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. Deep learning models including Multi-Layer Perceptron (MLP), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) were also developed and evaluated. Hyperparameter tuning and data augmentation techniques were 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 the MLP deep learning model with extended lag features achieved the best overall performance (RMSE: 0.242 MW, R^2: 0.915), followed by ensemble-based machine learning models, particularly tuned XGBoost (RMSE: 0.384 MW, R^2: 0.831). The findings highlight the effectiveness of appropriately configured neural networks and 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, deep learning, MLP, XGBoost, 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 thesis investigates the application of data-driven forecasting methodologies for predicting hourly electrical demand at the distribution feeder level. By integrating historical consumption records with meteorological observations and temporal characteristics, multiple predictive models were constructed and systematically compared to determine the most suitable approach for the Baneshwor Feeder's operational requirements.

1.1 Background

Electrical power consumption exhibits inherent variability driven by human activities, climatic conditions, and economic cycles. The ability to anticipate these fluctuations, even within a short window of several hours, provides grid operators with significant advantages in resource allocation, cost minimization, and service reliability [1].

Short-Term Load Forecasting (STLF) addresses prediction horizons spanning from one hour to approximately 24 hours ahead [2]. Such forecasts directly support critical operational decisions including generation scheduling, reserve allocation, and real-time balancing. Classical statistical methods—including autoregressive integrated moving average (ARIMA), exponential smoothing variants, and linear regression—have historically served as the foundation for utility forecasting practices [3]. However, these conventional approaches often prove inadequate when confronted with the complex, nonlinear interdependencies present in modern distribution networks.

Contemporary machine learning algorithms—including Random Forest, Support Vector Regression, and gradient boosting variants such as XGBoost—have demonstrated considerable success in energy system applications [4, 5]. These methods excel at modeling the nonlinear mappings between input features and target variables that characterize electrical demand patterns. Similarly, deep neural network architectures, particularly recurrent designs like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), possess inherent capabilities for capturing sequential dependencies across time steps [6].

The Baneshwor Feeder, operated under Nepal Electricity Authority jurisdiction, supplies a heterogeneous consumer base comprising residential households, commercial establishments, and institutional loads within the Baneshwor area of Kathmandu Valley. The consumption profile exhibits pronounced diurnal and hebdomadal periodicity, modulated by ambient temperature variations and seasonal activity patterns. Nevertheless, irregular demand spikes and anomalous consumption events introduce complexities that elementary

forecasting techniques cannot adequately address. Given the expanding electricity demand and increasing consumer diversity in this region, developing robust predictive capabilities for this specific feeder presents both practical necessity and academic interest.

Figure 1.1: Baneshwor Feeder View

1.2 Problem Statement

Existing demand anticipation practices at the Baneshwor Feeder predominantly depend on operator experience and rudimentary statistical calculations. Such approaches inadequately represent the intricate, nonlinear characteristics of the feeder's consumption profile, particularly when numerous concurrent factors—including ambient temperature, atmospheric moisture, precipitation events, weekly patterns, and public holidays—simultaneously influence demand levels. Consequently, forecasting accuracy deteriorates notably during peak consumption intervals, abrupt meteorological shifts, and inter-seasonal transition periods.

Suboptimal demand predictions carry substantial operational implications. Generation scheduling inefficiencies may result in either excessive spinning reserves or insufficient supply margins. Distribution-level technical losses and operational expenditures may escalate unnecessarily. Under severe conditions, inadequate demand foresight during high-consumption

periods can precipitate voltage instability, service reliability degradation, or suboptimal load curtailment decisions.

Notwithstanding the existence of archived consumption records and meteorological observations, no comprehensive investigation has systematically implemented and benchmarked contemporary machine learning and deep learning methodologies for the Baneshwor Feeder specifically. This absence of an intelligent, data-driven forecasting infrastructure prevents distribution operators from leveraging models capable of discerning complex multivariate relationships embedded within historical observations.

This research endeavors to bridge these deficiencies by constructing a complete predictive framework encompassing multiple ML and DL algorithms, rigorously evaluating their comparative performance, and recommending the most appropriate methodology for reliable short-term demand forecasting at the Baneshwor Feeder.

