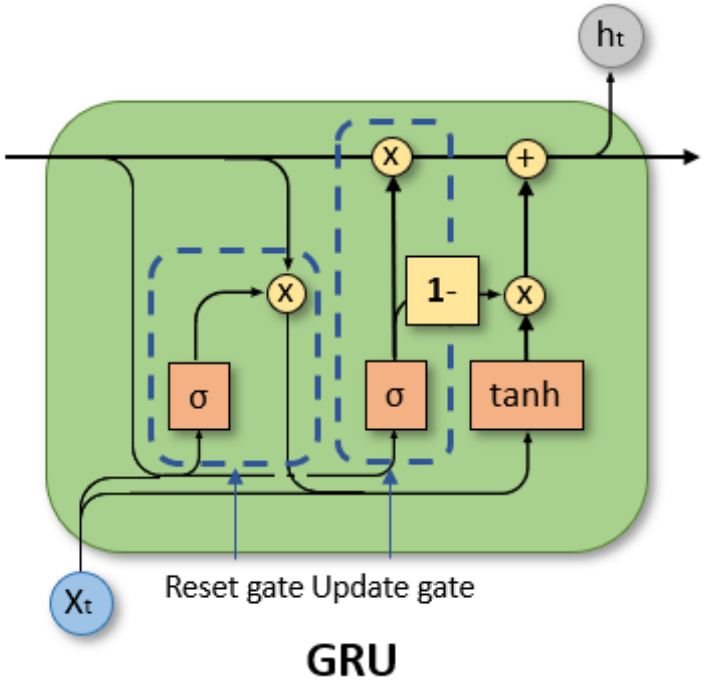

Figure 2.3 Architecture GRU

CHAPTER THREE: RESEARCH METHODOLOGY

This chapter details the procedural steps undertaken—spanning data collection, pre-processing, model construction, and accuracy assessment. The overall pipeline targets hour-ahead prediction, aligning with the timeframes utilities require for intraday market corrections. A schematic overview condenses the workflow, while subsequent sections elaborate on each stage.

3.1. Overall Workflow

The processing chain advances through sequential stages designed to yield hour-ahead demand estimates. Unprocessed hourly megawatt records from the Baneshwor Feeder are merged with meteorological measurements (thermal readings, moisture content, precipitation) and calendar indicators (weekday versus weekend status, public holidays). This hourly resolution corresponds to the settlement intervals characteristic of intraday trading platforms [3].

Initially, unprocessed entries undergo integrity verification—interpolating missing values, identifying anomalous readings, standardizing time references, and deriving additional variables such as sinusoidal hour and month representations. Subsequently, an exploratory investigation probes diurnal, weekly, and seasonal rhythms alongside weather-consumption linkages to select informative predictors.

Following this, ML algorithms receive a two-dimensional feature array while recurrent DL architectures ingest windowed sequences. Hyperparameter optimization proceeds via GridSearchCV for classical ML and systematic experimentation for LSTM/GRU. Ultimately, all models undergo benchmarking with RMSE, MAE, MAPE, and R^2—widely accepted indicators in energy-market prediction studies [5]—leading to a recommendation for operational deployment. Figure 3.1 presents the complete workflow.

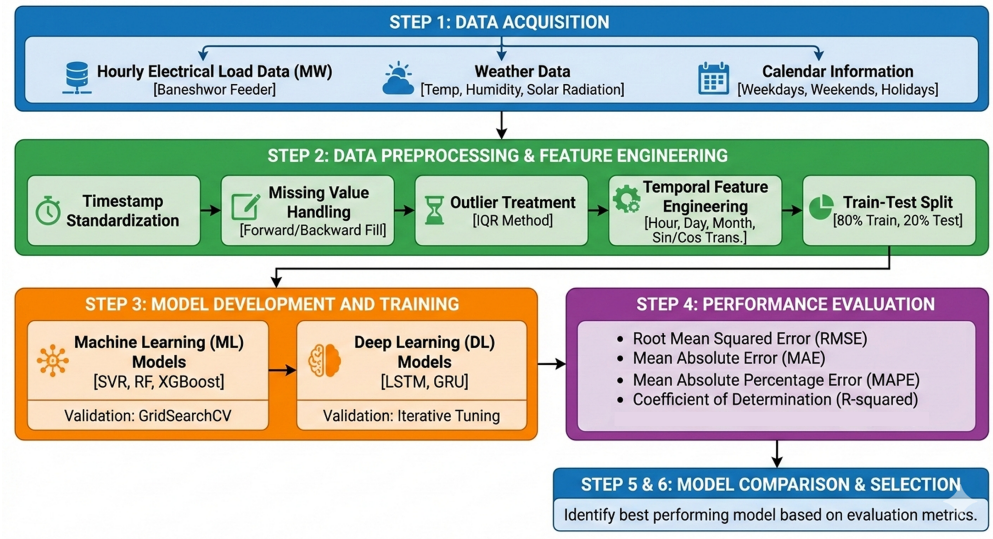

Figure 3.1 Methodology Block Diagram

3.2. Data Acquisition

Data assembly constituted the opening phase. Hourly consumption logs originated from archived station records at Baneshwor. Meteorological variables—ambient temperature, relative humidity, and incoming solar irradiance—were obtained through Nepal's Department of Hydrology and Meteorology. Temporal markers distinguishing workdays from weekends derived from official governmental calendars.

Two principal information streams supply the predictive models:

1. Hourly megawatt observations from the Baneshwor distribution line
2. Hourly climatological measurements (thermal, moisture, radiative)

Both datasets arrived as heterogeneous spreadsheet collections—varying layouts, irregular timestamps, data voids, and multiple tabs per reporting period—necessitating a bespoke multi-phase cleansing routine. Particulars for each source appear below.

3.2.1. Data Sources

a) **Electrical Load Data:** Consumption records originated from the Baneshwor Substation, a facility managed by the Nepal Electricity Authority (NEA). Hourly demand readings were extracted from archived operations logbooks kept at the station spanning fiscal years 2079 through 2082 BS. These logs supplied raw megawatt figures for the Baneshwor distribution circuit together with corresponding time stamps. Because the information existed as hand-entered spreadsheet files scattered across multiple workbooks, extensive preparatory work—header normalization, time-format unification, and anomaly screening—proved essential prior to analytical use. This station-level dataset underpins the forecasting inquiry, capturing authentic multi-year operational behavior of the distribution line.

b) **Weather Data:** Atmospheric observations came from the Department of Hydrology and Meteorology (DHM), Nepal's authoritative body for climatological monitoring. The compilation encompassed hourly ambient temperature, moisture percentage, and surface solar flux for the identical observation window. Such variables proved indispensable for representing environmental drivers of electricity usage patterns. The DHM records necessitated temporal realignment, gap filling through interpolation, and attenuation of spurious extremes to achieve harmonization with consumption data. Following cleansing and synchronization, these meteorological fields served as valuable exogenous predictors across both classical ML and neural-network pipelines.

Because the raw files came in varying formats—different months, unpredictable sheet names, Bikram Sambat (BS) dates, mixed day formats, half-hour readings, inconsistent header rows, and multiple sheets per month—a custom data acquisition pipeline was required.

3.2.2. Load Data Acquisition and Structuring Process

The original Excel files provided by the Nepal Electricity Authority (NEA) were highly heterogeneous in structure. Each month consisted of multiple workbooks, with each workbook containing several sheets. In many cases, sheets included mixed headers, irrelevant rows, inconsistent timestamp formats, and non-uniform naming conventions. To address these issues and convert the raw data into a single unified structure suitable for analysis, a multi-stage data processing pipeline was implemented.

In the first stage, a month-wise sheet extraction process was carried out. The script automatically scanned all monthly datasets and identified Excel sheets whose names contained "11KV" or its variations. The extracted sheets were then aligned with their corresponding time periods to ensure correct temporal ordering. Only rows with timestamps recorded at exact hourly intervals (HH:00) were retained, while half-hour readings such as 7:30 were intentionally excluded to maintain uniform hourly resolution. The valid hourly records from each sheet were compiled to produce clean month-wise datasets. At this stage, although the data were organized chronologically, the timestamps were still recorded in the Nepali calendar (BS) and exhibited format inconsistencies.

The second stage focused on date conversion, hour normalization, and daily data structuring. The BS date embedded within each sheet name was extracted and converted into the Gregorian (AD) calendar using Nepali date conversion libraries. Each sheet was read without assuming a fixed header position, allowing the script to dynamically identify the Time column and the corresponding POWER (MW) values. Every day was standardized to contain exactly twenty-four hourly records by indexing hours from 1 to 24, and any missing hours were filled using linear interpolation. Clean and consistent timestamps were then generated in the standard hourly format, ensuring temporal continuity across the dataset. This process resulted in one clean and complete daily record for each calendar day.

