
TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

A THESIS REPORT ON

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

**SUBMITTED BY:**

SUJIT KOIRALA
(PUL075MSPSE016)

**SUPERVISED BY:**

PROF. DR. ROGERS S PRESSMAN

**SUBMITTED TO:**

DEPARTMENT OF ELECTRICAL ENGINEERING

December, 2025

# Declaration

I hereby declare that this study/research entitled [**Put the title of the project/thesis here....**] is based on our original research work. Related works on the topic by other researchers have been duly acknowledged. I owe all the liabilities relating to the accuracy and authenticity of the data and any other information included hereunder.

**Name of the Student ( Roll no)**

Date:

# Recommendation

This is to certify that this project report entitled [**Put the title of the project/thesis here**] prepared and submitted by [**Put the name & roll no of the student here**], in partial fulfillment of the requirements of the Master degree of Engineering in....., awarded by Tribhuvan University, has been completed under my/our supervision. I/we recommend the same for acceptance by Tribhuvan University.

---

Name of the Supervisor: ABC
Designation: ABC
Organization: ABC
Date: ABC

---

Name of the Co-supervisor: ABC
Designation: ABC
Organization: ABC
Date: ABC

# Page of Approval

TRIBHUVAN UNIVERSIY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

This project/thesis entitled [**Put the title of the project/thesis here**] prepared and submitted by [**Put the Name and Roll no of the student here**] has been examined by us and is accepted for the award of the Master's degree in [Put name of the program] by Tribhuvan University.

.....
Supervisor
**Name**
Designation
Organization Name and Address.

.....
External examiner
**Name**
Designation
Organization Name and Address.

.....
Co-Supervisor (if Any)
**Name**
Designation
Organization Name and Address.

Date of approval:

# Copyright

# Acknowledgments

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)**

# Abstract

Accurate short-term electrical load forecasting plays a crucial role in the efficient planning and operation of modern power systems. With increasing load variability influenced by weather conditions, temporal patterns, and socio-economic activities, traditional statistical methods often struggle to capture complex and nonlinear demand behavior. This project focuses on short-term electrical load forecasting for the Lekhnath Feeder using machine learning–based approaches.

Historical hourly load data, along with meteorological variables such as air temperature, global solar radiation, and relative humidity, were used to develop predictive models. Comprehensive data preprocessing was performed, including missing value imputation, outlier treatment, temporal feature extraction, and cyclical encoding of time-based variables. Several machine learning models were implemented and evaluated, including Linear Regression, Ridge Regression, Support Vector Regression, Random Forest, Gradient Boosting, and XGBoost. Hyperparameter tuning was applied to improve model performance.

The models were assessed using standard evaluation metrics such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), and R-squared ($R^2$). The results show that ensemble-based models, particularly tuned XGBoost and Random Forest models, significantly outperform linear and baseline methods. The findings highlight the effectiveness of machine learning techniques for feeder-level short-term load forecasting and provide valuable insights for operational planning and decision-making in power distribution systems.

Keywords: *short-term load forecasting, machine learning, XGBoost, Random Forest, power distribution systems*

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| **NEA** | Nepal Electricity Authority |
| **STLF** | Short-Term Load Forecasting |
| **ML** | Machine Learning |
| **DL** | Deep Learning |
| **RNN** | Recurrent Neural Network |
| **LSTM** | Long Short-Term Memory |
| **GRU** | Gated Recurrent Unit |
| **MLP** | Multi-Layer Perceptron |
| **SVR** | Support Vector Regression |
| **RF** | Random Forest |
| **GBR** | Gradient Boosting Regressor |
| **XGBoost** | Extreme Gradient Boosting |
| **MAE** | Mean Absolute Error |
| **RMSE** | Root Mean Squared Error |
| **MAPE** | Mean Absolute Percentage Error |
| **SMAPE** | Symmetric Mean Absolute Percentage Error |
| $\mathbf{R^2}$ | Coefficient of Determination |
| **MW** | Megawatt |
| **BS** | Bikram Sambat (Nepali Calendar) |
| **AD** | Anno Domini (Gregorian Calendar) |
| **EDA** | Exploratory Data Analysis |
| **IQR** | Interquartile Range |
| **TFT** | Temporal Fusion Transformer |
| **API** | Application Programming Interface |

# 1. Introduction

This project focuses on short-term electrical load forecasting at the feeder level using data-driven machine learning and deep learning techniques. Historical load data combined with weather and temporal features were used to model and predict hourly power demand. Multiple forecasting models were developed and evaluated to identify the most effective approach for accurate and reliable load prediction.

## 1.1 Background

Electricity demand is never constant. It rises and falls with daily routines, temperature changes, business hours, and countless other factors. For a power system operator, being able to predict this demand even just a few hours ahead can make a huge difference. Accurate short-term forecasting helps optimize generation schedules, reduce operational costs, manage peak hours more confidently, and maintain a reliable supply.

Short-Term Load Forecasting (STLF) typically focuses on horizons ranging from one hour to a day ahead. These forecasts are critical for economic dispatch, unit commitment, load flow analysis, and real-time operation. Traditionally, utilities relied on statistical approaches such as linear regression, ARIMA, exponential smoothing, and Holt-Winters. These techniques can work well when patterns are simple, but they struggle with real-world load curves that are nonlinear, noisy, and influenced by many interacting variables.

Machine Learning models like Random Forest, Support Vector Regression, and XGBoost have shown strong results in several energy-related forecasting tasks. Their ability to capture nonlinear relationships makes them a natural fit for electricity load prediction. Likewise, Deep Learning approaches, especially recurrent neural networks such as LSTM and GRU, can learn temporal dependencies more effectively than traditional models.

The Baneshwor Feeder of the Nepal Electricity Authority serves a mixed group of consumers in the Baneshwor region. 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. This creates a strong motivation to investigate how modern ML and DL models can improve forecasting performance for this specific feeder.


Figure 1.1: Baneshwor Feeder View

## 1.2 Problem Statement

The current forecasting practices for the Baneshwor Feeder rely heavily on manual estimation or basic statistical techniques. These methods do not fully capture the nonlinear and dynamic nature of the load profile, especially when multiple influencing factors, like temperature, humidity, rainfall, weekends, and special events come into play. As a result, prediction errors tend to increase during peak hours, sudden weather changes, and seasonal transitions.

Inaccurate short-term forecasts have several consequences. They can affect how generation is scheduled, leading to either unnecessary reserve margins or inadequate supply. They may increase operational costs and technical losses at the distribution level. In the worst cases, poor foresight during high-demand periods can create voltage drops, reliability concerns, or inefficient load-shedding decisions.

Despite the availability of historical load and weather data, there has not been a systematic study applying and comparing advanced machine learning and deep learning approaches specifically for the Baneshwor Feeder. The lack of a data-driven forecasting system means operators do not yet benefit from models that are capable of learning complex relationships within the data.

This thesis aims to address these gaps by building a complete forecasting framework using multiple ML and DL models, evaluating their performance, and identifying the most suitable approach for accurate short-term load prediction of the Baneshwor Feeder.

