DESIGN AND IMPLEMENTATION

EDUCATIONAL ELECTRICITY LOAD FORCASTING SYSTEM

DAVID SANTIAGO TELLEZ MELO - 20242020107

ANA KARINA ROA MORA - 20232020118

DANIELA BUSTAMANTE GUERRA - 20241020131

ANDRÉS FELIPE CORREA MORA - 20221020141

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS
SATURDAY, OCTOBER 25
2025-III

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Outline

1. Context and key findings
2. System requirements
3. Architecture and data flow
4. Sensitivity and chaos management
5. Technical stack and modeling
6. Conclusions and Q&A

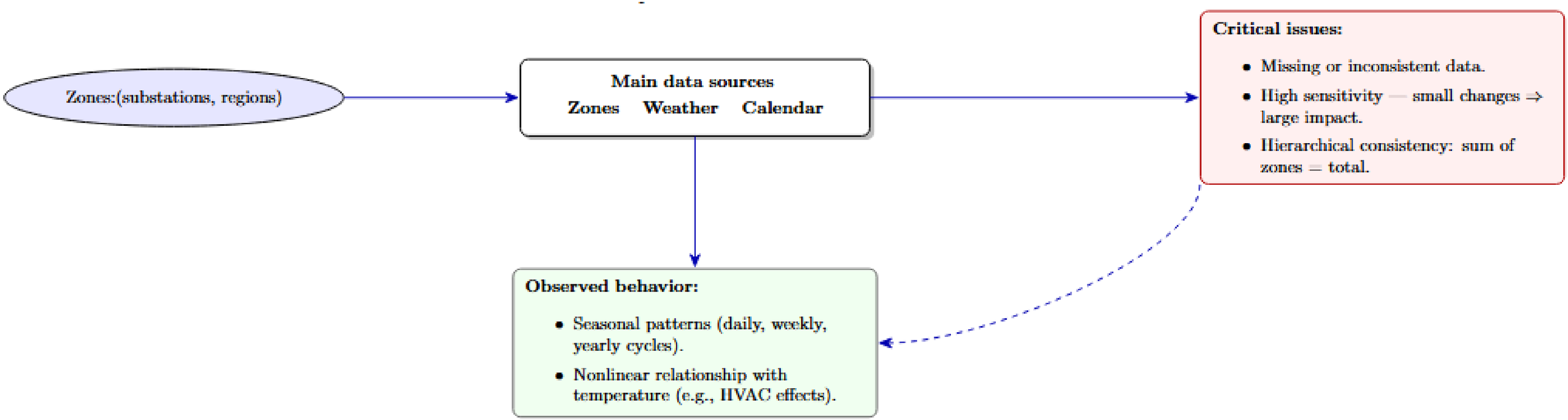
Why this problem?

Electricity demand constantly changes depending on the time of day, the weather, and human habits.
If companies don't predict it correctly, they can over- or under-generate, causing losses or blackouts.