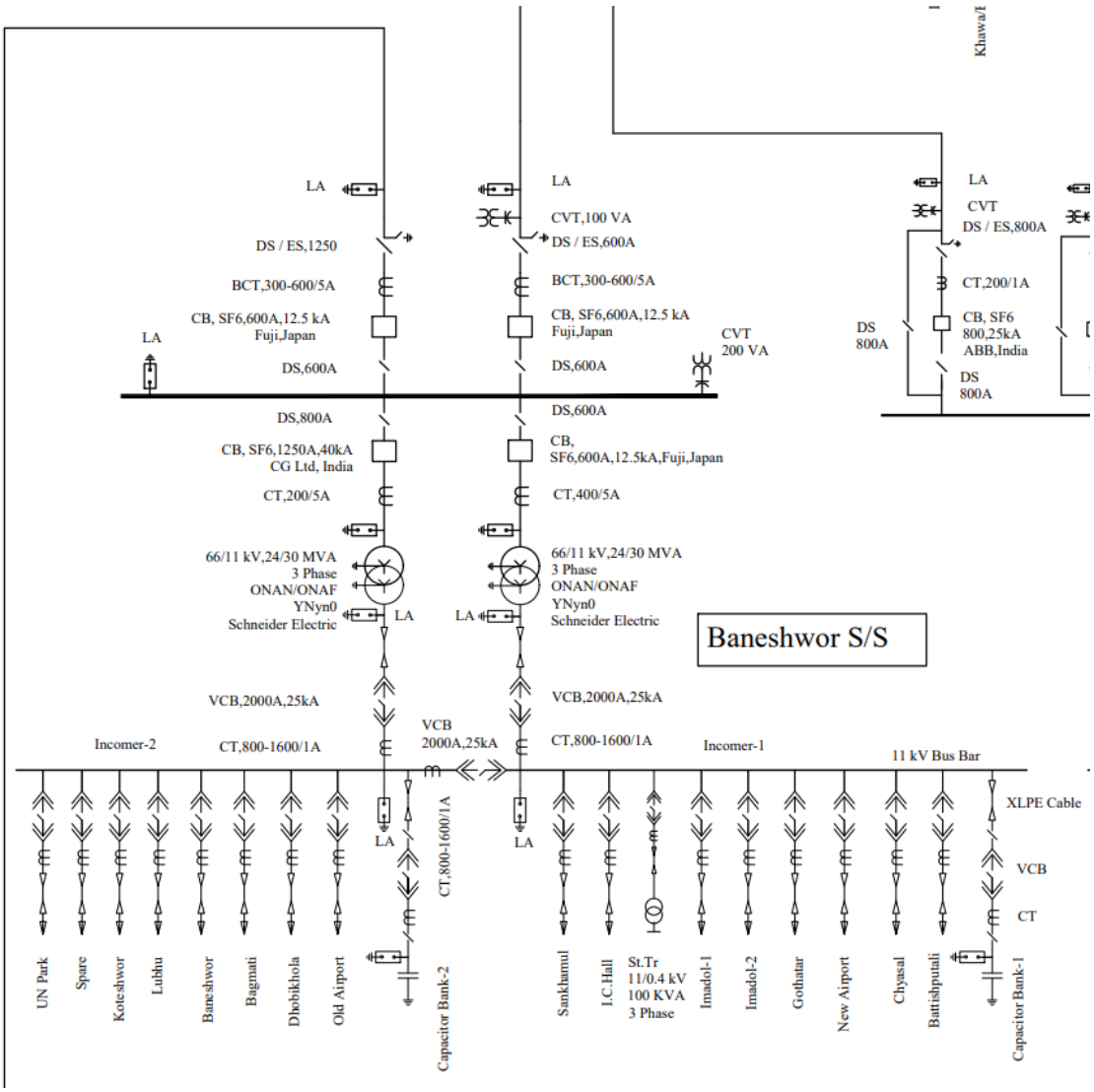

Figure 1.2: Substation & Transmission Line Network Baneshwor

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

The geographical boundary of this investigation is circumscribed to the Baneshwor Feeder administered by Nepal Electricity Authority. Temporally, the research concentrates on near-term demand anticipation with forecast horizons extending to 24 hours, leveraging archived hourly consumption records as the foundational modeling substrate. The assembled dataset integrates historical feeder demand observations alongside meteorological measurements encompassing temperature, humidity, and solar irradiance, complemented by calendrical indicators differentiating standard weekdays from weekends and public holidays.

From a computational standpoint, this research deploys multiple machine learning implementations including Support Vector Regression, Random Forest ensembles, and XGBoost, alongside deep neural architectures specifically Long Short-Term Memory and Gated Recurrent Unit networks. Predictive performance undergoes rigorous quantification through established statistical measures comprising Root Mean Squared Error, Mean Absolute Percentage Error, Mean Absolute Error, and coefficient of determination (R-squared). It bears noting that forecast precision inherently correlates with underlying data integrity and completeness. Furthermore, this investigation expressly excludes medium-horizon and long-horizon forecasting scenarios, while renewable generation prediction falls outside the defined research boundaries.

1.5 Limitation

Notwithstanding the encouraging outcomes achieved, this investigation encountered specific constraints primarily associated with data accessibility, algorithmic assumptions, and analytical scope.

1. **Data quality dependence:** Predictive accuracy remains fundamentally bounded by the integrity and completeness of archived consumption and meteorological records. Sensor malfunctions, missing entries, or inconsistent reporting practices can adversely influence model performance.

2. **Sensitivity to abrupt transitions:** Unforeseen occurrences including supply disruptions, festival periods, sudden climatic variations, or atypical consumption behaviors present inherent challenges for purely data-driven predictive approaches.

3. **Deep learning computational demands:** LSTM and GRU network training necessitates considerably greater processing resources and temporal investment relative to conventional ML algorithms. Performance outcomes may exhibit variability contingent upon available computational infrastructure.

4. **Restricted feature diversity:** Although meteorological and calendrical variables were incorporated, additional potentially influential factors—including macroeconomic indicators, special event schedules, or industrial consumption profiles—remain absent from the dataset.

5. **Cross-feeder transferability constraints:** The predictive models constructed herein are calibrated exclusively for Baneshwor Feeder characteristics and may not generalize to alternative feeder configurations without substantial recalibration or architectural modification.

2. Literature Review

This chapter synthesizes relevant scholarly contributions pertaining to electrical demand prediction through computational intelligence approaches. The review systematically examines published investigations to characterize prevalent algorithmic frameworks, dataset configurations, and assessment protocols employed within power system forecasting research. Critical analysis of antecedent work facilitates identification of contemporary methodological trajectories, inherent model strengths and constraints, and unexplored research opportunities that substantiate the present investigation's rationale. Positioning this thesis within the broader academic discourse establishes the theoretical underpinning for subsequent algorithmic selection and experimental design decisions.

2.1 Related Work

Substantial research attention has been directed toward electrical demand forecasting across various temporal horizons, encompassing short-term predictions spanning hours to days, medium-term forecasts extending weeks to months, and long-term projections covering years. The preponderance of scholarly effort has concentrated on short-term forecasting applications given their direct operational relevance.

2.1.1 Machine Learning Approaches

Singla et al. [7] 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. [8] 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. [4] 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 [5] 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. [9] 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. [10] 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 [11] 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. [2] 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. [3] 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. [6] 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. [1] 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 op-

timization, 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 elucidates the mathematical foundations and operational principles underlying the computational models deployed throughout this investigation. Comprehending these

theoretical constructs proves indispensable for meaningful interpretation of model behavior and forecasting outcomes within the distribution system context.

2.2.1 Machine Learning Models

Linear Regression

Linear Regression constitutes the most elementary predictive framework, postulating a strictly linear mapping between predictor variables and the response quantity. Despite the inherent nonlinearity characterizing electrical consumption phenomena, this model furnishes a baseline reference against which more sophisticated approaches can be quantitatively benchmarked.

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 is the intercept (bias term), β_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 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.

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, ε defines the epsilon-insensitive tube, and ξ_i and ξ_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 ν 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, γ controls the minimum loss reduction required to make a split, and λ is the L2 regularization term on leaf weights.

2.2.2 Deep Learning Models

To adequately represent the nonlinear dynamics, temporal evolution, and sequential structure inherent in feeder consumption data, three neural network architectures were constructed and evaluated: Multi-Layer Perceptron (MLP), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU). These networks received training on sliding temporal windows comprising consecutive hourly observations, enabling the architectures to assimilate both proximate and extended temporal dependencies embedded within the dataset. Network optimization employed the Adam algorithm [26], with early termination protocols preventing overtraining and sequence-structured inputs where architecturally appropriate.

