Universidad Distrital Francisco Jose de Caldas

Workshop 2

Systems Analysis: Global Energy Forecasting Competition 2012
Load Forecasting

Autores:

David Santiago Téllez Melo – 20242020107
Ana Karina Roa Mora – 20232020118
Daniela Bustamante Guerra – 20241020131
Andrés Felipe Correa Méndez – 20221020141

Fecha: 18 de octubre de 2025

1. Review of Workshop #1 Findings

The analysis conducted in Workshop #1 provided a comprehensive understanding of the structure and dynamics of the energy forecasting system proposed in the context of the *Global Energy Forecasting Competition 2012 (GEFCom2012)*. This study revealed that the system consists of multiple interdependent elements—load zones, climatic variables, and temporal factors—whose interaction defines a complex and partially chaotic environment.

Among the main findings, several critical constraints were identified that influence the system's performance: the need to maintain hierarchical consistency between individual zones and the total load; the incompleteness of temperature data, which introduces uncertainty into predictive models; and the high sensitivity to small variations in input variables, particularly during periods of peak demand.

The characteristics of the dataset reinforced this complexity: time series with strong seasonal behaviors, nonlinear relationships between temperature and consumption, and disturbances generated by external factors such as holidays or social events. These dynamics reflect the presence of chaotic elements, where small changes can be amplified and produce large deviations in the final forecasts.

Based on these findings, future design proposals should focus on developing robust models capable of handling incomplete data and nonlinear relationships, implementing hierarchical reconciliation methods to ensure consistency across system levels, and integrating sensitivity analysis and stochastic simulation techniques to mitigate the impact of inherent randomness in electricity demand.

Overall, these guidelines aim to build a forecasting system that is more stable, adaptable, and representative of the real behavior of energy demand.

2. System Requirements Definition

Based on the findings from the initial analysis, the following requirements are defined to guide the development of the electricity load forecasting system. The goal is to design an educational and functional tool that simulates energy demand behavior considering weather and historical patterns, while keeping the system simple and modular.

2.1 Functional Requirements

The system must be able to import and process a dataset containing historical electricity consumption and basic climatic variables such as average temperature or holiday

dates. From these inputs, the program should perform basic data cleaning (for example, handling missing or inconsistent values) and generate a simple forecast of future demand. For prediction, it is proposed to use a basic model such as **linear regression** or a **moving average**, so that students can easily understand the relationship between the variables and the forecasted output.

2.2 Performance and Reliability Requirements

The system should run correctly on any standard computer and produce results in reasonable time. The goal is not professional-level precision, but rather stability and consistency in the calculations. Additionally, the system must be **tolerant to common errors**, such as files with missing values or disordered columns. It should also allow **basic performance evaluation** of the model, displaying the error between actual and predicted values (for example, through RMSE or average percentage error). This helps users understand the model's limitations and strengths.

2.3 User-Centered Requirements

The design should prioritize **ease of use and visual clarity**. It is recommended to include a simple interface (either a console-based or a basic graphical interface) that displays clear results such as comparative tables and trend plots. The user should be able to adjust basic parameters, such as date ranges or which climatic variable to include. The system should be **interpretable**, allowing users to understand why the model produces certain results by relating them to the original data. It should also provide clear messages in case of errors or missing data, avoiding confusion during execution.

2.4 Overall Design Objective

The final system will serve as a practical representation of the energy forecasting process, applying the principles learned in the systems analysis. Rather than achieving a highly accurate model, the goal is to **understand system behavior**, how small data variations can affect predictions, and how basic techniques can improve reliability. This project integrates programming fundamentals, basic statistics, and data interpretation, offering a realistic and applied learning experience within the academic scope of the course.

3. High-Level Architecture

The proposed system architecture is designed as a modular pipeline that transforms raw multi-source data into reliable electricity load forecasts while maintaining hierarchical consistency across all zones. The design follows systems engineering principles to ensure scalability, maintainability, and robustness against the sensitive and chaotic elements identified in Workshop #1.

3.1 Architectural Overview

The architecture implements a five-stage processing pipeline where data flows sequentially from ingestion to evaluation, with strategic feedback loops for continuous model improvement. Each stage is encapsulated as an independent module with well-defined interfaces, enabling parallel development and testing.