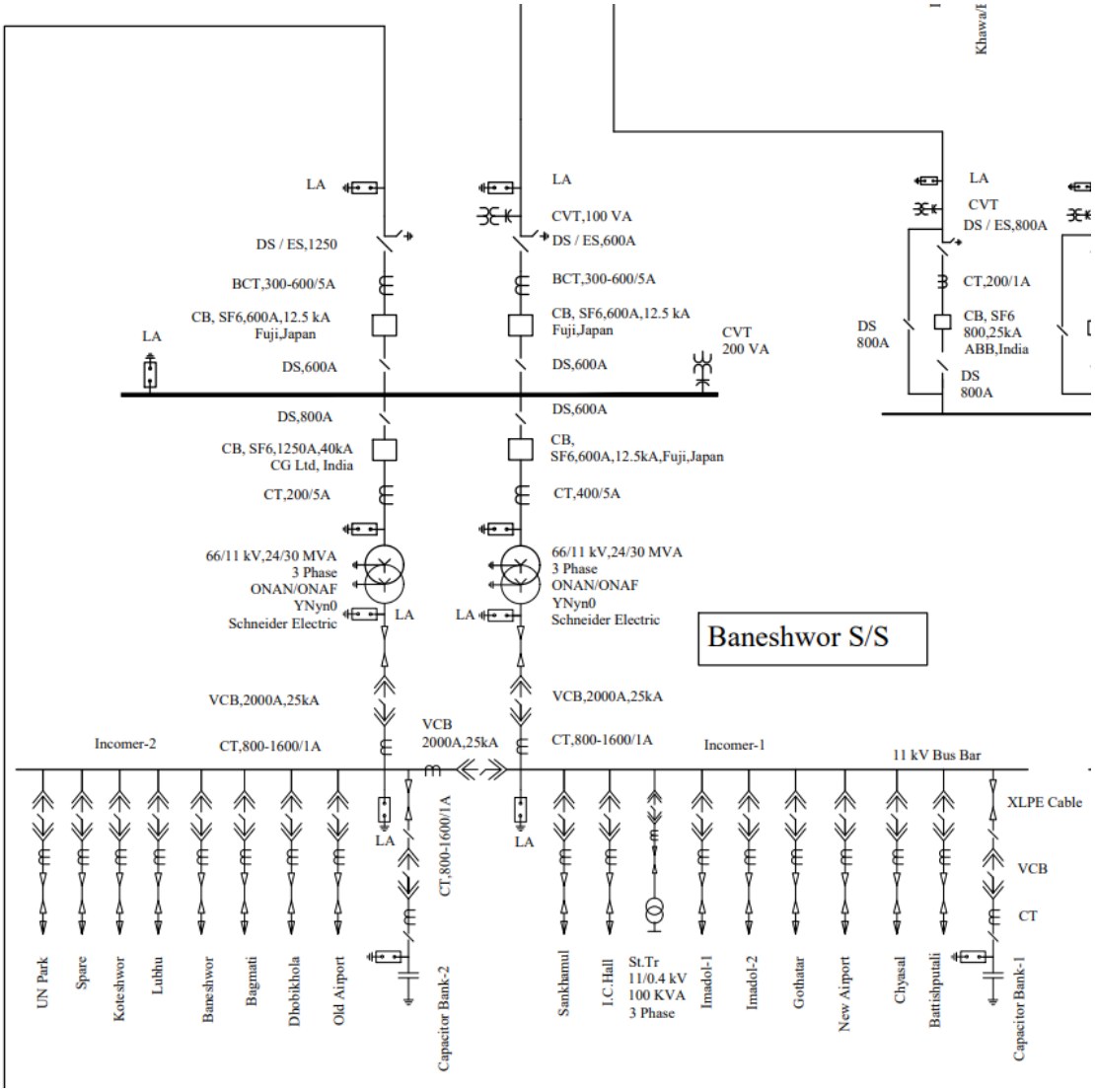


Figure 1.2: Substation & Transmission Line Network Baneshwor

## 1.3 Objectives

To develop and evaluate machine learning and deep learning models for short-term electrical load forecasting of the Baneshwor Feeder to improve prediction accuracy and operational efficiency.

1. To collect and preprocess historical load data and relevant influencing factors such as weather variables and calendar effects for the Baneshwor Feeder.

2. To implement and evaluate various machine learning models including Support Vector Regression (SVR), Random Forest (RF), and XGBoost, as well as deep learning models such as Multi-Layer Perceptron (MLP), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU), using standard error metrics (e.g., RMSE, MAPE, MAE, R-squared).

3. To recommend the most suitable forecasting model for operational use in the Baneshwor Feeder.

## 1.4 Scope

This study is geographically limited to the Baneshwor Feeder under the Nepal Electricity Authority. The temporal scope focuses on short-term load forecasting with a prediction horizon of up to 24 hours ahead, utilizing historical hourly load data as the primary foundation for model development. The dataset encompasses historical load data from the Baneshwor Feeder, complemented by weather data including temperature, humidity, and rainfall, along with calendar data distinguishing weekdays, weekends, and holidays.

From a technical perspective, the research implements several machine learning models including Support Vector Regression (SVR), Random Forest, and XGBoost, alongside deep learning architectures such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks. The performance of these models is rigorously evaluated using standard metrics including Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), Mean Absolute Error (MAE), and the coefficient of determination (R-squared). It should be noted that the accuracy of forecasts is inherently dependent on the quality and completeness of the historical data available. Additionally, this study does not extend to medium-term or long-term forecasting horizons, and the scope explicitly excludes renewable generation forecasting from its analysis.

## 1.5 Limitation

Despite the promising results obtained in this study, certain limitations were encountered, primarily related to data availability, model assumptions, and scope of analysis.

1. **Dependence on data quality:** Forecast accuracy is limited by the completeness and reliability of the historical load and weather data. Missing values, sensor errors, or inconsistent reporting can influence model performance.

2. **Model sensitivity to sudden changes:** Unexpected events such as outages, festivals, abrupt weather shifts, or abnormal consumption patterns are difficult for data-driven models to predict accurately.

3. **Deep learning computation constraints:** Training LSTM and GRU models requires more computational resources and time compared to ML models. Their performance may vary depending on the hardware used.

4. **Limited feature diversity:** Although weather and calendar data are included, other influential factors like economic activities, special events, or industrial load profiles are not part of the dataset.

5. **Generalization across feeders:** The models developed in this study are tailored specifically to the Baneshwor Feeder and may not generalize directly to other feeders without retraining or adaptation.

# 2. Literature Review

The literature review presents a comprehensive overview of existing research related to short-term electrical load forecasting using machine learning and deep learning techniques. It examines previously published studies to understand commonly used methodologies, datasets, and performance evaluation approaches in the domain of power system load forecasting. Reviewing prior work helps to identify current research trends, strengths, and limitations of existing models, while also highlighting gaps that motivate the need for this study. By situating the present research within the context of established knowledge, this chapter provides a foundation for model selection and methodological design adopted in this work.

## 2.1 Related Work

There are many previous works done for electrical load forecasting from short-term electrical load forecasting, to medium-term and long-term. Most of the studies have done short-term load forecasting.

### 2.1.1 Machine Learning Approaches

Singla et al. [1] employed Artificial Neural Networks for 24-hour short-term load forecasting, utilizing dew point temperature, dry bulb temperature, and humidity as input features. Their work demonstrated the effectiveness of ANN in capturing the relationship between weather variables and electrical load demand. Similarly, Desai et al. [2] utilized the Prophet model from Meta to perform short-term load forecasting, incorporating time, temperature, humidity, and weather forecast data as features. The Prophet model's ability to handle seasonal patterns and missing data made it suitable for load forecasting applications.

Matrenin et al. [3] conducted a study on medium-term load forecasting using ensemble machine learning models. They compared XGBoost and AdaBoost against traditional methods including SVR, decision trees, and Random Forest. Their results highlighted the superior performance of gradient boosting techniques for capturing complex load patterns. Aguilar Madrid & Antonio [4] tested five machine learning models and found XGBoost to be the most accurate for predictions, using historical load data, weather information, and holiday indicators as input features. Their comprehensive evaluation demonstrated XGBoost's ability to handle diverse feature sets effectively.

Guo et al. [5] analyzed three popular ML methods for load forecasting: Support Vector Machine, Random Forest, and LSTM. They proposed a fusion forecasting approach that

combined outputs from all three models, demonstrating that ensemble methods could improve prediction accuracy beyond individual model performance. Saglam et al. [6] 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.

Jain & Gupta [7] conducted a comprehensive evaluation of various machine learning algorithms for power load prediction, including Support Vector Machines, LSTM, ensemble classifiers, and Recurrent Neural Networks. Their study emphasized the importance of data preprocessing methods, feature selection strategies, and performance assessment metrics in achieving accurate forecasts. The research demonstrated that ensemble methods and deep learning approaches consistently outperformed traditional statistical models.

### 2.1.2 Deep Learning Architectures

Chapagain et al. [8] explored time series regression along with machine learning and deep learning models for electricity demand forecasting in Kathmandu Valley. They found LSTM demonstrating outstanding performance in terms of MAPE and RMSE, using deterministic variables such as day type and temperature. Their work validated the effectiveness of recurrent architectures for capturing temporal dependencies in load data.

Acharya et al. [9] performed short-term electrical load forecasting for the Gothatar feeder using six input features. They found that Recurrent Neural Networks outperformed baseline methods including Single Exponential Smoothing, Double Exponential Smoothing, and Holt-Winter's method. This study confirmed that RNNs could better model the nonlinear and time-dependent characteristics of feeder-level load patterns.

Cordeiro-Costas et al. [10] conducted a comprehensive comparison of load forecasting methods, including Random Forest, SVR, XGBoost, Multi-Layer Perceptron, and LSTM. They also explored Conv-1D models and found that LSTM achieved the lowest error rates across multiple evaluation metrics. Their research highlighted the trade-off between model complexity and forecasting accuracy in practical applications.

Dong et al. [11] provided a comprehensive survey on deep learning-based short-term electricity load forecasting covering the past decade. They examined the entire forecasting process, including data preprocessing, feature extraction, deep learning modeling and optimization, and results evaluation. The survey identified CNN-LSTM hybrid architectures as widely adopted solutions due to exceptional performance in capturing both spatial and temporal features. Their analysis revealed that most recent studies focused on short-term

horizons ranging from one hour to several days ahead.

### 2.1.3 Hybrid and Advanced Architectures

Wen et al. [12] proposed a hybrid deep learning model combining Gated Recurrent Units and Temporal Convolutional Networks with an attention mechanism for short-term load forecasting. The GRU captured long-term dependencies in time series data, while TCN efficiently learned patterns and features. The attention mechanism automatically focused on input components most relevant to the prediction task, significantly enhancing model performance. Their approach demonstrated superior accuracy compared to standalone architectures.

Alhussein et al. [13] developed a hybrid CNN-LSTM framework for short-term individual household load forecasting. The model used CNN layers for feature extraction from input data and LSTM layers for sequence learning. Evaluated on the Smart Grid Smart City dataset, the hybrid model achieved an average MAPE of 40.38%, outperforming standalone LSTM models that obtained 44.06% MAPE. This work demonstrated the effectiveness of combining convolutional and recurrent architectures for handling high volatility in household-level load data.