In the final stage, all structured monthly and yearly datasets were merged into

a single consolidated file. The script systematically extracted the valid Time and POWER (MW) columns, removed any remaining header fragments, and concatenated the data in chronological order. This process produced a fully unified dataset containing continuous hourly POWER (MW) measurements for the entire study period, which served as the foundation for subsequent data analysis and load forecasting model development.

3.2.3. Weather Data Acquisition and Structuring

Weather data was also provided in raw format with mixed timestamps. Two scripts were developed to clean and align it with the load data.

a) **Extracting and Cleaning Raw Weather File:** The script located the correct columns for time, temperature, humidity, and solar radiation, then removed any unusable rows. All timestamps were parsed into a consistent datetime format, after which the weather data was filtered to match the exact date range of the load dataset. The timestamps were then formatted as YYYY-MM-DD HH:MM. This stage produced a clean hourly weather dataset.

b) **Structuring Weather Timestamp Alignment:** Timestamps were shifted so that values such as "HH:45" were aligned to the next hour at "HH+1:00," and all "24:00" rollover cases were handled correctly. Missing or zero weather values were replaced using nearest-neighbor averages, while NaN solar radiation entries were set to zero. These steps produced the final clean weather file and ensured that all weather variables followed the exact hourly structure required for forecasting.

3.2.4. Final Merging of Load and Weather Data

In the final stage, load and weather datasets were merged into a single unified file. All load timestamps were carefully parsed, including proper handling of the special `24:00` time format, while weather timestamps were standardized to a consistent

format. Weather observations were then precisely aligned with their corresponding load timestamps, and any missing values in weather variables were filled using linear interpolation. The resulting dataset contained synchronized records of time, electrical load (MW), air temperature, global solar radiation, and relative humidity, and served as the primary input for all machine learning and deep learning models used in this thesis.

3.3. Data Preprocessing

Once load and weather data were merged into one hourly file, more cleaning was needed before modeling. Timestamps came in mixed formats—some even listed "24:00"—so a custom routine shifted those to 00:00 of the next day and standardized everything. Missing MW values (from incomplete logs or bad readings) were filled with forward-fill then backward-fill to keep the time series smooth. Weather gaps got linear interpolation; solar radiation was set to zero at night, and wild humidity or temperature spikes were smoothed using nearby values. The result was a clean, gap-free, time-aligned dataset.

Next came feature engineering to capture daily, weekly, and seasonal rhythms. From each timestamp, the pipeline extracted hour, day, month, day-of-week, week-of-year, and a weekend flag. Because time is cyclical—hour 23 wraps to hour 0—sine and cosine transforms were applied to hour, month, and day-of-week. These encodings let the models learn smooth periodic patterns instead of seeing artificial jumps at midnight or January 1. The final feature set combined weather variables with these temporal components.

A correlation check (Figure 3.2) showed which features matter most. Hour of day has the strongest link to load; solar radiation shows a moderate positive tie; temperature and humidity contribute weaker but still useful signals; calendar variables add subtle seasonal cues. All engineered features were kept based on this analysis and domain knowledge. After preprocessing, the dataset had no missing values, no weird timestamps, and no half-hour entries—ready for modeling.

Figure 3.2 Feature Correlation Matrix

3.4. Data Input Structure

This section describes the structure and format of the input data used for training the machine learning and deep learning models. Understanding the data input structure is essential for reproducibility and for applying similar methodologies to other forecasting problems.

3.4.1. Raw Data Format

The final merged dataset consists of hourly records with the following structure:

Table 3.1 Raw Input Data Format

Column	**Data Type**	**Unit**	**Description**
Time	Datetime	YYYY-MM-DD HH:MM	Hourly timestamp
MW	Float	Megawatt (MW)	Electrical load (target variable)
Air Temperature	Float	°C	Ambient temperature
Global Solar Radiation	Float	W/m^2	Solar irradiance
Relative Humidity	Float	%	Atmospheric humidity

Table 3.2 shows a sample of the cleaned dataset used in this study:

Table 3.2 Sample Data Records from Cleaned Dataset

Time	**MW**	**Air Temp (°C)**	**Solar Rad (W/m^2)**	**RH (%)**
2022-10-19 01:00	0.8	14.5	0.0	88.8
2022-10-19 02:00	0.8	14.4	0.0	87.9
2022-10-19 06:00	1.2	12.1	0.0	100.0
2022-10-19 12:00	1.8	22.0	822.0	46.5
2022-10-19 19:00	2.8	17.9	0.0	76.7

3.4.2. Feature Engineering and Input Vector

After preprocessing and feature engineering, each input sample contains the following features that are fed into the model nodes:

Table 3.3 Engineered Feature Set for Model Input

No.	Feature	Type	Description
1	Hour	Integer (0–23)	Hour of day
2	Day	Integer (1–31)	Day of month
3	Month	Integer (1–12)	Month of year
4	Day_of_Week	Integer (0–6)	Day of week (Mon=0)
5	Week_of_Year	Integer (1–52)	Week number
6	Is_Weekend	Binary (0/1)	Weekend indicator
7	Hour_Sin	Float (-1 to 1)	$\sin(2\pi \cdot \text{Hour}/24)$
8	Hour_Cos	Float (-1 to 1)	$\cos(2\pi \cdot \text{Hour}/24)$
9	Month_Sin	Float (-1 to 1)	$\sin(2\pi \cdot \text{Month}/12)$
10	Month_Cos	Float (-1 to 1)	$\cos(2\pi \cdot \text{Month}/12)$
11	DoW_Sin	Float (-1 to 1)	$\sin(2\pi \cdot \text{DoW}/7)$
12	DoW_Cos	Float (-1 to 1)	$\cos(2\pi \cdot \text{DoW}/7)$
13	Air_Temperature	Float	Scaled temperature
14	Global_Solar_Radiation	Float	Scaled solar radiation
15	Relative_Humidity	Float	Scaled humidity

3.4.3. Input Structure for Machine Learning Models

For machine learning models (Linear Regression, Ridge, SVR, Random Forest, Gradient Boosting, XGBoost), the input is structured as a 2D feature matrix:

$$X_{ML} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_n^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(m)} & x_2^{(m)} & \cdots & x_n^{(m)} \end{bmatrix} \tag{3.1}$$

where m is the number of samples and n is the number of features. The target vector is:

$$y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \tag{3.2}$$

where $y^{(i)}$ represents the load value (MW) at the next hour that the model predicts.

3.4.4. Input Structure for Deep Learning Models (LSTM/GRU)

For recurrent neural networks (LSTM and GRU), the input is structured as a 3D tensor to capture temporal sequences:

$$X_{DL} \in \mathbb{R}^{m \times T \times n} \tag{3.3}$$

where m is the number of samples, T is the sequence length (lookback window, typically 24 hours), and n is the number of features per timestep.

Figure 3.3 illustrates how the sliding window approach creates input sequences for the LSTM/GRU models:

	Input Sequence (T=24 hours)				**Target**
Sample	$t-24$	$t-23$	$\cdots$	$t-1$	t
1	$\mathbf{x}_1$	$\mathbf{x}_2$	$\cdots$	$\mathbf{x}_{24}$	y_{25}
2	$\mathbf{x}_2$	$\mathbf{x}_3$	$\cdots$	$\mathbf{x}_{25}$	y_{26}
3	$\mathbf{x}_3$	$\mathbf{x}_4$	$\cdots$	$\mathbf{x}_{26}$	y_{27}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$

Figure 3.3 Sliding Window Sequence Construction for LSTM/GRU Input

Each $\mathbf{x}_t$ represents the feature vector at timestep t:

$$\mathbf{x}_t = [\text{Hour}_t, \text{Hour\_Sin}_t, \text{Hour\_Cos}_t, \ldots, \text{Lag}_1, \text{Lag}_3, \ldots, \text{Lag}_{48}] \tag{3.4}$$

3.4.5. Lag Feature Configuration for Deep Learning

To enhance temporal pattern recognition, lag features are added to the input vector. Three configurations were tested:

Table 3.4 Lag Feature Configurations for Deep Learning Models

Configuration	**Lag Hours**	**Purpose**
Short	[1, 3, 6]	Recent hour patterns
Medium	[1, 3, 6, 12, 24]	Daily patterns
Long (Best)	[1, 3, 6, 12, 24, 48]	Multi-day dependencies

For example, with the long lag configuration, the lag features for predicting load at time t include:

- Lag_1: Load at $t-1$ (1 hour ago)
- Lag_3: Load at $t-3$ (3 hours ago)
- Lag_6: Load at $t-6$ (6 hours ago)
- Lag_{12}: Load at $t-12$ (12 hours ago)
- Lag_{24}: Load at $t-24$ (same hour yesterday)
- Lag_{48}: Load at $t-48$ (same hour two days ago)