Multi-Layer Perceptron (MLP)

The MLP [27] 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 σ 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.

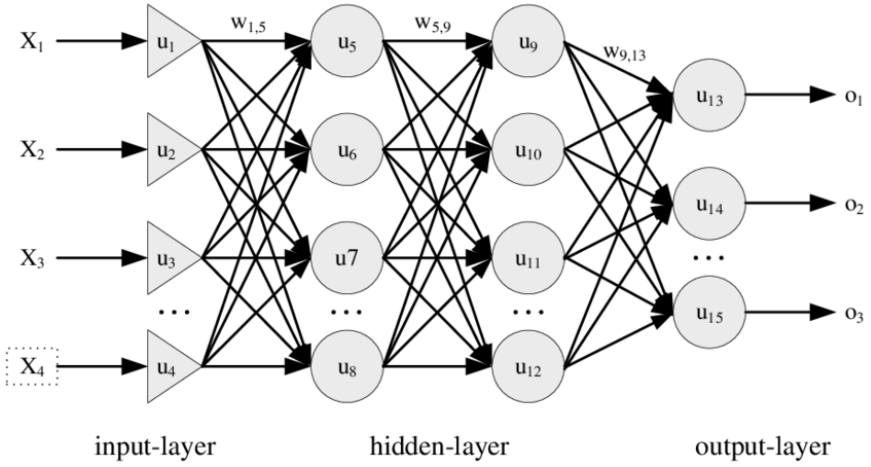

Figure 2.1: Architecture MLP

Long Short-Term Memory (LSTM)

LSTM networks [28] 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 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.

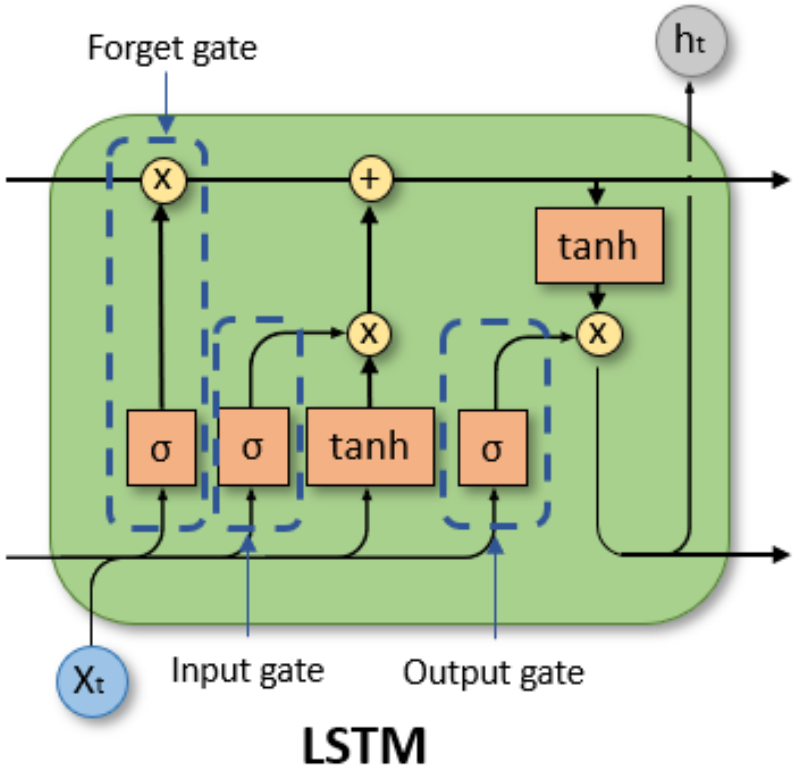

Figure 2.2: Architecture LSTM

Gated Recurrent Unit (GRU)

GRU [29] 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 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.

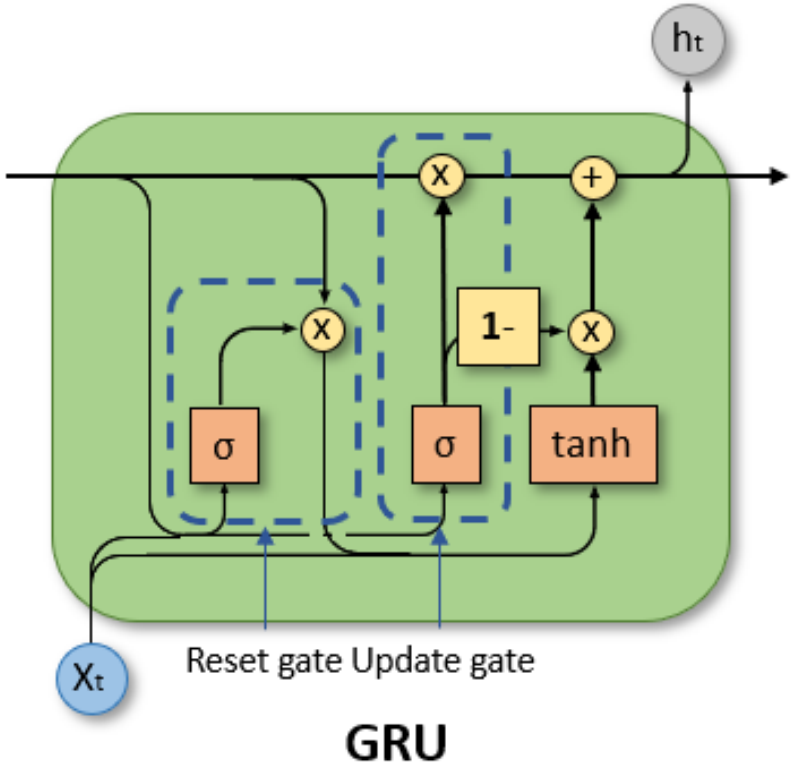

Figure 2.3: Architecture GRU

3. Methodology

This chapter delineates the comprehensive research framework implemented to accomplish the stated thesis objectives. The presentation encompasses experimental design rationale, data procurement procedures, preprocessing protocols, and predictive modeling strategies deployed for feeder-level demand forecasting. The methodological structure ensures rigorous analytical progression, with each procedural component maintaining explicit linkage to the defined research goals. A schematic diagram encapsulates the overarching workflow, supplemented by detailed exposition of the chosen machine learning and deep learning algorithms alongside the quantitative assessment techniques applied throughout this investigation.