Hasanat et al. [14] proposed a parallel multichannel network approach using 1D CNN and Bidirectional LSTM for load forecasting in smart grids. Unlike traditional stacked CNN-LSTM architectures that use convolutions as preprocessing steps, their model independently processed spatial and temporal characteristics through parallel channels. The research addressed the issue of temporal feature neglect in existing models and incorporated cyclic features through trigonometric transformations, achieving superior accuracy on diverse building types.

### 2.1.4 Transformer-Based Models

Chan & Yeo [15] proposed a sparse transformer-based approach for electricity load forecasting that addressed the computational complexity limitations of standard transformer architectures. Their model applied sparse attention mechanisms to 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. The model was enhanced to support multivariate inputs including weather data, demonstrating versatility in forecasting loads from individual households to city levels.

Zhang et al. [16] developed a Time Augmented Transformer model for short-term electrical load forecasting, incorporating temporal features and self-attention mechanisms to capture complex dynamic nonlinear sequence dependencies. Their experimental results showed

that multivariate inputs including weather and calendar features produced significantly better predictions than univariate approaches. The attention mechanism's capacity to capture complex dynamical patterns in multivariate data contributed to improved forecasting accuracy.

Lu & Chen [17] proposed a multivariate data slicing transformer neural network for load forecasting in power systems with high-penetration renewables. The transformer model excelled in capturing spatiotemporal relationships by modeling global correlations through self-attention mechanisms. Their approach demonstrated superior performance in handling the intermittency and volatility characteristics brought by renewable energy integration, outperforming traditional statistical models and conventional machine learning methods.

### 2.1.5 Comparative Studies and Ensemble Methods

Banik & Biswas [18] developed an enhanced stacked ensemble model combining Random Forest and XGBoost for renewable power and load forecasting. The Random Forest model first forecasted the target variable, followed by XGBoost improving predictions through combination of RF outputs. A meta-model using logistic regression then learned the optimal combination, achieving 99% accuracy on $R^2$ evaluation metrics for both short-term and long-term predictions in Agartala City dataset.

Kwon et al. [19] conducted extensive research on learning models combined with data clustering and dimensionality reduction for short-term electricity load forecasting. They adapted k-means clustering for data grouping and utilized kernel PCA, UMAP, and t-SNE for dimensionality reduction. Applied to neural network-based models on large-scale electricity usage data from 4,710 households, their approach demonstrated improved forecasting performance through effective data preprocessing and feature engineering.

Nabavi et al. [20] combined Discrete Wavelet Transform with LSTM to improve electricity load forecasting accuracy. The DWT decomposed load series into multiple frequency components, allowing LSTM to learn from denoised and structured representations. Their research demonstrated that preprocessing techniques significantly enhanced deep learning model performance, particularly for datasets with high noise levels and irregular patterns.

## 2.2 Theoretical Background of Forecasting Models

This section provides the theoretical foundations of the machine learning and deep learning models employed in this study. Understanding the mathematical formulations and working principles of these models is essential for interpreting their performance in short-term load forecasting applications.

### 2.2.1 Machine Learning Models

#### Linear Regression

Linear Regression is the simplest baseline model that assumes the relationship between input features and the target load value is linear. Although load patterns are nonlinear, this model helps establish a reference point for evaluating more complex methods.

**Mathematical Formulation:**

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

where $\beta_0$ is the intercept (bias term), $\beta_i$ are the coefficients (weights) learned via ordinary least squares minimization, and $x_i$ are the input features.

#### Ridge Regression

Ridge Regression [21] adds L2 regularization to the linear model to reduce overfitting and stabilize coefficient estimates. It is more robust than standard Linear Regression when dealing with many correlated features, which is the case in this study.

**Mathematical Formulation:**

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

where $\alpha$ 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.

#### Support Vector Regression (SVR)

SVR [22] models nonlinear relationships by mapping features into a high-dimensional space using kernel functions. It works well for complex regression problems with moderate dataset sizes.

**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, $\varepsilon$ defines the epsilon-insensitive tube, and $\xi_i$ and $\xi_i^*$ are slack variables for points outside the tube.

### Random Forest Regressor

Random Forest [23] is an ensemble method consisting of multiple decision trees. Each tree is trained on a random subset of features and samples (bootstrap aggregating). It is robust, stable, and handles nonlinearity effectively.

**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.

### Gradient Boosting Regressor

Gradient Boosting [24] builds trees sequentially, with each new tree correcting the errors (residuals) of the previous ensemble. It is particularly effective for structured tabular data like load forecasting.

**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 $\nu$ is the learning rate (shrinkage factor) that controls the contribution of each tree.

### XGBoost Regressor

XGBoost (Extreme Gradient Boosting) [25] is an optimized and regularized implementation of gradient boosting designed for efficiency, scalability, and high accuracy. It was one of the best-performing ML models in this study.

**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, $\gamma$ controls the minimum loss reduction required to make a split, and $\lambda$ is the L2 regularization term on leaf weights.

### 2.2.2 Deep Learning Models

To model the nonlinear, temporal, and sequential characteristics of feeder load data, three deep learning architectures were implemented: Multi-Layer Perceptron (MLP), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU). These models were trained on sliding windows of historical hourly sequences, allowing the networks to learn both short-term and long-term dependencies in the dataset. All models were trained with the Adam optimizer, early stopping, and sequence-based inputs where applicable.

#### Multi-Layer Perceptron (MLP)

The MLP [26] is a feed-forward network consisting of multiple fully connected layers. Although it does not inherently model temporal order, it can extract nonlinear patterns from feature-engineered inputs. In this study, the MLP receives sequences flattened into a fixed-size input vector.

**Mathematical Formulation:**

Hidden layer activation:

$$h = \sigma(W_1 x + b_1) \tag{2.8}$$

Output layer:

$$\hat{y} = W_2 h + b_2 \tag{2.9}$$

where $\sigma$ is the activation function (ReLU in this implementation), $W_1$ and $W_2$ are weight matrices for the hidden and output layers respectively, $b_1$ and $b_2$ are bias terms, $x$ is the input feature vector, and $h$ is the hidden layer output.

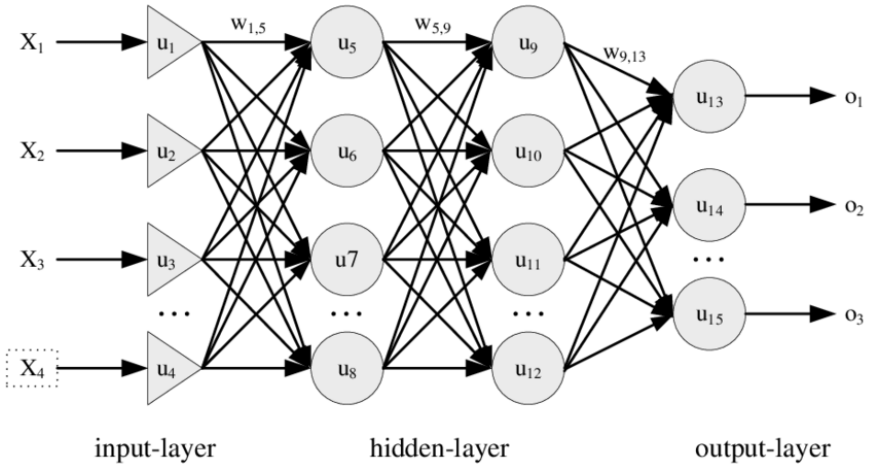


Figure 2.1: Architecture MLP

### Long Short-Term Memory (LSTM)

LSTM networks [27] are designed to maintain contextual memory over long sequences through a gating mechanism that controls information flow. This makes them naturally suited for load forecasting, where consumption patterns depend on previous hours. The LSTM architecture addresses the vanishing gradient problem that affects standard RNNs.

**Mathematical Formulation:**

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

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

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.10}$$

Input gate (determines what new information to store):

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.11}$$

Candidate cell state (creates new candidate values):

$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{2.12}$$

Cell state update (combines old and new information):

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{2.13}$$

Output gate (determines what to output):

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.14}$$

Hidden state (final output at time step $t$):

$$h_t = o_t \odot \tanh(c_t) \tag{2.15}$$

where $\sigma$ 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.

