

# Design and Implementation of an Educational Electricity Load Forecasting System: A Modular and Adaptive Approach

Technical Report

David Santiago Téllez Melo (20242020107)

Ana Karina Roa Mora (20232020118)

Daniela Bustamante Guerra (20241020131)

Andrés Felipe Correa Méndez (20221020141)

Faculty of Engineering, Systems Engineering Program

Universidad Distrital Francisco José de Caldas

Bogotá, Colombia

Emails: {dstellezm, akroam, dabustamanteg,  
afcorreame}@correo.udistrital.edu.co

October 2025

## Abstract

Electric load forecasting plays a crucial role in optimizing power grid operations and ensuring energy reliability. This report presents a modular, reproducible forecasting framework based on **PyTorch** that integrates two complementary models: a **Multilayer Perceptron (MLP)** for tabular, feature-engineered data and a **Long Short-Term Memory (LSTM)** network for sequential modeling.

The system design follows a multi-layer architecture, clearly separating data preprocessing, feature engineering, modeling, validation, and evaluation stages. This ensures flexibility, interpretability, and ease of debugging. The proposed approach incorporates calendar, temperature, and historical consumption data, along with derived indicators such as Heating and Cooling Degree Days (HDD/CDD), to enhance temporal feature richness. Models are trained and validated using a rolling-origin cross-validation scheme, which closely replicates real-world forecasting conditions.

Preliminary findings indicate that the architecture achieves consistent accuracy and supports model interchangeability without modifying the pipeline structure. The proposed system establishes a professional, reproducible workflow suitable for academic evaluation and educational use.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Methods and Materials</b>	<b>5</b>
2.1	System Architecture Overview . . . . .	5
2.2	Clean ML Implementation . . . . .	6
2.3	Handling Sensitivity and Chaos . . . . .	7
2.4	Data Sources and Preprocessing . . . . .	7
2.5	Validation and Metrics . . . . .	8
<b>3</b>	<b>Results</b>	<b>9</b>
<b>4</b>	<b>Discussion</b>	<b>10</b>
<b>5</b>	<b>Conclusion</b>	<b>11</b>
<b>6</b>	<b>References</b>	<b>12</b>
<b>A</b>	<b>Appendices</b>	<b>13</b>
<b>B</b>	<b>Acknowledgements</b>	<b>14</b>
<b>C</b>	<b>Glossary</b>	<b>15</b>

# List of Figures

1.1	General architecture of the proposed load forecasting system. Each module handles a specific layer of the workflow, ensuring modularity and reproducibility. . . . .	4
2.1	Integrated data ingestion, feature engineering, and forecasting workflow with feedback loop. . . . .	6
2.2	Clean ML Architecture Layers and Responsibilities . . . . .	7

# 1. Introduction

Electric load forecasting is a fundamental task for energy providers, enabling efficient grid management, cost optimization, and reliability in energy distribution. With the increasing integration of renewable energy sources and the variability of consumption patterns, accurate forecasting has become more challenging and crucial. Recent advances in machine learning and deep learning have significantly improved predictive capabilities, allowing models to capture nonlinear patterns and temporal dependencies in large datasets.

Traditional forecasting methods, such as ARIMA and exponential smoothing, often struggle to adapt to complex feature interactions and external variables such as temperature, holidays, or socioeconomic events. In contrast, neural networks—particularly **Multilayer Perceptrons (MLP)** and **Long Short-Term Memory (LSTM)** architectures—offer flexibility and robustness for modeling nonlinear and sequential relationships.

However, implementing these models in a reproducible and maintainable way remains a challenge, especially when dealing with heterogeneous data sources and long-term deployment. This work proposes a modular forecasting framework that separates each stage of the pipeline into clearly defined layers: data ingestion, feature engineering, model training, forecasting, and evaluation. Each layer is independently testable, reusable, and replaceable, which facilitates experimentation and scalability. The implementation relies on **PyTorch**, ensuring modern deep learning capabilities while maintaining transparency and reproducibility.

The system integrates various data sources, including historical load data, weather information, and calendar events, to create a rich set of engineered features. Using these, two models — an MLP and an LSTM — are trained and validated using rolling-origin cross-validation, a methodology that mimics real-world forecasting conditions by continuously shifting the training and validation windows.