3.2 Core Processing Pipeline

- **Data Ingestion Layer**: Responsible for collecting and fusing heterogeneous data sources including real-time weather feeds, historical load records, and calendar information. Implements data validation and format standardization.
- **Feature Engineering Engine**: Transforms raw inputs into predictive features through temporal encoding, weather-load relationship modeling, and holiday pattern extraction. Handles missing data and abnormal values through robust imputation techniques.
- **Model Training Framework**: Supports multiple neural architectures (MLP/LSTM) with hierarchical consistency constraints. Implements temporal cross-validation and hyperparameter optimization for model selection.
- **Forecast Generation Module**: Produces zone-level and system-level predictions while enforcing aggregation coherence. Includes uncertainty quantification and confidence interval calculation.
- **Evaluation & Monitoring System**: Continuously tracks model performance, detects data drift, and triggers recalibration when performance degrades beyond defined thresholds.

3.3 Systems Engineering Principles Integration

The architecture embodies key engineering principles through:

- **Modular Decomposition**: Clear separation of concerns with standardized interfaces between components

- **Scalable Data Flow**: Pipeline architecture supports parallel processing and distributed computation
- **Fault Tolerance**: Comprehensive error handling and data persistence at each processing stage
- **Maintainability**: Independent versioning of models, features, and data schemas
- **Traceability**: Full data lineage from raw inputs to final forecasts with audit capabilities

3.4 Architectural Diagram

The following diagram illustrates the complete system architecture, showing data flows, storage components, and feedback mechanisms that ensure robust forecasting performance.