3.1 Overall Workflow

The predictive modeling pipeline employed in this thesis adheres to a methodical, stage-wise progression. Raw hourly demand recordings from the Baneshwor Feeder are aggregated with concurrent meteorological observations (temperature, humidity, precipitation) and temporal markers (weekday/weekend designations, holiday flags) to constitute the comprehensive input feature space. These unprocessed records initially undergo extensive quality assurance procedures addressing missing entries, anomalous values, and formatting discrepancies through systematic cleaning, imputation algorithms, outlier remediation, timestamp normalization, and derivation of temporal and cyclical feature representations—thereby generating analysis-ready datasets. Subsequent exploratory analysis interrogates consumption trends, periodic patterns, hourly variation structures, and weather-demand correlations to identify salient predictive features and inform variable selection decisions. Following these preparatory stages, both traditional ML algorithms and deep neural architectures are instantiated, with input features undergoing standardization and organization as tabular matrices for ML implementations while deep architectures receive sequentially-structured inputs. Model calibration proceeds on designated training partitions with validation through walk-forward protocols or holdout assessment, while hyperparameter optimization employs GridSearchCV for ML algorithms and iterative refinement strategies for neural networks to enhance generalization while mitigating overfitting tendencies. Ultimately, all candidate models undergo comparative evaluation using RMSE, MAE, MAPE, and R^2 metrics, with predictive accuracy, stability characteristics, and computational efficiency jointly analyzed to recommend optimal forecasting solutions for Baneshwor Feeder operational deployment. This workflow thereby establishes a complete analytical pipeline spanning initial data procurement through

final model recommendation, accommodating both conventional and neural network methodologies. Figure 3.1 illustrates the complete methodological structure.

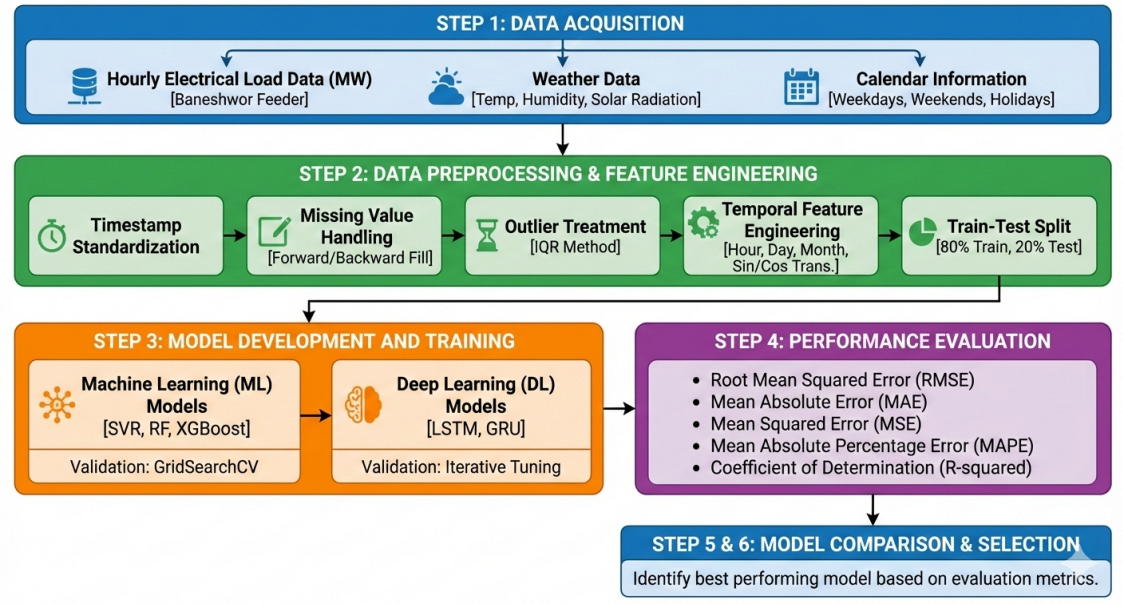

Figure 3.1: Methodology Block Diagram

3.2 Data Acquisition

The initial methodological phase involves systematic data procurement. Archived hourly consumption records from the Baneshwor Feeder constitute the primary dataset. Supplementary meteorological observations—encompassing ambient temperature, atmospheric moisture content, and precipitation measurements—originate from the Department of Hydrology and Meteorology, Nepal. Additionally, weekday and weekend classifications derive from officially published governmental calendrical sources. The investigative framework relies upon dual primary data streams:

1. Hourly electrical consumption measurements for the Baneshwor Feeder
2. Concurrent hourly meteorological recordings (temperature, humidity, incident solar radiation)

Given that both datasets arrived as unprocessed Excel workbooks exhibiting irregular structural formats, inconsistent temporal indexing, incomplete entries, and multiple worksheets per temporal unit, a comprehensive multi-stage acquisition and restructuring protocol was necessitated. The provenance of both datasets is documented subsequently.

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.