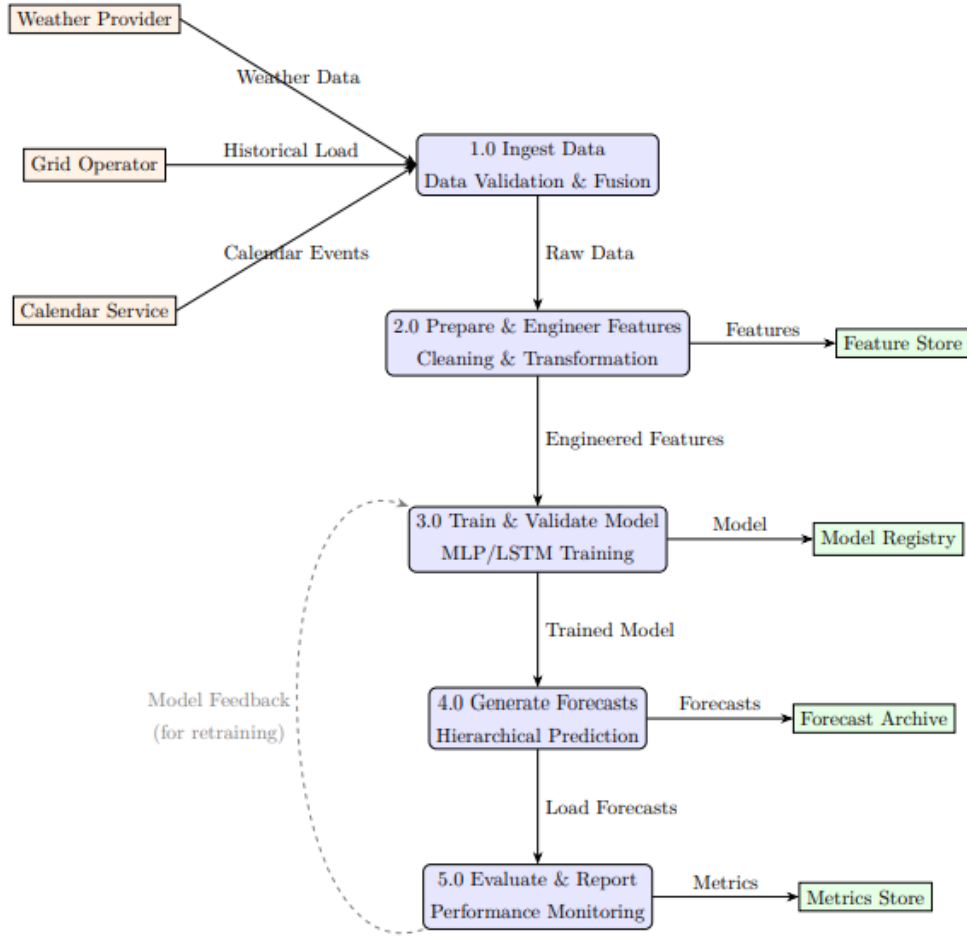


Figure 1.1: General architecture of the proposed load forecasting system. Each module handles a specific layer of the workflow, ensuring modularity and reproducibility.

## 2. Methods and Materials

This section provides a detailed description of the technical design and implementation of the proposed load forecasting system. The approach follows systems engineering principles, emphasizing modularity, reproducibility, and robustness under sensitive and chaotic data conditions.

### 2.1 System Architecture Overview

The system is designed as a **modular pipeline** that converts heterogeneous data sources into hierarchical electricity load forecasts. The architecture comprises five main stages:

- **Data Ingestion:** This module collects historical load data, weather information (temperature, humidity), and calendar events (holidays, weekends). Each source undergoes format standardization, missing value imputation, and anomaly detection. For example, rolling mean and linear interpolation are applied to fill gaps in temperature readings, while sudden spikes in consumption are smoothed to prevent propagation to the model.
- **Feature Engineering:** This stage generates predictive features including:
  - Temporal features: hour-of-day, day-of-week, month, season indicators.
  - Lagged load features: past 24 hours, 168 hours (week-long) moving averages.
  - Weather features: current temperature, lagged temperatures, heating/cooling degree days (HDD/CDD).
  - Interaction features: e.g., temperature  $\times$  hour-of-day to capture non-linear effects.

Feature normalization and scaling are applied to ensure numerical stability for neural network training.

- **Model Training:** Two neural architectures are implemented:
  1. **Multilayer Perceptron (MLP):** Designed for tabular features, consisting of several fully connected layers with Batch Normalization, ReLU activations, and dropout regularization to prevent overfitting.