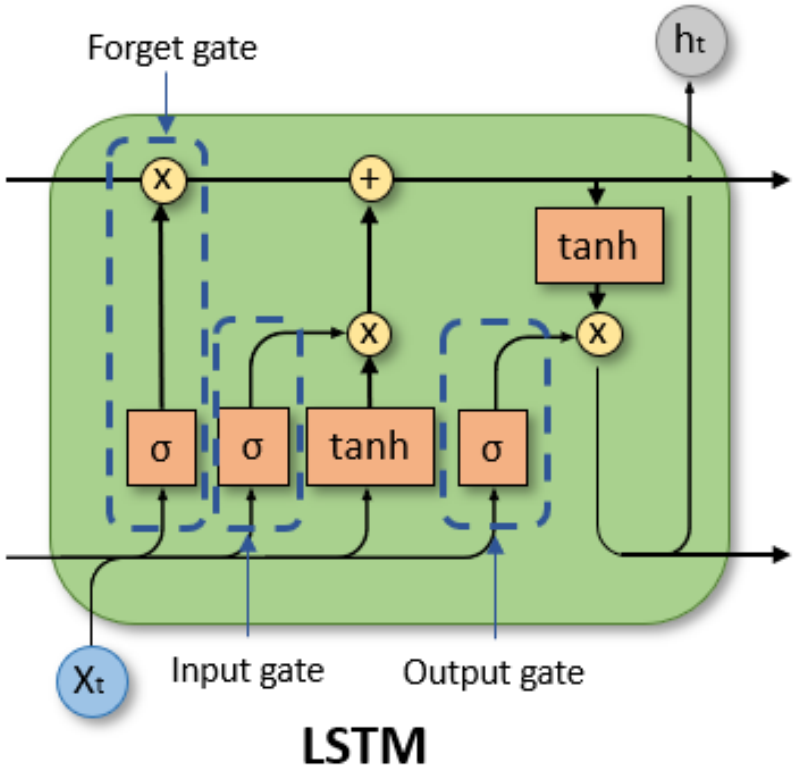


Figure 2.2: Architecture LSTM

### Gated Recurrent Unit (GRU)

GRU [28] is a streamlined version of LSTM with fewer parameters, combining the forget and input gates into a single update gate and merging the cell state with the hidden state. It often trains faster while achieving comparable performance to LSTM.

**Mathematical Formulation:**

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

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

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \tag{2.16}$$

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

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \tag{2.17}$$

Candidate hidden state (computes new candidate activation):

$$\tilde{h}_t = \tanh(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h) \tag{2.18}$$

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

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{2.19}$$

where $\sigma$ 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.

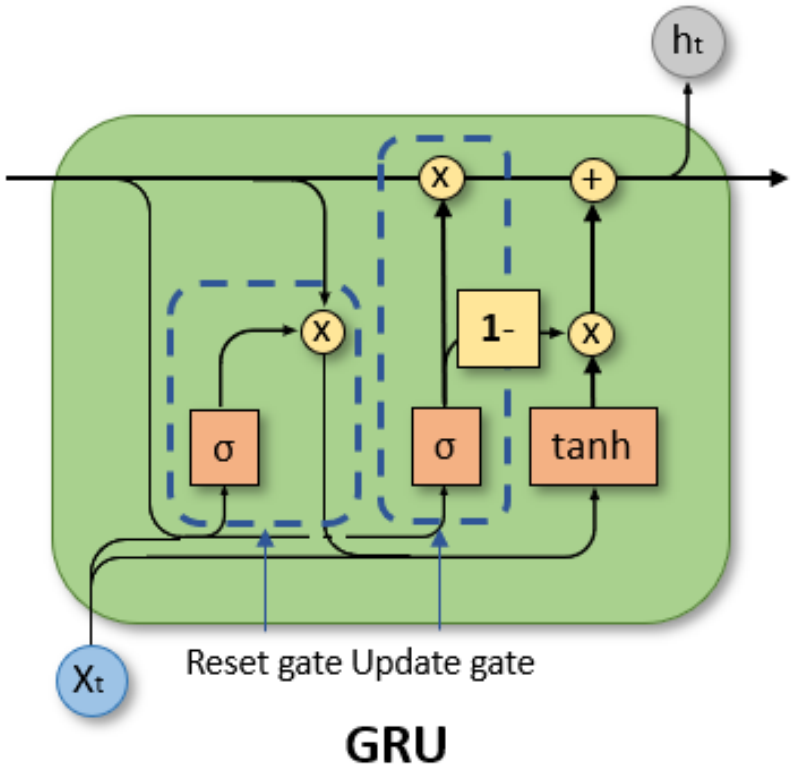


Figure 2.3: Architecture GRU

# 3. Methodology

This chapter describes the overall research methodology adopted to achieve the objectives of the thesis. It outlines the research design, data acquisition process, preprocessing techniques, and modeling approaches used for short-term electrical load forecasting. The methodology is structured to ensure a systematic analysis, with each step directly linked to the research objectives. A flowchart is used to summarize the overall framework, followed by a detailed explanation of the selected machine learning and deep learning models and the evaluation techniques employed in this study.

## 3.1 Overall Workflow

The forecasting framework adopted in this thesis follows a structured and sequential workflow in which hourly historical load data from the Baneshwor Feeder are collected together with weather variables (temperature, humidity, rainfall) and calendar information (weekday/weekend, holidays) to form the core input space for the forecasting models. The raw data are first subjected to a comprehensive preprocessing stage that addresses missing readings, outliers, and format inconsistencies through data cleaning, missing-value imputation, outlier treatment, timestamp conversion, and the engineering of time-based and cyclical features, thereby ensuring that the resulting dataset is suitable for model development. On the basis of this cleaned dataset, exploratory data analysis (EDA) is then carried out to examine load trends, seasonal patterns, hourly variations, and correlations with weather variables, so as to identify the most influential features and to guide feature selection. Subsequently, both machine learning models and deep learning models are developed, with input features standardized and organized as tabular matrices for the ML models and as sequential inputs for the DL models. These models are trained using appropriate training subsets and validated through walk-forward validation or hold-out testing, while hyperparameters are optimized using GridSearchCV for the ML models and iterative tuning strategies for the DL models to enhance generalization and mitigate overfitting. Finally, all models are evaluated and compared using RMSE, MAE, MAPE, and $R^2$, and their forecasting accuracy, stability, and computational efficiency are analyzed in order to recommend the most suitable model for operational short-term load forecasting in the Baneshwor Feeder. This workflow thus provides a complete pipeline from raw data acquisition to final model recommendation and supports both machine learning and deep learning approaches. The overall diagram of the methodology is shown in Figure 3.1.

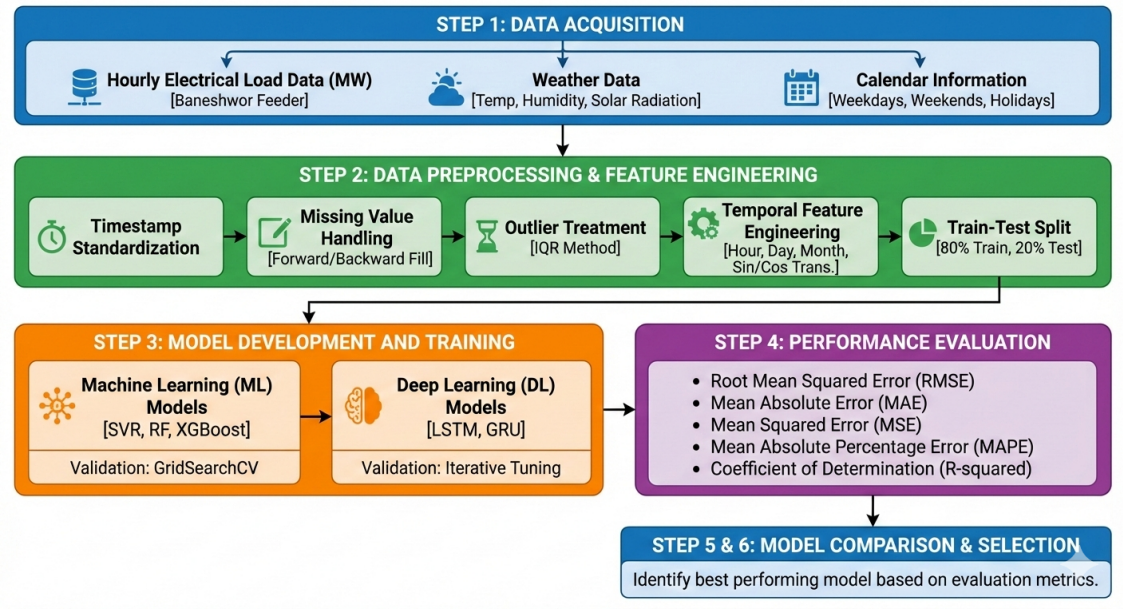


Figure 3.1: Methodology Block Diagram

## 3.2 Data Acquisition

The first step of our methodology is data acquisition. Hourly historical load data from Baneshwor Feeder will be collected. Weather data such as temperature, humidity, and rainfall data will be collected from the Department of Hydrology and Meteorology, Nepal. And finally, the information on weekdays, weekends are obtained from official government calendars. The study relies on two primary datasets:

1. Hourly electrical load data for the Baneshwor Feeder
2. Hourly weather data (temperature, humidity, and solar radiation)

Since both datasets were obtained as raw Excel files with irregular formats, inconsistent timestamps, missing entries, and multiple sheets per day/month, a multi-stage acquisition and structuring pipeline was developed. The sources of both datasets are listed below.