3.4.6. Complete Input Pipeline Summary

Figure 3.4 summarizes the complete data input pipeline from raw data to model nodes:

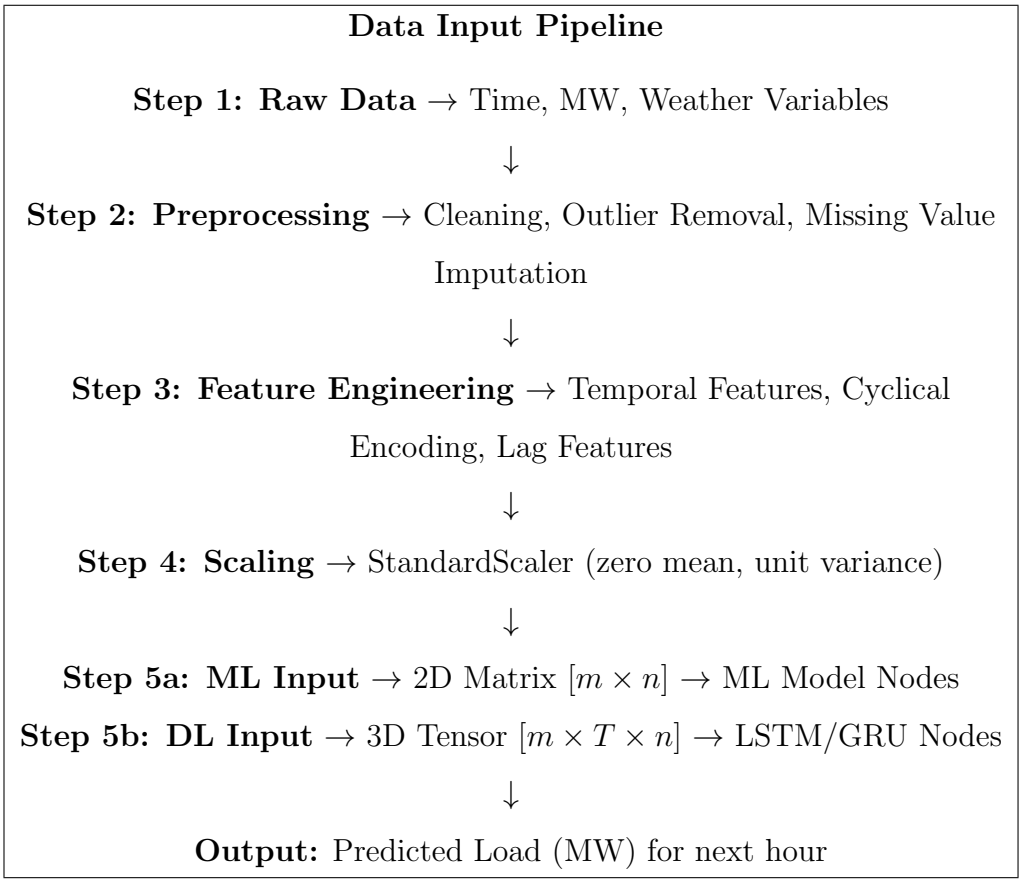

Figure 3.4 Complete Data Input Pipeline from Raw Data to Model Nodes

This structured input representation ensures that both machine learning and deep learning models receive appropriately formatted data that captures temporal patterns, weather influences, and historical load dependencies necessary for accurate hour-ahead load forecasting.

3.5. Model Development

Drawing on the conceptual groundwork laid out in Chapter 2, this investigation deployed an array of predictive algorithms for near-term demand estimation on the Baneshwor distribution line. On the classical ML front, six estimators were constructed: Linear Regression, Ridge Regression, Support Vector Regression (SVR), Random Forest Regressor, Gradient Boosting Regressor, and XGBoost Regressor. On the neural-network front, two sequential architectures were realized: Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU).

Every algorithm operated on the identical cleansed and augmented feature set detailed in preceding sections, guaranteeing equitable benchmarking. The classical pipeline encompassed z-score normalization of continuous variables, an 80–20 training-testing partition, and hyperparameter selection through GridSearchCV, whereas the recurrent networks ingested rolling windows of consecutive hourly snapshots and trained via the Adam optimizer with early termination to curb over-learning. Evaluation relied on RMSE, MAE, MAPE, and R^2.

3.6. Model Training and Validation

This phase addresses the procedures through which both classical ML and neural-network DL estimators were fitted, refined, validated, and readied for comparative assessment. Given the divergent requirements of ML versus DL pipelines, the training description is organized into two distinct subsections.

3.6.1. Training of Machine Learning Models

The classical learning algorithms fitted during this investigation comprise:

1. Support Vector Regression (SVR)
2. Random Forest Regressor (RF)
3. Gradient Boosting Regressor (GBR)
4. Extreme Gradient Boosting (XGBoost)
5. Linear Regression and Ridge Regression (reference baselines)

Every estimator operated on the identical final curated dataset described previously, incorporating meteorological attributes, cyclical time encodings, and validated consumption figures.

3.6.1.1. Input Preparation

For ML models, the dataset was used in a tabular format:

1. **Features (X):** Time features (hour, month, weekday), cyclical encodings, weather variables, and lag features if applied.
2. **Target (y):** Load at the next hour.

Since ML models do not operate on sequences, no sliding window was required.

3.6.1.2. Data Splitting

The dataset was split into 80 percent for training and 20 percent for testing. Shuffling was applied during the split to prevent temporal clustering and to ensure that the machine learning models were exposed to a diverse mix of seasonal and temporal patterns during training.

3.6.1.3. Feature Scaling

Some models required normalized inputs, so StandardScaler was applied to Linear Regression, Ridge, and SVR. Tree-based models such as Random Forest, Gradient Boosting, and XGBoost did not require any scaling.

3.6.1.4. Training Procedure

Each model was trained using its corresponding optimization approach:

- Least squares optimization for Linear Regression and Ridge
- Kernel-based optimization for SVR
- Ensemble tree learning for Random Forest and Gradient Boosting
- Gradient-boosted tree optimization for XGBoost

The training process involved fitting the models on the training set, generating predictions on the test set, and evaluating performance using RMSE, MAE, MAPE, and R^2.

3.6.1.5. Hyperparameter Tuning

All machine learning models were fine-tuned using GridSearchCV, which tested different combinations of hyperparameters:

- **SVR:** C, epsilon, and gamma
- **Random Forest and Gradient Boosting:** n_estimators, max_depth, and min_samples_split
- **XGBoost:** learning_rate, max_depth, subsample, and colsample_bytree
- **Ridge:** alpha

The best configurations were selected based on the lowest validation error.

3.6.2. Training of Deep Learning Models

Two deep learning models were implemented:

1. Long Short-Term Memory (LSTM)
2. Gated Recurrent Unit (GRU)

Since deep learning models learn from sequences rather than static features, the training process follows a different pipeline.

3.6.2.1. Sequence Construction

A sliding window method was used in which the model received the past N hours as input and predicted the load for the next hour. A typical window size of 24 hours was used, although this value can be adjusted in the implementation.

3.6.2.2. Train–Validation–Test Split

Deep learning models require sequential integrity, so the dataset was split chronologically:

- 70 percent for training
- 15 percent for validation
- 15 percent for testing

No shuffling was applied, ensuring that the model learned from the natural temporal progression of the data.

3.6.2.3. Lag Feature Configurations

To capture temporal dependencies at multiple scales, three different lag configurations were evaluated:

- **Short:** Lags at [1, 3, 6] hours
- **Medium:** Lags at [1, 3, 6, 12, 24] hours
- **Long:** Lags at [1, 3, 6, 12, 24, 48] hours

The extended lag configuration proved most effective for the recurrent models by providing explicit historical context at various temporal resolutions.

3.6.2.4. Data Augmentation

To improve model generalization and increase the effective training dataset size, data augmentation techniques were applied:

- **Noise injection:** Small random noise added to training samples
- **Jittering:** Slight perturbations to feature values
- **Scaling:** Random scaling of feature magnitudes

These augmentation methods doubled the effective training size (2x augmentation factor), helping to reduce overfitting and improve model robustness.

3.6.2.5. Feature Scaling

Two scaling approaches were evaluated:

- **MinMax Scaler:** Scales features to [0, 1] range
- **Standard Scaler:** Standardizes features to zero mean and unit variance

The standard scaler combined with the long lag configuration yielded the best results for the recurrent models.