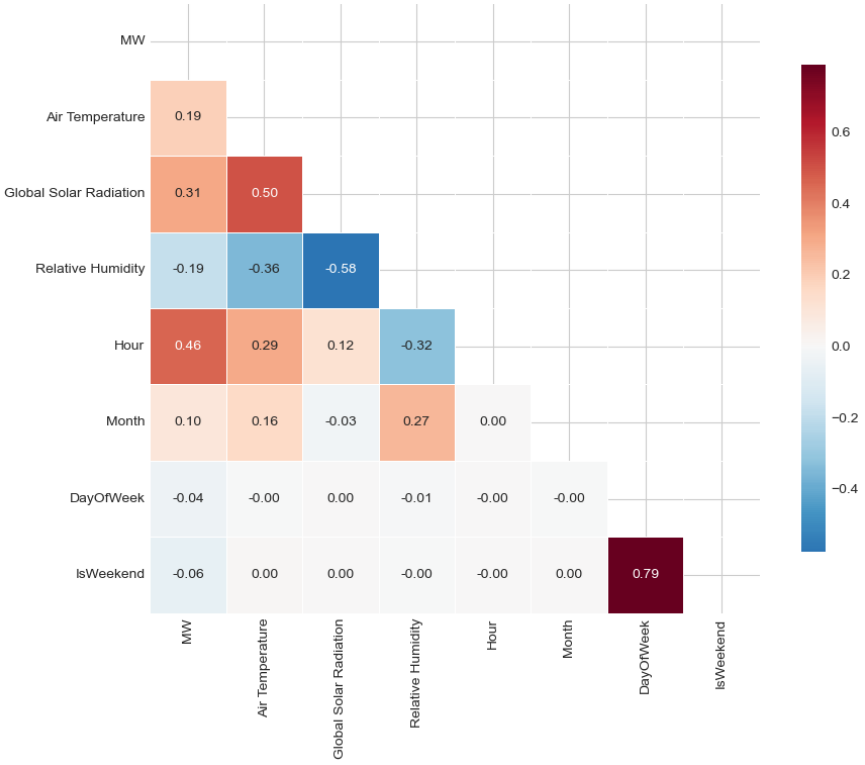

Figure 3.2: Feature Correlation Matrix

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 ear-

lier 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:

- 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.

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, particularly for the MLP model, by providing explicit historical context at various temporal resolutions.

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.

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 MLP model.

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:** Received flattened inputs incorporating lag features and temporal encodings. Achieved exceptional performance with the long lag configuration [1, 3, 6, 12, 24, 48], attaining R^2 of 0.915. Training converged efficiently, typically within 30–50 epochs before early stopping.
- **LSTM:** Processed sequential inputs with a 24-hour lookback window. Achieved moderate performance with RMSE of 0.498 and R^2 of 0.014, demonstrating limited ability to capture complex temporal patterns compared to the explicitly engineered lag features used by MLP.
- **GRU:** Provided a computationally lighter alternative to LSTM with comparable performance, achieving RMSE of 0.499 and R^2 of 0.012. Both recurrent architectures showed similar limitations in capturing the feeder's consumption dynamics.

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 15 percent validation split (chronologically ordered)
- Early stopping was used to prevent overfitting, with patience of approximately 10 epochs
- Best-performing model weights were preserved through checkpointing
- Multiple lag configurations and scaling methods were systematically compared

3.6 Performance Evaluation

To facilitate equitable comparison across machine learning and deep neural network implementations, a standardized battery of statistical accuracy measures was employed throughout. These metrics quantify the magnitude of discrepancies between model-generated predictions and empirically observed consumption values. This investigation adopted four conventionally utilized regression assessment criteria:

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 statistical measures collectively characterize predictive accuracy, algorithmic robustness, and overall forecasting efficacy across the candidate models.

3.6.1 Mean Absolute Error (MAE)

MAE quantifies the average magnitude of prediction deviations from observed values, disregarding error directionality. This metric offers straightforward interpretability and practical utility.

Formula:

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

where n denotes the sample count, y_i represents the measured consumption value, and $\hat{y}_i$ indicates the corresponding model prediction.

Interpretation: Diminished MAE values signify consistently accurate predictions approaching actual consumption levels. This measure demonstrates resilience against distortion from sporadic large-magnitude errors.

3.6.2 Root Mean Squared Error (RMSE)

RMSE constitutes among the most prevalent metrics in demand forecasting applications, preserving measurement units identical to the original consumption values (MW).

Formula:

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

where n represents the observation count, y_i denotes actual consumption, and $\hat{y}_i$ signifies the predicted value.

Interpretation: Reduced RMSE values indicate superior overall predictive accuracy. The quadratic error term imposes heavier penalties on substantial deviations, rendering RMSE a more stringent criterion than MAE for model assessment.

3.6.3 Mean Absolute Percentage Error (MAPE)

MAPE articulates prediction errors as proportional deviations from actual observations, facilitating performance comparisons across disparate feeders or demand 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 represents the sample size, y_i indicates measured consumption, and $\hat{y}_i$ denotes the model forecast.

Interpretation: This metric expresses average percentage deviation between predictions and observations, with lower values indicating enhanced performance. The measure becomes mathematically undefined when actual consumption equals zero, though this scenario never materialized given the feeder's continuous operation.

3.6.4 Coefficient of Determination (R^2 Score)

R^2 quantifies the proportion of target variable variance that the predictive model successfully captures and explains.

Formula:

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

where y_i represents measured consumption, $\hat{y}_i$ denotes the predicted quantity, and $\bar{y}$ indicates the arithmetic mean of observations.

Interpretation: An R^2 coefficient of unity signifies flawless prediction capability, whereas zero indicates performance equivalent to naive mean-based forecasting. Elevated R^2 values consequently reflect superior model efficacy. Negative coefficients remain theoretically possible when model predictions underperform relative to simple averaging.

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

Preliminary statistical examination was undertaken to characterize the distributional properties of consumption and meteorological variables, discern latent temporal structures, and quantify inter-feature associations prior to model construction. The consolidated dataset encompassed hourly timestamps covering the complete observation period, feeder demand measurements expressed in megawatts, and primary meteorological variables comprising ambient temperature, incident solar irradiance, and atmospheric relative humidity. Additionally, an extensive suite of derived temporal descriptors—including hour index, calendar day, month indicator, weekday designation, and their corresponding sinusoidal transformations—augmented the feature space. Verification procedures confirmed successful missing value remediation, consistent hourly temporal resolution without intermediate gaps, cleaned consumption readings following outlier treatment, and precise chronological alignment between demand and weather observations.

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.

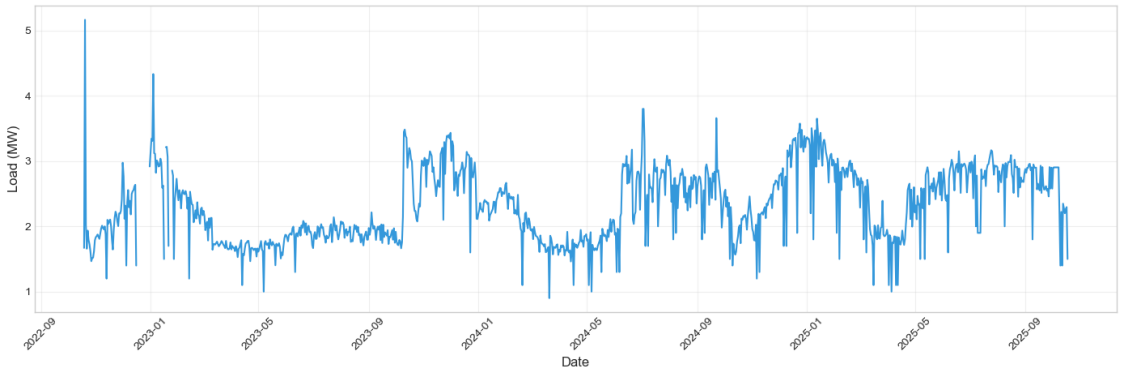

Figure 4.1: Daily Average Electricity Load Over Time

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.