### 3.2.1 Data Sources

a) **Electrical Load Data:** The electrical load data used in this study was obtained from the Baneshwor Substation, which operates under the Nepal Electricity Authority (NEA). Hourly feeder load readings were collected from archived operational log sheets maintained by the substation for the years 2079 to 2082 BS. These records provided raw POWER (MW) measurements for the Baneshwor Feeder, along with associated timestamp information. Since the data originated from manually recorded and distributed Excel files, several preprocessing steps—such as header correction, timestamp

standardization, and quality checks—were required before the dataset could be used for modeling. This substation-provided dataset forms the core of the forecasting analysis, representing real operational feeder behavior across multiple years.

b) **Weather Data:** Weather data was sourced from Nepal's Department of Hydrology and Meteorology (DHM), the official governmental agency responsible for climate and atmospheric measurements. The dataset included hourly records of air temperature, relative humidity, and global solar radiation for the corresponding study period. These variables were essential for capturing the environmental conditions influencing electricity consumption patterns. The DHM dataset required timestamp alignment, interpolation for missing values, and smoothing of extreme readings to ensure compatibility with the load dataset. Once cleaned and synchronized, the weather data served as an important set of exogenous features for both machine learning and deep learning models.

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 the load and weather datasets were fully acquired and merged into a single hourly dataset, several preprocessing steps were performed to prepare the data for machine learning and deep learning models. The merged dataset initially contained timestamps in multiple formats, including irregular representations such as "24:00." All timestamps were therefore parsed and standardized using a custom parsing routine, where "24:00" was shifted to 00:00 of the following day, and the final format was normalized to a consistent hourly representation. Missing electrical load (MW) values caused by incomplete feeder logs and invalid records were handled using a forward-fill followed by backward-fill strategy to preserve temporal continuity without introducing artificial variations. Similarly, missing weather values were treated using linear interpolation, with nighttime solar radiation values set to zero and extreme humidity or temperature readings smoothed using neighboring observations. These steps ensured a clean, continuous, and time-aligned dataset.

After cleaning, temporal feature engineering was applied to capture the inherent daily, weekly, and seasonal patterns in electricity consumption. From each timestamp, multiple time-based features were extracted, including hour, day, month, day of week, week of year, and a weekend indicator. To better represent the cyclical nature of time, sine and cosine transformations were applied to hour, month, and day-of-week values. These cyclic encodings allow machine learning and deep learning models to learn smooth periodic relationships, such as the transition from late night hours to early morning, rather than treating time variables as discontinuous linear values. The resulting feature set combined both weather variables and engineered temporal components, forming a comprehensive input representation for modeling.

To understand feature relevance and interdependencies, a correlation analysis was conducted on all numerical variables. The correlation matrix, shown in Figure 3.2, indicates that hour of day has the strongest relationship with electrical load, while global solar radiation shows a moderate positive correlation. Temperature and humidity exhibit weaker but meaningful correlations, and calendar-related variables contribute subtle seasonal trends. Based on this analysis and domain knowledge, all engineered features were retained. After preprocessing, the final dataset contained no missing values, no irregular timestamps, and no half-hour entries, resulting in a fully standardized and reliable hourly time-series dataset used for both machine learning and deep learning model development.

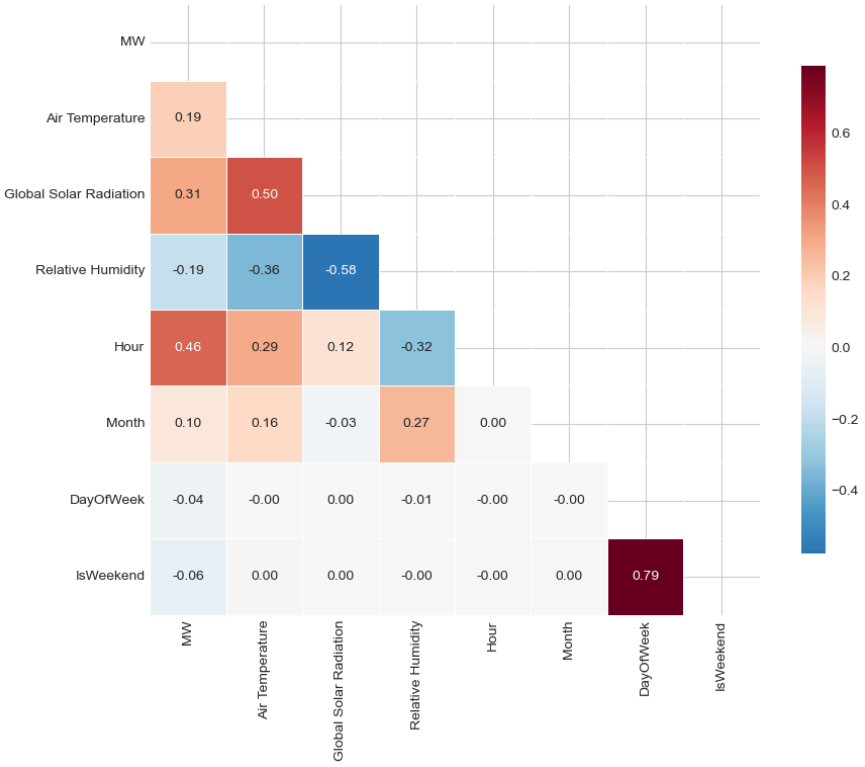


Figure 3.2: Feature Correlation Matrix

## 3.4 Model Development

Based on the theoretical foundations presented in Chapter 2, this study implemented multiple forecasting models to predict short-term electrical load for the Baneshwor Feeder. For machine learning, six models were developed: Linear Regression, Ridge Regression, Support Vector Regression (SVR), Random Forest Regressor, Gradient Boosting Regressor, and XGBoost Regressor. For deep learning, three architectures were implemented: Multi-Layer Perceptron (MLP), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU). All models were trained on the cleaned and feature-engineered dataset described in the earlier sections, with each model receiving the same input feature set to ensure fair comparison. The machine learning pipeline involved standard scaling of the numerical features, an 80–20 train–test split, and hyperparameter tuning performed using GridSearchCV, while the deep learning models were trained on sliding windows of historical hourly sequences using the Adam optimizer with early stopping to prevent overfitting. Model performance was evaluated using RMSE, MAE, MAPE, and $R^2$.

## 3.5 Model Training and Validation

This stage focuses on how both machine learning (ML) and deep learning (DL) models were trained, tuned, validated, and prepared for final performance comparison. Since ML and DL models require different handling, the training process is presented in two separate parts.

### 3.5.1 Training of Machine Learning Models

The machine learning models trained in this study include:

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 (baseline models)

All models used the same final preprocessed dataset described earlier, containing weather features, temporal encodings, and cleaned load values.

#### 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.

#### 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.

#### 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.

#### 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 $R^2$.

#### 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.

### 3.5.2 Training of Deep Learning Models

Three deep learning models were implemented:

1. Multi-Layer Perceptron (MLP)
2. Long Short-Term Memory (LSTM)
3. Gated Recurrent Unit (GRU)

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

#### 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.

### Train–Validation–Test Split

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

- 80 percent for training
- 10 percent for validation
- 10 percent for testing

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

### 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)

### Regularization and Stability

To avoid overfitting, several regularization techniques were applied:

- Dropout layers were included in the MLP and LSTM-based models
- Batch normalization was applied where appropriate
- Early stopping was used to monitor validation loss, and training automatically stopped when the validation loss stopped improving for several consecutive epochs

### Model-Specific Training Notes

- **MLP:** Used flattened inputs from the sliding window and relied on dense layers to learn nonlinear mappings. Generally converged quickly during training.
- **LSTM:** Captured long-term temporal dependencies but required more computation. Performed best when extended historical context was important.
- **GRU:** Provided a faster and lighter alternative to LSTM, often achieving comparable accuracy while offering a strong balance for medium-complexity time-series tasks.

### 3.5.3 Validation Approach

A consistent evaluation strategy was applied across all models:

**Machine Learning Models:**

- Validated using GridSearchCV with five-fold cross-validation
- Best hyperparameters were chosen based on the minimum validation loss

**Deep Learning Models:**

- Validated using a 10 percent validation split
- Early stopping was used to prevent overfitting
- Best-performing model weights were preserved through checkpointing

## 3.6 Performance Evaluation

To compare the machine learning and deep learning models fairly, a consistent set of standard error metrics was used. These metrics measure the difference between the predicted load values and the actual observed load. In this study, four commonly used regression evaluation metrics were selected:

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

These metrics help assess accuracy, robustness, and the overall predictive performance of the forecasting models.

### 3.6.1 Mean Absolute Error (MAE)

MAE represents the average magnitude of errors between forecasted and actual values, without considering direction. It is simple and easy to interpret.

**Formula:**

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.1}$$

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

**Interpretation:** Lower MAE indicates that the model's predictions are consistently closer to the actual load. This metric is robust against the influence of individual large errors.

### 3.6.2 Root Mean Squared Error (RMSE)

RMSE is one of the most widely used metrics in load forecasting, and its units are the same as the original POWER (MW) values.

**Formula:**

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.2}$$

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

**Interpretation:** Lower RMSE reflects a more accurate model overall. Because it strongly penalizes large errors, it serves as a stricter evaluation metric than MAE.