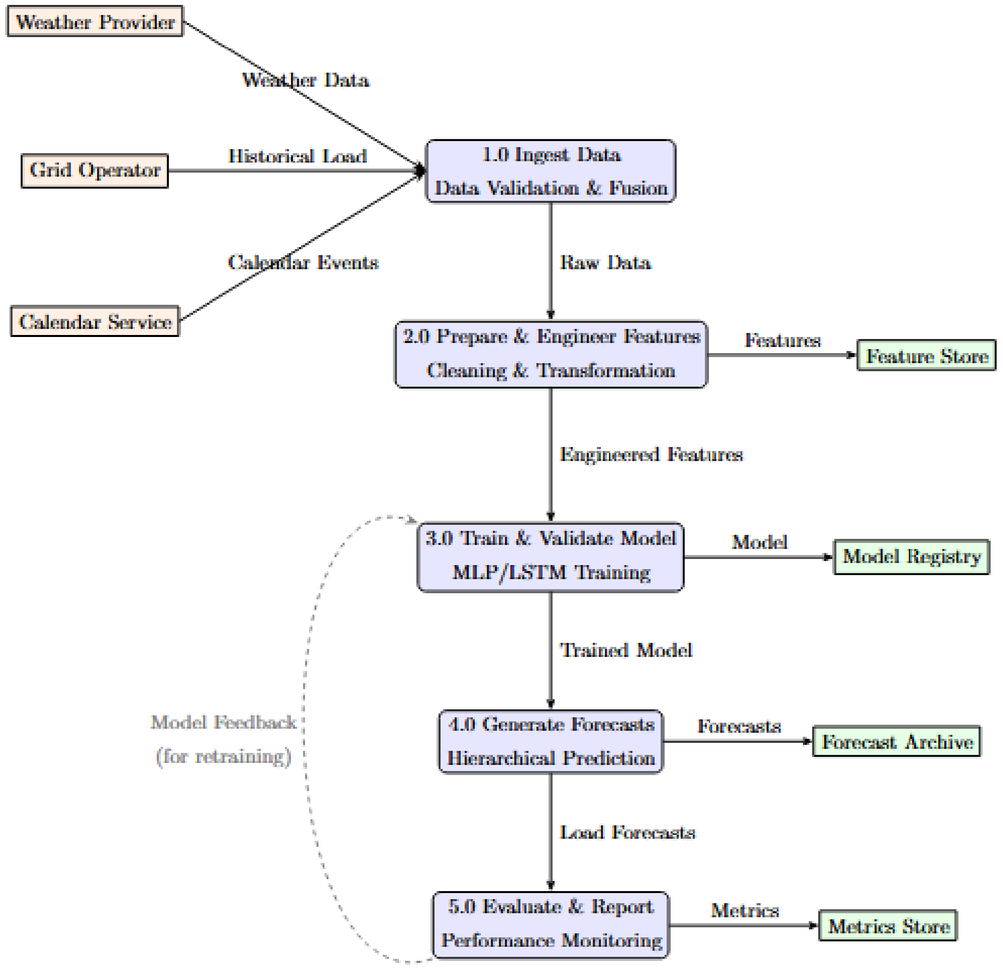
In 2012, a global competition called GEFCom2012 was held, where researchers and companies attempted to predict actual electricity demand using weather and consumption data.

- Load forecasting estimates how much electricity will be used in the future.
- It’s essential for planning generation, distribution, and costs.
- The project is inspired by the GEFCom2012 Kaggle competition.
- Goal: create an educational and functional forecasting tool.

Key Findings in Workshop #1



Trasnformation and Functional requirements



Performance and user requirements

We focus primarily on functional usability for any type of user, since this is a purely educational project, and our stakeholders, as end users, must be any type of person interested in knowledge about the electrical load forecasting educational system.

Runs on standard computers.

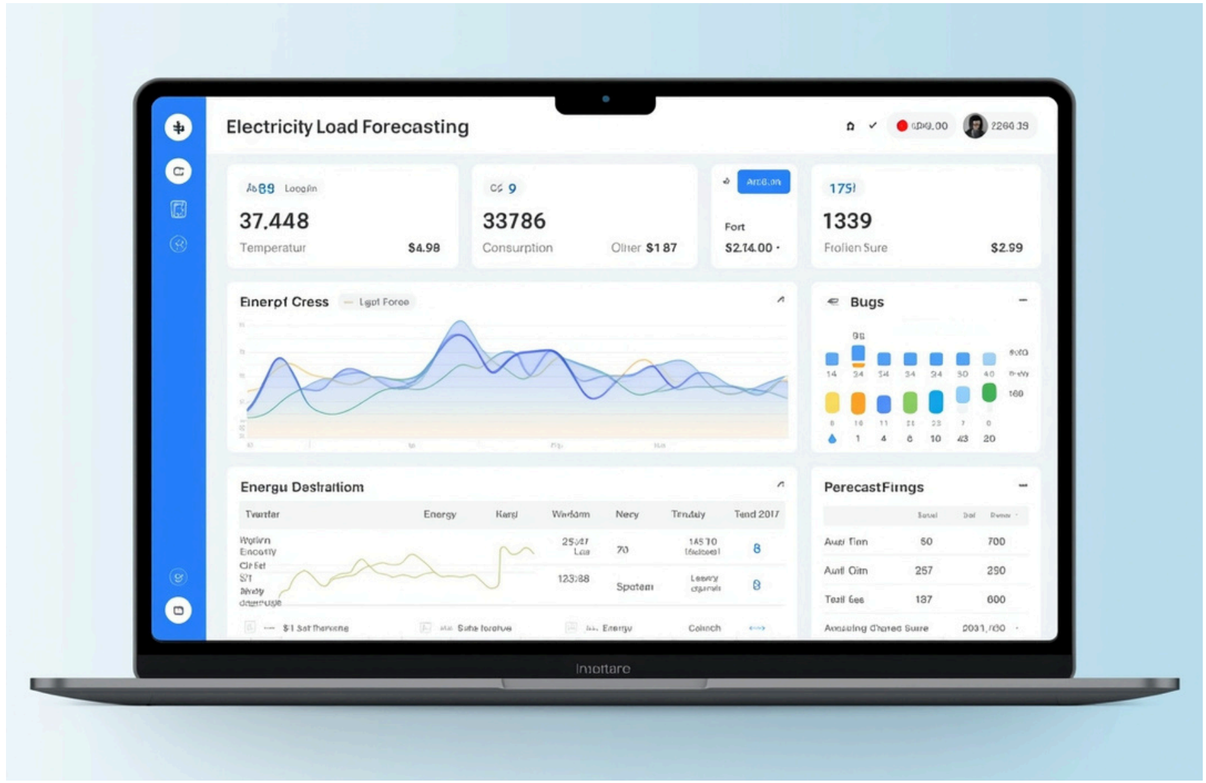
Simple and error-tolerant interface.