3.6.2.6. Model Training Configuration

All deep learning models were trained with the following configuration:

- **Optimizer:** Adam
- **Loss Function:** Mean Squared Error (MSE)
- **Batch Size:** Typically 32 or 64
- **Epochs:** Training continued for multiple epochs until early stopping criteria were met
- **Weight Initialization:** Xavier/Glorot initialization (TensorFlow defaults)

3.6.2.7. Regularization and Stability

To mitigate excessive fitting to training samples, multiple stabilization strategies were employed:

- Dropout layers inserted within the LSTM and GRU topologies
- Batch normalization applied at appropriate network stages
- Early termination logic that monitored validation loss and halted training once improvement stalled across several successive epochs

3.6.2.8. Model-Specific Training Notes

- **LSTM:** Ingested sequential inputs spanning a 24-hour retrospective window. Delivered solid accuracy with RMSE of 0.314 and R^2 of 0.857, affirming its competence in extracting temporal regularities from consumption data.
- **GRU:** Offered a computationally leaner substitute for LSTM while attaining marginally superior results, registering RMSE of 0.289 and R^2 of 0.879. This architecture emerged as the most effective neural-network option for the prediction task at hand.

3.6.3. Validation Approach

A uniform assessment protocol governed all algorithmic comparisons:

Machine Learning Models:

- Evaluated via GridSearchCV employing five-fold cross-validation
- Optimal hyperparameters selected according to lowest validation-set error

Deep Learning Models:

- Assessed using a 15 percent validation partition (temporally ordered)
- Early termination engaged to prevent over-learning, with a patience threshold around 10 epochs
- Weights from the best-scoring configuration retained via checkpointing
- Diverse lag arrangements and normalization schemes benchmarked methodically

3.7. Performance Evaluation

To ensure equitable comparison across ML and DL estimators, an identical suite of accuracy indicators was applied throughout. These quantities gauge the proximity of predictions to ground-truth values:

1. Mean Absolute Error (MAE)
2. Root Mean Squared Error (RMSE)
3. Mean Absolute Percentage Error (MAPE)
4. Coefficient of Determination (R^2)

Collectively, they capture precision, consistency, and aggregate prediction quality.

3.7.1. Mean Absolute Error (MAE)

MAE quantifies the average magnitude of prediction discrepancies without regard to sign. It offers intuitive interpretability and remains relatively insensitive to sporadic large deviations.

Formula:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.5}$$

where n is the number of samples, y_i is the actual value, and $\hat{y}_i$ is the prediction.

What it means: Lower MAE means predictions stay close to reality on average.

3.7.2. Root Mean Squared Error (RMSE)

RMSE is one of the most common metrics in load forecasting. It stays in the same units as the original data (MW) and punishes big errors harder than small ones.

Formula:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.6}$$

where n is the count, y_i is actual load, and $\hat{y}_i$ is predicted load.

What it means: Lower RMSE indicates better overall accuracy. Because errors are squared, large misses hurt the score more than with MAE.

3.7.3. Mean Absolute Percentage Error (MAPE)

MAPE shows errors as a percentage of actual values, making it easy to compare performance across feeders or demand levels.

Formula:

$$\text{MAPE} = \frac{100}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \tag{3.7}$$

where n is the sample count, y_i is actual load, and $\hat{y}_i$ is the forecast.

What it means: Lower MAPE is better. The formula breaks when actual load is zero, but that never happened here since the feeder always runs.

3.7.4. Coefficient of Determination (R^2 Score)

R^2 tells how much of the variation in load the model explains.

Formula:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{3.8}$$

where y_i is actual, $\hat{y}_i$ is predicted, and $\bar{y}$ is the mean of all actuals.

What it means: R^2 of 1 means perfect predictions; 0 means the model is no better than just guessing the average. Higher is better, though negative values are possible if the model is really bad.

CHAPTER FOUR: RESULTS AND DISCUSSION

This chapter shows how all the ML and DL models performed on the Baneshwor Feeder forecasting task. Every model was measured with the same metrics (MAE, RMSE, MAPE, R^2) so the comparison stays fair.

4.1. Exploratory Data Analysis

Before building models, the data was explored to understand patterns, spot outliers, and check feature relationships. The merged dataset covers the full study period at hourly resolution, including MW readings, temperature, solar radiation, and humidity, plus the derived time features (hour, day, month, etc.) and their sine-cosine encodings. Quality checks confirmed no remaining gaps, no half-hour rows, and proper alignment between load and weather records.

Hourly MW values were averaged by day and plotted across the study period. The trend shows clear daily, weekly, and seasonal swings. Winter months have lower solar radiation and slightly different load behavior. Occasional dips match known outages or holidays. Solar radiation follows a strong daytime arc, which ties into its moderate correlation with load. Overall, the Baneshwor Feeder behaves like a typical mixed-use distribution feeder—strong daily cycles plus some seasonal variation.

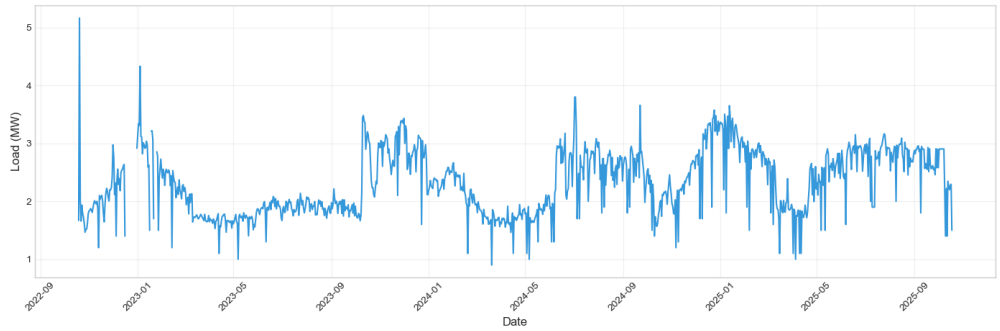

Figure 4.1 Daily Average Electricity Load Over Time

The hourly load values were averaged across the full dataset to understand the feeder's daily consumption pattern. The analysis showed that the minimum load typically occurs around 3:00 AM, which reflects low residential and commercial activity during that time. Load levels begin to rise through the morning and reach a peak at around 19:00, with the average peak load reaching approximately 3.16 MW. This aligns with evening lighting needs and heightened residential usage. A boxplot comparing load against hour of day further illustrated that evening hours exhibit higher variance, while midnight to early-morning hours display more stable and lower demand. These observations confirm that the hour of the day is one of the strongest predictors of load in this feeder.

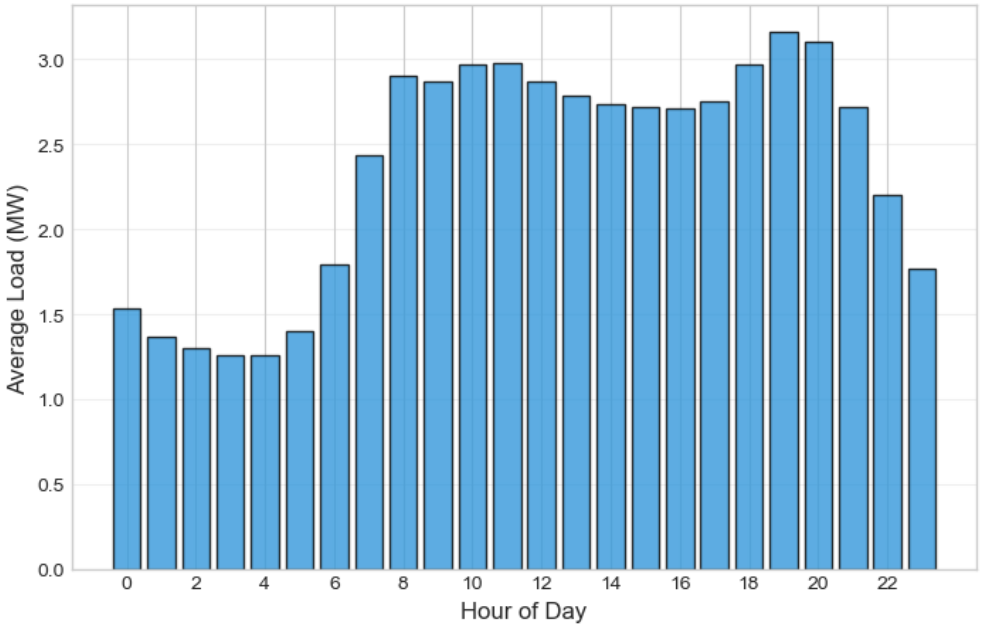

Figure 4.2 Load Distribution by Hour

Aggregating consumption by calendar month indicated that warmer periods coincide with elevated ambient temperatures, though the associated demand response exhibits year-round variability. Seasonal fluctuations are discernible yet remain secondary to the pronounced diurnal cycles characterizing this distribution line. Usage tends to climb during celebratory periods marked by heightened domestic activity. These annual rhythms are adequately encoded via the Month attribute and its paired sinusoidal transformations.