### 3.6.3 Mean Absolute Percentage Error (MAPE)

MAPE expresses the prediction error as a percentage of the actual value. This makes it easy to compare forecasting performance across different feeders or scales.

**Formula:**

$$\text{MAPE} = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{3.3}$$

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

**Interpretation:** This metric shows the average percentage deviation between predictions and actual values, with lower values indicating better performance. It becomes undefined when the actual load equals zero, though this was not an issue here since the feeder load never drops to zero.

### 3.6.4 Coefficient of Determination ($R^2$ Score)

$R^2$ indicates how much of the variance in the actual load values is explained by the model.

**Formula:**

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{3.4}$$

where $y_i$ is the actual load value, $\hat{y}_i$ is the predicted load value, and $\bar{y}$ is the mean of actual values.

**Interpretation:** An $R^2$ value of 1 represents perfect prediction, while a value of 0 indicates that the model performs no better than simply using the average load. Higher $R^2$

values therefore indicate better model performance. Negative $R^2$ values are possible when the model performs worse than the mean baseline.

# 4. Results & Discussion

This chapter presents the performance of all machine learning and deep learning models trained for short-term load forecasting of the Baneshwor Feeder. All models were evaluated using the same performance metrics (MAE, RMSE, MAPE, $R^2$), ensuring a fair comparison.

## 4.1 Exploratory Data Analysis

Exploratory data analysis was conducted to understand the statistical behavior of the load and weather variables, identify underlying temporal patterns, and observe the relationships between different features before model training. The final dataset consisted of hourly timestamps spanning the entire study period, feeder load values in megawatts, and the key weather variables of air temperature, global solar radiation, and relative humidity. It also included a comprehensive set of engineered temporal features such as hour, day, month, weekday, and the corresponding cyclical encodings. A quick inspection confirmed that there were no missing values after imputation, the dataset maintained a uniform hourly structure with no half-hour gaps, the POWER (MW) readings were clean after capping and smoothing, and all weather and load timestamps were fully synchronized.

To understand how the load varies over time, the hourly POWER (MW) values were resampled into daily averages and visualized across the entire study period. The resulting trend showed clear daily, weekly, and seasonal fluctuations in the feeder's behavior. Winter months displayed slightly lower solar radiation levels along with moderately higher load during certain intervals. Occasional dips in the curve aligned with known outages or special events. Solar radiation exhibited a strong daytime pattern, reinforcing its moderate correlation with load. Overall, this broad visualization confirmed that the Baneshwor Feeder operates as a typical mixed-load distribution feeder with pronounced daily cycles and noticeable seasonal influences.

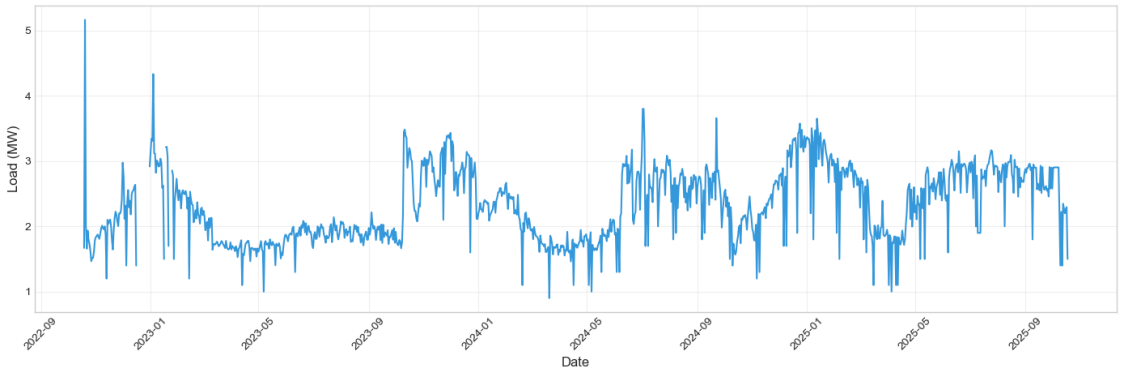


Figure 4.1: Daily Average Electricity Load Over Time

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.

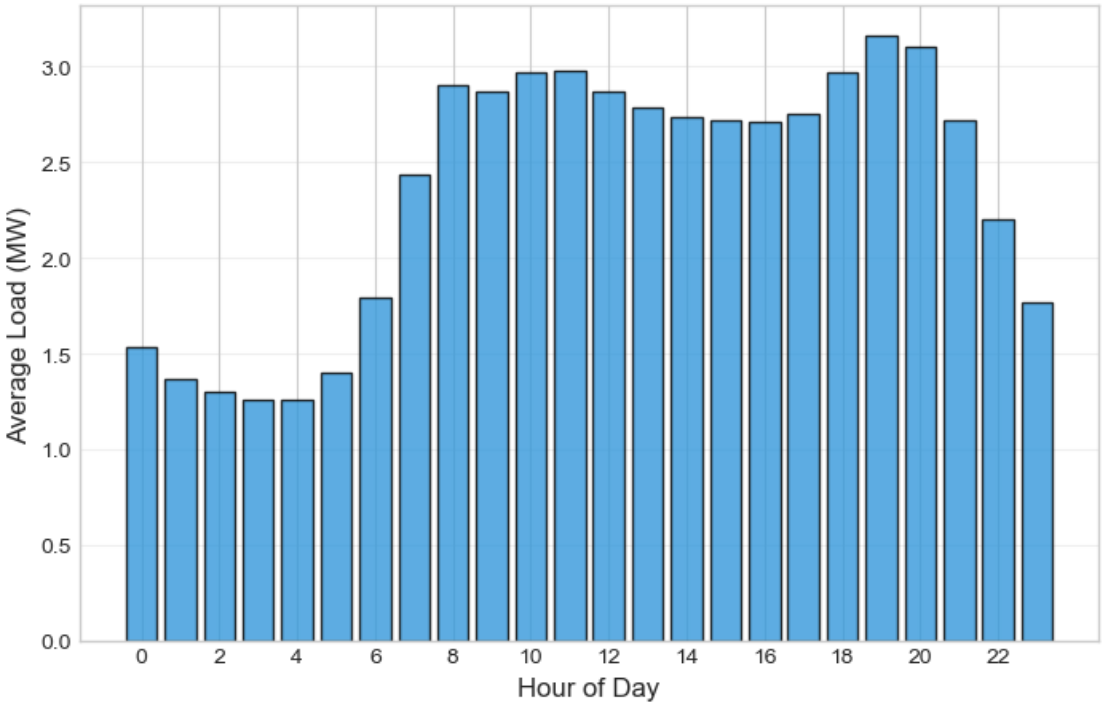


Figure 4.2: Load Distribution by Hour

Monthly averages revealed that warmer months experience higher temperatures, although the corresponding load behavior varies across the year. Seasonal patterns are present but not as dominant as the daily cycles observed in the feeder. Consumption typically increases during festival seasons when household activity rises. These seasonal shifts are effectively captured through the Month feature and its corresponding cyclical encodings.

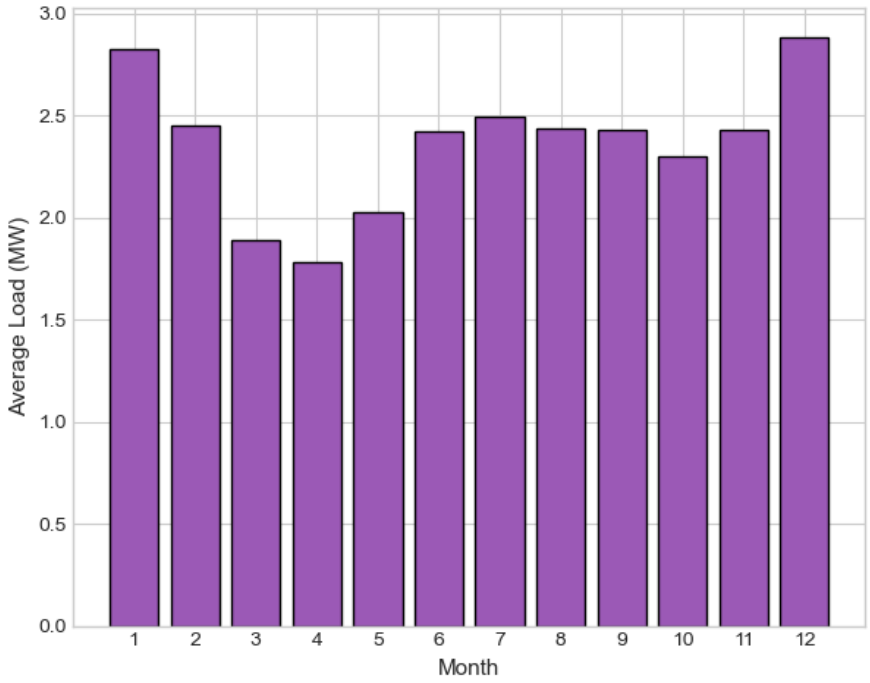


Figure 4.3: Average Load by Month

All weather variables were examined individually to understand their behavior and potential influence on load. Air temperature ranged from roughly 1°C to 33°C and followed a clear daily cycle, with warmer afternoons and cooler nights. Global solar radiation showed a distinct daytime-only pattern, peaking sharply around midday and dropping to zero during nighttime hours. Relative humidity tended to be higher during nighttime and rainy months and displayed a slight inverse relationship with temperature. The temperature–load relationship showed a weak positive correlation, with load increasing moderately as temperatures rise, which is typical for mixed-load areas where fans and cooling appliances see greater use. Solar radiation displayed a moderate positive correlation with load, as higher midday radiation often coincides with active residential and commercial activity. Humidity exhibited a weak negative correlation, since high humidity is generally associated with cloudy or rainy conditions during which daytime load may decrease slightly.