Allows selecting date range and weather variables.

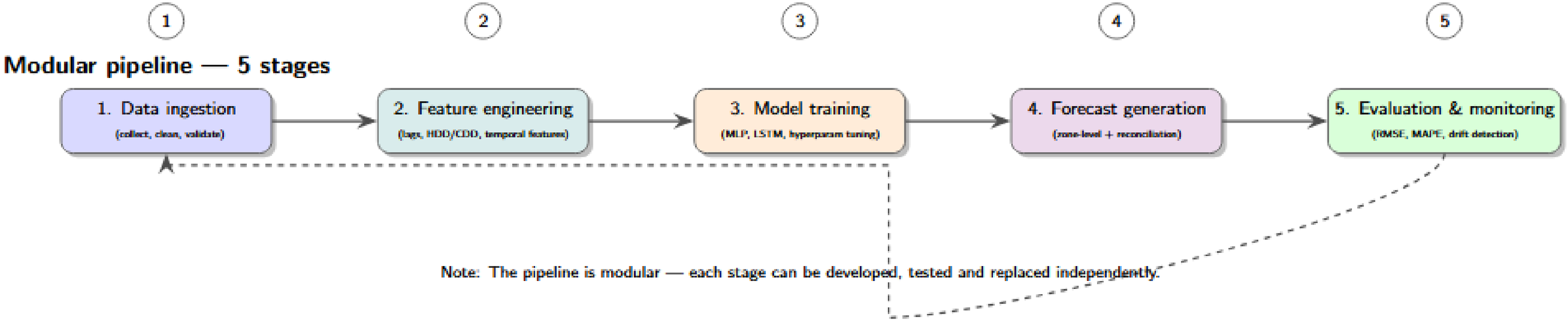
Clear, interpretable outputs and messages.

Mockups

This mochup is made by AI, using diferent tools to give you a better idea of what we want to do.



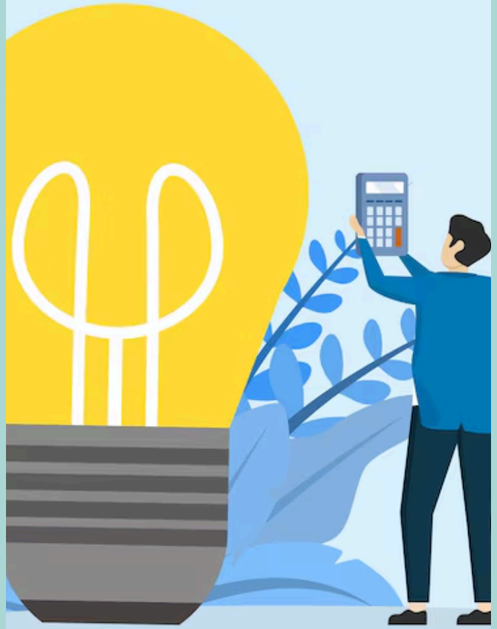
High-level architecture



Data Ingestion & Feature Engineering

Ingestion
Combines weather, calendar, and load data.