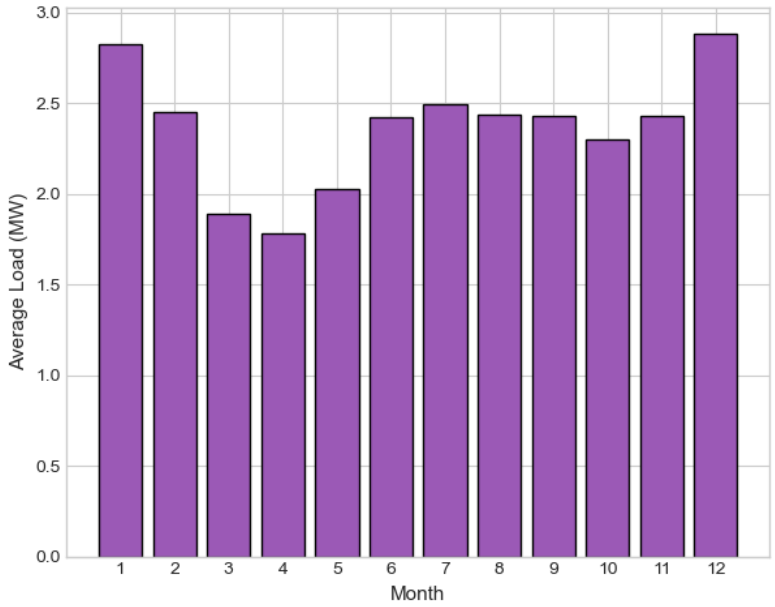

Figure 4.3 Average Load by Month

Each meteorological variable was inspected individually to assess its temporal behavior and prospective influence on consumption. Ambient temperature fluctuated between approximately 1°C and 33°C, exhibiting a recognizable diurnal oscillation with warmer midday hours and cooler nocturnal periods. Surface solar irradiance followed an unambiguous daytime-only profile, cresting sharply near solar noon and vanishing after sunset. Atmospheric moisture content tended toward higher values during evening hours and monsoon months, displaying a mild inverse association with thermal readings. The link between temperature and demand manifested as a weak positive correlation—consumption rose modestly alongside rising temperatures, consistent with mixed-use zones where ventilation and cooling appliances see amplified operation. Solar flux exhibited a moderate positive correlation with consumption, reflecting the overlap between peak irradiance intervals and active residential-commercial periods. Humidity showed a weak negative association, attributable to the tendency for elevated moisture to accompany overcast or precipitating conditions that can slightly suppress daytime demand.

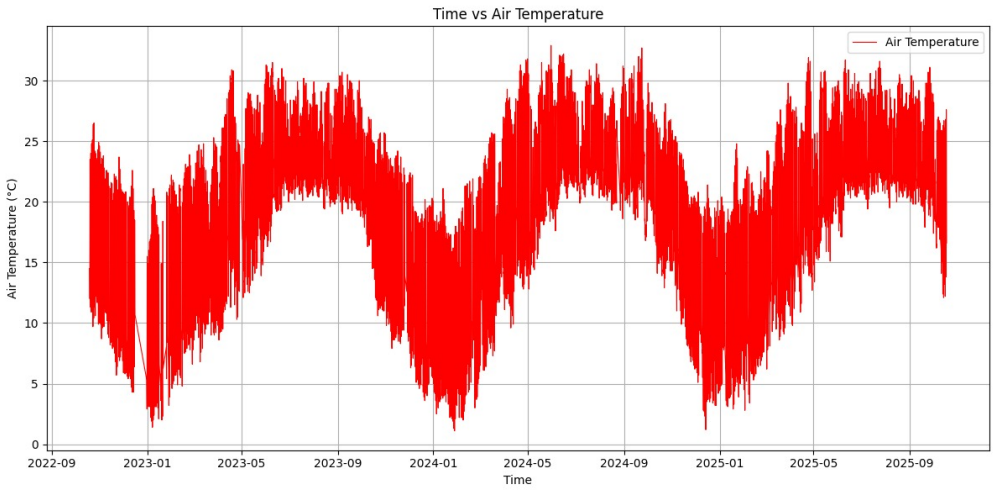

Figure 4.4 Air Temperature Variation Over the Study Period

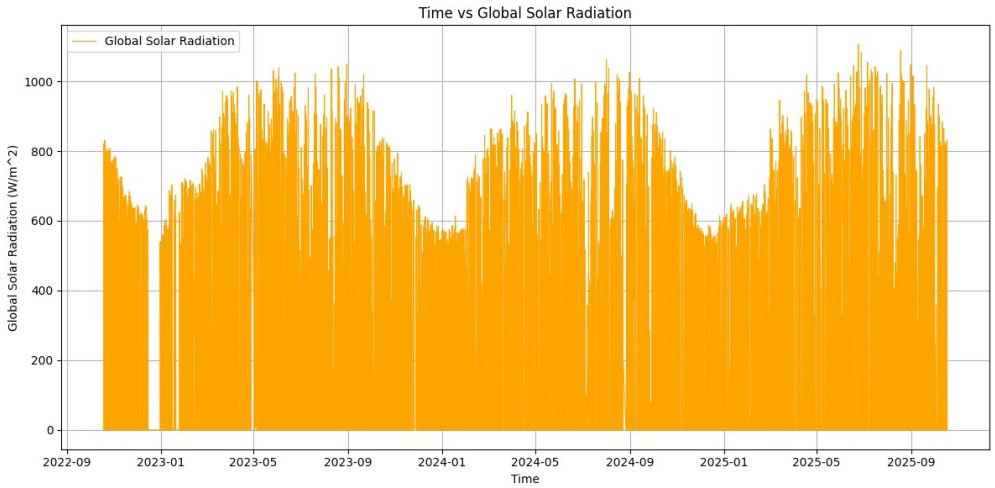

Figure 4.5 Global Solar Radiation Variation Over the Study Period

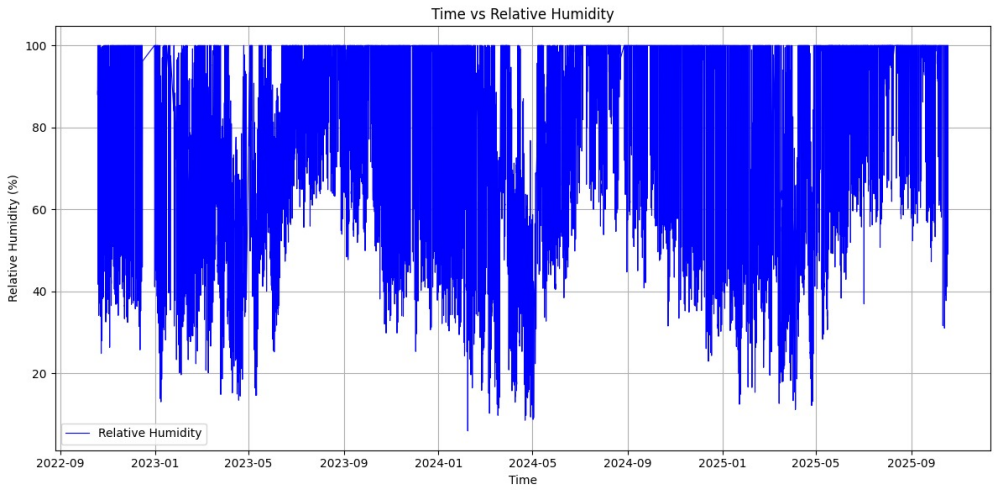

Figure 4.6 Relative Humidity Variation Over the Study Period

4.1.1. Outlier Treatment

Throughout the data integrity audit, anomalous entries within the megawatt column were flagged and addressed. For the Baneshwor distribution circuit, readings reaching or exceeding 10 MW lie outside physically plausible bounds given the line's rated capacity and operational profile. Such extreme figures most likely originate from sensor faults, transcription errors, or instrumentation artifacts rather than genuine consumption events.

In total, 56 observations registering MW values at or above 10 were classified as anomalies, constituting roughly 0.23% of the aggregate sample. These irregular entries spanned 10.30 MW through 404.00 MW—magnitudes far exceeding the realistic operating envelope of this feeder. A deterministic threshold rule was applied to excise all records meeting or surpassing the 10 MW cutoff.

Figure 4.7 contrasts the MW distribution prior to and following anomaly removal. The left-hand panel depicts the raw distribution with extreme tails reaching 404 MW, whereas the right-hand panel illustrates the cleansed distribution confined to the viable single-digit domain (0.00–8.50 MW). Post-treatment, the refined dataset comprised 24,352 records deemed suitable for algorithmic training and validation.