## 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 4.1 summarizes the hyperparameter search space explored for each machine learning model. The best-performing configurations were selected based on the lowest validation error.

Table 4.1: Hyperparameter Search Space for Machine Learning Models

| **Model** | **Hyperparameters** |
|---|---|
| Ridge Regression | $\alpha$ = [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 4.2 presents the performance of all machine learning models.

Table 4.2: 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 |

**Discussion of Results**

The machine learning models were evaluated using MAE, RMSE, MAPE, and R-squared ($R^2$) to assess their forecasting effectiveness. Linear Regression and Ridge Regression served as baseline models and produced relatively higher error values and lower $R^2$ scores, indicating their limited ability to capture nonlinear relationships between load demand and influencing variables. Support Vector Regression improved upon linear models by reducing error metrics; however, its performance remained inferior to ensemble-based approaches, particularly in terms of RMSE.

Tree-based ensemble models demonstrated superior performance across all evaluation metrics. Random Forest and Gradient Boosting significantly reduced MAE and RMSE while achieving higher $R^2$ values, reflecting improved generalization and better handling of nonlinear patterns. Among these, the tuned XGBoost model outperformed all other machine learning models, achieving the lowest RMSE (0.384 MW), the highest $R^2$ (0.831), and the lowest MAPE (12.693%). The improvement over other ensemble models can be attributed to XGBoost's gradient-based optimization, regularization mechanisms, and effective handling of feature interactions, which collectively enhanced model robustness.

Overall, the comparative evaluation confirms that ensemble-based machine learning models, particularly XGBoost, provide the most reliable and accurate predictions for feeder-level short-term load forecasting.

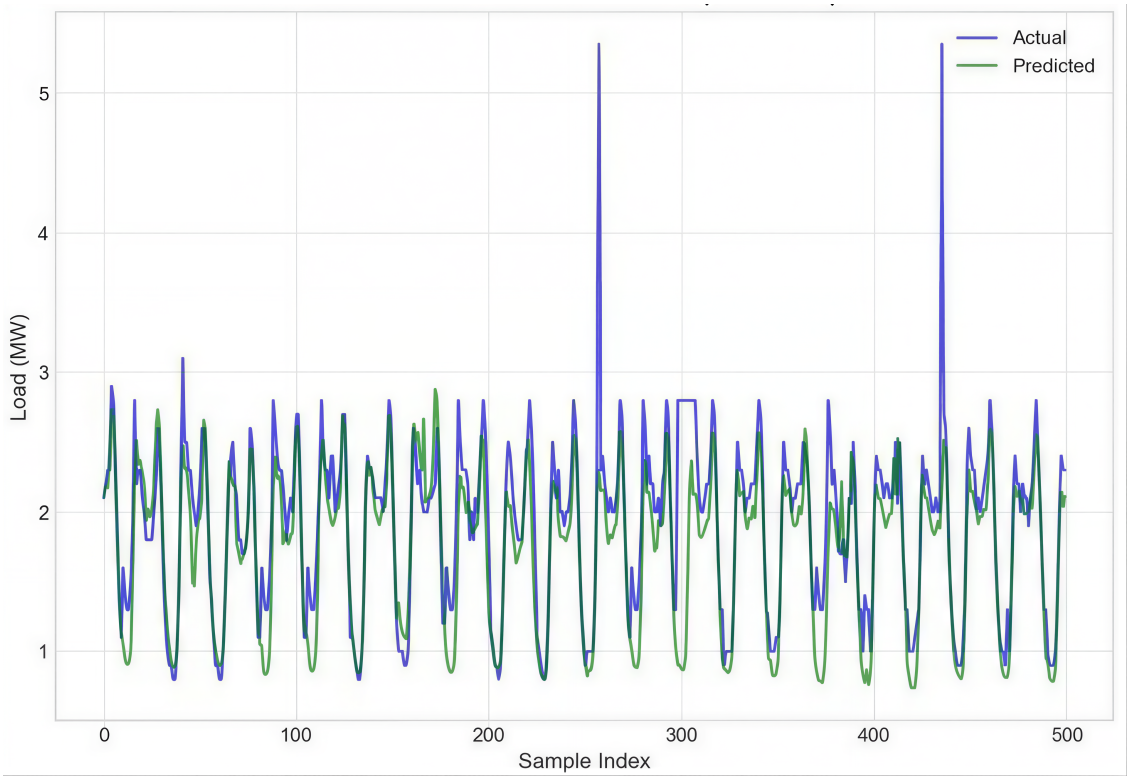


Figure 4.4: XBoost (Tuned) Actual vs Predicted

### 4.2.2 Deep Learning Model Performance

Deep learning models were trained using sliding-window sequences to capture temporal dependencies in the feeder load data. Three architectures were implemented: LSTM, GRU, and MLP. Hyperparameter optimization was performed using an iterative search strategy, with configurations evaluated based on validation loss. Table 4.3 presents the hyperparameter search space explored for the deep learning models.

Table 4.3: Hyperparameter Search Space for Deep Learning Models

| Component | Values |
|---|---|
| **MLP / LSTM / GRU Components** | |
| Number of layers | [2, 3, 4] |
| Hidden units | [32, 64, 128] |
| Dropout rate | [0.0, 0.1, 0.2] |
| Activation function | ReLU |
| **Sequence Modeling Parameters** | |
| Look-back window | [10, 20, 30] |
| Batch size | [4, 8, 16] |
| **Training Parameters** | |
| Learning rate | [0.001, 0.0001] |
| Optimizer | Adam |
| Epochs per trial | 30 |

After conducting extensive training using the cleaned dataset, the models were evaluated using the same metrics as the ML models. The results obtained for the deep learning models are presented in Table 4.4.

Table 4.4: Deep Learning Models Evaluation Matrix

| Sn.No. | Model | MAE | RMSE | MAPE | $R^2$ |
|---|---|---|---|---|---|
| 1 | LSTM | 0.467 | 0.498 | 14.155 | 0.014 |
| 2 | GRU | 0.482 | 0.499 | 15.186 | 0.012 |
| 3 | MLP | 0.618 | 0.499 | 20.062 | 0.011 |

### Discussion of Results

Deep learning models, including MLP, LSTM, and GRU, were evaluated using the same performance metrics for a fair comparison. Among these models, the LSTM achieved the best performance with the lowest MAE (0.467) and MAPE (14.155%), followed closely by GRU. The MLP produced the highest error values among the three architectures, with an MAE of 0.618 and MAPE of 20.062%. All three models exhibited $R^2$ scores close to zero, indicating weak explanatory power and limited ability to capture variance in the load data.

Both LSTM and GRU exhibited relatively high MAE and RMSE values compared to the machine learning models, with RMSE remaining close to 0.499 MW for all three architectures. These results suggest that the deep learning models were unable to effectively learn

stable temporal patterns from the available dataset. Compared to machine learning models, deep learning approaches showed limited generalization and higher sensitivity to data volume and structure. The lack of performance gains over simpler ML architectures indicates that increased model complexity did not translate into improved forecasting performance in this case.

In comparison to machine learning models, deep learning models were clearly outperformed across all evaluation metrics. This highlights that, for the given data scale, feature representation, and forecasting horizon, traditional machine learning models with explicit feature engineering are more suitable than deep learning architectures for feeder-level short-term load forecasting.

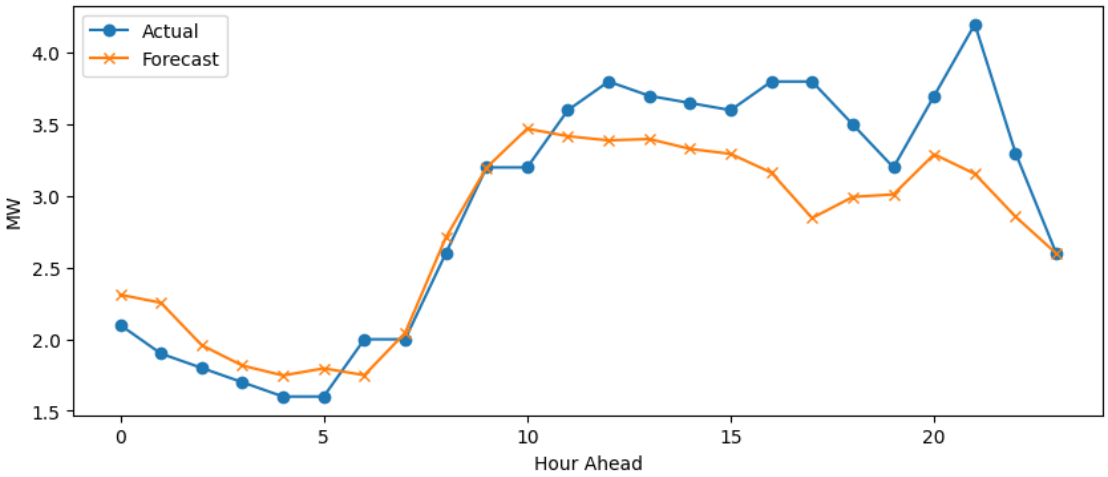


Figure 4.5: LSTM Actual vs Predicted

# 5. Conclusion

## 5.1 Conclusion

This thesis developed and evaluated a comprehensive short-term electrical load forecasting framework for the Baneshwor Feeder using machine learning and deep learning approaches. The study followed a systematic methodology involving data acquisition, preprocessing, temporal feature engineering, and comparative evaluation of multiple forecasting models under a unified framework.