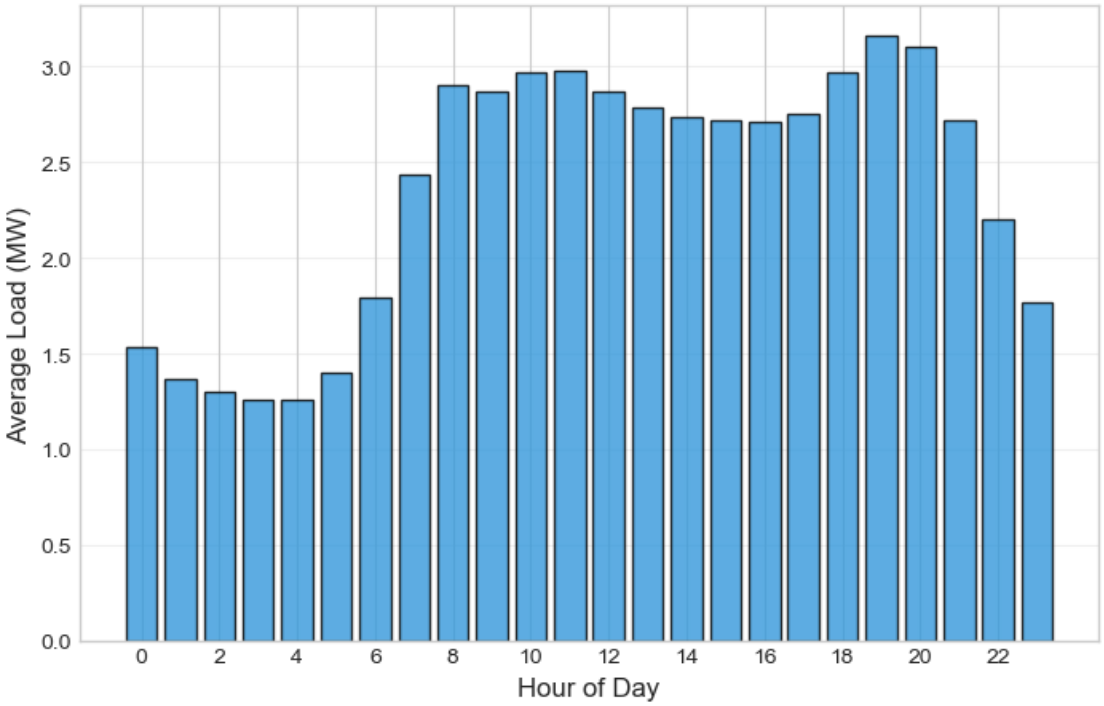

Figure 4.2: Load Distribution by Hour

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.

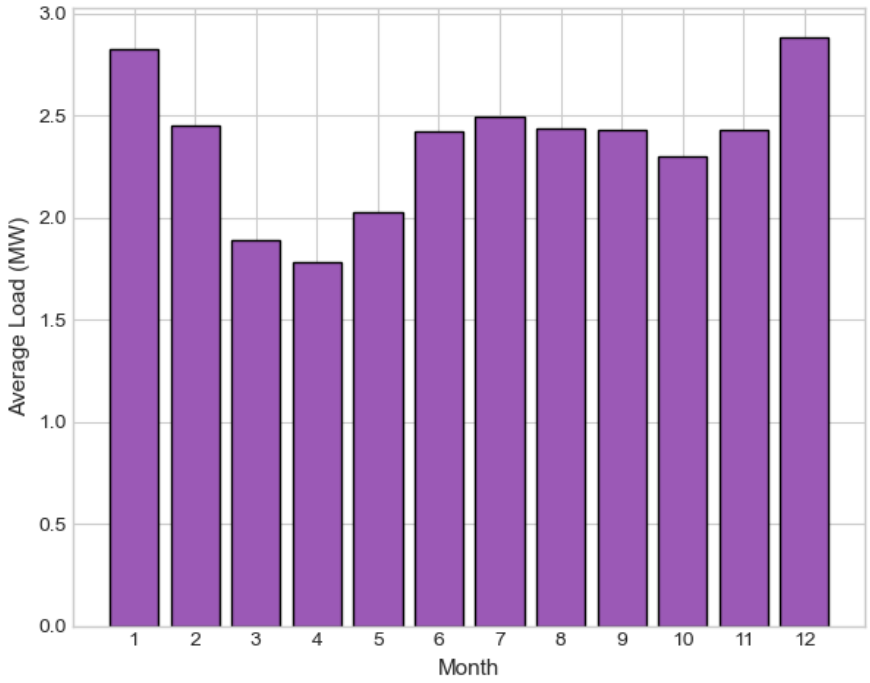

Figure 4.3: Average Load by Month

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	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 4.2 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

Machine learning algorithm performance was quantified through MAE, RMSE, MAPE, and R-squared (R^2) to comprehensively assess forecasting capability. Linear Regression and Ridge Regression functioned as baseline comparators, yielding comparatively elevated error metrics and diminished R^2 coefficients, thereby evidencing their constrained capacity for capturing nonlinear interdependencies between consumption demand and predictor variables. Support Vector Regression demonstrated improvement over linear alternatives by reducing error magnitudes; nonetheless, its performance remained subordinate to ensemble-based methodologies, particularly regarding RMSE outcomes.

Tree-based ensemble algorithms exhibited markedly superior performance across all quantitative criteria. Both Random Forest and Gradient Boosting achieved substantial MAE and RMSE reductions while attaining higher R^2 coefficients, reflecting enhanced generalization capabilities and improved nonlinear pattern recognition. Among these, the hyperparameter-optimized XGBoost implementation surpassed all competing machine learning algorithms, achieving optimal RMSE (0.384 MW), maximal R^2 (0.831), and minimal MAPE (12.693%). The performance differential relative to alternative ensemble approaches stems from XGBoost's gradient-based optimization strategy, integrated regularization mechanisms, and effective feature interaction handling, which collectively enhance model robustness and predictive reliability.