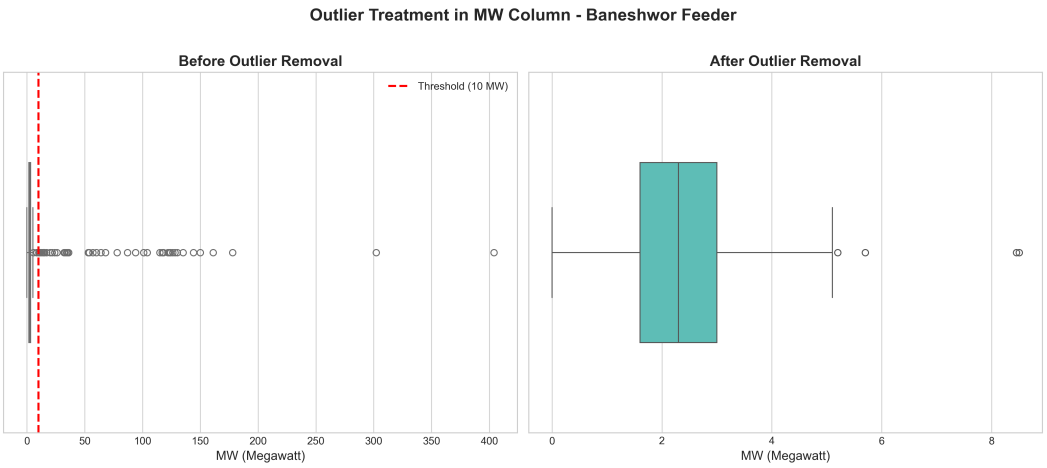

Figure 4.7 Boxplot of MW Column Before and After Outlier Removal

In addition to active power, the exogenous weather variables were also screened for anomalous values. Relative humidity readings below physically realistic levels and a few saturated values above the upper bound were removed, yielding a more compact distribution centered between roughly 60% and 100%. Air temperature exhibited a small number of extreme low and high values that were inconsistent with the local climate; these were treated using the same boxplot-based rule, resulting in a stable range of approximately 1°C to 33°C. Global solar radiation initially contained several unrealistically high spikes (exceeding 1000 W/m^2), which were removed so that the cleaned series remained within a plausible envelope of about 0–800 W/m^2. Figures 4.8–4.10 summarize the effect of the outlier removal procedure on these three weather variables, where the left panel shows the original distribution and the right panel shows the cleaned distribution used for model training.

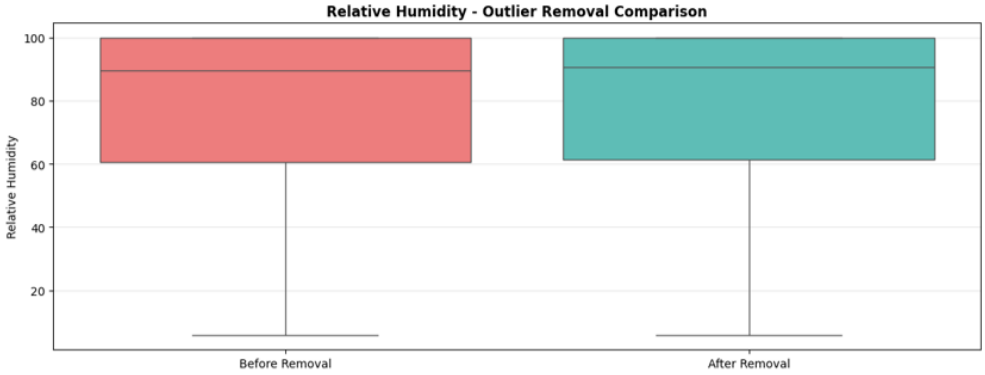

Figure 4.8 Relative Humidity – Outlier Removal Comparison

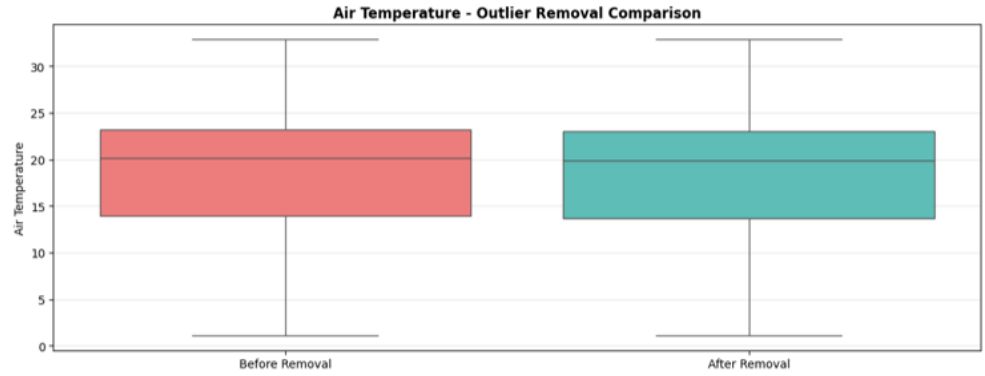

Figure 4.9 Air Temperature – Outlier Removal Comparison

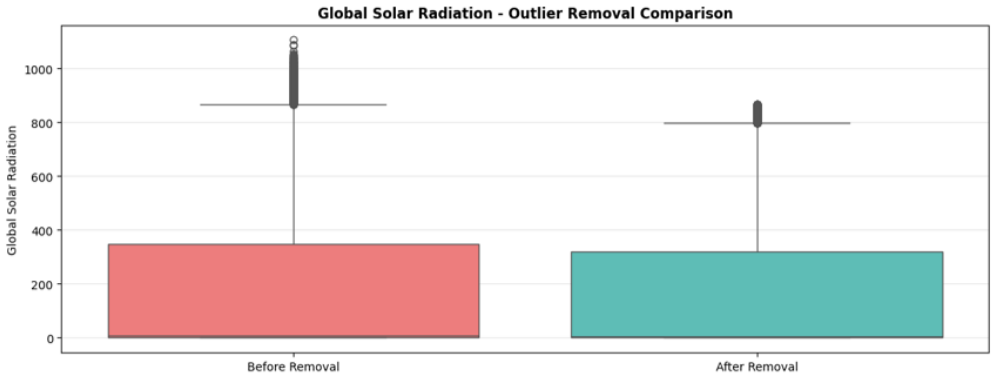

Figure 4.10 Global Solar Radiation – Outlier Removal Comparison

Table 4.1 presents the descriptive statistics of the MW column after outlier removal. The mean load of 2.35 MW and median of 2.30 MW indicate a relatively symmetric distribution. The standard deviation of 0.92 MW reflects moderate variability in hourly demand, while the minimum (0.00 MW) and maximum (8.50

MW) values confirm that all records now fall within the technically feasible range for the Baneshwor Feeder.

Table 4.1 Descriptive Statistics of MW Column (After Outlier Removal)

Statistic	**Value**
Mean	2.3483 MW
Median	2.3000 MW
Standard Deviation	0.9186 MW
Variance	0.8438 MW^2
Minimum	0.0000 MW
Maximum	8.5000 MW

4.2. Model Performance Results

4.2.1. Machine Learning Model Performance

All machine learning models were trained on the final feature-engineered dataset and evaluated on the test set. To optimize model performance, hyperparameter tuning was conducted using GridSearchCV with five-fold cross-validation. Table 4.2 summarizes the hyperparameter search space explored for each machine learning model. The best-performing configurations were selected based on the lowest validation error.

Table 4.2 Hyperparameter Search Space for Machine Learning Models

Model	**Hyperparameters**
Ridge Regression	α = [0.001, 0.01, 0.1, 1, 10, 100]
Random Forest	Number of trees = [100, 200]
	Max depth = [10, 15, 20]
	Min samples split = [2, 5]
	Min samples leaf = [1, 2]
Gradient Boosting	Estimators = [100, 150, 200]
	Learning rate = [0.05, 0.1, 0.15]
	Max depth = [3, 5, 7]
XGBoost	Estimators = [100, 200]
	Max depth = [4, 6, 8]
	Learning rate = [0.05, 0.1]
	Subsample = [0.8, 1.0]
SVR	C = [1, 10, 100]
	Gamma = [scale, 0.01, 0.1]
	Epsilon = [0.01, 0.1, 0.5]

After tuning, the models were evaluated on the test set. Table 4.3 presents the performance of all machine learning models.

Table 4.3 Machine Learning Models Evaluation Matrix

Sn.No.	Model	MAE	RMSE	MAPE	R^2
1	XGBoost (Tuned)	0.257	0.384	12.693	0.831
2	Random Forest (Tuned)	0.294	0.435	14.290	0.783
3	Random Forest	0.305	0.444	14.825	0.774
4	XGBoost	0.313	0.449	15.242	0.769
5	Gradient Boosting	0.330	0.469	16.062	0.749
6	SVR	0.318	0.483	15.193	0.732
7	Ridge Regression	0.502	0.649	25.021	0.518
8	Linear Regression	0.502	0.649	25.021	0.518

4.2.1.1. Discussion of Results

Each ML estimator underwent evaluation with MAE, RMSE, MAPE, and R^2. Linear and Ridge Regression functioned as reference benchmarks; their elevated error magnitudes and modest R^2 values indicate an inability to represent the nonlinear characteristics inherent in consumption data. SVR delivered measurable improvement, lowering deviations appreciably, yet still lagged behind the ensemble approaches.