Experimental results show that machine learning models, particularly ensemble-based methods, significantly outperform deep learning architectures for the given dataset. Among all evaluated models, the tuned XGBoost model achieved the best performance, with an RMSE of 0.384 MW and an $R^2$ value of 0.831, demonstrating its strong capability to model nonlinear and weather-dependent load behavior. Other tree-based models, such as Random Forest and Gradient Boosting, also produced competitive results, further confirming the effectiveness of ensemble learning techniques for feeder-level load forecasting.

In contrast, deep learning models including LSTM, GRU, and MLP did not achieve comparable performance. Their high error values and near-zero $R^2$ scores indicate limitations in learning meaningful temporal relationships from the available data, suggesting that deep learning models may require larger datasets or different input structures to be effective in this context.

Overall, this study confirms that accurate feeder-level short-term load forecasting can be achieved through a carefully designed machine learning pipeline. The findings emphasize that model selection should be guided by data characteristics, feature availability, and operational requirements rather than model complexity alone.

## 5.2 Research Limitations

Although the study achieved strong forecasting accuracy, several limitations should be acknowledged:

1. **Limited dataset size:** Deep learning models typically require long historical sequences and larger datasets. After preprocessing and windowing, the amount of usable sequential data became relatively small.

2. **Irregular load behaviour:** Feeder-level load profiles often contain noise, outages,

fluctuations, and sudden spikes, which are difficult for DL models to learn without additional contextual variables.

3. **Short sequence window:** The sliding window used for DL models may not fully capture weekly or monthly patterns that influence feeder demand.

4. **Weather data resolution:** Weather data was available at hourly intervals only; finer granularity or additional environmental factors (e.g., wind speed, rainfall intensity) could further improve models.

5. **No real-time deployment environment:** The study focused on model development and evaluation, and did not include live deployment, automation, or integration with NEA's operational systems.

These limitations provide important context when interpreting results and designing future enhancements.

## 5.3 Implications

The findings of this thesis carry several meaningful implications for utilities, researchers, and system planners:

1. **Practical adoption for distribution feeders:** Ensemble machine learning models—particularly XGBoost—can provide accurate, low-error forecasts suitable for operational planning, peak management, and scheduling.

2. **Value of feature engineering:** Carefully constructed temporal and weather features significantly improved forecasting accuracy, highlighting the importance of domain knowledge in model design.

3. **Limitations of DL for smaller datasets:** The underperformance of LSTM and GRU models reinforces that deep learning is not always the optimal choice, especially when data is limited, noisy, or discontinuous.

4. **Foundation for advanced decision-making tools:** Forecasts from XGBoost can support demand-side management, smart grid optimization, load shifting strategies, and distributed energy resource planning.

5. **Transferability:** The pipeline developed in this study can be extended to other NEA feeders with minimal modifications, promoting scalable forecasting across the network.

## 5.4 Future Work

Although this study demonstrates effective short-term load forecasting at the feeder level using machine learning models, several extensions can be explored to further enhance forecasting accuracy and practical applicability.

1. **Incorporation of Additional Influencing Factors:** Future work may include additional exogenous variables such as holiday indicators, special events, and socio-economic factors to better capture demand variations that are not explained by weather and temporal features alone.

2. **Extension to Multi-Feeder and Long-Term Forecasting:** The proposed methodology can be extended to multiple feeders and adapted for medium-term and long-term load forecasting to support broader power system planning and expansion studies.

3. **Improved Deep Learning Architectures and Data Volume:** Deep learning models such as LSTM and GRU may benefit from larger datasets, finer temporal resolution, and advanced architectures—including attention-based or hybrid ML–DL models—to improve their forecasting capability.

4. **Real-Time Deployment and Online Learning:** Future research can focus on deploying the forecasting model in a real-time operational environment, incorporating online or incremental learning techniques to adapt to evolving load patterns.

# References


[1] M. K. Singla, J. Gupta, P. Nijhawan, and A. S. Oberoi. Electrical load forecasting using machine learning. *International Journal*, 8(3), 2019.

[2] S. Desai, T. Dalal, S. Kadam, and S. Mishra. Electrical load forecasting using machine learning. In *2021 International Conference on System, Computation, Automation and Networking (ICSCAN)*, pages 1–6, 2021.

[3] P. Matrenin, M. Safaraliev, S. Dmitriev, S. Kokin, A. Ghulomzoda, and S. Mitrofanov. Medium-term load forecasting in isolated power systems based on ensemble machine learning models. *Energy Reports*, 8:612–618, 2022.

[4] E. Aguilar Madrid and N. Antonio. Short-term electricity load forecasting with machine learning. *Information*, 12(2), 2021.

[5] W. Guo, L. Che, M. Shahidehpour, and X. Wan. Machine learning-based methods in short-term load forecasting. *The Electricity Journal*, 34(1):106884, 2021.

[6] M. Saglam, X. Lv, C. Spataru, and O. A. Karaman. Instantaneous electricity peak load forecasting using optimization and machine learning. *Energies*, 17(4), 2024.

[7] S. Jain and A. Gupta. Comparative analysis of machine learning algorithms for short-term power load prediction. *International Journal of Energy Systems*, 19(2):55–68, 2024.

[8] K. Chapagain, S. Acharya, H. Bhusal, S. Katuwal, O. Lakhey, P. Neupane, R. K. Sah, B. Tamang, and Y. Rajbhandari. Short-term electricity demand forecasting for kathmandu valley, nepal. *Kathmandu University Journal of Science, Engineering and Technology*, 15(3), 2021.

[9] S. Acharya, M. C. Luintel, and M. Badrudoza. Short-term load forecasting of gothatar feeder of nepal electricity authority using recurrent neural network. *Unpublished manuscript*, n.d.

[10] M. Cordeiro-Costas, D. Villanueva, P. Eguía-Oller, M. Martínez-Comesaña, and S. Ramos. Load forecasting with machine learning and deep learning methods. *Applied Sciences*, 13(13):7933, 2023.

[11] X. Dong, Z. Li, L. Sun, and Y. Zhang. A decade of deep learning-based short-term electricity load forecasting: A comprehensive review. *Energy AI*, 8:100212, 2024.

[12] J. Wen, H. Li, and T. Zhao. Gru–tcn hybrid neural network with attention for short-term load forecasting. *Energy*, 290:130423, 2024.

[13] O. Alhussein, K. Aurangzeb, S.-B. Kim, and J.-H. Baek. Convolutional neural network and long short-term memory hybrid deep learning model for individual load forecasting. *Energies*, 13(19), 2020.

[14] H. Hasanat, A. Biswas, and M. Abdullah. Parallel multichannel cnn–bilstm architecture for smart-grid load forecasting. *IEEE Access*, 12:33211–33225, 2024.

[15] K. Chan and C. Yeo. Sparse transformer-based architecture for electricity load forecasting. *Applied Energy*, 352:122211, 2024.

[16] Y. Zhang, Q. Li, and M. Chen. Time-augmented transformer for short-term electrical load forecasting. *IEEE Transactions on Power Systems*, 37(6):5214–5225, 2022.

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

[18] S. Banik and S. Biswas. A stacked ensemble learning model for renewable power and load forecasting. *Energy Reports*, 10:112–123, 2024.

[19] S. Kwon, K. Lee, and J. Park. Electricity load forecasting using clustering and dimensionality reduction with advanced neural models. *IEEE Transactions on Smart Grid*, 11(5):3980–3992, 2020.

[20] S. Nabavi, A. Koohi, and M. Rahmani. Wavelet-transform-enhanced lstm for short-term electricity load forecasting. *Energy and Buildings*, 293:113350, 2024.

[21] Arthur E. Hoerl and Robert W. Kennard. *Ridge regression: Biased estimation for nonorthogonal problems*, volume 12. Taylor & Francis, 1970.

[22] Harris Drucker, Christopher J. C. Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. *Advances in Neural Information Processing Systems*, 9:155–161, 1997.

[23] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[24] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

[25] 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*, pages 785–794. ACM, 2016.

[26] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[27] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[28] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.