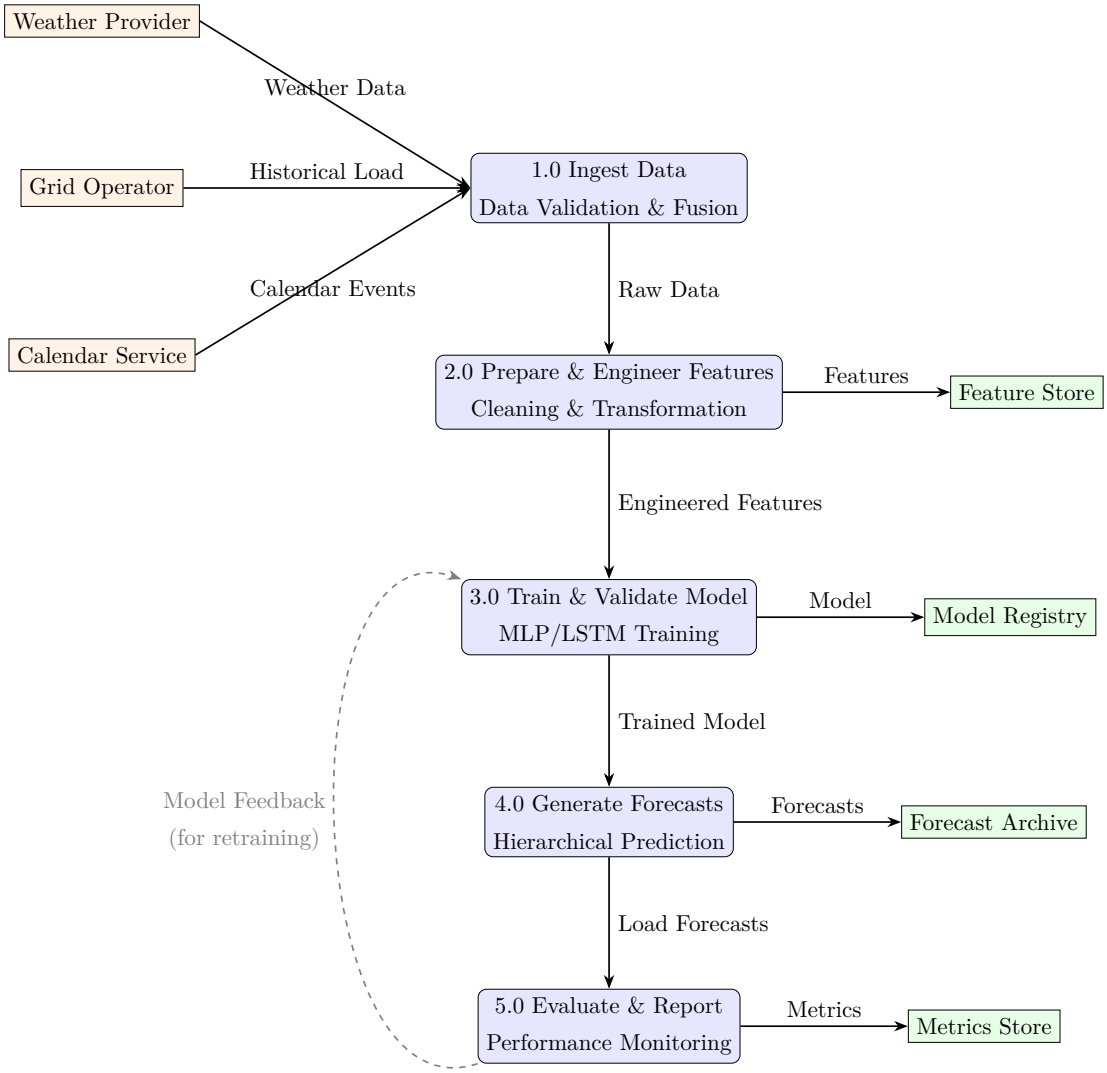

Figura 1: Integrated data ingestion, feature engineering, and forecasting workflow with clean feedback loop.

3.5 Data Flow and Integration Patterns

The architecture implements several key data flow patterns:

- **Unidirectional Pipeline**: Data progresses sequentially through processing stages, maintaining clear responsibility boundaries and enabling stage-specific optimization
- **Storage Decoupling**: Each major processing stage outputs to dedicated storage, allowing independent scaling and fault recovery
- **Feedback Integration**: Performance metrics from the evaluation stage inform model retraining decisions, creating a continuous improvement cycle
- **External API Integration**: Real-time connections to weather services and grid operational data ensure forecasting relevance

3.6 Design Rationale and Competitive Alignment

This architectural approach directly addresses the GEFCom2012 competition requirements while providing a foundation for real-world deployment:

- **Hierarchical Consistency**: The modular design allows explicit handling of zone-level and system-level coordination through specialized reconciliation components
- **Temperature Sensitivity**: Dedicated feature engineering and model validation components manage the nonlinear relationships between weather variables and electricity demand
- **Temporal Complexity**: The pipeline captures multiple time scales (hourly, daily, seasonal) through specialized feature creation and model architectures
- **Uncertainty Propagation**: Multiple validation stages and ensemble techniques address the chaotic elements inherent in energy consumption patterns

The architecture represents an optimal balance between competition-specific optimizations and general forecasting principles, ensuring both immediate performance and long-term maintainability.

4. Addressing Sensitivity and Chaos

The energy load forecasting problem inherently involves sensitive and chaotic behaviors. Small changes in temperature, social activity, or missing data can produce disproportionate variations in electricity demand predictions. Therefore, the system design explicitly incorporates mechanisms to manage data instability, noise, and unexpected feedback loops.

4.1 Handling Data Sensitivity

The system includes a **data stability pipeline** that performs continuous validation of input variables. Before model training, all datasets are analyzed for missing values, abnormal spikes, or sudden shifts using rolling statistical checks. When anomalies are detected, the system applies automatic data smoothing or imputation routines (e.g., rolling means or interpolation). This prevents small irregularities from propagating and amplifying within the model.

4.2 Noise and Uncertainty Management

To ensure robustness against random noise, the model employs:

- **Dropout layers** in neural networks, which reduce overfitting to unstable patterns.
- **Rolling-origin cross-validation**, ensuring that performance metrics are evaluated on unseen temporal segments, thus reducing bias from transient events.
- **Ensemble averaging**, combining multiple training runs to smooth stochastic variability in the outputs.

4.3 Error Control and Feedback Monitoring

Given the dynamic nature of real-world load forecasting, the system integrates a lightweight monitoring mechanism. After each forecast cycle:

- The model compares recent predictions with actual load data.
- If deviations exceed a defined threshold (e.g., more than 15 % RMSE increase), an **automatic recalibration routine** is triggered.
- This recalibration retrains the model using the most recent stable window of data.

Additionally, a **feedback alert module** logs abnormal residual patterns or data drift, notifying the operator that unexpected behaviors may be emerging in the energy consumption profile.

Chaotic Behavior Awareness

Electricity consumption can exhibit chaotic signatures, where seemingly minor variations (such as a holiday shift or sudden weather front) lead to non-linear changes in demand. To capture these effects, the system stores lag-based and nonlinear interaction features (e.g., temperature $\times$ hour-of-day). These features help the model learn partial chaotic dependencies and dampen instability in long-term predictions.

4.4 Visualization and Sensitivity Diagram

To illustrate how sensitivity and feedback are managed, Figure 2 shows a simplified system loop, where external disturbances are monitored and corrected through adaptive recalibration.