Random Forest and Gradient Boosting outpaced all alternatives across every indicator, modeling intricate variable interactions that elementary estimators overlook. Optimized XGBoost claimed the top ML position with RMSE 0.384, R^2 0.831, and MAPE 12.69%. Its advantage stems from iterative gradient refinement, integrated penalty terms, and effective treatment of predictor interdependencies.

From a trading vantage, a MAPE hovering around 12–13% marks a meaningful advance over conventional statistical baselines and satisfies requirements for hour-ahead intraday recalibration [3]. Each fractional-point enhancement translates into reduced settlement exposure over extended horizons. The consistent R^2 performance spanning peak and off-peak intervals attests to model dependability

during high-stakes periods.

In summary: tree-based ensembles—XGBoost in particular—produce the strongest ML outcomes for distribution-line near-term prediction and dovetail effectively with market-operation imperatives.

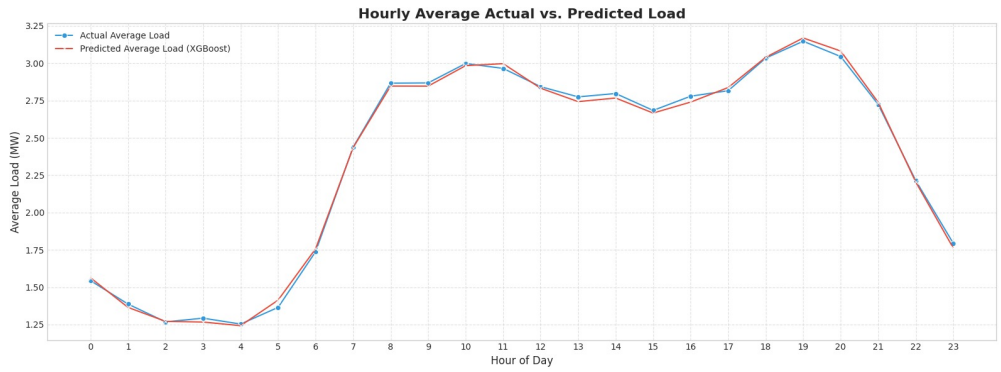

Figure 4.11 XBoost (Tuned) Actual vs Predicted

4.2.2. Deep Learning Model Performance

Neural-network estimators were fitted using historical lag features and rolling-window sequences to encode temporal structure within the feeder consumption series. Two recurrent designs were realized: LSTM and GRU. Systematic experiments explored multiple lag arrangements and normalization strategies to pinpoint optimal setups. Table 4.4 enumerates the hyperparameter and configuration space examined for the deep learning algorithms.

Table 4.4 Hyperparameter Search Space for Deep Learning Models

Component	Values
Lag Feature Configurations	
Short lags	[1, 3, 6] hours
Medium lags	[1, 3, 6, 12, 24] hours
Long lags	[1, 3, 6, 12, 24, 48] hours
Feature Scaling Methods	
MinMax Scaler	Scales to [0, 1] range
Standard Scaler	Zero mean, unit variance
LSTM / GRU Architecture	
Hidden units	[32, 64, 128]
Dropout rate	[0.0, 0.1, 0.2]
Activation function	ReLU
Sequence Modeling (LSTM/GRU)	
Look-back window	24 hours
Batch size	32
Training Parameters	
Learning rate	0.001
Optimizer	Adam
Max epochs	100
Early stopping patience	10 epochs
Data augmentation	2x (noise, jittering, scaling)

Upon completing extensive training runs on the sanitized and augmented dataset, the neural-network estimators underwent assessment with the identical accuracy measures applied to classical ML counterparts. Table 4.5 summarizes the top-performing configuration for each recurrent architecture. The GRU implementation employing the extended lag schedule [1, 3, 6, 12, 24, 48] coupled with z-score normalization registered the strongest deep-learning outcome.

Table 4.5 Deep Learning Models Evaluation Matrix

Sn.No.	Model	MAE	RMSE	MAPE	R^2
1	GRU	0.192	0.289	7.364	0.879
2	LSTM	0.205	0.314	7.612	0.857

4.2.2.1. Discussion of Results

Both LSTM and GRU underwent assessment using identical accuracy indicators. GRU achieved the leading position with MAE 0.192, RMSE 0.289, MAPE 7.36%, and R^2 0.879. LSTM trailed narrowly with RMSE 0.314 and R^2 0.857. The fact that both architectures exceeded 0.85 on R^2 confirms their capacity to capture temporally structured behavior within the consumption series.

GRU's slight advantage likely derives from its more streamlined gating mechanism, which operates effectively with a reduced parameter count and accelerated convergence. Both networks assimilated the rolling-window sequences and profited from the extended lag arrangement [1, 3, 6, 12, 24, 48 hours], endowing them with direct visibility into historical values spanning multiple temporal horizons.

From a market-operations perspective, a MAPE near 7% falls securely within the 5–10% corridor generally deemed tolerable for intraday settlement purposes [6]. Such precision can materially trim deviation-related charges. The multi-resolution lag features equip the algorithm to monitor hour-over-hour and day-over-day rhythms—precisely the intelligence utilities require when recalibrating commitments at short notice.

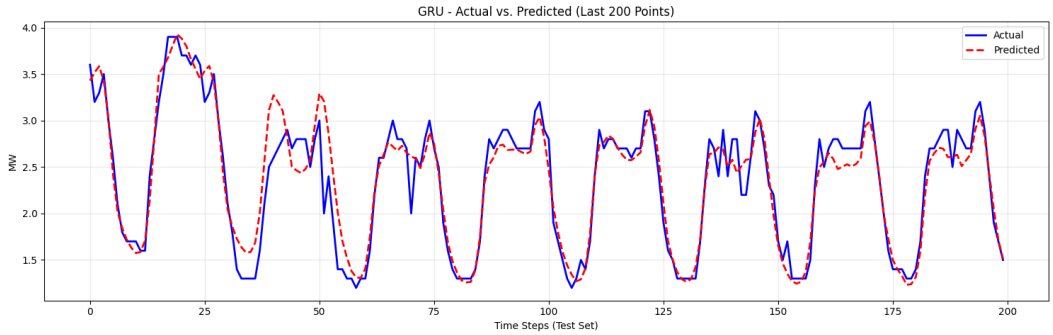

Figure 4.12 GRU Model Actual vs Predicted Values (Last 200 Test Points)

CHAPTER FIVE: CONCLUSION

5.1. Conclusion

This thesis assembled and validated a comprehensive near-term demand prediction framework for the Baneshwor distribution line employing both classical ML and neural-network DL strategies. The investigation proceeded through a structured sequence: data collection and cleansing, attribute derivation, algorithm training, parameter refinement, and comparative benchmarking using established accuracy indicators. The entire architecture targets the one-hour-ahead window—the rapid-turnover intraday interval during which grid operators must rebalance positions swiftly.

Within liberalized power markets, prediction quality carries direct financial consequences. Day-ahead auctions demand forecasts roughly 24 hours prior for commitment scheduling, while intraday sessions rely on hour-ahead estimates for position correction and real-time equilibration. The algorithms developed here align with these operational cadences, facilitating reduced imbalance exposure and tighter supply-demand coordination [3, 11].

Outcomes demonstrate that both classical ML and recurrent DL can attain accuracy levels suitable for market engagement. Among ML estimators, optimized XGBoost registered RMSE 0.384, R^2 0.831, and MAPE 12.69%. Random Forest and Gradient Boosting trailed only marginally, corroborating the aptitude of tree-based ensembles for distribution-line demand estimation.

Among DL architectures, both sequential models achieved MAPE below 8%, falling within the band deemed practical for intraday operations [6]. GRU delivered the strongest overall performance: RMSE 0.289, R^2 0.879, MAPE 7.36%. LSTM followed closely with RMSE 0.314 and R^2 0.857. The extended lag arrangement [1, 3, 6, 12, 24, 48 hours] enabled both architectures to discern the hour-ahead regularities that intraday settlement hinges upon.

In short, this work proves that careful feature engineering combined with either tuned boosted trees or well-configured recurrent networks can deliver forecasts accurate enough for hour-ahead market operations. Choosing between ML and DL comes down to data size, compute budget, and operational needs.

As Nepal moves toward more market-driven power trading and cross-border exchange, these hour-ahead forecasting tools lay groundwork for efficient participation. Better forecasts translate to less wasted hydropower, lower import bills, and a stronger position in regional energy trade—benefits that ripple through the whole economy.