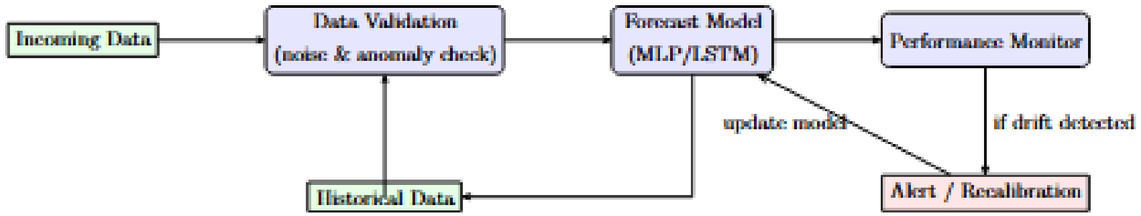
Validation
Fixes missing values and anomalies.



Feature creation
- Time lags (24h, 168h)
- HDD/CDD (Heating/Cooling Degree Days)
- Hour, day, week, seasonal indicators
- Interactions (e.g., temp × hour)

Imputation
Rolling mean and interpolation.

Model Training and Forecast Generation



Model Training

- Neural networks used: MLP (tabular input) and LSTM (time-sequence input).
- Rolling-origin validation: sliding time windows for realistic evaluation.
- Objective: learn patterns between temperature, time, and consumption.

Forecast Generation

- Predictions made per zone.
- Hierarchical reconciliation: ensures zone forecasts sum to total.
- Output: coherent and interpretable forecasts for all regions.

Evaluation, Monitoring & Feedback Loop

Evaluation

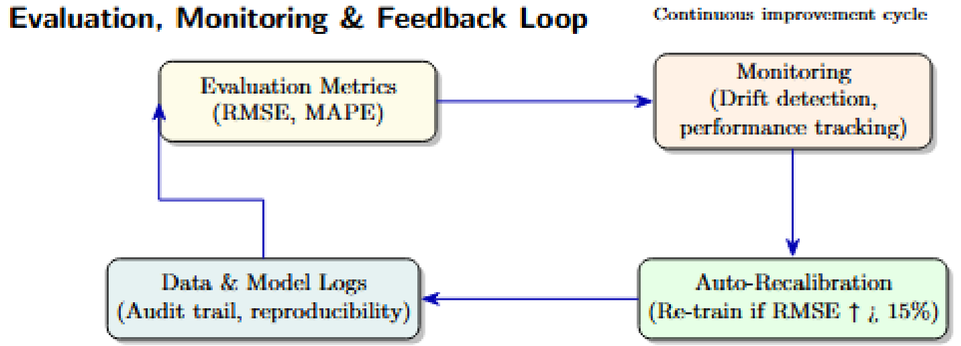
- Metrics: RMSE (absolute error) and MAPE (percentage error).
- Continuous validation across time windows.

Monitoring

- Detects model drift when RMSE ↑ > 15%.
- Triggers auto-retraining with updated data.

Feedback Loop

- Stores logs and metrics for full traceability.
- Ensures continuous improvement and model stability.



The feedback loop ensures the model stays accurate over time by detecting performance drift, retraining automatically, and keeping complete logs for transparency and reproducibility.

Technical Stack & Clean ML Philosophy

Tool / Library	Purpose	Why Used
Python	Base language	Simple and standard in ML
pandas	Data handling	Easy cleaning and filtering
NumPy	Math operations	Fast numerical processing
scikit-learn	Classical models	Great for education
PyTorch	Deep learning	Flexible (MLP, LSTM)
matplotlib	Visualization	Plot results and metrics
holidays (opt.)	Calendar features	Add day/holiday info