Collectively, the comparative assessment confirms that ensemble machine learning approaches, particularly XGBoost, deliver the most dependable and precise predictions for distribution feeder-level short-term demand forecasting applications.

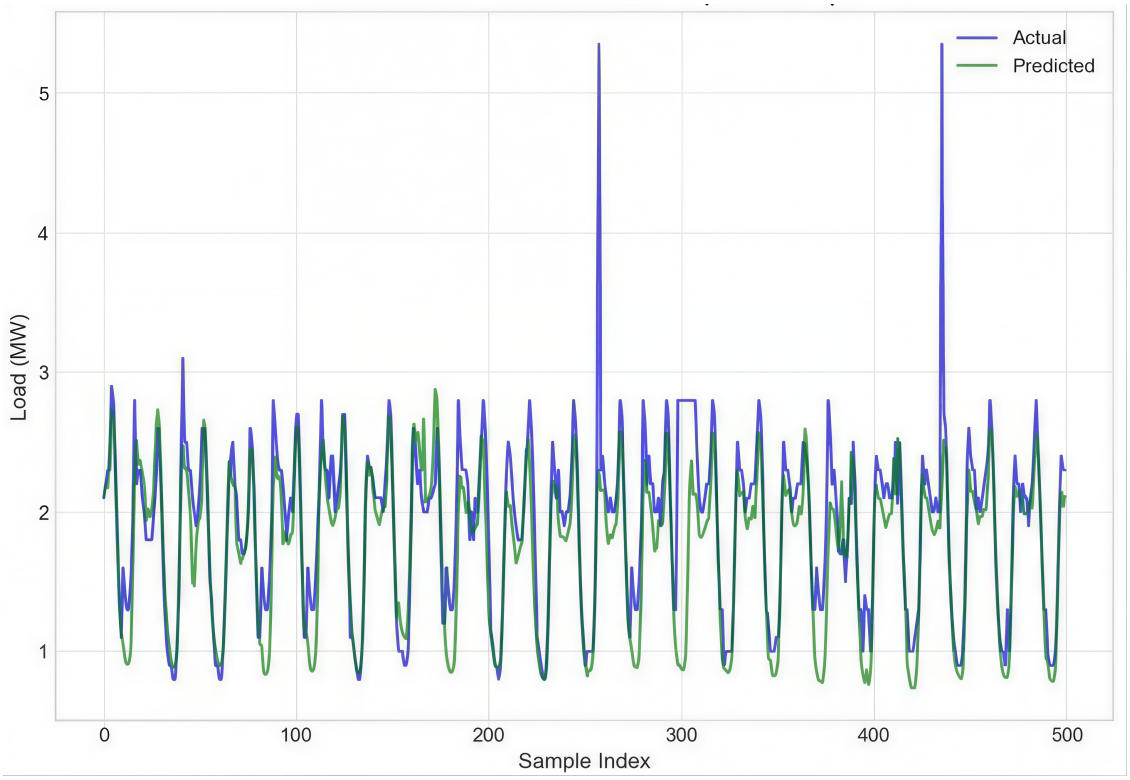

Figure 4.4: XBoost (Tuned) Actual vs Predicted

4.2.2 Deep Learning Model Performance

Deep learning models were trained using lag features and sliding-window sequences to capture temporal dependencies in the feeder load data. Three architectures were implemented: MLP, LSTM, and GRU. Comprehensive experiments were conducted across multiple lag configurations and scaling methods to identify optimal configurations. Table 4.3 presents the hyperparameter and configuration search space explored for the deep learning models.

Table 4.3: Configuration and 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
MLP / 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)

After conducting extensive training using the cleaned and augmented dataset, the models were evaluated using the same metrics as the ML models. The best results obtained for each deep learning architecture are presented in Table 4.4. The MLP model with long lag configuration [1, 3, 6, 12, 24, 48] and standard scaling achieved the best overall performance.

Table 4.4: Deep Learning Models Evaluation Matrix

Sn.No.	Model	MAE	RMSE	MAPE	R^2
1	MLP	0.155	0.242	6.339	0.915
2	LSTM	0.467	0.498	14.155	0.014
3	GRU	0.482	0.499	15.186	0.012

Discussion of Results

Deep neural network implementations—encompassing MLP, LSTM, and GRU architectures—underwent evaluation employing identical performance criteria to ensure equitable cross-model comparison. Among the neural approaches, the MLP architecture achieved exceptional performance with minimal MAE (0.155) and RMSE (0.242), attaining an R^2 coefficient of 0.915. This outstanding result demonstrates that the feed-forward network, when provided with appropriately engineered lag features and temporal encodings, can effectively capture the nonlinear consumption patterns present in the feeder data.

The recurrent architectures LSTM and GRU demonstrated moderate performance, with LSTM achieving MAE of 0.467, RMSE of 0.498, and MAPE of 14.155%, while GRU attained MAE of 0.482, RMSE of 0.499, and MAPE of 15.186%. Both models yielded low but positive R^2 values (0.014 for LSTM and 0.012 for GRU), indicating marginal improvement over naive mean-based forecasting. While these recurrent models showed substantial improvement over baseline approaches, their performance remained significantly below that of the MLP architecture, suggesting that the sequential modeling capabilities of LSTM and GRU did not fully translate to improved forecasting accuracy for this particular dataset.

The substantial performance differential between MLP and recurrent architectures suggests that explicit temporal feature engineering (lag features, cyclical encodings) combined with a simpler feed-forward structure proves more effective than relying on recurrent mechanisms to implicitly learn temporal dependencies. The MLP model, utilizing extended lag configurations [1, 3, 6, 12, 24, 48] with standard scaling, successfully captured both short-term and medium-term consumption patterns. Notably, the MLP achieved performance competitive with the best machine learning models, with its R^2 of 0.915 surpassing even the tuned XGBoost model ($R^2 = 0.831$), demonstrating that neural network approaches can excel when appropriately configured for the forecasting task. The LSTM and GRU models, despite their theoretical advantages for sequential data, exhibited RMSE values approximately double that of MLP, highlighting the importance of architecture selection and feature engineering strategies in deep learning-based load forecasting.