2. **Long Short-Term Memory (LSTM):** Designed for sequential modeling of temporal dependencies. Input sequences of past 168 hours predict the next horizon. Dropout and gradient clipping are applied to improve stability.

Both models are trained using stochastic gradient descent with adaptive learning rate (Adam optimizer). Hyperparameters such as number of layers, hidden units, dropout rate, and batch size are tuned through cross-validation.

- **Forecast Generation:** This module produces zone-level and aggregated forecasts. Hierarchical reconciliation ensures that sum of individual zones equals total system demand. Uncertainty is quantified via ensemble averaging and confidence interval estimation.
- **Evaluation and Monitoring:** Model performance is continuously monitored using rolling-origin cross-validation. Metrics include Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), and residual analysis. If errors exceed predefined thresholds, automatic retraining is triggered using the most recent stable dataset.

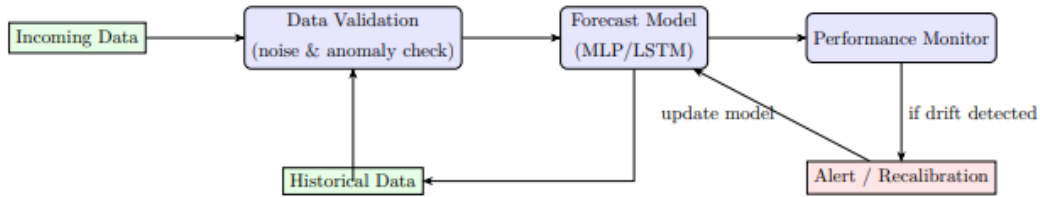


Figure 2.1: Integrated data ingestion, feature engineering, and forecasting workflow with feedback loop.

## 2.2 Clean ML Implementation

The system follows a "Clean ML Architecture", which separates responsibilities across layers to improve maintainability and testability:

- **Data Layer:** Handles reading, cleaning, and standardizing datasets. Implements missing value imputation and anomaly detection.
- **Feature Layer:** Constructs engineered features from historical load, weather, and calendar data. Supports creation of lagged and interaction features.
- **Model Layer:** Encapsulates PyTorch MLP and LSTM models. Provides modular interfaces for training, inference, and hyperparameter tuning.
- **Validation Layer:** Implements rolling-origin cross-validation, calculates RMSE and MAPE, and supports ensemble evaluation.

- **Application Layer:** Coordinates the workflow, manages model saving/loading, and generates outputs for visualization or reporting.

Layer	Responsibility	Example Component
Data Layer	Read, clean, and standardize raw data.	<code>data_repo.py</code>
Feature Layer	Add calendar, weather, and lagged features.	<code>features/</code>
Model Layer	Define and train PyTorch models (MLP or LSTM).	<code>torch_models.py</code> , <code>torch_train.py</code>
Validation Layer	Handle rolling-origin cross-validation and metrics.	<code>cv.py</code> , <code>metrics.py</code>
Application Layer	Coordinate the workflow and generate the submission file.	<code>train.py</code> , <code>infer.py</code>

Figure 2.2: Clean ML Architecture Layers and Responsibilities

## 2.3 Handling Sensitivity and Chaos

Electricity demand is highly sensitive to small changes in temperature, holidays, and social events. The system incorporates multiple mechanisms to improve robustness:

- Continuous input validation and anomaly smoothing to prevent propagation of errors.
- Dropout regularization in neural networks to reduce overfitting to noisy patterns.
- Ensemble averaging across multiple training runs to reduce stochastic variability.
- Feedback-based retraining triggered by deviations above thresholds in RMSE or MAPE.

## 2.4 Data Sources and Preprocessing

The dataset includes:

- **Historical load:** Hourly electricity consumption per zone.
- **Weather data:** Temperature, humidity, and derived HDD/CDD indicators.
- **Calendar events:** Holidays, weekends, and special events affecting consumption.

Preprocessing involves merging all sources into a unified dataset, handling missing data, detecting anomalies, and scaling features to ensure numerical stability.

## 2.5 Validation and Metrics

Models are validated using **rolling-origin cross-validation**, simulat

### 3. Results

## 4. Discussion

## 5. Conclusion

## 6. References

## A. Appendices

## B. Acknowledgements

## C. Glossary

- **MLP:** Multilayer Perceptron
- **LSTM:** Long Short-Term Memory
- **HDD/CDD:** Heating/Cooling Degree Days