5.2. Research Limitations

Notwithstanding the favorable outcomes, several constraints merit acknowledgment:

1. **Data volume:** An extended historical archive could sharpen seasonal representation and enhance out-of-sample robustness.
2. **Anomalous feeder events:** Outages, consumption spikes, and irregular occurrences can disrupt DL convergence when supplementary contextual signals are unavailable.
3. **Meteorological resolution:** Only hourly climatological readings were accessible; sub-hourly granularity or additional variables (wind velocity, rainfall intensity) could strengthen predictive capability.
4. **Absence of real-time trials:** The analysis remained within an offline experimental setting; live integration with NEA operational systems was not attempted.
5. **Lack of market-based validation:** Although the framework conforms to trading timelines, the algorithms have not been exercised under authentic auction conditions where price dynamics and bidding protocols introduce further complexity.

These considerations should inform any future extension or practical deployment of the methodology.

5.3. Implications

The outcomes yield actionable insights for grid operators, infrastructure planners, and trading entities:

1. **Viable at feeder scale:** Refined XGBoost (R^2 0.831, MAPE 12.69%) alongside GRU (R^2 0.879, MAPE 7.36%) generate sufficiently accurate predictions for dispatch coordination and peak-period handling.

2. **Suited to trading environments:** GRU's 7.36% MAPE falls squarely within acceptable bounds for intraday settlement, enabling position refinement and deviation management as Nepal's power sector liberalizes [3].

3. **Value of engineered attributes:** Sinusoidal encodings, historical lags, and climatological inputs lifted every estimator, underscoring the importance of domain-informed feature construction.

4. **Effectiveness of sequential networks:** GRU and LSTM alike captured temporal regularities adeptly, affirming that recurrent designs suit consumption prediction when configured appropriately.

5. **Basis for market-facing applications:** Generated forecasts can inform demand-response initiatives, intelligent-grid controllers, load-shifting schemes, and bilateral trading arrangements.

6. **Transferable methodology:** The established workflow extends to additional NEA distribution circuits with modest adaptation, permitting system-wide rollup.

5.4. Recommendation

Although the developed models deliver satisfactory performance, several avenues could elevate accuracy and real-world utility:

1. **Expand the predictor set:** Public holidays, cultural celebrations, wholesale price indices, and macroeconomic proxies might capture consumption fluctuations beyond what meteorological and temporal variables alone reveal.

2. **Explore contemporary architectures:** Self-attention modules, transformer encoders, or convolutional-recurrent hybrids may extract additional predictive signal from hour-ahead windows.

3. **Operationalize the solution:** Deploying the top-performing algorithm (GRU or XGBoost) within a streaming inference setup featuring online weight updates would enable continual adaptation to evolving consumption trends.

4. **Integrate with trading platforms:** As Nepal transitions toward competitive power exchanges, coupling forecast outputs with automated offer engines and decision-support dashboards would translate predictions into executable market positions [10].

5. **Incorporate uncertainty quantification:** Probabilistic outputs such as prediction intervals or quantile estimates would equip utilities to hedge against price volatility more effectively [3].

REFERENCES

[1] E. Aguilar Madrid and N. Antonio. "Short-term electricity load forecasting with machine learning". In: *Information* 12.2 (2021). DOI: 10.3390/info12020050.

[2] K. Chapagain et al. "Short-term electricity demand forecasting for Kathmandu Valley, Nepal". In: *Kathmandu University Journal of Science, Engineering and Technology* 15.3 (2021). DOI: 10.3126/kuset.v15i3.63328.

[3] Rafał Weron. "Electricity price forecasting: A review of the state-of-the-art with a look into the future". In: *International Journal of Forecasting* 30.4 (2014), pp. 1030–1081.

[4] Daniel S Kirschen and Goran Strbac. "Fundamentals of Power System Economics". In: *John Wiley & Sons* (2018).

[5] Tao Hong and Shu Fan. "Energy forecasting: Past, present, and future". In: *Foresight: The International Journal of Applied Forecasting* 40 (2016), pp. 43–48.

[6] Jesus Lago et al. "Forecasting day-ahead electricity prices: A review of state-of-the-art algorithms, best practices and an open-access benchmark". In: *Applied Energy* 293 (2021), p. 116983.

[7] Mohammad Shahidehpour, Hatim Yamin, and Zuyi Li. "Market operations in electric power systems: Forecasting, scheduling, and risk management". In: *John Wiley & Sons* (2002).

[8] S. Acharya, M. C. Luintel, and M. Badrudoza. "Short-term load forecasting of Gothatar feeder of Nepal Electricity Authority using Recurrent Neural Network". In: *Unpublished manuscript* (2021).

[9] M. K. Singla et al. "Electrical load forecasting using machine learning". In: *International Journal* 8.3 (2019).

[10] Antonio J Conejo, Miguel Carrión, and Juan M Morales. "Decision making under uncertainty in electricity markets". In: *International Series in Operations Research & Management Science* 153 (2010).

[11] Hamidreza Zareipour, Kankar Bhattacharya, and Claudio A Canizares. "Electricity market price volatility: The case of Ontario". In: *Energy Policy* 35.9 (2007), pp. 4739–4748.

[12] S. Desai et al. "Electrical load forecasting using machine learning". In: *2021 International Conference on System, Computation, Automation and Networking (ICSCAN)*. 2021, pp. 1–6. DOI: 10.1109/ICSCAN53069.2021.9526444.

[13] P. Matrenin et al. "Medium-term load forecasting in isolated power systems based on ensemble machine learning models". In: *Energy Reports* 8 (2022), pp. 612–618. DOI: 10.1016/j.egyr.2021.11.175.

[14] W. Guo et al. "Machine learning-based methods in short-term load forecasting". In: *The Electricity Journal* 34.1 (2021), p. 106884. DOI: 10.1016/j.tej.2020.106884.

[15] M. Saglam et al. "Instantaneous electricity peak load forecasting using optimization and machine learning". In: *Energies* 17.4 (2024). DOI: 10.3390/en17040777.

[16] S. Jain and A. Gupta. "Comparative analysis of machine learning algorithms for short-term power load prediction". In: *International Journal of Energy Systems* 19.2 (2024), pp. 55–68.

[17] M. Cordeiro-Costas et al. "Load forecasting with machine learning and deep learning methods". In: *Applied Sciences* 13.13 (2023), p. 7933. DOI: 10.3390/app13137933.

[18] X. Dong et al. "A decade of deep learning-based short-term electricity load forecasting: A comprehensive review". In: *Energy AI* 8 (2024), p. 100212.

[19] J. Wen, H. Li, and T. Zhao. "GRU–TCN hybrid neural network with attention for short-term load forecasting". In: *Energy* 290 (2024), p. 130423.

[20] O. Alhussein et al. “Convolutional neural network and long short-term memory hybrid deep learning model for individual load forecasting”. In: *Energies* 13.19 (2020). DOI: 10.3390/en13195396.

[21] H. Hasanat, A. Biswas, and M. Abdullah. “Parallel multichannel CNN–BiLSTM architecture for smart-grid load forecasting”. In: *IEEE Access* 12 (2024), pp. 33211–33225.

[22] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[23] K. Chan and C. Yeo. “Sparse transformer-based architecture for electricity load forecasting”. In: *Applied Energy* 352 (2024), p. 122211.

[24] Y. Zhang, Q. Li, and M. Chen. “Time-augmented transformer for short-term electrical load forecasting”. In: *IEEE Transactions on Power Systems* 37.6 (2022), pp. 5214–5225.

[25] X. Lu and Y. Chen. “Multivariate data-slicing transformer neural network for load forecasting in renewable-integrated power systems”. In: *Electric Power Systems Research* 229 (2024), p. 110138.

[26] S. Banik and S. Biswas. “A stacked ensemble learning model for renewable power and load forecasting”. In: *Energy Reports* 10 (2024), pp. 112–123.

[27] Benjamin F Hobbs et al. “Evaluation of a truthful revelation auction in the context of energy markets with nonconcave benefits”. In: *Journal of Regulatory Economics* 18.1 (2001), pp. 5–32.

[28] Arthur E. Hoerl and Robert W. Kennard. *Ridge regression: Biased estimation for nonorthogonal problems*. Vol. 12. 1. Taylor & Francis, 1970, pp. 55–67.

[29] Harris Drucker et al. “Support vector regression machines”. In: *Advances in Neural Information Processing Systems* 9 (1997), pp. 155–161.

[30] Leo Breiman. “Random forests”. In: *Machine Learning* 45.1 (2001), pp. 5–32.

[31] Jerome H. Friedman. “Greedy function approximation: A gradient boosting machine”. In: *Annals of Statistics* 29.5 (2001), pp. 1189–1232.

[32] Tianqi Chen and Carlos Guestrin. “XGBoost: A scalable tree boosting system”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2016, pp. 785–794.

[33] Diederik P. Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[34] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[35] Kyunghyun Cho et al. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *arXiv preprint arXiv:1406.1078* (2014).