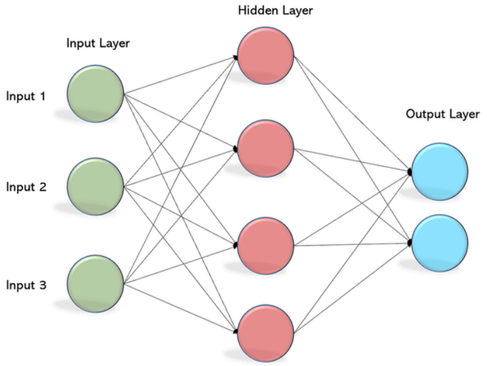
Clean ML Layers & Repository Structure

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`

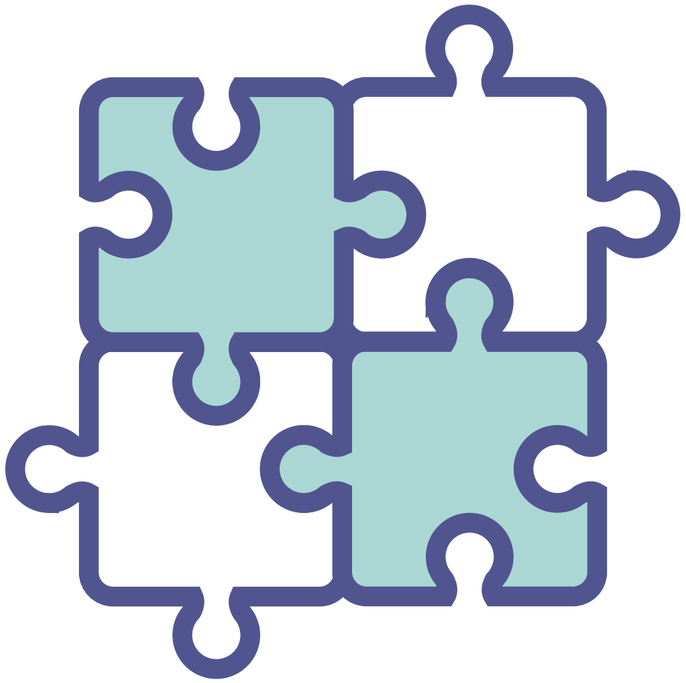
MODELING STRATEGY (MLP VS LSTM)

Comparation

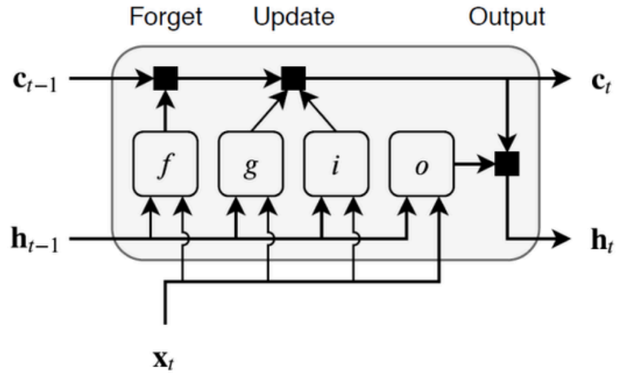
MLP



- Tabular input (features)
- Fast and interpretable
- Good for engineered features



LSTM



- Sequential input (time series)
- Captures temporal dependencies
- Higher computational cost

Conclusions and Q&A

System Results

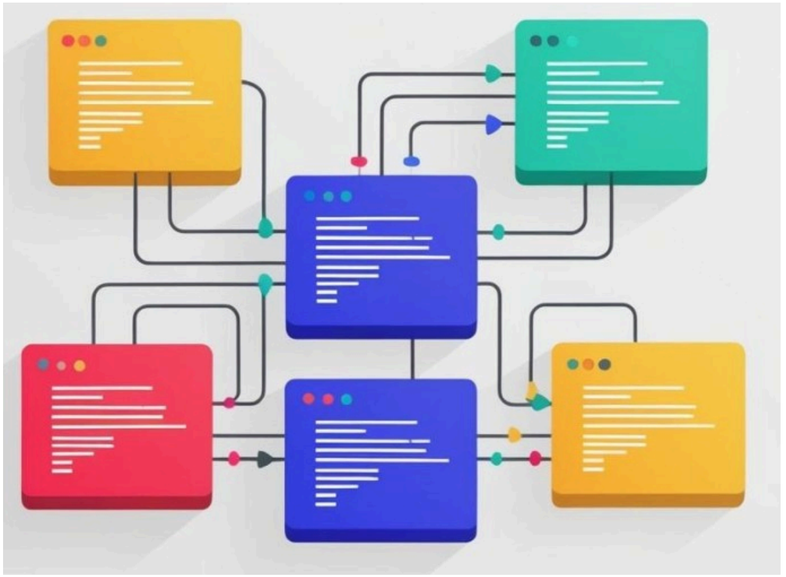
The system produces coherent forecasts per zone and total demand, showing clear visualizations and metrics (RMSE, MAPE). It keeps complete logs for traceability and debugging.

Key Learnings

Data quality proved more critical than model complexity. Proper preprocessing, feature design, and temporal validationsignificantly improved forecast accuracy. MLP vs LSTM comparison highlighted interpretability vs temporal learning.

Future Improvements

Integrate ensemble models and real-time pipelines.
Enhance hierarchical reconciliation methods.
Develop an interactive educational dashboard for exploration.



Thank you very much!