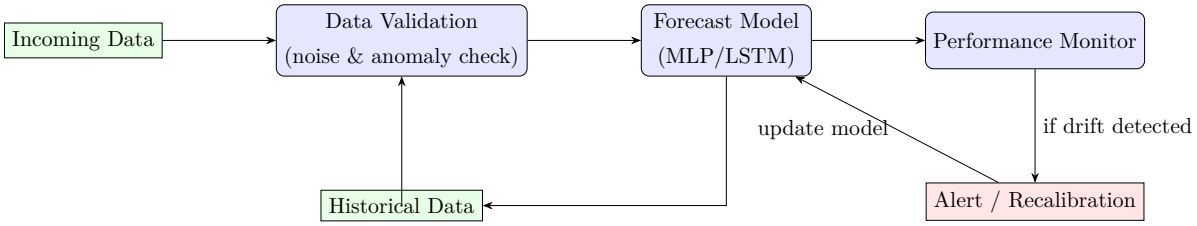

Figura 2: Sensitivity and chaos management loop in the proposed system.

4.5 Summary

By integrating automatic data validation, ensemble averaging, dropout regularization, and feedback-based recalibration, the system remains resilient under unpredictable fluctuations. This approach aligns with the broader principle of **adaptive system design**, where the model continuously learns and corrects itself as new chaotic or unstable patterns emerge in the data.

5. Technical Stack and Architectural Justification

5.1 Selected Stack: Minimal Clean ML + PyTorch

The proposed implementation follows a **Minimal Clean ML architecture**, supported by a single major framework **PyTorch** complemented by lightweight Python libraries such as pandas, NumPy, and scikit-learn. This configuration achieves an optimal balance between clarity, reproducibility, and technical rigor, while avoiding unnecessary complexity.

5.2 Why this setup was chosen

The Global Energy Forecasting Competition (GEFCom 2012) focuses on time series load forecasting using weather and calendar data. The key requirements are the following:

1. Accurate short-term forecasts (hourly or daily).
2. Robust cross-validation with temporal separation.
3. Interpretability and reproducibility.

Given that this project is not intended to scale into a production system, the chosen setup provides enough flexibility to model complex temporal dependencies while keeping the workflow simple, transparent, and maintainable.

Rationale

- **PyTorch** was selected as the main framework because it is general purpose, industry standard, and capable of handling both tabular neural networks (MLP) and sequence models (LSTM). This allows experimentation with classical and deep learning approaches under one unified framework.
- **pandas** and **NumPy** efficiently manage data cleaning and feature engineering (lags, rolling means, weather joins, HDD/CDD, calendar features).
- **scikit-learn** supports cross-validation, scaling, and baselines such as Ridge regression, ensuring comparability with traditional models.
- **matplotlib** enables quick diagnostic plots (seasonal patterns, residuals, feature correlations).
- **holidays** (optional) adds calendar and holiday indicators, essential for load forecasting tasks.

This setup covers the complete pipeline—from raw data ingestion to final forecast submission—without the overhead of orchestration or tracking frameworks (e.g., Prefect, MLflow), which are unnecessary for small-scale academic work.

5.3 Clean ML Architecture Overview

The system follows a **Clean ML architecture**, a simplified adaptation of Clean Architecture for machine learning. It organizes the codebase into modular and testable layers:

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

This modular structure ensures that:

- Data preprocessing is completely separated from model logic.
- Models can be replaced (e.g., MLP → LSTM) without altering other layers.
- Each component is independently testable and easy to debug.

5.4 Modeling Strategy

Two PyTorch-based models are proposed:

1. **MLP (Multilayer Perceptron) for tabular features:** Input consists of engineered features such as lags, rolling statistics, calendar variables, temperature, and HDD/CDD indicators. The architecture follows a pattern of *BatchNorm → (Linear + ReLU + Dropout)* → Output*. It is fast to train, interpretable, and performs well when feature engineering is strong.
2. **LSTM (Long Short-Term Memory) for sequence modeling:** The input is a temporal window (e.g., past 168 hours) used to predict the next horizon. This architecture captures long-term dependencies and is ideal for experiments involving temporal dynamics.

Both approaches are evaluated using rolling-origin cross-validation, which mimics real-world forecasting by training on past periods and validating on future ones. The official competition metric is implemented precisely for each fold to guarantee consistent evaluation.

5.5 Conclusion

This setup delivers:

- A professional architecture aligned with software engineering principles.
- A powerful yet approachable modeling core built on PyTorch.
- A fully reproducible workflow ideal for documentation, evaluation, or portfolio presentation.