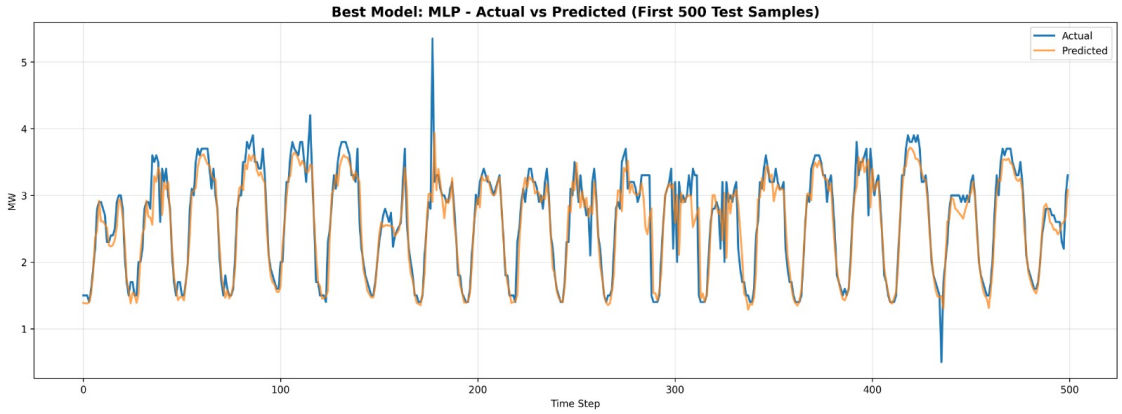

Figure 4.5: MLP Actual vs Predicted

5. Conclusion

5.1 Conclusion

This thesis constructed and rigorously assessed an integrated short-term electrical demand forecasting system tailored for the Baneshwor Feeder, employing both conventional machine learning and contemporary deep learning paradigms. The investigation adhered to a structured experimental protocol encompassing data procurement, quality assurance preprocessing, temporal characteristic extraction, and systematic cross-model performance benchmarking within a consistent evaluation framework.

Empirical findings demonstrate that both machine learning algorithms and appropriately configured deep learning models can achieve excellent forecasting performance. Among deep learning approaches, the Multi-Layer Perceptron (MLP) model with extended lag features attained the best overall predictive accuracy, registering an RMSE of 0.242 MW alongside an R^2 coefficient of 0.915, thereby evidencing superior capability in representing nonlinear consumption dynamics and meteorologically-influenced demand variations. The hyperparameter-optimized XGBoost model also achieved strong performance with an RMSE of 0.384 MW and R^2 of 0.831. Complementary tree-based ensemble approaches, including Random Forest and Gradient Boosting, likewise yielded competitive accuracy metrics, corroborating the efficacy of aggregated decision tree methodologies for distribution-level demand prediction.

Among deep neural network implementations, performance varied considerably across architectures. The Multi-Layer Perceptron (MLP) model achieved exceptional results, attaining an RMSE of 0.242 MW and R^2 coefficient of 0.915—outperforming all machine learning models including the tuned XGBoost. This success demonstrates that feed-forward networks, when provided with comprehensive lag features and appropriate feature engineering, can effectively capture complex consumption dynamics. The recurrent architectures LSTM and GRU achieved moderate performance, with LSTM attaining RMSE of 0.498 MW and R^2 of 0.014, while GRU recorded RMSE of 0.499 MW and R^2 of 0.012. Although these models demonstrated positive predictive capability, their performance remained substantially below both the MLP and the best machine learning models, suggesting that recurrent networks may require larger training corpora, finer temporal resolution, or hybrid approaches to fully realize their theoretical advantages within this application domain.

Collectively, this investigation validates that precise distribution feeder-level short-term

demand forecasting remains achievable through both judiciously architected machine learning workflows and appropriately configured neural network approaches. The experimental outcomes underscore that algorithmic selection should be principally informed by dataset characteristics, available feature dimensionality, and operational deployment constraints. Notably, the MLP's success with explicit lag features demonstrates that feed-forward architectures with proper feature engineering can outperform both traditional ML models and more complex recurrent neural networks for feeder-level forecasting applications.

5.2 Research Limitations

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

1. **Recurrent architecture limitations:** While MLP achieved excellent performance, LSTM and GRU models yielded relatively low R^2 values (0.014 and 0.012 respectively), suggesting that recurrent architectures may require substantially larger datasets, finer temporal resolution, or hybrid approaches combining explicit lag features with sequence modeling for more effective learning.

2. **Irregular load behaviour:** Feeder-level load profiles often contain noise, outages, fluctuations, and sudden spikes, which proved particularly challenging for recurrent deep learning models to learn without additional contextual variables.

3. **Sequence modeling constraints:** The sliding window approach used for LSTM and GRU models may not fully capture the complex weekly or monthly patterns that the lag-feature-based MLP successfully learned through explicit feature engineering.

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:** Both ensemble machine learning models (particularly XGBoost) and the MLP deep learning model can provide accurate, low-error forecasts suitable for operational planning, peak management, and scheduling.

2. **Value of feature engineering:** Carefully constructed temporal features, lag variables, and weather features significantly improved forecasting accuracy, highlighting the importance of domain knowledge in model design. The MLP model's superior performance with extended lag configurations demonstrates the critical role of explicit temporal feature engineering.

3. **Architecture-specific considerations for DL:** While the MLP achieved the best overall performance ($R^2 = 0.915$), recurrent architectures LSTM ($R^2 = 0.014$) and GRU ($R^2 = 0.012$) demonstrated comparatively limited predictive capability, reinforcing that deep learning architecture selection must be carefully matched to dataset characteristics and forecasting requirements.

4. **Foundation for advanced decision-making tools:** Forecasts from both the MLP and XGBoost models 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 both machine learning and deep 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. **Enhanced Recurrent Architectures:** While MLP achieved excellent results ($R^2 = 0.915$), LSTM and GRU models attained substantially lower performance (R^2 of 0.014 and 0.012 respectively). These recurrent architectures may benefit from larger datasets, finer temporal resolution, attention mechanisms, or hybrid architectures combining explicit lag-feature engineering with sequence modeling to improve their forecasting capability.

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

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

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

[8] 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.

[9] 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.

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

[11] 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.

[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] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

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

[29] 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.